Universiteit Leiden

Master Computer Science

Disentangling Shape and Texture Dimensions in Object-Centric Representations

Name:	Riccardo Majellaro
Student ID:	s3094952
Date:	21/07/2023
Specialisation:	Artificial Intelligence
1st supervisor:	Thomas Moerland
2nd supervisor:	Aske Plaat

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

In the field of computer vision, recent deep learning techniques aimed at extracting, without supervision, structured representations centered around objects from raw perceptual data. Disentangling factors of variation in these object-centric representations can enhance the robustness of latent features, enabling more efficient and effective learning of multiple downstream tasks. Despite previous literature in representation learning focused on this aspect, to the best of our knowledge, no approaches dealt with the explicit separation of shape and texture factors within the context of object-centric learning. In this thesis, we propose a novel architectural design that biases object-centric models toward encoding texture and shape information into two non-overlapping subsets of the latent space dimensions that are known a priori. Additionally, this design allows the introduction of a self-supervised loss that, along with the traditional reconstruction loss, helps to achieve our goal by encouraging accurate transfer and adaptation of textures between entities presenting distinct shapes. After extending the architecture of Slot Attention to follow our design, we managed to obtain the desired disentanglement, while also improving the performance of the original model. Consequently, our models display the capability of composing new entities by transferring, interpolating, noising, or sampling textures and shapes of the objects in a scene. These results demonstrate a step forward in the direction of more robust and interpretable object-centric representations in the field of computer vision.

Contents

1 Introduction

Learning to represent data in such ways that are convenient to be leveraged by a multitude of tasks is one of the most important problems in deep learning. It becomes particularly complex when approached in unsupervised or self-supervised settings, although opens the possibility of exploiting large unlabeled datasets. In the context of computer vision, recent literature [1–10] focused on representing, without supervision, visual scenes composed of multiple objects as sets of object-centered latent vectors. This strategy allows for more structured representations compared to single flat vectors encoding complete images. In the latter, in fact, objects can be separated into distinct latent components, with the downside of not sharing features, even if related to common properties such as the position. Instead, with object-centric vectors, each visual entity is encoded in the same space, improving generalization and interpretability since they are clearly divided and described by the same features. Our work is based on this topic, which is often referred to as object-centric representation learning.

Disentanglement is considered one of the most important properties for learning robust representations of complex and structured visual scenes [11]. The term refers to the encoding of different factors of variation in the data into separate dimensions of the latent space. This property, other than enabling an easier interpretation of the learned features, leads to representations that can be meaningful even outside the data distribution. As a consequence, the disentanglement should help the learning of relevant but unknown downstream tasks, for instance in the context of reinforcement learning. Prior work in object-centric representation learning [1, 2] exploited VAEs [12] and spatial broadcast decoders [13] to help the disentanglement of the features. Biza et al. [10] proposed instead an extension of the popular Slot Attention architecture aimed at learning object representations invariant to position, orientation, and scale, allowing for the disentanglement of those three factors. However, to the best of our knowledge, no research has been carried out on the explicit disentanglement of texture and shape dimensions in object-centric learning, especially with non-probabilistic models. By the term explicit, we refer to the process of choosing (and thus knowing a priori) which latent components will be responsible for encoding certain factors. One possible benefit of explicit disentanglement is the more reliable modeling of structured latent spaces. Moreover, it offers the possibility of designing objectives that exploit this prior knowledge from the beginning, removing the need to analyze and interpret the latent components during or after the training process.

Our work focuses on biasing object-centric models toward explicit disentanglement, thus in a known a priori structure, of the features responsible for encoding shape and texture information. We achieve this with a novel architectural design that takes advantage of the separate prediction of mask and texture typically present in object-centric models. The idea is to employ two encoder-decoder pairs, one encoding shape information and decoding object masks and the other representing and predicting object textures. Additionally, either a fixed filter (e.g. Sobel filter) or a learned bottleneck filter is used to remove texture information from the input image. The resulting filtered image represents the input of the shape encoder, which helps prevent texture information from flowing into shape-related latent components. This design can be adapted to suit different existent architectures, such as Slot Attention [5], IODINE [2], and MONet [1]. Furthermore, as anticipated, the a priori knowledge of which latent space components will encode texture and shape information allows for the introduction of novel objectives to support the learning of object-centric representations. We provide an example by proposing Latent Compositionality Loss (LCL), a self-supervised objective that, paired with the usual reconstruction loss, explicitly pushes the model to learn accurate transfer and adaptation of textures between entities presenting different shapes, helping to obtain the desired disentanglement property.

To evaluate our work, we extend the architecture of Slot Attention following the proposed design and denominate it Disentangled Invariant Slot Attention (DISA). The term invariant refers to the use of slot-centric reference frames introduced by Biza et al. [10] with Invariant Slot Attention (ISA). Our experiments demonstrate the introduction of the desired disentanglement property in the learned representations, while also obtaining an improvement over the performance of the original model on three well-known synthetic datasets.

In the following two sections, we cover the necessary background information to understand this thesis (Section 2), other than presenting work that is related to ours (Section 3). Our methodology is detailed in Section 4, while Section 5 and Section 6 present our experiments and results. Finally, we conclude the thesis in Section 7 with a discussion of our proposals and findings, as well as potential directions for future research.

2 Background

This section introduces key concepts essential for the complete understanding of our work. We begin with the attention mechanism in Section 2.1, a component at the core of Slot Attention (Section 2.2) and, therefore, of DISA. Then, we present the slot-centric reference frames (Section 2.3), the key to inducing the disentanglement of position and scale in the object-centric representations learned by Slot Attention. Lastly, we introduce the Sobel filter (Section 2.4), employed in our methodology to detect the edges of objects in an image.

2.1 Attention Mechanism

Consider three sets of vectors called *queries*, *keys*, and *values*, represented respectively by the matrices $\mathrm{Q} \in \mathbb{R}^{N_q \times d_k}$, $\mathrm{K} \in \mathbb{R}^{N_v \times d_k}$ and $\mathrm{V} \in \mathbb{R}^{N_v \times d_v}$. Keys and values are paired, in the sense that the i-th key K_i is associated with the i-th value V_i. For each query Q_i, the attention mechanism outputs a vector A_i computed as a convex combination of the values, thus

$$\mathrm{A}_i = \sum_{j=1}^{N_v} w_{ij} V_j \quad \text{with} \quad \sum_{j=1}^{N_v} w_{ij} = 1 \ . \tag{1}$$

The weights, which can be seen as a probability distribution over both keys and values, are obtained through a normalized similarity function between the given query and all the keys. This function is commonly the dot product, defined as

$$\mathrm{Q}_i \cdot \mathrm{K}_j = \sum_{l=1}^{d_k} Q_{il} K_{jl} = ||\mathrm{Q}_i|| \, ||\mathrm{K}_j|| \cos\alpha \ , \tag{2}$$

where α is the angle between Q_i and K_i, taking therefore into account the magnitudes of the two vectors and the angle between them. To normalize the similarities, the softmax function is frequently employed, allowing us to finally define the weight w_{ij} as

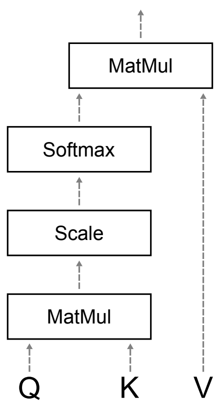
$$w_{ij} = \mathrm{softmax}(\mathrm{Q}_i \mathrm{K}^\top)_j = \frac{e^{\mathrm{Q}_i \cdot \mathrm{K}_j}}{\sum_{l=1}^{N_v} e^{\mathrm{Q}_i \cdot \mathrm{K}_l}} \ . \tag{3}$$

Usually, each dot product in Equation 3 is scaled by $\frac{1}{\sqrt{d_k}}$ in order to prevent it from getting too large in magnitude for high values of d_k, which could result in extremely small gradients during training [14]. In a more compact formulation, the complete attention mechanism can be written as

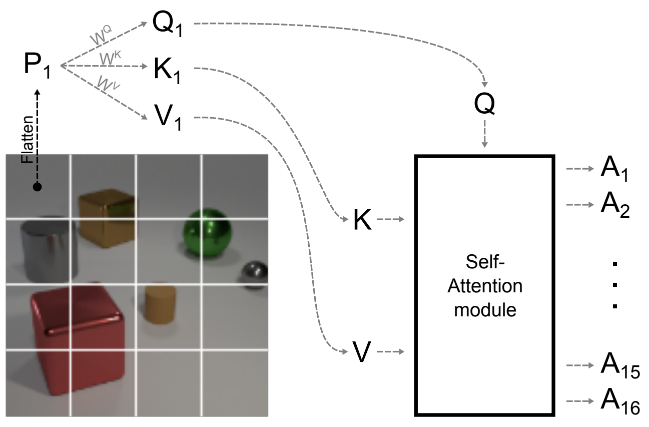
$$\mathrm{A}(\mathrm{Q}, \mathrm{K}, \mathrm{V}) = \mathrm{softmax}\left(\frac{\mathrm{Q}\mathrm{K}^\top}{\sqrt{d_k}}\right) \mathrm{V} \in \mathbb{R}^{N_q \times d_v} \ , \tag{4}$$

where the softmax is individually applied to each row of the matrix $\frac{\mathrm{QK}^\top}{\sqrt{d_k}} \in \mathbb{R}^{N_q \times N_v}$. Figure 1a shows a scheme of the aforementioned computations involved in the attention mechanism.

It is now left to understand how Q, K, and V are actually obtained. There are two main strategies, one called *self-attention* and the other *cross-attention*. In self-attention, Q, K, and V originate from the same source, such as a sequence of text embeddings, in the context of Natural Language Processing (NLP), or a sequence of patch embeddings, in the context of Computer Vision (CV). Intuitively, each vector in the input sequence is able to attend to every element in that same sequence and compute as output a vector that relates all the sequence information. Figure 1b illustrates a simple example of self-attention in CV inspired by the Vision Transformer (ViT) [15]. An image is divided into N_p patches, all of which are first flattened and then projected in a d_k-dimensional space through three linear transformations $\mathrm{Q} = \mathrm{PW}^{\mathrm{Q}}$, $\mathrm{K} = \mathrm{PW}^{\mathrm{K}}$, and $\mathrm{V} = \mathrm{PW}^{\mathrm{V}}$, where $\mathrm{P} \in \mathbb{R}^{N_p \times d_p}$ represents the matrix of flattened patches, $\mathrm{W}^{\mathrm{Q}}, \mathrm{W}^{\mathrm{K}} \in \mathbb{R}^{d_p \times d_k}$, and $\mathrm{W}^{\mathrm{V}} \in \mathbb{R}^{d_p \times d_v}$ are respectively the queries, keys, and values parameters matrices. After the projections, Q, K, and V are processed as in Equation 4, yielding a vector (embedding) A_i for each patch ($\mathrm{A} \in \mathbb{R}^{N_p \times d_v}$).



(a)



(b)

Figure 1: (**a**) Scheme of the attention mechanism. $\mathrm{Q} \in \mathbb{R}^{N_q \times d_k}$ represents the queries, $\mathrm{K} \in \mathbb{R}^{N_v \times d_k}$ the keys, and $\mathrm{V} \in \mathbb{R}^{N_v \times d_v}$ the values. The output is a $N_q \times d_v$ matrix. (**b**) Example of self-attention on $N_p = 16$ image patches inspired by the Vision Transformer (ViT). The patches P are flattened and projected as queries, keys, and values. The attention mechanism outputs an embedding A_i for each patch P_i.

In cross-attention, instead, the queries derive from a different source than keys and values. Figure 2 presents an example of cross-attention inspired by the DEtection TRansformer (DETR) [16]. Suppose that we want to detect a max of N_q objects within an image (i.e. inferring their bounding boxes and class labels) by using cross-attention. A simple strategy could be to divide the image into patches, flatten them, and then project them as keys and values, but not as queries. Q is instead a set of learned vectors, called object queries in this example, that is directly fed to the cross-attention module. As output, N_q vectors (one per object query) are returned as a matrix A and finally passed through a shared feed-forward network (FFN) predicting bounding boxes and class labels. The queries can therefore learn to look, e.g., for some specific patterns or in various specific positions inside the image and, when nothing is found, should predict the absence of an object.

