Towards Interpretable Controllability in Object-Centric Learning

Anonymous CVPR submission

Paper ID *****

Abstract

Object-centric learning (OCL) is extensively researched to better understand complex scenes by acquiring object representations or slots. While recent studies in OCL have made strides with complex images, the interpretability and interactivity over object representation remain largely uncharted. In this paper, we introduce a novel method, Slot Attention with Image Augmentation (SlotAug), to explore the possibility of learning interpretable controllability over slots in a self-supervised manner by utilizing image augmentation. We also devise the concept of sustainability in controllable slots by introducing iterative and reversible controls over slots with two proposed submethods: Auxiliary Identity Manipulation and Slot Consistency Loss.

1. Introduction

Compositional comprehension of visual scenes [10, 18, 28], essential for various computer vision tasks such as localization [4] and reasoning [27], requires human-like understanding of complex world [24, 35, 36]. In response to this, *object-centric learning* (OCL) has emerged as an active research area [11, 22, 25]. OCL aims to enable a model to decompose an image into objects, and to acquire their representations, *slots*, without human-annotated labels.

In this work, we advance the field of OCL in terms of the interpretability of object representation. To achieve interpretable controllability, we propose a method that enables the manipulation of slots through human interpretable instructions in a self-supervised manner. We address the training-inference discrepancy by incorporating image augmentation and *slot manipulation* into our training pipeline. Consequently, we resolve the discrepancy and streamline the way to interact with slots in the inference.

Second, to attain sustainability in object representation, we introduce *Auxiliary Identity Manipulation* (AIM) and *Slot Consistency Loss* (SCLoss). AIM is designed to assist in learning the concept of multi-round manipulation. AIM is implemented by incorporating an *auxiliary manipulation* into the intermediate stage of slot manipulation, where the auxiliary manipulation introduces no semantic changes to object properties. This simple auxiliary process can expose our model to multi-round manipulation: we can make two-round manipulations with one instruction from the augmentation and the other from the auxiliary manipulation. Moreover, SCLoss enables learning the concept of reversible manipulation, such as the relationship between moving an object to the right and returning to the left. After being trained with SCLoss, our model produces consistent and reusable representations that can undergo multiple modifications.

Extensive experiments demonstrate the interpretable and sustainable controllability of our model. To assess interpretability, we conduct object manipulation experiments where slots are guided by semantically interpretable instructions. In evaluating sustainability, we introduce novel experiments, like the durability test. Our evaluation encompasses not only pixel space such as object-level image editing, but also slot space such as property prediction, providing a comprehensive examination of our method.

2. Method

2.1. Slot Attention with Image Augmentation

Data augmentation. We introduce a simple data augmentation scheme that, for a given input reference image $img_{ref} \in \mathbb{R}^{H \times W \times 3}$, generates an augmented image $img_{aug} \in \mathbb{R}^{H \times W \times 3}$ and the transformation instructions between them, $insts_{ref2aug} \in \mathbb{R}^{K \times L}$, where K and L indicate the number of slots and the number values for object properties. img_{aug} is produced by randomly translating, scaling, or color shifting img_{ref}. To transform img_{ref} into img_{aug}, we employ a set of instructions $insts_{ref2aug}$. These instructions comprise a list of values that dictate the augmentation, including translation values, a scaling factor, and color shift values in the HSL color space. We also have the inverse instructions, $insts_{aug2ref} \in \mathbb{R}^{K \times L}$, allowing us to revert img_{aug} back to img_{ref}. Henceforth, for the sake of simplicity in notation, we employ r and a as shorthand for ref and aug. For instance, the expression img_r and img_{r2a} equal img_{ref} and $img_{ref2aug}$, respectively. More details are described in the Appendix.

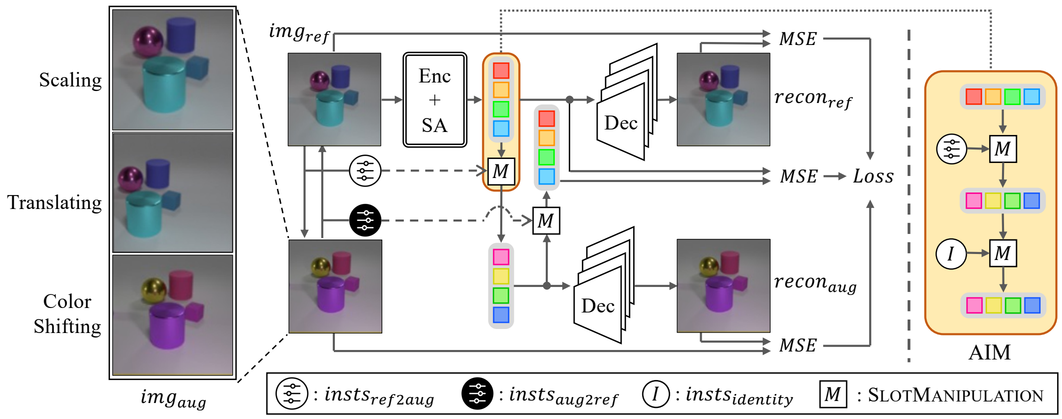

Figure 1. **Model architecture.** From a given image img_{ref}, we generate an augmented image img_{aug} (leftmost part of the figure), and the instruction $insts_{ref2aug}$ and its inverse $insts_{aug2ref}$. Our model produces slots from img_{ref} and decodes them to reconstruct the given image ($recon_{ref}$). The slots are also manipulated, along with $insts_{ref2aug}$, by SLOTMANIPULATION. We incorporate Auxiliary Identity Manipulation (AIM) into this manipulation process. The details are provided in the right part of the figure. The manipulated slots are then simultaneously 1) decoded to the reconstructed augmented image $recon_{aug}$, and 2) re-manipulated by SLOTMANIPULATION with $insts_{aug2ref}$. Our total loss consists of the reconstruction losses of reference and augmented images, and the slot-level cycle consistency.

Training. We propose a novel training process that leverages image augmentation (Fig. 1). Our training scheme enables learning interpretable controllability which allows us to interact with the model via semantically interpretable instructions. Our training process involves data augmentation, spatial binding, slot manipulation, and image reconstruction via slot decoding. For a given input image, we initially perform data augmentation to yield img_r, img_a, $insts_{r2a}$, and $insts_{a2r}$. Then, the model performs SPATIALBINDING on img_r to produce $slots_{ref}$, or $slots_r$.

Thereafter, the model conducts SLOTMANIP (Alg. 1) to modify $slots_r$ based on $insts_{r2a}$. In SLOTMANIP, we utilize a newly introduced property encoder denoted by *PropEnc* which is 3-layer MLPs. This PropEnc generates vector representations, $\texttt{inst\_vec}$, which capture the essence of transformation instructions. Each PropEnc$_j$ generates an $\texttt{inst\_vec}_j$ that encodes the values of $insts_{r2a}$ for the j-th property. These vectors are then added to $slots_r$ to reflect the effect of $insts_{r2a}$. This addition is followed by a residual connection, along with layer normalization and another MLP to generate $slots_{r2a}$.

