POST: Progressive Object-Slot Tokenization for Multimodal Large Language Models

Junyu Zhou[1] **Han Li**[1] **Wenrui Dai**[1] **Fan He**[2] **Ziyang Zheng**[1]
Hang Xu[2] **Chenglin Li**[1] **Junni Zou**[1] **Hongkai Xiong**[1]
[1]Shanghai Jiao Tong University [2]Yinwang Intelligent Technology Co. Ltd.
{blabla, qingshi9974, daiwenrui, zhengziyang, lcl1985,
zoujunni, xionghongkai}@sjtu.edu.cn
317356583@qq.com, xbjxh@live.com
Corresponding author: Wenrui Dai.

Abstract

Multimodal large language models (MLLMs) leverage object-agnostic tokenization by default, which represents images as flat grids of patch tokens and uniformly discretizes semantically rich foreground regions and information-sparse background areas. This introduces substantial redundancy and constitutes a major source of inference cost. Existing object-level tokenizers could either discard intra-region saliency with predefined region grouping and coarse pooling or limit to a single semantic scale by employing slot attention to the final encoder feature map only. In this paper, we propose a two-stage framework named POST that decouples unsupervised object-centric grouping from language-grounded multimodal reasoning to achieve progressively semantic-aligned object-level visual tokenizers. Specifically, we develop Progressive Slot Attention (PSA) in Stage I to introduce slot attention across multiple layers of a frozen ViT encoder and propagate slot states through a cross-layer refinement chain for learning hierarchical slot–patch correspondences without segmentation supervision. Subsequently, we transfer the learned PSA to the LLaVA pipeline in Stage II for object-level visual tokenization. We design slot-weighted merging to aggregate ViT patch tokens into object-grounded visual tokens and remove low-support slots with adaptive slot pruning. Remarkably, POST eliminates the need for external segmentation models, enables progressive object-centric refinement, and adapts the visual token budget to image complexity in visual tokenization. Experimental results show that PSA achieves strong unsupervised object-centric segmentation on PASCAL VOC and COCO. Using approximately 8.3% visual-token budget of the full-token LLaVA-1.5, POST achieves a favorable accuracy–efficiency trade-off and improves referring expression comprehension on RefCOCO/+/g. It also outperforms recent patch-level and object-level token compression baselines at comparable budgets.

1 Introduction

Multimodal large language models (MLLMs) [1, 2] have emerged as a dominant framework for vision–language reasoning, where visual features from a pretrained encoder are projected into the embedding space of a large language model (LLM). Most MLLMs adopt image–text contrastively pretrained Vision Transformers [3, 4] as visual encoders, yielding a fixed grid of patch tokens that is concatenated with the text prompt at each LLM forward pass.

This *object-agnostic* tokenization does not distinguish patches covering semantically rich foreground and information-sparse background, and causes redundancy by encoding them with identical budgets despite evidently different task-relevant contents. This redundancy dominates the inference cost of

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

MLLM, due to the quadratic complexity of self-attention computation. Object-centric representation offers a principled paradigm to simultaneously achieve improved efficiency and stronger semantic grounding by relating each token to a coherent visual entity rather than an arbitrary patch.

Existing visual token compression methods broadly fall into two categories. *Patch-level pruning and merging* [5–11] ranks or aggregates tokens on the flat patch grid using attention- or similarity-based criteria. Despite being effective at moderate compression ratios, these methods neglect explicit object structures and may discard fine-grained foreground evidence under aggressive reduction. *Object-level tokenization* aligns visual tokens with semantic entities, but still suffers severe limitations. Segmentation- and clustering-based methods [12–14] obtain object regions from external segmenters or feature-space clustering, and usually attenuate salient intra-region evidence by compressing each region through equally-weighted pooling. Slot-Attention-based methods [15, 16] learn soft region assignments to avoid external segmenters, but typically restrict object–slot binding to a single routing stage by applying slot competition only to the final encoder feature map. Moreover, they are trained with indirect object-level supervision (*e.g.*, relying on patch reconstruction or generation-oriented tokenization) rather than explicit object separation. In parallel, unsupervised object-centric learning [17–20] produces strong slot–patch priors, but is not well explored in MLLM tokenization.

To address these limitations, we propose POST, a two-stage framework for progressively semantically aligned object-level visual tokenization in MLLMs. POST eliminates the need for external segmentation models, enables progressive object-centric refinement rather than post-hoc single-layer routing, and adapts visual token budget to image complexity. In **Stage I**, we develop PSA that introduces slot attention across the last p blocks of a frozen ViT encoder and propagates refined slots through a cross-layer refinement chain. Different from single-layer slot competition, PSA enables progressive patch-to-object routing such that lower-level grouping cues are gradually consolidated into object-level semantics. PSA learns hierarchical slot–patch correspondences with diffusion-based reconstruction and slot–image contrastive alignment without requiring segmentation supervision. In **Stage II**, the learned progressive slot–patch attention is transferred to LLaVA-1.5 for object-level visual tokenization. We produce object-level visual tokens by aggregating ViT patch tokens with slot-weighted merging, and achieve an image-adaptive token set by removing low-support slots with adaptive slot pruning.

The contributions of this paper are summarized below.

- We develop Progressive Slot Attention (PSA) that integrates slot attention into multiple layers of a frozen ViT and progressively refines slots with cross-layer object-centric routing. Contrary to single-layer slot competition, PSA enables hierarchical slot–patch correspondences that evolve from local grouping cues to object-level semantics without requiring segmentation labels.
- We propose the Progressive Object-Slot Tokenizer (POST) for MLLMs by transferring the slot–patch attention learned by PSA to guide object-level visual tokenization. POST aggregates ViT patch features with slot-weighted merging and adaptively prunes low-support slots to replace dense patch grids with compact object-level visual tokens.
- PSA achieves strong unsupervised object-centric segmentation on PASCAL VOC and COCO. When transferred to MLLMs, POST retains approximately 8.3% visual token budget of the LLaVA-1.5, while improving referring expression comprehension. It outperforms patch-level compression baselines, and remains competitive with existing object-level tokenizers under comparable token budgets.

2 Related Work

2.1 Unsupervised Object-Centric Learning

Unsupervised object-centric learning (OCL) decomposes visual scenes into entity-level representations without manual annotation. Slot attention [21] provides a standard formulation, where a fixed set of slots competes for input features through slot-wise normalized attention, yielding a permutation-equivariant set of object summaries. Subsequent methods improve scalability to natural images through stronger decoders and representations. SLATE [22] and STEVE [23] use autoregressive Transformer decoders for sharper reconstruction, while DINOSAUR [17] and SPOT [18] reconstruct self-supervised DINO/DINOv2 [24] features instead of pixels. Diffusion-based methods such as

SlotDiffusion [19], LSD [25], GLASS [26], SlotAdapt [27], and CODA [20] use slot-conditioned diffusion decoders, including adaptations of pretrained Stable Diffusion [28]. CODA further introduces register slots and contrastive alignment. Another line of work [29, 30] adapts the number of slots per image through dynamic allocation or merging.

Despite these advances, existing OCL methods mainly use slots for generation or segmentation, leaving their transferability to multimodal reasoning largely underexplored. Moreover, Slot Attention is typically attached to a single encoder layer, forcing multi-scale visual cues into one routing stage and missing the progressive grouping structure distributed across encoder depth.

2.2 Object-Level Visual Tokenizers for Multimodal LLMs

LLaVA-style MLLMs [1, 2] represent images as uniform grids of CLIP [4] patch tokens, where semantically distinct regions are encoded with the same patch-wise granularity and subsequently passed to the LLM. This design increases the visual context length and motivates visual token compression. Related work spans patch-level pruning and merging, learned visual resampling, and object-level tokenization.

Patch-level pruning and merging operate directly on the patch sequence. ToMe [5] merges similar ViT tokens, while FastV [6], SparseVLM [7], and PyramidDrop [8] prune tokens with text-conditioned attention. VisionZip [9] and LLaVA-PruMerge [10] combine attention-guided token selection with feature aggregation, while HiPrune [11] selects tokens using hierarchical encoder attention. However, these methods operate on flat patch grids, and might remove fine-grained foreground evidence that is important for multimodal reasoning with aggressive compression.

Learned visual resampling aggregates dense visual features into a compact token set. BLIP-2's Q-Former [31] and Flamingo's Perceiver Resampler [32] use learned queries, while TokenLearner [33] learns spatial attention maps for aggregation. These methods typically use fixed token budgets without explicitly enforcing object-centric grouping.