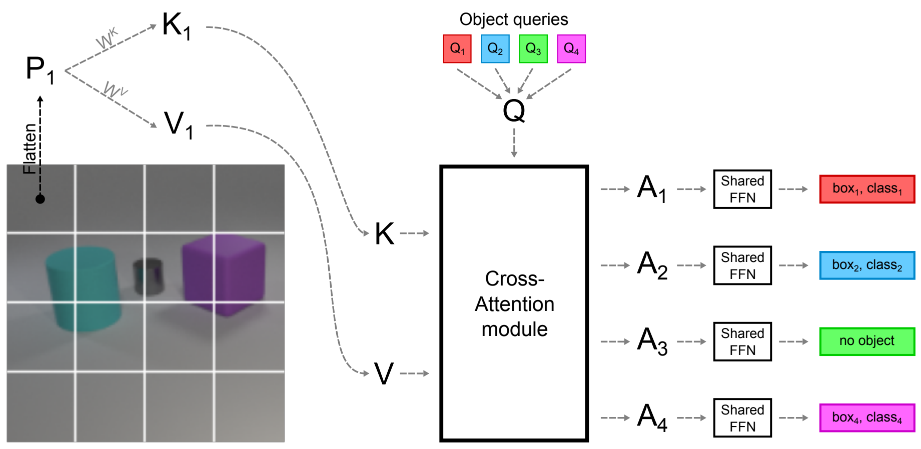


Figure 2: Example of cross-attention with $N_q = 4$ object queries inspired by the DEtection TRansformer (DETR). The patches P are flattened and projected as keys and values. The queries are 4 learned vectors. The attention mechanism outputs a vector A_i for each object query Q_i, which is then passed through a shared FFN that predicts bounding box and class label of a potential object.

In the last two examples, the presence of positional encodings (or embeddings) has been neglected. However, the attention mechanism is invariant with respect to the order of keys and values, meaning that permuting the rows of K and V (in the same way) would produce an equivalent output. If the queries are instead permuted, the output is permuted accordingly. Consequentially, considering again the examples in Figures 1b and 2, we should augment the flattened patches with information regarding their relative or absolute position in the image, so that the model is able to reliably know and exploit their locations. In the next subsection, a practical example involving positional embeddings in computer vision will be covered more in detail.

2.2 Slot Attention

The *Slot Attention* (SA) module is an architectural component proposed by Locatello et al. [5] in 2020 and it is based on the concept of cross-attention. Its aim is to bind a set of latent vectors called *slots* to different objects, seen as parts of the input perceptual representations through multiple iterations of a competitive attention mechanism. The set of N_s slots $\mathrm{S} \in \mathbb{R}^{N_s \times d_s}$ is initialized by independently sampling each from a Gaussian distribution with shared and learnable parameters $\mu, \sigma \in \mathbb{R}^{d_s}$, which allows Slot Attention to generalize the number of slots at test time. In the paper, the N_x input representations $\mathrm{X} \in \mathbb{R}^{N_x \times d_x}$ are feature vectors at the output of a CNN backbone augmented with positional embeddings. These inputs are normalized over both dimensions using layer normalization (LayerNorm) [17], then finally projected as keys and values $\mathrm{K}, \mathrm{V} \in \mathbb{R}^{N_x \times d}$ of a common dimension d through the linear transformations $k, v : \mathbb{R}^{d_x} \to \mathbb{R}^{d}$. At each of the T iterations, the slots from the previous round are (independently) normalized, mapped as queries by $q : \mathbb{R}^{d_s} \to \mathbb{R}^{d}$, and then refined. Figure 3 illustrates a single refinement step t of the Slot

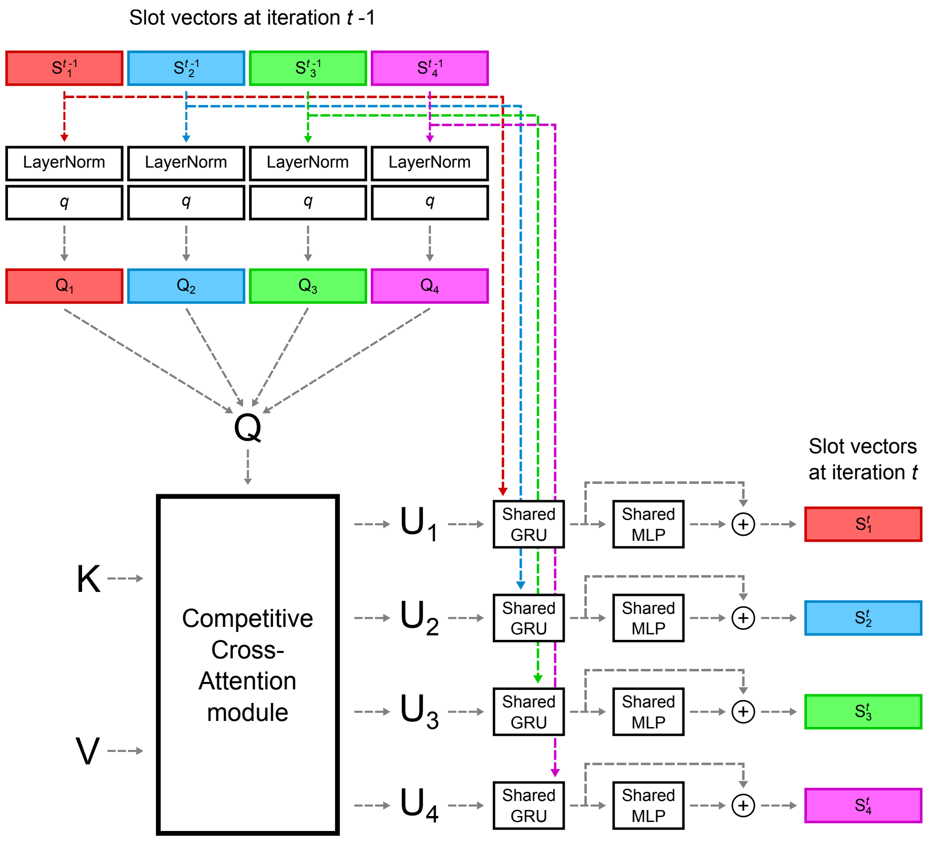


Figure 3: Illustration of the computations inside the Slot Attention module for a given iteration t.

Attention mechanism. As shown, the previous slots S^{t-1} (where S^0 are the sampled ones) are mapped as queries and fed to the competitive cross-attention module, which yields one vector U_i for each Q_i, referred to as *update*. The updates are then passed as inputs to a shared Gated Recurrent Unit (GRU) [18], while the associated slot vectors from S^{t-1} are set as hidden states. The outputs, which are the updated hidden states, are transformed by a shared MLP with a residual connection [19], resulting in the new slot vectors S^t. The competition between slots for explaining part of the input is induced by an additional normalization in the cross-attention mechanism, performed by using the softmax function over the queries before linearly normalizing over the keys. Formally, the attention coefficients are computed as

$$\hat{w}_{ij} = \frac{e^{\mathrm{M}_{ij}}}{\sum_{l=1}^{N_s} e^{\mathrm{M}_{lj}}} \quad \text{where} \quad \mathrm{M} = \frac{\mathrm{QK}^\top}{\sqrt{d}} \in \mathbb{R}^{N_s \times N_x} , \tag{5}$$

producing a probability distribution over the slots for each input vector. After that, the update vectors U_i are calculated as in Equation 1:

$$\mathrm{U}_i = \sum_{j=1}^{N_x} w_{ij} V_j \quad \text{with} \quad w_{ij} = \frac{\hat{w}_{ij}}{\sum_{l=1}^{N_x} \hat{w}_{il}} , \tag{6}$$

where the weights are normalized over the inputs so that $\sum_{j=1}^{N_x} w_{ij} = 1$.

Regarding the positional embeddings, the authors employ a grid with shape $H \times W \times 4$, where H and W are respectively the height and the width of the feature maps at the output of the CNN backbone (note that $H \times W = N_x$). As shown in Figure 4, the four channels of the grid represent linear gradients from 0 to 1 pointing in the four cardinal directions, so that every point on the grid is encoded by its normalized distances from the four borders. After flattening

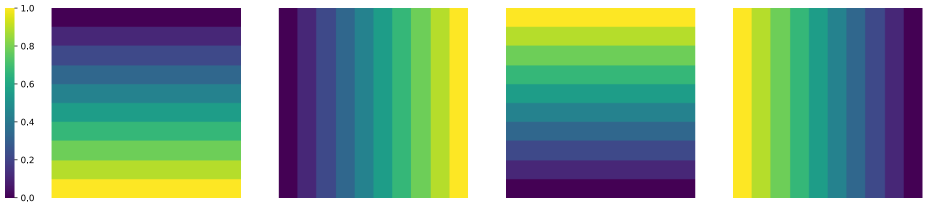


Figure 4: Visualization of the four channels of the positional grid used by Slot Attention. Here the grid has shape $10 \times 10 \times 4$. Each channel represents a linear gradient $[0, 1]$ pointing in one of the four cardinal directions. From the left: downward, right, upward, and left.

the grid as $G \in [0, 1]^{N_x \times 4}$, each 4-dimensional vector is linearly mapped to dimension d_x through $g : [0, 1]^4 \rightarrow \mathbb{R}^{d_x}$ and summed to its corresponding feature vector:

$$\mathrm{X} = \tilde{\mathrm{X}} + g(G) \;, \tag{7}$$

where $\tilde{\mathrm{X}} \in \mathbb{R}^{N_x \times d_x}$ denotes the feature maps from the CNN with flattened height and width dimensions.

It is possible to exploit Slot Attention (coupled with a backbone) as a structured encoder in an autoencoder architecture, such as the one in Figure 5. The model is trained end-to-end to represent an input image as a set of latent vectors that are decoded back to the original image. A shared spatial broadcast decoder [13] is used to decode the slots individually: each is repeated in a $H \times W$ grid, augmented with positional embeddings, and passed through a series of convolutional layers. The output is an $H \times W \times 4$ tensor, where the first three channels correspond to RGB components, while the last one is the alpha mask of the decoded object. The reconstructed image is computed as the sum of the RGB predictions weighted by the alpha masks. At training time, the model parameters are optimized to minimize the mean squared error (MSE), that is the sum of the squared differences between the reconstructed and target (input) pixels, divided by the number of pixels.

2.3 Slot-Centric Reference Frames

With *Invariant Slot Attention* (ISA), Biza et al. [10] presented a mechanism for introducing per-object invariances to pose transformations in the latent representations. To achieve this, slot-centric relative grids are produced by translating, scaling, and rotating the absolute positional encodings, with the aim of processing the feature vectors in canonical reference frames. The rotation invariance is not covered in this section as it is not employed in our method.

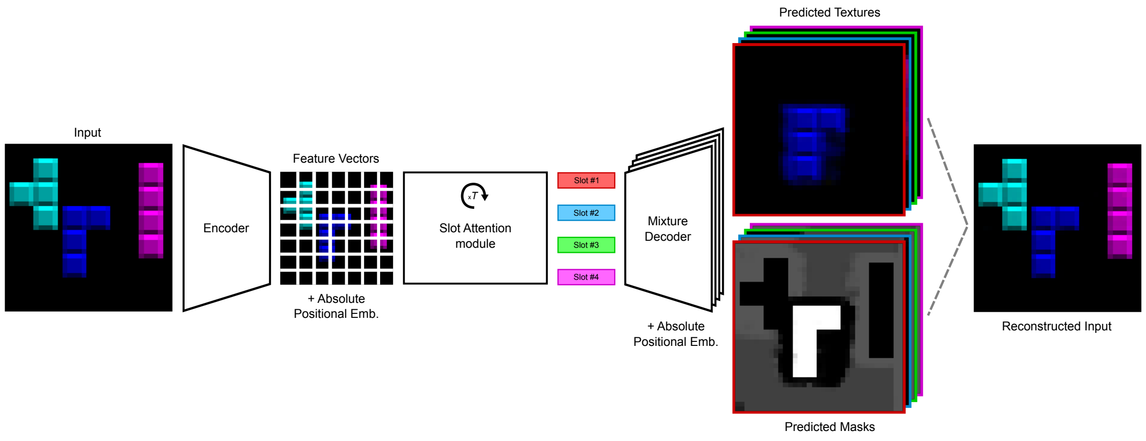


Figure 5: Slot Attention architecture.

At the first iteration of Slot Attention, translation and scaling factors $\mathcal{S}^i_{\text{pos}}$ and $\mathcal{S}^i_{\text{scale}}$ associated with the slot S_i can be randomly sampled or learned. The relative grid is then computed as

$$G^i = \frac{G - \mathcal{S}^i_{\text{pos}}}{\mathcal{S}^i_{\text{scale}}} \, , \tag{8}$$

and its projection is summed to the feature vectors $\tilde{\mathrm{X}}$ to obtain per-object inputs:

$$\mathrm{X}^i = \tilde{\mathrm{X}} + g(G^i) \, . \tag{9}$$

These are mapped as keys and values and processed by competitive cross-attention with the query derived from S_i, yielding an update vector and the attention weights w_i. During the next iteration, w_i is used to extract the new translation and scaling factors as

$$\mathcal{S}^i_{\text{pos}} = \frac{\sum_{j=1}^{N_x} w_{ij} G^i_j}{\sum_{j=1}^{N_x} w_{ij}} \tag{10}$$

and

$$\mathcal{S}^i_{\text{scale}} = \sqrt{\frac{\sum_{j=1}^{N_x} (w_{ij} + \epsilon)(G^i_j - \mathcal{S}^i_{\text{pos}})^2}{\sum_{j=1}^{N_x} (w_{ij} + \epsilon)}} \, , \tag{11}$$

which are again employed to calculate the relative grid of the i-th object. Note that, differently from SA, in ISA the grids have two dimensions (horizontal and vertical) instead of four, with values in the range $[-1, 1]$.

