

QASA: Quality-Guided K-Adaptive Slot Attention for Unsupervised Object-Centric Learning

Tianran Ouyang¹, Xingping Dong^{1*}, Jing Zhang², Mang Ye¹, Kaihao Zhang²,
and Bo Du¹

¹ School of Computer Science, National Engineering Research Center for Multimedia Software, Institute of Artificial Intelligence, Hubei Key Laboratory of Multimedia and Network Communication Engineering, Wuhan University, Wuhan, China

² Australian National University, Canberra, Australia
{ouyangtianran, xingpingdong, yemang, dubo}@whu.edu.cn
jing.zhang@anu.edu.au, super.khzhang@gmail.com

Abstract. Slot Attention, an approach that binds different objects in a scene to a set of "slots", has become a leading method in unsupervised object-centric learning. Most methods assume a fixed slot count K , and to better accommodate the dynamic nature of object cardinality, a few works have explored K -adaptive variants. However, existing K -adaptive methods still suffer from two limitations. First, they do not explicitly constrain slot-binding quality, so low-quality slots lead to ambiguous feature attribution. Second, adding a slot-count penalty to the reconstruction objective creates conflicting optimization goals between reducing the number of active slots and maintaining reconstruction fidelity. As a result, they still lag significantly behind strong K -fixed baselines. To address these challenges, we propose **Quality-Guided K -Adaptive Slot Attention (QASA)**. First, we decouple slot selection from reconstruction, eliminating the mutual constraints between the two objectives. Then, we propose an unsupervised Slot-Quality metric to assess per-slot quality, providing a principled signal for fine-grained slot-object binding. Based on this metric, we design a Quality-Guided Slot Selection scheme that dynamically selects a subset of high-quality slots and feeds them into our newly designed gated decoder for reconstruction during training. At inference, token-wise competition on slot attention yields a K -adaptive outcome. We conduct experiments on both object discovery and object property prediction. Results show that QASA substantially outperforms existing K -adaptive methods on both real and synthetic datasets. Moreover, on real-world datasets, QASA surpasses K -fixed methods. The code is available at <https://github.com/ouyangtianran/QASA-tianran>.

* Corresponding author.

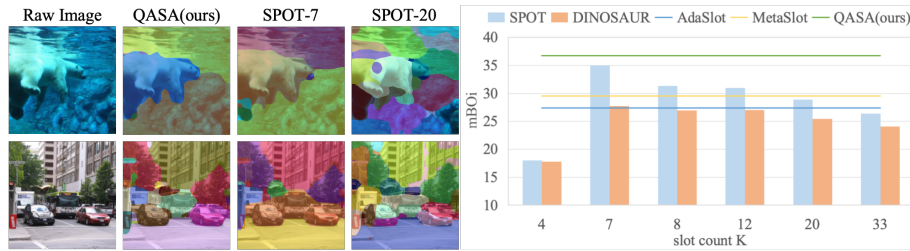


Fig. 1: mBOi vs. slot count K on COCO [25]. K -fixed methods, such as SPOT [20] and DINO SAUR [33], show large performance fluctuations as K varies. By contrast, K -adaptive methods do not rely on a carefully tuned K . Our method substantially outperforms existing K -adaptive baselines, including MetaSlot [26] and AdaSlot [16], and even surpasses strong K -fixed baselines.

1 Introduction

Object-centric learning (OCL) seeks to decompose scenes into sets of object representations that align with the causal structure of the physical world. Because scenes are compositional, such representations can improve robustness to distribution shifts [8] and support downstream tasks such as reasoning and control [1, 3, 29, 40]. Motivated by the abundance of unlabeled images, we focus on *unsupervised object-centric learning*, where models learn object-centric representations directly from visual data. Most unsupervised OCL methods optimize reconstruction objectives with set-structured bottlenecks that map an image to a small set of slots, which are then decoded back to the input space [27]. Within this line, Slot Attention has emerged as a mainstream approach [16, 33, 35, 36, 42].

Most current Slot Attention approaches fix the slot count K a priori. However, the number of objects varies dynamically across different images, making object-slot mismatches inevitable. As shown in Fig. 1, for K -fixed methods, model performance is highly sensitive to the chosen value of K . The contradiction between a fixed K and dynamically varying object counts hinders the further development of such techniques. A pioneering K -adaptive approach, AdaSlot [16], trains a slot selector by adding a slot-count penalty to the training objective. However, the reconstruction objective tends to use more slots, which conflicts with the slot-count penalty. Other lines of work perform slot selection via principal component analysis (CoSA [23]) or by maintaining a unified slot codebook (MetaSlot [26]). However, all these methods directly or indirectly rely on the prominence of object features, and thus tend to select only the most prominent objects for binding. This reliance on prominence leads to ambiguous slot-object attribution, where slots fail to bind precisely to specific objects and instead capture only coarse trends. Empirically, existing K -adaptive methods lag far behind K -fixed baselines, particularly in real-world scenarios. These issues limit the practical adoption of K -adaptive Slot Attention.

To address these challenges, we propose **Quality-Guided K -Adaptive Slot Attention (QASA)**. Unlike prior approaches that train a small auxiliary net-

work to select slots during reconstruction, QASA decouples slot selection from reconstruction. To select high-quality slots under unsupervised conditions, we introduce a novel Slot-Quality metric. Inspired by the competitive token assignment in Slot Attention [27, 33], we define slot’s quality as the proportion of its attention mass within its winning region. Ideally, a high-quality slot wins entirely on its own tokens and assigns negligible mass elsewhere. Building on this metric, we propose a Quality-Guided Slot Selection scheme that, for each image, iteratively adds the highest-quality slots to a qualified set until a coverage threshold is met. We feed the slots into gated decoders, using gating to suppress the contributions of unselected slots, and compute the reconstruction loss. For the object discovery task, we evaluate QASA on real-world datasets (COCO [25] and PASCAL VOC [15]) and synthetic datasets (MOVi-C and MOVi-E [17]). QASA substantially outperforms existing K-adaptive methods. Furthermore, it achieves state-of-the-art performance on real-world data relative to K-fixed methods. For the object property prediction task, QASA surpasses all baselines. Our key contributions are as follows:

- We introduce a novel unsupervised **Slot-Quality** metric, defined as the proportion of its attention mass within its winning region, providing an effective signal to guide precise slot-object binding.
- We propose a **Quality-Guided Slot Selection** module that jointly leverages quality, coverage, and novelty to guide the model to focus only on the most matching slot subset for each image.
- We design **gated Transformer** and **gated MLP** decoders. Under both types of decoders, our method surpasses existing approaches, demonstrating the high compatibility of our proposed quality-guided mechanism.
- Extensive experiments across object discovery and property prediction tasks demonstrate the broad superiority of QASA. Specifically, it outperforms existing K -adaptive baselines by an average of **8.4%** on mBOi for object discovery and achieves best results in the object property prediction task.

2 Related work

Object-Centric Learning. Object-centric learning (OCL) represents scenes as compositions of objects. Early approaches fall into two families: spatial-attention models that infer object bounding boxes [7, 14, 24] and scene-mixture models trained with variational inference [4, 12, 18]. Slot Attention [27] crystallizes a set-structured bottleneck by iteratively assigning features to a set of slots and has become a mainstream paradigm in recent years. Subsequent work has extended Slot Attention along four main directions: (i) *Reconstruction target*: As self-supervised pretrained models have become strong foundations for downstream tasks [5, 9, 31, 32], DINOSAUR [33] replaces pixel reconstruction with pretrained feature reconstruction to better scale to real images. (ii) *decoders*: SLATE [34] conditions an autoregressive/quantized decoder on slots to enhance detail and compositionality, and diffusion-based decoders further improve fidelity (e.g., LSD [19] and SlotDiffusion [39]); (iii) *training strategy*:

SPOT [20] leverages self-training with a patch-order permutation to strengthen slot pipelines; (iv) *video*: SAVi/SAVi++ [11, 22] extend slot attention to videos, while VideoSAUR [41] and SlotContrast [30] further incorporate self-supervision. However, these advances still assume a pre-specified slot count K . When K mismatches scene complexity, under- and over-segmentation and wasted capacity occur, hampering learning and transfer [16]. We therefore treat K as *per-image adaptive*, retaining an image-specific set of slots.

