A Additional Implementation Details

A.1 Object Discovery Algorithm

We further describe the details of the object discovery process (see Algorithm 1). The input is the set of pixel embeddings for an image. We apply LayerNorm [2] to normalize all pixel embeddings and add positional encodings to the pixel embeddings (Line 3). We use Dijkstra's algorithm to compute the shortest path distance of a sampled pixel embedding to all other embeddings as follows. We uniformly sample an unprocessed pixel embedding $\mathbf{p}_i \in \mathcal{U}$ that has not been assigned to an object (Line 6). We initialize the distance from $\mathbf{p}_i$ to itself as zero and to all the other pixels as infinity (Lines 7-10). Then we select an unvisited pixel embedding $\mathbf{p}_m$ that has the minimum distance to $\mathbf{p}_i$ (Line 13). For each neighbour of the pixel corresponding to $\mathbf{p}_m$, let $\mathbf{p}_k$ be its embedding (Lines 14-15). Here, we consider the neighbours of a pixel to be the 8 surrounding pixels and the distance between a pair of neighbouring pixels is the similarity measure between their corresponding embeddings given by:

$$\text{sim}(\mathbf{p}_m, \mathbf{p}_k) = \sqrt{\sum_{d=1}^{D} (\mathbf{p}_m[d] - \mathbf{p}_k[d])^2}, \tag{10}$$

where $\mathbf{p}_m[d]$ denotes the d^{th} value of the embedding $\mathbf{p}_m$.

We update the shortest path distance between $\mathbf{p}_i$ and $\mathbf{p}_k$ if the distance is shorter through $\mathbf{p}_m$ (Lines 16-18). We mark $\mathbf{p}_m$ as visited (Line 20) and consider $\mathbf{p}_m$ to be part of the same object as $\mathbf{p}_i$ (Lines 21-23). We repeat the process for the next unvisited pixel embedding until all pixels have been visited (Line 24). The discovered object O_c is added to the set of objects found thus far $\mathcal{O}$ (Line 25). The entire process is repeated till all pixel embeddings have been assigned to some object (line 26).

Algorithm 1 Object Discovery Process.

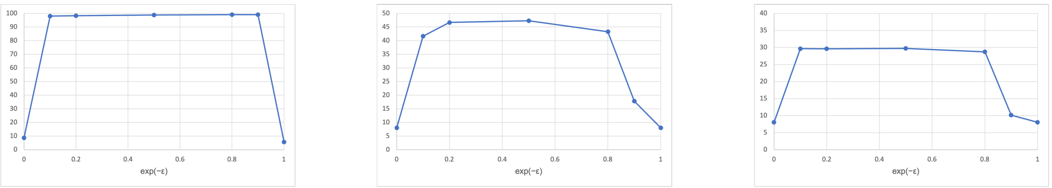
```
 1: Input: Set of pixel embeddings $\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$
 2: Output: Set of objects $\mathcal{O} = \{O_1, \ldots, O_M\}$
 3: $\mathcal{P} \leftarrow$ LayerNorm($\mathcal{P}$) + Position Enc.
 4: $\mathcal{U} \leftarrow \mathcal{P}$; $\mathcal{O} \leftarrow \emptyset$; $c \leftarrow 1$      // Initialization
 5: repeat
 6:     $\mathbf{p}_i \leftarrow$ UniformSampling($\mathcal{U}$)
 7:     dist[$i$,$i$] $\leftarrow 0$
 8:     for $j = 1$ to $N$, $j \neq i$ do
 9:         dist[$i, j$] $\leftarrow \infty$
10:     end for
11:     $O_c \leftarrow \emptyset$; $\mathcal{V} \leftarrow \emptyset$     // Initialization
12:     repeat
13:         let $\mathbf{p}_m \in \mathcal{P} - \mathcal{V}$ be the pixel embedding with minimum distance to $\mathbf{p}_i$
14:         for each neighbor of the pixel corresponding to the embedding $\mathbf{p}_m$ do
15:             let $\mathbf{p}_k$ of be the embedding of the neighbour
16:             if dist[$i, k$] $>$ dist[$i, m$] $+$ sim($\mathbf{p}_m, \mathbf{p}_k$) then
17:                 dist[$i, k$] = dist[$i, m$] $+$ sim($\mathbf{p}_m, \mathbf{p}_k$)
18:             end if
19:         end for
20:         $\mathcal{V} \leftarrow \mathcal{V} \cup \{\mathbf{p}_m\}$
21:         if $dist[i, m] < \epsilon$ then
22:             $O_c \leftarrow O_c \cup \{\mathbf{p}_m\}; \mathcal{U} \leftarrow \mathcal{U} - \{\mathbf{p}_m\}$
23:         end if
24:     until $|\mathcal{V}| == N$
25:     $\mathcal{O} \leftarrow \mathcal{O} \cup \{O_c\}$; $c \leftarrow c + 1$
26: until $|\mathcal{U}| == \emptyset$
```

A.2 Positional Encodings

We add fixed positional encondings to the last two pixel embedding dimensions, where the encondings have values from 0 to 1 according to the pixel embedding's relative location to the top and left of the image. The top-left pixel has positional encondings of values $(0, 0)$, while the bottom-right pixel has values $(1, 1)$. All other postional encondings have uniformly spaced values between $[0, 1]$ according to the height and width of the image.

A.3 Clustering Threshold

We carried out initial experiments to choose the clustering threshold. Figure 4 shows the mIoU scores on Multi-dSprites, SVHN and CLEVRTEX when we vary the negative exponent of the threshold value. We see that the value can range from 20% ($\epsilon = 1.6$ to $\epsilon = 2.0$) to 80% (about $\epsilon = 0.2$) without affecting the performance of OC-Net. As such, we set the threshold to $\epsilon = 0.7$ so that two pixels will belong to the same object if their normalized feature similarity is more than 50%.



(a) Multi-dSprites (b) SVHN (c) CLEVRTEX

Figure 4: mIoU scores with different values of clustering threshold.

A.4 Mask Information Extraction

We define the object representation as the sum of the extracted mask information and the average of the pixel embeddings in $\mathcal{O}_j$. We extract mask information by transforming the object mask $\mathbf{m}_j$ by a mask transformation matrix $\mathbf{A}$. This extraction is performed in a positionally-invariant manner by initializing the rows of $\mathbf{A}$ with a set of unit vectors which point from the object center to the border pixels. After initialization, $\mathbf{A}$ is refined via gradient descent.

Based on initial experiments shown in Table 6, when compared to random initialization (OC-Net w/ random $\mathbf{A}$ init), we see that initializing $\mathbf{A}$ with unit vectors enhances the learned representation especially in terms of object shape information.

