Object-Centric Semantic Vector Quantization

Yi-Fu Wu[*]
Rutgers University

Minseung Lee
KAIST

Sungjin Ahn
KAIST

Editors: Marco Fumero, Emanuele Rodolà, Clementine Domine, Francesco Locatello, Gintare Karolina Dziugaite, Mathilde Caron

Abstract

Neural discrete representations are crucial components of modern neural networks. However, their main limitation is that the primary strategies such as VQ-VAE can only provide representations at the patch level. Therefore, one of the main goals of representation learning, acquiring conceptual, semantic, and compositional abstractions such as the color and shape of an object, remains elusive. In this paper, we present the first approach to *semantic* neural discrete representation learning. The proposed model, called **S**emantic **V**ector-**Q**uantized Variational Autoencoder (SVQ), leverages recent advances in unsupervised object-centric learning to address this limitation. Specifically, we observe that a simple approach quantizing at the object level poses a significant challenge and propose constructing scene representations hierarchically, from low-level discrete concept schemas to object representations. Additionally, we suggest a novel method for training a prior over these semantic representations, enabling the ability to generate images following the underlying data distribution, which is lacking in most object-centric models. In experiments on various 2D and 3D object-centric datasets, we find that our model achieves superior generation performance compared to non-semantic vector quantization methods such as VQ-VAE and previous object-centric generative models. Furthermore, we find that the semantic discrete representations can solve downstream scene understanding tasks that require reasoning about the properties of different objects in the scene.

1 Introduction

While there have been various findings regarding the purpose of the brain, it is fair to say that the human brain has at least two key functions. First, it constructs a good representation that captures the structure of the world through perception. Second, it imagines or generates various possibilities of the world. Similarly, AI systems that aim to be as generally capable as humans would also need to realize similar capabilities computationally. Building such a learning system that can both structurally *recognize* and *generate* has long been a desired vision in machine learning, from Helmholtz machines [1, 2] to Variational Autoencoders [3, 4]. Although there could be various approaches to achieving this, in this work, we focus on a specific class of models, which we call Generative Structured Representation Models, which satisfy the following desiderata.

First, when it comes to representating a visual scene, it appears that we do not perceive the scene simply as a monolithic vector of features. Instead, we view it structurally and semantically, recognizing it as a composition of meaningful components such as objects and their attributes like shape, color, and position [5, 6, 7]. Various works in AI, particularly object-centric approaches [8], have

[*]Correspondence to yifu.wu@gmail.com.

Proceedings of the I edition of the Workshop on Unifying Representations in Neural Models (UniReps 2023).

Table 1: Desiderata for Generative Structured Representation Models and Related Models

	VAE	VQ-VAE	Slot Attention	SysBinder	SVQ (Ours)
Semantic Decomposition	Factor	✘	Object	Object & Factor	Object & Factor
Discrete	✘	✓	✘	✘	✓
Sampling	✓	✓	✘	✘	✓

demonstrated that this structural decomposition facilitates relational reasoning [9, 10, 11, 12] and out-of-distribution generalization [13, 10] due to improved compositional generalization. It has also been shown that a monolithic vector representation of a scene, such as VAE, fails in multi-object scenes [9, 13, 10].

Moreover, this structured and semantic understanding can be categorized and conceptualized *discretely* in an unsupervised way. Such an ability is critical for organizing and comprehending the complexity of the environment, e.g., via language, as well as for implementing modularity [14] or symbolic reasoning [15]. In AI, discrete representations are also useful to leverage powerful learning models like transformers. One of the most popular models for discrete representation learning in AI is VQ-VAE [16]. It has been shown to be beneficial for image generation [17, 18] and probability density modeling [19]. However, VQ-VAE and its variants, such as dVAE [20, 21] and VQ-GAN [18], represent a scene as a grid of small patches, lacking the capability to capture the scene's holistic structure and semantics.

Besides, the ability to generate samples that adhere to the observed data distribution is foundational for endowing AI the capabilities to imagine and simulate, e.g., for planning [22, 23]. However, only a certain class of representation learning models supports this essential ability. While models like Slot Attention [24] and SysBinder [25] offer structured, object-centric representations, in its original form it is unclear how to support density-based sampling. In contrast, VAE-based models, such as VAE and VQ-VAE, generally support this ability to sample from a prior distribution, but they either do not provide object-centric structures (VAE) or are limited to patch-based representations (VQ-VAE).