K -Adaptive Slot Attention. K -Adaptive Slot Attention aims to match model capacity to scene complexity instead of fixing K globally. Some K -fixed methods, such as SOLV [2] and DIAS [43], adjust the effective slot count by merging similar slots during inference, but cannot handle scenes with more objects than the preset K . In contrast, K -adaptive methods use a large slot budget or candidate pool, providing stronger adaptability. A representative K -adaptive method is AdaSlot [16], which starts from a large $K_{\max}$ and learns a selector with expected- K regularization, together with a masked decoder. Some methods expand the candidate slots by introducing external slot repositories. CoSA [23] selects slots from a grounded slot dictionary via principal component analysis. MetaSlot [26] goes further by learning a slot codebook to enlarge the candidate set. Outside the Slot Attention family, GENESIS-V2 employs a stochastic stick-breaking prior to infer a variable number of components [13]. While K -adaptive designs have improved flexibility, they often favor prominent objects over fine structures and still trail K -fixed methods. Our slot-quality criterion retains high-quality fine-grained slots, substantially narrowing this gap and surpassing K -fixed methods on complex scenes.

3 Method

3.1 Overview

Slot Attention typically comprises three parts: *encoder*, *slot-attention* and *decoder*. First, an encoder extracts N patch-wise features $Y \in \mathbb{R}^{N \times d_y}$ from an input image X . Then, a slot-attention module groups these features into K latent vectors $U = (u_1, \dots, u_K) \in \mathbb{R}^{K \times d_u}$, referred to as *slots*, each intended to represent one object in the image. Finally, a decoder reconstructs a target signal, either the original image X or the features Y , from these slots. Slot Attention is commonly evaluated using attention masks A that indicate the association of each image patch with a specific slot. These masks can be obtained either from the slot-attention module or from the decoder, denoted A^{slot} and A^{dec} , respectively. In this paper, we consistently use masks A^{slot} for evaluation.

Prior K -adaptive methods mitigate the fixed- K limitation through slot masking, selection, or pruning. AdaSlot [16] learns a selector with an expected- K regularizer, but reconstruction naturally favors activating more slots, creating a conflict between reconstruction fidelity and slot-count reduction. CoSA [23] decouples slot selection using a PCA-based criterion, while MetaSlot [26] maintains a global codebook to quantize and prune redundant slots. However, these criteria

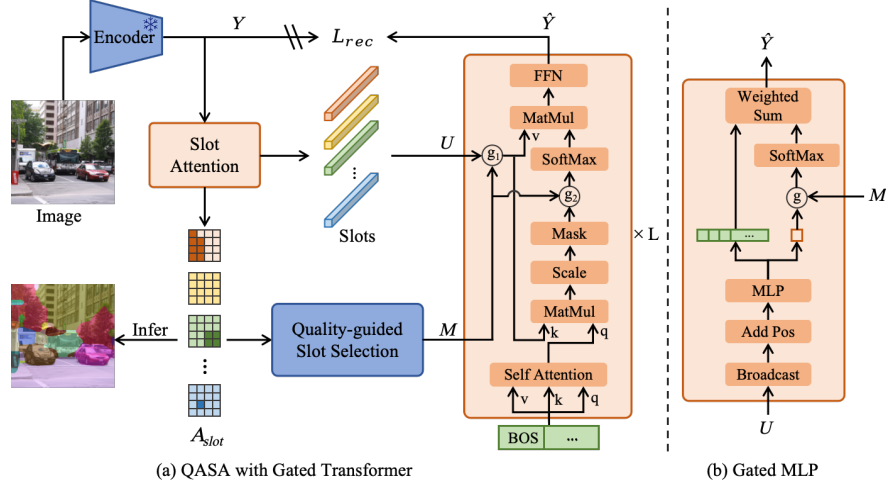


Fig. 2: Framework of Quality-Guided K -Adaptive Slot Attention (QASA). The overall training pipeline follows the canonical Slot Attention architecture: an image is encoded by the encoder, fed into the Slot Attention module, and then decoded; the reconstruction loss is computed against embeddings produced by a *frozen* encoder. The key difference is that we compute a slot-quality metric from the Slot Attention maps A^{slot} and use it to select a subset of slots, allowing only the selected slots to participate in decoding. At inference, no slot selection is applied. All slots compete via A^{slot} for each token, yielding a K -adaptive result.

mainly aim to reduce slot usage or retain prominent components, without explicitly specifying what properties the retained slots should satisfy. As a result, selected slots may still capture broad or diffuse regions rather than fine-grained object-specific structures with attention concentrated on the target object. We instead introduce an unsupervised *Slot-Quality* metric that evaluates whether a slot binds fine-grained target features within its winning region, providing an explicit criterion for selecting high-quality slots. Building on this metric, we introduce **Quality-Guided K -Adaptive Slot Attention (QASA)**, which selects slots through a metric-based procedure, thereby avoiding the contradiction introduced by slot-count regularization. As shown in Fig. 2(a), QASA consists of four parts: *encoder*, *slot-attention*, *slot selection* and *gated decoder*.

During training, we set a maximum slot count K_{max} , and we select an appropriate set of slots according to A^{slot} (detailed in § 3.2) as $M = F_{\text{select}}(A^{\text{slot}})$. Then we feed the slots together with the selection mask into our proposed gated decoder (detailed in §. 3.3), obtaining $\hat{Y} = \text{Dec}_{\text{gated}}(U, M)$. Following DINO SAUR [33] and SPOT [20], we use DINO [6] as the image encoder and reconstruct its features $\hat{Y} \in \mathbb{R}^{N \times d_y}$ as the training target. The model is trained with the reconstruction loss: $\mathcal{L}_{\text{rec}} = \frac{1}{N \cdot d_y} \|Y - \hat{Y}\|_2^2$.

During inference, no slot selection is applied and *all* slots compete, based on A^{slot} , for each token to produce a K -adaptive result. Concretely, for each

patch $y_t \in Y$ we take a hard winner over the slot-attention map A^{slot} via $w_t = \operatorname{argmax}_{i \in \{1, \dots, K_{\max}\}} A_{t,i}^{\text{slot}}$, and assign patch y_t to the slot with the strongest association, i.e., y_t is assigned to slot w_t . This induces a hard partition of tokens, $\mathcal{S}_i = \{t \mid w_t = i\}$, and we treat the union of patches indexed by $\mathcal{S}_i$ as the region assigned to slot i . Since the slot-attention masks are defined on the encoder’s token grid (low resolution), we upsample them to the original image resolution before evaluating against ground-truth masks.

3.2 Quality-Guided Slot Selection

To perform training-free slot selection, the central challenge is to evaluate the quality of the slots in an unsupervised setting. The attention map of the slot-attention module A^{slot} provides an excellent signal. Ideally, the attention of a high-quality slot A_i^{slot} would be 1 over its object region and 0 elsewhere, but in the unsupervised setting the object regions are unknown. Inspired by Slot Attention’s competitive assignment mechanism, in which attention maps induce competition among slots for responsibility over all tokens, we propose a competition-based slot-quality metric.

Let $A \in [0, 1]^{N \times K_{\max}}$ denote the slot-attention probabilities for all tokens (patches) and slots, normalized over slots (i.e., $\sum_{i=1}^{K_{\max}} A_{t,i} = 1$ for each token t). For each token t , define its winner slot:

$$w_t = \operatorname{argmax}_{i \in \{1, \dots, K_{\max}\}} A_{t,i}. \quad (1)$$

For slot i , define the winner mass and total mass:

$$W_i^{\text{win}} = \sum_{t: w_t=i} A_{t,i}, \quad W_i = \sum_{t=1}^N A_{t,i}. \quad (2)$$

The slot-quality score is

$$Q_i = \frac{W_i^{\text{win}}}{W_i + \epsilon} \in [0, 1], \quad (3)$$

which measures the purity of slot_i ’s mass within the region where it wins.

Based on *slot-quality*, we propose a Quality-Guided Slot Selection scheme that also considers *coverage*, as detailed in Algorithm 1. We set a *per-token coverage-quality threshold* $\tau \in (0, 1]$ and a *coverage-rate threshold* $\rho \in (0, 1]$. A token t is marked as *covered* if the cumulative attention mass from the selected slots $S \subseteq \{1, \dots, K_{\max}\}$ exceeds τ :

$$\text{Covered}_S(t) = \mathbb{1} \left[\sum_{i \in S} A_{t,i} \geq \tau \right]. \quad (4)$$

The overall coverage rate is then

$$\text{Coverage}(S) = \frac{1}{N} \sum_{t=1}^N \text{Covered}_S(t). \quad (5)$$

Algorithm 1 Quality-Guided Slot Selection

Input: Slot-attention probabilities $A \in [0, 1]^{N \times K_{\max}}$, thresholds τ, ρ, μ .
Output: Selection mask $M \in \{0, 1\}^{K_{\max}}$.