Table 6: R^2 scores for object property prediction on simulated datasets

	Multi-dSprites			Tetrominoes-NM		
Method	Color	Position	Shape	Color	Position	Shape
OC-Net w/ random **A** init	97.7±0.8	98.0±0.1	77.1±0.0	100.0±0.0	99.1±0.2	89.6±0.0
OC-Net	**98.0±0.6**	**98.3±0.1**	**78.1±0.0**	**100.0±0.0**	**99.4±0.1**	**98.7±0.0**

A.5 Computational Resources

An 8X Tesla V100 (32GB) GPU server is used to train OC-Net and all comparison baselines for our experiments.

B Datasets

1. Multi-dSprites [23] (Apache-2.0 License): This simulated dataset consists of sprites-based images of 64×64 size with ground truth segmentation masks and sprite properties available at https://github.com/deepmind/multi_object_datasets. Each image consists of multiple oval, heart or square-shaped sprites with some occlusions. We use the variant where 2-5 randomly colored sprites appear on a randomly sampled grayscale background.

2. Tetrominoes-NM [23] (Apache-2.0 License): This dataset is a subset of the original simulated Tetrominoes dataset where each 35 × 35 image consists of 3 Tetris-like shapes sampled from 6 colors and 17 shapes. We download the data from https://github.com/deepmind/multi_object_datasets. Ground truth segmentation masks and properties are provided. We filter out images whose ground truth segmentation requires knowledge of the object shapes for testing.

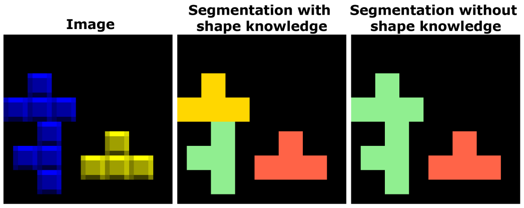


Image
Segmentation with shape knowledge
Segmentation without shape knowledge

Figure 5: Images in Tetrominoes dataset that require knowledge of object shapes for segmentation

3. Street View House Numbers (SVHN) [36] (CC0 Public Domain): This real-world dataset consists of house numbers in Google Street View images with character level ground truth bounding boxes. We use the dataset labeled as 'extra' [36], available at http://ufldl.stanford.edu/housenumbers. All original images and their ground truths are resized and cropped to size 64 × 64. The dimensions of the raw images in this set vary, hence we resize all heights of the image to 64 pixels first before cropping the width to 64. We modify the ground truth bounding boxes according to the same transformations. Since bounding boxes are provided as ground truths, we expand the object masks predicted by each method into their best-fit bounding boxes according to the masks' convex hull before computing the ARI, mDice and mIoU scores.
4. Indian Diabetic Retinopathy Image Segmentation Dataset (IDRiD) [40] (CC-BY 4.0): This is a publicly available real-world dataset with 81 retinal images where each image retina image has multiple red lesions (microaneurysms and hemorrhages) or yellow lesions (hard exudates and soft exudates). We obtain the 81 images with fine-grained segmentation ground truths from https://ieee-dataport.org/open-access/indian-diabetic-retinopathy-image-dataset-idrid. Each image has a manually labelled segmentation mask of the lesions and also the optic disc. We filter out the red lesions and resize all raw images and their ground truths to size 64 × 64. This dataset contains retina images which have more severe diabetic retinopathy, hence the number of possible lesions to be detected is large. We filter out the relatively smaller red lesions and focus the object discovery task on the optic disc and yellow lesions. We cluster the ground truth pixel masks into connected regions as the different objects to be discovered by the models.
5. CLEVRTEX [26] (CC-BY 4.0): This dataset features scenes with diverse shapes, textures and photo-mapped materials, created using physically based rendering techniques. The data is available at https://www.robots.ox.ac.uk/ vgg/data/clevrtex/. Each image contains 3-10 objects of 4 possible shapes randomly arranged on a background. The objects and backgrounds can take materials from a total of 60 possible materials. The scenes contain realistic reflections, highlights, shadows and lighting effects.
6. CLEVRTEX-OOD [26] (CC-BY 4.0): For further evaluation, we use the CLEVRTEX-OOD (out-of-distribution) test set containing 10K images with 25 new (unseen) materials and 4 new shapes (cone, torus, icosahedron, and a teapot) that are not part of CLEVRTEX. The data is available at https://www.robots.ox.ac.uk/ vgg/data/clevrtex/.
7. Flowers [37] (CC0 Public Domain): The dataset has 17 flower classes (e.g. buttercup, daffodil, iris, pansy), with photographs exhibiting typical (large) variations in viewpoint, scale, illumination and background. Segmenting such photographs is challenging due to both the variety of colours and the variety of shapes. We download the data from https://www.robots.ox.ac.uk/ vgg/data/flowers/.
8. Birds [48] (CC0 Public Domain): The Caltech-UCSD Birds-200-2011 (CUB-200-2011) dataset is the most widely-used dataset for fine-grained visual categorization task. It contains 11,788 images of 200 subcategories belonging to birds. Each image has detailed

annotations: 1 subcategory label, 15 part locations, 312 binary attributes and 1 bounding box http://www.vision.caltech.edu/datasets/cub_200_2011/.

9. COCO [32] (CC-BY 4.0): This is the Microsoft Common Objects in Context dataset used for large-scale object segmentation. We use the variant that focuses on multi-object segmentation as proposed by [50], which we download from this link https://www.dropbox.com/sh/u1p1d6hysjxqauy/AACgEh0K5ANipuIeDnmaC5mQa?dl=0

C Baseline Models

1. SLIC [1] is a clustering algorithm that clusters pixels into superpixels by using an efficient adaptation of the k-means algorithm. We use the python implementation and for each dataset, we perform a grid search for the optimal hyperparameters which produce the best results. For example, for the Tetrominoes dataset, we set the optimal clustering threshold value as 10 and the initial number of clusters as 12.
2. Felzenszwalb's Algorithm [11] is a graph-based segmentation algorithm that groups pixels together through a hand-crafted boundary detection procedure. We use the python implementation and for each dataset, we perform a grid search for the optimal hyperparameters which produce the best results. For example, for the Tetrominoes dataset, we set the optimal clustering threshold 1000, the minimum cluster size as 10, and the image smoothening value as 0.1.
3. Slot Attention [35] initializes a set of random object representations called slots which are iteratively refined by slot-normalized cross-attention on the outputs of a simple convolutional neural network (CNN). The slots are then decoded individually and combined to reconstruct the input. All Slot Attention baselines are trained with 500,000 iterations. We use the default training hyperparameters from the official reference implementation.
4. EfficientMORL [11] uses a hierarchical variational auto-encoder to extract disentangled object representations and refine the representations by a lightweight network before reconstructing the input. For all real-world and complex textures datasets, we fine-tune the per-pixel GECO reconstruction target to -2.206 which significantly outperformed the suggested settings in both the original paper and CLEVRTEX experiments[26].
5. GENESIS-V2 [13] obtains pixel embeddings through a U-Net which are then clustered using a stochastic stick-breaking process. The clusters are then decoded to reconstruct the input. We similarly fine-tune the model to use the output standard deviation of 0.7 and the equivalent per-pixel GECO reconstruction target as EfficientMORL and achieved much higher results than those reported in the CLEVRTEX paper [26].
6. SLATE [45] replaces the decoder in Slot Attention with a more expressive transformer-based autoregressive decoder conditioned on the slots. The original model uses patch sizes of 4×4 pixels as inputs and obtains poor scores for the simulated Multi-dSprites and Tetrominoes datasets which require fine-grained understanding of each pixel. Hence, we extend the model to have 1×1 input for these datasets.
7. SysBinder [46] enhances the slots of Slot Attention with factor representations called block-slots which provides within-slot disentanglement. Similar to SLATE, the original model uses patch sizes of 4×4 pixels as inputs and obtains poor scores for the simulated datasets. Hence, we extend the model to have 1×1 input for these datasets.
8. BO-QSA [22] initializes Slot Attention's object representations as learnable embeddings instead of sampling from a learnable Gaussian distribution and supplements the training with bi-level optimization. All BO-QSA baselines are trained with 500,000 iterations. We use the default training hyperparameters from the official reference implementation. We train all datasets with both the mixture-based decoder and autoregressive transformer-based decoder and report the highest scores.