Lastly, $slots_{r2a}$ is decoded by the decoder to create the $recon_a$, the reconstruction for the augmented image img_a. The MSE between img_a and $recon_a$ serves as a training loss, $\mathcal{L}_{\text{aug}}$. To ensure stable training, we also adopt an additional loss, $\mathcal{L}_{\text{ref}}$, the MSE between the img_r and $recon_r$, the reconstructed reference image decoded from $slots_r$. Accordingly, our training loss for image reconstruction is defined as $\mathcal{L}_{\text{recon}} = \mathcal{L}_{\text{ref}} + \mathcal{L}_{\text{aug}}$.

Inference. To perform object manipulation, the model takes the position of the target object, along with the instruction to be carried out. We use the Hungarian algorithm [23] to find the slot for the object closest to the given position.

Algorithm 1 Slot manipulation algorithm in pseudo-code. J represents the number of object properties, while $\mathtt{P}_{j,f}$ and $\mathtt{P}_{j,l}$ indicate the first and last indices of the j-th property values. We use Layer Normalization $\mathtt{LN}$ to normalize vectors.

```
function SLOTMANIP(slots, insts)
    for j = 0 ... J do
        inst_j = insts[:, P_{j,f} : P_{j,l}]
        inst_vec_j = PropEnc_j(LN(inst_j))
        slots = slots + inst_vec_j
    end for
    slots = slots + MLP(LN(slots))
    return slots
end function
```

To predict the position of an object encoded in a slot, we compute the center of mass acquired from the alpha mask by the decoder or from the attention map between the visual encodings and the slot. After figuring out the desired slot, we perform slot manipulation with the given instructions.

2.2. Sustainability in Object Representation

In this work, we introduce *sustainability*: the concept that slots should sustain their integrity even after iterative manipulations. Thus, sustainability is a key feature that contributes to the reliable and reusable object representation.

Auxiliary Identity Manipulation (AIM) serves as the identity operation for slot manipulation, indicating no changes in object properties. By manipulating slots with instructions having zero values for translation, one for scaling, and so on, AIM is supposed to make each slot preserve the identity of the object. We incorporate AIM into the training process to make the model recognize and maintain the in-

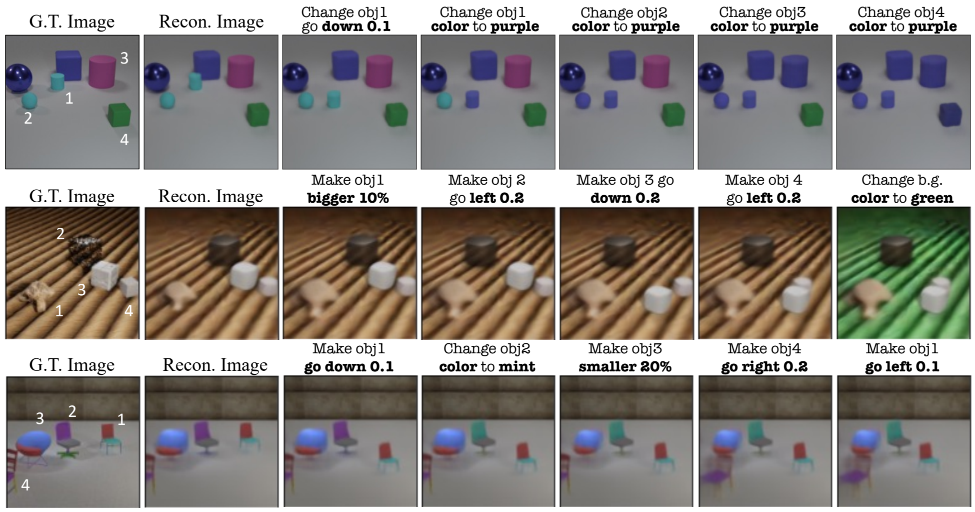

Figure 2. **Results of object manipulation.** The first two columns are the ground truths and reconstructions. The other columns are the results of manipulation along the human-interpretable instructions. The instructions are shown in the text for easy understanding. The instantiation of instruction can be found in the Appendix. From the first row onwards, the results are for CLEVR, CLEVRTEX, and PTR.

trinsic properties of objects during iterative manipulations. AIM is applied to the slot manipulation process as follows:

$$\begin{aligned} slots'_{r2a} &= f(f(slots_r, insts_{r2a}), insts_{id}) \\ &= f(slots_{r2a}, insts_{id}), \end{aligned} \tag{1}$$

where f represents the SLOTMANIP function, and $insts_{id}$, or $insts_{identity}$, is the instruction that contains the identity elements for manipulating properties. In the followings, $slots'_{r2a}$ is notated as $slots_{r2a}$ for simplicity.

Slot Consistency Loss (SCLoss) addresses the issue of a slot diverging significantly from its original state after iterative manipulations, even when a user intends to restore the corresponding object to its original state. To implement SCLoss, we introduce $slots_{restored}$, which is derived by executing a series of SLOTMANIP operations on $slots_r$ using $insts_{r2a}$ and $insts_{a2r}$. Supposed that our goal is to ensure $slots_r$ and $slots_{restored}$ have the same representation, we set the MSE between them as SCLoss. As a result, the model learns to keep the two distinct slots representing the same object as close as possible and to be robust against multiple rounds of manipulation. The equation of SCLoss, $\mathcal{L}_{\text{cycle}}$, and the total training loss, $\mathcal{L}_{\text{total}}$, are as follows:

$$\mathcal{L}_{\text{cycle}} = \frac{1}{K} \| f(slots_{r2a}, insts_{a2r}) - slots_r \|_2^2, \tag{2}$$

$$\mathcal{L}_{\text{total}} = w_{recon}\mathcal{L}_{\text{recon}} + w_{cycle}\mathcal{L}_{\text{cycle}}, \tag{3}$$

where K is the number of slots, f is the SLOTMANIP function, and w_{recon} and w_{cycle} are the weights for the losses.

3. Experiments

Experimental Details. We evaluate models on three multi-object datasets: CLEVR6 [18], CLEVRTEX6 [19] and PTR [16]. Regarding the baseline models, we basically employ the model architecture of Slot Attention. For CLEVRTEX6 and PTR, we replace the encoder with ViT [5] pretrained by MAE [15] and the decoder with that of SRT [30] while using an increased size of the slot attention module. The additional details for adopting large models are described in the Appendix. To clarify the methods used in ablative studies, we categorize our model into three versions: **v1**, which is exclusively trained with image augmentation; **v2**, which improves upon v1 with AIM; **v3**, which extends v1 with both AIM and SCLoss. More details including the training schemes are described in the Appendix.

3.1. Interpretable Controllability: Image Editing

As shown in Fig. 2, we can manipulate various properties of a single object, even multiple times, according to user-given instructions. Based on this observation, we can ascertain that our object representations retain the intrinsic properties of objects seamlessly even after manipulation. This interpretable controllability is achieved with a neglectable compromise of the performance on the object discovery task. It is also worth noting that we can also manipulate the background as shown in the second row of Fig. 2 since we employ SLASH [20] which treats the background as a single entity. Deeper discussions with theoretical proof and empirical results are provided in the Appendix.

