Unsupervised Multi-Object Segmentation Using Attention and Soft-Argmax

Bruno Sauvalle, Arnaud de La Fortelle

Unsupervised multi-object segmentation using attention and soft-argmax

Bruno Sauvalle Arnaud de La Fortelle
Centre de Robotique, Mines ParisTech PSL University
{bruno.sauvalle,arnaud.de_la_fortelle}@mines-paristech.fr

Abstract

We introduce a new architecture for unsupervised object-centric representation learning and multi-object detection and segmentation, which uses a translation-equivariant attention mechanism to predict the coordinates of the objects present in the scene and to associate a feature vector to each object. A transformer encoder handles occlusions and redundant detections, and a convolutional autoencoder is in charge of background reconstruction. We show that this architecture significantly outperforms the state of the art on complex synthetic benchmarks.

1. Introduction

We consider in this paper the tasks of object-centric representation learning and unsupervised object detection and segmentation: Starting from a dataset of images showing various scenes cluttered with objects, our goal is to build a structured object-centric representation of these scenes, i.e. to map each object present in a scene to a vector representing this object and allowing to recover its appearance and segmentation mask. This task is very challenging because the objects appearing in the images may have different shapes, locations, colors or textures, can occlude each other, and we do not assume that the images share the same background. However the rewards of object-centric representations could be significant since they allow to perform complex reasoning on images or videos [11, 39] and to learn better policies on downstream tasks involving object manipulation or localization [42, 46]. The main issue with object-representation learning today is however that existing models are able to process synthetic toy scenes with simple textures and backgrounds but fail to handle more complex or real-world scenes [27].

We propose to improve upon this situation by introducing a translation-equivariant and attention-based approach for unsupervised object detection, so that a translation of the input image leads to a similar translation of the coordinates of the detected objects, thanks to an attention map which is used not only to associate a feature vector to each object present in the scene, but also to predict the coordinates of these objects.

The main contributions of this paper are the following:

- We propose a theoretical justification for the use of attention maps and soft-argmax for object localization.
- We introduce a new translation-equivariant and attention-based object detection and segmentation architecture which does not rely on any spatial prior.
- We show that the proposed model substantially improves upon the state of the art on unsupervised object segmentation on complex synthetic benchmarks.

The paper is organized as follows: In section 2, we provide some theoretical motivation for using attention maps and soft-argmax for object localization. In section 3, we review related work on unsupervised object instance segmentation. In section 4 we describe the proposed model. Experimental results are then provided in section 5.

2. Motivation for using attention maps and soft-argmax for object localization

It is widely recognized that the success of convolutional neural networks is associated with the fact that convolution layers are equivariant with respect to the action of the group of translations, which makes these layers efficient for detecting features which naturally have this property. It is also easy to show that linear convolution operators are the only linear operators which are equivariant with respect to the natural action of the translation group on feature maps.

We introduce the following notations to describe the action of the translation group: We consider a grayscale image as a scalar-valued function $\varphi(i, j)$ defined on $\mathbb{Z}^2$ and an element of the group of translations as a vector (u, v) in $\mathbb{Z}^2$. The natural action T of the group of translations on an image can be described by the formula

$$T_{u,v}(\varphi)(i, j) = \varphi(i - u, j - v). \tag{1}$$

A model layer L is called equivariant with respect to translations if it satisfies

$$L(T_{u,v}\varphi) = T_{u,v}(L(\varphi)). \tag{2}$$

Let's now consider a localization model M which takes as input an image $\varphi(i,j)$ showing one object and produces as output the coordinates of the object present in this image. Such a model does not produce a feature map, so that the previous definition of translation equivariance can not be used for this model. We remark however that the group of translations acts naturally on $\mathbb{Z}^2$ by the action $T'_{u,v}(i,j) = i+u, j+v$, and that the model M should have the equivariance property

$$M(T_{u,v}\varphi) = T'_{u,v}(M(\varphi)). \tag{3}$$

Indeed, if the complete image is translated by a vector (u,v), then the object present in this image is also translated, so that the associated coordinates have to be shifted according to the vector (u,v).

It is not difficult to see that in the same way that convolutional operators are the only linear operators equivariant with respect to translations, it is also possible to fully describe which elementary operators follow this specific equivariance property. We first remark however that we have to restrict the space of possible input maps φ: if φ is a constant function, it does not change under the action of the translation group, so that the equivariance property 3 cannot be satisfied with such a function. We then suppose that φ satisfies $\sum_p \varphi(p) = 1$ and consider that the domain of the operator M is the corresponding affine space $\mathcal{A}$. We also replace the linearity condition by an the following affinity condition:

For all $\alpha_i \in \mathbb{R}, \varphi_i \in \mathcal{A}$ so that $\sum_i \alpha_i = 1$, we have $M(\sum_i \alpha_i \varphi_i) = \sum_i \alpha_i M(\varphi_i)$.

We then have the following proposition:

Proposition 2.1 *An affine operator M which satisfies the equivariance property 3 has to be of the form*

$$M(\varphi) = C + \sum_{p\in\mathbb{Z}^2} \varphi(p)p \tag{4}$$

for some constant C in $\mathbb{R}^2$.

Proof: We write the input map φ as a sum of spatially shifted version of the function $\delta \in \mathcal{A}$ satisfying $\delta(p) = 1$ for $p = (0,0)$ and $\delta(p) = 0$ for $p \neq (0,0)$:

$$\varphi(p) = \sum_{q\in\mathbb{Z}^2} \varphi(q)\delta(p-q) \tag{5}$$

We then use the the affine property of M and equivariance property 3:

$$M(\varphi) = M(\sum_q \varphi(q)\delta(p-q)) \tag{6}$$

$$= \sum_q \varphi(q) M(\delta(p-q)) = \sum_q \varphi(q)(M(\delta)+q) \tag{7}$$

$$= (\sum_q \varphi(q))M(\delta) + \sum_q \varphi(q)q \tag{8}$$

$$= M(\delta) + \sum_q \varphi(q)q, \tag{9}$$

which proves the proposition since $M(\delta)$ is a constant.

The proposition 2.1 can be interpreted as stating that in order to get an equivariant localization operator, the most straightforward method is to build a normalized attention map φ from the input image and compute the coordinates of the detected object using an attention mechanism with φ as attention map and pixel coordinates as target values. One remarks that it is precisely what the soft-argmax operator is doing: It takes an unnormalized scalar map ϕ as input, normalizes it using a softmax operator, and then perform localization using the same formula as in 2.1:

$$\begin{aligned} \text{soft-argmax}(\phi) &= \sum_{p\in\mathbb{Z}^2} \text{softmax}(\phi)(p)p \\ &= \sum_{p\in\mathbb{Z}^2} \frac{e^{\phi(p)}}{\sum_{q\in\mathbb{Z}^2} e^{\phi(q)}} p \end{aligned} \tag{10}$$

This operation is called soft-argmax because it allows to compute in a differentiable way an estimate of the coordinates of the maximum of the input map ϕ. Using soft-argmax then appears to be the most natural way to get an equivariant localization operator.

3. Related work

Unsupervised object detection and segmentation Unsupervised object detection and segmentation models are generally reconstruction models: They try to reconstruct the input image using a specific image rendering process which induces the required object-centric structure. In order to ensure that objects are properly detected, various objectness priors have been defined and implemented:

- pixel similarity priors. Some models consider the task of object segmentation as a clustering problem, which can be addressed using deterministic [22, 31] or probabilistic [15, 20, 41] methods: If the feature vectors associated to two different pixels of an image are very similar, then it is considered that these pixels should both belong to the same object or to the background.

- independence priors. Some models assume that the images are sampled from a distribution which follows a probabilistic model featuring some independence priors between objects and the background, and use variational [19, 16] or adversarial [7, 3] methods to learn these distributions.

- disentanglement of appearance and location. Foreground objects appearing in the scenes of a given dataset can have similar shapes and appearances but very different scales and locations. Object discovery is performed by disentangling the object appearance generation process, which is performed by a convolutional glimpse generator [1, 28, 10, 37, 24, 23] or a learned dictionary [32, 36], from the translation and scaling of the objects appearing in a scene, which is usually done by including a spatial transformer network [**?**] in the model. The model described in this paper belongs to this category and uses an convolutional glimpse generator.

Object detection and segmentation without spatial prior State of the art supervised detection and segmentation models usually rely on predefined reference anchors or center points which are spatially organized according to a periodic grid structure. The use of periodic grids has also been proposed for unsupervised object detection [30, 24, 23, 36]. Alternative detection methods relying on heatmaps produced by a U-net [34] or stacked U-nets [33] networks, which predict for each pixel the probability of presence of one object on this pixel have been implemented in the supervised setting [29, 13]. For some specific applications such as human pose estimation or anatomical landmark localization [40], some supervised models predict one heatmap per object. The use of soft-argmax for converting heatmaps to object coordinates has been implemented in the supervised [38, **?**, 6], semi-supervised [**?**] and unsupervised settings [18, 17] but has never been proposed for unsupervised object detection or segmentation. More recently, transformer-based [**?**] models using object [5, 47, 12] or mask [9, 8] queries have been proposed which not not rely explicitly on a spatial grid. These models show that transformers are efficient in the supervised setting to avoid multiple detections of the same object.

4. Description of proposed model

4.1. Model architecture

The overall architecture of the model is described in Fig 1.

The proposed model is composed of a a foreground model and a background model.

The background model is a deterministic convolutional autoencoder: We rely on the classical assumption [43] that background images lie on a low-dimensional manifold, and use the autoencoder to learn this manifold.