D Upper Bound for Downstream Generalization Error

In this section, we present the proof of Theorem 3.1.

Proof. Let the matrix of object representations be $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_M] \in \mathbb{R}^{D \times M}$ corresponding to the object training samples $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{M}$, and let the matrix of unknown labels be $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_M]^\top \in \mathbb{R}^{M \times R}$.

Recall that $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_R]^\top \in \mathbb{R}^{R \times D}$ in Equation 4 is the minimum norm solution of the downstream predictor:

$$\mathbf{W} = \underset{\mathbf{W}'}{\text{minimize}}\, ||\mathbf{W}'||_F \text{ s.t. } \mathbf{W}' \in \underset{\hat{\mathbf{W}}}{\arg\min}\, \frac{1}{M} \sum_{i=1}^{M} ||\hat{\mathbf{W}} \cdot \mathbf{z}_i - \mathbf{y}_i||^2. \tag{11}$$

To solve for $\mathbf{W}$, we first define the vectorization:

$$\mathbf{w} = \text{vec}(\mathbf{W}) = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_R \end{bmatrix} \in \mathbb{R}^{DR}. \tag{12}$$

With this we can express:

$$\mathbf{W}\mathbf{z}_i = (\mathbf{z}_i^\top \otimes \mathbf{I})\mathbf{w} = \tilde{\mathbf{Z}}_i \mathbf{w}, \tag{13}$$

where $\otimes$ is the Kronecker product, $\tilde{\mathbf{Z}}_i = (\mathbf{z}_i^\top \otimes \mathbf{I}) \in \mathbb{R}^{R \times DR}$ and $\mathbf{I}_R \in \mathbb{R}^{R \times R}$ is the identity matrix. Then, we obtain our optimization objective

$$f(\mathbf{W}) = \sum_{i=1}^{M} ||\mathbf{W} \cdot \mathbf{z}_i - \mathbf{y}_i||^2 = \sum_{i=1}^{M} ||\mathbf{y}_i - \tilde{\mathbf{Z}}_i \mathbf{w}||^2. \tag{14}$$

Since $f(\mathbf{W})$ is convex, setting its derivative to zero obtains the following equation:

$$0 = \nabla_{\mathbf{W}} \sum_{i=1}^{M} ||\mathbf{y}_i - \tilde{\mathbf{Z}}_i \mathbf{w}||^2 = \sum_{i=1}^{M} 2 \cdot \tilde{\mathbf{Z}}_i^\top (\mathbf{y}_i - \tilde{\mathbf{Z}}_i \mathbf{w}) = \sum_{i=1}^{M} (\tilde{\mathbf{Z}}_i^\top \mathbf{y}_i - \tilde{\mathbf{Z}}_i^\top \tilde{\mathbf{Z}}_i \mathbf{w}). \tag{15}$$

From this equation we derive:

$$\sum_{i=1}^{M} \tilde{\mathbf{Z}}_i^\top \mathbf{y}_i = \sum_{i=1}^{M} \tilde{\mathbf{Z}}_i^\top \tilde{\mathbf{Z}}_i \mathbf{w} \implies \tilde{\mathbf{Z}}^\top \mathbf{y} = \tilde{\mathbf{Z}}^\top \tilde{\mathbf{Z}} \mathbf{w}, \tag{16}$$

where

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_M \end{bmatrix} \in \mathbb{R}^{MR} \quad \text{and} \quad \tilde{\mathbf{Z}} = \begin{bmatrix} \tilde{\mathbf{Z}}_1 \\ \tilde{\mathbf{Z}}_2 \\ \vdots \\ \tilde{\mathbf{Z}}_M \end{bmatrix} \in \mathbb{R}^{MR \times DR}, \tag{17}$$

and taking the pseudoinverse $(\tilde{\mathbf{Z}}^\top \tilde{\mathbf{Z}})^\dagger$, we obtain the solution:

$$\mathbf{w} = (\tilde{\mathbf{Z}}^\top \tilde{\mathbf{Z}})^\dagger \tilde{\mathbf{Z}}^\top \mathbf{y}. \tag{18}$$

Since $\tilde{\mathbf{Z}} = (\mathbf{Z}^\top \otimes \mathbf{I}_R) \in \mathbb{R}^{MR \times DR}$, we substitute this in to equation 18 and use properties of the Kronecker product to obtain:

$$\begin{aligned} \text{vec}(\mathbf{W}) = \mathbf{w} &= ((\mathbf{Z}^\top \otimes \mathbf{I}_R)^\top (\mathbf{Z}^\top \otimes \mathbf{I}_R))^\dagger (\mathbf{Z}^\top \otimes \mathbf{I}_R)^\top \mathbf{y} \\ &= ((\mathbf{Z}^\top \otimes \mathbf{I}_R)^\top (\mathbf{Z}^\top \otimes \mathbf{I}_R))^\dagger (\mathbf{Z}^\top \otimes \mathbf{I}_R)^\top \mathbf{y} \\ &= (\mathbf{Z}\mathbf{Z}^\top \otimes \mathbf{I}_R)^\dagger (\mathbf{Z} \otimes \mathbf{I}_R) \mathbf{y} \\ &= ((\mathbf{Z}\mathbf{Z}^\top)^\dagger \mathbf{Z} \otimes \mathbf{I}_R) \mathbf{y} \\ &= \text{vec}(\mathbf{Y}^\top \mathbf{Z}^\top (\mathbf{Z}\mathbf{Z}^\top)^\dagger) \end{aligned} \tag{19}$$

Therefore, denoting the unnormalized covariance as $\Sigma_{\mathbf{Z}} = \mathbf{Z}\mathbf{Z}^\top$, we have:

$$\mathbf{W} = \mathbf{Y}^\top \mathbf{Z}^\top (\mathbf{Z}\mathbf{Z}^\top)^\dagger = \mathbf{Y}^\top \mathbf{Z}^\top (\Sigma_{\mathbf{Z}})^\dagger. \tag{20}$$