In this work, we observe that, despite its significance, there is currently no method that simultaneously satisfies all of the mentioned criteria of Generative Structured Representation Models, as summarized in Table 1. To address this issue, we propose the **S**emantic **V**ector-**Q**uantized (SVQ) Variational Autoencoder. Our model achieves discrete semantic decomposition by learning hierarchical, composable factors that closely align with the objects and the properties of objects in visual scenes. Similar to patch-based vector quantization methods, we can train an autoregressive prior to learn the distribution of the dataset. However, unlike VQ-VAE, we achieve this by learning the distribution of semantic discrete tokens, rather than patch tokens. As a result, the generation (or imagination) process is to compose semantic concepts such as objects and their attributes, rather than stitching a grid of patches.

In our experiments, we demonstrate the following practical benefits of our method. First, we find that for multi-object scenes, SVQ is able to model the prior distribution better than patch-based methods, as measured by the quality of the samples generated. Second, we find that SVQ representations outperform patch-based VQ representations on downstream tasks that require knowledge of the different properties of the objects in the scene. We also find evidence that SVQ representations can generalize better to out-of-distribution tasks compared to patch-based VQ representations and SysBinder continuous representations. Lastly, we show that despite introducing a discrete bottleneck, SVQ can work on the challenging CLEVRTex [26] dataset, one of the most complex datasets used in recent unsupervised object-centric representation learning models.

Our contributions are as follows: First, we introduce SVQ, the first model to obtain *semantic* neural discrete representations without any supervision about the underlying factors in the scene. Second, by training a prior over these discrete representations, we are able to obtain an object-centric density model, capable of capturing the underlying data distribution and generating new samples. Third, we evaluate our model on several 2D and 3D datasets including the challenging CLEVRTex dataset, showing superior downstream task performance and image generation quality.

2 Background

2.1 Vector-Quantized Variational Autoencoder (VQ-VAE)

The VQ-VAE [16] is a model that learns to compress high-dimensional data into a discretized latent space. The latent space is maintained by a codebook of prototype vectors $\mathbf{e} \in \mathbb{R}^{K \times d}$ where K is the size of the codebook and d is the dimensionality of each prototype vector. An input $\mathbf{x}$ is first passed through encoder $E(\mathbf{x})$ to obtain latents $\mathbf{z}_e \in \mathbb{R}^d$. A nearest-neighbor lookup between $\mathbf{z}_e$ and each of the prototype vectors in the codebook yields a quantized representation $\mathbf{z}_q = \mathrm{Quantize}(\mathbf{z}_e) = \mathbf{e}_k$ where $k = \arg\min_j ||\mathbf{z}_e - \mathbf{e}_j||_2$. The decoder D then uses $\mathbf{z}_q$ to reconstruct the input: $\hat{\mathbf{x}} = \mathrm{D}(\mathbf{z}_q)$. The model is trained with the following loss:

$$\mathcal{L} = \underbrace{||\mathbf{x} - \hat{\mathbf{x}}||_2^2}_{\text{Reconstruction}} + \underbrace{||sg[\mathbf{z}_e] - \mathbf{z}_q||_2^2}_{\text{Codebook}} + \beta \underbrace{||\mathbf{z}_e - sg[\mathbf{z}_q]||_2^2}_{\text{Commitment}} .$$

The first term is a reconstruction loss and is used to train the encoder and decoder. A straight-through estimator [27] is used to estimate the gradients through the quantization step by copying the gradients from $\mathbf{z}_q$ to $\mathbf{z}_e$. The second term is the codebook loss which encourages the prototype vectors in the codebook to be close to the output of the encoder. The third term, scaled by a constant hyperparameter β, is the commitment loss and helps to stabilize the training by encouraging the output of the encoder to not deviate too much from the chosen prototype vectors. Instead of the codebook loss, we use exponential moving average (EMA) updates on the codebook, which we found to speed up training in our experiments [17, 28, 29].

When VQ-VAEs are applied to images $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$, the encoder $E(\mathbf{x})$ is typically implemented as a convolution encoder, outputting a feature map of latents $\mathbf{z}_e \in \mathbb{R}^{H_z \times W_z \times d}$. This means that each latent corresponds to a local area represented by a convolutional feature cell and thus can only capture information in a local receptive field (Figure 1a). However, images typically contain multiple objects, and the discrete factors underlying visual scenes typically correspond to different properties of the objects in the scene, such as shape, color, type, and so on. The local patches from convolutional feature maps are inadequate to capture this rich structure.