3.2. Sustainability: Durability Test

To assess the sustainability of our model, we devise a novel evaluation, called *durability test*. In the durability test, we evaluate how many manipulations a model can endure while preserving object representation intact. As shown in Fig.

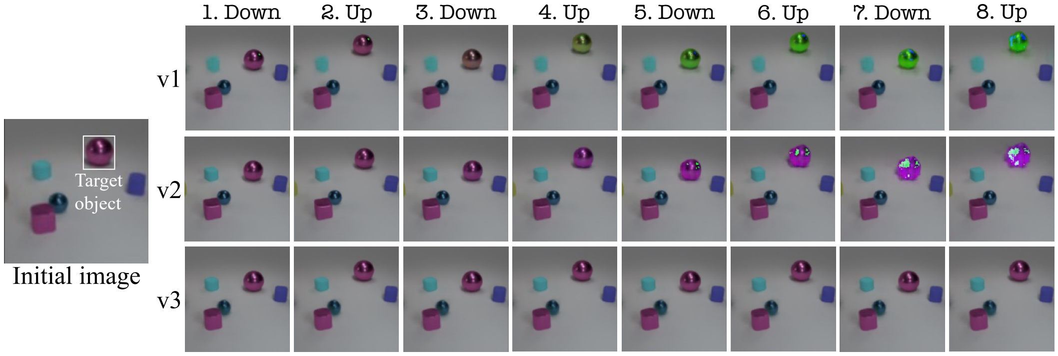

Figure 3. **Durability test.** The leftmost image is the initial image for the test. Each row shows the results of that each model is instructed to alternately move the target object up and down four times each.

Table 1. **Results on durability test** with MSE on CLEVR6.

	Slot (↓)	Obj. Pos. (↓)
	Single step (x8)	
(v1) Train w/ aug.	50.8	0.14
(v2) + AIM	39.7	0.15
(v3) + AIM + SCLoss	**0.25**	**0.01**
	Multiple steps (x4)	
(v1) Train w/ aug.	54.0	0.16
(v2) + AIM	41.4	0.11
(v3) + AIM + SCLoss	**0.31**	**0.02**

3, while v1 fails to keep the color after the second round, v2 relatively preserves the color well for the fourth round. Nevertheless, from the fifth round, the texture progressively diverges from its original. Different from the v1 and v2, v3 demonstrates strong durability despite a greater number of manipulations. We also perform quantitative evaluations on 100 randomly selected samples from CLEVR6 to measure the intrinsic deformity of slots and extrinsic change of object properties, especially position. As shown in Tab. 1, we achieve better sustainability as the model evolves from v1 to v2 and v3 by utilizing the proposed AIM and SCLoss.

3.3. Slot Space Analysis: Property Prediction

In addition to the pixel space analysis, for a comprehensive assessment of the effectiveness of our method, we extend our examination to the analysis of the latent slot space. To evaluate the quality of slots concerning human-interpretable object properties, such as size, color, material, shape, and position, we conduct a *property prediction* task using CLEVR6. This task enables us to scrutinize how well the slots are distributed within the latent vector space following the properties of corresponding objects. The analysis allows for an understanding of object representations that impart semantic existence significance beyond mere segmenting objects in an image.

A *property predictor*, consisting of 3-layer MLPs, takes slots as input and predicts a property of objects. Each property predictor is trained by supervised learning using the ground truths matched by the Hungarian algorithm [23]. To assess the quality of object representations, we freeze the OCL models that produce slots. As shown in Tab. 2, our model outperforms the baseline method [20] across all properties including those, like material and shape, that are not addressed during training. Moreover, in Fig. 4, qualitative results using t-SNE [37] show that while the original slots do not appear to be well-clustered, slots obtained by SlotAug exhibit better adaptability to the downstream task. This observation reinforces our quantitative findings.

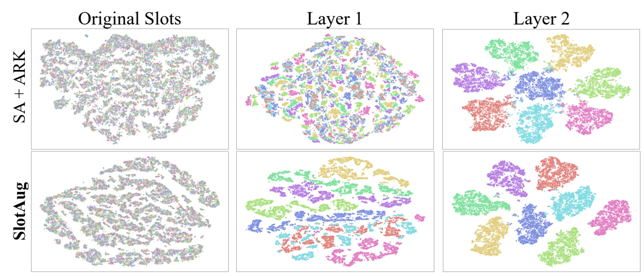

Figure 4. **t-SNE of slots on property prediction for color.** Each row shows the results of the baseline model (SA + ARK) and our model (SlotAug), respectively. The first column is the result of the original slots obtained from the spatial binding. The second and third columns are the results of the intermediate outputs from the first and second MLP layers of the property predictor, respectively.

Table 2. **Property prediction results** on CLEVR6 in F1 score. The number of classes are shown in the parenthesis after the property name. We set two distance thresholds for position prediction.

	SA + ARK	SlotAug (Ours)
Size (2)	69.7	**82.2**
Color (8)	63.5	**78.2**
Material (2)	70.4	**82.6**
Shape (3)	59.1	**73.0**
Pos@0.15	71.1	**84.2**
Pos@0.05	51.8	**77.2**

References

[1] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 37–49. JMLR Workshop and Conference Proceedings, 2012. 6

[2] Ondrej Biza, Sjoerd van Steenkiste, Mehdi SM Sajjadi, Gamaleldin F Elsayed, Aravindh Mahendran, and Thomas Kipf. Invariant slot attention: Object discovery with slot-centric reference frames. *arXiv preprint arXiv:2302.04973*, 2023. 6, 8

[3] 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. 6

[4] Minsu Cho, Suha Kwak, Cordelia Schmid, and Jean Ponce. Unsupervised object discovery and localization in the wild: Part-based matching with bottom-up region proposals. In *CVPR*, pages 1201–1210, 2015. 1

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020. 3, 8

[6] Gamaleldin Elsayed, Aravindh Mahendran, Sjoerd van Steenkiste, Klaus Greff, Michael C Mozer, and Thomas Kipf. Savi++: Towards end-to-end object-centric learning from real-world videos. *Advances in Neural Information Processing Systems*, 35:28940–28954, 2022. 8

[7] Martin Engelcke, Adam R Kosiorek, Oiwi Parker Jones, and Ingmar Posner. Genesis: Generative scene inference and sampling with object-centric latent representations. *arXiv preprint arXiv:1907.13052*, 2019. 6

[8] Martin Engelcke, Oiwi Parker Jones, and Ingmar Posner. Genesis-v2: Inferring unordered object representations without iterative refinement. *Advances in Neural Information Processing Systems*, 34:8085–8094, 2021. 6

[9] Jerome Feldman. The neural binding problem (s). *Cognitive neurodynamics*, 7:1–11, 2013. 6