Substituting this into the downstream prediction error $\delta_{\mathbf{Z}} = \frac{1}{M}\sum_{i=1}^{M} ||\mathbf{W}\cdot\mathbf{z}_i - \mathbf{y}_i||$ from equation 4, we have,

$$\begin{aligned}
\delta_{\mathbf{Z}} &= \frac{1}{M}\sum_{i=1}^{M} ||\mathbf{W}\cdot\mathbf{z}_i - \mathbf{y}_i|| \\
&= \frac{1}{M}\sum_{i=1}^{M}\sqrt{\sum_{r=1}^{R}((\mathbf{W}\cdot\mathbf{z}_i)[r] - \mathbf{y}_i[r])^2} \\
&\leq \sqrt{\frac{1}{M}\sum_{i=1}^{M}\sum_{r=1}^{R}((\mathbf{W}\cdot\mathbf{z}_i)[r] - \mathbf{y}_i[r])^2} \\
&= \frac{1}{\sqrt{M}}||\mathbf{W}\mathbf{Z} - \mathbf{Y}_{\mathcal{S}}^{\top}||_F \\
&= \frac{1}{\sqrt{M}}||\mathbf{Y}^{\top}\mathbf{Z}^{\top}(\Sigma_{\mathbf{Z}})^{\dagger}\mathbf{Z} - \mathbf{Y}_{\mathcal{S}}^{\top}||_F \qquad \text{[From Eqn (20)]} \\
&= \frac{1}{\sqrt{M}}||\mathbf{Y}^{\top}(\mathbf{Z}^{\top}(\Sigma_{\mathbf{Z}})^{\dagger}\mathbf{Z} - \mathbf{I})||_F \\
&= \frac{1}{\sqrt{M}}||(\mathbf{I} - \mathbf{Z}^{\top}(\Sigma_{\mathbf{Z}})^{\dagger}\mathbf{Z})\mathbf{Y}||_F,
\end{aligned} \tag{21}$$

where $\mathbf{I} \in \mathbb{R}^{M\times M}$.

Defining the projection matrix $\mathbf{P}_{\mathbf{Z}} = \mathbf{I} - \mathbf{Z}^{\top}(\Sigma_{\mathbf{Z}})^{\dagger}\mathbf{Z}$, we arrive at our upper bound:

$$\begin{aligned}
\delta_{\mathbf{Z}} &\leq \frac{1}{\sqrt{M}}||(\mathbf{I} - \mathbf{Z}^{\top}(\Sigma_{\mathbf{Z}})^{\dagger}\mathbf{Z})\mathbf{Y}||_F \\
&\leq \frac{1}{\sqrt{M}}||\mathbf{P}_{\mathbf{Z}}\mathbf{Y}||_F \\
&\leq ||\mathbf{P}_{\mathbf{Z}}\mathbf{Y}||_F \leq ||\mathbf{P}_{\mathbf{Z}}||_F||\mathbf{Y}||_F.
\end{aligned} \tag{22}$$

□

Elaborating on section 3.2, we minimize $\delta_{\mathbf{Z}}$ by minimizing the term $||\mathbf{P}_{\mathbf{Z}}||_F$. Since $M > D$ and $\mathbf{Z}$ is a real matrix, $||\mathbf{P}_{\mathbf{Z}}||_F$ is minimized when the row rank of $\Sigma_{\mathbf{Z}}$ is maximized [44, 3]. We note that the rank of a diagonal matrix is equal to the number of non-zero eigenvalues, and that the eigenvalues of a diagonal matrix is its diagonal entries. From this, we maximize the rank of $\Sigma_{\mathbf{Z}}$ by maximizing the values of its diagonal entries with a separation term $\mathcal{L}_{sep}$ while regularizing it to be a diagonal matrix by minimizing its off-diagonal terms with an entanglement term $\mathcal{L}_{ent}$. In this work, we demonstrate the benefits of these object-centric regularization terms as an alternative to reconstruction loss for multi-object representation learning.

E Object Property Prediction based on Learned Object Representation

Additional Scores. In Table 7, we show the additional results of the object property prediction task on the CLEVRTEX dataset.

Informativeness Scores. In Table 8, we show the detailed results of the object property prediction task on Multi-dSprites and Tetrominoes-NM.

Disentanglement and Completeness Estimates. To further evaluate the quality of the obtained object representations, we follow [34, 11] and estimate the disentanglement and completeness scores [10]. Given K properties and D object representation dimensions, we use features learned by the GBT to derive an importance matrix $\mathbf{I}$ of K rows and D columns, where every row denotes the importance of each dimension of the input object representation in predicting the property. We use this to compute the entropy of predictive importance of each dimension across all properties and

Table 7: R^2 scores for object property prediction on CLEVRTEX

Method	CLEVRTEX	
	Position	Shape
Slot Attention	47.5±19.6	30.6
EfficientMORL	21.8±2.0	18.5
GENESIS-V2	79.8±8.0	35.2
SLATE	62.0±8.0	30.5
SysBinder	38.2±3.1	29.6
BO-QSA	66.5±0.3	28.7
OC-Net	**80.7±2.4**	**36.1**

Table 8: Detailed R^2 scores for object property prediction on simulated datasets

(a) Multi-dSprites

Method	Red	Green	Blue	X-coord	Y-coord	Shape
Slot Attention	62.06	68.44	86.19	96.73	96.95	38.20
EfficientMORL	84.13	81.79	93.53	95.90	95.77	61.67
GENESIS-V2	86.40	76.08	71.73	97.57	96.55	75.82
SLATE	86.85	88.17	87.60	87.47	93.63	31.74
SysBinder	73.87	72.54	74.42	71.73	66.89	33.33
BO-QSA	97.64	94.54	96.62	97.63	97.17	75.20
OC-Net	**98.71**	**97.82**	**97.47**	**98.33**	**98.29**	**78.07**

(b) Tetrominoes-NM

Method	Red	Green	Blue	X-coord	Y-coord	Shape
Slot Attention	85.00	80.83	93.54	99.15	98.32	36.25
EfficientMORL	91.45	95.20	97.91	98.35	97.38	68.54
GENESIS-V2	84.58	94.79	84.79	96.38	92.77	37.92
SLATE	81.24	86.50	88.79	89.13	90.13	10.5
SysBinder	97.30	98.43	97.98	75.88	79.71	19.94
BO-QSA	97.30	98.43	98.65	98.79	99.03	52.46
OC-Net	**100.0**	**100.0**	**100.0**	**99.50**	**99.34**	**98.65**

define the disentanglement value as the weighted sum:

$$\mathrm{DIS}(\mathbf{I}) = \sum_{d=1}^{D} w_d(1 - \mathrm{H}(\mathbf{I}[1,d], \ldots, \mathbf{I}[K,d])), \tag{23}$$

where $\mathrm{H}(.)$ is the entropy and $w_d = \frac{\sum_{k=1}^{K} \mathbf{I}[k,d]}{\sum_{d=1}^{D}\sum_{k=1}^{K} \mathbf{I}[k,d]}$ is the weight of d. Conversely, we compute the entropy of the rate that each property is captured by a dimension and define the completeness value as:

$$\mathrm{COM}(\mathbf{I}) = \sum_{k=1}^{K} w_k(1 - \mathrm{H}(\mathbf{I}[k,1], \ldots, \mathbf{I}[k,D])), \tag{24}$$

where $w_k = \frac{\sum_{d=1}^{D} \mathbf{I}[k,d]}{\sum_{d=1}^{D}\sum_{k=1}^{K} \mathbf{I}[k,d]}$.