Since the slots vectors returned by ISA do not encode information about the position, scale, and rotation of the objects, it is necessary to use relative grids even during the decoding phase. In this case, the last translation and scaling factors are exploited to compute the grids as in Equation 8 and added to the broadcasted slots.

2.4 Sobel Filter

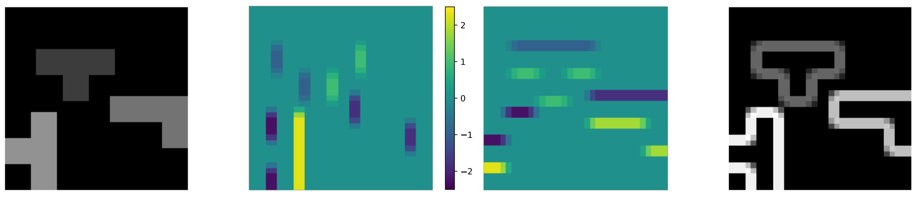
The Sobel filter [20], or Sobel–Feldman operator, was presented in 1968 by Irwin Sobel and Gary Feldman during a talk at the Stanford Artificial Intelligence Laboratory. This operator uses two 3×3 filters to approximate the horizontal and vertical derivatives of an image:

$$\mathcal{K}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \quad \text{and} \quad \mathcal{K}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} . \tag{12}$$

A 2-dimensional convolution operation with $\mathcal{K}_x$ and $\mathcal{K}_y$ is applied on the input image $\mathcal{X}$, producing two maps of the approximated horizontal and vertical derivatives at each pixel, respectively $\mathcal{G}_x = \mathcal{K}_x * \mathcal{X}$ and $\mathcal{G}_y = \mathcal{K}_y * \mathcal{X}$. Figure 6b shows an example of the resulting $\mathcal{G}_x$ and $\mathcal{G}_y$ of the grayscale image 6a. To obtain the final filtered input containing the detected edges, the maps are aggregated as

$$\mathcal{G} = \sqrt{\mathcal{G}_x^2 + \mathcal{G}_y^2} . \tag{13}$$

The filtered image associated with the previous example is shown in figure 6c.



(a) (b) (c)

Figure 6: Example of the Sobel filter applied to a simple grayscale image. (**a**) Input image $\mathcal{X}$. (**b**) Maps of the approximated horizontal ($\mathcal{G}_x$, left) and vertical ($\mathcal{G}_y$, right) derivatives of $\mathcal{X}$ at each pixel. (**c**) Filtered input image $\mathcal{G}$ computed combining $\mathcal{G}_x$ and $\mathcal{G}_y$ following Equation 13.

When dealing with RGB images, there are two possible approaches. The first is to simply convert the image to grayscale before applying the Sobel filter. The second, which we employ in our method, consists in applying the filter independently to each channel, then averaging the three resulting filtered images.

3 Related Work

Object-centric representation learning The architectural design we propose is aimed at models learning object-centric representations of visual scenes in an unsupervised or self-supervised manner, which are recently capturing more attention in the fields of computer vision and deep learning. A first line of work heading toward this direction is defined by AIR [21], which performs probabilistic inference employing a recurrent neural network (RNN) that attends and processes one object in a scene at a time. Subsequent literature extended this idea in order to solve some of its limitations, such as SQAIR [22], SuPAIR [23], and others [24–26]. Further advancements in object-centric representation learning include Tagger [27], Multi-Entity Variational Autoencoder (MVAE) [28], and Neural Expectation Maximization (N-EM) [29] with its extension R-NEM [30]. More recently, MONet [1], IODINE [2], and GENESIS [3, 4] achieved meaningful decomposition of non-trivial scenes characterized by a non-fixed number of objects, e.g. in the CLEVR dataset [31]. Finally, Slot Attention [5], along with various extensions [6–10]), introduces its iterative mechanism, emerging for being faster to train and more memory efficient than its predecessors, while matching or even outperforming the other methods.

Disentanglement in object-centric learning Within the object-centric representation learning literature, works such as [1,2,10,32,33] study and aim to achieve the disentanglement of different factors of variation. In particular, Biza et al. [10] propose an extension of Slot Attention (i.e. ISA) that learns object representations invariant to changes in position, scale, and rotation with the use of slot-centric reference frames, allowing for the explicit disentanglement of those three factors. We exploit the concept of slot-centric reference frames in our work to avoid position and scale information to flow into texture-related latent components. Furthermore, as in ISA, we possess a priori knowledge of which of the latent dimensions are associated with a disentangled factor, differently from, for instance, MONet [1] and IODINE [2], where this knowledge can only emerge after training. However, to the best of our knowledge, no research dealt with the explicit disentanglement of texture and shape factors in object-centric learning. The goal of our work is to address this last problem.

Texture and shape disentanglement Outside the scope of object-centric learning, approaches including [34–36] attempt to disentangle shape and texture in the domain of single-object images. Deforming Autoencoders [34] employs a pair of decoders, of which one synthesizes appearance in a deformation-free coordinate system, while the other estimates a deformation field that warps the texture into the input image. One aspect that aligns with our work is the adoption of two separate decoders for texture and shape synthesis. Lorenz et al. [35] aim, without supervision, to disentangle appearance and shape in the representations of multiple parts of a single object class. To do so, they set into the reconstruction task three invariance and equivariance constraints by exploiting texture and shape transformations of an input image. Differently, TSD-GAN [36] uses an adversarial framework to learn to both reconstruct the item in an input image and mix it with the item from another sample. These last two works share some similarities with our latent compositionality loss.

4 Methodology

The primary aim of our work is to induce object-centric models to learn representations characterized by the explicit, and thus known a priori, separation of features encoding texture and shape information. To allow for this, we propose an architectural design that can be adapted to suit different existent architectures, such as Slot Attention, IODINE, and MONet.

Consider a generic architecture for unsupervised object discovery, encoding an image into a set of latent representations (slots) and decoding them back as pairs of mask and texture reconstructions, schematized in Figure 7. We want

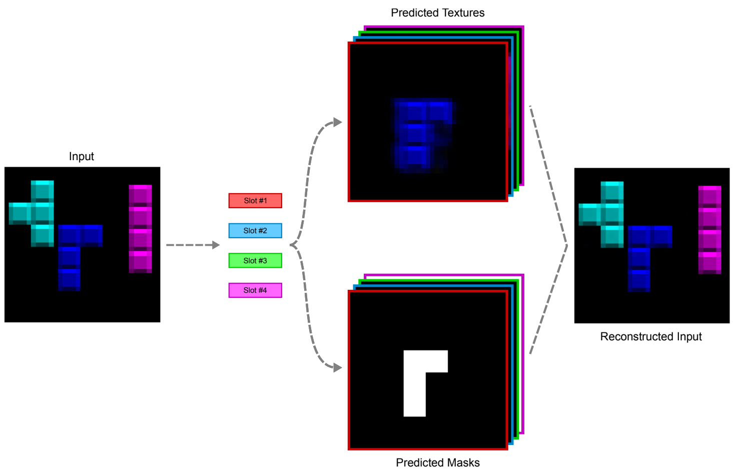


Figure 7: High-level architecture of a generic object-centric autoencoder for unsupervised object discovery. An input image is encoded into a set of latent vectors, which are then decoded back as pairs of mask and texture reconstructions.

the latent vectors to have a subset of their components encoding only texture information (e.g. material and color), and the remaining dimensions encoding only shape information. In order to enforce this disentanglement, a possible strategy is to encode the two mentioned subsets starting from different sources: the texture-related part from the original image, and the shape-related from a filtered version of it containing the borders of the objects. The architecture would therefore assume the structure presented in Figure 8, where we refer to these parts as *texture slots* and *mask slots*. However, directly decoding the entire vectors into the aforementioned pairs provides a low guarantee that the texture slots will not also include shape information (present in the input image), or that the mask slots will be even exploited by the decoder. Instead, following the architectural design illustrated in Figure 9, the masks can be inferred exclusively based on the shape-related components, ensuring that these will include the necessary shape information. Then, the complete representations are decoded into texture reconstructions. Since the shape already has to be encoded in one subset of the components, there is now a higher guarantee that the decoder will learn to exploit their information, leading the texture-related components not to encode it too. Information about scale and position can be either included

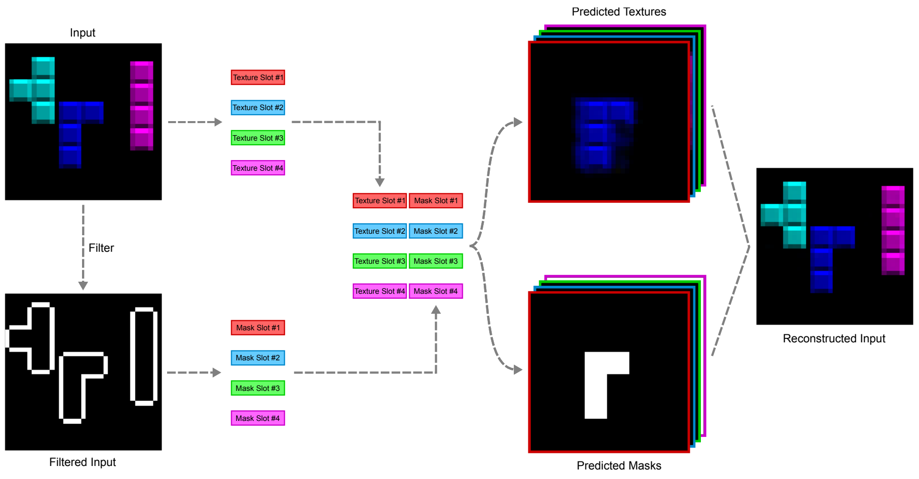


Figure 8: High-level architecture of an object-centric autoencoder for unsupervised object discovery. Each object is represented by a latent vector, of which one part (texture slot) should contain texture information extracted from the input image, while the second part (mask slot) shape information extracted from the filtered image. A single decoder maps the latent vectors into mask and texture reconstructions. This architectural design provides a low guarantee that the texture slots will not also include shape information, or that the mask slots will be even exploited by the decoder.

in the shape-related parts of the vectors or disentangled such as in ISA.

To filter the input image two strategies are followed. The first uses the previously introduced Sobel filter, producing a map of the edges in the image, while the other involves a learned bottleneck between the image and the object-centric encoder. The latter projects the RGB image into a single channel that is supposed to act as an edge (or border) detector. Further details are presented in the following subsection.

We evaluate our idea by applying it to Slot Attention and denote the resulting architecture as *Disentangled Invariant Slot Attention* (DISA), covered in detail in Section 4.2.

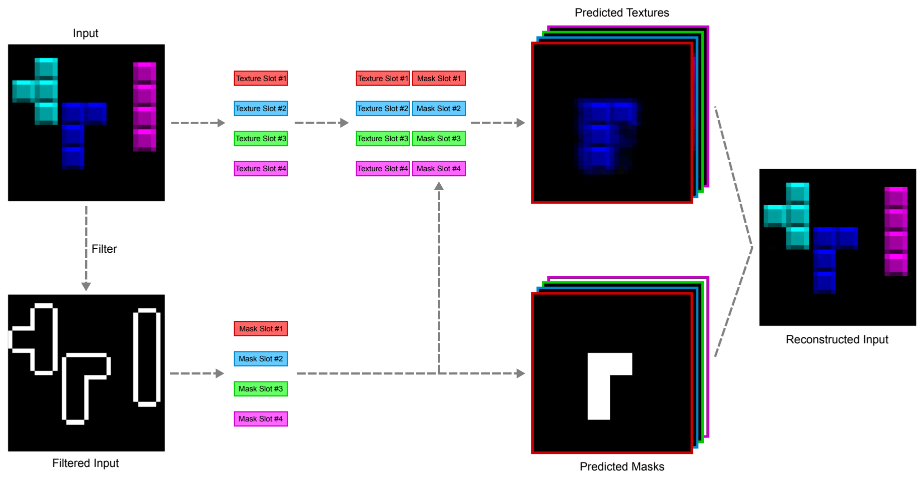


Figure 9: Proposed architectural design of an object-centric autoencoder for unsupervised object discovery. Each object is represented by a latent vector, of which one part (texture slot) should contain texture information extracted from the input image, while the second part (mask slot) shape information extracted from the filtered image. There are two decoders, one predicting the masks from the mask slots and the other decoding the complete latent vectors into texture reconstructions. This design provides a better guarantee, compared to the one in Figure 8, that the shape information will be encoded exclusively in the mask slots.

