SPOT: Self-Training with Patch-Order Permutation for Object-Centric Learning with Autoregressive Transformers

Ioannis Kakogeorgiou[1] Spyros Gidaris[2] Konstantinos Karantzalos[1] Nikos Komodakis[3,4,5]
[1]National Technical University of Athens [2]valeo.ai
[3]University of Crete [4]IACM-Forth [5]Archimedes/Athena RC

Figure 1. SPOT: Our novel framework enhances unsupervised object-centric learning in slot-based autoencoders using self-training and sequence permutations in the transformer decoder. It improves object-specific slot generation, excelling in complex real-world images.

Abstract

Unsupervised object-centric learning aims to decompose scenes into interpretable object entities, termed slots. Slot-based auto-encoders stand out as a prominent method for this task. Within them, crucial aspects include guiding the encoder to generate object-specific slots and ensuring the decoder utilizes them during reconstruction. This work introduces two novel techniques, (i) an attention-based self-training approach, which distills superior slot-based attention masks from the decoder to the encoder, enhancing object segmentation, and (ii) an innovative patch-order permutation strategy for autoregressive transformers that strengthens the role of slot vectors in reconstruction. The effectiveness of these strategies is showcased experimentally. The combined approach significantly surpasses prior slot-based autoencoder methods in unsupervised object segmentation, especially with complex real-world images. We provide the implementation code at https://github.com/gkakogeorgiou/spot.

1. Introduction

Decomposing a scene into separate objects is crucial for AI progress. While current AI methods often use labeled segmentations or diverse signals like text, video, motion, or depth, humans can typically achieve scene decomposition with visual cues alone. Unsupervised object-centric learning, inspired by human abilities and utilizing abundant unlabeled image data, seeks to represent a scene as a composition of distinct objects using only visual information.

Auto-encoding-based frameworks stand at the forefront of object-centric learning approaches [6, 14, 15, 17, 32, 34, 36, 37, 44]. Here, the emergence of object-centric representations is enabled due to architectural inductive biases such as the use of bottleneck modules that force the network to prioritize the encoding of salient object features. Their simple design, combined with the ability to operate unsupervised, makes them a standout choice. One notable advance in this regard involves approaches employing 'slot'-structured bottlenecks [34, 41, 44]. These auto-encoders, characterized by their slot-based architecture, consist of two core components. First is the encoder, responsible for transforming input data into a set of latent vectors referred to as 'slots', each intended to represent an individual object within an image. The second is the decoder, burdened with the challenging task of reconstructing the input based on information derived from the extracted slots, guiding the learning of object-centric representations.

To advance this paradigm, our work introduces SPOT – a dual-stage strategy designed to elevate object-centric learning so as to more effectively handle complex real-world images, which is one of the pivotal challenges in this domain [65]. SPOT aims to refine both components of slot-

based auto-encoders, enhancing both the encoder's precision in generating object-specific 'slots' and the decoder's ability to utilize these slots during reconstruction. To that end, it makes two key technical advancements.

Improving slot generation through self-training. The encoding of object-specific information into the slots is achieved through an iterative attention mechanism that forces slots to compete over image patches. This competition leads to the generation of slot-based attention masks that indicate the association of each image patch with a specific slot [34, 41]. Besides the encoder, slot-based attention masks are also generated at the decoder side. For autoregressive transformer decoders [41, 44], these are produced by the cross-attention module between each output patch and the extracted slots. Empirically, we observe that masks produced during decoding demonstrate superior object decomposition, i.e., better object segmentation, compared to those from encoding [41]. Building on this insight, we propose a self-training scheme that distills slot-based attention masks from the decoder to the encoder, thereby enhancing the object segmentation information captured by the slots.

Enhanced autoregressive decoders with sequence permutations. Object-centric models commonly utilize weak slot-wise MLP decoders [54]. Drawing inspiration from large language models [5], Singh et al [44] propose the adoption of more expressive autoregressive transformer decoders in the context of object-centric learning [44, 45], surpassing the performance of MLP-based counterparts. Moreover, multiple studies have underscored the importance of increasing decoder capacity to effectively apply object-centric learning to complex scenes [8, 22, 24, 41, 46, 59, 60]. However, autoregressive transformer models may face overfitting challenges, particularly when accustomed to teacher-forcing training [58], relying excessively on past ground-truth tokens. This tendency leads them to neglect slot vectors, offering weaker and less robust supervisory signals for their learning. To mitigate this, we propose the introduction of sequence permutations, altering the autoregressive transformer's prediction order. This modification amplifies the role of slot vectors in the reconstruction process, resulting in improved object-centric representations.

In summary, our contributions are threefold: **(1)** We enhance unsupervised object-centric learning in slot-based autoencoders by introducing self-training. This involves distilling slot-attention masks from the initially trained teacher model's decoder to the slot-attention module of the student model, thereby improving precision in generating object-specific slots. **(2)** We amplify the role of slot vectors in the reconstruction process by introducing sequence permutations that alter the prediction order of the autoregressive transformer decoder. This modification leads to a more robust supervisory signal for object-centric learning. **(3)** Empirical evidence demonstrates the synergistic effectiveness of the above strategies. The combined approach forms the SPOT framework, significantly outperforming prior slot-based autoencoder methods in unsupervised object segmentation, particularly with complex real-world images.

2. Related work

Unsupervised object-centric learning aims to decompose multi-object scenes into meaningful object entities. Previous studies utilize auto-encoding frameworks [6, 14, 15, 17, 32, 34, 36, 37, 44], with slot-attention bottlenecks [34] emerging as a prominent paradigm. Despite their notability, early object-centric models struggle with complex (real-world) scenes [25, 65]. To address this, some methods focus on improving precision and stability of the encoder's slot-attention module [8, 23, 26, 29, 47], employing bi-level optimization [8, 23], architectural modifications [26, 29], or regularization losses [47]. Another line of work focuses on designing a decoder that supports good decomposition [24, 44, 60], where SLATE [44] utilizes an autoregressive transformer decoder, and others propose diffusion-based methods [24, 60]. DINOSAUR [41] uses self-supervised pre-trained features as reconstruction targets, proving more effective in complex scenes. Additionally, some works extend slot-based auto-encoders to exploit video data [1, 28, 45, 49, 56, 67], motion [3, 4], depth [13], or text [61]. Our work also aims at improving the effectiveness of object centric learning in handling complex real-world data but does so using static images and is thus complementary to the above approaches.

Furthermore, object-centric learning has been explored through contrastive frameworks [2, 20, 21, 35, 57], primarily for pre-training representations.