We show the results in Table 9. We find that the object representations obtained by OC-Net achieve both higher disentanglement and completeness than all baselines.

Table 9: Disentanglement and completeness scores for object representations

	Multi-dSprites		Tetrominoes-NM	
Method	Disentanglement	Completeness	Disentanglement	Completeness
Slot Attention	68.97	49.40	56.52	44.44
EfficientMORL	60.97	67.94	54.47	61.59
GENESIS-V2	71.59	64.70	39.58	54.13
SLATE	84.06	55.57	52.10	31.61
SysBinder	80.12	51.45	66.40	61.74
BO-QSA	90.05	77.01	69.47	59.89
OC-Net	**93.99**	**86.36**	**99.22**	**77.00**

F Additional Experiments

F.1 Additional Experiments on Quality of Discovered Objects

We further evaluate OC-Net against additional works that that explore image segmentation by modelling a graph on top of hand-crafted features and learned features.

The Normalized Cut (Ncut) algorithm [43] performs unsupervised image segmentation by treating image segmentation as a graph partitioning problem and uses a hand-crafted criterion to measure both the total dissimilarity between the different groups as well as the total similarity within the groups in order to determine the final segmentation groups.

MaskCut [47] was recently proposed to perform unsupervised image segmentation by leveraging on a pre-trained model. MaskCut first extracts learned features of the input image by using the DINO model, which was pre-trained on the ImageNet dataset using self-supervised learning techniques[7], before applying the Ncut algorithm on the extracted features to determine the segmentation.

The mIoU results in Table 10 below show that OC-Net significantly outperforms all other graph-based methods:

Table 10: mIoU scores for discovered foreground objects

Method	Multi-dSprites	Tetro-NM	SVHN	IDRiD	CTEX	CTEX-OOD
Felzenszwalb	95.0±0.0	96.9±0.0	39.8±0.0	15.4±0.0	26.8±0.0	23.4±0.0
Ncut	58.9±0.0	57.4±0.0	32.2±0.0	4.3±0.0	22.9±0.0	18.6±0.0
MaskCut	47.1±0.0	69.3±0.0	31.7±0.0	7.8±0.0	33.5±0.0	34.1±0.0
OC-Net	**99.1±0.0**	**100.0±0.0**	**49.9±0.1**	**31.2±0.2**	**37.5±0.7**	**35.0±0.6**

F.2 Experiments on Model Speed

We perform additional experiments to evaluate the speed of OC-Net. Table 11 shows the average time per iteration and the total training time of the various methods on the Multi-dSprites dataset. We train all models on an 8X Tesla V100 (32GB) GPU server.

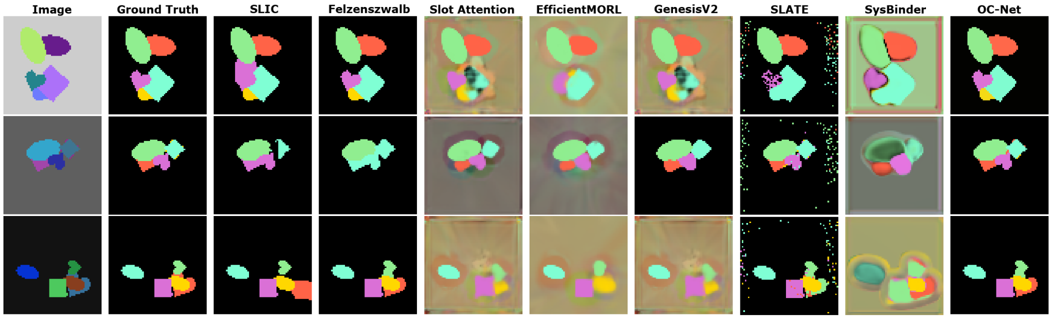
G Additional Visualizations

G.1 Object Discovery

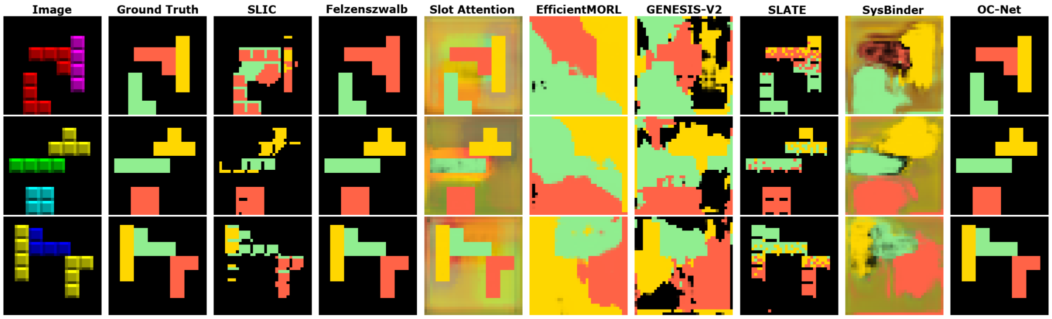
We visualize the results of additional samples from simulated datasets in Figure 6, real-world datasets in Figure 7, complex texture datasets in Figure 8 and common object datasets in Figure 9. For the common object datasets, we perform additional comparison with SLASH [27].