4.1 Bottleneck Filter

As mentioned, we want to avoid encoding texture-related information in the mask slots by filtering it out from the input image. This could be achieved with an ideal filter able to produce segmentation masks such as the one in Figures 8 and 9, which would simply not provide that information to the encoder of the mask slots. We ignore models pre-trained with supervised learning on this task as we assume the complete absence of labels. While the Sobel filter can be helpful, it does have a drawback in that it takes into account all the edges, including the ones of objects' textures. An alternative strategy is to learn a single-channel bottleneck that would act as the filter. The bottleneck is trained with the rest of the model using the reconstruction loss and is pushed to consider all the edges necessary to allow the identification of the entities in the scene. Figure 10 shows the architecture of the bottleneck filter. An RGB image passes through the first 2d convolutional layer yielding a set of feature maps that are fed to the second convolutional layer. This computes another set of feature maps that are finally passed through the last layer having one kernel and thus producing the filtered single-channel image. Every layer has SAME padding, hence the input resolution is maintained, and its output is activated by the Leaky ReLU function. Moreover, at the second and third layers, the image is concatenated to the feature maps along the depth (channel) dimension. The middle layer can be repeated multiple times in order to allow for additional processing before the final convolution.

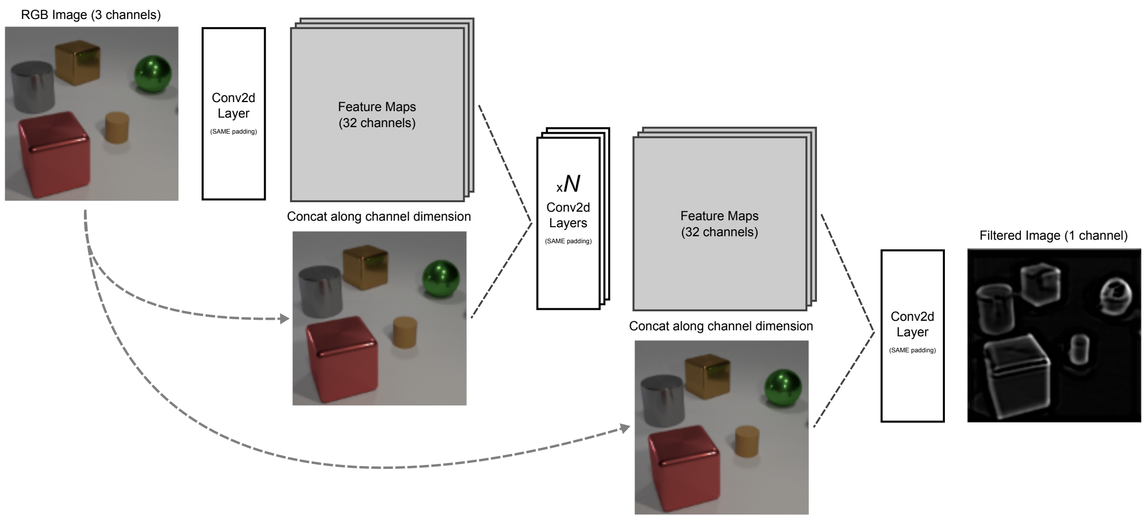

RGB Image (3 channels)
Conv2d Layer
(SAME padding)
Feature Maps (32 channels)
Concat along channel dimension
xN
Conv2d Layers
(SAME padding)
Feature Maps (32 channels)
Concat along channel dimension
Conv2d Layer
(SAME padding)
Filtered Image (1 channel)

Figure 10: Architecture of the bottleneck filter. An RGB image passes through the first conv2d layer and yields a set of, e.g., 32 feature maps. These are fed to the second conv2d layer that computes the second set of feature maps, finally passed through the last layer having one kernel and thus producing the filtered single-channel image. Every layer has SAME padding and, at the second and third layers, the image is concatenated to the feature maps along the depth dimension. The middle layer can be repeated multiple times in order to allow for additional processing before the final convolution.

4.2 Disentangled Invariant Slot Attention

Disentangled Invariant Slot Attention (DISA) is an object-centric architecture that follows the design presented in Figure 9 and leverages both the Slot Attention mechanism and the slot-centric reference frames of ISA. DISA (shown in Figure 11) is composed of two CNN encoders extracting two sets of feature vectors, one from the input image and the other from its filtered version. The SA module serves to identify the objects from the second set of feature vectors, producing an attention mask and a mask slot for each of them. Since the feature vectors #2 are augmented with absolute positional embeddings, the mask slots are supposed to encode shape, position, and scale information. With the attention masks, we can compute position and scaling factors such as in Section 2.3, that allow us to obtain the objects' relative grids used to augment the feature vectors #1. Following Equation 6, the texture slots are produced by projecting the augmented feature vectors as values and aggregating them with the normalized attention masks as weights. Therefore, the texture slots should encode texture information according to the attention masks, while ignoring absolute position and scale due to the use of relative frames. Finally, a pair of decoders is used to reconstruct the image: the mask slots are fed to decoder #2 that infers the set of masks, while the complete slot vectors (concatenation of mask and texture slots) are given as input to decoder #1 in order to predict the textures coherently with the shape of the objects. As in Slot Attention, the final predicted image is the combination of texture reconstructions weighted by the masks.

4.3 Latent Compositionality Loss

Since there is still no strong guarantee that a model designed following Figure 9 will learn representations characterized by the disentanglement of features encoding texture and shape information, we propose a self-supervised loss that can be paired with the usual reconstruction loss to explicitly push the model to learn such representations. The idea is to encode an image, permute the textures of the objects in it while maintaining their shapes, positions, and sizes fixed, then decode these permuted vectors back. To allow for this permutation, the representations should be object-centric, with their components related to texture information known a priori as, for instance, in our proposed architectural design. If we had a target image associated with these permuted vectors, we could simply optimize the model with an additional MSE term between the (permuted) reconstructed and target images. However, given that we have no access to that information, an alternative approach is to work in the latent space. In fact, it is possible to encode the reconstructed image and constrain the obtained latent object vectors to be as close as possible to the initially permuted representations. An illustration of this strategy, which we refer to as *Latent Compositionality Loss* (LCL) can be found in Figure 12. Intuitively, the model should learn to reconstruct the switched textures well enough for the encoder #1 to recognize them again, but also to adapt them to the assigned shapes in such a way that the encoder #2 can still detect them. Note that we use the L_1 loss in our experiments, but the L_2 loss would also be suited for the task.

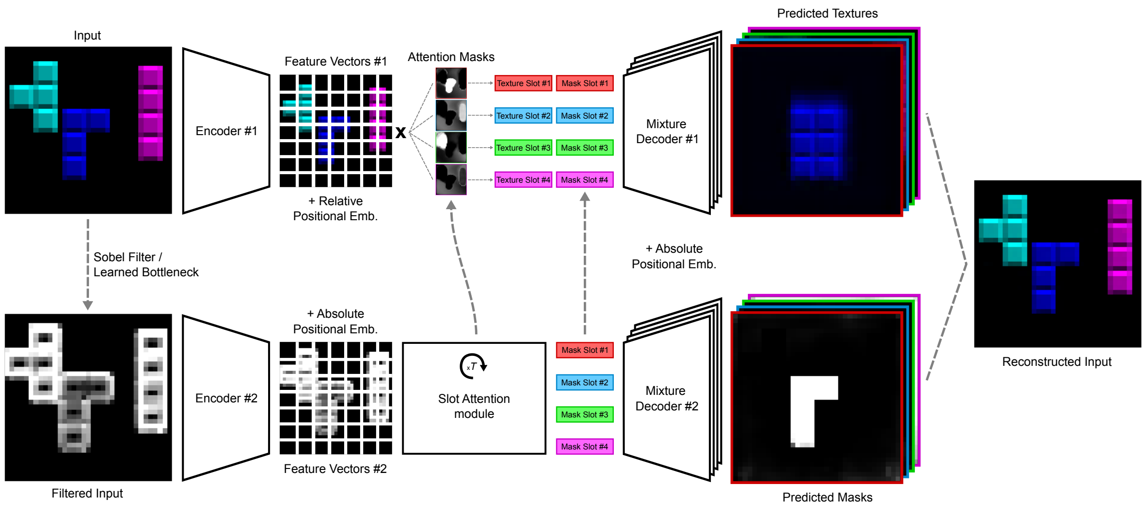


Figure 11: Architecture of Disentangled Invariant Slot Attention (DISA). Each object is represented by a latent vector, of which one part (texture slot) should contain texture information extracted from the input image (encoder #1), while the second part (mask slot) shape information extracted from the filtered image (encoder #2). Specifically, the mask slots are obtained from the feature vectors extracted by encoder #2, augmented with absolute positional embeddings, and fed as keys and values to the Slot Attention module. The texture slots arise from the combination of the attention masks extracted from the last iteration of the SA module and the feature vectors from the encoder #1, augmented with relative positional embeddings. There are two decoders, one predicting the masks from the mask slots (decoder #2) and the other decoding the complete latent vectors into texture reconstructions (decoder #1).

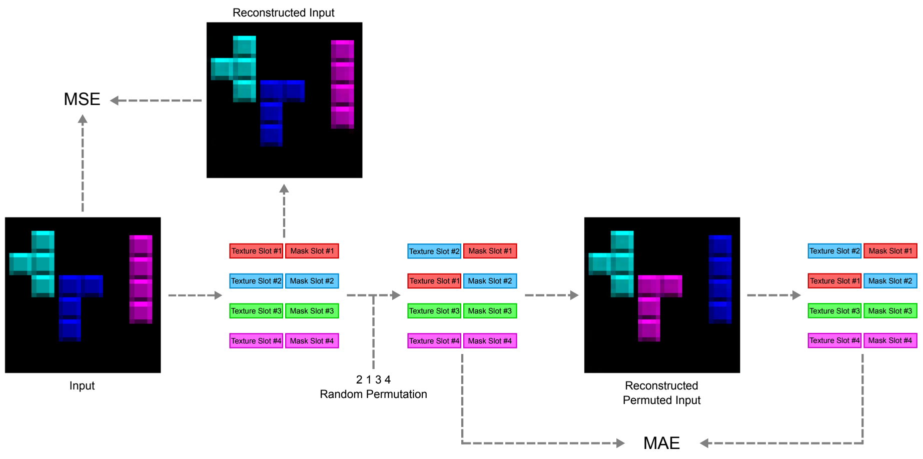


Figure 12: Illustration of the proposed latent compositionality loss (LCL). First, an input image is encoded into a set of object-centric representations, in which the components related to texture and shape information are known a priori. Then, the texture-related parts of these object vectors are randomly permuted, and the new set of permuted representations is decoded into an image that, ideally, should depict the input image with the object textures swapped accordingly to the sampled permutation. This image is finally encoded with the same initial encoder (or encoders), and the extracted representations are paired through the Hungarian matching algorithm with those earlier permuted. Precisely, we match the slots using the cosine similarity of their mask-related components. A mean absolute error (MAE) is employed as loss function to minimize the distance between these pairs of object vectors. Note that, as this loss is used along with the reconstruction loss, the first (non-permuted) set of representations is additionally decoded into the reconstructed input image: in this case a mean squared error (MSE) between input and reconstructed images can be utilized as loss function.

5 Experimental Setup

We first evaluate DISA on the task of unsupervised object discovery to empirically demonstrate its competitiveness with the baseline model, namely Slot Attention. The Adjusted Rand Index, or ARI [37, 38], is used in this case to measure the scene decomposition ability based on the predicted masks, while the quality of the reconstructions is assessed with the mean squared error (MSE). Subsequently, we present different quantitative and qualitative results that allow us to gain knowledge about the degree of disentanglement between dimensions encoding texture and shape information in its latent space. These represent the core experiments of this work, as they investigate the property we aim to induce in the representations learned by our models. We additionally conduct a brief analysis of the OOD capabilities of the bottleneck filter.

In this section, we present the datasets and training setups used in our experiments. The results are reported in Section 6.

5.1 Datasets

We evaluate DISA on three multi-object synthetic datasets [39]: Tetrominoes, Multi-dSprites, and CLEVR.

Tetrominoes consists of 35×35 RGB images presenting 3 "Tetris"-like blocks and a black background. The blocks are characterized by one of 6 possible colors (red, green, blue, yellow, magenta, cyan) and a shape and orientation within 19 possible. Figure 13 shows a few examples extracted from the dataset. The tetrominoes can appear anywhere in the image, but cannot overlap with each other. Furthermore, multiple shapes/orientations/colors of the same kind can be included in a single sample. We employ a total of 60K samples for the training set and 320 for the test set, in line with [2, 5].

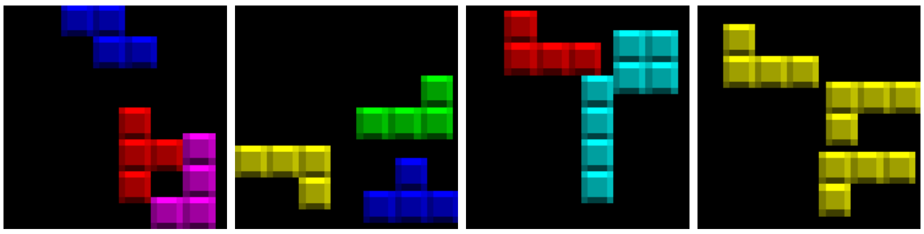


Figure 13: Samples extracted from both train and test partitions of the Tetrominoes dataset.