[10] Martin A Fischler and Robert A Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on computers*, 100(1):67–92, 1973. 1

[11] Klaus Greff, Antti Rasmus, Mathias Berglund, Tele Hao, Harri Valpola, and Jürgen Schmidhuber. Tagger: Deep unsupervised perceptual grouping. *Advances in Neural Information Processing Systems*, 29, 2016. 1

[12] 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. pages 2424–2433. PMLR, 2019. 6

[13] Klaus Greff, Sjoerd Van Steenkiste, and Jürgen Schmidhuber. On the binding problem in artificial neural networks. *arXiv preprint arXiv:2012.05208*, 2020. 6

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 8

[15] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 3, 8

[16] Yining Hong, Li Yi, Joshua B Tenenbaum, Antonio Torralba, and Chuang Gan. Ptr: A benchmark for part-based conceptual, relational, and physical reasoning. *NeurIPS*, 2021. 3

[17] Daniel Im Im, Sungjin Ahn, Roland Memisevic, and Yoshua Bengio. Denoising criterion for variational auto-encoding framework. In *Proceedings of the AAAI conference on artificial intelligence*, 2017. 9

[18] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, pages 2901–2910, 2017. 1, 3

[19] Laurynas Karazija, Iro Laina, and Christian Rupprecht. Clevrtex: A texture-rich benchmark for unsupervised multi-object segmentation. *NeurIPS*, 2021. 3

[20] Jinwoo Kim, Janghyuk Choi, Ho-Jin Choi, and Seon Joo Kim. Shepherding slots to objects: Towards stable and robust object-centric learning. *arXiv preprint arXiv:2303.17842*, 2023. 3, 4, 8, 9

[21] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 6

[22] Thomas Kipf, Gamaleldin F Elsayed, Aravindh Mahendran, Austin Stone, Sara Sabour, Georg Heigold, Rico Jonschkowski, Alexey Dosovitskiy, and Klaus Greff. Conditional object-centric learning from video. *arXiv preprint arXiv:2111.12594*, 2021. 1, 6

[23] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 1955. 2, 4

[24] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40:e253, 2017. 1

[25] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. 33:11525–11538, 2020. 1, 6, 8

[26] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015. 6

[27] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *arXiv preprint arXiv:1904.12584*, 2019. 1

[28] David Marr. *Vision: A computational investigation into the human representation and processing of visual information*. MIT press, 2010. 1

[29] Mehdi SM Sajjadi, Daniel Duckworth, Aravindh Mahendran, Sjoerd van Steenkiste, Filip Pavetic, Mario Lucic, Leonidas J Guibas, Klaus Greff, and Thomas Kipf. Object scene representation transformer. 35:9512–9524, 2022. 6, 8

[30] Mehdi SM Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lučić, Daniel Duckworth, Alexey Dosovitskiy, et al. Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6229–6238, 2022. 3, 8

[31] Maximilian Seitzer, Max Horn, Andrii Zadaianchuk, Dominik Zietlow, Tianjun Xiao, Carl-Johann Simon-Gabriel, Tong He, Zheng Zhang, Bernhard Schölkopf, Thomas Brox, et al. Bridging the gap to real-world object-centric learning. *arXiv preprint arXiv:2209.14860*, 2022. 6

[32] Gautam Singh, Fei Deng, and Sungjin Ahn. Illiterate dall-e learns to compose. *arXiv preprint arXiv:2110.11405*, 2021. 6, 9, 11

[33] Gautam Singh, Yi-Fu Wu, and Sungjin Ahn. Simple unsupervised object-centric learning for complex and naturalistic videos. *Advances in Neural Information Processing Systems*, 35:18181–18196, 2022. 6

[34] Gautam Singh, Yeongbin Kim, and Sungjin Ahn. Neural systematic binder. In *ICLR*, 2023. 6

[35] Elizabeth S Spelke and Katherine D Kinzler. Core knowledge. *Developmental science*, 10(1):89–96, 2007. 1

[36] Anne Treisman. The binding problem. *Current opinion in neurobiology*, 6(2):171–178, 1996. 1, 6

[37] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9 (11), 2008. 4

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. 30, 2017. 9

[39] Yanbo Wang, Letao Liu, and Justin Dauwels. Slot-vae: Object-centric scene generation with slot attention. 2023. 6

[40] Nicholas Watters, Loic Matthey, Christopher P Burgess, and Alexander Lerchner. Spatial broadcast decoder: A simple architecture for learning disentangled representations in vaes. *arXiv preprint arXiv:1901.07017*, 2019. 8

[41] Ziyi Wu, Nikita Dvornik, Klaus Greff, Thomas Kipf, and Animesh Garg. Slotformer: Unsupervised visual dynamics simulation with object-centric models. *arXiv preprint arXiv:2210.05861*, 2022. 6

[42] Sirui Xie, Ari S Morcos, Song-Chun Zhu, and Ramakrishna Vedantam. Coat: Measuring object compositionality in emergent representations. pages 24388–24413. PMLR, 2022. 6

A. Related Works

The binding problem in artificial neural networks [13], inspired by cognitive science [9, 36], is a subject of active exploration, aiming to attain human-like recognition abilities by understanding the world in terms of symbol-like entities such as objects. In computer vision, object-centric learning (OCL) focuses on comprehending visual scenes by considering objects and their relationships without labeled data [8, 41, 42]. MONet [3], IODINE [12], and GENESIS [7] have adopted autoencoding architectures [1, 21, 26] to accomplish self-supervised OCL, and Slot Attention [25] introduced the concept of slot competition, which enables parallel updates of slots with a single visual encoding and decoding stage. Recent studies have leveraged large-scale models to learn object representations in complex images [31, 32], multi-view images [29], and videos [22, 33].

Several studies have shown the possibility of interacting with object representation to manipulate the objects. VAE-based models such as IODINE [12] and Slot-VAE [39] showed that adjusting the values of slots can change object properties. SysBinder [34] demonstrated that replacing factor-level slot, called block, between slots exchanges the corresponding properties. However, these works have difficulties in determining ways to interact with slots as they require manual efforts to identify the features associated with specific properties. ISA [2] incorporates spatial symmetries of objects using slot-centric reference frames into the spatial binding process, enhancing interactivity of object representation for spatial properties such as position and scale. Meanwhile, our method itself has no constraint on the types of the target property, showing its expandability toward extrinsic properties such as the shape and material of objects if there exist proper image augmentation skills or labeled data.

B. Spatial Binding in Slot Attention

The core mechanism of the slot attention, the spatial binding, is described in Alg. 2. Given an input image `img` $\in \mathbb{R}^{H\times W\times 3}$, CNN encoder generates a visual feature map `input` $\in \mathbb{R}^{N\times D_{enc}}$, where H, W, N, and D_{enc} are the height and width of the input image, the number of pixels in the input image ($= HW$), and the channel of the visual feature map. The slot attention module takes `slots` and `inputs`, and projects them to dimension D_{slot} through linear transformations k, q, and v. Dot-product attention is applied to generate an attention map, `attn`, with query-wise normalized coefficients, enabling slots to compete for the most relevant pixels of the visual feature map. The attention map coefficients weight the projected visual feature map to produce updated slots, `updates`. With the iterative mechanism of the slot attention module, the slots can gradually refine their representations.