Table 11: Processing speed and training time across various methods on Multi-dSprites

	Multi-dSprites	
Method	Time / Iteration (ms)	Training Time (hours)
Slot Attention	75	5.26
EfficientMORL	139	11.60
GENESIS-V2	132	11.07
SLATE	385	21.4
SysBinder	235	13.06
BO-QSA	66	4.63
OC-Net	137	0.17

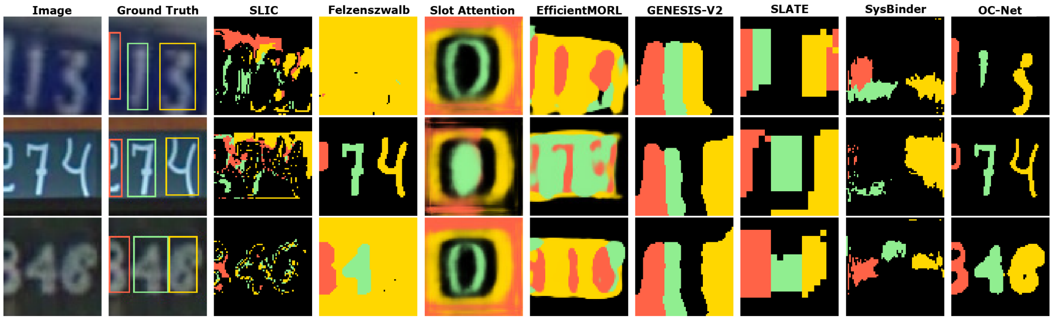


(a) Multi-dSprites

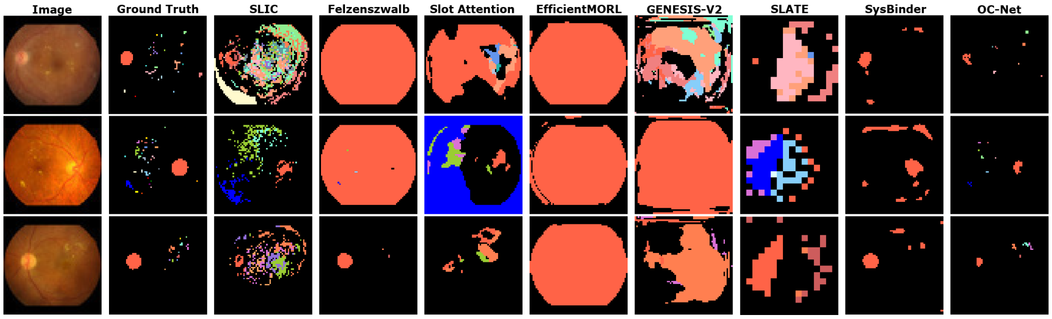


(b) Tetrominoes-NM

Figure 6: Supplementary visualization of discovered objects on simulated datasets.

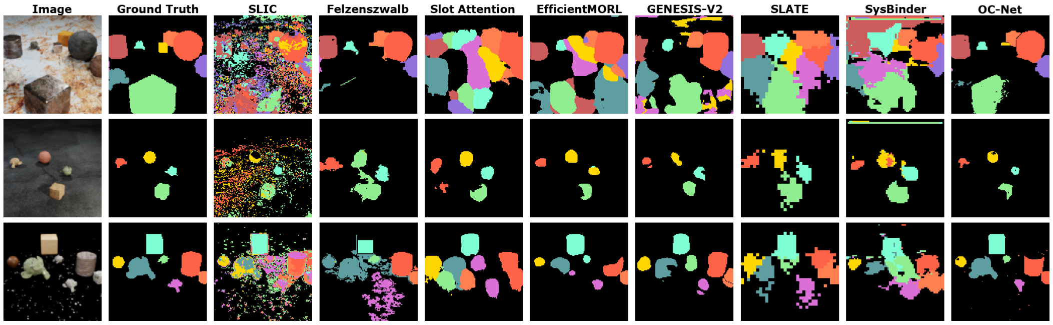


(a) SVHN

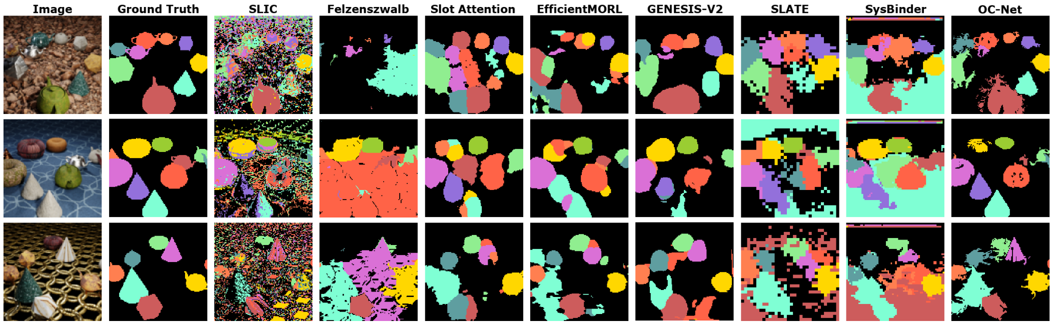


(b) IDRiD

Figure 7: Supplementary visualization of discovered objects on real-world datasets.

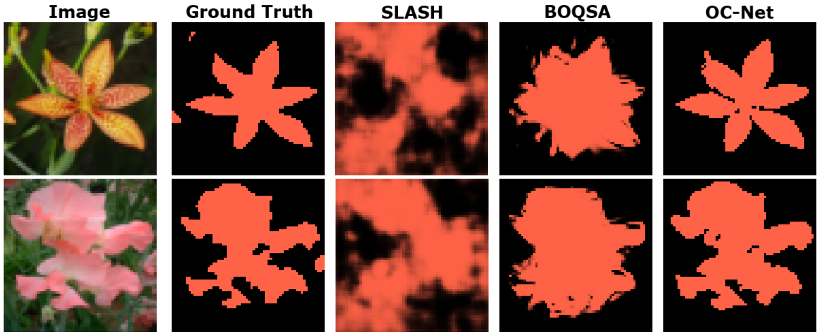


(a) CLEVRTEX

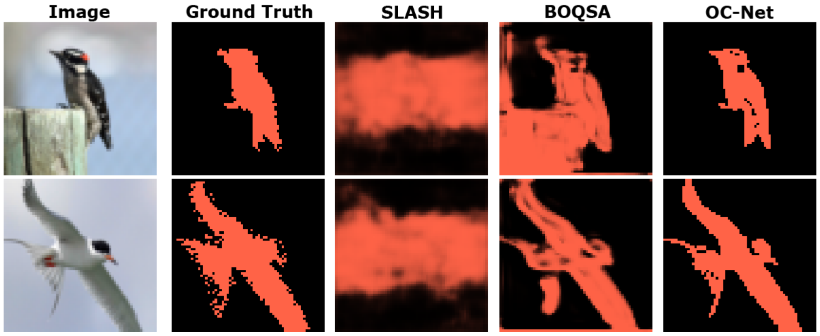


(b) CLEVRTEX-OOD

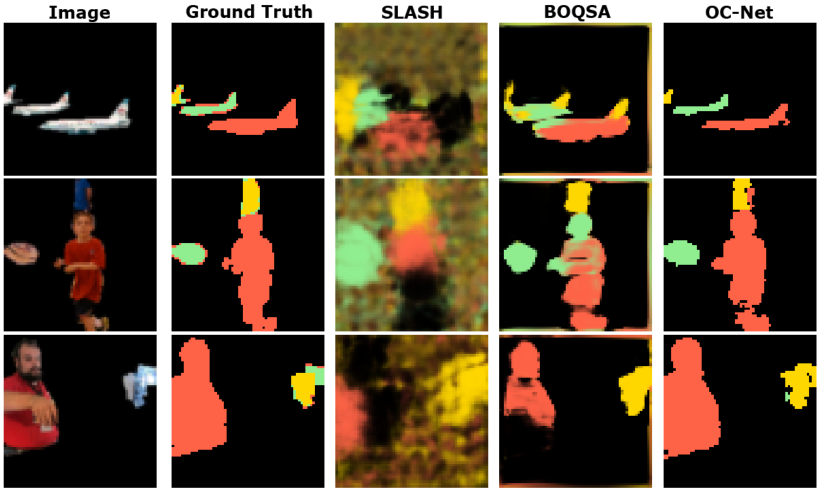
Figure 8: Supplementary visualization of discovered objects on complex texture datasets.