Multi-dSprites is a dataset based on dSprites [40], where three types of shapes, namely oval, heart, and square, are represented over a plain background. Both the sprites and the background are colored with randomly sampled RGB values. The number of entities in the image can vary from 1 to 4 excluding the background, and partial occlusion can be present. Each image has a resolution of 64×64. Examples from the Multi-dSprites dataset are visualized in Figure 14. Again, we employ a total of 60K samples for the training set and 320 for the test set, as in [2, 5].

CLEVR, introduced by Johnson et al. [31], is a collection of 240×320 images of 3D scenes containing a number of objects ranging from 3 to 10 (excluding the background). There are three types of shapes, i.e. cube, sphere, and cylinder, two possible sizes (small and large), eight colors (gray, red, blue, green, brown, purple, cyan, and yellow), and two materials (shiny "metal" and matte "rubber"). We also employ a filtered version of this dataset, called CLEVR6, that has a maximum of 6 entities in the scenes, background excluded. The number of training samples, in this case, is 70K, while the test samples are 15K. In all our experiments, a center crop of 192×192 is applied, followed by a resize to a resolution of 128×128.

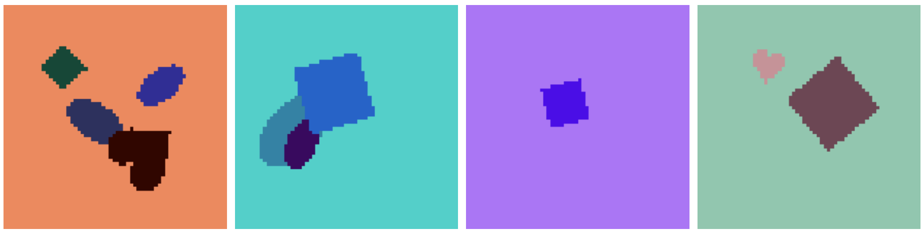


Figure 14: Samples extracted from both train and test partitions of the Multi-dSprites dataset.

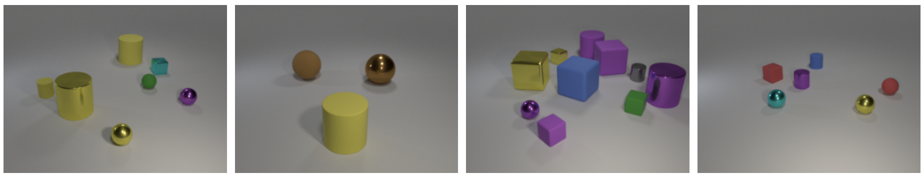


Figure 15: Samples extracted from both train and test partitions of the CLEVR dataset.

5.2 Training Setup

Object discovery SA and DISA are trained as autoencoders on the three datasets mentioned above for an equal number of steps. In the case of CLEVR, we use the version filtered to a maximum of 6 objects, i.e. CLEVR6. To take into account the stochastic nature of the experiments, we run three repetitions for each configuration. As already mentioned, the goal of this work is not to obtain state-of-the-art results in unsupervised object discovery, but to induce a specific property in the model while maintaining the performances of the baseline. Therefore, compared to previous work [1, 2, 5, 8, 9], we train for a reduced number of steps to save computations and thus energy consumption. On Tetrominoes, we perform roughly 100K steps (107 epochs $\times$ 938 batches), on Multi-dSprites around 250K (267 epochs $\times$ 938 batches), and on CLEVR6 nearly 150K (274 epochs $\times$ 547 batches). A batch size of 64 and a learning rate of 4×10^{-4} (using Adam as optimizer [41]) are employed on all datasets. An initial warm-up and an exponential decay schedule are applied to the learning rate. As in SA, we use the mean squared error as the reconstruction loss function. The number of iterations in the Slot Attention mechanism is fixed to 3, while the number of slots is set to 4 in Tetrominoes, 6 in Multi-dSprites, and 7 in CLEVR6. Moreover, the slots are initially sampled from a learned distribution, hence we do not directly learn them. We set the dimensionality of the slot vectors to 64 and, in DISA, we define the texture slot as the first half and the mask slot as the second half (32 components each) of a representation. Apart from the number of steps, these hyperparameters are equal to the ones used in [5] for the object discovery task. Note that, due to memory limitations, on CLEVR6 we train on two NVIDIA GeForce RTX 2080 Ti GPUs with 11GB of RAM (data parallelism) instead of a single one. The results of the object-discovery task are presented in Section 6.1.

When employing the LCL along with the reconstruction loss on Tetrominoes, we use learned initial slot vectors instead of sampled ones in order to improve the consistency between the permuted original representations and those encoded from the reconstructed permuted image.

Property prediction We train simple MLPs to predict the shapes and textures (i.e. color and material) of the objects in a scene by exploiting their representations extracted using DISA (pre-trained for object discovery). Specifically, for each property, we train an MLP having a single hidden layer of 256 nodes (activated with ReLU). The input layer has 32 nodes because only the first (texture slot) or second (mask slot) half of a representation is used to predict an object property. On every dataset, we perform about 10k steps with a batch size of 64 and a learning rate set to 3×10^{-4} (the optimizer is again Adam). Since we deal with categorical variables, the cross-entropy loss is employed with all the properties. The only exception is the color in the dataset Multi-dSprites, which can assume any value in the RGB space, and therefore has to be tackled as a regression problem. We decide to use the MSE as the loss function in this case. In Section 6.2 the results of the property prediction task are shown.

6 Results

6.1 Object Discovery

These evaluations aim to investigate whether DISA is competitive with the baseline (Slot Attention) on the task of unsupervised object discovery. The results are averaged over the three trained repetitions of each configuration. Additionally, each test image is evaluated 5 times to better deal with stochasticity. DISA-SB and DISA-BN refer to, respectively, DISA with Sobel filter and with bottleneck filter, while DISA-SB-LCL indicates DISA-SB trained with the additional latent compositionally loss. The application of the LCL is limited to DISA-SB on Tetrominoes in this work, with no necessity observed for its integration on Multi-dSprites, as evidenced by the findings presented in Section 6.2. Conversely, its application on CLEVR remains to be addressed in future research. Table 1 summarizes the ARI scores computed for the foreground objects. It is clear that, on the considered datasets, our model matches and even surpasses Slot Attention at decomposing the images into objects. In fact, in all four cases, either the Sobel or bottleneck configuration of DISA is able to reach higher ARI scores than SA. Note that, in Multi-dSprites, the poor results (of both SA and DISA) are mainly caused by the tendency to divide some entities into two or more slots when the number of objects in the scene is lower than the one of slots. However, the results in [5] suggest that, even on Multi-dSprites, more extensive training most probably leads to more stable decompositions. In Table 2, the mean squared error between the input and reconstructed test images is reported in order to measure the ability of DISA to decode the images. Even in this case, DISA achieves better performances than the baseline on all the experimented datasets.

	Tetrominoes	Multi-dSprites	CLEVR6	CLEVR
SA	98.4 ± 0.6	55.9 ± 26.1	92.2 ± 0.3	92.5 ± 1.3
DISA-SB	98.1 ± 0.8	59.3 ± 2.2	**95.7** ± 1.9	**94.0** ± 1.1
DISA-BN	98.6 ± 0.2	**60.5** ± 3.2	80.9 ± 17.5	79.23 ± 18.5
DISA-SB-LCL	**99.2** ± 0.3			

Table 1: Adjusted Rand Index (ARI) score (↑) for the task of unsupervised object discovery. The values are averaged over 3 repetitions and represented as percentages, in the format mean ± stddev. As in [1–3,5], the results exclude the background and only consider the foreground objects. DISA models augment the input images with a random plain noise during training, while SA does not. The performances of SA do not improve with noise addition on the considered datasets. More details on this can be found in the last paragraph of Section 6.1. Note that the LCL is only applied to DISA-SB on Tetrominoes.

Plain noise trick On Tetrominoes and CLEVR, DISA presents an issue in that it frequently tends to learn masks that, instead of precisely segmenting an object, select areas of arbitrary shape around the entities. Despite maintaining good decomposition capabilities as the objects get correctly divided into separate slots (foreground ARI score of 98.2 ± 0.5), this behavior can hinder the

	Tetrominoes	Multi-dSprites	CLEVR6	CLEVR
SA	6.77 ± 2.01	22.97 ± 19.06	5.90 ± 0.50	10.20 ± 0.29
DISA-SB	9.07 ± 3.72	**8.33** ± 0.68	**3.27** ± 0.09	7.30 ± 2.83
DISA-BN	**4.03** ± 0.58	9.07 ± 0.21	4.30 ± 1.31	**6.83** ± 1.93
DISA-SB-LCL	6.50 ± 1.72			

Table 2: MSE (↓) for the task of unsupervised object discovery. The values are averaged over 3 repetitions and represented in the format mean ± stddev. All the numbers in the table are scaled by 10^4. DISA models augment the input images with a random plain noise during training, while SA does not. The performances of SA do not improve with noise addition on the considered datasets. More details on this are included in the last paragraph of Section 6.1. Note that the LCL is only applied to DISA-SB on Tetrominoes.

shape information from being encoded in the mask slots. Consequently, the texture slots would have to include shape information, leading therefore to an incorrect disentanglement. We can notice that the issue seems to arise when a dataset is characterized by a fixed background, but not with a dynamic one as in Multi-dSprites. The simple introduction of random plain noise added to the input images (Figure 16) can avoid this problem: in fact, the intuition behind the solution is to induce slight changes in the background texture, so that it is no longer fixed, while however not heavily altering the foreground objects. We hence adopt this augmentation trick when training our DISA models, and also evaluate its effect on the original Slot Attention. Regarding SA, we find that it does not improve its performance and, since SA is not affected by the above problem, there is no need for the augmentation trick to be used. Additional experiments are required to gain more insights on this, for instance, to understand whether fixed non-plain backgrounds are as affected as fixed plain ones.

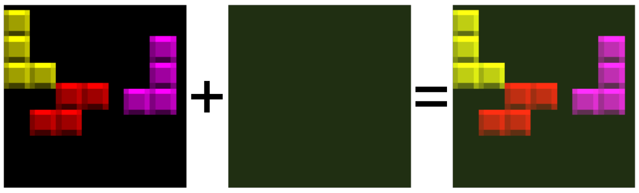
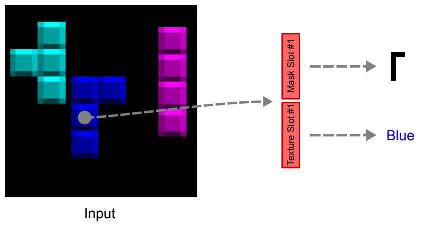


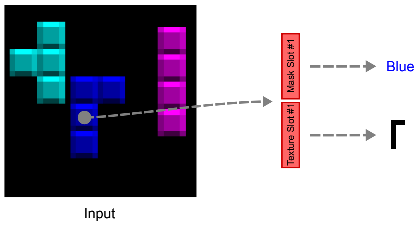
Figure 16: Example of the plain noise augmentation applied on an image from Tetrominoes. The augmentation induces slight changes in the color of a fixed background, turning it dynamic and helping the model avoid suboptimal solutions.

6.2 Disentanglement Analysis

In order to study the effectiveness of DISA in constraining the texture information in the first half of the object vectors and the shape information in the second half, we exploit the task of property prediction in two ways. Firstly we train multiple MLPs, each predicting one property describing textures or shapes of the objects based on the respective components, such as the shape of a tetromino given its mask slot. This task is schematized in Figure 17a. It is expected from this configuration to achieve very good results, as we leverage the halves of the representations that correspond to the properties we aim to infer. Instead, if we switched the parts and tried to predict, e.g., the color of an object from its mask slot, we would ideally expect to obtain poor results. An "inverse" property prediction task is carried out to gain insight into this aspect (Figure 17b). We analyze each of the three repetitions independently and report in this section only the results related to the first repetition. In line with previous literature [2,42,43], we measure the performances using the accuracy for categorical variables and the coefficient of determination (R^2 score) for numerical variables (i.e. only the color in Multi-dSprites). To take into account the stochasticity introduced by the sampling of the initial slots, we report the mean and standard deviation of accuracy/R^2 computed over ten repetitions on the test set. Note that, since the objects can be represented in any order, the ground truth labels must be associated with the correct predicted slots. This is done by computing cosine similarities between the target and predicted masks, then coupling each ground truth label with the output of its most similar slot.



(a) Property prediction



(b) Inverse property prediction

Figure 17: This figure illustrates the two property prediction tasks we propose to quantitatively study the degree of texture and shape disentanglement in the representations of DISA. (**a**) Prediction of an object property based on the associated components. For instance, the color from the texture-related latent features. (**a**) Inverse property prediction task, where properties are predicted based on the "wrong" part of the object representation, e.g., the color from the shape-related components.