The foreground model is also deterministic and associates to each object in the scene an appearance vector z^{what} which is used to produce a glimpse of the object, which is then scaled and translated at the right position on the image using a spatial transformer network.

The foreground encoding and reconstruction process can be described as follows: First, a high resolution feature map generator takes a color image of size $h \times w$ as input and produces a high resolution feature map Φ of dimension d_Φ and several scalar attention logit maps $A_1, ..., A_K$. We will use in this paper the transformer-based Segformer model [45], which produces feature maps of size $h^* \times w^* = h/4 \times w/4$. The hyperparameter K is set to the maximum number of objects on a scene in the dataset. The scalar attention logit maps $A_1, ..., A_K$ are transformed into a normalized attention maps $\mathcal{A}_1, ..., \mathcal{A}_K$ using a softmax operator:

$$\mathcal{A}_k(i,j) = \frac{e^{A_k(i,j)}}{\sum_{i',j'} e^{A_k(i',j')}} \tag{11}$$

We normalize the pixel indices (i, j) from the range $[1, .., w^*]$ and $[1, .., h^*]$ to the range $[-1, 1]$ required by spatial transformer networks using the formulas

$$x(i) = 2\frac{i-1}{w^*-1} - 1 \tag{12}$$

$$y(j) = 2\frac{j-1}{h^*-1} - 1, \tag{13}$$

and predict initial estimates x_k^0, y_k^0 of the coordinates of the detected objects as the the center of mass of the attention maps $\mathcal{A}_k$:

$$x_k^0 = \sum_{i=1,j=1}^{w^*,h^*} \mathcal{A}_k(i,j)x(i) \tag{14}$$

$$y_k^0 = \sum_{i=1,j=1}^{w^*,h^*} \mathcal{A}_k(i,j)y(j) \tag{15}$$

We also build K object query feature vectors $\phi_1^0, ..., \phi_K^0$ of dimension d_Φ using the same attention maps $\mathcal{A}_1, .., \mathcal{A}_K$ as weights and the feature map Φ as target values:

$$\phi_k^0 = \sum_{i=1,j=1}^{w^*,h^*} \mathcal{A}_k(i,j)\Phi(i,j) \tag{16}$$

A transformer encoder then takes the K triplets $(\phi_k^0, x_k^0, y_k^0)_{1 \leq k \leq K}$ as inputs and produces a refined version $(\phi_k, x_k, y_k)_{1 \leq k \leq K}$ taking into account possible detection redundancies and object occlusions. More precisely, we use a learned linear embedding to increase the dimension of the triplets (ϕ_k^0, x_k^0, y_k^0) from $d_\Phi + 2$ to the input dimension d_T of the transformer encoder, and a learned linear projection to reduce the dimension of the outputs of the transformer encoder from d_T back to $d_\Phi + 2$. The transformer encoder does not take any positional encoding as input, considering that the transformation which has to be performed should not depend on the ordering of the detections. We force the

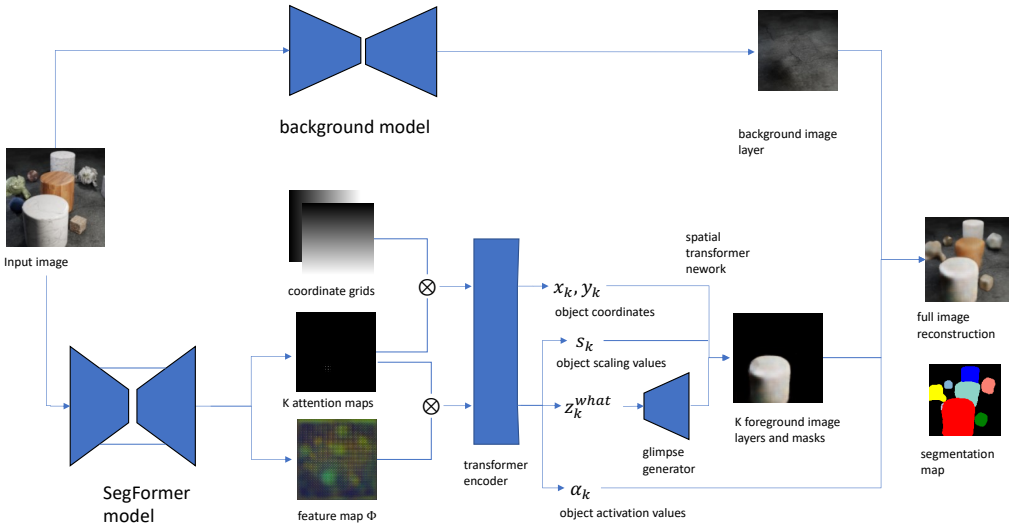

Figure 1. **Overview of proposed model**. A High resolution feature map generator (Segformer model) is trained to produce a high resolution feature map Φ and K scalar attention maps (one per object query). These maps are used to predict the coordinates and scales of the detected objects and the associated feature vectors, which are refined by a transformer encoder and then used as inputs to a glimpse generator and a spatial transformer network to produce K object image layers and masks. A convolutional autoencoder is in charge of background reconstruction.

final values of x_k and y_k to stay in the range $[-1, 1]$ using clamping. Each transformed feature vector ϕ_k is then split in three terms: $\phi_k = (s_k, \alpha_k, z_k^{what})$.

- The first term s_k is an inverse scaling factor. It is a scalar if objects in the dataset have widths and heights which are similar (isotropic scaling), or a pair of scalars s_k^x, x_k^y if this is not the case (anisotropic scaling). We force the values of s_k to stay within a fixed range using a sigmoid function. The maximum value of this range ensures that a non-zero gradient will be available. The minimum value is set higher than 1 to make sure that the glimpse generator will not try to generate a full image layer.
- The second term is a scalar which is assumed to predict the activation level α_k of the object, which will be used to predict whether it is visible or not. We force this activation value to be positive using an exponential map.
- The remaining coordinates form a vector z_k^{what} which codes for the appearance of the object.

We then use a convolutional glimpse generator to build a color image o_k of the associated object together with the associated scalar mask m_k, using z_k^{what} as input. These images and masks are translated to the positions (x_k, y_k) and scaled according to the inverse scaling factor s_k using a spatial transformer network. We note L_k and M_k for $k \in \{1, .., K\}$ the corresponding object image layers and masks, and L_0 the background image produced by the background model, so that we have a total of $K + 1$ image layers.

We now have to decide for each pixel whether this pixel should show the background layer or one of the K object layers. In order to do this in a differentiable way, we multiply the predicted object masks M_k with the associated object activation levels α_k, and normalize the results to get one normalized weights distribution $(w_k)_{0 \leq k \leq K}$ per pixel:

$$w_k(i,j) = \frac{\alpha_k M_k(i,j)}{\sum_{k' \in 0..K} \alpha_{k'} M_{k'}(i,j)}, \tag{17}$$

considering that the mask M_0 associated to the background is set to 1 everywhere and that it has a fixed learned activation factor α_0.

The final reconstructed image $\hat{X}$ is then equal to the weighted sum of the various image layers using the weights w_k:

$$\hat{X}(i,j) = \sum_{k=0}^{K} w_k(i,j) L_k(i,j) \tag{18}$$

During inference the segmentation map is built by assigning to each pixel the layer index $k \in \{0, .., K\}$ for which $w_k(i,j)$ is the maximum. The background model is not needed to get the segmentation maps during inference.

4.2. Model training

4.2.1 loss function

In order to train the proposed model, we use a main reconstruction loss function and an auxiliary loss:

reconstruction loss The local L_1 reconstruction error associated to the pixel (i, j) is

$$l_{i,j} = \sum_{c=1}^{3} |\hat{x}_{c,i,j} - x_{c,i,j}|, \tag{19}$$

where $x_{c,i,j}$ and $\hat{x}_{c,i,j}$ are the values of the color channel c at the position (i, j) in the input image and reconstructed image.

The reconstruction loss is defined as the mean square of this reconstruction error.

$$\mathcal{L}_{rec} = \frac{1}{hw} \sum_{i=1,j=1}^{w,h} l_{i,j}^2 \tag{20}$$

pixel entropy loss For a given pixel (i, j), we expect the distribution of the weights $w_0(i, j), .., w_K(i, j)$ to be one-hot, because we assume that the objects are opaque. We observe that a discrete distribution is one-hot if and only if it has a zero entropy, so that minimizing the entropy of this distribution would be a reasonable way to enforce a stick-breaking process. Considering however that the entropy function has a singular gradient near one-hot distributions, we use the square of the entropy function to build the loss function. We then define the pixel entropy loss as

$$\mathcal{L}_{pixel} = \frac{1}{hw} \sum_{i=1,j=1}^{w,h} (\sum_{k=0}^{K} w_k(i, j) \log(w_k(i, j) + \epsilon))^2, \tag{21}$$

where $\epsilon = 10^{-20}$ is introduced to avoid any numerical issue with the logarithm function.

This auxiliary loss is weighted using the weight λ_{pixel} before being added to the reconstruction loss. During our experiments, we observed that the pixel entropy loss could prevent a successful initialization of the localization process during the beginning of the training. As a consequence, we smoothly activate this auxiliary loss during initialization using a quadratic warmup of the weight.

The full loss function is then equal to

$$\mathcal{L} = \mathcal{L}_{rec} + \min(1, \frac{step}{N_{pixel}})^2 \lambda_{pixel} \mathcal{L}_{pixel}, \tag{22}$$

where $step$ is the current training iteration index and N_{pixel} is a fixed hyperparameter.