2.2 Object-Centric Representations

The goal of unsupervised object-centric representation learning is to decompose a scene into a set of representations each capturing a different object in the scene. It is shown that this structural decomposition, matching to the true factor structure of the world, facilitates some high-level cognition abilities such as relational reasoning [9, 10, 11, 12] and out-of-distribution generalization [13, 10]. We build on top of Slot Attention [24], a spatial attention-based object-centric representation method.

Given an image $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$, slot attention learns a set of slots, $\mathbf{s} = \{\mathbf{s}_1, \ldots, \mathbf{s}_N\}$, where $\mathbf{s}_n \in \mathbb{R}^{d_s}$ and N is the total number of slots. An encoder is applied to $\mathbf{x}$ and, after adding a positional encoding, the result is flattened to an L-length input feature vector $\mathbf{F} \in \mathbb{R}^{L \times d_F}$. Then, an iterative attention mechanism is used to spatially group the input features $\mathbf{F}$ to the slot representations $\mathbf{s}$. First, the slots are randomly initialized from a Gaussian distribution with learned parameters. Then, in each iteration, the slots are used as queries in an *inverted* version of dot-product attention [30] with the input features $\mathbf{F}$ as the keys and values. Instead of normalizing over the keys as is done in traditional dot-product attention, normalization is done over the queries (ie. slots). Additionally, a weighted mean is used to aggregate the values instead of the normal weighted sum, which is shown to stabilize training. The result is then used to update the slots with a per-slot GRU [31] followed by a per-slot residual MLP, both with shared parameters across the slots.

The slot representations are then used in a decoder to reconstruct the image and the entire model is trained with an image reconstruction loss. The original formulation of slot attention used a spatial broadcast decoder [32] to create masked images per slot which are then combined to form a final reconstructed image. Recently, [21] proposed using a transformer decoder to reconstruct the image while attending to the slots with cross attention. This method was shown to scale to more complex scenes than the spatial broadcast decoder [33] and is what we choose to use in our model.

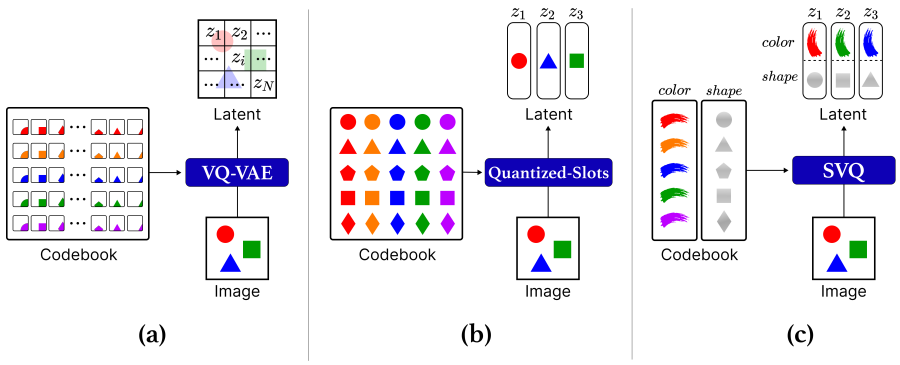

Figure 1: Comparison between VQ-VAE, Quantized Slots, and SVQ. (a) VQ-VAE quantizes the scene at a local patch level and may not capture the semantic structure of the scene. (b) Quantized Slots (QS) would quantize the scene at the slot level but require a separate code for every possible configuration of an object. (c) SVQ quantizes at the block level, representing each factor (such as color or shape) as a code. In this example, to represent all possible object configurations, SVQ requires only 10 codebook entries at the block level while QS requires 25.

3 Semantic Vector Quantization

Given a slot attention encoder that can obtain a set of representations of the objects in a scene, one may think of a hypothetical method, applying vector quantization to the slot representation itself to obtain a set of semantic discrete representations (Figure 1b). While these representations would indeed correspond to the different objects in a scene, this scheme would essentially require one codebook entry per possible object configuration and would be insufficient for anything beyond trivially simple scenes.

For example, consider a simple scene containing a single object in a fixed position that only varies by color and shape. Assume there are c possible colors and s possible shapes for the object. With slot-level quantization, in order to represent all the potential objects, the codebook would require at least $c \times s$ entries. This is because each slot representation is a single entangled representation so each combination of factors needs to be represented by a separate code. If instead, we were able to disentangle the object-level representations into factor-level representations—representations that align with the underlying latent factors of variation of each object—we would be able to describe the potentially large combinatorial space of each object with a much small number of discrete factors. In the above example, if we had a fully disentangled representation of the color and the shape, we would be able to represent all possible scenes with $c + s$ codes (Figure 1c). See Appendix A.2 for further discussion.