Autoregressive transformer decoders [52] excel in natural language processing [5, 48, 66] and have recently proven effective in computer vision, exemplified by models like iGPT [9]. Notably, SLATE [44] and STEVE [45] have applied them to object-centric learning, highlighting their efficacy in handling intricate visual scenes. However, despite their success, autoregressive transformers face training stability challenges in object-centric learning, potentially due to their high capacity [41]. They tend to overly depend on past ground-truth tokens, neglecting input from the encoder, as observed in pretraining image encoders with image caption tasks [50]. In our analysis, we validate this issue in the context of object-centric learning and propose a simple patch-order permutation strategy, which is very easy to integrate with existing autoregressive transformer models.

Self-training approaches refine models using their own predictions on unlabeled data, enhancing the training set. They are widely used in semi-supervised learning for tasks like image classification [55, 62], semantic segmentation [12, 63, 68], and object detection [39, 51, 64], as

well as in unsupervised domain adaptation [33, 38, 69] and unsupervised localization [42, 43, 53]. Additionally, they enhance performance in supervised classification. This work introduces self-training to the domain of unsupervised object-centric learning, demonstrating its notable effectiveness in this specific context.

3. Method

Slot-based auto-encoders. Following the common practice in object-centric learning, we use a slot-based auto-encoding framework [34]. This framework's encoder consists of two modules: an image encoder extracting n patch-wise features from image X and a slot-attention module that groups these features into $k < n$ latent vectors $U = (\mathbf{u}_1; \ldots; \mathbf{u}_k) \in \mathbb{R}^{k \times d_u}$ referred to as 'slots', each representing an object in the image. The decoder aims to reconstruct a target signal, such as the original image X, from these slot vectors. Here, following DINOSAUR [41], we employ self-supervised pre-trained feature encoders (e.g., DINO [7]) for instantiating the image encoder as well as extracting the reconstruction targets. Defining $Y, \hat{Y} \in \mathbb{R}^{n \times d_y}$ as the target features and the predicted reconstructions, respectively, the model is trained by minimizing the reconstruction loss:

$$L_{\text{REC}} = \frac{1}{n \cdot d_y} ||Y - \hat{Y}||_2^2. \tag{1}$$

Defining the reconstruction task with high-level features provides a valuable training signal for learning object-centric representations from real-world data [41].

Auto-regressive transformer decoders. A crucial element in slot-based auto-encoders is the decoder's design [24, 44, 45, 60]. Here, we employ an autoregressive transformer [52] as our decoder. Autoregressive transformers predict the feature $\hat{\mathbf{y}}_i$ at position i based on prior target features $Y_{<i}$ and slots U. This prediction is jointly performed for all token positions using teacher-forcing[1]:

$$\hat{Y} = \textsc{Decoder}(Y_<; U), \tag{2}$$

where $Y_< \in \mathbb{R}^{n \times d_y}$ is the decoder's input, consisting of target features Y right-shifted by one position (excluding the last token), with a learnable Beginning-Of-Sentence ([BOS]) token prepended: $Y_< = (\mathbf{y}_{[\text{BOS}]}; \mathbf{y}_1; \ldots; \mathbf{y}_{n-1})$. The decoder is composed of a sequence of transformer blocks [52], each incorporating a causal self-attention layer (to avoid attending to 'future' tokens), a patch-to-slot cross-attention layer allowing to utilize information from slot vectors U, and a feed-forward layer (see Fig. 5).

Objective. The primary objective of slot-based auto-encoders is to decompose the input image into its individual objects. This is typically evaluated by examining masks linked to each slot, indicating the association of each image patch with a specific slot / object. These masks, called slot-attention masks, can be derived from either the slot-attention module or the employed decoder.

In this work, we present SPOT, a novel two-stage training method that enhances object-centric learning on real-world data through self-training (Sec. 3.1) and sequence permutation in the autoregressive decoder (Sec. 3.2).

3.1. Self-training via slot-attention distillation

We use the matrix $A \in [0, 1]^{n \times k}$ to denote slot attention masks. Each row of A is a probability distribution in a k-dimensional simplex, indicating the assignment of each image patch to k slots. As previously outlined, we can derive slot-attention masks from two places:

Slot-attention module. This module employs an iterative attention-based approach that begins with the initial query slot vectors $\tilde{U}$ to produce the output slot vectors U. At its core, the module incorporates a modified slot-to-patch cross-attention layer. The slot attention module's matrix A is derived from this layer, specifically from the last iteration's cross-attention map

$$A_{\text{SLOT}} = \textsc{Softmax}\left(\frac{Q_E K_E^\top}{\sqrt{d_p}}\right)^\top \in \mathbb{R}^{n \times k}, \tag{3}$$

where $K_E \in \mathbb{R}^{n \times d_p}$ are keys computed from the patch-wise features extracted by the image encoder from the input image X, and $Q_E \in \mathbb{R}^{k \times d_p}$ are queries computed from slot vectors of the previous iteration. The $\textsc{Softmax}$ is applied along the slots dimension to enforce competition. The initial slots vectors $\tilde{U}$ are either independently sampled from a Gaussian distribution [34] or are trainable parameters [23].

Decoder module. As mentioned earlier, transformer-based decoders incorporate patch-to-slot cross-attention layers for leveraging information from slot vectors U. Here the attention masks A, denoted as $A_{\text{DEC}} \in \mathbb{R}^{n \times k}$, are computed as the average (across H heads) of patch-to-slot cross-attention maps from the final transformer layer:

$$A_{\text{DEC}} = \frac{1}{H} \sum_{j=1}^{H} \textsc{Softmax}\left(\frac{Q_j K_j^\top}{\sqrt{d_h}}\right), \tag{4}$$

where $K_j \in \mathbb{R}^{k \times d_h}$ are keys computed from the slots U, and $Q_j \in \mathbb{R}^{n \times d_h}$ are queries computed from the transformed (from previous layers) decoder input $Y_<$. The $\textsc{Softmax}$ is applied along the dimension of slots.

In our empirical analysis of the attention masks from these two modules (see Tab. 1 entry (a) in Sec. 4), we note that the decoder masks exhibit superior performance

[1] While joint prediction is typically a training-only practice, in our context, is extended to testing due to the availability of target features.