Tetrominoes A visualization of these experiments on Tetrominoes is reported in Figure 18. Here we can see that, as expected, all three DISA models reach near-perfect accuracy on both shape and color prediction when exploiting the correct halves. When inverting them, instead, we notice that the shape prediction accuracy is not affected, while the color accuracy drops to the score of a random guess. Therefore, on one hand, shape information is incorrectly included in the texture slots while, on the other hand, the color information is successfully restricted from flowing to the mask slots. By contrast, training with reconstruction and latent compositionality loss leads DISA to correctly encode texture and shape information in not intersecting dimensions of its latent space, achieving thus the goal of this work on Tetrominoes. This is empirically proved by the fact that, in the inverse property prediction task, the accuracy of both shape and color is approximately identical to that of a random guess.

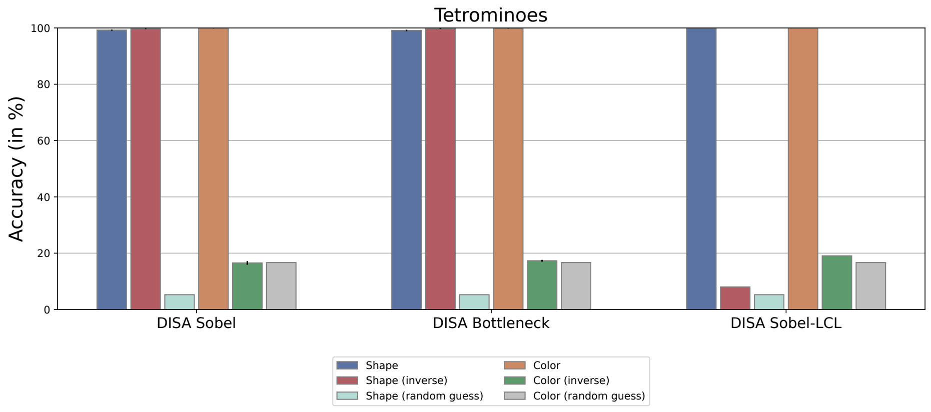


Figure 18: Quantitative results of DISA (first training repetition) on the normal and inverse property prediction tasks (Figure 17) conducted on Tetrominoes. We report the measurements for DISA with Sobel filter (left), learned bottleneck (center), and Sobel filter along with LCL (right). The predicted properties are shape and color, both categorical, for which we show the prediction accuracy (in %) on the two tasks. A random guess is used as a baseline for comparison. The color information is consistently constrained in the correct latent components, while for the shape it only occurs when the LCL is employed.

Multi-dSprites Differently from Tetrominoes, on Multi-dSprites (Figure 19) we find that DISA is able to disentangle shape and texture dimensions without the help of a more explicit loss such as the LCL. In fact, in this case, the inverse shape prediction accuracy of Sobel and bottleneck models are close to that of a random choice. Again, the texture information is correctly constrained in the first half of the latent components, as the negative R^2 scores indicate that the inverse color predictions are notably worse than those of a model constantly inferring the mean target value.

CLEVR6 On CLEVR6 (Figure 20) we see results similar to those on Tetrominoes without LCL: the accuracy of the inverse shape predictions is close to the normal one, while the accuracy of inverse color and material predictions are near those of random guesses.

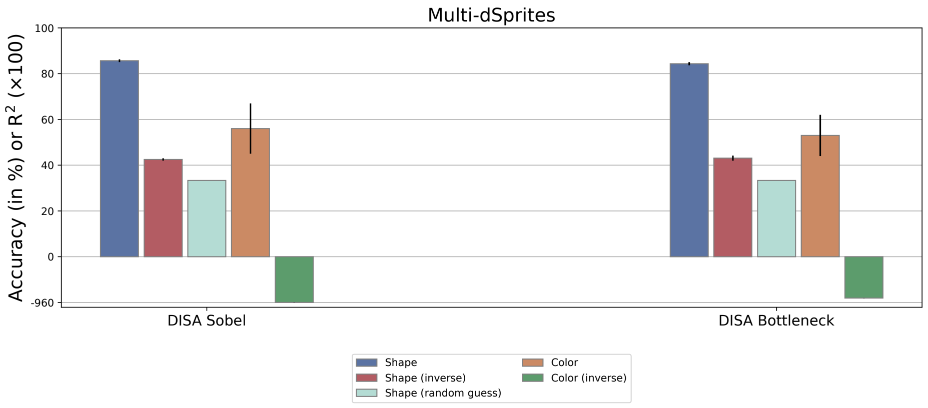


Figure 19: Quantitative results of DISA (first training repetition) on the normal and inverse property prediction tasks (Figure 17) conducted on Tetrominoes. We report the measurements for DISA with Sobel (left) and bottleneck (center) filter. The predicted properties are shape (categorical) and color (numerical). For the shape, the prediction accuracy (in %) on the two tasks is used and compared with a baseline random guess. With the color, we show the R^2 score. Both models are able to disentangle color and shape components correctly.

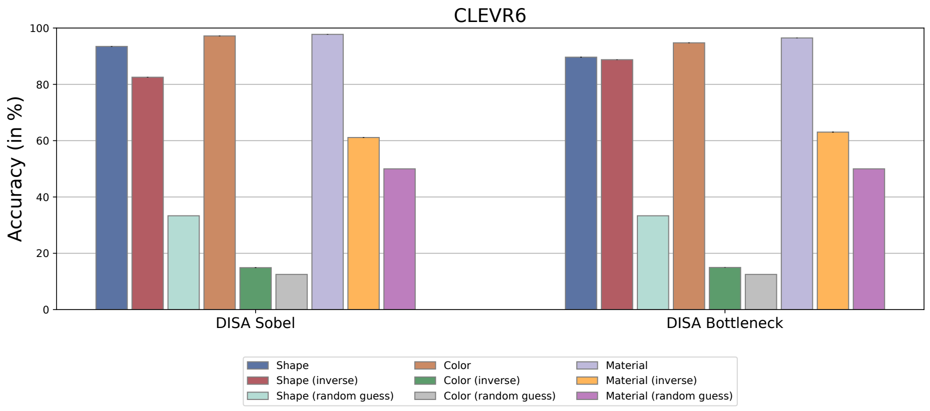
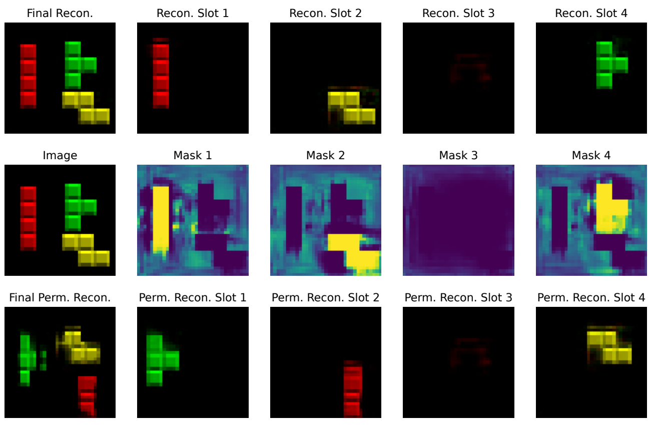


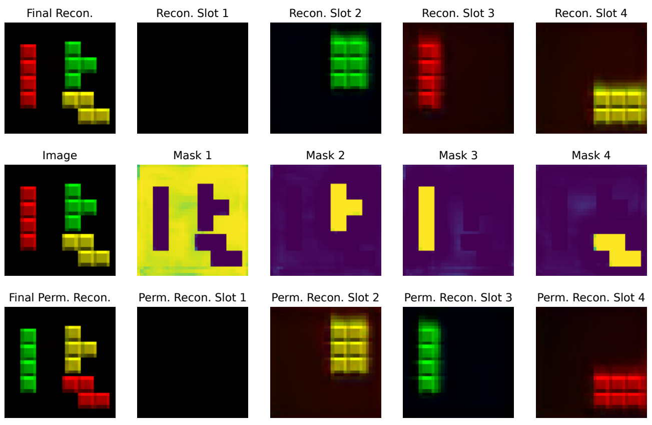
Figure 20: Quantitative results of DISA (first training repetition) on the normal and inverse property prediction tasks (Figure 17) conducted on Tetrominoes. We report the measurements for DISA with Sobel (left) and bottleneck (center) filter. The predicted properties are shape, color, and material, all categorical, for which we report the prediction accuracy (in %) on the two tasks. A random guess is used as a baseline for comparison. The color and material information is consistently constrained in the correct latent components, while it does not occur for the shape.

6.3 Compositionality Capabilities

In this section, the compositionality capabilities of DISA are explored through a series of qualitative results. Precisely, for a given image from the test set, we take the representations of the objects, randomly permute their texture slots, and decode the permuted representations. Ideally, the model would output a reconstruction where the shapes, positions, and sizes of the entities are preserved, while their textures are different but adapted to fit the same masks. We expect this experiment to back the results presented in Section 6.2. Furthermore, we show that these capabilities extend to the generation of new objects by transferring, interpolating, noising, or sampling textures and shapes of the objects in a scene.



(a) DISA with Sobel filter (rep. 1/3)
Permutation applied to the texture slots: 4-1-3-2.



(b) DISA with Sobel filter and LCL (rep. 1/3)
Permutation applied to the texture slots: 1-4-2-3.

Figure 21: Examples showing the compositionality capabilities of DISA models trained on Tetrominoes. The center row contains the input image and the predicted object masks. We set the number of slots as the number of entities in the image, i.e. four included the background. At the top, the combined reconstruction and the individual textures associated with each slot are shown, while the bottom row presents the reconstructions after permuting the texture slots. (**a**) DISA with Sobel filter resulting from the first training repetition. The two remaining repetitions and all three runs of DISA with bottleneck filter display equal behavior, thus are not included here. (**b**) DISA with Sobel filter and latent compositionality loss obtained from the first repetition. Both the other two repetitions exhibit similar behavior.

Tetrominoes Figure 21 reports the results on an image extracted from the test set of Tetrominoes. On the left (Figure 21a) is shown the model resulting from the first repetition of DISA with Sobel filter, which displays equal behavior to the two remaining repetitions and all three runs of DISA with bottleneck filter. Here, transferring the texture slot of one object to another translates into passing both color and shape, coherently with Figure 18 as shape information is also encoded into the first half of the representations and the color only in the second half. Furthermore, the position is correctly determined by the mask slots, meaning that the relative reference frames are working as intended. Explicitly inducing the disentanglement through the LCL, on the contrary, allows DISA to achieve perfect compositionality capabilities (21b). This result is consistent in all three repetitions, despite the decoding of slightly different masks and textures with respect to the ones in this figure. As visible, the colors are in fact transferred between objects without affecting the starting shapes. To accomplish this, the learned strategy is to decode textures that circumscribe the predicted shapes with rectangles, which permits the use of the same texture for all the tetrominoes that share an equal circumscribed (again, by rectangles) area. However, this behavior is not required as the texture decoder is also conditioned by the mask slots, implying that it is allowed to reconstruct textures already characterized by the correct shape. This is observable in the second and third repetitions.

Figure 22 shows the ability of DISA to generate new objects by altering the texture and shape of an object in the given input image. In Figure 22a, we change the green square by interpolating its texture with that of the blue s-shaped tetromino. The obtained texture maintains the original structure, perfectly fitting the associated shape, but switches the color to a combination of green and light blue. Instead, by noising the mask slot encoding the shape of the green square, we can represent new shapes such as the one in Figure 22b. In this case, it is also interesting to notice that the decoded texture correctly changes size according to the new shape.