This observation motivates us to design an architecture that further disentangles slot representations to factor representations that reflect the underlying discrete factors of the objects in the scene, and to perform vector quantization on these factor representations. Under this scheme, each object representation would be composed of multiple discrete factors, and each factor would have its own codebook that can be shared across objects. The resulting model, the **S**emantic **V**ector-**Q**uantized Variational Autoencoder (SVQ), is depicted in Figure 2a and described below.

To obtain factored representations, we follow an approach motivated by Neural Systematic Binder (SysBinder) [25], where a binding mechanism is introduced to produce disentangled factors within a slot. Specifically, the following modifications are applied to slot attention: First, we maintain M codebooks $\mathbf{C} \in \mathbb{R}^{M \times K \times d_c}$, each with K discrete prototype vectors of dimension $d_c = \frac{d_s}{M}$. Then, we split each of the N d_s-dimensional slot representations into M equally-sized blocks, each of which will represent one factor. We denote the full set of block representations as $\mathbf{z}_e \in \mathbb{R}^{N \times M \times d_c}$. Crucially, we replace the slot-level GRUs and residual MLPs with block-level equivalents that have shared parameters across blocks corresponding to the same factor. At the end of each slot attention iteration, we apply vector quantization for each block using its corresponding codebook to obtain a set of quantized blocks $\mathbf{z}_q \in \mathbb{R}^{N \times M \times d_c}$. For $n \in [1, N], m \in [1, M]$,

$$\mathbf{z}_q^{n,m} = \mathbf{C}_{m,k} \text{ where } k = \arg\min_j ||\mathbf{z}_e^{n,m} - \mathbf{C}_{m,j}||_2 \ ,$$

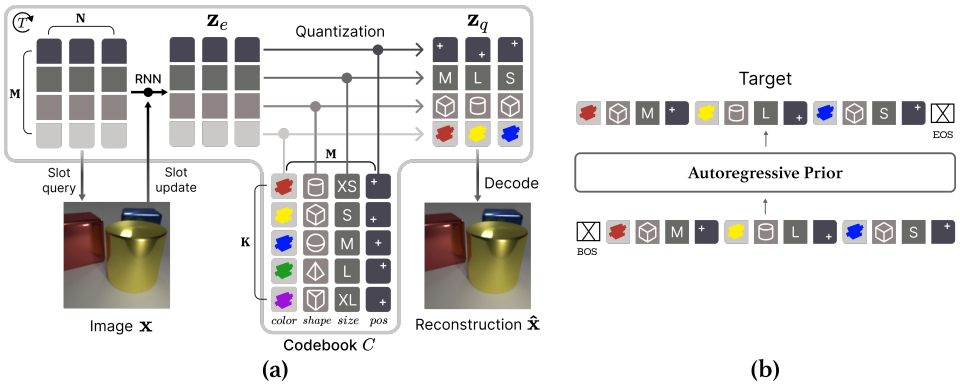

Figure 2: (a) Overall architecture of SVQ. We maintain M learned codebooks and split each slot into M blocks. At the end of each Slot Attention iteration, we apply vector quantization to each block representation to obtain a set of discrete codes for each slot. Each block ends up specializing in different underlying factors of variation for the objects in the scene. (b) The Semantic Prior. After training the model, we freeze SVQ and train and autoregressive prior over the discrete latent codes. Sampling from this prior allows us to generate an image one object at a time, based on their properties.

where $\mathbf{z}_q^{n,m}$ denotes the m-th block in the n-th slot and $\mathbf{C}_{m,k}$ is the k-th prototype vector in the m-th codebook. By sharing the codebook for each block across all of the slots, each block ends up specializing in different underlying factors of the objects in the scene, such as color, shape, and position. Thus, these quantized representations are semantic in the sense that they contain factor-level representations mapping to the underlying structure of the scene.

To reconstruct the image, we use the autoregressive transformer decoder described in Section 2.2 and condition on $\mathbf{z}_q$ via cross attention. Similar to Singh et al. [25], we first let the blocks within a slot interact with a single-layer transformer and then add a block-level positional encoding before inputting the representations as cross attention in the transformer decoder. We train the model with the reconstruction loss, the VQ-VAE commitment loss, and we update the codebooks with EMA updates. To prevent codebook collapse, we also incorporate random restarts for the embeddings, similar to previous work [28]. To achieve this, we keep a count of the usage of each code in the codebooks and randomly reset it to be near one of the encoder outputs of the current batch if its usage falls below a threshold.