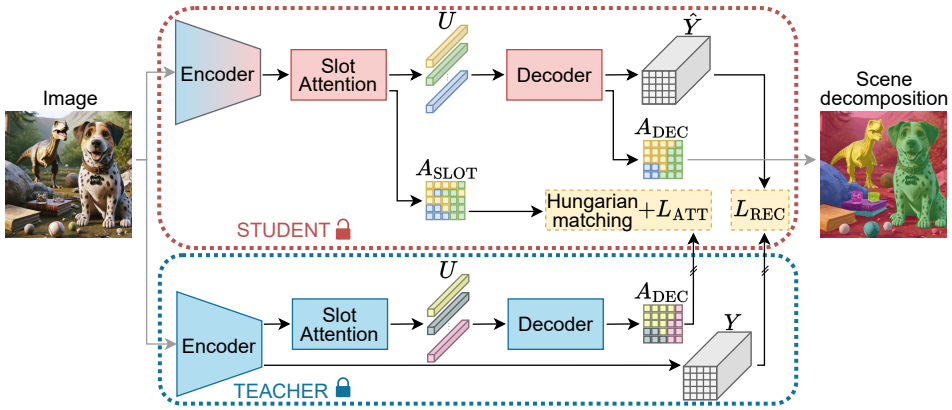


Figure 2. *Enhancing unsupervised object-centric learning via self-training*. Our two-stage approach starts with exclusive training in the initial stage (not depicted) using the reconstruction loss L_{REC}. In the following stage, shown here, a teacher-student framework is applied. The teacher model, trained in the first stage, guides the student model with an additional loss L_{ATT}, distilling attention masks A_{DEC} from the teacher's decoder to the slot-attention masks A_{SLOT} in the student's encoder.

in grouping patches into object-centric slots. This observation aligns with the findings reported by Seitzer et al. [41].

Two-stage training with slot-attention distillation. Motivated by this observation, we propose to improve the slot-attention module with a self-training scheme depicted in Fig. 2. Our training approach involves two stages. In the initial stage, the model is trained exclusively using the L_{REC} loss. In the second stage, we employ a teacher-student framework. The pre-trained model serves as the teacher, guiding the training of a new model (referred to as the student) with an additional loss L_{ATT} that distills attention masks A_{DEC} from the teacher's decoder, denoted as A_T, to the attentions masks A_{SLOT} of the student's encoder, denoted as A_S.

This distillation enhances the grouping capability of the student's slot-attention module, resulting in improved slot representations.

Slot-attention distillation loss L_{ATT}. To distill the teacher's attention masks A_T to the student, we first convert them from soft to hard-assignment masks by applying row-wise to A_T the `argmax` and `one-hot` operators:

$$A'_T = \texttt{one-hot}(\texttt{argmax}(A_T)) \in \{0,1\}^{n\times k}. \quad (5)$$

Then, we use Hungarian matching [30] to map the k slots in the teacher's masks A'_T with the k slots in the student's masks A_S, using the IoU between the masks as cost function. The matching process results in the A''_T masks. Finally, we apply the cross-entropy loss between A''_T and A_S:

$$L_{\text{ATT}} = \frac{1}{n}\langle A''_T, \log(A_S)\rangle_F, \quad (6)$$

where $\langle\cdot,\cdot\rangle_F$ denotes the Frobenius inner product.

The total training loss in the second training stage is

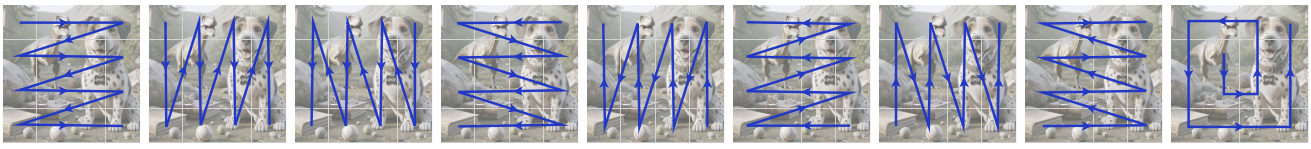
$$L = L_{\text{REC}} + \lambda L_{\text{ATT}}, \quad (7)$$

where λ is the loss weight of the distillation objective.

Stabilizing image encoder training with L_{ATT}. We employ self-supervised pre-trained Vision Transformer [11] (ViT) models as image encoders, empirically shown to achieve superior object-centric learning results [41]. During the first training stage, it is crucial, as emphasized in [41], to keep the ViT image encoder frozen for training stability to achieve good results. Our analysis in Sec. 4.2 reveals an additional advantage of the self-training loss: serving as a stabilization factor, it facilitates fine-tuning of ViT in the second training stage, maximizing its learning capacity. We attribute this stabilizing effect to the self-training loss explicitly guiding the encoder to generate object-specific slot masks, in contrast to the reconstruction loss L_{REC}.

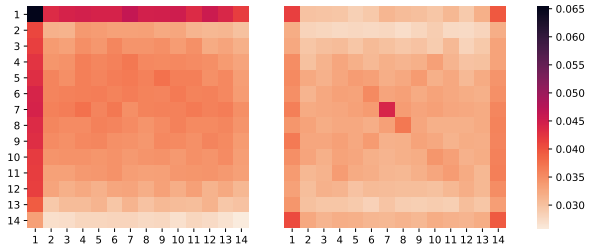
3.2. Autoregressive transformer decoder with sequence permutations

Autoregressive transformer decoders outperform simpler MLPs with spatial-broadcasting decoders [41], thanks to their effectiveness in ensuring global consistency in predictions. However, their high representational capacity, as discussed in [50], may limit the encoder's effective learning by diminishing the strength of the supervisory signal backpropagated to it. This is manifested as follows: initial patch tokens heavily rely on slot vector information due to limited context from 'past' tokens. As decoding progresses, dependence on features of earlier tokens grows, reducing the significance of slot vector information. Consequently, later tokens provide a weaker supervisory signal for learning slot vectors in the encoder. Fig. 4(a) illustrates this variability in signal strength based on token location. In this figure, the magnitude of the L_{REC} loss gradient with respect to slots (averaged over data samples) is presented, highlighting that tokens in the first row and column contribute significantly higher magnitude gradients during backpropagation to slots compared to their later counterparts.



(a) Left to right Top to bottom (b) Top to bottom Left to right (c) Top to bottom Right to left (d) Right to left Top to bottom (e) Bottom to top Right to left (f) Right to left Bottom to top (g) Bottom to top Left to right (h) Left to right Bottom to top (i) Spiral

Figure 3. *Sequence permutations in SPOT*. The sequence of patches used for autoregressive-based decoder predictions.



(a) Decoder w/o permutations grads. of default perm. (b) Decoder with permutations grads. of random perm.

Figure 4. L_1 gradients norms for each patch's reconstruction loss with respect to the decoder's input slots (aggregated across all the slots, four decoder blocks, and the entire COCO validation set). Subplots show gradients with: (a) default permutation and (b) randomly sampled sequence permutations.

Autoregressive decoding with permuted sequences. To tackle this issue, we propose introducing sequence permutations to alter the order in which the autoregressive transformer predicts, based on a predefined set of permutations. In essence, employing a sequence permutation requires the transformer to make predictions not only in the traditional left-to-right and top-to-bottom patch-order (as depicted in Fig. 3(a)), but also with additional patch-orderings shown in Fig. 3(b-i). The set of sequence permutations comprises eight ordering variations, each associated with a distinct starting patch and direction (horizontal/vertical), complemented by an additional spiral permutation originating from a patch at the center of the image.