4.2.2 curriculum training

The interaction between the background reconstruction model and the foreground model during training is a very challenging issue, because of the competition between them to reconstruct the image. We handle this problem as in [23] by implementing curriculum training. We will then evaluate two methods to train the proposed model:

- baseline training (BT) : The background and foreground models are initialized randomly and trained simultaneously.
- curriculum training (CT): The training of the model is split in three phases :
  1. The background model is pretrained alone, using the methodology and robust loss function described in [35].
  2. The weights of the background model are then frozen and the foreground model is trained using the frozen background model.
  3. The background and foreground models are then fine-tuned simultaneously.

5. Experimental results

5.1. Evaluation on public benchmarks

We perform a quantitative evaluation of the proposed model on the following datasets: CLEVRTEX [27], CLEVR [25], ShapeStacks [21] and ObjectsRoom [26].

We implement on ShapeStacks, ObjectsRoom and CLEVR the same preprocessing as in [15]. We use the same hyperparameter values on these datasets, except for the hyperparameter K related to the number of object queries, which is set to the maximum number of objects in each dataset (i.e. 3 on ObjectsRoom, 6 on ShapeStacks and 10 on CLEVRTEX and CLEVR). We use isotropic scaling on CLEVR and ShapeStacks and anisotropic scaling on the other datasets.

We use the versions B3 of the Segformer model, and rely on the Hugging Face implementation of this model, with pretrained weights on ImageNet-1k for the hierarchical transformer backbone, but random initialization for the MLP decoder which is used as a feature map generator. We use the standard Pytorch implementation of the transformer encoder. The architecture of the backgroud model autoencoder is the same as in [35]. The glimpse generator is a sequence of transpose convolution layers, group normalization [44] layers and CELU [2] non-linearities, and is described in the supplementary material.

We use Adam as optimizer. The training process includes a quadratic warmup of the learning rate since the model contains a transformer encoder. We also decrease the learning rate by a factor of 10 when the number of training steps reaches 90% of the total number of training steps. The total number of training steps of the baseline training (BT) scenario is 125 000. In the curriculum training (CT) scenario, the number of training steps for background model pretraining (phase 1) is 500 000 on CLEVRTEX, ShapeStacks and ObjectsRoom, but 2500 on CLEVR, which shows a fixed background, as recommended in [35]. The number

of training steps of phase 2 (training with frozen pretrained background model) is 30 000, and the number of training steps of the final fine-tuning phase (phase 3) is 95 000.

Full implementation details and hyperparameter values are provided in the supplementary material, and the model code will be made available on the Github platform.

In order to compare our results with published models, we compute the following evaluation metrics: mean intersection over union (mIoU) and adjusted rand index restricted to foreground objects (ARI-FG). We also provide the mean square error (MSE) between the reconstructed image and the input image, which provides an estimate of the accuracy of the learnt representation. We use the same definitions and methodology as [27] for these metrics. We provide the mean segmentation covering (defined in [16]) restricted to foreground objects (MSC-FG) on ObjectsRoom and ShapeStacks where mIoU baseline values are not available.

We call AST-Seg (Attention and Soft-argmax with Transformer using Segformer) the proposed model, and AST-Seg-B3-BT, AST-Seg-B3-CT respectively the models using a Segformer B3 feature map generator trained under the baseline training or curriculum training scenarios.Table 1 and 2 provide the results obtained on these datasets with a comparison with published results.

The proposed model trained under the baseline training scenario gets better average results than existing models on the CLEVR and CLEVRTEX dataset, but shows a very high variance. For example, on the CLEVR dataset, the model may fall during training in a bad minimum where the background model tries to predict the foreground objects. Using curriculum training allows to avoid this issue, get stable results on all datasets, and obtain a very significant mIoU improvement on the most complex datasets CLEVR and CLEVRTEX.

Following the methodology proposed in [27], we also evaluated the generalization capability of a model trained on CLEVRTEX when applied to datasets containing out of distribution images showing unseen textures and shapes or camouflaged objects (OOD and CAMO datasets [27]). The results of this evaluation are provided in Table 3 and show that the proposed model generalizes well, although it is deterministic and does not use any specific regularization scheme.

Some segmentation prediction samples are provided in Fig 8. Other image samples are available in the supplementary material.The main limitation of the proposed model is the management of shadows, which may be considered by the model as separate objects or integrated to object segmentations.

5.2. Ablation study and additional experiments

We provide in Table 4 results obtained using various ablations or modifications on the model architecture or loss function, which show that:

- The model remains competitive if the transformer encoder is removed by setting $(\phi_k, x_k, y_k)_{1\leq k\leq K} = (\phi_k^0, x_k^0, y_k^0)_{1\leq k\leq K}$. The results on the ShapeStacks and ObjectsRoom datasets are even improved with this simplified architecture, with a surprisingly strong improvement on the Shapestacks dataset, which shows the efficiency of the attention and soft-argmax mechanism. The transformer encoder is however necessary on the more complex CLEVR and CLEVRTEX datasets.

- Training with a number of slots slightly higher than the maximum number of objects does not lead to significant changes in the results. A more substantial increase of the number of slots however leads to poor results on scenes with complex textures due to the increasing fragmentation the objects. This is very different from the situation observed on query-based supervised detection models like DETR, where the number of queries has to be very high compared to the number of objects.

- It is possible to replace the Segformer high resolution feature map generator with any other generator. The proposed model was originally designed with a custom Unet feature map generator, which gets similar results as the Segformer model on CLEVR, ShapeStacks and ObjectsRoom, but underperforms on the more complex CLEVRTEX dataset. The architecture of this Unet is described in the supplementary material.

- Using a pretrained backbone is necessary to get good performances with a Segformer feature map generator.

- We tested an alternative training scenario where the background model remains frozen during the complete training of the foreground model (125 000 iterations). The main advantage of this scenario is that it is significantly faster and requires less memory, since the backgrounds of the training images can be pre-computed and memorized. The accuracy of the results is however lower than the curriculum training scenario proposed in this paper, except for the ObjectsRoom dataset.

- Switching between isotropic scaling and anisotropic scaling does not make much difference, except for the ShapeStacks dataset, where the proposed model can consider that each block tower is a single object if anisotropic scaling is enabled.

Table 1. Benchmark results on CLEVR and CLEVRTEX. Results are shown ($\pm\sigma$) calculated over 3 runs. Source: [27]

Model	CLEVR			CLEVRTEX		
	↑mIoU (%)	↑ARI-FG (%)	↓MSE	↑mIoU (%)	↑ARI-FG (%)	↓MSE
SPAIR [10]	65.95 ± 4.02	77.13 ± 1.92	55 ± 10	0.00 ± 0.00	0.00 ± 0.00	1101 ± 2
SPACE [30]	26.31 ± 12.93	22.75 ± 14.04	63 ± 3	9.14 ± 3.46	17.53 ± 4.13	298 ± 80
GNM [23]	59.92 ± 3.72	65.05 ± 4.19	43 ± 3	42.25 ± 0.18	53.37 ± 0.67	383 ± 2
MN [36]	56.81 ± 0.40	72.12 ± 0.64	75 ± 1	10.46 ± 0.10	38.31 ± 0.70	335 ± 1
DTI [32]	48.74 ± 2.17	89.54 ± 1.44	77 ± 12	33.79 ± 1.30	79.90 ± 1.37	438 ± 22
Gen-V2 [15]	9.48 ± 0.55	57.90 ± 20.38	158 ± 2	7.93 ± 1.53	31.19 ± 12.41	315 ± 106
eMORL [14]	50.19 ± 22.56	93.25 ± 3.24	33 ± 8	12.58 ± 2.39	45.00 ± 7.77	318 ± 43
MONet [4]	30.66 ± 14.87	54.47 ± 11.41	58 ± 12	19.78 ± 1.02	36.66 ± 0.87	146 ± 7
SA [31]	36.61 ± 24.83	95.89 ± 2.37	23 ± 3	22.58 ± 2.07	62.40 ± 2.23	254 ± 8
IODINE [19]	45.14 ± 17.85	93.81 ± 0.76	44 ± 9	29.17 ± 0.75	59.52 ± 2.20	340 ± 3
AST-Seg-B3-BT	71.92 ± 32.94	76.05 ± 36.13	51 ± 63	57.30 ± 15.72	71.79 ± 22.88	152 ± 39
AST-Seg-B3-CT	**90.27** ± **0.20**	**98.26** ± **0.07**	**16** ± **1**	**79.58** ± **0.54**	**94.77** ± **0.51**	**139** ± **7**

Table 2. Benchmark results on ObjectsRoom and ShapeStacks. Source: [15].