Algorithm 2 Spatial binding in slot attention algorithm in pseudo-code format. The input image is encoded into a set of N vectors of dimension D_{input} which is mapped to a set of K vectors with dimension D_{slot}. Slots are initialized from a Gaussian distribution with learned parameters $\mu, \sigma \in \mathbb{R}^{D_{slot}}$. The number of iterations is set to $T = 3$.

function SPATIALBINDING($\mathtt{img} \in \mathbb{R}^{H\times W\times 3}$)
    $\mathtt{inputs} = \mathtt{Encoder(img)}$
    $\mathtt{inputs} = \mathtt{LayerNorm(inputs)}$
    for $t = 0 \dots T$ **do**
        $\mathtt{slots\_prev} = \mathtt{slots}$
        $\mathtt{slots} = \mathtt{LayerNorm(slots)}$
        $\mathtt{attn} = \mathtt{Softmax}(\frac{1}{\sqrt{D_{slot}}} k(\mathtt{inputs}) \cdot q(\mathtt{slots})^T, \mathtt{axis=`slots'})$
        $\mathtt{updates} = \mathtt{WeightedMean(weights=attn+}\epsilon\mathtt{,\ values=}v\mathtt{(inputs))}$
        $\mathtt{slots} = \mathtt{GRU(state=slots\_prev,\ inputs=updates)}$
        $\mathtt{slots} = \mathtt{slots} + \mathtt{MLP(LayerNorm(slots))}$
    end for
    return slots
end function

C. Implementation and experimental details

C.1. Training

We use a single V100 GPU with 16GB of RAM with 1000 epochs and a batch size of 64. The training takes approximately 65 hours (wall-clock time) using 12GB of RAM for the CLEVR6 dataset, and 22 hours using 9GB of RAM for the Tetrominoes dataset, both with 16-bit precision.

C.2. Image Augmentation

Upon receiving an input image img_{input}, we produce four outputs: a reference image, denoted as img_{ref}, an augmented image, represented as img_{aug}, and the transformation instructions between them, indicated as $insts_{ref2aug}$ and $insts_{aug2ref}$.

In the data augmentation process, three pivotal variables are defined. The first is the template size $\mathcal{T}$, employed for the initial cropping of img_{input} prior to the application of transformation (240 for CLEVR6 and 80 for Tetrominoes). Next, the crop size $\mathcal{C}$ is used to crop the transformed image before resizing it to $\mathcal{M}$ (192 for CLEVR6 and 64 for Tetrominoes). This two-stage cropping procedure mitigates the zero-padding that results from transformations. Lastly, the image size $\mathcal{M}$ denotes the final image size post data augmentation (128 for CLEVR6 and 64 for Tetrominoes).

In the training phase, img_{ref} is obtained by applying a center-crop operation on img_{input} using $\mathcal{C}$ and then resizing it to $\mathcal{M}$. The generation of img_{aug} is more complex, entailing the application of a random transformation from a set of three potential transformations. Initially, img_{input} is cropped using $\mathcal{T}$, and the transformation process is implemented. Following this, the transformed image is cropped by $\mathcal{C}$ and then resized to $\mathcal{M}$, yielding img_{aug}. The detailed description for each transformation is as follows:

Translating. We set a maximum translation value $d_{max} = \frac{\mathcal{T}-\mathcal{C}}{2}$. A value is randomly chosen within the range of $(-d_{max}, d_{max})$ for translation along the x-axis (d_x) and the y-axis (d_y) respectively.

Scaling. The maximum and minimum scaling factors, s_{max} and s_{min}, are computed by $\frac{\mathcal{T}}{\mathcal{C}}$ and $\frac{\mathcal{C}}{\mathcal{T}}$, respectively. A float value s, serving as a scaling factor, is then randomly sampled from within the range of (s_{max}, s_{min}). One thing to note is that calculating the transformation instructions is not straightforward due to the potential translation of objects during scaling. Thus, to calibrate the instructions, we infer translation values from the predicted object positions before scaling. The position prediction is calculated as the weighted mean on the attention maps between the visual encodings and slots. With this position prediction, we add the translation term into the scaling process so that the model should perform both object-level scaling and translating: $\vec{d} = (s-1)(\vec{p} - \vec{c})$, where $\vec{d}$ represents the vector of the translation value, $\vec{p}$ refers to the vector of the predicted object position, and $\vec{c}$ is the vector corresponding to the position of image center.

Color shifting. In this study, we employ the HSL (hue, saturation, and lightness) color space for effective object color manipulation. The input image, initially in RGB space, is converted to HSL space. We adjust the hue by rotating it using randomly sampled angles that span the entire hue space. For saturation, we apply a scaling factor, determined by the exponential of a value randomly drawn from (-1, 1), a hyper parameter. Our primary focus lies on the internal color of objects, leaving lightness untouched. Nonetheless, adjustments to lightness can be made if necessary.

Instruction. Each transformation instruction is a list of 6 values: one scaling factor (Λscale), two translation parameters ($\Delta x, \Delta y$), and three color shifting parameters in HSL (Δhue, Λsaturation, Λlightness) where Λ and Δ means the multiplicative and additive factor for the corresponding values, respectively. The identity instruction, $insts_{identity}$, contains the base values for each transformation. Thus, $insts_{identity}$ has 1 for scaling, $(0,0)$ for translation, and $(0,1,1)$ for color shifting. For the inverse instruction , $insts_{aug2ref}$ has the values of $-insts_{ref2aug}$ for additive factors, and $\frac{1}{insts_{ref2aug}}$ for multiplicative factors.

C.3. Model

Basically, our model framework is built on Slot Attention [25], thereby the encoder, decoder, and slot attention module are the same as that of Slot Attention except for the inclusion of the Attention Refining Kernel (ARK) from SLASH [20]. For Tetorminoes and CLEVR, we employ a 4-layer CNN encoder and a 6-layer Spatial Broadcast (SB) decoder [40] with a hidden dimension of 64. Within the slot attention module, we set the slot dimension to 64, perform the binding process for 3 iterations, and use a kernel size of 5 for the ARK. Please refer to the original papers [20, 25] for additional details for Slot Attention.

For CLEVRTEX6 and PTR datasets which include more complicated objects, we adopt larger models with a slot dimension, D_{slot}, of 256. As encoders, we use 1) Resnet34 [14] following [2, 6] and 2) ViT-base [5], with the patch size of 8, pretrained via MAE [15] As decoders, we use an increased size of SB decoder consisting of 8-layer CNNs with a hidden dimension of 128, and a Transformer-based decoder proposed in SRT [30]. The original SRT decoder is designed to operate at the image level, and the following research OSRT [29] introduce a modification to decode slots simultaneously. In this paper, we slightly modified it to decode each slot independently following the spatial broadcast decoder. This selection is made to demonstrate that our proposed method is not limited to CNN-based spatial broadcast decoders used in Slot Attention but can robustly operate within transformer-based decoders as well, given the appropriate conditions for independence.