- 1: **Compute qualities:** obtain Q using Eq. 3 (with w_t from Eq. 1 and W_i^{win}, W_i from Eq. 2).
- 2: $S \leftarrow \emptyset, \quad M \leftarrow \mathbf{0}_{K_{\max}}$.
- 3: $\pi \leftarrow \text{argsort}_{i \in \{1, \dots, K_{\max}\}}(-Q_i)$ $\triangleright$ indices in descending Q
- 4: **for** $j = 1$ to $K_{\max}$ **do**
- 5: $i \leftarrow \pi_j$.
- 6: **if** $\text{novelty}(i \mid S)$ from Eq. 6 $< \mu$ **then continue**
- 7: **end if**
- 8: $S \leftarrow S \cup \{i\}$.
- 9: **if** $\text{Coverage}(S)$ from Eq. 5 (with Covered_S from Eq. 4) $\geq \rho$ **then**
- 10: **break**
- 11: **end if**
- 12: **end for**
- 13: Set $M_i \leftarrow 1$ for all $i \in S$; **return** M .

We greedily add slots in descending order of quality Q_i until $\text{Coverage}(S) \geq \rho$. To avoid repeatedly selecting slots that mostly explain already-covered regions, we enforce a *novelty constraint*: a candidate slot i is skipped if the fraction of its mass on already-covered tokens exceeds a threshold. Writing $C_S = \{t \mid \text{Covered}_S(t) = 1\}$ for the current covered set, the novelty of slot i w.r.t. S is

$$\text{novelty}(i \mid S) = 1 - \frac{\sum_{t \in C_S} A_{t,i}}{\sum_{t=1}^n A_{t,i} + \epsilon}. \quad (6)$$

We accept i only if $\text{novelty}(i \mid S) \geq \mu$ with $\mu \in [0, 1)$. The resulting mask M is used during training; no mask is applied at inference.

3.3 Gated Decoder

After completing slot selection, we need to remove the influence of unselected slots during decoding. A straightforward approach is to dynamically gate all slots according to the mask, suppressing the contribution of unselected slots during decoding. For a fair comparison, we simply augment the MLP [33] and Transformer [20] decoders used in prior K -fixed methods with our gating mechanism, without introducing any additional network components.

Gated Transformer Decoder. We adapt the standard Transformer decoder [20] by introducing a dual gating mechanism into its cross-attention layers, as illustrated in Fig. 2(a). This mechanism leverages the selection mask $M \in \{0, 1\}^{K_{\max}}$ from our quality-guided selection process (Algorithm 1) to create two distinct gates for fine-grained control over slot participation.

Given the selection mask M , we define a K/V gate $g_1 \in \{\varepsilon_1, 1\}_{\max}^K$ and a logits gate $g_2 \in \{\varepsilon_2, 1\}_{\max}^K$, where $0 < \varepsilon_1, \varepsilon_2 < 1$. For each slot $i \in \{1, \dots, K_{\max}\}$,

the gates are defined as

$$(g_1)_i = M_i + (1 - M_i)\varepsilon_1, \quad (g_2)_i = M_i + (1 - M_i)\varepsilon_2. \quad (7)$$

Within each cross-attention block, the attention mechanism is modified as follows. First, the key (K) and value (V) vectors from the slots are multiplicatively modulated by the gate g_1 to physically suppress the information flow from unselected slots. Second, an additive bias based on the gate g_2 is applied to the attention logits before the softmax operation. This penalizes the scores of unselected slots, ensuring they do not win the attention competition. The full gated attention output is computed as:

$$\text{Output} = \text{softmax} \left(\frac{Q(K \odot g_1)^\top}{\sqrt{d_k}} + \log(g_2) \right) (V \odot g_1), \quad (8)$$

where $\odot$ denotes the element-wise product broadcasted across the feature dimension. This mechanism ensures unselected slots are robustly pruned from both the information pathway (via g_1) and the competitive attention allocation (via g_2).

Gated MLP Decoder. The MLP decoder predicts per-slot reconstructions $\{\hat{y}_{i,t} \in \mathbb{R}^d\}_{i=1}^{K_{\max}}$ for tokens $t = 1, \dots, N$ together with mixture logits $\ell \in \mathbb{R}^{K_{\max} \times N}$ over the slot dimension. Given a binary selection mask $M \in \{0, 1\}^{K_{\max}}$ (active set $S = \{i \mid M_i = 1\}$), we apply hard gating at the logit level, as shown in Fig. 2(b), and then normalize only over active slots:

$$\ell'_{i,t} = \begin{cases} \ell_{i,t}, & M_i = 1, \\ -\infty, & M_i = 0, \end{cases} \quad \text{where } i \in \{1, \dots, K_{\max}\}, \quad t \in \{1, \dots, N\}. \quad (9)$$

$$\alpha_{i,t} = \frac{\exp(\ell'_{i,t})}{\sum_{j=1}^{K_{\max}} \exp(\ell'_{j,t})} = \frac{M_i \exp(\ell_{i,t})}{\sum_{j=1}^{K_{\max}} M_j \exp(\ell_{j,t})}, \quad (10)$$

The final reconstruction aggregates per-slot predictions using the gated mixture:

$$\hat{y}_t = \sum_{i=1}^{K_{\max}} \alpha_{i,t} \hat{y}_{i,t}, \quad t \in \{1, \dots, N\}. \quad (11)$$

The masks reported for evaluation are $\alpha \in [0, 1]^{K_{\max} \times N}$ reshaped to the token grid. In practice, we implement the $-\infty$ gate by filling inactive-slot logits with a large negative constant $-\mathcal{C}$ (with $\mathcal{C} \gg 0$) before the softmax. Equivalently, $\ell'_{i,t} = \ell_{i,t} - (1 - M_i)\mathcal{C}$. We ensure $|S| \geq 1$.

4 Experiments

4.1 Setup

For object discovery, we evaluate on two widely used real-world datasets, **MS COCO 2017** [25] and **PASCAL VOC 2012** [15], and two synthetic datasets,

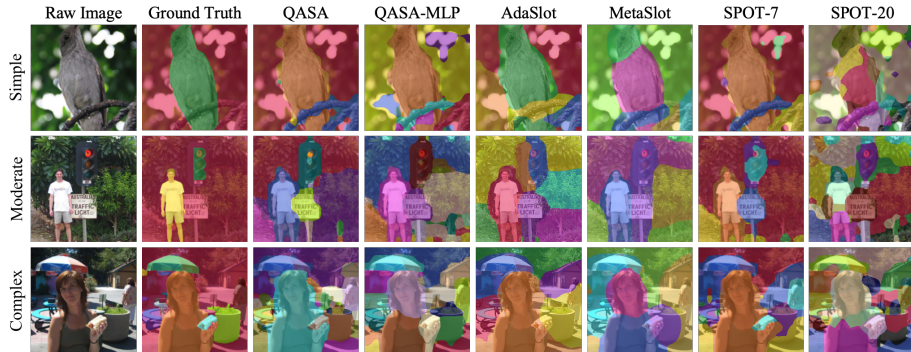


Fig.3: Visualizations on COCO. The MetaSlot examples of AdaSlot [16] and MetaSlot [26] are generated using the released checkpoint. "SPOT-7" and "SPOT-20" denote SPOT [20] with $K = 7$ and $K = 20$, respectively. The SPOT-7 examples are generated using the released checkpoint. The SPOT-20 examples are generated by training a model under the same settings as in the original paper.

MOVi-C and **MOVi-E** [17]. Our emphasis is on real-world scenes. COCO contains diverse and cluttered images with many co-occurring objects and is particularly challenging, whereas VOC typically features one or a few salient objects and serves as a complementary regime. For synthetic evaluation, MOVi provides rendered scenes with high-quality 3D-scanned assets. MOVi-C has 3–10 objects per scene, and MOVi-E has 11–23. For object property prediction, we follow the DINOSAUR and conduct experiments on COCO [33].