Model	ObjectsRoom				ShapeStacks			
	↑ARI-FG (%)	↑MSC-FG (%)	↑mIoU (%)	↓MSE	↑ARI-FG (%)	↑MSC-FG (%)	↑mIoU (%)	↓MSE
MONet-g [4, 15]	54 ± 0	33 ± 1	n/a	n/a	70 ± 4	57 ± 12	n/a	n/a
Gen-v2 [15]	84 ± 1	58 ± 3	n/a	n/a	**81** ± **0**	68 ± 1	n/a	n/a
SA [31]	79 ± 2	64 ± 13	n/a	n/a	76 ± 1	70 ± 5	n/a	n/a
AST-Seg-B3-BT	74.96 ± 10.02	69.86 ± 10.13	74.50 ± 8.61	11.7 ± 2.1	73.77 ± 7.56	74.12 ± 8.63	70.18 ± 12.68	11.8 ± 7.0
AST-Seg-B3-CT	**87.23** ± **0.88**	**82.22** ± **0.96**	**85.02** ± **0.79**	**6.7** ± **0.9**	79.34 ± 0.73	**77.65** ± **1.3**	**78.84** ± **0.21**	**4.5** ± **0.2**

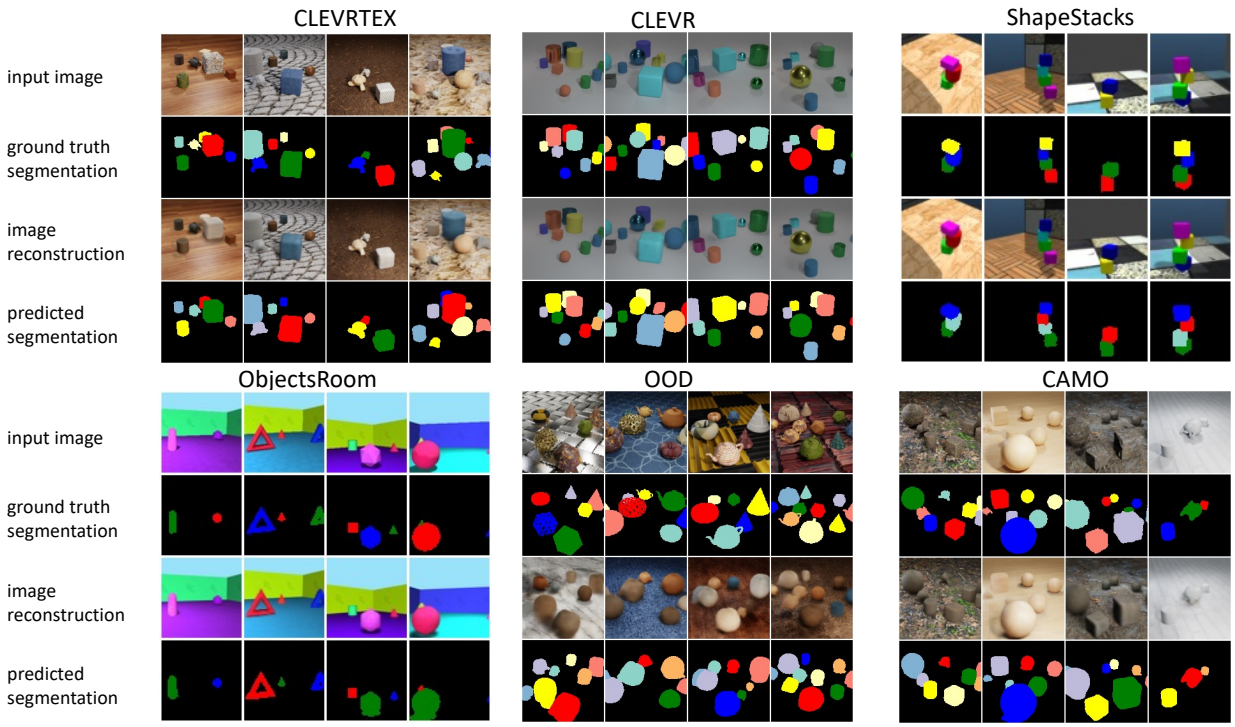

Figure 2. Examples of segmentation predictions on CLEVRTEX, CLEVR, ShapeStacks, ObjectsRoom, OOD and CAMO test datasets (Results on OOD and CAMO datasets are obtained using a model trained on CLEVRTEX only)

5.3. Computation time

All experiments have been performed using a Nvidia RTX 3090 GPU and a AMD 7402 EPYC CPU. Some training durations are provided in Table 5.

Table 3. Benchmark generalization results on CAMO, and OOD for a model trained on CLEVRTEX. Results are shown ($\pm\sigma$) calculated over 3 runs. Source: [27]

Model	OOD			CAMO		
	↑mIoU (%)	↑ARI-FG (%)	↓MSE	↑mIoU (%)	↑ARI-FG (%)	↓MSE
SPAIR [10]	0.00 ± 0.00	0.00 ± 0.00	1166 ± 5	0.00 ± 0.00	0.00 ± 0.00	668 ± 3
SPACE [30]	6.87 ± 3.32	12.71 ± 3.44	387 ± 66	8.67 ± 3.50	10.55 ± 2.09	251 ± 61
GNM [23]	40.84 ± 0.30	48.43 ± 0.86	626 ± 5	17.56 ± 0.74	15.73 ± 0.89	353 ± 1
MN [36]	12.13 ± 0.19	37.29 ± 1.04	409 ± 3	8.79 ± 0.15	31.52 ± 0.87	265 ± 1
DTI [32]	32.55 ± 1.08	73.67 ± 0.98	590 ± 4	27.54 ± 1.55	72.90 ± 1.89	377 ± 17
Gen-V2 [15]	8.74 ± 1.64	29.04 ± 11.23	539 ± 147	7.49 ± 1.67	29.60 ± 12.84	278 ± 75
eMORL [14]	13.17 ± 2.58	43.13 ± 9.28	471 ± 51	11.56 ± 2.09	42.34 ± 7.19	269 ± 31
MONet [4]	19.30 ± 0.37	32.97 ± 1.00	**231** ± **7**	10.52 ± 0.38	12.44 ± 0.73	**112** ± **7**
SA [31]	20.98 ± 1.59	58.45 ± 1.87	487 ± 16	19.83 ± 1.41	57.54 ± 1.01	215 ± 7
IODINE [19]	26.28 ± 0.85	53.20 ± 2.55	504 ± 3	17.52 ± 0.75	36.31 ± 2.57	315 ± 3
AST-Seg-B3-CT	**67.50** ± **0.75**	**83.14** ± **0.75**	832 ± 24	**73.07** ± **0.65**	**87.27** ± **3.78**	145 ± 6

Table 4. Results of ablation study and additional experiments (results over 1 run, except for starred values, which are averages over 3 runs)

Dataset	CLEVRTEX		CLEVR		ShapeStacks		ObjectsRoom	
	mIoU	ARI-FG	mIoU	ARI-FG	mIoU	ARI-FG	mIoU	ARI-FG
full model AST-Seg-B3-CT (reference)	79.58*	94.77*	90.27*	98.26*	78.84*	79.34*	85.02*	87.23*
model without transformer encoder	75.69	94.41	77.16	93.09	82.99*	82.29*	85.51*	88.49*
K = 1 + maximum number of objects	79.11*	94.78*	91.03*	98.17*	78.87	80.05	82.90	86.45
K = 2 × maximum number of objects	62.10	89.96	90.56	98.29	54.88	65.16	66.78	78.58
using a Unet instead of Segformer feature generator	66.82	88.25	90.70	98.17	75.51	77.78	85.59	87.93
random initialization of Segformer backbone	61.74	80.22	88.94	97.77	62.73	68.40	77.71	79.23
training without pixel entropy loss	70.18	91.81	85.54	96.09	52.17	60.08	84.21	86.19
training using frozen pretrained background model	75.30	95.31	81.46	98.29	55.06	66.24	85.82	87.78
isotropic scaling	78.68	94.78					84.91	87.20
anisotropic scaling			87.21	98.53	45.47	36.43		

Table 5. Training computation time with one Nvidia RTX 3090 GPU (curriculum training)

Dataset	image size	background model pretraining (phase 1)		full model training (phase 2 & 3)	
		number of iterations	training time	number of iterations	training time
CLEVRTEX	128 × 128	500000	57 h 47 mn	125000	16 h 00 mn
CLEVR	128 × 128	2500	20 mn	125000	12 h 03 mn
ObjectsRoom	64 × 64	500000	14 h 57 mn	125000	6 h 31 mn
ShapeStacks	64 × 64	500000	14 h 20 mn	125000	6 h 22 mn

6. Conclusion

We have described in this paper a new architecture for unsupervised object-centric representation learning and object detection and segmentation, which relies on attention and soft-argmax, and shown that this new architecture substantially improves upon the state of the art on existing benchmarks showing synthetic scenes with complex shapes and textures. We hope this work may help to extend the scope of structured object-centric representation learning from research to practical applications.

Acknowledgment We thank Sascha Hornauer for useful comments on the first draft of this paper.

References

[1] S. M. Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Koray Kavukcuoglu, and Geoffrey E. Hinton. Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems*, pages 3233–3241, mar 2016.

[2] Jonathan T. Barron. Continuously Differentiable Exponential Linear Units. *arXiv*, (3):1–2, 2017.

[3] Adam Bielski and Paolo Favaro. Emergence of object segmentation in perturbed generative models. *Advances in Neural Information Processing Systems*, 32(NeurIPS):1–11, 2019.

[4] Christopher P. Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. MONet: Unsupervised Scene Decomposition and Representation. *arxiv preprint*, jan 2019.

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-End Object Detection with Transformers. *Lecture Notes in Computer Science (including subseries Lecture Notes in*

Artificial Intelligence and Lecture Notes in Bioinformatics), 12346 LNCS:213–229, 2020.

[6] Prashanth Chandran, Derek Bradley, Markus Gross, and Thabo Beeler. Attention-driven cropping for very high resolution facial landmark detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 5860–5869, 2020.

[7] Mickaël Chen, Thierry Artières, and Ludovic Denoyer. Unsupervised object segmentation by redrawing. *Advances in Neural Information Processing Systems*, 32, 2019.

[8] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention Mask Transformer for Universal Image Segmentation. 2021.

[9] Bowen Cheng, Alexander G. Schwing, and Alexander Kirillov. Per-Pixel Classification is Not All You Need for Semantic Segmentation. (NeurIPS):1–17, 2021.

[10] Eric Crawford and Joelle Pineau. Spatially Invariant Unsupervised Object Detection with Convolutional Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3412–3420, 2019.

[11] David Ding, Felix Hill, Adam Santoro, Malcolm Reynolds, and Matt Botvinick. Attention over learned object embeddings enables complex visual reasoning. In *Advances in Neural Information Processing Systems*, volume 11, pages 9112–9124, 2021.