3.1 Semantic Prior

Given these semantic discrete codes representing the different objects in the scene, we can now freeze the SVQ and train an autoregressive prior $p(\mathbf{z}_q)$ over these codes to model the structure of the data (Figure 2b). We can then sample from this prior to obtain codes for new scenes and use these codes in the SVQ decoder to generate new images. Compared to patch-based VQ methods that generate tokens that correspond to a spatially local region of an image, this *semantic* prior generates an image one object at a time, based on their properties.

We implement the prior using a simple autoregressive transformer decoder. First, we flatten $\mathbf{z}_q$ along the slot and block dimensions to a vector with dimensions $NM \times d_c$. We then apply a positional encoding across all slots and blocks and input the resulting vector to a transformer decoder with an objective of predicting the discrete code of the next block. Although slot attention does not guarantee any specific ordering of the slots, the blocks within the slots are arranged in a predefined order. Therefore, the positional encoding is important in providing information about the ordering of the blocks as well as which block belongs to which slot.

Note that generating the latents of one image requires sampling NM blocks, but does not depend on the size of the image. This is different than VQ-VAE, which scales with the size of the feature map and may become expensive for high-resolution images.

4 Related Work

Neural Discrete Representation Learning. Our work builds on top of neural discrete representation learning, which has played a pivotal role in the advancement of generative models for images in recent years [16, 17, 20, 18, 34]. These methods typically follow a two-stage approach. First, an image is encoded into a CNN feature map, which is then tokenized using vector quantization [35] into a set of discrete latent variables. In the second stage, a powerful autoregressive prior is then trained to model the distribution of these discrete tokens, allowing for sampling new images from this distribution. Our model also follows this two-stage approach, except our latents correspond to the properties of objects instead of cells in a CNN feature map.

Unsupervised Object-Centric Learning. Recent unsupervised object-centric learning methods have been shown to decompose an image or video into a set of latents, each representing an object in the scene [36, 37, 38, 24, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 47, 51, 25, 52, 33, 53, 54, 55, 56, 9, 57, 58]. While most of these methods result in a distributed representation per object, there have been several attempts at learning more structured or disentangled representations, such as those methods that decompose the latents into what and where components [46, 59, 60, 61, 48, 47, 51, 49] or those that learn disentangled latents via a VAE [37, 58]. Closely related to our work, recent methods have been designed to learn factor-level disentanglement [25, 62]. However, these methods still operate with continuous latents instead of discrete tokens and do not support sampling new images. While there are several object-centric learning methods that do support sampling new images [40, 41, 48, 63], these also do not use semantic discrete latents as we do in our work.

5 Experiments

Datasets. We evaluate our model on two variants of a 2D Sprites dataset [64, 10] and three variants of the CLEVR dataset [65], CLEVR-Easy, CLEVR-Hard, CLEVR-Tex. In the 2D Sprites datasets, objects of varying shapes and colors are placed in a scene. In total, there are 7 possible colors and 12 possible shapes. In each image, one object has a single property that is unique from the other objects. All other properties are shared by at least two objects. This structure allows us to evaluate if the prior correctly models the dependencies between the properties of the scene. We test versions of this dataset with and without textured backgrounds [66]. CLEVR-Easy, CLEVR-Hard, and CLEVR-Tex were previously used in [25] and are modified from the original CLEVR [65] and CLEVR-Tex [26] datasets to have larger objects so properties such as shape and texture are more clearly visible. In CLEVR-Easy, objects may differ by only shape, color, and position. In this dataset, there are 3 possible shapes and 8 possible colors. In CLEVR-Hard, objects may differ by shape, color, position, size, and material. There are 3 possible shapes, 137 possible colors, and 2 possible materials (shiny or matte). In CLEVR-Tex, there are 4 possible shapes and 58 possible textures for the objects and background.

Baselines. We compare our model with several patch-based quantization methods: VQ-VAE [16] with a PixelCNN [19] prior, and dVAE [20, 21] with a transformer decoder prior. For the dVAE baseline, we use the dVAE weights that are trained along with the SVQ. This provides a more direct ablation comparing the semantic prior of SVQ with the patch-based transformer decoder prior since the dVAE decoder is shared across these models and will not contribute to differences in image quality. We also compare with GENESIS-v2 [41], a continuous latent object-centric model with an autoregressive prior that can also generate samples.