Metrics. Following recent works [16, 20, 33, 36], we report Mean Best Overlap (mBO) at the instance (mBOi) and class (mBOc) levels. For mBO, each ground-truth mask is paired with the single predicted mask that maximizes IoU, allowing many-to-one matches, and scores are then averaged per instance (mBOi) or per class (mBOc). We also report mean Intersection over Union (mIoU) under a stricter one-to-one protocol, where predicted and ground-truth masks are matched with the Hungarian algorithm before averaging. In line with previous works [20, 36], we do not report FG-ARI in our comparisons. FG-ARI ignores background regions and is insensitive to boundary quality, which can yield misleadingly optimistic segmentation scores [20]. For property prediction, we evaluate macro-averaged Top-1/Top-5 accuracy for classification and the coefficient of determination (R^2) for center coordinates (R^2 -X, R^2 -Y) [33].

Implementation Details. Unless otherwise specified, all settings follow SPOT [20]. For all datasets, we set $\rho = 0.8$ and $\mu = 0.3$. For COCO and VOC, we set $\tau = 0.5$. And for MOVi, we set $\tau = 0.8$. For COCO and VOC, we set $K_{\max}$ to the maximum number of masks plus one (to include background), i.e., $K_{\max} = 33$ and $K_{\max} = 20$, respectively.¹ For MOVi-C and MOVi-E, we round the maximum mask count up to the nearest multiple of ten and use $K = 20$ and

¹ In the VOC training set, apart from a single sample with 38 masks, all others have at most 19 masks, so we ignore this outlier.

Table 1: Comparison of K -adaptive methods. Results for AdaSlot and MetaSlot are taken from their original papers [16, 26]. Dashes indicate results not reported.

Method	Dec.	K -type	COCO		VOC		MOVi-C	MOVi-E
			mBOi	mIoU	mBOi	mIoU	mBOi	mBOi
AdaSlot [16]	MLP	adaptive	27.4	–	–	–	35.6	29.8
MetaSlot [26]	MLP	adaptive	29.5	27.9	43.9	42.1	35.0	–
QASA	Trans.	adaptive	36.7	35.0	49.7	47.9	46.9	39.1
QASA _{MLP}	MLP	adaptive	33.6	32.5	47.3	45.5	42.2	34.6

$K = 30$, respectively. We use $4 \times$ RTX 4090 GPUs for COCO and $2 \times$ RTX 4090 GPUs for the other datasets. With the Transformer decoder, the per-GPU batch size is 256. With the MLP decoder, the per-GPU batch size is 32. We train for 100 epochs on COCO and MOVi, and for 600 epochs on PASCAL VOC. For training stability, we adopt a warm-up strategy in which all slots participate in reconstruction in the early epochs. With the Gated Transformer decoder, we set $warmup_{gate} = 10$ on VOC and 20 on MOVi-C/E. With the Gated MLP decoder, we set $warmup_{gate} = 10$ on VOC, 10 on MOVi-C, and 5 on MOVi-E. On COCO, we disable this warm-up (i.e., $warmup_{gate} = 0$). Detailed experimental settings are provided in the supplementary material.

4.2 Object Discovery

Comparison with K -adaptive methods. To the best of our knowledge, AdaSlot [16], CoSA [23], and MetaSlot [26] are the only three K -adaptive Slot Attention methods to date. Because CoSA lacks publicly available experimental artifacts for comparison, we report results against AdaSlot and MetaSlot only. As shown in Tab. 1, our method QASA outperforms AdaSlot and MetaSlot by a large margin on both real-world (COCO) and synthetic (MOVi) datasets, with an average improvement of **8.4%** in mBOi. Even when using an MLP decoder to match their setup, QASA_{MLP} still surpasses them substantially, achieving a **4.7%** gain in mBOi. We present visualizations stratified by scene complexity to intuitively demonstrate the advantages of QASA on COCO, as shown in Fig. 3. It is clear that QASA outperforms AdaSlot [16] and MetaSlot [26] across all three levels (simple, moderate, and complex). QASA adapts to varying object counts and captures fine-grained object details, whereas AdaSlot largely partitions images into coarse blob-like regions. This issue is especially pronounced in complex scenes. MetaSlot can capture certain fine details, but its performance degrades markedly in complex scenes. On the synthetic datasets [17], our method likewise outperforms AdaSlot [16], as shown in Fig. 4, capturing sharper object boundaries rather than merely coarse blob-like outlines. The visualizations show that QASA learns slots that capture fine object details and bind more precisely to objects, while preserving the ability to adapt the number of objects per image.

To more directly illustrate the advantage of QASA, we further evaluate representative K -fixed methods (SPOT [20] and DINOSAUR [33]) on COCO under

Table 2: Comparison with K -fixed methods on COCO [25], PASCAL VOC [15], and MOVi-C/E [17]. Results for SLATE, DINOSAUR, DINOSAUR_{MLP}, and SPOT are taken from SPOT [20]. Results for the remaining methods are taken from their original papers [36]. Dashes indicate results not reported.

Method	Dec.	K -type	COCO			VOC			MOVi-C		MOVi-E	
			mBOc	mBOi	mIoU	mBOc	mBOi	mIoU	mBOi	mIoU	mBOi	mIoU
SLATE [34]	–	–	33.6	29.1	–	41.5	35.9	–	39.4	37.8	30.2	28.6
DINOSAUR [33]	Trans.	fixed	39.7	31.6	–	51.2	44.0	–	42.4	–	–	–
DINOSAUR _{MLP} [33]	MLP	fixed	30.9	27.7	–	40.9	39.5	–	39.1	–	35.5	–
SPOT [20]	Trans.	fixed	44.7	35.0	33.0	55.6	48.3	–	47.3	46.7	40.1	39.3
SPOT+FS+RC [36]	Trans.	fixed	45.3	35.7	–	56.5	49.3	–	49.0	47.8	–	–
DINOSAUR+FS+RC [36]	MLP	fixed	40.1	31.2	–	52.9	45.2	–	46.0	44.8	–	–
QASA	Trans.	adaptive	45.5	36.7	35.0	57.9	49.7	47.9	46.9	46.1	39.1	37.8
QASA _{MLP}	MLP	adaptive	36.5	33.6	32.5	53.8	47.3	45.5	42.2	41.1	34.6	33.0

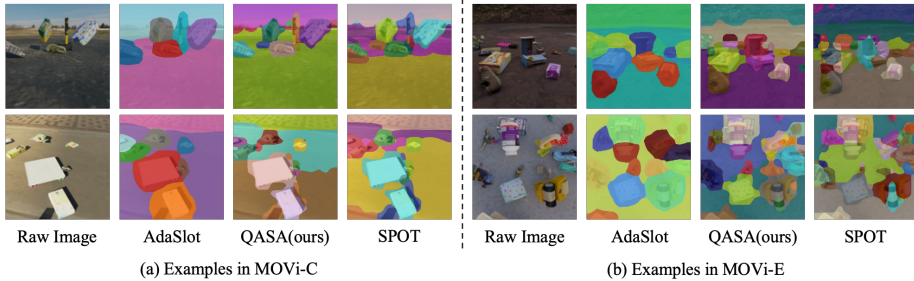


Fig. 4: Visualizations on MOVi-C/E. The AdaSlot examples are taken from its original paper [16], the SPOT examples are generated by training a model under the same settings as in the original paper.

varying K (Fig. 1). As K changes, their performance fluctuates markedly. This sensitivity necessitates exhaustive per-dataset K sweeps, which are time- and compute-intensive, yet these sweeps still do not resolve the structural mismatch between a fixed slot budget and image-dependent object counts. Existing K -adaptive methods (AdaSlot [16] and MetaSlot [26]) can adaptively select the number of slots used per image, which saves substantial tuning effort and resources. However, they show a clear performance gap compared with K -fixed methods because they lack the slot-count prior. By contrast, QASA not only adaptively selects appropriate slots for each image for reconstruction, but also surpasses K -fixed methods in performance, thereby addressing the shortcoming of K -adaptive methods.

Comparison with K -fixed methods. Tab. 2 summarizes recent K -fixed methods, including SLATE [34], DINOSAUR [33], SPOT [20], and +FS+RC [36]. On the real-world datasets COCO and VOC, QASA achieves *state-of-the-art* results across all metrics, despite lacking the strong inductive bias provided by a pre-set K . On the synthetic datasets, our method does not outperform all K -fixed

Table 3: Object property prediction results. We report classification accuracy and R^2 scores using an MLP predictor and a linear predictor. DINOSAUR [33] and MetaSlot [26] are trained with their official code. SPOT [20] and AdaSlot [16] are evaluated using slots extracted from their official checkpoints. All baselines use the same backbone and evaluation setup as our model.