[12] Bin Dong, Fangao Zeng, Tiancai Wang, Xiangyu Zhang, and Yichen Wei. SOLQ: Segmenting Objects by Learning Queries. (2017):1–12, 2021.

[13] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. CenterNet: Keypoint triplets for object detection. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-Octob:6568–6577, 2019.

[14] Patrick Emami, Pan He, Sanjay Ranka, and Anand Rangarajan. Efficient Iterative Amortized Inference for Learning Symmetric and Disentangled Multi-Object Representations. 2021.

[15] Martin Engelcke, Oiwi Parker Jones, and Ingmar Posner. GENESIS-V2: Inferring Unordered Object Representations without Iterative Refinement. 2021.

[16] Martin Engelcke, Adam R. Kosiorek, Oiwi Parker Jones, and Ingmar Posner. GENESIS: Generative Scene Inference and Sampling with Object-Centric Latent Representations. Technical report, 2019.

[17] Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Deep spatial autoencoders for visuomotor learning. *Proceedings - IEEE International Conference on Robotics and Automation*, 2016-June:512–519, 2016.

[18] Ross Goroshin, Michael Mathieu, and Yann Lecun. Learning to linearize under uncertainty. *Advances in Neural Information Processing Systems*, 2015-Janua:1234–1242, 2015.

[19] Klaus Greff, Raphael Lopez Kaufman, Rishabh Kabra, Nick Watters, Chris Burgess, Daniel Zoran, Loie Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In *36th International Conference on Machine Learning, ICML 2019*, volume 2019-June, pages 4317–4343, mar 2019.

[20] Klaus Greff, Antti Rasmus, Mathias Berglund, Tele Hotloo Hao, Jürgen Schmidhuber, and Harri Valpola. Tagger: Deep unsupervised perceptual grouping. *Advances in Neural Information Processing Systems*, (Nips):4491–4499, 2016.

[21] Oliver Groth, Fabian B. Fuchs, Ingmar Posner, and Andrea Vedaldi. ShapeStacks: Learning Vision-Based Physical Intuition for Generalised Object Stacking. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11205 LNCS:724–739, 2018.

[22] Jyh Jing Hwang, Stella Yu, Jianbo Shi, Maxwell Collins, Tien Ju Yang, Xiao Zhang, and Liang Chieh Chen. SegSort: Segmentation by discriminative sorting of segments. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-Octob:7333–7343, 2019.

[23] Jindong Jiang and Sungjin Ahn. Generative neurosymbolic machines. *Advances in Neural Information Processing Systems*, 2020-Decem(NeurIPS), 2020.

[24] Jindong Jiang, Sepehr Janghorbani, Gerard de Melo, and Sungjin Ahn. SCALOR: Generative World Models with Scalable Object Representations. 2019.

[25] Justin Johnson, Li Fei-Fei, Bharath Hariharan, C. Lawrence Zitnick, Laurens Van Der Maaten, and Ross Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:1988–1997, 2017.

[26] Rishabh Kabra, Chris Burgess, Loic Matthey, Raphael Lopez Kaufman, Klaus Greff, Malcolm Reynolds, and Alexander Lerchner. Multi-Object Datasets. https://github.com/deepmind/multi-object-datasets/, 2019.

[27] Laurynas Karazija, Iro Laina, and Christian Rupprecht. ClevrTex: A Texture-Rich Benchmark for Unsupervised Multi-Object Segmentation. (NeurIPS), 2021.

[28] Adam R. Kosiorek, Hyunjik Kim, Ingmar Posner, and Yee Whye Teh. Sequential attend, infer, repeat: Generative modelling of moving objects. In *Advances in Neural Information Processing Systems*, volume 2018-Decem, pages 8606–8616, 2018.

[29] Hei Law and Jia Deng. CornerNet: Detecting Objects as Paired Keypoints. *International Journal of Computer Vision*, 128(3):642–656, 2020.

[30] 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. 2020.

[31] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 2020-Decem(NeurIPS):1–14, 2020.

[32] Tom Monnier, Elliot Vincent, Jean Ponce, and Mathieu Aubry. Unsupervised Layered Image Decomposition into Object Prototypes. 2021.

[33] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9912 LNCS:483–499, 2016.

[34] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Nassir Navab, Joachim Hornegger, William M Wells, and Alejandro F Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[35] Bruno Sauvalle and Arnaud de La Fortelle. Autoencoder-based background reconstruction and foreground segmentation with background noise estimation. *arxiv preprint arXiv:2112.08001*, 2021.

[36] Dmitriy Smirnov, Michael Gharbi, Matthew Fisher, Vitor Guizilini, Alexei A. Efros, and Justin Solomon. MarioNette: Self-Supervised Sprite Learning. (NeurIPS), 2021.

[37] Karl Stelzner, Robert Peharz, and Kristian Kersting. Faster Attend-Infer-Repeat with Tractable Probabilistic Models. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 5966–5975, may 2019.

[38] Xiao Sun, Bin Xiao, Fangyin Wei, Shuang Liang, and Yichen Wei. Integral human pose regression. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11210 LNCS:536–553, 2018.

[39] Qu Tang, Zhu Xiangyu, Lei Zhen, and Zhaoxiang Zhang. Object Dynamics Distillation for Scene Decomposition and Representation. In *International Conference on Learning Representations*, 2022.

[40] Aleksei Tiulpin, Iaroslav Melekhov, and Simo Saarakkala. KNEEL: Knee anatomical landmark localization using hourglass networks. In *Proceedings - 2019 International Conference on Computer Vision Workshop, ICCVW 2019*, pages 352–361, 2019.

[41] Sjoerd Van Steenkiste, Klaus Greff, Michael Chang, and Jürgen Schmidhuber. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, pages 1–15, 2018.

[42] Rishi Veerapaneni, John D. Co-Reyes, Michael Chang, Michael Janner, Chelsea Finn, Jiajun Wu, Joshua B. Tenenbaum, and Sergey Levine. Entity Abstraction in Visual Model-Based Reinforcement Learning. (CoRL):1–21, 2019.

[43] John Wright, Yigang Peng, Yi Ma, Arvind Ganesh, and Shankar Rao. Robust principal component analysis: Exact recovery of corrupted low-rank matrices by convex optimization. *Advances in Neural Information Processing Systems 22 - Proceedings of the 2009 Conference*, pages 2080–2088, 2009.

[44] Yuxin Wu and Kaiming He. Group Normalization. *International Journal of Computer Vision*, 128(3):742–755, 2020.

[45] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers. In *Advances in Neural Information Processing Systems*, volume 15, pages 12077–12090, 2021.

[46] Andrii Zadaianchuk, Maximilian Seitzer, and Georg Martius. Self-supervised Visual Reinforcement Learning with Object-centric Representations. In *International Conference on Learning Representations*, 2021.

[47] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable Transformers for End-to-End Object Detection. 2020.

7. Supplementary Material

7.1. Hyperparameter values

The hyperparameter values used for the proposed model are listed in Table 6.

7.2. Pseudo-code for objects encoder and decoder

The full encoding and rendering process is described in Algorithms 1 and 2.

7.3. Additional implementation details

The glimpse convolutional generator is described in Table 7.

Synthetic datasets and preprocessing codes were downloaded from the following public repositories:

- `https://www.robots.ox.ac.uk/~vgg/data/clevrtex/`
- `https://ogroth.github.io/shapestacks/`
- `https://github.com/deepmind/multi_object_datasets`
- `https://github.com/applied-ai-lab/genesis`.

The Segformer pretrained weights were downloaded from the following link:
`https://huggingface.co/nvidia/mit-b3`

The architecture of the U-net implemented for the ablation study is described in Table 8. It contains a sequence of downsample blocks which output feature maps of decreasing sizes, a center block which takes as input the feature map produced by the last downsample block, and upsample blocks, which take as input both the output of the previous upsample or center block and the feature map of the same size produced by corresponding downsample block.

- A downsample block is composed of a convolutional layer with stride 2 and kernel size 4, with batch normalization and CELU, followed by a residual convolutional layer with stride 1 and kernel size 3 with batch normalization and CELU.

Table 6. Hyperparameter values

hyperparameter description	notation	value
Background model pretraining:		
batch size		128
learning rate		2.10^{-3}
number of background model training iterations:		
- datasets with fixed backgrounds (CLEVR)		2500
- datasets with complex backgrounds (CLEVRTEX, ShapeStacks,ObjectsRoom)		500000
Foreground model training:		
batch size		64
learning rate		4.10^{-5}
Adam β_1		0.90
Adam β_2		0.98
Adam ϵ		10^{-9}
number of foreground model training iterations		125000
number of steps of phase 2 (CT scenario)		30000
number of steps of learning rate warmup phase		5000
number of steps of pixel entropy loss weight warmup phase	N_{pixel}	10000
initial value of background activation before training	α_0	e^{11}
dimension of z_{what}	$d_{z_{what}}$	32
pixel entropy loss weight	λ_{pixel}	1.10^{-2}
minimum value of inverse scaling factor	s_{min}	1.3
maximum value of inverse scaling factor	s_{max}	24
dimension of inputs and outputs of transformer encoder	d_T	256
number of heads of transformer encoder layer		8
dimension of feedforward transformer layer		512
number of layers of transformer encoder		6

- The center block is composed of a convolutional layer with stride 1 and kernel size 3 with batch normalization and CELU.
- An upsample block is composed of a residual convolutional layer with stride 1 and kernel size 3 with batch normalization and CELU, followed by a transpose convolutional layer with stride 2 and kernel size 4, with batch normalization and CELU.