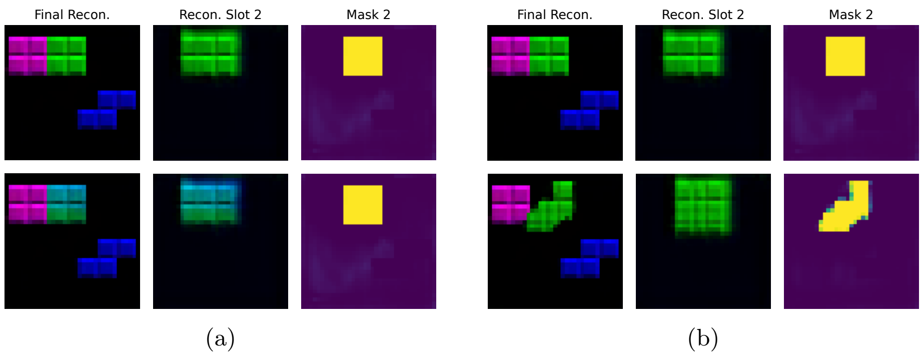
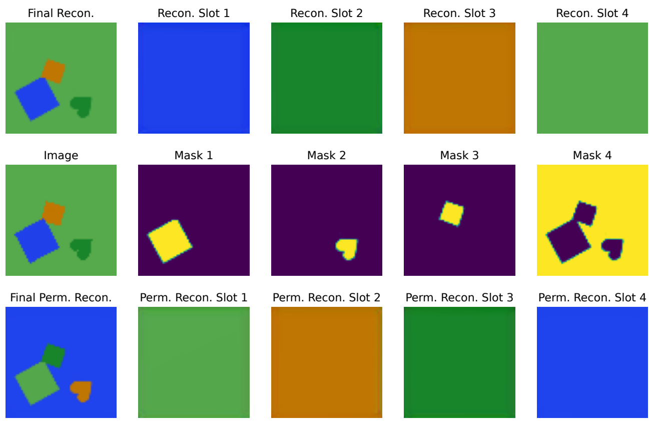
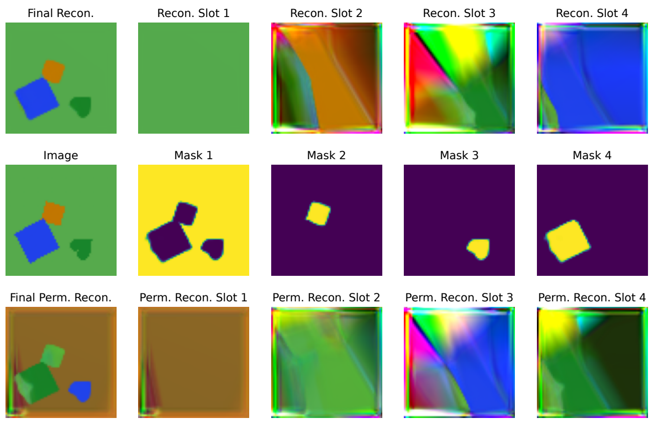


Figure 22: Examples of DISA's capability to generate objects with new textures and shapes on Tetrominoes. The top row contains the reconstruction of the input image, the predicted texture associated with the green square, and its predicted mask. At the bottom, the same visualization is presented after altering the texture or shape of the green square. DISA with Sobel filter and latent compositionality loss obtained from the first training repetition is used in both figures. (**a**) Replace the texture of the green square with the interpolation between its texture slot and the one of the blue s-shaped tetromino. (**b**) Add random noise to the mask slot of the green square.

Multi-dSprites In five runs out of six, DISA develops an interesting and effective strategy allowed by the presence of exclusively plain textures in the Multi-dSprites dataset. In fact, as shown in Figure 23a, the models learn to decode an object texture as a (nearly) constant color, hence ignoring both shape and position information from the second half of its representation. As a consequence, the areas defined by the predicted masks are always fully covered by the color in their corresponding texture reconstructions. Therefore, the shapes of the entities completely rely on the masks, allowing for perfect compositionality



(a) DISA with Sobel filter (rep. 1/3)
Permutation applied to the texture slots: 4-3-2-1.



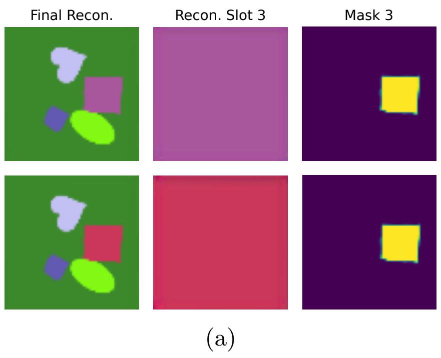
(b) DISA with bottleneck filter (rep. 3/3)
Permutation applied to the texture slots: 2-1-4-3.

Figure 23: Examples showing the compositionality capabilities of DISA models trained on Multi-dSprites. We set the number of slots as the number of entities in the image, i.e. four included the background. The center row contains the input image and the predicted object masks. At the top, the combined reconstruction and the individual textures associated with each slot are shown, while the bottom row presents the reconstructions after permuting the texture slots. (**a**) DISA with Sobel filter resulting from the first training repetition. The two remaining repetitions and the first two runs of DISA with bottleneck filter display equal behavior, thus are not included here. (**b**) DISA with bottleneck filter obtained from the third repetition.

capabilities. For what concerns the one different run of the six, specifically the third repetition of DISA with bottleneck filter, an example of its behavior can be found in Figure 23b. Here, the texture reconstructions present the correct colors only in the zones nearby the objects, leaving more room for errors when switching object textures (e.g. slots 1 and 4 of the figure).

In Figure 24a, differently from Tetrominoes, we try to replace the texture of the pink square in the image with a randomly sampled one. Coherently with the behavior learned by the model, the sampled texture is a simple plain color covering the whole image. When noising the mask slot encoding the shape of the pink square, we obtain a novel shape not contained in the dataset, visualized in Figure 24b. This shape resembles a right triangle with, however, rounded corners and not perfectly defined lines (especially the hypotenuse).

CLEVR6 From Figure 20, we saw that DISA encodes shape information both in the texture and mask slots. Interestingly, qualitative results (Figures 25a and 25b) show that five out of the six models are able to frequently display good compositionality capabilities. Colors and materials are indeed correctly transferred through the texture slots between objects while being shaped, scaled, and positioned accordingly to the masks. However, this is not always the case, as in both Figures 25a and 25b where the texture of the metal-yellow sphere is passed to the former rubber-cyan cube. Even if the object gets decoded with the right material, color, position, and scale, it is clear that the texture is still not shaped to perfectly fit the mask. Additionally, it can be noted that the first repetition of DISA with Sobel filter divides the background across all the slots, contrary



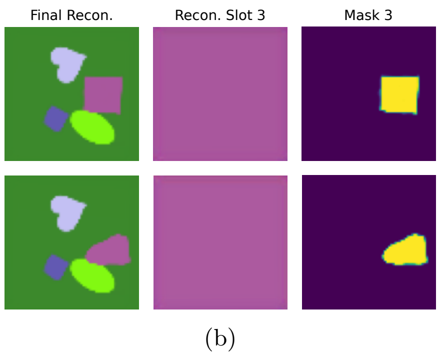
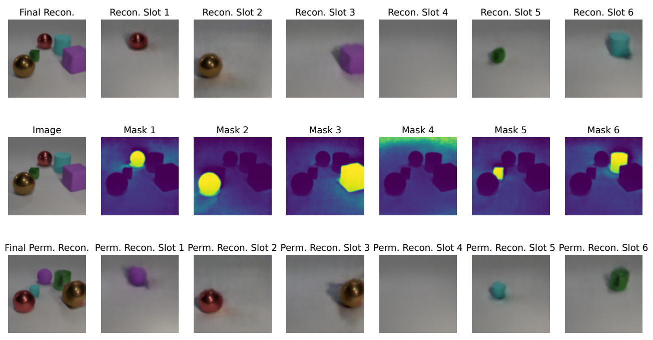


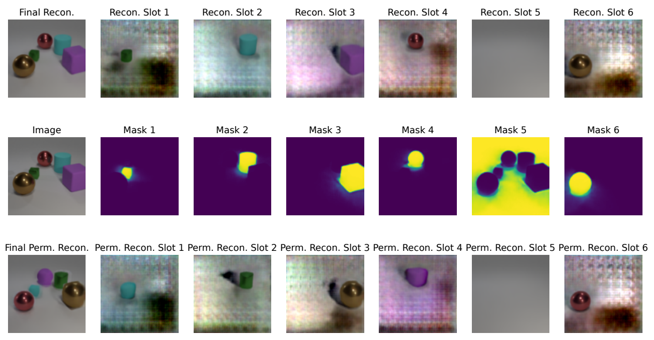
Figure 24: Examples of DISA's capability to generate objects with new textures and shapes on Multi-dSprites. The top row contains the reconstruction of the input image, the predicted texture associated with the pink square, and its predicted mask. At the bottom, the same visualization is presented after altering the texture or shape of the pink square. DISA with Sobel filter obtained from the first training repetition is used in both figures. (**a**) Randomly sample the texture of the pink square. (**b**) Add random noise to the mask slot of the pink square.

to all the remaining runs of our architecture on this dataset. Nevertheless, it is the model producing the most robust compositionality results, rarely suffering from errors such as incorrect scaling of the textures when adapting them to new masks. This type of mistake is visible in the metal-green cylinder and the metal-red sphere (third row of Figure 25b), where the objects are slightly smaller than they are supposed to be. Finally, Figure 25b displays a result of the only model failing to decompose CLEVR scenes. In this case, the predicted masks are constant for all the inputs, leading to incorrect segmentation of the objects and consequent inability to disentangle shape and texture dimensions.

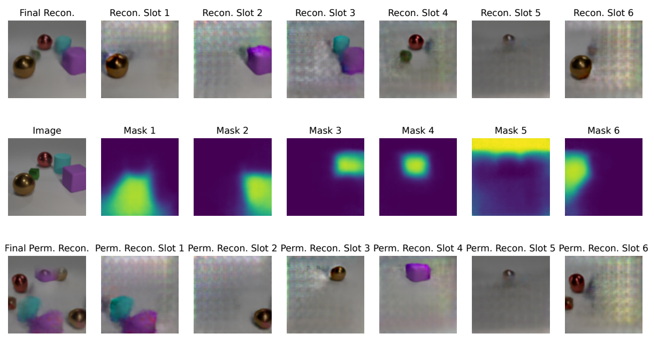
Figure 26 shows the generation of new entities on CLEVR6. Similarly to Tetrominoes, in Figure 26a we replace the texture of the green sphere with the one obtained by interpolating that of the pink sphere and of the gray cylinder. The resulting texture presents a combination of pink and gray colors, while the material is neither clearly defined as metal nor rubber. Finally, we noise the mask slot encoding the shape of the green sphere and display the resulting object in Figure 26b. Here, we can notice again that the decoded texture successfully changes size according to the new shape. However, on this dataset, this happens less frequently than with the first two, and the produced shapes are mostly close to the original ones. This limitation can probably be due to the imperfect disentanglement of shape and texture.



(a) DISA with Sobel filter (rep. 1/3)
Permutation applied to the texture slots: 3-1-2-4-6-5.

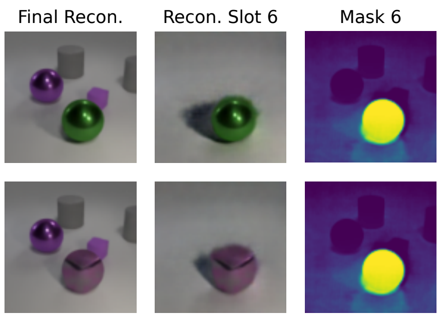


(b) DISA with Sobel filter (rep. 3/3)
Permutation applied to the texture slots: 2-1-6-3-5-4.

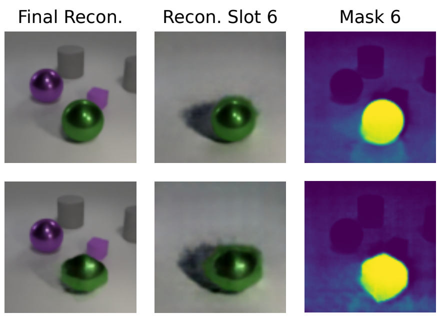


(c) DISA with bottleneck filter (rep. 3/3)
Permutation applied to the texture slots: 3-6-1-2-5-4.

Figure 25: Examples showing the compositionality capabilities of DISA models trained on CLEVR6. We set the number of slots as the number of entities in the image, i.e. six included the background. The center row contains the input image and the predicted object masks. At the top, the combined reconstruction and the individual textures associated with each slot are shown, while the bottom row presents the reconstructions after permuting the texture slots. (**a**) DISA with Sobel filter resulting from the first training repetition. Out of the six runs of both DISA-SB and DISA-BN, this is the only one producing masks where the background is divided across all the slots. (**b**) DISA with Sobel filter obtained from the third repetition. The second repetition and the first two runs of DISA with bottleneck filter exhibit equal behavior, thus are not included here. (**c**) DISA with bottleneck filter from the third training repetition. Out of the six runs of both DISA-SB and DISA-BN, this is the only one constantly producing the same masks, leading to frequently failing at decomposing scenes into objects.