Method	Dec.	K -type	MLP predictor				Linear predictor			
			class-top1	class-top5	R^2 -X	R^2 -Y	class-top1	class-top5	R^2 -X	R^2 -Y
DINOSAUR [33]	Trans.	fixed	47.1	75.5	55.4	59.1	39.7	68.0	48.8	52.0
DINOSAUR _{MLP} [33]	MLP	fixed	46.9	75.4	62.9	63.3	39.1	67.5	59.2	59.9
SPOT [20]	Trans.	fixed	45.9	72.7	51.9	56.4	45.2	72.1	49.2	53.4
AdaSlot [16]	MLP	adaptive	49.3	79.0	72.7	69.8	33.9	66.3	58.6	53.3
MetaSlot [26]	MLP	adaptive	40.9	70.0	55.0	55.3	37.9	66.7	52.0	53.2
QASA	Trans.	adaptive	59.6	85.0	67.9	71.7	57.3	82.3	67.9	69.9
QASA _{MLP}	MLP	adaptive	58.9	84.1	74.0	76.7	46.7	76.7	62.3	56.5

methods. We attribute this to the more concentrated object-count distribution in synthetic data, where a carefully chosen fixed K provides larger gains, and additional analysis is provided in the appendix. We also note that the comparison with +FS+RC [36] is not directly comparable, since it uses an extra foreground background indicator learned with additional training. Overall, unlike AdaSlot [16] and MetaSlot [26], whose strong performance is largely confined to toy or synthetic settings, QASA surpasses K -fixed methods on real data and therefore offers clear practical benefits.

4.3 Object Property Prediction

To further demonstrate the downstream potential of the slots learned by our method [28], we conduct experiments on the object property prediction task. We evaluate DINOSAUR [33], SPOT [20], AdaSlot [16], and MetaSlot [26] as baselines, training and testing all methods under the same protocol, with details provided in the appendix. Following prior work [33], we conduct experiments on COCO using both MLP and linear predictors. As shown in Tab. 3, our method consistently outperforms all baselines under both predictors. With the stronger MLP predictor, QASA improves top-1 classification by 10.3% and increases the average R^2 by 4.1%. These results indicate that QASA learns a more informative scene representation, where each slot encodes more accurate object information, which supports its strong potential for downstream applications.

4.4 Ablation Study

For an in-depth analysis of the contributions of each component, we conduct several ablation studies. All ablations use the Transformer decoder. Unless otherwise specified, all hyperparameters remain unchanged.

Components of Quality-Guided Slot Selection. Starting from a model without gating, we progressively enable the components of the quality-guided

Table 4: Influence of key components in Slot Selection. Coverage, Quality, and Novelty denote the corresponding mechanisms in the Quality-Guided Slot Selection. All results are evaluated on COCO [25].

Coverage	Quality	Novelty	mBOc	mBOi	mIoU
×	×	×	25.9	25.5	24.9
✓	×	×	34.5	25.3	22.7
✓	✓	×	45.0	35.0	32.9
✓	✓	✓	45.5	36.7	35.0

Table 5: Ablation on Gates in the Transformer decoder. All results are evaluated on COCO [25].

g_1	g_2	mBOc	mBOi	mIoU
×	×	25.9	25.5	24.9
×	✓	26.1	25.5	24.9
✓	×	43.1	33.2	30.9
✓	✓	45.5	36.7	35.0

slot selection (*coverage*, *quality*, and *novelty*), as shown in Tab. 4. Enabling *coverage* alone chiefly lifts mBOc but slightly harms mBOi and mIoU because diffuse/low-quality slots are admitted to meet the coverage target. Adding *quality* yields the largest jump overall by prioritizing high-quality slots, improving one-to-one alignment and boundary tightness. Incorporating *novelty* further gives a gain on mBOc and clear gains on mBOi/mIoU by filtering overlapping slots.

Gates in the Transformer decoder. We further ablate the dual gating mechanism, as reported in Tab. 5. Using only g_2 (logit gating) fails to enforce effective masking, likely because unattenuated logits can overwhelm the negative bias. Using only g_1 (key/value gating) already yields strong control, and combining g_1 with g_2 leads to additional gains. Concretely, g_1 scales the key and value vectors of unselected slots toward zero, which effectively shrinks their attention logits, while g_2 further adds a large negative bias to these logits, providing an extra margin that suppresses the influence of unselected slots.

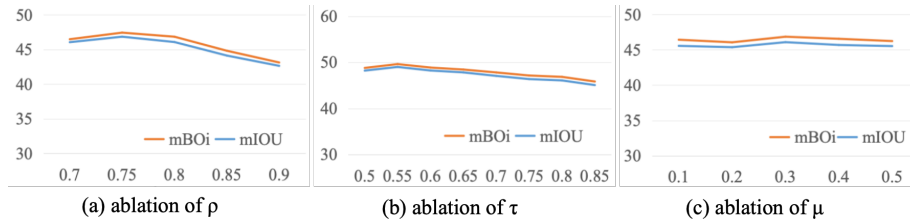


Fig. 5: Influence of ρ , τ and μ . All results are evaluated on MOVi-C.

Influence of ρ , τ , and μ . Fig. 5 shows how mBOi and mIoU vary with the coverage-rate threshold ρ , the per-token coverage-quality threshold τ , and the novelty threshold μ . The threshold ρ determines whether the selected slots achieve sufficient spatial coverage and is therefore the basis of gating. In practice, ρ around 0.7 to 0.8 already yields reliable gating. The threshold τ controls the token-level covered decision. Once ρ is fixed, performance is largely insensitive to τ over a wide range. Adding novelty consistently improves performance, and

the gains are stable across a wide range of μ . We do not tune these thresholds per dataset, since minor changes from fine tuning are not our focus, and the current results are sufficient to demonstrate the advantages of our model.

Table 6: Effect of $K_{\max}$ on MOVi-C and MOVi-E [17].

	$K_{\max}$	15	18	20	22	24	32
MOVi-C	mBOi	44.4	46.5	46.9	46.4	46.3	45.4
	mIoU	43.4	45.7	46.1	45.6	45.7	45.0
	$K_{\max}$	24	28	30	32	34	64
MOVi-E	mBOi	38.3	39.1	39.1	39.3	39.5	38.7
	mIoU	36.8	37.8	37.8	38.0	38.3	38.3

Choice of $K_{\max}$. Unlike K -fixed methods that require searching for an optimal K , K -adaptive methods only specify $K_{\max}$ and adaptively select effective slots per image. In practice, setting $K_{\max}$ to the maximum number of objects plus one (for background) usually works well. However, on the synthetic datasets MOVi-C and MOVi-E, we observe that moderately increasing $K_{\max}$ can yield better results, as shown in Tab. 6. The gains from such slot redundancy are noticeably smaller on MOVi-E than on MOVi-C. We hypothesize this is because synthetic backgrounds are highly homogeneous, causing several slots to specialize to a few background modes. Increasing the number of redundant slots therefore helps. Since MOVi-E has more complex backgrounds than MOVi-C, this effect is reduced. Importantly, even without such redundancy ($K_{\max}=15$ for MOVi-C and $K_{\max}=24$ for MOVi-E), QASA still performs strongly. The table also shows that QASA is highly robust to the choice of $K_{\max}$: performance remains stable over a wide range, and even when $K_{\max}$ is set far beyond the maximum object count ($K_{\max}=32$ for MOVi-C and $K_{\max}=64$ for MOVi-E), the degradation is minor, further supporting the effectiveness of the proposed K -adaptive mechanism.

5 Conclusion

We introduce a *Slot-Quality* metric that drives slots toward fine, object-specific features. Building on this metric, we present QASA, a K -adaptive Slot Attention method that selects the appropriate slots *per image*, better handling uncertainty in object cardinality. We decouple slot selection from decoding, removing the conflict between slot-count regularization and reconstruction. The framework supports both Transformer and MLP decoders. On object discovery, QASA markedly outperforms existing K -adaptive methods, with up to an 8.4% average improvement on mBOi. Without hand-tuning an optimal K , QASA achieves state-of-the-art performance on real-world datasets and outperforms K -fixed methods. On object property prediction, QASA substantially surpasses all baselines on both classification and position regression, demonstrating strong

downstream potential. With a simple architecture, QASA dynamically determines the slot count from each scene and learns more informative and structured representations, which further highlights the promise of K -adaptive OCL.