7.4. Additional image samples

Additional image samples are provided in Figures 1-6.

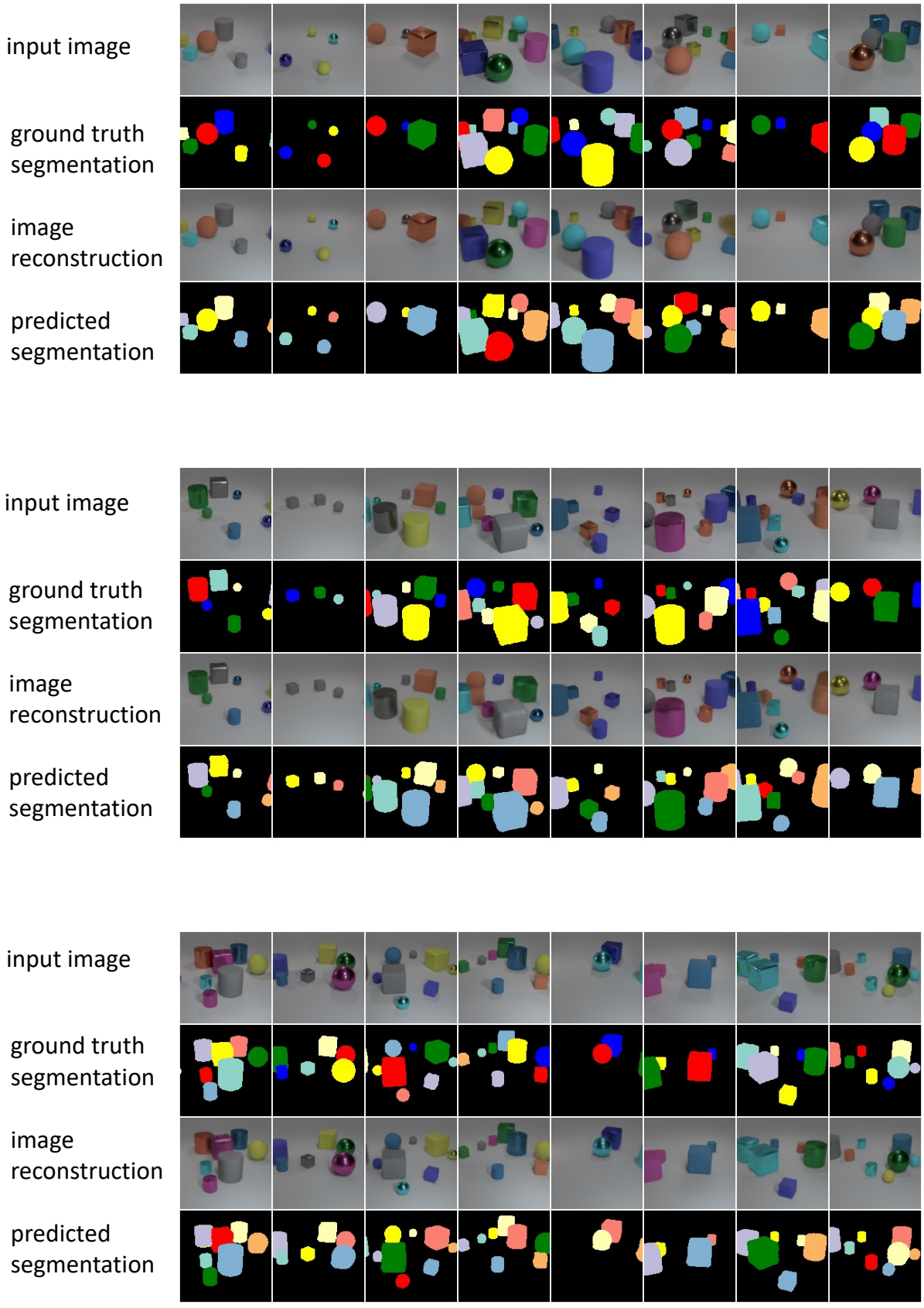

Figure 3. Examples of segmentation predictions on CLEVR test dataset

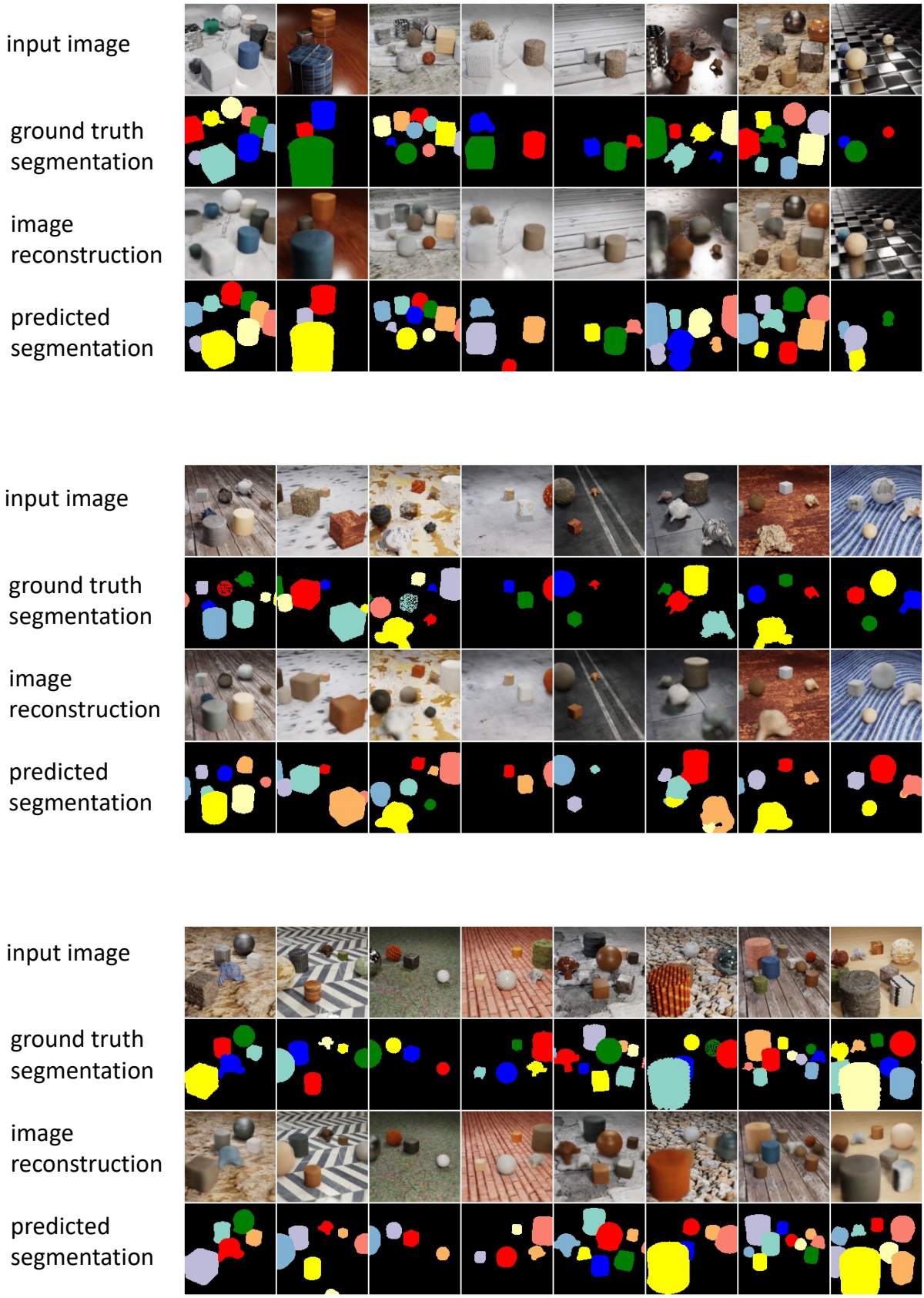

Figure 4. Examples of segmentation predictions on CLEVRTEX test dataset

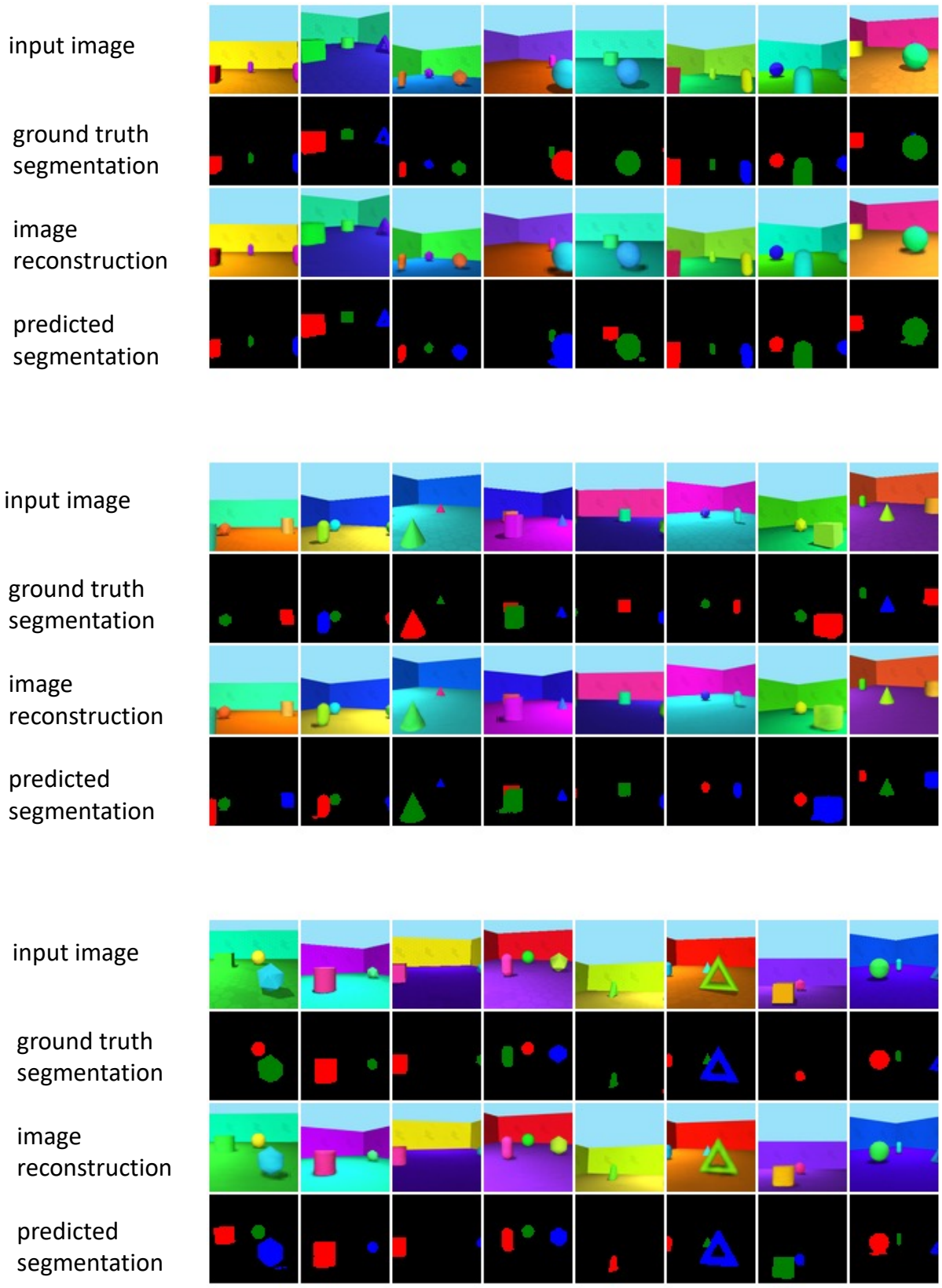