(a)

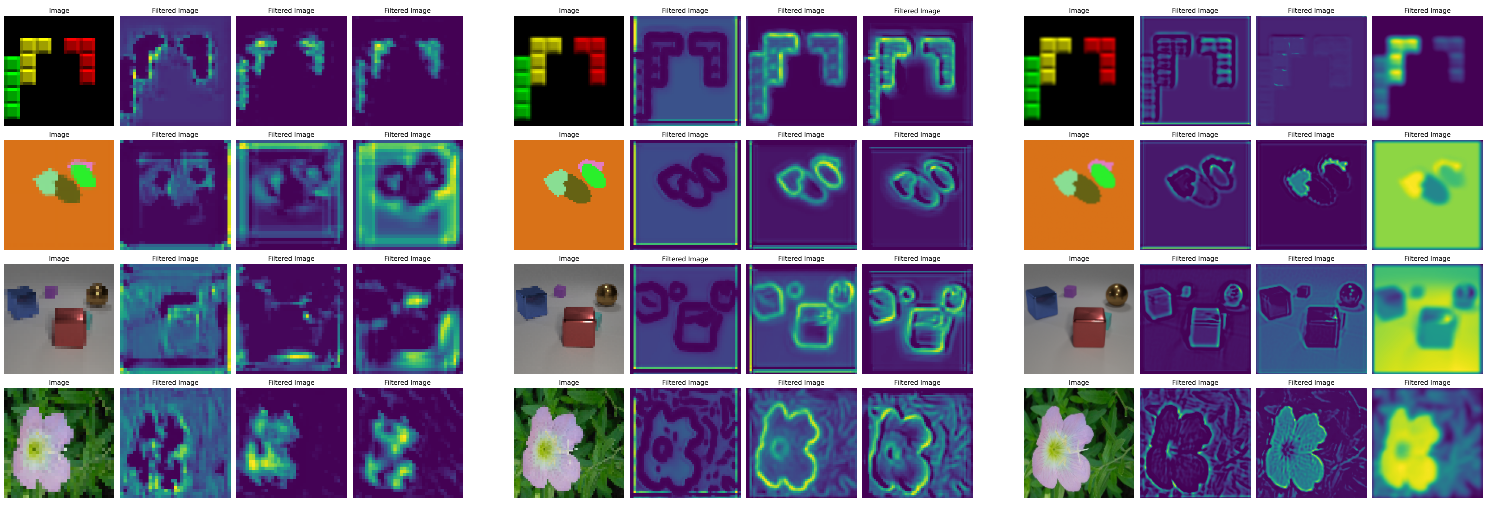


(b)

Figure 26: Examples of DISA's capability to generate objects with new textures and shapes on CLEVR6. The top row contains the reconstruction of the input image, the predicted texture associated with the green sphere, and its predicted mask. At the bottom, the same visualization is presented after altering the texture or shape of the green sphere. DISA with Sobel filter obtained from the first training repetition is used in both figures. (**a**) Replace the texture of the green sphere with the interpolation between the texture slots of the purple sphere and the gray cylinder. (**b**) Add random noise to the mask slot of the green sphere.

6.4 Bottleneck Filter Generalization

A brief qualitative experiment is conducted to study the segmentation capabilities of our bottleneck filter. Specifically, for each of the nine runs of DISA-BN (3 datasets times 3 repetitions), we visualize the output generated by the learned filter on a test image sampled from all the datasets we trained on. An additional image from 102-Flower [44] is considered in order to analyze the behavior of the filters on a real-world picture with a more complex background and edges. Since the datasets are characterized by various resolutions, we resize the images appropriately for each model. In Figure 27, the first columns present the (resized) input images, while the second, third, and fourth columns show the outputs of the filters for, respectively, the first, second, and third repetitions. On Tetrominoes (Figure 27a), a highly precise segmentation is not required to identify an object: it is sufficient for the bottleneck to provide information about position and shape, as the foreground objects are non-overlapping and of fixed size. This is visible in the first row and leads to learning less general filters, as it is clear from the imprecise segmentation in the remaining three rows. A positive note is that the texture-related edges inside the entities can be ignored by the filter. Instead, when DISA is trained on Multi-dSprites (Figure 27b) and CLEVR6 (Figure 27c), the bottleneck filters learn to capture more general borders. This is especially true when trained on CLEVR6. Finally, it is important to highlight that the filtering strategies learned by DISA can largely vary between different runs, even on the same dataset.



(a) (b) (c)

Figure 27: Output of the learned bottleneck filter on test images from Tetrominoes (first row), Multi-dSprites (second row), CLEVR6 (third row), and 102-Flower (fourth row). The first column displays the input image, while the remaining ones present the output from the filters learned by the three repetitions of DISA-BN trained on a Tetrominoes (**a**), Multi-dSprites (**b**), and CLEVR6 (**c**). The behavior of the filter varies across different datasets and repetitions, while consistently showing good generalization capabilities (especially in (**b**) and (**c**)).

7 Discussion and Conclusion

This work introduced a novel architectural design for inducing the disentanglement of texture and shape factors in object-centric models. We evaluated on three well-known synthetic datasets (Tetrominoes, Multi-dSprites, and CLEVR) the architecture resulting from applying our proposed design to Slot Attention, which we referred to as Disentangled Invariant Slot Attention (DISA). Specifically, after training DISA in an unsupervised fashion by reconstructing the input images, the ARI score of the foreground objects and the MSE were measured and compared with those of the original Slot Attention. Furthermore, we investigated whether our models successfully constrained shape and texture information in the latent components we set a priori. We achieved this through the training of two MLPs for each object property (e.g. color), one to predict the given property starting from its related components, and the other from the unrelated ones. Qualitative results showing the capabilities of DISA in transferring and adapting textures across distinct objects were also reported.

Foreground ARI score (Table 1) and reconstruction MSE (Table 2) demonstrated that our method does not negatively impact the performance of SA and, on the contrary, it leads to a slight improvement in sample efficiency. However, the quantitative results concerning the disentanglement were less consistent across the considered datasets and thus more complex to interpret. While Multi-dSprites (Figure 19) showed that our design alone can prevent shape information from being encoded in the texture slots, the success of this outcome is likely dependent on dataset characteristics such as the plain textures of the objects. Conversely, with both Tetrominoes (left and center bars of Figure 18) and CLEVR (Figure 20), DISA could not manage to obtain correct separations by itself. Despite this limitation, qualitative experiments on CLEVR (Figure 25) suggested that the texture decoder tends to ignore the shape information wrongly included in the texture slots, while it focuses instead on the one present in the appropriate part of the representations. With more extensive training, this could potentially lead to the gradual disappearance of shape information from the first half of the latent vectors, as the texture encoder no longer requires it. Regarding the texture information, our proposed design consistently succeeded in preventing it from being encoded within the shape-related components across all the experiments.

When introducing the self-supervised LCL during the training of DISA with Sobel filter on Tetrominoes (right bars of Figure 18), we observed that it is possible to correctly divide the shape and texture information as intended, solving thus the previously mentioned issue and achieving the purpose of this work. In fact, since the model has to learn to accurately combine the mask slot of an entity with the texture slot transferred from a distinct object, it is induced to encode the shape information exclusively in the second half of the representations. Without this behavior, shape information would also be transferred across objects and the compositionality capabilities measured by the LCL would be negatively affected.

Finally, we analyzed the OOD behavior of the bottleneck filter (Figure 27) and found that, when trained on Multi-dSprites and CLEVR, it is able to learn generic segmentation abilities, while however presenting distinct strategies on each run. On Tetrominoes, given the strictly limited variability of the data in terms of textures and shapes, the learned bottleneck struggles to correctly adapt to distribution shifts.

Future Work Extending the proposed architectural design to increase the guarantee of shape information not flowing into texture-related components would be a promising future step. Achieving this goal without relying on auxiliary self-supervised losses, e.g. LCL, could represent a significant improvement over our work, as it would remove both the additional computational cost and tuning brought by the second objective.

A further potential direction could also be the development of new self-supervised losses such as the LCL. These losses should leverage the a priori knowledge about the disentangled latent space with the aim of introducing new properties into the model or biasing it toward some desired direction.

Finally, it is worth exploring the integration of state-of-the-art extensions of Slot Attention to achieve a SOTA explicitly disentangled object-centric model, able to effectively separate texture, shape, position, and scale factors in the currently most effective and efficient way.

References

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

[2] Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Chris Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference, 2020.

[3] Martin Engelcke, Adam R. Kosiorek, Oiwi Parker Jones, and Ingmar Posner. Genesis: Generative scene inference and sampling with object-centric latent representations, 2020.

[4] Martin Engelcke, Oiwi Parker Jones, and Ingmar Posner. Genesis-v2: Inferring unordered object representations without iterative refinement, 2022.

[5] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention, 2020.

[6] Gautam Singh, Fei Deng, and Sungjin Ahn. Illiterate dall-e learns to compose, 2022.

[7] 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, 2022.

[8] Michael Chang, Thomas L. Griffiths, and Sergey Levine. Object representations as fixed points: Training iterative refinement algorithms with implicit differentiation, 2023.

[9] Baoxiong Jia, Yu Liu, and Siyuan Huang. Improving object-centric learning with query optimization, 2023.

[10] Ondrej Biza, Sjoerd van Steenkiste, Mehdi S. M. Sajjadi, Gamaleldin F. Elsayed, Aravindh Mahendran, and Thomas Kipf. Invariant slot attention: Object discovery with slot-centric reference frames, 2023.

[11] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives, 2014.

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

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

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

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby.

An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[16] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers, 2020.

[17] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[18] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[20] Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing, 1968.

[21] 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, 2016.

[22] Adam R. Kosiorek, Hyunjik Kim, Ingmar Posner, and Yee Whye Teh. Sequential attend, infer, repeat: Generative modelling of moving objects, 2018.

[23] Karl Stelzner, Robert Peharz, and Kristian Kersting. Faster attend-infer-repeat with tractable probabilistic models, 2019.

[24] Eric Crawford and Joelle Pineau. Spatially invariant unsupervised object detection with convolutional neural networks, 2019.

[25] Jindong Jiang, Sepehr Janghorbani, Gerard de Melo, and Sungjin Ahn. Scalor: Generative world models with scalable object representations, 2020.

[26] 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.

[27] Klaus Greff, Antti Rasmus, Mathias Berglund, Tele Hotloo Hao, Jürgen Schmidhuber, and Harri Valpola. Tagger: Deep unsupervised perceptual grouping, 2016.

[28] Charlie Nash, S. M. Ali Eslami, Christopher P. Burgess, Irina Higgins, Daniel Zoran, Théophane Weber, and Peter W. Battaglia. The multi-entity variational autoencoder, 2018.

[29] Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Neural expectation maximization, 2017.

[30] Sjoerd van Steenkiste, Michael Chang, Klaus Greff, and Jürgen Schmidhuber. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions, 2018.

[31] 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, 2017.

[32] Titas Anciukevicius, Christoph H. Lampert, and Paul Henderson. Object-centric image generation with factored depths, locations, and appearances, 2020.

[33] Amin Mansouri, Jason Hartford, Kartik Ahuja, and Yoshua Bengio. Object-centric causal representation learning, 2022.

[34] Zhixin Shu, Mihir Sahasrabudhe, Alp Guler, Dimitris Samaras, Nikos Paragios, and Iasonas Kokkinos. Deforming autoencoders: Unsupervised disentangling of shape and appearance, 2018.

[35] Dominik Lorenz, Leonard Bereska, Timo Milbich, and Björn Ommer. Unsupervised part-based disentangling of object shape and appearance, 2019.

[36] Han Yan, Haijun Zhang, Jianyang Shi, Jianghong Ma, and Xiaofei Xu. Toward intelligent fashion design: A texture and shape disentangled generative adversarial network, 2023.

[37] William M Rand. Objective criteria for the evaluation of clustering methods, 1971.

[38] Lawrence Hubert and Phipps Arabie. Comparing partitions, 1985.

[39] 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.

[40] Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

[41] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[42] Andrea Dittadi, Samuele Papa, Michele De Vita, Bernhard Schölkopf, Ole Winther, and Francesco Locatello. Generalization and robustness implications in object-centric learning, 2022.

[43] Samuele Papa, Ole Winther, and Andrea Dittadi. Inductive biases for object-centric representations in the presence of complex textures, 2022.

[44] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes, 2008.

A Additional Results

A.1 Robustness Under Color and Texture Variations

	Plain	Plain-BG	Stripes	Stripes-BG	Color	Color-BG
SA	**24.3** ± 9.0	73.7 ± 5.9	31.3 ± 9.1	72.7 ± 8.3	1.0 ± 0.5	70.2 ± 7.5
SA-NOISE	32.5 ± 5.7	**58.2** ± 7.0	29.9 ± 4.3	**56.2** ± 10.8	0.1 ± 0.7	58.0 ± 5.1
DISA-SB	76.1 ± 11.7	79.2 ± 10.4	33.0 ± 14.4	65.0 ± 5.4	**0.0** ± 0.6	6.2 ± 2.1
DISA-BN	**24.3** ± 7.2	62.6 ± 6.5	**18.2** ± 8.2	67.1 ± 8.0	0.1 ± 0.4	65.5 ± 6.9
DISA-SB-LCL	63.8 ± 11.6	68.5 ± 10.9	24.4 ± 2.7	82.9 ± 1.9	**0.0** ± 0.2	**4.7** ± 0.9

Table 3: Decrease in the Adjusted Rand Index (ARI) score (↓) under texture variations on Tetrominoes. Each row reports the distances between the corresponding mean ARI score on the original test set (Table 1) and the means on six variations of it. As in [1–3, 5], the results exclude the background and only consider the foreground objects. SA-NOISE and DISA models augment the input images with a random plain noise during training, while SA does not.