(a) Flowers



(b) Birds



(c) COCO

Figure 9: Supplementary visualization of discovered objects on common object datasets.

G.2 Model Generalizability

We visualize the results of training on Multi-dSprites and testing on Tetrominoes-NM in Figure 10, IDRiD in Figure 11, SVHN in Figure 12, CLEVRTEX in Figure 13 and CLEVRTEX-OOD in Figure 14.

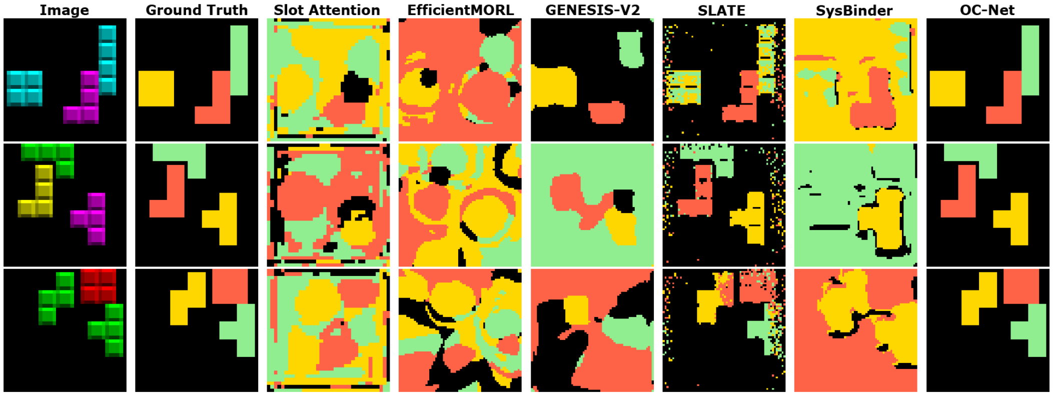


Figure 10: Visualization of model generalizability on Tetrominoes-NM after training on Multi-dSprites.

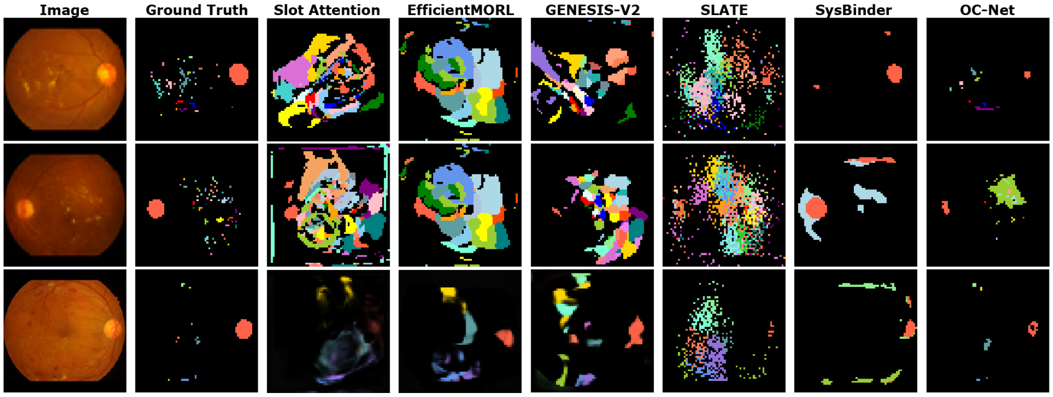


Figure 11: Visualization of model generalizability on IDRiD after training on Multi-dSprites.

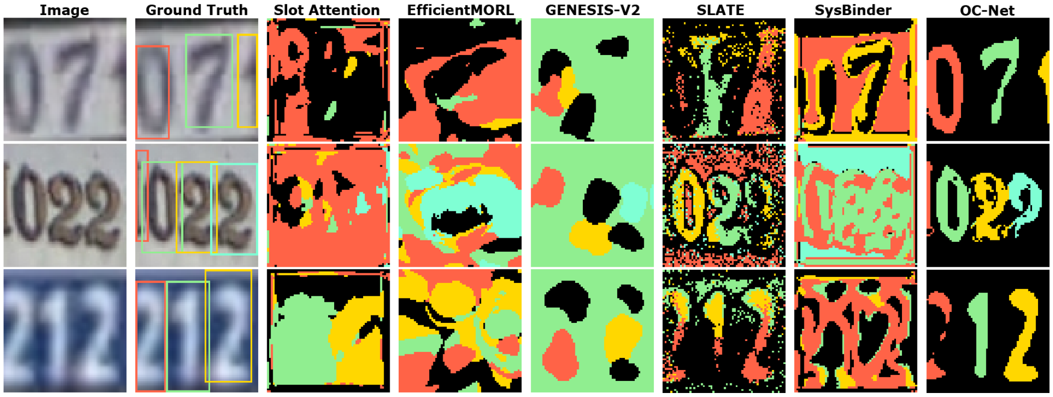


Figure 12: Visualization of model generalizability on SVHN after training on Multi-dSprites.

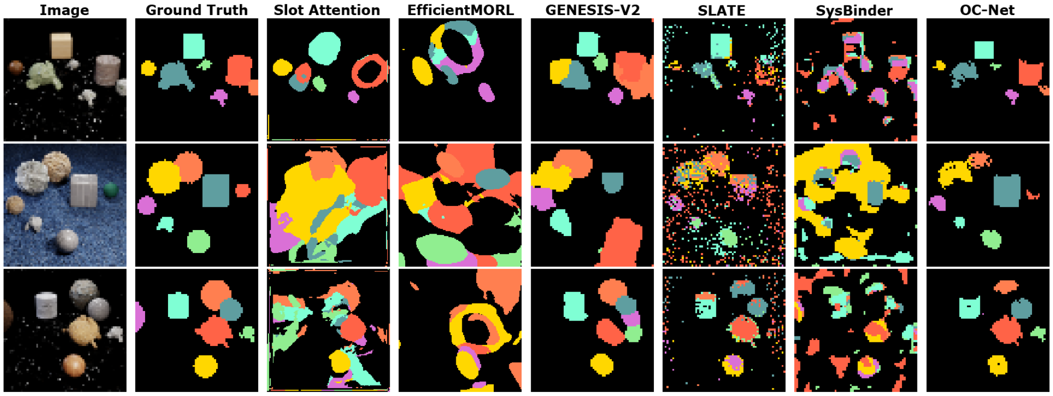


Figure 13: Visualization of model generalizability on CLEVRTEX after training on Multi-dSprites.