Object- and semantic-level tokenization instead aggregates patches into entity-aligned tokens. EPOC [12] and AdaTok [14] use large pretrained segmentation models, such as SegFormer [34] and SAM [35], to derive object-level regions, introducing substantial encoder-side cost. Their equal-weight region pooling may further suppress salient intra-region evidence. SeTok [13] avoids external segmenters by clustering patch embeddings, but its grouping is mainly determined by feature-space proximity. More closely related to our work, OC-VTP [15] and Slot-MLLM [16] employ Slot Attention for object-level tokenization, but apply slot competition only to the final encoder feature map. This single-step routing limits object–slot separability, while their objectives provide limited object-centric supervision: OC-VTP emphasizes patch reconstruction and coverage, whereas Slot-MLLM couples slots with a tokenizer optimized for image generation.

These limitations motivate POST to learn progressive slot–patch correspondences through unsupervised object-centric pretraining and transfer them to LLaVA-style MLLMs for object-level visual tokenization. Unlike segmentation- or clustering-based tokenizers, POST does not rely on external region proposals or equal-weight region pooling. Instead, it learns soft slot–patch attention that provides intra-region saliency weights for token aggregation. Unlike prior slot-based tokenizers, POST distributes Slot Attention across multiple encoder layers and refines slots progressively, producing more stable and semantically faithful object-level visual tokens without segmentation supervision.

3 Method

We present POST, a two-stage framework for object-level visual tokenization in MLLMs. Standard visual tokens are arranged as flat patch grids, which mix semantically distinct regions and introduce substantial redundancy for the LLM. Rather than applying post-hoc token reduction, POST learns object-centric slot–patch correspondences and uses them to construct compact object-level visual tokens. In Stage I, PSA distributes Slot Attention across multiple encoder layers and progressively refines slots, producing stable object-aligned slot–patch attention. In Stage II, the learned attention is transferred to the MLLM tokenizer, where it guides slot-weighted token aggregation and adaptive slot pruning. This replaces single-layer post-hoc slot routing with progressive object-centric refinement, yielding image-adaptive token budgets without external segmentation models. We review Slot

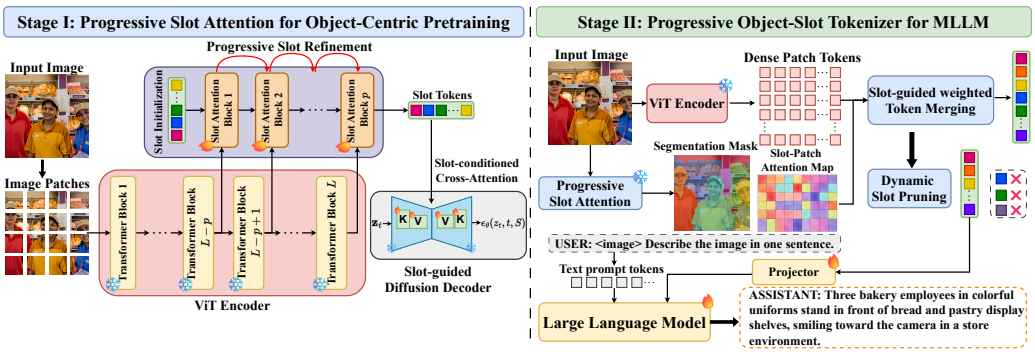

Figure 1: Overview of POST. **Stage I** (PSA) inserts Progressive Slot Attention into multiple layers of a frozen DINOv2 encoder and trains it with a diffusion-based reconstruction objective, learning transferable hierarchical slot–patch correspondences. **Stage II** transfers the learned progressive slot attention into the LLaVA pipeline, where slot-guided token merging and adaptive slot pruning compress CLIP patch tokens into a compact set of object-grounded visual tokens for multimodal reasoning.

Attention and its limitations in Section 3.1 and elaborate Stage I in Section 3.2 and Stage II in Section 3.3. Figure 1 illustrates the full pipeline.

3.1 Preliminary: Slot Attention

Given ViT patch features $\mathbf{X}=\{\mathbf{x}_i\}_{i=1}^{N}\subset\mathbb{R}^d$, Slot Attention [21] decomposes an image into K latent slots $\mathbf{S}=\{\mathbf{s}_k\}_{k=1}^{K}\subset\mathbb{R}^{d_s}$, each representing a corresponding object-centric region. At each iteration, slots are mapped to queries and patches to keys/values, and attention scores are normalised *across slots* to induce object-wise competition:

$$a_{ki} = \mathrm{softmax}_k\left(\frac{q(\mathbf{s}_k)^\top k(\mathbf{x}_i)}{\sqrt{d_s}}\right). \tag{1}$$

Each slot then aggregates its assigned patches using within-slot normalized weights, and a GRU [36] updates the slot state from the previous state and the aggregated value.

$$\bar{a}_{ki} = \frac{a_{ki}}{\sum_{i'=1}^{N} a_{ki'} + \epsilon}, \qquad \mathbf{s}_k \leftarrow \mathrm{GRU}\left(\mathbf{s}_k, \sum_{i=1}^{N} \bar{a}_{ki}\, v(\mathbf{x}_i)\right). \tag{2}$$

Iterating Eqs. (1)–(2) yields compact, permutation-equivariant object–slot bindings, where inter-slot competition encourages distinct object regions to be captured by different slots. Supervising slots via patch-feature reconstruction further sharpens this binding.

However, vanilla Slot Attention is typically applied to a *single* feature map, requiring one routing stage to reconcile low-level visual cues with high-level object semantics. This makes the resulting decomposition sensitive to the insertion depth. To address this limitation, we introduce *Progressive Slot Attention* (PSA, Section 3.2), which distributes slot routing across successive ViT layers and propagates slot states through cross-layer refinement. By progressively consolidating part-level cues into object-level semantics, PSA yields more stable and coherent object–slot bindings.

3.2 Stage I: PSA for Object-Centric Pretraining

Motivation. To overcome the limitations of single-layer slot routing, PSA distributes Slot Attention across the last p blocks of a frozen ViT encoder and propagates slot states from shallower to deeper layers. This converts slot grouping from a one-shot assignment into a progressive refinement process, producing multi-scale object–slot bindings as a transferable object-aware prior (Figure 2).

Progressive multi-layer Slot Attention. Let f_ψ denote a frozen ViT encoder with L blocks, producing patch features $\mathbf{X}^{(l)}\in\mathbb{R}^{N\times d}$ at layer l. We attach p Slot Attention blocks $\{\Phi_j\}_{j=0}^{p-1}$ to the last p encoder blocks, where Φ_j operates on features from layer $l_j = L-p+j$, with p=3 by default.

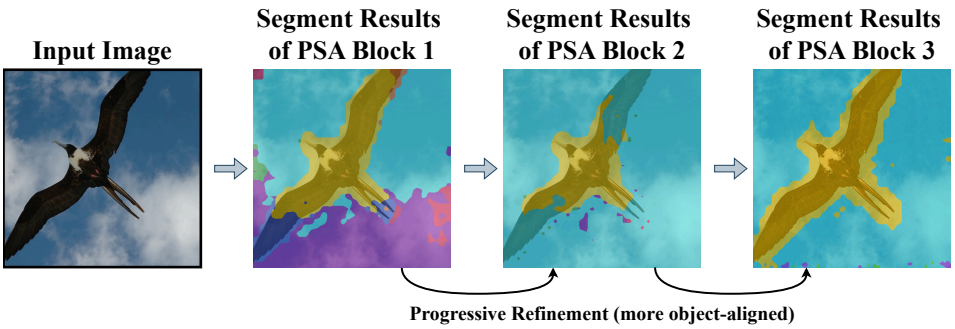

Figure 2: Illustration of Progressive Slot Attention. PSA integrates Slot Attention blocks into the final p layers of a frozen ViT encoder and propagates refined slots across layers. Through cross-layer refinement, the slot–patch attention maps evolve from fragmented part-level responses to coherent object-centric segmentations, producing increasingly object-aligned visual tokens.

(1) Cross-layer slot initialization. At the first Slot Attention block, each slot is sampled from a shared Gaussian distribution:

$$\mathbf{s}_k^{(0)} \sim \mathcal{N}\big(\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2)\big)\,, \qquad \boldsymbol{\mu}, \boldsymbol{\sigma} \in \mathbb{R}^{d_s}, \quad k = 1, \ldots, K, \tag{3}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are shared learnable parameters. The attention and update functions are also shared across slots, making the parameterization independent of K. Subsequent blocks use the preceding block's refined slots as their initial slot states.