During each training step, we randomly select one sequence permutation from the set to guide predictions by the autoregressive transformer. This introduces variability in token positions, as tokens later in the sequence in the default order may now occupy initial positions in the permuted sequence. Consequently, these tokens must rely more on slot information, offering stronger supervisory signal to the encoder after the permutation. This improvement is evident in Fig. 4(b), where the magnitudes of the gradients w.r.t. the slots of the model that use sequence permutations in the decoder are less sensitive w.r.t. the token location they are coming from (as opposed to the model without permutations Fig. 4(a)). Thus, the model trained with sequence permutations benefits from a more uniform reliance on the slots across all token positions. Further discussion on the autoregressive transformer is in the supplementary material.

We illustrate the workflow of our autoregressive decoder with sequence permutations in Fig. 5. To enable the decoder

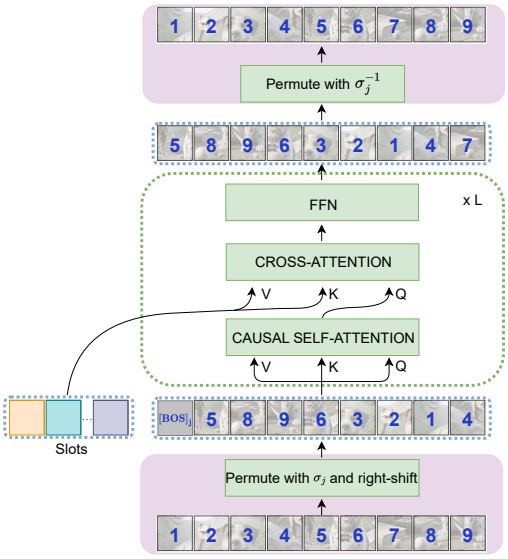


Figure 5. *Autoregressive (AR) decoding via sequence permutations*. Violet boxes indicate differences from typical AR decoder.

to recognize the current sequence ordering and thus what is the next-patch that it must predict, a different [BOS] token is used for each different permutation. More formally, we define 9 different sequence permutations $\sigma_j : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$, $j = 1, \ldots, 9$. For each permutation j, the permuted input to the autoregressive transformer is

$$Y_{<\sigma_j} = (\mathbf{y}_{[\mathrm{BOS}]_j}; \mathbf{y}_{\sigma_j(1)}; \ldots; \mathbf{y}_{\sigma_j(n-1)}), \quad (8)$$

where $\mathbf{y}_{[\mathrm{BOS}]_j}$ is the [BOS] token for the j permutation. The decoder produces predictions $\hat{Y}_{\sigma_j}$ for this j permutation

$$\hat{Y}_{\sigma_j} = (\mathbf{y}_{\sigma_j(1)}; \ldots; \mathbf{y}_{\sigma_j(n)}) = \textsc{Decoder}(Y_{<\sigma_j}; U), \quad (9)$$

which are then re-ordered to $\hat{Y}$ using the inverse permutation σ_j^{-1}, followed by applying the L_{REC} loss.

Therefore, in our decoder, the sole modifications involve permuting the inputs and outputs and introducing multiple [BOS] tokens (one per permutation) instead of a single one. Other than that, the core architecture of the decoder remains unchanged. During training, a single randomly sampled permutation per training step is employed, maintaining the training cost. At inference, we simply use the default permutation (Fig. 3(a)) preserving the same inference cost. Alternatively, computing attention masks A_{DEC} for each per-

	SP	ST	ENS	Epochs	Decoder				Slot Attention				Max(Dec, Slot Att)			
					MBO^i	MBO^c	mIoU	FG-ARI	MBO^i	MBO^c	mIoU	FG-ARI	MBO^i	MBO^c	mIoU	FG-ARI
(a)				50	32.0$_{\pm 0.1}$	41.4$_{\pm 0.3}$	30.0$_{\pm 0.1}$	32.3$_{\pm 0.6}$	30.0$_{\pm 0.2}$	38.9$_{\pm 0.4}$	28.1$_{\pm 0.2}$	29.5$_{\pm 0.6}$	32.0$_{\pm 0.1}$	41.4$_{\pm 0.3}$	30.0$_{\pm 0.1}$	32.3$_{\pm 0.6}$
(b)	✓			50	32.7$_{\pm 0.2}$	40.9$_{\pm 0.5}$	30.8$_{\pm 0.1}$	35.6$_{\pm 0.5}$	31.1$_{\pm 0.1}$	38.9$_{\pm 0.3}$	29.4$_{\pm 0.1}$	33.4$_{\pm 0.4}$	32.7$_{\pm 0.2}$	40.9$_{\pm 0.5}$	30.8$_{\pm 0.1}$	35.6$_{\pm 0.5}$
(c)	✓		✓	50	32.9$_{\pm 0.2}$	41.2$_{\pm 0.5}$	31.0$_{\pm 0.1}$	36.0$_{\pm 0.5}$	31.1$_{\pm 0.1}$	38.9$_{\pm 0.3}$	29.4$_{\pm 0.1}$	33.4$_{\pm 0.4}$	32.9$_{\pm 0.2}$	41.2$_{\pm 0.5}$	31.0$_{\pm 0.1}$	36.0$_{\pm 0.5}$
(d)				100	32.3$_{\pm 0.3}$	42.1$_{\pm 0.2}$	30.2$_{\pm 0.3}$	31.8$_{\pm 0.9}$	30.5$_{\pm 0.2}$	39.8$_{\pm 0.1}$	28.5$_{\pm 0.3}$	30.0$_{\pm 0.7}$	32.3$_{\pm 0.3}$	42.1$_{\pm 0.2}$	30.2$_{\pm 0.3}$	31.8$_{\pm 0.9}$
(e)		✓		50+50	30.1$_{\pm 0.5}$	38.2$_{\pm 1.0}$	28.3$_{\pm 0.4}$	22.5$_{\pm 1.6}$	33.2$_{\pm 0.1}$	43.6$_{\pm 0.2}$	31.1$_{\pm 0.0}$	34.4$_{\pm 0.5}$	33.2$_{\pm 0.1}$	43.6$_{\pm 0.2}$	31.1$_{\pm 0.0}$	34.4$_{\pm 0.5}$
(f)	✓	✓		50+50	**34.7**$_{\pm 0.1}$	**44.3**$_{\pm 0.3}$	**32.7**$_{\pm 0.1}$	**36.6**$_{\pm 0.3}$	**33.7**$_{\pm 0.1}$	**43.1**$_{\pm 0.4}$	**31.8**$_{\pm 0.1}$	**37.8**$_{\pm 0.5}$	**34.7**$_{\pm 0.1}$	**44.3**$_{\pm 0.3}$	**32.7**$_{\pm 0.1}$	**37.8**$_{\pm 0.5}$
(g)	✓	✓	✓	50+50	**35.0**$_{\pm 0.1}$	**44.7**$_{\pm 0.3}$	**33.0**$_{\pm 0.1}$	**37.0**$_{\pm 0.2}$	**33.7**$_{\pm 0.1}$	**43.1**$_{\pm 0.4}$	**31.8**$_{\pm 0.1}$	**37.8**$_{\pm 0.5}$	**35.0**$_{\pm 0.1}$	**44.7**$_{\pm 0.3}$	**33.0**$_{\pm 0.1}$	**37.8**$_{\pm 0.5}$