Figure 5. Examples of segmentation predictions on ObjectsRoom test dataset

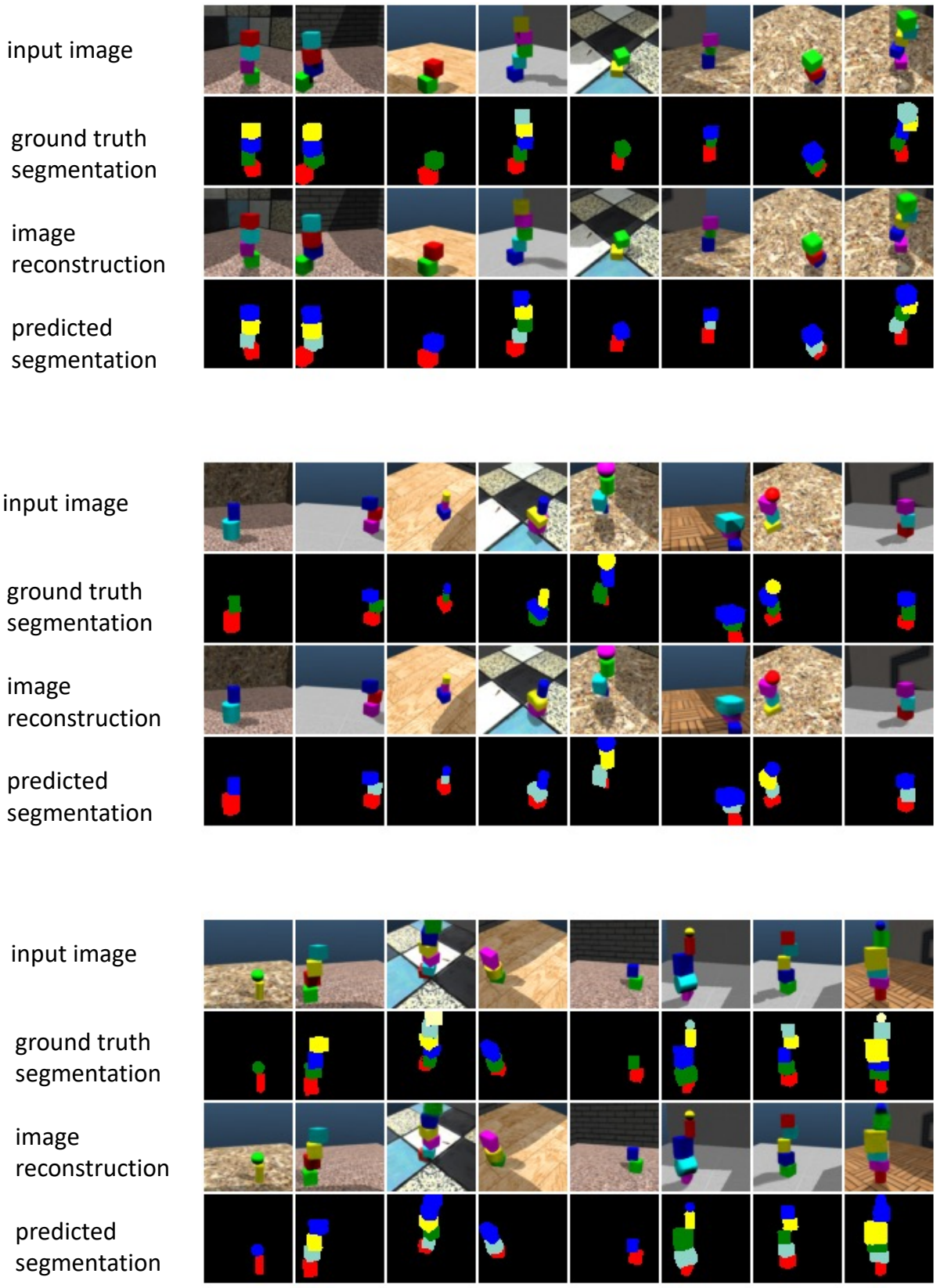

Figure 6. Examples of segmentation predictions on ShapeStacks test dataset (using a model without transformer)

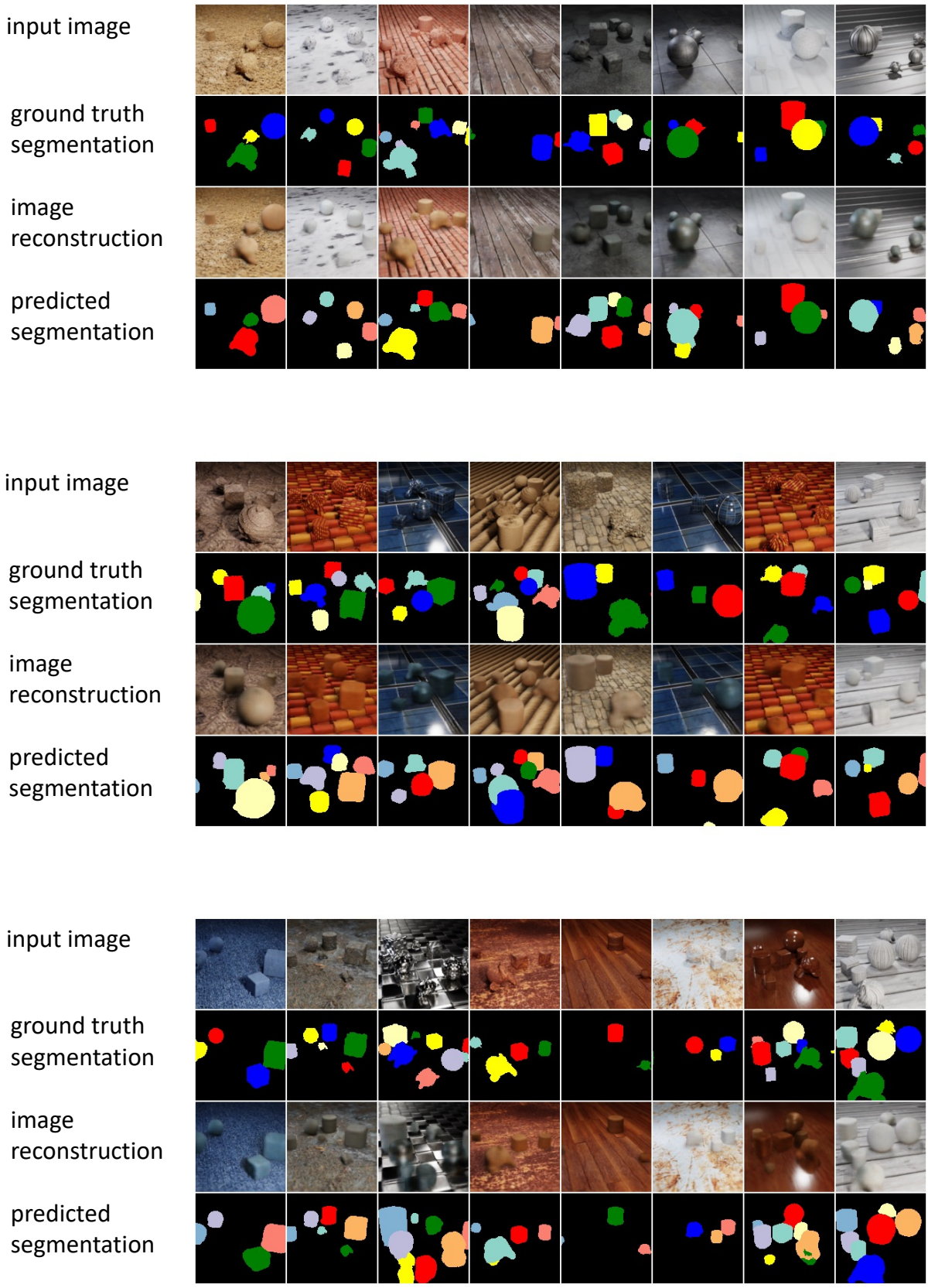

Figure 7. Examples of segmentation predictions on CAMO test dataset using a model trained on CLEVRTEX only