(2) Iterative refinement with competitive binding. Each Slot Attention block refines slots for T iterations indexed by t, with T=3 by default. At each iteration, slot-normalized attention induces competitive binding, and patch-normalized weights aggregate a per-slot update:

$$\begin{aligned} \bar{a}_{ki}^{(t)} &= \frac{a_{ki}^{(t)}}{\sum_{j=1}^{N} a_{kj}^{(t)} + \epsilon}, & \mathbf{u}_k^{(t)} &= \sum_{i=1}^{N} \bar{a}_{ki}^{(t)} v\Big(\mathbf{x}_i^{(l)}\Big)\,, \\ \tilde{\mathbf{s}}_k^{(t)} &= \mathrm{GRU}\Big(\mathbf{s}_k^{(t-1)}, \mathbf{u}_k^{(t)}\Big)\,, & \mathbf{s}_k^{(t)} &= \tilde{\mathbf{s}}_k^{(t)} + \mathrm{MLP}\Big(\mathrm{LN}\Big(\tilde{\mathbf{s}}_k^{(t)}\Big)\Big)\,, \end{aligned} \tag{4}$$

where ϵ is a small constant for numerical stability, and $\mathrm{LN}(\cdot)$ denotes Layer Normalization.

(3) Cross-layer propagation. The refined slots from Φ_j are used to initialize Φ_{j+1}, forming a single refinement chain from shallow to deep encoder features. Each block is defined as

$$\mathbf{S}^{(j+1)},\ \mathbf{A}^{(j)} = \Phi_j\Big(\mathbf{X}^{(l_j)},\ \mathbf{S}^{(j)}\Big)\,, \qquad l_j = L - p + j, \quad j = 0, \ldots, p-1. \tag{5}$$

The final slot–patch attention map $\mathbf{A} = \mathbf{A}^{(p-1)} \in [0,1]^{K \times N}$ is used downstream. During Stage I, f_ψ remains frozen and the Slot Attention blocks are trained progressively. We first train Φ_0 with its output slots connected to the decoder, then freeze Φ_0 and train Φ_1 on top of its refined slots. This procedure is repeated until Φ_{p-1} is trained. All blocks are trained within a single progressive schedule; implementation details and total pretraining costs are provided in Appendix A.1.

Reconstruction-based supervision via a diffusion decoder. We supervise the final slots $\mathbf{S}$ with image reconstruction using a pretrained latent diffusion decoder [25, 19, 20]. Following CODA [20], the U-Net cross-attention input is replaced with $[\mathbf{S};\ \mathbf{r}]$, where $\mathbf{r}$ denotes fixed register slots derived from padding-token embeddings of the frozen CLIP text encoder. These register slots serve as semantically neutral attention sinks, absorbing residual background attention without introducing text-side bias. We finetune only the key, value, and output projections in U-Net cross-attention, while keeping the remaining U-Net and VAE frozen.

Let $\mathbf{z}$ be the VAE latent, $\mathbf{z}_\gamma$ its noisy version at log-SNR γ, and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Let ϕ denote the PSA parameters and θ the trainable cross-attention projections. The pretraining objective is

$$\mathcal{L}_{\mathrm{pre}}(\phi, \theta) = \underbrace{\mathbb{E}_{(\mathbf{z},\mathbf{S}),\boldsymbol{\epsilon},\gamma}\Big[\big\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{z}_\gamma, \gamma, \mathbf{S}, \mathbf{r})\big\|_2^2\Big]}_{\mathcal{L}_{\mathrm{dm}}} - \lambda_{\mathrm{cl}} \underbrace{\mathbb{E}_{(\mathbf{z},\tilde{\mathbf{S}}),\boldsymbol{\epsilon},\gamma}\Big[\big\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\bar{\theta}}(\mathbf{z}_\gamma, \gamma, \tilde{\mathbf{S}}, \mathbf{r})\big\|_2^2\Big]}_{\mathcal{L}_{\mathrm{cl}}}, \tag{6}$$

where $\tilde{\mathbf{S}}$ denotes hard-negative slots mixed across images in a mini-batch, and $\bar{\theta}$ is stop-gradient. This objective encourages slots to capture image-specific object structure.

3.3 Stage II: Slot-Guided Visual Tokenization

We integrate the Stage I-pretrained PSA into LLaVA-style MLLMs as an object-aware visual tokenizer. The tokenizer transfers the learned slot–patch correspondence to the MLLM vision pipeline, aggregates patch features into slot-level tokens, and adaptively prunes redundant slots. PSA is frozen during vision-language alignment and fine-tuned during instruction tuning (Appendix A.2).

Integrating PSA into the MLLM vision pipeline. Stage II uses the Stage I-pretrained PSA as an object-level grouping prior for CLIP visual token reduction. The DINOv2 ViT-B/14 and CLIP ViT-L/14 branches use identical 336×336 preprocessing and patch size 14. After removing non-spatial tokens, both yield aligned 24×24 patch grids with matching row-major indices.

For each image, PSA predicts a slot–patch attention map $\mathbf{A} \in [0,1]^{K \times N}$ on the shared patch grid. Instead of feeding PSA slot embeddings to the LLM, we use $\mathbf{A}$ to aggregate CLIP patch tokens, thereby preserving CLIP as the language-aligned visual representation while injecting object-centric grouping weights.

Given CLIP patch tokens $\mathbf{Y} \in \mathbb{R}^{N \times d_c}$, $\mathbf{A}$ is transferred by patch index and aggregates $\mathbf{Y}$ into compact slot-level tokens $\mathbf{T}$ (Eq. (7)) without spatial resampling. These object-level tokens replace the dense patch-token sequence, pass through the standard MLLM projector, and are optimized with the language-modeling objective. This decouples object grouping from semantic visual encoding: the Stage I encoder provides reliable slot–patch correspondences, while CLIP provides the language-aligned features consumed by the LLM. Although this dual-encoder design adds one visual encoding pass, PSA is lightweight and the added encoder cost is modest relative to the LLM latency reduction from token compression (Section 4.4).

Slot-guided weighted merging. Let $\mathbf{Y} \in \mathbb{R}^{N \times d_c}$ denote the CLIP patch tokens fed to the MLLM projector. For each slot k, we use its attention row $\mathbf{a}_k$ in $\mathbf{A}$ as a soft assignment over patches and aggregate tokens with weights normalized over patches:

$$\hat{a}_{ki} = \frac{a_{ki}}{\sum_{j=1}^{N} a_{kj} + \epsilon}, \qquad \mathbf{t}_k = \sum_{i=1}^{N} \hat{a}_{ki}\, \mathbf{y}_i. \tag{7}$$

This yields one object-level token per slot by aggregating information over the full attention support rather than selecting a single patch. Simpler alternatives, such as hard pruning and uniform merging, discard the soft assignment structure and perform worse in our ablations (Section 4.3).

Adaptive slot pruning. A fixed slot budget K does not reflect image-dependent scene complexity, since inactive or background slots often receive only sparse patch support. We therefore estimate slot validity from the support mass of its attention map and retain only slots that cover sufficient patches:

$$m_k = \sum_{i=1}^{N} \mathbb{I}(a_{ki} > \tau_a), \qquad \mathcal{V} = \{k \mid m_k \geq \tau_s\}, \qquad K_{\text{eff}} = |\mathcal{V}|. \tag{8}$$

This yields an image-adaptive token set $\mathbf{T} = \{\mathbf{t}_k\}_{k \in \mathcal{V}} \in \mathbb{R}^{K_{\text{eff}} \times d_c}$, where the effective token count $K_{\text{eff}} \leq K$ increases for complex scenes and decreases with fewer valid object-level regions.

4 Experiments

4.1 Experimental Setup

We evaluate our framework on two tasks: unsupervised object-centric segmentation for Stage I and multimodal understanding for Stage II. We use $4\times$ NVIDIA H100 with 80 GB GPUs for training and a single H100 for inference (with the breakdown of end-to-end efficiency in Appendix B.4). Full implementation details and training schedules are deferred to Appendix A.

Unsupervised object-centric segmentation. We pretrain and evaluate PSA on PASCAL VOC 2012 [37] and COCO 2017 [38] datasets, comparing against feature-reconstruction OCL baselines (DINOSAUR [17], SPOT [18]) and recent diffusion-decoder methods (SlotDiffusion [19], LSD [25], SlotAdapt [27], CODA [20]). Following [18, 20], we report FG-ARI, mBO^i/mBO^c, and mIoU.

Multimodal understanding. For Stage II, we integrate the Stage I-pretrained PSA into LLaVA-1.5-7B with Vicuna-7B, serving as a progressive object-slot visual tokenizer. The training data

Table 1: Unsupervised object-centric segmentation on PASCAL VOC and COCO. We report FG-ARI, mBO^i/mBO^c, and mIoU, grouped by decoder family (MLP, autoregressive, and diffusion). **Bold** and underline denote the best and second-best results, respectively.