Acknowledgments

This work was supported by the Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (No. JYB2025XDXM101), the National Natural Science Foundation of China (Grant No. 62471342), the Fundamental Research Funds for the Central Universities (No. 2042026kf0044), the New Cornerstone Science Foundation through the XPLOER PRIZE, and the Innovative Research Group Project of Hubei Province under Grant No. 2024AFA017.

References

1. Assouel, R., Rodriguez, P., Taslakian, P., Vazquez, D., Bengio, Y.: Object-centric compositional imagination for visual abstract reasoning. In: ICLR 2022 Workshop on the Elements of Reasoning: Objects, Structure and Causality (2022)
2. Aydemir, G., Xie, W., Guney, F.: Self-supervised object-centric learning for videos. *NeurIPS* **36**, 32879–32899 (2023)
3. Biza, O., Platt, R., van de Meent, J.W., Wong, L.L., Kipf, T.: Binding actions to objects in world models. In: ICLR 2022 Workshop on the Elements of Reasoning: Objects, Structure and Causality (2022)
4. Burgess, C.P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M., Lerchner, A.: Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390* (2019)
5. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: ICCV. pp. 9650–9660 (2021)
6. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: ICCV. pp. 9650–9660 (2021)
7. Crawford, E., Pineau, J.: Spatially invariant unsupervised object detection with convolutional neural networks. In: AAAI. vol. 33, pp. 3412–3420 (2019)
8. Dittadi, A., Papa, S., De Vita, M., Schölkopf, B., Winther, O., Locatello, F.: Generalization and robustness implications in object-centric learning. *ICML* (2022)
9. Dong, X., Shen, J., Shao, L.: Rethinking clustering-based pseudo-labeling for unsupervised meta-learning. In: ECCV. pp. 169–186. Springer (2022)
10. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR (2020)
11. Elsayed, G., Mahendran, A., Van Steenkiste, S., Greff, K., Mozer, M.C., Kipf, T.: Savi++: Towards end-to-end object-centric learning from real-world videos. *NeurIPS* **35**, 28940–28954 (2022)
12. Engelcke, M., Kosiorek, A.R., Jones, O.P., Posner, I.: Genesis: Generative scene inference and sampling with object-centric latent representations. *ICLR* (2020)

13. Engelcke, M., Parker Jones, O., Posner, I.: Genesis-v2: Inferring unordered object representations without iterative refinement. *NeurIPS* **34**, 8085–8094 (2021)
14. Eslami, S., Heess, N., Weber, T., Tassa, Y., Szepesvari, D., Hinton, G.E., et al.: Attend, infer, repeat: Fast scene understanding with generative models. *NeurIPS* **29** (2016)
15. Everingham, M., Winn, J.: The pascal visual object classes challenge 2012 (voc2012) development kit. *PASCAL* **8**(5), 2–5 (2011)
16. Fan, K., Bai, Z., Xiao, T., He, T., Horn, M., Fu, Y., Locatello, F., Zhang, Z.: Adaptive slot attention: Object discovery with dynamic slot number. In: *CVPR*. pp. 23062–23071 (2024)
17. Greff, K., Belletti, F., Beyer, L., Doersch, C., Du, Y., Duckworth, D., Fleet, D.J., Gnanapragasam, D., Golemo, F., Herrmann, C., et al.: Kubric: A scalable dataset generator. In: *CVPR*. pp. 3749–3761 (2022)
18. Greff, K., Kaufman, R.L., Kabra, R., Watters, N., Burgess, C., Zoran, D., Matthey, L., Botvinick, M., Lerchner, A.: Multi-object representation learning with iterative variational inference. In: *ICML*. pp. 2424–2433. PMLR (2019)
19. Jiang, J., Deng, F., Singh, G., Ahn, S.: Object-centric slot diffusion. In: *NeurIPS* (2023)
20. Kakogeorgiou, I., Gidaris, S., Karantzas, K., Komodakis, N.: Spot: Self-training with patch-order permutation for object-centric learning with autoregressive transformers. In: *CVPR*. pp. 22776–22786 (2024)
21. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. *ICLR* (2015)
22. Kipf, T., Elsayed, G.F., Mahendran, A., Stone, A., Sabour, S., Heigold, G., Jonschkowski, R., Dosovitskiy, A., Greff, K.: Conditional object-centric learning from video. In: *ICLR* (2022)
23. Kori, A., Locatello, F., Ribeiro, F.D.S., Toni, F., Glocker, B.: Grounded object-centric learning. In: *ICLR* (2024)
24. Kosiorek, A., Kim, H., Teh, Y.W., Posner, I.: Sequential attend, infer, repeat: Generative modelling of moving objects. *NeurIPS* **31** (2018)
25. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: *ECCV*. pp. 740–755 (2014)
26. Liu, H., Zhao, R., Chen, H., Pajarinen, J.: Metaslot: Break through the fixed number of slots in object-centric learning. *NeurIPS* (2025)
27. Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., Kipf, T.: Object-centric learning with slot attention. *NeurIPS* **33**, 11525–11538 (2020)
28. Mamaghan, A.M.K., Papa, S., Johansson, K.H., Bauer, S., Dittadi, A.: Exploring the effectiveness of object-centric representations in visual question answering: Comparative insights with foundation models. In: *ICLR* (2025)
29. Mambelli, D., Träuble, F., Bauer, S., Schölkopf, B., Locatello, F.: Compositional multi-object reinforcement learning with linear relation networks. In: *ICLR 2022 Workshop on Generalizable Policy Learning in Physical World* (2022)
30. Manasyan, A., Seitzer, M., Radovic, F., Martius, G., Zadaianchuk, A.: Temporally consistent object-centric learning by contrasting slots. In: *CVPR*. pp. 5401–5411 (2025)
31. Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al.: Dinov2: Learning robust visual features without supervision. *TMLR* pp. 1–31 (2024)
32. Ouyang, T., Dong, X., Ye, M., Du, B., Shao, L., Shen, J.: Semantic-aware pseudo-labeling for unsupervised meta-learning. *IEEE TPAMI* **47**(07), 5475–5488 (2025)

33. Seitzer, M., Horn, M., Zadaianchuk, A., Zietlow, D., Xiao, T., Simon-Gabriel, C.J., He, T., Zhang, Z., Schölkopf, B., Brox, T., et al.: Bridging the gap to real-world object-centric learning. In: ICLR (2023)
34. Singh, G., Ahn, S., Deng, F.: Illiterate dall-e learns to compose. In: ICLR (2022)
35. Singh, K., Schaub-Meyer, S., Roth, S.: Glass: Guided latent slot diffusion for object-centric learning. In: CVPR. pp. 28673–28683 (2025)
36. Tian, P., Yang, S., Yu, H., Kot, A.: Pay attention to the foreground in object-centric learning. In: CVPR. pp. 30281–30290 (2025)
37. Tschannen, M., Kumar, M., Steiner, A., Zhai, X., Houlsby, N., Beyer, L.: Image captioners are scalable vision learners too. NeurIPS **36**, 46830–46855 (2023)
38. Watters, N., Matthey, L., Burgess, C.P., Lerchner, A.: Spatial broadcast decoder: A simple architecture for disentangled representations in vaes. ICLR Workshops (2019)
39. Wu, Z., Hu, J., Lu, W., Gilitschenski, I., Garg, A.: Slotdiffusion: Object-centric generative modeling with diffusion models. NeurIPS **36**, 50932–50958 (2023)
40. Zadaianchuk, A., Seitzer, M., Martius, G.: Self-supervised visual reinforcement learning with object-centric representations. In: ICLR (2021)
41. Zadaianchuk, A., Seitzer, M., Martius, G.: Object-centric learning for real-world videos by predicting temporal feature similarities. NeurIPS **36**, 61514–61545 (2023)
42. Zhao, R., Wang, H., Kannala, J., Pajarinen, J.: Multi-scale fusion for object representation. In: ICLR (2025)
43. Zhao, R., Zhao, Y., Kannala, J., Pajarinen, J.: Slot attention with re-initialization and self-distillation. In: ACMMM. pp. 4185–4192 (2025)

Supplementary Material

A Implementation Details

For a fair comparison and to demonstrate the effectiveness of our method, we align our hyperparameters and settings as closely as possible with the latest open-source baseline, SPOT [20]. Details are provided below.