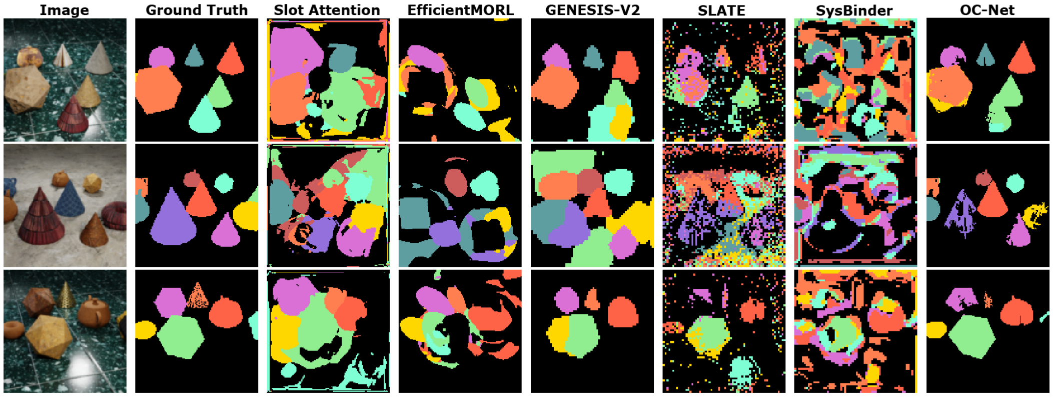


Figure 14: Visualization of model generalizability on CLEVRTEX-OOD after training on Multi-dSprites.

H Additional Discussions and Limitations

H.1 Preservation of Pixel Information for Fine-Grained Object Discovery

OC-Net uses a single 1×1 convolutional layer to obtain pixel embeddings which are individually refined and clustered before the groups of pixel embeddings are processed into object representations. In contrast, existing state-of-the-art methods such as Slot Attention, GENESIS-V2 and EfficientMORL all use encoder networks with multiple larger sized 3×3 or 5×5 convolutions. Larger sized convolutions are very popular in general computer vision where they are typically used for edge detection especially in the early layers of a deep neural network. For fine-grained segmentation without labels, however, we show the superiority of using paths between pixels to perform both edge/border detection and also segmentation of image components whose borders stretch outside of the image, such as backgrounds. Considering that almost all state-of-the-art methods in multi-object representation learning use larger-sized convolutions as the default, we hypothesize that further exploration of these insights will prove useful to the advancement of this field.

H.2 Generative vs Latent-Space Regularization Approach for Object-Centric Learning

In contrast to the majority of existing methods, in this work we consider a regularization-based approach and focus on the object discovery process instead of optimizing for component-based image reconstruction.

Optimizing for reconstruction demands much more training samples and runtime to ensure that suitable distributions are learned for precise reconstruction. If we assume the usual input image of size $H \times W$ and having 3 channels for Red, Green and Blue, this amounts to a reconstruction objective with at least $H \times W \times 3 \times 255$ possible permutations. When $H = W = 64$, this amounts to more than 3 million possible outputs. Early works suggested that component-based reconstruction offered an efficient alternative to full-scene image generation if a common repeating structure across samples could be exploited [5]. Inspired by this, existing state-of-the-art works such as EfficientMORL and GENESIS-V2 attempted to handle the large generation space via a combination of component-based reconstruction and constraints such as defining a learnable Gaussian distribution for generated pixels, and use the Generalized ELBO with Constrained Optimization (GECO) objective [41] to assist in stable convergence for multiple initialization seeds, since GECO reformulates the ELBO to allow the Kullback–Leibler (KL) divergence to grow large so that a predefined reconstruction threshold can first be attained. While these tactics are often effective in stabilizing the training of generative networks, they present fundamental limitations for the task of object discovery since the optimization objective primarily aims to reconstruct the input, which may often cause the components that make up the reconstructed image to be simply the most efficient image partitioning instead of semantically meaningful segments [35, 9].

To address these fundamental limitations, we instead proposed a regularization-based approach where feature connectivity guides the discovery of objects and two regularization terms that refine the representation space of the discovered objects. Such regularization-based techniques have been explored in the self-supervised learning domain [3]. We show that by focusing on the quality of objects discovered and their respective representations, such a regularization-based approach can better handle the diversity in real-world scenes in the task of object discovery.

H.3 Limitation: Memorization of Object Shapes

The original Tetrominoes dataset [23] includes samples that require the knowledge of object shapes in order to accurately separate the objects. This reveals a key limitation to our approach especially in cases where the task requires explicit memorization of objects, such as by requesting for a precise generation of an object that was seen previously. One straightforward solution could be to extend OC-Net with a dictionary of object prototypes in order to bridge this gap [51]. However, as discussed in [25] and as demonstrated in this work, demanding the memorization of object shapes in a model could be a barrier to generalizable scene decompositions outside of the training distribution, and hence in this work we decided to focus on generalizable object discovery. Further, state-of-the-art methods follow an implicit memorization approach by attempting to learn patterns within the data distribution during training, which proves to be a fundamental limitation when facing the diversity of image distributions in the real-world. Therefore, there remains fundamental challenges related to the successful design of a visual memory module which can handle both the complexity of real-world images and the complex part-whole hierarchy of objects, which we leave for future work.

I Ethics Statement

Our analysis, which is focused on publicly available multi-object simulated, real-world and complex texture data has no immediate impact on general society. To the best of our knowledge, none of the datasets used in this study contain personally identifiable information or offensive content. As with any model that performs scene understanding, applications with potential negative societal impact such as in the area of surveillance cannot be fully excluded upon future research in this area. At this point in time, however, there remains challenging open questions that need to be reliably addressed before enabling the deployment of OC-Net and its counterparts to real-world settings, and hence a direct application of this method for malicious purposes is currently unlikely.

J Third-Party Assets

OC-Net is implemented using PyTorch [39].In addition to various open-sourced Python packages, we make use of the following third-party assets:

- Locatello et al. [35] (Apache-2.0 license): Implementation of Slot Attention[1],
- Emami et al. [11] (MIT license): Implementation of EfficientMORL[2],
- Engelcke et al. [13] (GPL-3.0 license): Implementation of GENESIS-V2[3],
- Singh et al. [45] (MIT license): Implementation of SLATE[4],
- Singh et al. [46] (MIT license): Implementation of SysBinder[5].
- Jia et al. [22]: Implementation of BO-QSA[6].

---

[1] https://github.com/google-research/google-research/tree/master/slot_attention

[2] https://github.com/pemami4911/EfficientMORL

[3] https://github.com/applied-ai-lab/genesis

[4] https://github.com/singhgautam/slate

[5] https://github.com/singhgautam/sysbinder

[6] https://github.com/YuLiu-LY/BO-QSA