Method	PASCAL VOC				COCO			
	FG-ARI↑	mBO^i↑	mBO^c↑	mIoU↑	FG-ARI↑	mBO^i↑	mBO^c↑	mIoU↑
MLP decoders								
SlotAttention [21]	12.3	24.6	24.9	-	21.4	17.2	19.2	-
DINOSAUR [17]	24.6	39.5	40.9	-	40.5	27.7	30.9	-
Autoregressive transformer decoders								
SLATE [22]	15.6	35.9	41.5	-	32.5	29.1	33.6	-
DINOSAUR [17]	24.8	44.0	51.2	-	34.1	31.6	39.7	-
SPOT [18]	19.7	48.3	55.6	46.8	37.8	35.0	**44.7**	33.0
Diffusion decoders								
SlotDiffusion [19]	17.8	50.4	55.3	44.9	37.2	31.0	35.0	31.2
StableLSD [25]	-	32.1	35.4	31.5	-	25.9	30.0	24.7
SlotAdapt [27]	29.6	51.5	51.9	-	41.4	35.1	39.2	36.1
CODA [20]	**32.2**	55.4	61.3	50.8	**47.5**	36.6	41.4	36.4
Ours	26.7	**59.5**	**63.5**	**52.8**	41.7	**36.9**	42.0	**36.8**

and configuration follow the official LLaVA-1.5-7B setup. Baselines include LLaVA-1.5-7B-based token-compression methods from three categories: patch-level pruning (FastV [6], SparseVLM [7], VisionZip [9], PyDrop [8], HiPrune [11]), patch-level merging (VisionZip+ [9], PruMerge [10]), and object-level tokenization (EPOC [12], Slot-MLLM [16], OC-VTP [15], AdaTok [14]). Evaluation is conducted on standard multimodal understanding benchmarks, including MME [39], MMBench [40], GQA [41], POPE [42], VQAv2 [43], MMMU [44], SEED [45], and RefCOCO [46, 47]. We also report the average number of visual tokens fed into the LLM. We reproduce the full-token LLaVA-1.5-7B baseline, EPOC*, and Slot-MLLM* under the same Stage II training and evaluation settings as POST. Other baseline results are taken from the published literature. Implementation details are provided in Appendix A.2.

Table 1 reports Stage I segmentation results. Tables 2 and 3 report Stage II multimodal understanding and referring expression comprehension, respectively.

4.2 Main Results

We present the main results in two parts: unsupervised object-centric segmentation with PSA in Section 4.2.1, and multimodal understanding with POST for object-centric tokenization in Section 4.2.2.

4.2.1 Unsupervised Object-Centric Segmentation

We evaluate Stage I (PSA) on PASCAL VOC and COCO, with results grouped by decoder family in Table 1. Compared with state-of-the-art diffusion-based OCL methods, PSA achieves stronger object-centric segmentation, setting the best results on PASCAL VOC across mBO^i, mBO^c, and mIoU, and obtaining competitive or superior performance on the more challenging COCO benchmark. These results indicate that progressive cross-layer slot refinement improves slot–patch alignment and object-level feature disentanglement without segmentation supervision. Figure 3 further supports the quantitative results. Compared with SPOT, SlotDiffusion, and CODA, PSA produces cleaner object boundaries and more coherent slot assignments, particularly in cluttered multi-object scenes. Additional visualizations are provided in the appendix.

4.2.2 Multimodal Understanding and Referring Expression Comprehension

Tables 2 and 3 report the Stage II results. We evaluate fixed-budget configurations and Ours†, which adaptively determines the number of visual tokens per image through adaptive slot pruning.

Comparison with token-compression baselines. As shown in Table 2, POST outperforms patch-level pruning and merging methods under comparable token budgets, indicating that object-level tokenization is more effective than compression over flat patch grids. Compared with existing

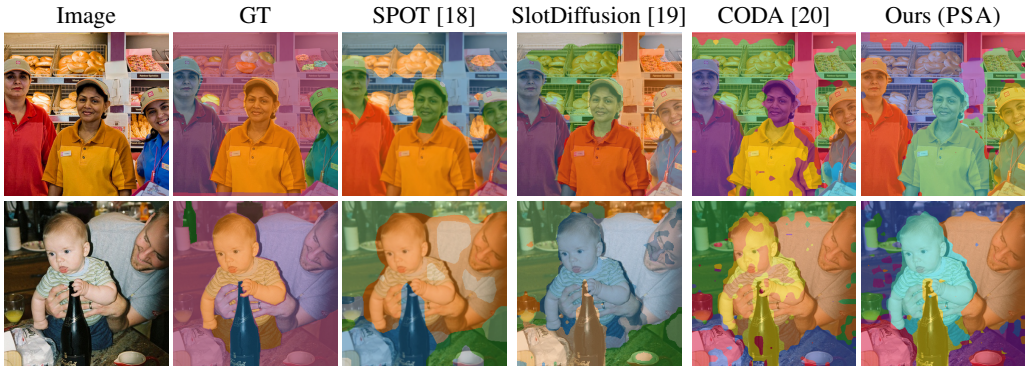

Figure 3: Qualitative comparison of unsupervised object-centric segmentation on COCO (top) and PASCAL VOC (bottom). Columns show the original image, ground-truth mask, and predictions from SPOT, SlotDiffusion, CODA, and Ours (PSA). Additional examples are provided in Appendix C.

Table 2: Comparison of visual tokenization methods on LLaVA-1.5-7B. *Token Number* denotes the average number of visual tokens fed into the LLM. Our 32/64/192-token settings use fixed budgets, while Ours† uses adaptive slot pruning and reports the mean K_{eff} over the seven benchmarks. * denotes reproduction with the official LLaVA training data and scripts. "-" indicates unreported results. **Bold** and underline denote the best and second-best benchmark results among object-level tokenizers. POST, EPOC, Slot-MLLM, and AdaTok use VLM training, and OC-VTP requires no VLM fine-tuning. Patch-level baselines are predominantly training-free. OC-VTP results follow arXiv:2511.20439v1. HiPrune's MMMU and SEED results are taken from the comparison in [15]. Reproduction details are provided in Appendix A.2.

Method	Token Number↓	MME↑	MMB↑	POPE↑	GQA↑	VQAv2↑	MMMU↑	SEED↑
LLaVA-1.5-7B [2]	576	1862	64.3	85.9	62.0	78.5	35.3	58.6
LLaVA-1.5-7B*	576	1819	62.7	85.4	61.2	76.4	35.6	60.2
Patch-level Token Pruning								
FastV [6]	64	1256	48.1	48.0	46.1	55.0	34.0	51.9
SparseVLM [7]	64	1505	56.2	75.1	52.7	68.2	32.7	51.1
VisionZip [9]	64	1690	60.1	77.0	55.1	72.4	36.2	52.2
PyDrop [8]	64	1561	58.8	55.9	47.5	69.2	-	50.4
HiPrune [11]	64	1646	59.5	73.0	53.6	69.2	36.7	55.2
Patch-level Token Merging								
VisionZip+ [9]	64	1756	61.5	80.9	57.0	74.2	-	-
PruMerge [10]	54	1749	60.9	76.3	55.9	72.0	-	-
Object-Level Tokenization								
EPOC* [12]	31.82	1396	42.9	80.4	55.2	64.1	34.1	48.4
Slot-MLLM* [16]	32	1511	49.7	79.8	58.1	68.9	28.0	47.4
OC-VTP [15]	64	1708	59.8	83.0	55.7	73.6	35.6	57.9
AdaTok [14]	53	**1819**	61.3	83.3	58.6	74.2	-	-
Ours	32	1776	63.7	83.6	59.8	73.1	33.8	57.0
Ours	64	1791	63.8	**84.2**	60.0	73.8	35.8	57.3
Ours	192	1813	64.8	83.9	**60.6**	74.0	35.2	**58.1**
Ours†	48.32	1794	**65.4**	84.0	60.1	**74.8**	**36.0**	**58.1**

object-level tokenizers, POST is competitive or superior on most benchmarks. With adaptive slot pruning, Ours† retains an average of 48.32 visual tokens, approximately 8.3% of the full-token budget. Compared with the reproduced full-token baseline, it scores higher on MMBench and MMMU and lower on the other five general multimodal benchmarks. At a matched 32-token budget, POST outperforms Slot-MLLM on all seven benchmarks at an additional latency of 1.92 ms per image (Table 6). The progressive-refinement ablation is reported in Appendix Table 7.

Referring expression comprehension. We further evaluate fine-grained visual grounding on RefCOCO/+/g (Table 3). POST consistently outperforms object-level tokenization baselines across

Table 3: Referring expression comprehension on RefCOCO/+/g. For Ours†, *Token Number* reports the mean effective slot count K_{eff} across the three benchmarks. * indicates reproduction using the official LLaVA training data and scripts. **Bold** denotes the best result in each column.