Table 1. *Ablation study on COCO*. Results for slot masks generated by Decoder, Slot Attention, and their max (Max(Dec, Slot Att)) using mean and std over 3 seeds. SP: sequence permutation, ST: self-training, ENS: ensembling of nine permutations at test-time.

mutation and averaging them (i.e., test-time ensembling using the permutations), as explored in Sec. 4, yields a slight performance improvement.

3.3. Incorporating permutations into self-training

We employ sequence permutation to enhance the transformer decoder in both training stages. This improves the 1st-stage decoder, which thus serves as a better teacher for the 2nd self-training stage. In the 2nd stage, beyond the student, we also incorporate random sequence permutations in the teacher's decoder when generating the target slot-attention masks for the self-training loss.

For the 1st-stage training, we use as initial slots independently sampled vectors from a Gaussian distribution, following [34, 41]. For the 2nd stage, we initialize slots as learnable vectors employing bi-level optimization [8, 23].

4. Experiments

4.1. Setup

Datasets. We utilized the MS COCO 2017 [31] dataset for its diverse collection of real-world images, each featuring multiple co-occurring objects. This dataset poses a significant challenge for object-centric learning models due to the complexity of the scenes. We also considered the PASCAL VOC 2012 [16] dataset, which comprises images often containing a single or just a few salient objects, offering a comparatively more straightforward evaluation. Furthermore, we used the synthetic datasets MOVi-C and MOVi-E [18], which contain approximately 1000 realistic 3D-scanned objects. MOVi-C includes scenes with 3-10 objects, whereas MOVi-E contains scenes with 11-23 objects per scene. Although MOVi-C/E are originally video-based, we adapt them by selecting random frames following [41].

Metrics To assess object-centric learning, we use Mean Best Overlap at the instance (MBO^i) and class (MBO^c) levels, Foreground Adjusted Rand index (FG-ARI), and mean Intersection over Union (mIoU). MBO^i identifies the best overlap for each ground truth mask, while mIoU employs Hungarian matching for a one-to-one correspondence between predicted and ground truth segments. Our main focus is on MBO and mIoU metrics because they consider background pixels, comprehensively evaluating how closely masks fit around objects. In contrast, FG-ARI, a cluster similarity metric, exclusively focuses on foreground pixels, potentially giving a misleading impression of segmentation quality by ignoring the fidelity of predicted masks while also promoting over-segmentation. While we report FG-ARI, it is not our primary focus, aligning with concerns raised in other studies [14, 25, 40, 60].

Implementation details. We employ the Adam optimizer [27] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, no weight decay, and a batch size of 64. For each training stage on COCO and PASCAL, we use 50 and 560 training epochs, respectively. For MOVi-C/E experiments, we use 65 and 30 epochs for the first and second stages, respectively.

We implement our SPOT models using ViT-B/16 [11] for the encoder (by default initialized with DINO [7]) and 4 transformer layers in our decoder. Unless stated otherwise, loss weight λ is 0.005. Following [41], on COCO, PASCAL, MOVi-C, and MOVi-E we use 7, 6, 11, and 24 slots, respectively. All models are trained on a single GPU with 24 Gbytes. We provide learning rate schedules and further implementation details in the supplementary material.

4.2. Analysis

In this section, we analyze our approach, emphasizing the impact of self-training and sequence-permutation in the autoregressive decoder, along with related design choices. Our primary experimentation focuses on the challenging COCO dataset, complemented by evaluations on MOVi-C.

(A) Sequence-permutation impact. Tab. 1 evaluates the influence of sequence-permutation with and without self-training. Without self-training, sequence permutations (Tab. 1 (a)→(b)) enhances instance-specific metrics (MBO^i, mIoU, FG-ARI) on both the decoder and slot-attention. The class-specific metric MBO^c remains stable (within std).

When self-training is enabled, the significance of sequence permutations becomes even more evident (Tab. 1 (e)→(f)). Omitting them results in a notable performance

(a) Loss Weight λ					(b) Permutations		(c) Enc. fine-tuning	
0	0.002	0.005	0.01	0.02	Default	Random	✗	✓
30.7$\pm$2.2	34.6$\pm$0.3	**34.7$\pm$0.1**	34.1$\pm$0.5	33.4$\pm$1.5	34.6$\pm$0.2	**34.7$\pm$0.1**	32.4$\pm$0.2	**34.7$\pm$0.1**

Table 2. *Analysis of self-training hyper-parameters on COCO*. Results are the mean and standard deviation of the decoder's MBO^i over 3 seeds. Section (a) studies the impact of the loss weight λ, section (b) the impact of using random or the default permutation for generating the target mask A_T with the teacher model, and section (c) the impact of fine-tuning the image encoder during the self-training stage.

Decoder	MBO^i	MIoU	FG-ARI
Transformer	32.0	30.0	32.3
Transformer w/ PA	27.8	26.5	35.3
Transformer w/ SP	**32.7**	**30.8**	**35.6**

Table 3. *Autoregressive decoder comparison on COCO*. Evaluation metrics for slot masks generated by the autoregressive transformer decoder trained conventionally (Transformer), with sequence permutation during training (Transformer w/ SP), or parallel prediction for 25% of training iterations (Transformer w/ PA), as discussed in [50]. No self-training is employed.

drop across all metrics, especially impacting decoder-specific ones (Decoder cols.). Notably, without sequence-permutations, self-training boosts the slot-attention module (Slot Attention cols.) but hampers decoder performance (Decoder cols.). This discrepancy highlights the autoregressive decoder's susceptibility to neglecting slot input during reconstruction tasks. An overfitting behavior of this type may be attributed to the accelerated learning dynamics of slot vectors during training, caused by the self-training loss L_{ATT} applied to them, making it challenging for the autoregressive decoder to effectively leverage them. Sequence permutations play a crucial role in compelling the decoder to prioritize slots, underscoring their importance in autoregressive decoders for slot-centric learning.