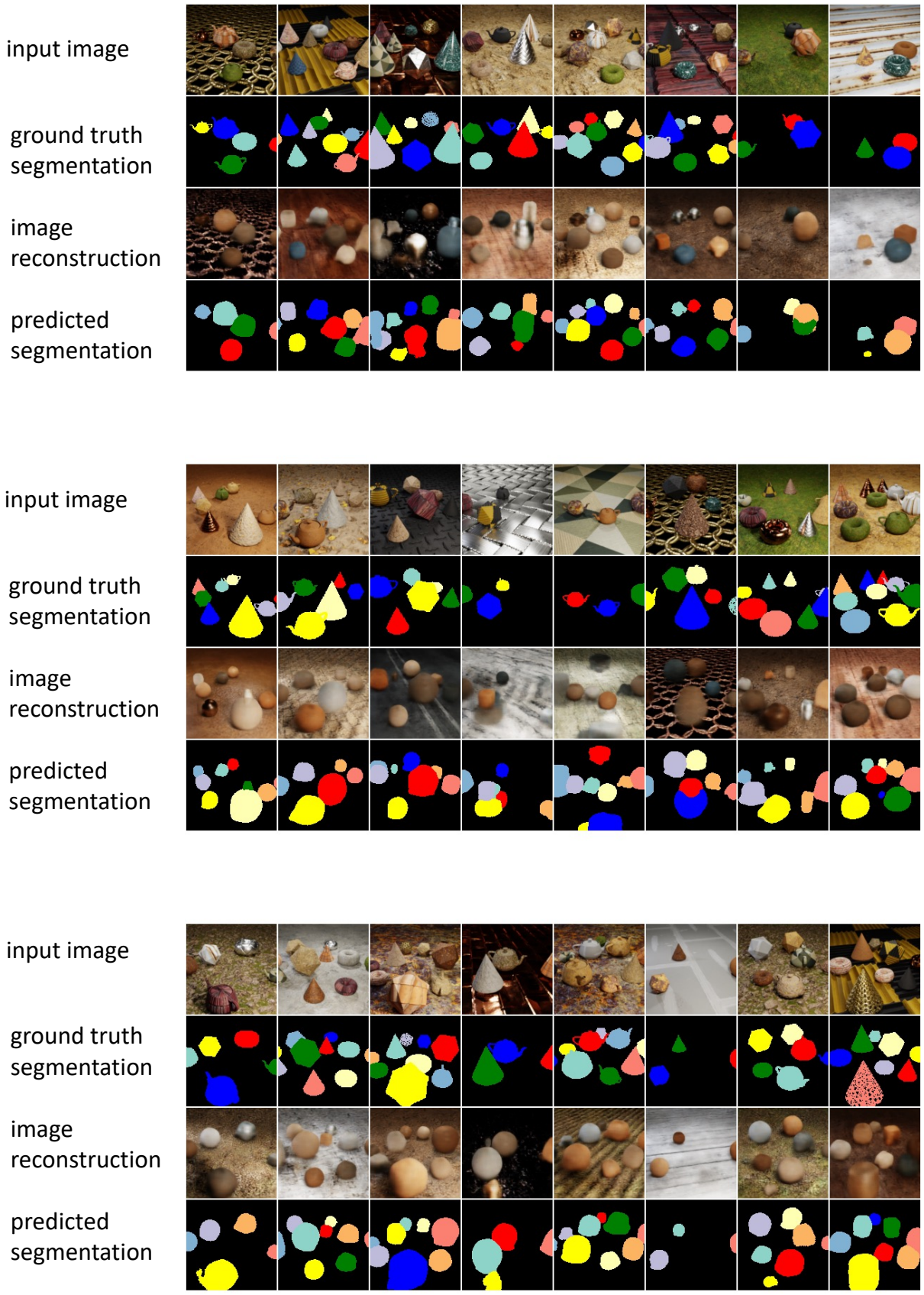

Figure 8. Examples of segmentation predictions on OOD test dataset using a model trained on CLEVRTEX only

Algorithm 1: Encoding

Input: input image $\mathbf{X}$

Output: object latents $\{z_k^{what}, x_k, y_k, s_k, \alpha_k\}_{1 \geq k \geq K}$

// feature and attention maps generation

$(\Phi, A_1, .., A_K)$ = SegformerForSemanticSegmentation($\mathbf{X}$)

for $k \leftarrow 1$ **to** K **do**

  $\mathcal{A}_k(i,j)$ = Softmax$(A_k)(i,j) = \frac{e^{A_k(i,j)}}{\sum_{i,j} e^{A_k(i,j)}}$

end

// computation of positions and feature vectors before transformer refinement

for $i \leftarrow 1$ **to** w^*, $j \leftarrow 1$ **to** h^* **do**

  $x(i) = 2\frac{i-1}{w^*-1} - 1$; $y(j) = 2\frac{j-1}{h^*-1} - 1$

end

for $k \leftarrow 1$ **to** K **do**

  $x_k^0 = \sum_{i,j} x(i)\mathcal{A}_k(i,j)$; $y_k^0 = \sum_{i,j} y(j)\mathcal{A}_k(i,j)$

  $\phi_k^0 = \sum_{i,j} \Phi(i,j)\mathcal{A}_k(i,j)$

end

// transformer refinement of positions and feature vectors

$(x_k, y_k, \phi_k)_{1 \geq k \geq K}$ = LinearProjection(TransformerEncoder(LinearEmbedding($(x_k^0, y_k^0, \phi_k^0)_{1 \geq k \geq K}$)))

// latent computations

for $k \leftarrow 1$ **to** K **do**

  x_k = clamp$(x_k, min = -1, max = 1)$; y_k = clamp$(y_k, min = -1, max = 1)$

  $(s_k, \alpha_k, z_k^{what}) = \phi_k$

  $s_k = s_{min} + (s_{max} - s_{min})\sigma(s_k)$

  $\alpha_k = e^{\alpha_k}$

end

Output: $\{z_k^{what}, x_k, y_k, s_k, \alpha_k\}_{1 \geq k \geq K}$

Algorithm 2: Rendering

Input: object latents $\{z_k^{what}, x_k, y_k, s_k, \alpha_k\}_{1 \geq k \geq K}$, background image L_0, background mask $M_0 = 1$, learned background activation α_0

Output: Image reconstruction $\hat{\mathbf{X}}$

// Obtain the object appearance $\mathbf{o}_k$ and segmentation mask $\mathbf{m}_k$

for $k \leftarrow 1$ **to** K **do**

  $\mathbf{o}_k, \mathbf{m}_k$ = GlimpseGenerator($\mathbf{z}_k^{\text{what}}$)

end

// translation and scaling using a spatial transformer network (STN)

for $k \leftarrow 1$ **to** K **do**

  L_k = STN$(\mathbf{o}_k, x_k, y_k, s_k)$

  M_k = STN$(\mathbf{m}_k, x_k, y_k, s_k)$

end

// occlusion computations

for $k \leftarrow 0$ **to** K **do**

  $w_k = \frac{\alpha_k M_k}{\sum_{i=0}^{K} \alpha_i M_i}$

end

// combination of image layers

$\hat{\mathbf{X}} = \sum_{k=0}^{K} w_k L_k$;

Output: $\hat{\mathbf{X}}$

Table 7. glimpse generator architecture

64x64 images

Layer	Size	Ch	Stride	Padding	Norm./Act.
Input	1	$d_{z_{what}}$			
Transp Conv 2×2	2	64	2	0	GroupNorm(4,64) /CELU
Transp Conv 4×4	4	32	2	1	GroupNorm(2,32)/CELU
Transp Conv4×4	8	16	2	1	GroupNorm(1,16)/CELU
Transp Conv 4×4	16	8	2	1	GroupNorm(1,8)/CELU
Transp Conv 4×4	32	4	2	1	
Sigmoid	32	4			

128x128 images

Layer	Size	Ch	Stride	Padding	Norm./Act.
Input	1	$d_{z_{what}}$			
Transp Conv 2×2	2	128	2	0	GroupNorm(8,128) /CELU
Transp Conv 4×4	4	64	2	1	GroupNorm(4,64)/CELU
Transp Conv4×4	8	32	2	1	GroupNorm(2,32)/CELU
Transp Conv 4×4	16	16	2	1	GroupNorm(1,16)/CELU
Transp Conv 4×4	32	8	2	1	GroupNorm(1,8)/CELU
Transp Conv 4×4	64	4	2	1	
Sigmoid	64	4			

Table 8. U-net architecture (ablation study)

Layer	Ch	Stride	Padding	Norm./Act.
Input	3			
Conv 3×3	80	1	1	BatchNorm /CELU
Downsample block	128			
Downsample block	192			
Downsample block	256			
Downsample block	256			
Downsample block	256			
Center block	256			
Upsample block	256			
Upsample block	256			
Upsample block	192			
Upsample block	128			
Upsample block	80			
Conv 3×3 with skip connection	d_Φ	1	1	BatchNorm /CELU
Residual Conv 3×3	d_Φ	1	1	
Conv 1×1	d_Φ	1	1	