Method	Token Number↓	RefCOCO↑	RefCOCO+↑	RefCOCOg↑
LLaVA-1.5-7B* [2]	576	28.7	27.7	55.7
EPOC* [12]	31.82	25.7	25.3	48.6
Slot-MLLM* [16]	32	26.9	24.6	49.6
Ours w/ vanilla SlotAttention	64	29.9	29.1	62.3
Ours†	56.71	**31.0**	**29.6**	**64.8**

Table 4: Ablation on the number of progressive Slot Attention blocks p on PASCAL VOC. The baseline uses a single post-hoc Slot Attention block on the final encoder feature map. Encoder time is decomposed into ViT forward time and PSA time. **Bold** denotes the best result in each column.

Method	FG-ARI↑	mBO^{i}↑	mBO^{c}↑	mIoU↑	ViT time (ms)↓	PSA time(ms)↓	All Enc. time (ms)↓
Baseline	**32.2**	55.4	61.3	50.8	**15.02**	**1.22**	**16.27**
PSA (p=2)	25.8	57.6	62.4	51.9	15.05	2.26	17.34
PSA (p=3)	26.7	**59.5**	63.5	**52.8**	15.05	3.26	18.34
PSA (p=4)	28.3	59.2	**63.7**	52.1	15.08	4.24	19.34

Table 5: Ablation on different slot-to-token aggregation operators and adaptive slot pruning in Stage II. **Bold** and underline denote the best and second-best results, respectively.

Hard Pruning	Uniform Mean Merging	Slot-Weighted Merging	Adaptive Slot Pruning	Token Number↓	MME↑	MMB↑	POPE↑	GQA↑
✓				192	1719	62.7	**84.8**	59.6
✓			✓	72.30	1735	63.6	<u>84.6</u>	59.8
	✓			192	1758	64.4	84.4	<u>60.5</u>
	✓		✓	47.38	1769	62.9	84.0	60.4
		✓		192	**1813**	<u>64.8</u>	83.9	**60.6**
		✓	✓	48.32	<u>1794</u>	**65.4**	84.0	60.1

all three benchmarks. Replacing PSA with vanilla Slot Attention (Ours w/ vanilla SlotAttention) leads to consistent performance degradation, confirming the importance of progressive multi-layer refinement for accurate visual grounding.

4.3 Ablation Studies

We present the main ablations on the number of progressive Slot Attention blocks in Stage I and the token aggregation operator in Stage II. Additional ablations are provided in Appendix B.

Number of progressive Slot Attention blocks. Table 4 evaluates different numbers of PSA blocks p on PASCAL VOC, using a single post-hoc Slot Attention block on the final encoder feature map as the baseline. Progressive cross-layer refinement improves region-overlap performance. It yields the best mBO^{i} and mIoU at p=3, while p=4 provides only a marginal mBO^{c} gain and lowers both mBO^{i} and mIoU. Since PSA time remains small relative to the ViT forward pass, PSA improves segmentation quality with limited encoder overhead. We therefore use p=3 by default.

Object-level tokenizer design. Table 5 evaluates the tokenizer design in Stage II. Hard pruning selects the highest-attended CLIP patch for each slot. Mean merging uniformly averages attended patches, whereas slot-weighted merging aggregates them with normalized slot–patch attention. Under the same initial slot number K=192, slot-weighted merging performs best on MME, MMBench, and GQA, whereas hard pruning performs best on POPE. With slot-weighted merging, adaptive pruning reduces the average token count from 192 to 48.32, improving MMBench and POPE while lowering MME and GQA. We adopt slot-weighted merging with adaptive pruning as the default tokenizer.

Table 6: Latency analysis on LLaVA-1.5-7B. *Encoder Time* includes all visual encoding modules, including two ViT forward passes and PSA for POST. *LLM Latency* and *Per Image Time* denote LLM forward time and total wall-clock time per image. The full-token MME score is cited from the original paper, and its reproduced score is reported in Table 2.

Method	MME↑	Token Number↓	Encoder Time/ms↓	LLM Latency/ms↓	Per Image Time/ms ↓	FLOPs↓
LLaVA-1.5-7B [2]	**1862**	576	**6.91**	60.48	70.97	8.670×10^{12}
EPOC [12]	1396	**31.82**	126.03	<u>33.32</u>	168.63	$\mathbf{1.336\times10^{12}}$
Slot-MLLM [16]	1511	<u>32</u>	<u>8.62</u>	34.86	**46.98**	<u>1.337×10^{12}</u>
Ours	1776	<u>32</u>	13.47	**32.53**	<u>48.90</u>	1.340×10^{12}
Ours†	<u>1794</u>	48.32	14.65	35.31	50.71	1.569×10^{12}

4.4 Latency and Efficiency Analysis

Table 6 reports the end-to-end inference cost on LLaVA-1.5-7B. Compared with the original LLaVA-1.5-7B baseline using 576 visual tokens, POST replaces the dense patch sequence with compact object-level tokens, substantially reducing LLM latency, per-image inference time, and FLOPs while maintaining strong multimodal performance. The additional PSA encoding branch introduces moderate encoder overhead, but this cost is outweighed by the reduction in LLM computation. Compared with EPOC, which relies on an external segmentation module, POST achieves much lower end-to-end latency and substantially higher MME. Compared with Slot-MLLM under the same token budget, POST achieves stronger multimodal performance with comparable inference cost, demonstrating a favorable accuracy–efficiency trade-off. Adaptive POST reduces end-to-end latency from 70.97 to 50.71 ms (28.5%) and FLOPs by 81.9%. Additional efficiency evaluation results are provided in Appendix B.4.

5 Conclusion

We presented POST, a two-stage framework for object-level visual tokenization in MLLMs. PSA learns hierarchical slot–patch correspondences through progressive multi-layer Slot Attention on a frozen ViT encoder, without segmentation supervision. POST transfers this grouping prior to a LLaVA-style pipeline via slot-weighted token merging and adaptive slot pruning, replacing dense patch grids with compact object-grounded visual tokens. With approximately 8.3% of the full-token LLaVA-1.5 visual-token budget, POST achieves a favorable accuracy–efficiency trade-off and improves referring expression comprehension. These results show that progressive object-slot visual tokenization provides an effective route toward more efficient and semantically grounded multimodal reasoning.

Acknowledgement

This work was in part supported by the National Natural Science Foundation of China, under Grants 62431017, 62320106003, U24A20251, 62371288, 62125109, 62401357, in part by the National Key Research and Development Program of China under Grant 2025YFF0515600, in part by the Fundamental Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (JYB2025XDXM611), and in part by Yinwang Intelligent Technology Research Project.

References

[1] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, volume 36, pages 34892–34916, 2023.

[2] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26296–26306, 2024.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly,

Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[4] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya 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 *Proceedings of the 38th International Conference on Machine Learning*, pages 8748–8763, 2021.

[5] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In *International Conference on Learning Representations*, 2023.

[6] Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, pages 19–35, 2024.

[7] Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis A Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and Shanghang Zhang. SparseVLM: Visual token sparsification for efficient vision-language model inference. In *Proceedings of the 42nd International Conference on Machine Learning*, pages 74840–74857, 2025.

[8] Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Jiaqi Wang, Feng Wu, and Dahua Lin. Conical visual concentration for efficient large vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14593–14603, 2025.

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

[10] Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. LLaVA-PruMerge: Adaptive token reduction for efficient large multimodal models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22857–22867, 2025.

[11] Jizhihui Liu, Guangdao Zhu, and Feiyi Du. HiPrune: Training-free visual token pruning via hierarchical attention in vision-language models (student abstract). *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(48):41275–41277, 2026.

[12] Delong Chen, Samuel Cahyawijaya, Jianfeng Liu, Baoyuan Wang, and Pascale Fung. Subobject-level image tokenization. In *Proceedings of the 42nd International Conference on Machine Learning*, pages 7719–7738, 2025.

[13] Shengqiong Wu, Hao Fei, Xiangtai Li, Jiayi Ji, Hanwang Zhang, Tat-Seng Chua, and Shuicheng Yan. Towards semantic equivalence of tokenization in multimodal LLM. In *International Conference on Learning Representations*, 2025.

[14] Xinliang Zhang, Lei Zhu, Hangzhou He, Shuang Zeng, Ourui Fu, Jiakui Hu, Zhengjian Yao, and Yanye Lu. AdaTok: Adaptive token compression with object-aware representations for efficient multimodal LLMs. *arXiv preprint arXiv:2511.14169*, 2025.