Test-time ensembling of permutations. Comparing models (b) to (c) and (f) to (g) in Tab. 1, we observe a slight performance gain by using test-time ensembling of permutations. However, this comes at the expense of increased inference time. Nonetheless, our method demonstrates very robust performance even without test-time ensembling, emphasizing that the core influence of sequence-permutation lies in enhancing the effectiveness and stability of training slot-based auto-encoders with autoregressive decoders.

Comparison to other AR training approaches. Motivated by CapPa [50], we explored employing a training procedure that switches between autoregressive and parallel non-autoregressive decoding[2], with the latter being used at 25% of training steps, as an alternative to sequence permutations for enhancing the autoregressive decoder. Our findings, outlined in Tab. 3, reveal that, in the context of object-centric learning, parallel decoding during training does not help, unlike our sequence permutation approach.

Last, the positive impact of sequence permutations is also demonstrated in MOVi-C (results in supplementary).

[2]Here, given as input position embeddings, the transformer predicts at all positions in parallel without causal masking in self-attention.

Decoder	ST	MBO^i	MBO^c	MIoU	FG-ARI
MLP	✗	26.7	30.3	25.6	38.7
	✓	**28.4**	**32.4**	**27.0**	**42.5**

Table 4. *Self-training with MLP decoder on COCO*. Evaluation metrics for slot masks generated by the decoder. ST: self-training.

(B) Self-training impact. Comparing entries (f) to (b) in Tab. 1, we see that self-training yields significant gains. It also helps the max performance (columns Max(Dec, Slot Att)) in the case of not using sequence permutation.

Autoregressive and MLP-based decoders. As discussed earlier, when it comes to autoregressive decoder performance, optimal results are obtained when self-training is paired with sequence-permutation, which is crucial for preventing the decoder from ignoring slot information and overfitting. Notably, our self-distillation scheme also extends beyond autoregressive decoders. To showcase this, we apply it with MLP-based decoders where it brings significant performance improvements as demonstrated in Tab. 4.

In Tab. 2, section (a) illustrates the sensitivity of the self-distillation scheme to the loss weight λ, with λ=0.005 producing optimal results. Performance remains relatively stable for loss weights around this value.

In section (b) of Tab. 2, we investigate the impact of extracting the teacher's slot-mask A_T with a standard or random permutation during the self-distillation scheme. This design choice does not appear to have a significant impact, with a random permutation being our default choice.

Finally, in section (c) of Tab. 2, we highlight an additional benefit derived from the self-training stage, particularly in the successful fine-tuning of the image encoder during that stage. We stress that, without the self-training loss, allowing the image encoder to be fine-tuned becomes unstable, yielding subpar results (Tab. 2 (a) for $\lambda = 0$ in L_{ATT}). This observation aligns with Seitzer et al.'s findings [41]. The self-training loss acts as a stabilizing factor, enabling further utilization of the pre-trained image encoder.

4.3. Comparison with object-centric methods

Method comparison. In Tab. 5, we compare our object-centric learning method SPOT with prior approaches across the MOVi-C, MOVi-E, PASCAL, and COCO datasets using MBO^i and, when applicable, MBO^c metrics. More detailed results, including the FG-ARI metric are provided in

METHOD	COCO		PASCAL		MOVI-C		MOVI-E	
	MBO^i	MBO^c	MBO^i	MBO^c	MBO^i	MIOU	MBO^i	MIOU
SA [34]†	17.2	19.2	24.6	24.9	26.2$_{\pm 1.0}$	-	24.0$_{\pm 1.2}$	-
SLASH [26]	-	-	-	-	-	27.7$_{\pm 5.9}$	-	-
SLATE [44]†	29.1	33.6	35.9	41.5	39.4$_{\pm 0.8}$	37.8$_{\pm 0.7}$	30.2$_{\pm 1.7}$	28.6$_{\pm 1.7}$
CAE [36]†	-	-	32.9$_{\pm 0.9}$	37.4$_{\pm 1.0}$	-	-	-	-
DINOSAUR [41]	32.3$_{\pm 0.4}$	38.8$_{\pm 0.4}$	44.0$_{\pm 1.9}$	51.2$_{\pm 1.9}$	42.4	-	-	-
DINOSAUR-MLP [41]	27.7$_{\pm 0.2}$	30.9$_{\pm 0.2}$	39.5$_{\pm 0.1}$	40.9$_{\pm 0.1}$	39.1$_{\pm 0.2}$	-	35.5$_{\pm 0.2}$	-
Rotating Features [37]	-	-	40.7$_{\pm 0.1}$	46.0$_{\pm 0.1}$	-	-	-	-
SlotDiffusion [60]	31.0	35.0	**50.4**	55.3	-	-	30.2	30.2
(Stable-)LSD [24]	30.4	-	-	-	45.6$_{\pm 0.8}$	44.2$_{\pm 0.9}$	39.0$_{\pm 0.5}$	37.6$_{\pm 0.5}$
SPOT w/o ENS (ours)	34.7$_{\pm 0.1}$	44.3$_{\pm 0.3}$	48.1$_{\pm 0.4}$	55.3$_{\pm 0.4}$	47.0$_{\pm 1.2}$	46.4$_{\pm 1.2}$	39.9$_{\pm 1.1}$	39.0$_{\pm 1.1}$
SPOT w/ ENS (ours)	**35.0**$_{\pm 0.1}$	**44.7**$_{\pm 0.3}$	48.3$_{\pm 0.4}$	**55.6**$_{\pm 0.4}$	**47.3**$_{\pm 1.2}$	**46.7**$_{\pm 1.3}$	**40.1**$_{\pm 1.2}$	**39.3**$_{\pm 1.2}$

Table 5. *Comparison with object-centric methods on COCO, PASCAL, MOVi-C and MOVi-E datasets*. SPOT results are the mean and std over 3 seeds. DINOSAUR uses an autoregressive decoder and DINO [7] ViT encoder (ViT-B/16 for PASCAL and MOVi-C, ViT-S/8 for COCO). DINOSAUR-MLP uses an MLP decoder and DINO ViT encoder (ViT-B/16 for COCO and PASCAL, ViT-S/8 for MOVi-C/E). †: COCO and PASCAL results of SA and SLATE are from [60], MOVi-C/E results are from [41] for SA and from [24] for SLATE, PASCAL results of CAE are from [37].

(a) Ground Truth

(b) SPOT

(c) SPOT w/o self-training

(d) SPOT w/o self-training & w/o sequence permutation

Figure 6. Example results on COCO 2017, using 7 slots.

the supplementary material. Our method outperforms others in all scenarios except PASCAL's MBO^i, where it ranks second to SlotDiffusion [60]. Notably, on the demanding COCO dataset, our approach excels, surpassing the prior state-of-the-art by 2.7 and 5.9 points in MBO^i and MBO^c metrics, respectively. This underscores its superiority in unsupervised object-centric learning with real-world data.