In Alg. 1 of the main paper, the Property Encoder (`PropertyEncoder`) takes as input the values that correspond to specific properties. Accordingly, the input size for the property encoder is 1 for scaling, 2 for translation, and 3 for color shifting. Each property is encoded via Property Encoder, a 3-layer MLP with ReLU activation functions, resulting in a `inst_vec`, a vector of dimension D_{slot}.

D. Discussion: How does it work?

To begin with, we would like to highlight our unique approach to the training procedure. While our training incorporates manipulations at the *image-level*, it can be perceived as training the model at the individual *object-level*. In this section, we discuss on how this transition is achieved without the need for an additional tuning process, and present empirical results that support our claim.

As we discussed shortly in Sec. 3.1. in the main paper, the success of transitioning from image-level augmentation during training to object-level manipulation during inference can be attributed primarily to the fact that the entire process for each slot, including object discovery and decoding, exclusively influences the reconstruction of its respective *object*. A mathematical proof is provided below to show how an image-level reconstruction loss can be disentangled into object-level reconstruction losses.

$$\mathcal{L}_{\text{recon}} = \|\hat{\mathcal{I}} - \mathcal{I}\|_2^2 \tag{4}$$

$$= \| \sum_{k=1}^{\mathcal{K}} (\hat{\mathcal{I}}_k^{rgb} \odot \hat{\mathcal{I}}_k^{\alpha}) - \mathcal{I}\|_2^2 \tag{5}$$

$$= \| \sum_{k=1}^{\mathcal{K}} (\hat{\mathcal{I}}_k^{rgb} \odot \hat{\mathcal{I}}_k^{\alpha}) - \sum_{k=1}^{\mathcal{K}} (\mathcal{I} \odot \hat{\mathcal{I}}_k^{\alpha})\|_2^2 \tag{6}$$

$$= \| \sum_{k=1}^{\mathcal{K}} (\hat{\mathcal{I}}_k^{rgb} \odot \hat{\mathcal{I}}_k^{\alpha} - \mathcal{I} \odot \hat{\mathcal{I}}_k^{\alpha})\|_2^2 \tag{7}$$

$$\approx \| \sum_{k=1}^{\mathcal{K}} (\hat{\mathcal{O}}_k - \mathcal{O}_k)\|_2^2, \tag{8}$$

$$= \sum_{k=1}^{\mathcal{K}} \|(\hat{\mathcal{O}}_k - \mathcal{O}_k)\|_2^2 + \sum_{\substack{i,j=1 \\ i \neq j}}^{\mathcal{K}} (\hat{\mathcal{O}}_i \cdot \hat{\mathcal{O}}_j - 2\,\hat{\mathcal{O}}_i \cdot \mathcal{O}_j + \mathcal{O}_i \cdot \mathcal{O}_j) \tag{9}$$

$$\approx \sum_{k=1}^{\mathcal{K}} \|(\hat{\mathcal{O}}_k - \mathcal{O}_k)\|_2^2, \tag{10}$$

where $\mathcal{K}$ is the number of slots, $\hat{\mathcal{I}} \in \mathbb{R}^{H\times W\times 3}$ represents the reconstructed image, and $\mathcal{I} \in \mathbb{R}^{H\times W\times 3}$ represents the input image. $\hat{\mathcal{I}}_k^{rgb} \in \mathbb{R}^{H\times W\times 3}$ and $\hat{\mathcal{I}}_k^{\alpha} \in \mathbb{R}^{H\times W\times 1}$ are the reconstruction results generated by the decoder using the k-th slot as input: an RGB and an alpha map (or an attention mask), respectively. $\hat{\mathcal{O}}_k \in \mathbb{R}^{H\times W\times 3}$ is the predicted image for the specific object that is bounded with the k-th slot, while $\mathcal{O}_k \in \mathbb{R}^{H\times W\times 3}$ is the corresponding ground-truth object image.

From Eq. (4) to Eq. (5), we follow the decoding process of Slot Attention [25]. In particular, each k-th slot is decoded independently, resulting in the reconstructed RGB image $\hat{\mathcal{I}}_k^{rgb}$ and the reconstructed alpha map $\hat{\mathcal{I}}_k^{\alpha}$. The final reconstruction image $\hat{\mathcal{I}}$ is generated by aggregating $\hat{\mathcal{I}}_k^{rgb}$ using a pixel-level weighted average, where the weights are determined by $\hat{\mathcal{I}}_k^{\alpha}$. It is crucial to recognize that $\hat{\mathcal{I}}_k^{\alpha}$ serves as an attention mask, as elaborated below:

$$\sum_{k=1}^{\mathcal{K}} \hat{\mathcal{I}}_k^{\alpha}(x,y) = 1 \quad \text{for all } \; x, y, \tag{11}$$

where $\hat{\mathcal{I}}_k^{\alpha}(x, y)$ is a value for the position (x, y). This characteristic plays a pivotal role in our approach, facilitating the transition from Eq. (5) to Eq. (6). In this transformation, the input image $\mathcal{I}$ is effectively weighted by the set of $\mathcal{K}$ alpha maps, denoted as $\hat{\mathcal{I}}_k^{\alpha}$, where k spans from 1 to $\mathcal{K}$. Then, as both the first and second terms in Eq. (6) involve the same sigma operations, we can simplify the expression by combining the individual subtraction operations into a single sigma operation (Eq. (7)).

Subsequently, we approximate Eq. (7) as Eq. (8) to get an object-level disentangled version of the reconstruction loss. Here we assume that both $\hat{\mathcal{O}}_k$ and $\mathcal{O}_k$ only consist of a specific region of interest within the input image. This region corresponds to the target object which is bound to the k-th slot, while the remaining areas are masked out and assigned a value of zero. We can make this assumption based on the successful performance of the previous object-centric learning model, SLASH [20]. SLASH has demonstrated effective capabilities in focusing on and capturing specific objects of interest within an image, by introducing the Attention Refining Kernel (ARK). By incorporating ARK into our model, we confidently assume that $\hat{\mathcal{O}}_k$ and $\mathcal{O}_k$ primarily represent the target object while masking out other irrelevant parts as zero as shown in Fig. 5.

Here, we would like to note that ARK is an optional component in our method, not a necessity. The use of ARK is not intended to enhance object discovery performance in a single training session; rather, it is employed to ensure consistent results across multiple experiments. If our proposed training scenario arises where bleeding issues do not occur in the original SA, it can be achieved without the need for ARK. To substantiate this claim, we present qualitative results in Fig. 7, where we train SlotAug with the original SA (without ARK). One can easily catch that the object manipulation fails in the case of bleeding problem. Specifically, the analysis for the failure case in bleeding problem is as follows: 1) Obviously, if the attention map corresponding to the target object encompasses other objects, it becomes impossible to exclusively manipulate solely the target object, leading to unexpected artifacts in other objects. 2) Whenever tinting instructions are applied, objects become gray and we attribute this to the backgrounds – having a gray color – intervening with the target objects during training.