[15] Guangyuan Li, Rongzhen Zhao, Jinhong Deng, Yanbo Wang, and Joni Pajarinen. Object-centric vision token pruning for vision language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pages 7040–7049, 2026.

[16] Donghwan Chi, Hyomin Kim, Yoonjin Oh, Yongjin Kim, Donghoon Lee, Daejin Jo, Jongmin Kim, Junyeob Baek, Sungjin Ahn, and Sungwoong Kim. Slot-MLLM: Object-centric visual tokenization for multimodal LLM. *arXiv preprint arXiv:2505.17726*, 2025.

[17] Maximilian Seitzer, Max Horn, Andrii Zadaianchuk, Dominik Zietlow, Tianjun Xiao, Carl-Johann Simon-Gabriel, Tong He, Zheng Zhang, Bernhard Schölkopf, Thomas Brox, and Francesco Locatello. Bridging the gap to real-world object-centric learning. In *International Conference on Learning Representations*, 2023.

[18] Ioannis Kakogeorgiou, Spyros Gidaris, Konstantinos Karantzalos, and Nikos Komodakis. SPOT: Self-training with patch-order permutation for object-centric learning with autoregressive Transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22776–22786, 2024.

[19] Ziyi Wu, Jingyu Hu, Wuyue Lu, Igor Gilitschenski, and Animesh Garg. SlotDiffusion: Object-centric generative modeling with diffusion models. In *Advances in Neural Information Processing Systems*, volume 36, pages 50932–50958, 2023.

[20] Bac Nguyen, Yuhta Takida, Naoki Murata, Chieh-Hsin Lai, Toshimitsu Uesaka, Stefano Ermon, and Yuki Mitsufuji. Improved object-centric diffusion learning with registers and contrastive alignment. In *International Conference on Learning Representations*, 2026.

[21] 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, pages 11525–11538, 2020.

[22] Gautam Singh, Fei Deng, and Sungjin Ahn. Illiterate DALL-E learns to compose. In *International Conference on Learning Representations*, 2022.

[23] Gautam Singh, Yi-Fu Wu, and Sungjin Ahn. Simple unsupervised object-centric learning for complex and naturalistic videos. In *Advances in Neural Information Processing Systems*, volume 35, pages 18181–18196, 2022.

[24] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

[25] Jindong Jiang, Fei Deng, Gautam Singh, and Sungjin Ahn. Object-centric slot diffusion. In *Advances in Neural Information Processing Systems*, volume 36, pages 8563–8601, 2023.

[26] Krishnakant Singh, Simone Schaub-Meyer, and Stefan Roth. GLASS: Guided latent slot diffusion for object-centric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28673–28683, 2025.

[27] Adil Kaan Akan and Yücel Yemez. Slot-guided adaptation of pre-trained diffusion models for object-centric learning and compositional generation. In *International Conference on Learning Representations*, 2025.

[28] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[29] Ke Fan, Zechen Bai, Tianjun Xiao, Tong He, Max Horn, Yanwei Fu, Francesco Locatello, and Zheng Zhang. Adaptive slot attention: Object discovery with dynamic slot number. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23062–23071, 2024.

[30] Hongjia Liu, Rongzhen Zhao, Haohan Chen, and Joni Pajarinen. MetaSlot: Break through the fixed number of slots in object-centric learning. In *Advances in Neural Information Processing Systems*, volume 38, pages 67319–67344, 2025.

[31] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning*, pages 19730–19742, 2023.

[32] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro,

Jacob L Menick, Sebastian Borgeaud, Andy Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikołaj Bińkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karén Simonyan. Flamingo: A visual language model for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 35, pages 23716–23736, 2022.

[33] Michael S. Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. TokenLearner: Adaptive space-time tokenization for videos. In *Advances in Neural Information Processing Systems*, volume 34, pages 12786–12797, 2021.

[34] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. SegFormer: Simple and efficient design for semantic segmentation with transformers. In *Advances in Neural Information Processing Systems*, volume 34, pages 12077–12090, 2021.

[35] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollar, and Ross Girshick. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[36] Kyunghyun Cho, Bart Van Merriënboer, Çağlar Gulçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734, 2014.

[37] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[38] 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 *European Conference on Computer Vision*, pages 740–755, 2014.

[39] Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, Yunsheng Wu, Rongrong Ji, Caifeng Shan, and Ran He. MME: A comprehensive evaluation benchmark for multimodal large language models. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

[40] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. MMBench: Is your multi-modal model an all-around player? In *European Conference on Computer Vision*, pages 216–233, 2024.

[41] Drew A Hudson and Christopher D Manning. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6700–6709, 2019.

[42] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 292–305, 2023.

[43] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6904–6913, 2017.

[44] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. MMMU: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9556–9567, 2024.

[45] Bohao Li, Yuying Ge, Yixiao Ge, Guangzhi Wang, Rui Wang, Ruimao Zhang, and Ying Shan. SEED-Bench: Benchmarking multimodal large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13299–13308, 2024.

[46] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. ReferItGame: Referring to objects in photographs of natural scenes. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 787–798, 2014.

[47] Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L. Yuille, and Kevin Murphy. Generation and comprehension of unambiguous object descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11–20, 2016.

[48] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[49] Pooyan Rahmanzadehgervi, Logan Bolton, Mohammad Reza Taesiri, and Anh Totti Nguyen. Vision language models are blind. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, pages 18–34, 2024.

[50] Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai Charitha Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, Austin Wang, Rob Fergus, Yann LeCun, and Saining Xie. Cambrian-1: A fully open, vision-centric exploration of multimodal LLMs. In *Advances in Neural Information Processing Systems*, volume 37, pages 87310–87356, 2024.

A Implementation Details

This section supplements the experimental setup in Section 4.1 with training and implementation details.

A.1 Stage I: Unsupervised Object-Centric Pretraining

PSA attaches $p{=}3$ progressive Slot Attention blocks to the last three blocks of a frozen DINOv2 ViT-B/14 encoder, with $T{=}3$ refinement iterations per block. The slot decoder is based on Stable Diffusion v1.5. The VAE and U-Net are kept frozen, except for the key, value, and output projections in the U-Net cross-attention layers, which are optimized together with the Slot Attention blocks $\{\Phi_j\}_{j=0}^{p-1}$ and the shared Gaussian parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. Register slots $\mathbf{r}$ are obtained from padding-token embeddings produced by the frozen CLIP text encoder.

We train PSA separately on COCO 2017 and PASCAL VOC 2012, with dataset-specific slot numbers of $K{=}7$ for COCO and $K{=}6$ for VOC. The contrastive coefficient in Eq. (6) is set to $\lambda_{\text{cl}}{=}0.1$. Training uses AdamW with mixed precision on $4\times$ NVIDIA H100 80 GB GPUs. To stabilize progressive refinement, we adopt a layer-wise freezing schedule. We train the shallowest Slot Attention block first, and then activate deeper blocks at 50K-step intervals, freezing each preceding block before training the next. The complete progressive schedule spans 500K steps on COCO and 250K steps on VOC.

Stage I pretraining cost. Stage I pretraining uses $4\times$ NVIDIA H100 80 GB GPUs and requires approximately 85 GPU-hours on COCO and 42 GPU-hours on VOC. These costs include the complete progressive training of all PSA blocks. The Stable Diffusion decoder, VAE, and contrastive hard-negative branch are used only for pretraining supervision and are discarded in Stage II, so MLLM training and inference incur no diffusion-related computation.

A.2 Stage II Multimodal Training Details

Stage II follows the LLaVA-1.5-7B architecture, comprising a frozen CLIP ViT-L/14 vision encoder, an MLP multimodal projector, and a Vicuna-7B language model. The slot-guided tokenizer is inserted between the vision encoder and the projector, with the CLIP encoder kept frozen throughout Stage II. Training follows the standard two-phase LLaVA-1.5 protocol. Adaptive slot pruning is enabled in both phases with an initial slot count of $K{=}192$. The attention, update, and projection functions in PSA are shared across slots, making the learned routing function independent of K. Increasing K therefore provides a denser set of candidate slots before pruning, without requiring additional PSA pretraining. Stage I uses $K{=}7$ on COCO and $K{=}6$ on VOC. Stage II samples $K{=}192$ candidates from the same shared Gaussian distribution, without copying, interpolating, or expanding slot-specific parameters. During vision-language alignment, PSA remains frozen and only the MLP projector is trained on 558K image-caption pairs. During instruction tuning, PSA is unfrozen and jointly fine-tuned with the MLP projector and Vicuna-7B on 665K instruction-following pairs. Adaptive slot pruning uses $\tau_a{=}10^{-3}$ and $\tau_s{=}1$, calibrated on a held-out validation split and fixed across all Stage II experiments. Both phases use AdamW, mixed-precision training, and DeepSpeed ZeRO-2 on $4\times$ NVIDIA H100 80 GB GPUs, with learning-rate schedules and batch configurations following the public LLaVA-1.5-7B recipe [2].