In supplementary material, we provide additional results about the impact of bi-level optimized queries [23] on DINOSAUR [41] and SlotDiffusion [60] frameworks. Additionally, we examine SPOT with other pre-trained image features using MoCo-v3 [10] and MAE [19] encoders.

4.4. Qualitative results

In Fig. 6, we present the effects of self-training and sequence permutation. This combination effectively mitigates over-segmentation issues while preserving a high degree of detailed segmentation. More examples of the efficacy of SPOT are provided in Fig. 1, showcasing its robust performance across various images.

5. Conclusion

In conclusion, SPOT advances unsupervised object-centric learning for real-world images by enhancing slot-based auto-encoders through two key strategies. Firstly, a self-training scheme uses decoder-generated attention masks to improve slot attention in the encoder. Secondly, a novel patch-order permutation strategy for autoregressive transformers boosts the decoder's performance without additional training cost. The synergistic application of these strategies enables SPOT to achieve state-of-the-art results in real-world object-centric learning.

We note that sequence permutation decoding might be more broadly beneficial to other computer vision tasks employing autoregressive decoders.

We discuss the limitations of our method and directions for future work in the supplementary material.

Acknowledgements This research work was supported by the Hellenic Foundation for Research and Innovation (HFRI) and the General Secretariat of Research and Innovation (GSRI) under the 4th Call for H.F.R.I. Scholarships to PhD Candidates (grant: 11252). It was also partially supported by the RAMONES and iToBos EU Horizon 2020 projects under grants 101017808 and 965221, respectively. NTUA thanks NVIDIA for the support with the donation of GPU hardware.

References

[1] Görkay Aydemir, Weidi Xie, and Fatma Guney. Self-supervised object-centric learning for videos. In *NeurIPS*, 2023. 2

[2] Federico Baldassarre and Hossein Azizpour. Towards self-supervised learning of global and object-centric representations. In *ICLRW*, 2022. 2

[3] Zhipeng Bao, Pavel Tokmakov, Allan Jabri, Yu-Xiong Wang, Adrien Gaidon, and Martial Hebert. Discorying object that can move. In *CVPR*, 2022. 2

[4] Zhipeng Bao, Pavel Tokmakov, Yu-Xiong Wang, Adrien Gaidon, and Martial Hebert. Object discovery from motion-guided tokens. In *CVPR*, 2023. 2

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020. 2

[6] Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019. 1, 2

[7] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021. 3, 6, 8

[8] Michael Chang, Tom Griffiths, and Sergey Levine. Object representations as fixed points: Training iterative refinement algorithms with implicit differentiation. In *NeurIPs*, 2022. 2, 6

[9] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *ICML*, 2020. 2

[10] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *ICCV*, 2021. 8

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Jakob Uszkoreit, Mostafa Dehghani Neil Houlsby, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 4, 6

[12] Ye Du, Yujun Shen, Haochen Wang, Jingjing Fei, Wei Li, Liwei Wu, Rui Zhao, Zehua Fu, and Qingjie LIU. Learning from future: A novel self-training framework for semantic segmentation. In *NeurIPS*, 2022. 2

[13] Gamaleldin Fathy Elsayed, Aravindh Mahendran, Sjoerd van Steenkiste, Klaus Greff, Michael Curtis Mozer, and Thomas Kipf. SAVi++: Towards end-to-end object-centric learning from real-world videos. In *NeurIPS*, 2022. 2

[14] Martin Engelcke, Adam R. Kosiorek, Oiwi Parker Jones, and Ingmar Posner. Genesis: Generative scene inference and sampling with object-centric latent representations. In *ICLR*, 2020. 1, 2, 6

[15] SM Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Geoffrey E Hinton, et al. Attend, infer, repeat: Fast scene understanding with generative models. *NeurIPs*, 2016. 1, 2

[16] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 2009. 6

[17] Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In *ICML*, 2019. 1, 2

[18] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J. Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam Laradji, Hsueh-Ti (Derek) Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, Cengiz Oztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. Kubric: A scalable dataset generator. In *CVPR*, 2022. 6

[19] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 8

[20] Olivier J Hénaff, Skanda Koppula, Jean-Baptiste Alayrac, Aaron Van den Oord, Oriol Vinyals, and Joao Carreira. Efficient visual pretraining with contrastive detection. In *ICCV*, 2021. 2

[21] Olivier J Hénaff, Skanda Koppula, Evan Shelhamer, Daniel Zoran, Andrew Jaegle, Andrew Zisserman, João Carreira, and Relja Arandjelović. Object discovery and representation networks. In *ECCV*, 2022. 2

[22] Allan Jabri, Sjoerd van Steenkiste, Emiel Hoogeboom, Mehdi SM Sajjadi, and Thomas Kipf. Dorsal: Diffusion for object-centric representations of scenes et al. *arXiv preprint arXiv:2306.08068*, 2023. 2

[23] Baoxiong Jia, Yu Liu, and Siyuan Huang. Improving object-centric learning with query optimization. In *ICLR*, 2022. 2, 3, 6, 8

[24] Jindong Jiang, Fei Deng, Gautam Singh, and Sungjin Ahn. Object-centric slot diffusion. In *NeurIPS*, 2023. 2, 3, 8

[25] Laurynas Karazija, Iro Laina, and Christian Rupprecht. Clevrtex: A texture-rich benchmark for unsupervised multi-object segmentation. In *NeurIPS Datasets and Benchmarks Track*, 2021. 2, 6

[26] Jinwoo Kim, Janghyuk Choi, Ho-Jin Choi, and Seon Joo Kim. Shepherding slots to objects: Towards stable and robust object-centric learning. In *CVPR*, 2023. 2, 8

[27] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 6

[28] Thomas Kipf, Gamaleldin Fathy Elsayed, Aravindh Mahendran, Austin Stone, Sara Sabour, Georg Heigold, Rico Jonschkowski, Alexey Dosovitskiy, and Klaus Greff. Conditional object-centric learning from video. In *ICLR*, 2022. 2

[29] Avinash Kori, Francesco Locatello, Francesca Toni, and Ben Glocker. Unsupervised conditional slot attention for object centric learning. *arXiv preprint arXiv:2307.09437*, 2023. 2

[30] Harold W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955. 4

[31] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 6

[32] Zhixuan Lin, Yi-Fu Wu, Skand Vishwanath Peri, Weihao Sun, Gautam Singh, Fei Deng, Jindong Jiang, and Sungjin Ahn. Space: Unsupervised object-oriented scene representation via spatial attention and decomposition. In *ICLR*, 2020. 1, 2

[33] Hong Liu, Jianmin Wang, and Mingsheng Long. Cycle self-training for domain adaptation. In *NeurIPS*, 2021. 3