Training. We employ the Adam optimizer [21] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and no weight decay. The learning rate follows a linear warm-up from 0 to a peak value for 10000 iterations and then decreases via a cosine annealing schedule. For COCO and PASCAL, the peak learning rate is 4×10^{-4} and the minimum learning is 4×10^{-7} . For MOVi, the peak learning rate is 2×10^{-4} and the minimum learning is 4×10^{-5} . We use 4× RTX 4090 GPUs for COCO and 2× RTX 4090 GPUs for the other datasets. With the Transformer decoder, the per-GPU batch size is 256. With the MLP decoder, the per-GPU batch size is 32. We train for 100 epochs on COCO and MOVi, and for 600 epochs on PASCAL VOC. Dataset selection, preprocessing, and size configurations are identical to SPOT [20] and are therefore omitted for brevity.

Encoder. We implement our QASA using a ViT-B/16 encoder [10] initialized with DINO [6], and we omit the final layer norm.

Slot Attention. For all experiments, we use 3 Slot Attention iterations with slot dimension $d_u = 256$ and a slot-attention MLP hidden dimension of 1024. For COCO and VOC, we set $K_{\max}$ to the maximum number of masks plus one for background, yielding $K_{\max} = 33$ and $K_{\max} = 20$, respectively. In the VOC training set, except for one sample with 38 masks, all others have at most 19 masks. For MOVi-C and MOVi-E, we round the maximum mask count up to the nearest multiple of ten and use $K_{\max} = 20$ and $K_{\max} = 30$, respectively.

Slot Selection. We set $\rho = 0.8$ and $\mu = 0.3$ for all datasets. For COCO and VOC, we use $\tau = 0.5$. For MOVi, we use $\tau = 0.8$. To stabilize training, we adopt a warm-up strategy: during the first $warmup_{gate}$ epochs, no mask is applied and all slots participate in reconstruction. With the Transformer decoder, we set $warmup_{gate} = 10$ on VOC and 20 on MOVi-C/E. With the MLP decoder, we set $warmup_{gate} = 10$ on VOC, 10 on MOVi-C, and 5 on MOVi-E. On COCO, we disable this warm-up (i.e., $warmup_{gate} = 0$).

Gated Transformer Decoder. We use 4 Transformer blocks, each with 6 heads. Consistent with SPOT [20], we adopt its sequence-permutation strategy, using random permutations during training and all permutations at inference, but we do not use parallel decoding [37]. Given the selection mask M , we use a K/V gate $g_1 \in \{\varepsilon_1, 1\}^{K_{\max}}$ and a logits gate $g_2 \in \{\varepsilon_2, 1\}^{K_{\max}}$. We set $\varepsilon_1 = 10^{-6}$ and $\varepsilon_2 = 10^{-3}$ in all experiments.

Gated MLP Decoder. For the MLP-based decoder, we adopt the spatial broadcast mechanism from prior work [33, 38]. Each slot is expanded into N tokens aligned with image patches, and we add learnable positional embeddings to preserve spatial identity. The resulting tokens are processed independently by a four-layer MLP with ReLU activations and hidden size 2048. For each slot,

the decoder predicts both a reconstruction and an alpha map, with per-token output dimension $d_y + 1$, where d_y corresponds to reconstruction channels and the extra channel corresponds to alpha. The alpha map acts as an attention map that indicates the active regions of the slot, and the final reconstruction is obtained by aggregating per-slot reconstructions using the alpha maps as weights. In this MLP decoder, the predicted alpha maps are also used as the slot-attention masks. Given the selected-slot mask M , we apply hard gating at the alpha-logit level by filling logits of inactive slots with a large negative constant $C = -10^9$ before the softmax.

B Details on Object Property Prediction

We investigate the utility of our learned representations for downstream applications by predicting object properties on the COCO dataset.

Setup. We follow the experimental design of DINOSAUR [33]. We extract slot representations using our unsupervised model (frozen) and use them to supervise a predictor for object category and center-of-mass coordinates (x, y) .

Baselines. Existing OCL works do not follow a unified protocol for this task. Many papers do not evaluate on this task, or compare against only one baseline. Backbone choices and evaluation metrics also vary across works. To ensure fair comparison, we adopt a single shared setup and evaluate DINOSAUR [33], SPOT [20], AdaSlot [16], and MetaSlot [26] as baselines, training and testing all methods under the same protocol. These baselines are representative, covering both K -fixed and K -adaptive approaches, including established and recent methods, and spanning both MLP and Transformer decoders. Together, they provide a comprehensive assessment of property prediction performance.

Architecture. We evaluate both MLP and linear predictors. For the MLP predictor, we use a single-hidden-layer MLP with a hidden dimension of 256. It operates per slot, applying shared weights independently to each slot representation in parallel to produce a vector of property predictions.

Loss and Matching. We employ cross-entropy loss for discrete classification (80 COCO categories) and Mean Squared Error (MSE) for continuous coordinate regression. The total loss is the sum of these terms. To align the set of predictions with the set of ground truth objects, we utilize the Hungarian algorithm to minimize the total matching loss.

Data and Training Details. The training set consists of 10,000 images randomly sampled from COCO, with an additional 1,000 images reserved for validation. We train the predictor for 15,000 steps using the Adam optimizer with a learning rate of 1×10^{-3} and a batch size of 64. Evaluation is performed on the standard 5,000 validation images, excluding those containing no objects. We report the macro-averaged Top-1 and Top-5 accuracy for classification, and the coefficient of determination (R^2) for x and y coordinates. We evaluate the checkpoint with the lowest validation loss. For fair comparison, we do not apply any truncation to the ground truth.

C Further Analysis

Dataset distribution. We analyze the object-count distribution of the validation sets on COCO and MOVi-C/E, as shown in Fig. 6. The synthetic datasets MOVi-C/E exhibit a more concentrated distribution with a narrower range of object counts. This property provides a strong inductive bias for choosing a fixed K , which benefits K -fixed methods. In contrast, COCO shows a more long-tailed distribution with a much wider range of object counts, which weakens the bias of a fixed K . This dataset discrepancy helps explain why our method shows more pronounced gains on COCO. It also suggests that our model has greater potential in real-world settings where object cardinality varies substantially.

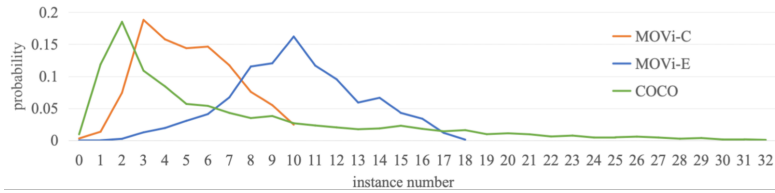


Fig. 6: Object number distribution on COCO and MOVi-C/E.

Slot count adaptivity. The foreground active slots distribution matches the GT object-count distribution (Fig. 7), confirming QASA’s adaptivity.

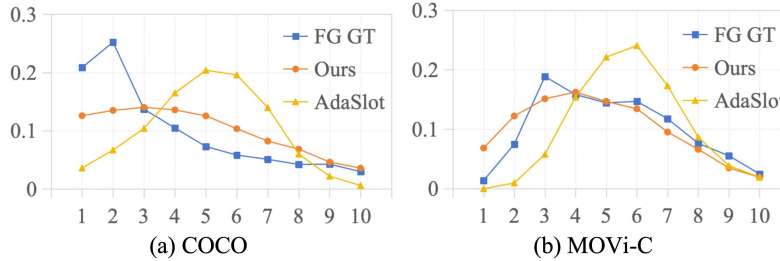


Fig. 7: Active slot count distribution.

Quality distribution analysis (Fig. 8). The selected and discarded Q distributions are clearly separated, with selected slots concentrated at high scores. Moreover, Q has little correlation with object size, indicating no size bias.

Additional time and space cost. Results in Tab. 7 show that our method is efficient. During training, the additional time and memory overhead is small and remains acceptable. During evaluation, we do not apply quality-guided gating. All slots compete for each token, and the extra runtime cost is negligible.

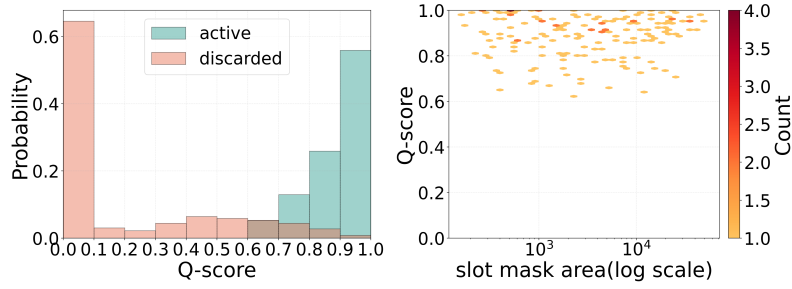


Fig. 8: Slot-Quality distribution on COCO.