Baseline comparison protocols. EPOC* and Slot-MLLM* are reproduced from public implementations using the same LLaVA-1.5-7B training data, preprocessing, and lmms-eval v0.6 protocol as POST. Other baseline results are taken from the published literature. We report published AdaTok results, since there is no available public implementation. AdaTok requires VLM training, whereas OC-VTP requires no VLM fine-tuning. Patch-level methods are predominantly training-free and serve as complementary deployment baselines.

B Additional Ablation Results

We provide additional ablations and analyses of PSA and POST.

Progressive segmentation across PSA depth. To assess cross-layer refinement, we evaluate the intermediate outputs after each PSA block in a three-block stack ($p{=}3$) on PASCAL VOC

Table 7: Unsupervised segmentation quality of PSA on PASCAL VOC. "After n-th PSA block" denotes the intermediate output after the n-th block in the refinement chain (p=3). Progressive frozen training yields steadily improved overlap-based metrics across PSA depth.

Variant / Output	FG-ARI↑	mBO^i↑	mBO^c↑	mIoU↑
PSA w/o progressive frozen training				
After 1st PSA block	35.9	23.7	26.8	22.1
After 2nd PSA block	33.4	23.8	27.0	22.1
After 3rd PSA block	16.9	51.2	59.4	49.2
PSA w/ progressive frozen training (**Ours**)				
After 1st PSA block	26.0	55.9	59.6	51.0
After 2nd PSA block	26.8	58.5	63.0	51.8
After 3rd PSA block	26.7	**59.5**	**63.5**	**52.8**

(Table 7). With layer-wise freezing, region-overlap metrics improve steadily along the refinement chain, increasing from 55.9 to 59.5 on mBO^i, from 59.6 to 63.5 on mBO^c, and from 51.0 to 52.8 on mIoU, while FG-ARI remains stable. Without layer-wise freezing, intermediate outputs have poor region overlap, and the final output scores lower on all four metrics than its progressively trained counterpart. These results support layer-wise freezing for stable cross-layer refinement.

Direct PSA slot tokens vs. PSA-guided CLIP token merging. We further ablate how PSA should be integrated into Stage II. We compare directly using PSA slot embeddings as visual tokens with using PSA attention to merge CLIP patch tokens. Both settings use the same PSA attention map and adaptive slot pruning, yielding the same effective token count. As shown in Table 8, PSA-guided CLIP token merging achieves stronger performance, especially on MME, MMBench, and GQA. This indicates that PSA is better used as an object-level grouping prior over the language-aligned CLIP feature space, rather than as a direct replacement for CLIP visual tokens.

Table 8: Ablation on the Stage II integration strategy. Directly using PSA slot embeddings yields weaker multimodal performance, whereas PSA-guided CLIP token merging preserves language-aligned visual semantics. Both settings use the same PSA attention map and adaptive slot pruning, yielding the same effective token count.

Token Source	Token Number↓	MME↑	MMB↑	POPE↑	GQA↑
PSA slot embeddings	48.32	1651	60.2	82.5	57.5
PSA-guided CLIP token merging (**Ours**†)	48.32	**1794**	**65.4**	**84.0**	**60.1**

B.1 COCO Image-Overlap Analysis

COCO 2017 images used for Stage I pretraining may also appear in COCO-derived benchmarks such as VQAv2 and RefCOCO/+/g. Stage I uses no VQA answers, referring expressions, bounding boxes, or segmentation labels. To assess the effect of Stage I image exposure, we pretrain PSA on VOC while keeping Stage II data, optimization, preprocessing, and evaluation fixed. The VOC variant retains 43.72 visual tokens on average and improves RefCOCO/+/g over the full-token baseline by 5.3, 3.8, and 3.6 points (Table 9). These results indicate that the grounding gains are not solely attributable to COCO image exposure during Stage I. COCO-derived Stage II data remain unchanged in this control.

Table 9: COCO image-overlap analysis. COCO- and VOC-pretrained PSA are compared under identical Stage II training and evaluation protocols. Token counts summarize the general multimodal evaluation; grounding-specific averages can differ.

Method	Tokens↓	MME↑	MMB↑	POPE↑	GQA↑	VQAv2↑	MMMU↑	SEED↑
Full-token baseline	576	1819	62.7	85.4	61.2	76.4	35.6	60.2
POST, COCO-pretrained PSA	48.32	1794	65.4	84.0	60.1	74.8	36.0	58.1
POST, VOC-pretrained PSA	43.72	1783	64.7	82.2	60.2	73.9	37.2	57.4

Method	RefCOCO↑	RefCOCO+↑	RefCOCOg↑
Full-token baseline	28.7	27.7	55.7
POST, COCO-pretrained PSA	31.0	29.6	64.8
POST, VOC-pretrained PSA	34.0	31.5	59.3

B.2 LLM Backbone Transfer and Fine-Grained Perception

We evaluate LLM backbone transfer by replacing Vicuna-7B with Qwen3-8B and Qwen3-14B [48], keeping the visual encoders fixed and initializing PSA from the same Stage I checkpoint. For each backbone, POST and its full-token baseline use matched Stage II training and evaluation protocols (Table 10). Across all three backbones, POST reduces visual tokens by 91.6%–92.1% and improves RefCOCO/+/g performance, demonstrating that its compression and grounding benefits extend beyond Vicuna-7B. POST maintains broadly comparable general multimodal performance under substantial token compression, with accuracy trade-offs on some benchmarks. Fine-grained perception is largely preserved, with VLMs Are Blind [49] and CV-Bench [50] scores remaining within 2.1 points of the corresponding full-token baselines across all three backbones.

Table 10: LLM backbone transfer and fine-grained perception. POST and the full-token baseline use matched Stage II training and evaluation protocols for each backbone. The visual encoders are fixed, and PSA is initialized from the same Stage I checkpoint across Vicuna-7B, Qwen3-8B, and Qwen3-14B.

Method	Backbone	Tokens↓	MME↑	MMB↑	POPE↑	GQA↑	VQAv2↑	MMMU↑	SEED↑
Full-token	Vicuna-7B	576	1819	62.7	85.4	61.2	76.4	35.6	60.2
POST	Vicuna-7B	48.32	1794	65.4	84.0	60.1	74.8	36.0	58.1
Full-token	Qwen3-8B	576	1799	72.2	86.7	61.1	76.1	44.6	63.6
POST	Qwen3-8B	47.22	1805	71.8	84.9	60.7	72.4	45.2	61.8
Full-token	Qwen3-14B	576	1795	73.6	86.1	62.4	76.7	47.0	63.9
POST	Qwen3-14B	45.32	1791	72.1	84.6	61.5	74.2	45.9	62.2

Method	Backbone	RefCOCO↑	RefCOCO+↑	RefCOCOg↑	VLMs Are Blind↑	CV-Bench↑
Full-token	Vicuna-7B	28.7	27.7	55.7	31.8	57.2
POST	Vicuna-7B	31.0	29.6	64.8	30.3	55.5
Full-token	Qwen3-8B	30.7	29.9	60.6	33.3	63.7
POST	Qwen3-8B	38.1	34.9	63.0	33.3	62.3
Full-token	Qwen3-14B	30.3	28.7	61.2	34.7	67.6
POST	Qwen3-14B	36.2	33.0	63.6	34.6	65.5

B.3 Alternative Decoder Supervision

We compare MLP, autoregressive Transformer, and diffusion decoders under matched backbone, dataset, slot count, and optimization settings (Table 11). PSA improves mBO^i, mBO^c, and mIoU over vanilla Slot Attention with all three decoders, supporting its effectiveness across decoder families. The effect on FG-ARI varies by decoder: it improves with MLP and Transformer supervision but decreases with diffusion supervision. Diffusion achieves the highest overlap scores, while MLP and Transformer decoders offer lower forward latency. Relative to vanilla Slot Attention, PSA adds at most 0.97 ms of encoder latency and 1.39 ms of total latency per image.

Table 11: Stage I decoder comparison on PASCAL VOC. Forward latency is measured on one NVIDIA H100 80 GB GPU at batch size 1 and 336×336 input resolution. Encoder time includes the frozen ViT and slot-attention modules; total time additionally includes the reconstruction decoder. Diffusion timing includes VAE encoding and one U-Net denoising step, without iterative sampling.