Eq. (8) can be broken down into two separate summations. The first one is our target term that is the sum of object-level MSE losses, and the second term is the residual term. Lastly, the transition from Eq. (9) to Eq. (10) constitutes a significant simplification in the representation of the loss function. This is a valid transformation under the assumption follows:

$$\hat{\mathcal{O}}_i \cdot \hat{\mathcal{O}}_j = \hat{\mathcal{O}}_i \cdot \mathcal{O}_j = \mathcal{O}_i \cdot \mathcal{O}_j = 0 \quad \text{if } i \neq j. \tag{12}$$

This assumption postulates that the inner product of different object images, whether they are predicted or ground-truth, is always zero. We assert that this assumption is justifiable, much like the previous one, given the promising results obtained in our object discovery experiments. The loss computation is thus decomposed into individual components for each slot, which lends itself to an interpretation of object-level loss.

The conversion from image-level MSE loss to a sum of individual object-level MSE losses provides a new perspective on our training method. Despite the use of image-level manipulations, the underlying core of the training process inherently engages with object-level representations. This demonstrates how a simple methodological addition, incorporating image augmentation into the training process, can lead to considerable gains in the model's capacity for user-intention-based object manipulation.

Fig. 5 empirically demonstrates the effectiveness of our model, leveraging Slot Attention for controllability over slots. Conversely, it was noted that the well-known alternative framework for object-centric learning, SLATE [32], employing image tokenization from Discrete VAE (dVAE) [17] and Transformer-based auto-regressive decoding [38], struggled with the manipulation of slots, as illustrated in Fig. 6. The same slot manipulation strategy via Property Encoder was used for comparison. Other training environments are just the same as the official paper [32] except for the addition of the training loss for the reconstruction of the augmented images.

E. Conclusion

We presented an OCL framework, SlotAug, for exploring the potential of interpretable controllability in object-centric learning. To achieve this goal, we tackled the object manipulation task, where we added some conditions regarding interpretability and interactivity, via controlling object representations called slots. We employed image augmentation for training our model in a self-supervised manner to resolve the lack of labeled data. Moreover, we introduced a concept of sustainability in slots, achieved by the proposed method AIM and SCLoss. We substantiated the effectiveness of our methods by providing extensive empirical studies and theoretical evidence in the Appendix. These empirical studies include pixel- and slot-space analyses on tasks such as the durability test and property prediction. Though our work remains several questions detailed in the Appendix and represents just one step on a long journey of OCL, we firmly believe that our work is a foundational piece in the field of interpretable OCL and propel the ongoing effort to equip machines with human-like comprehension abilities.

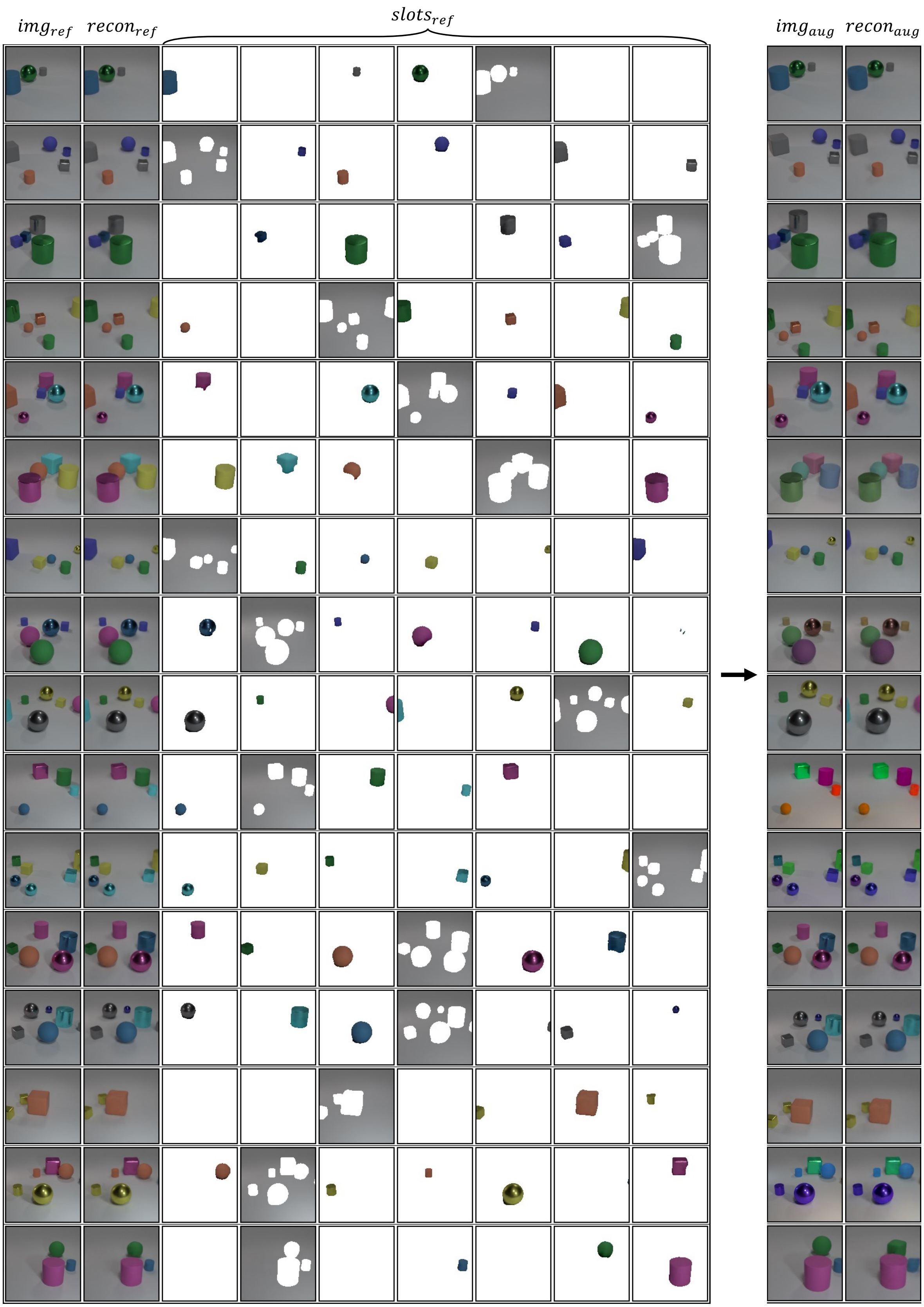

Figure 5. **Training results of our method.** The leftmost column is the reference images, img_{ref}. The second leftmost column is the reconstruction of the reference images, $recon_{ref}$. The middle columns show the object discovery results where each column corresponds to a single slot in $slots_{ref}$. The second rightmost column is the augmented images, img_{aug}. The rightmost column is the reconstruction of the augmented images, $recon_{aug}$.