Table 7: Cost comparison. We report time and per-GPU memory usage for 1-epoch COCO training on 4x RTX 4090s.

Method	Train time	Train space	Eval time
SPOT	5m 27s	15.8 GiB	30 s
Ours	5m 47s	17.2 GiB	32 s

D Additional Examples

We provide additional visualizations of QASA on COCO, PASCAL VOC, and MOVi-C/E, as shown in Figs. 9, 10, 11, and 12.

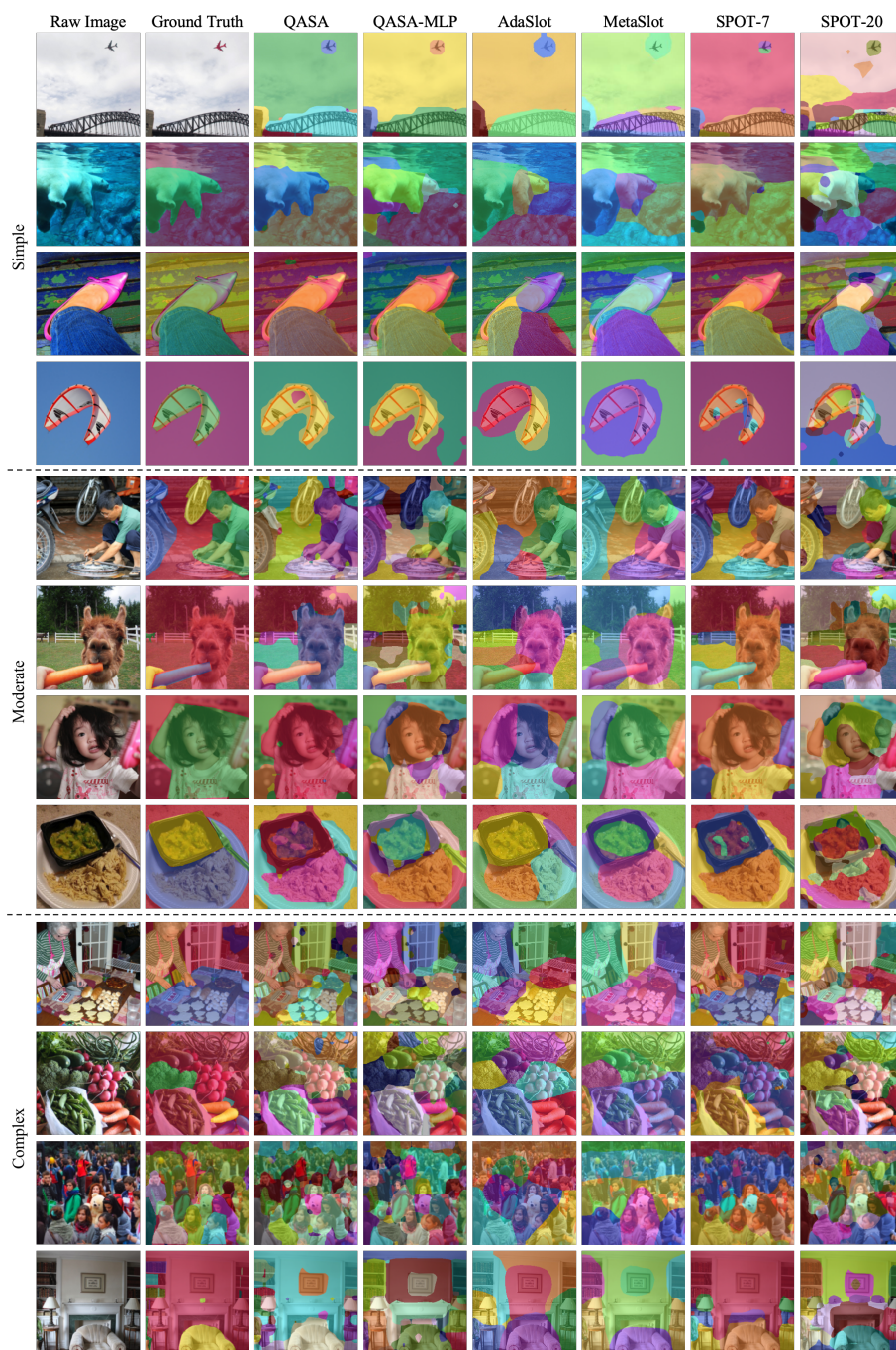


Fig. 9: Additional examples of QASA on COCO.

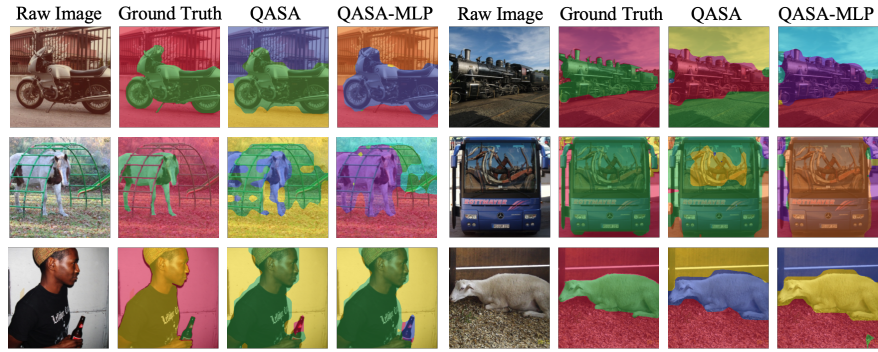


Fig. 10: Additional examples of QASA on PASCAL VOC.

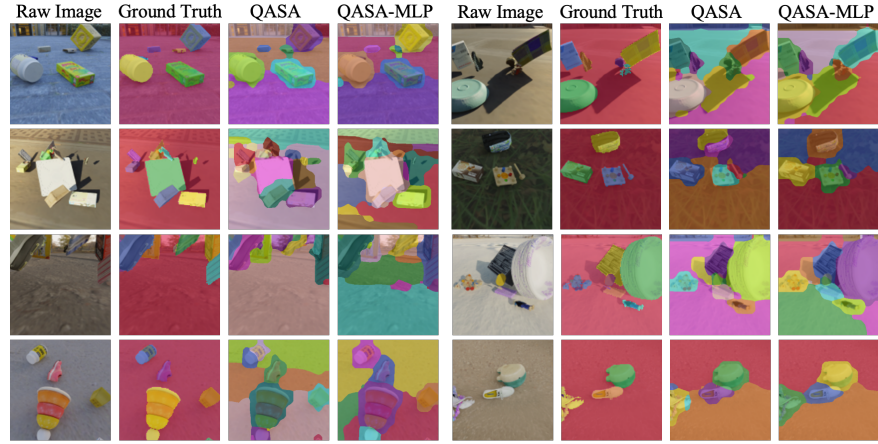


Fig. 11: Additional examples of QASA on MOVi-C.

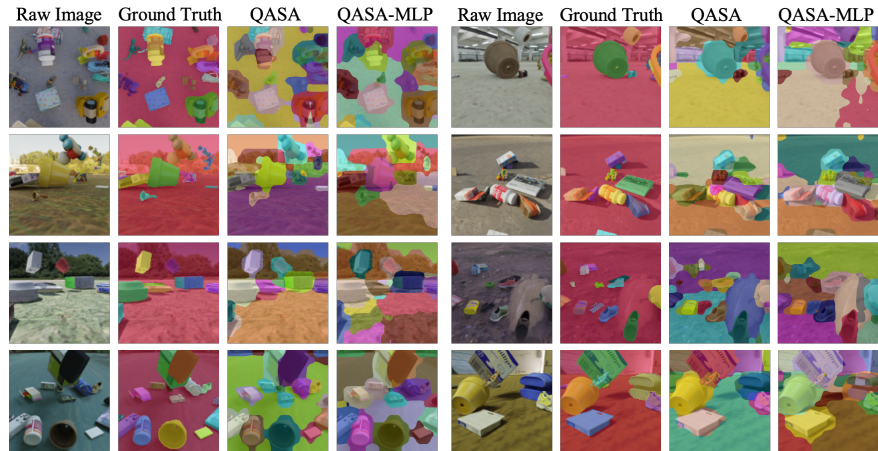


Fig. 12: Additional examples of QASA on MOVi-E.