Encoder	Decoder	mBO^i	mBO^c	mIoU	FG-ARI	Encoder Time/ms	Total Time/ms
Vanilla Slot Attention	MLP	39.7	41.1	30.1	26.5	9.71	11.81
PSA (Ours)	MLP	42.1	46.9	33.5	28.3	10.65 (+0.94)	12.77 (+0.96)
Vanilla Slot Attention	Transformer	45.2	52.3	43.7	17.8	9.68	12.08
PSA (Ours)	Transformer	49.7	57.3	47.9	20.2	10.65 (+0.97)	13.07 (+0.99)
Vanilla Slot Attention	Diffusion	55.4	61.3	50.8	32.2	9.75	31.91
PSA (Ours)	Diffusion	59.5	63.5	52.8	26.7	10.66 (+0.91)	33.30 (+1.39)

B.4 End-to-End Efficiency Breakdown

Table 12 reports inference efficiency on one NVIDIA H100 80 GB GPU at batch size 1 and 336×336 resolution. DINOv2 accounts for most of POST's added visual-encoding cost; the three PSA blocks contribute only 2.17 ms in total. The 7.74 ms encoding overhead is outweighed by a 25.17 ms reduction in LLM latency, yielding lower end-to-end latency. Token compression also reduces activation memory and increases throughput relative to the full-token baseline.

Table 12: End-to-end efficiency on LLaVA-1.5-7B. All measurements use one H100 80 GB GPU, batch size 1, and 336×336 inputs.

Metric	Full-token LLaVA-1.5-7B	POST (Ours†)
Visual tokens	576	48.32
Visual encoding/ms	6.91	14.65
CLIP/ms	6.91	6.27
DINOv2/ms	–	6.21
PSA blocks/ms	–	2.17
LLM latency/ms	60.48	35.31
End-to-end latency/ms	70.97	50.71
Throughput/img/s	14.09	19.72
Activation memory/MiB	601.04	37.04
FLOPs	8.670×10^{12}	1.569×10^{12}

C Additional Qualitative Results

We provide two qualitative analyses. Figures 4 and 5 compare unsupervised object-centric segmentation on COCO and PASCAL VOC against SPOT, SlotDiffusion, and CODA. Across both segmentation datasets, PSA produces cleaner object boundaries and more consistent instance separation, particularly in cluttered multi-object scenes. Figure 6 visualizes the object-level tokenization results of POST during MLLM inference, compared with ToMe and Slot-MLLM under the same visual inputs. POST yields more object-consistent token groupings than ToMe and Slot-MLLM, preserving complete foreground entities while suppressing fragmented or background-dominated tokens.

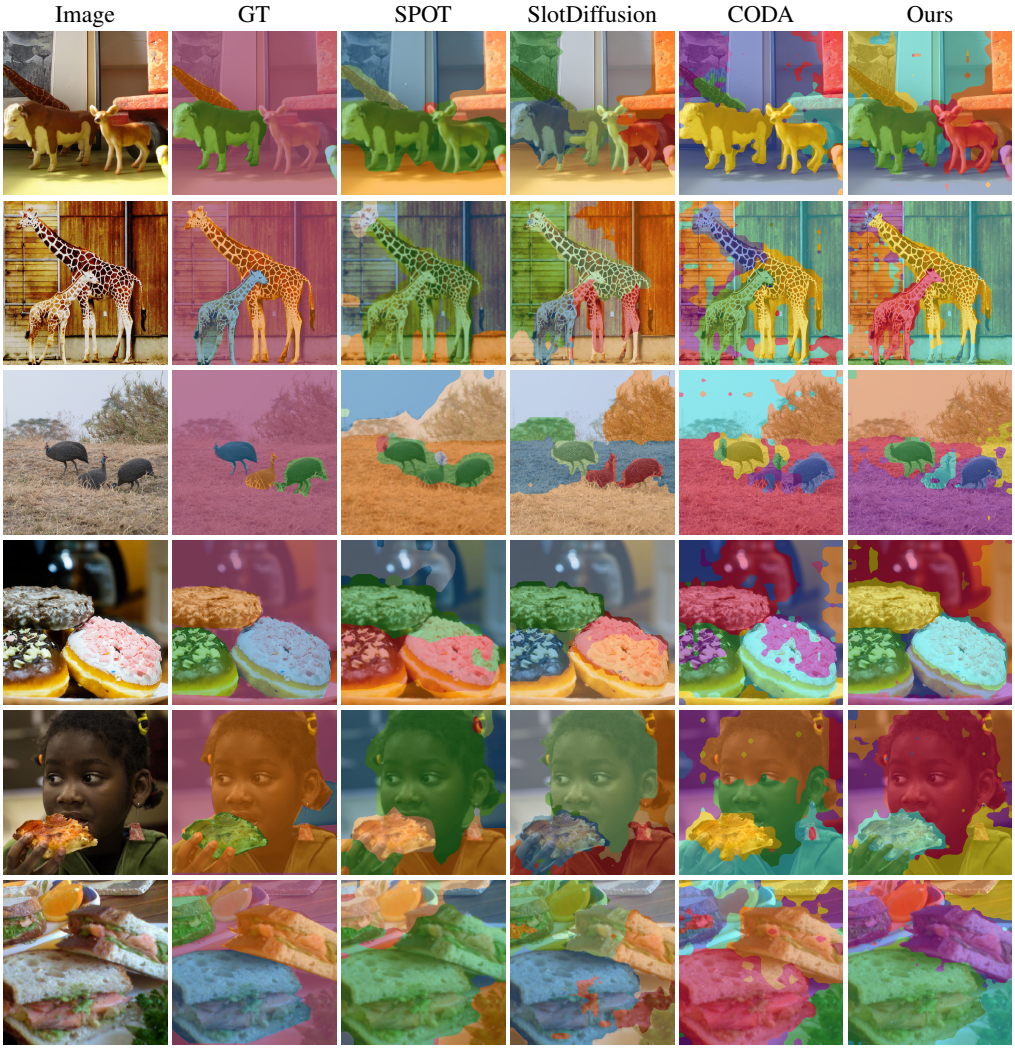

Figure 4: Additional qualitative segmentation results on COCO. Columns show the input image, ground-truth mask, and predictions from SPOT, SlotDiffusion, CODA, and PSA (Ours).



Figure 5: Additional qualitative segmentation results on PASCAL VOC. Columns show the input image, ground-truth mask, and predictions from SPOT, SlotDiffusion, CODA, and PSA (Ours).



Figure 6: Qualitative comparison of token-merging behaviour. Each group of four columns shows, from left to right, the input image and the merged token maps produced by ToMe, Slot-MLLM, and POST (Ours).

D Limitations and Future Directions

POST uses a dual-encoder design in Stage II: DINOv2 with PSA provides object-level grouping, while CLIP supplies language-aligned visual features. This design adds visual-encoding cost and implementation complexity. In our evaluated setting, however, the reduction in LLM latency outweighs the additional encoding overhead, yielding lower end-to-end latency (Appendix B.4).

Token compression reduces accuracy on some multimodal benchmarks, reflecting an accuracy–efficiency trade-off. Although our transfer experiments cover Vicuna-7B and Qwen3-8B/14B, they use the same DINOv2–CLIP configuration. Extending POST to other visual resolutions, patch layouts, or dynamic grids requires encoder-specific Stage I PSA pretraining or a learnable grid-alignment module.

Diffusion-based Stage I pretraining introduces additional computational cost. MLP and Transformer decoders offer lower forward latency but achieve lower overlap scores in our experiments (Appendix B.3), highlighting a trade-off between decoder efficiency and object-grouping quality.

Future work will explore single-encoder architectures that unify object-centric grouping and language-aligned visual representations, as well as resolution-adaptive tokenization for broader visual-encoder compatibility. We will also investigate objectives that more closely couple slot formation with instruction following and reasoning to improve the accuracy–efficiency trade-off, and evaluate POST on larger and more recent MLLM families.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract clearly states the following claims about the paper's contributions and scope.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We have provided a separate "Limitations" section in the Appendix.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

Justification: This paper does not include theoretical results.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The detailed experimental settings and information are provided in the Appendix, and this information is sufficient to reproduce the main experimental results.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: All the code will be made public after acceptance.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: All the experimental settings are clarified in the Appendix.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: Following previous studies we do not report error bars.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The computing requirements are provided in the Appendix.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We ensure our research adheres to the guidelines.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: The tasks we tackling does not have apparent societal impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: This paper poses no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All the assets used are properly credited and are the license and terms of use explicitly mentioned and properly respected.

    Guidelines:

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

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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

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

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [Yes]

Justification: The method is evaluated within an existing LLaVA-1.5-7B MLLM pipeline, as described in the experimental setup. No LLM is used to generate training data, labels, or evaluation results.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.