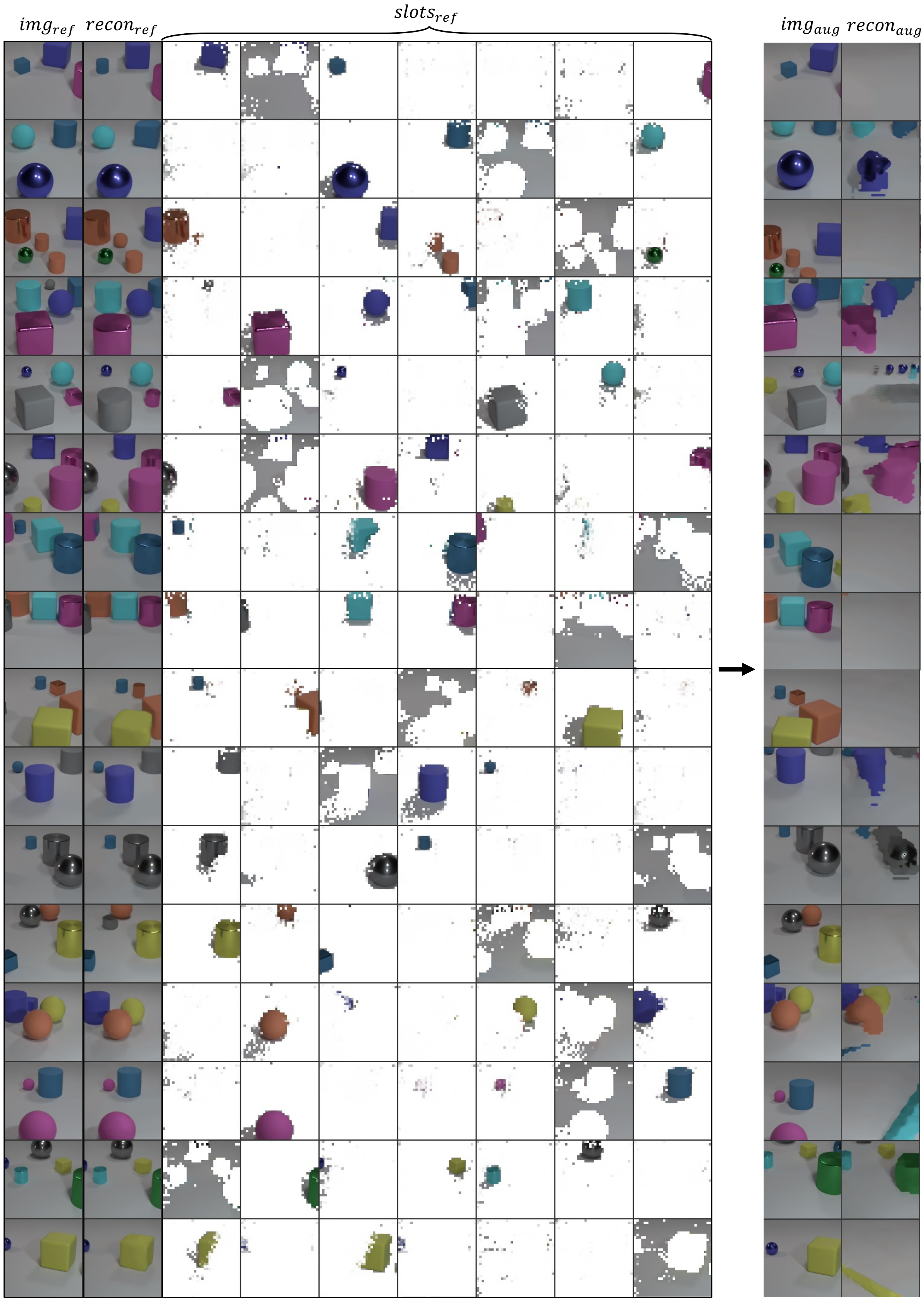

Figure 6. **Training results of SLATE [32] for slot manipulation.** The leftmost column is the reference images, img_{ref}; The second leftmost column is the reconstruction of the reference images, $recon_{ref}$. The middle columns show the object discovery results where each column corresponds to a single slot in $slots_{ref}$. The second rightmost column is the augmented images, img_{aug}. The rightmost column is the reconstruction of the augmented images, $recon_{aug}$.

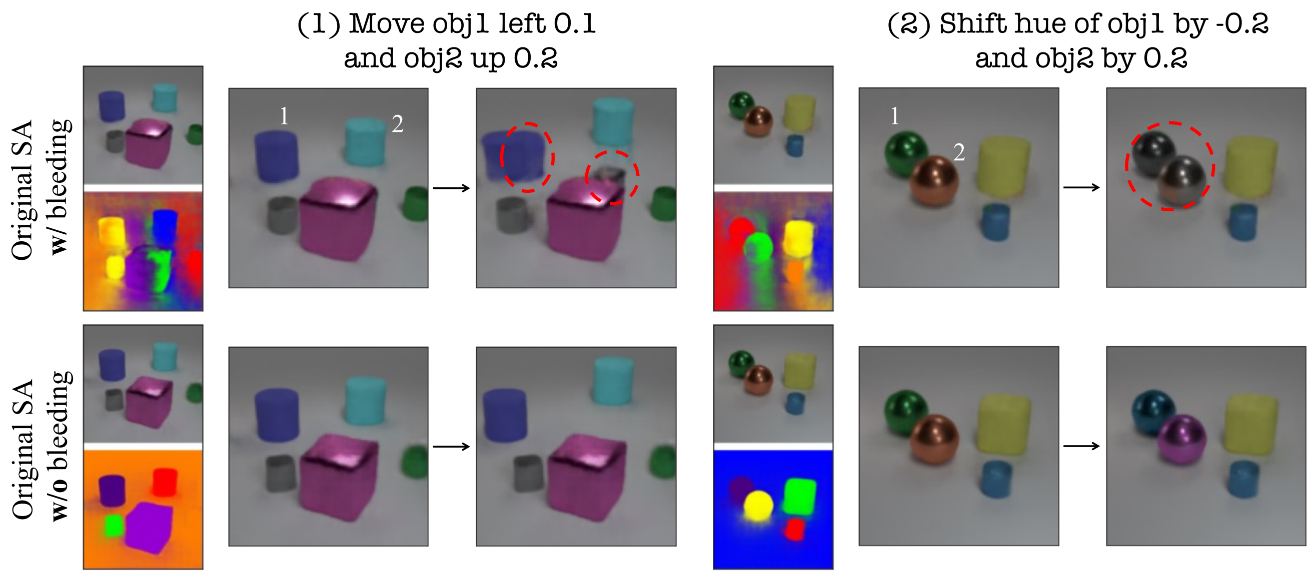

Figure 7. **Visualization of object manipulation results affected by the bleeding problem with the original Slot Attention.** The first row demonstrates the cases where bleeding problem emerges, while the second row shows the cases where the object discovery is done successfully.