[34] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *NeurIPS*, 2020. 1, 2, 3, 6, 8

[35] Sindy Löwe, Klaus Greff, Rico Jonschkowski, Alexey Dosovitskiy, and Thomas Kipf. Learning object-centric video models by contrasting sets. *arXiv preprint arXiv:2011.10287*, 2020. 2

[36] Sindy Löwe, Phillip Lippe, Maja Rudolph, and Max Welling. Complex-valued autoencoders for object discovery. *Transactions on Machine Learning Research*, 2022. 1, 2, 8

[37] Sindy Löwe, Phillip Lippe, Francesco Locatello, and Max Welling. Rotating features for object discovery. *NeurIPs*, 2023. 1, 2, 8

[38] Ke Mei, Chuang Zhu, Jiaqi Zou, and Shanghang Zhang. Instance adaptive self-training for unsupervised domain adaptation. In *ECCV*, 2020. 3

[39] Matthias Minderer, Alexey Gritsenko, and Neil Houlsby. Scaling open-vocabulary object detection, 2023. 2

[40] Tom Monnier, Elliot Vincent, Jean Ponce, and Mathieu Aubry. Unsupervised Layered Image Decomposition into Object Prototypes. In *ICCV*, 2021. 6

[41] 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 *ICLR*, 2023. 1, 2, 3, 4, 6, 7, 8

[42] Oriane Siméoni, Gilles Puy, Huy V Vo, Simon Roburin, Spyros Gidaris, Andrei Bursuc, Patrick Pérez, Renaud Marlet, and Jean Ponce. Localizing objects with self-supervised transformers and no labels. In *BMVC*, 2021. 3

[43] Oriane Siméoni, Chloé Sekkat, Gilles Puy, Antonin Vobecky, Éloi Zablocki, and Patrick Pérez. Unsupervised object localization: Observing the background to discover objects. In *CVPR*, 2023. 3

[44] Gautam Singh, Fei Deng, and Sungjin Ahn. Illiterate dall-e learns to compose. In *ICLR*, 2022. 1, 2, 3, 8

[45] Gautam Singh, Yi-Fu Wu, and Sungjin Ahn. Simple unsupervised object-centric learning for complex and naturalistic videos. In *NeurIPS*, 2022. 2, 3

[46] Gautam Singh, Yeongbin Kim, and Sungjin Ahn. Neural systematic binder. In *ICLR*, 2023. 2

[47] Andrew Stange, Robert Lo, Abishek Sridhar, and Kousik Rajesh. Exploring the role of the bottleneck in slot-based models through covariance regularization. *arXiv preprint arXiv:2306.02577*, 2023. 2

[48] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and Brian Fuller. Llama 2: Open foundation and fine-tuned chat models, 2023. 2

[49] Manuel Traub, Sebastian Otte, Tobias Menge, Matthias Karlbauer, Jannik Thuemmel, and Martin V. Butz. Learning what and where: Disentangling location and identity tracking without supervision. In *ICLR*, 2023. 2

[50] Michael Tschannen, Manoj Kumar, Andreas Peter Steiner, Xiaohua Zhai, Neil Houlsby, and Lucas Beyer. Image captioners are scalable vision learners too. In *NeurIPs*, 2023. 2, 4, 7

[51] Renaud Vandeghen, Gilles Louppe, and Marc Van Droogenbroeck. Adaptive self-training for object detection. In *ICCV Workshop*, 2023. 2

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 2, 3

[53] Xudong Wang, Rohit Girdhar, Stella X Yu, and Ishan Misra. Cut and learn for unsupervised object detection and instance segmentation. *arXiv preprint arXiv:2301.11320*, 2023. 3

[54] Nick Watters, Loic Matthey, Chris P. Burgess, and Alexander Lerchner. Spatial broadcast decoder: A simple architecture for disentangled representations in VAEs. In *ICLR workshops*, 2019. 2

[55] Chen Wei, Kihyuk Sohn, Clayton Mellina, Alan Yuille, and Fan Yang. Crest: A class-rebalancing self-training framework for imbalanced semi-supervised learning. In *CVPR*, 2021. 2

[56] Marissa A. Weis, Kashyap Chitta, Yash Sharma, Wieland Brendel, Matthias Bethge, Andreas Geiger, and Alexander S. Ecker. Benchmarking unsupervised object representations for video sequences. *Journal of Machine Learning Research*, 2021. 2

[57] Xin Wen, Bingchen Zhao, Anlin Zheng, Xiangyu Zhang, and Xiaojuan Qi. Self-supervised visual representation learning with semantic grouping. *NeurIPs*, 2022. 2

[58] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989. 2

[59] Ziyi Wu, Nikita Dvornik, Klaus Greff, Thomas Kipf, and Animesh Garg. Slotformer: Unsupervised visual dynamics simulation with object-centric models. In *ICLR*, 2023. 2

[60] Ziyi Wu, Jingyu Hu, Wuyue Lu, Igor Gilitschenski, and Animesh Garg. Slotdiffusion: Object-centric generative modeling with diffusion models. In *NeurIPS*, 2023. 2, 3, 6, 8

[61] Jiarui Xu, Shalini De Mello, Sifei Liu, Wonmin Byeon, Thomas Breuel, Jan Kautz, and Xiaolong Wang. Groupvit: Semantic segmentation emerges from text supervision. In *CVPR*, 2022. 2

[62] I Zeki Yalniz, Hervé Jégou, Kan Chen, Manohar Paluri, and Dhruv Mahajan. Billion-scale semi-supervised learning for image classification. *arXiv preprint arXiv:1905.00546*, 2019. 2

[63] Lihe Yang, Wei Zhuo, Lei Qi, Yinghuan Shi, and Yang Gao. St++: Make self-training work better for semi-supervised semantic segmentation. In *CVPR*, 2022. 2

[64] Qize Yang, Xihan Wei, Biao Wang, Xian-Sheng Hua, and Lei Zhang. Interactive self-training with mean teachers for semi-supervised object detection. In *CVPR*, 2021. 2

[65] Yafei Yang and Bo Yang. Promising or elusive? unsupervised object segmentation from real-world single images. In *NeurIPS*, 2022. 1, 2

[66] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPs*, 2019. 2

[67] Andrii Zadaianchuk, Maximilian Seitzer, and Georg Martius. Object-centric learning for real-world videos by predicting temporal feature similarities. In *NeurIPS*, 2023. 2

[68] Yi Zhu, Zhongyue Zhang, Chongruo Wu, Zhi Zhang, Tong He, Hang Zhang, R Manmatha, Mu Li, and Alexander J Smola. Improving semantic segmentation via efficient self-training. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 2

[69] Yang Zou, Zhiding Yu, B. V. K. Vijaya Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *ECCV*, 2018. 3