5.1 Generating Samples with Semantic Prior

5.1.1 2D Sprites

We show the sample generations for the 2D Sprites datasets in Figure 3 and the FID results in Table 2. We additionally calculate generation accuracy by manually inspecting 128 images per model to check if the generated images follow the constraints of the dataset. That is, each image must have exactly one object that has a unique property. All other properties in the scene will have at least one duplicate among the other objects.

We see that for the simplest dataset with 3 objects and no background, SVQ achieves the lowest FID of the models and comparable generation accuracy to dVAE, generating about 75% of the scenes

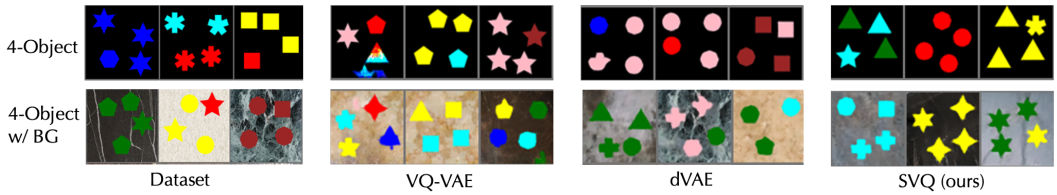

Figure 3: Generated samples for the 4-object 2D Sprites and 4-object 2D Sprites with background datasets.

correctly. This setting may be simple enough that dVAE with a transformer prior can capture the structure of the scene even with a patch-based discrete latent. As the scene complexity increases with more objects and textured background, SVQ starts to outperform the baselines in terms of generation accuracy. Inspecting the qualitative results, we see that in the dataset with the background, VQ-VAE and dVAE start generating occasional blurry objects, whereas SVQ maintains clean-looking objects that match the ground truth dataset. This may be because SVQ can segment the background into its own slot and factor the texture into a discrete latent, cleanly separating the representation of the objects from the background. The patch-based methods, however, may have a harder time separating the foreground from the background resulting in messier generations. Interestingly, despite the blurry shapes, VQ-VAE achieves the lowest FID score on the 2D Sprites dataset with background. We hypothesize this may be because the model spends more capacity modeling the background correctly instead of the foreground, which may produce a better FID score, but not necessarily better generation accuracy. This is confirmed by the low generation accuracy of the VQ-VAE model this dataset, only generating 19.5% of the scenes correctly.

Table 2: FID and Generation Accuracy on the 2D Sprites datasets. For Generation Accuracy, 128 samples were inspected manually to determine if they matched the constraints of the scene (ie. exactly one unique property among all the shapes). Underlined numbers indicate a minor difference from the best value.

	FID ↓			Generation Accuracy (in %) ↑		
Dataset	VQ-VAE	dVAE	SVQ (ours)	VQ-VAE	dVAE	SVQ (ours)
2D Sprites (3 obj)	14.81	7.26	**6.61**	28.91	**75.78**	75.00
2D Sprites (4 obj)	26.35	19.15	**17.93**	21.88	62.50	**66.41**
2D Sprites w/ bg (4 obj)	**58.14**	66.08	58.50	19.53	30.47	**42.19**

5.1.2 CLEVR

In Figure 4, we show sample generations after training the models on the CLEVR-Easy, CLEVR-Hard, and CLEVR-Tex datasets. We report the Frechet Inception Distance (FID) in Table 3. We find that compared to the other models, GENESIS-v2 generates very blurry images and completely fails on CLEVR-Tex, resulting in a high FID. While VQ-VAE produces sharper images, several of the generated shapes are malformed or have mixed colors. The dVAE-generated images look closer to the ground truth dataset, but still have some errors such as overlapping objects (first image) and generating scenes with more objects than seen in the training set (third image). SVQ has the lowest FID for all of these datasets and the generated images look very close to the ground truth dataset, indicating the usefulness of having a semantic prior for generating these multi-object scenes.

In Appendix A.1, we show additional analysis of the learned codebook on the CLEVR-Easy dataset.

Table 3: FID for the various models on the CLEVR datasets.

	FID ↓			
Dataset	GENESIS-v2	VQ-VAE	dVAE	SVQ (ours)
CLEVR-Easy	115.56	57.06	40.30	**32.50**
CLEVR-Hard	93.01	73.33	65.89	**43.12**
CLEVR-Tex	225.08	178.59	112.80	**84.52**

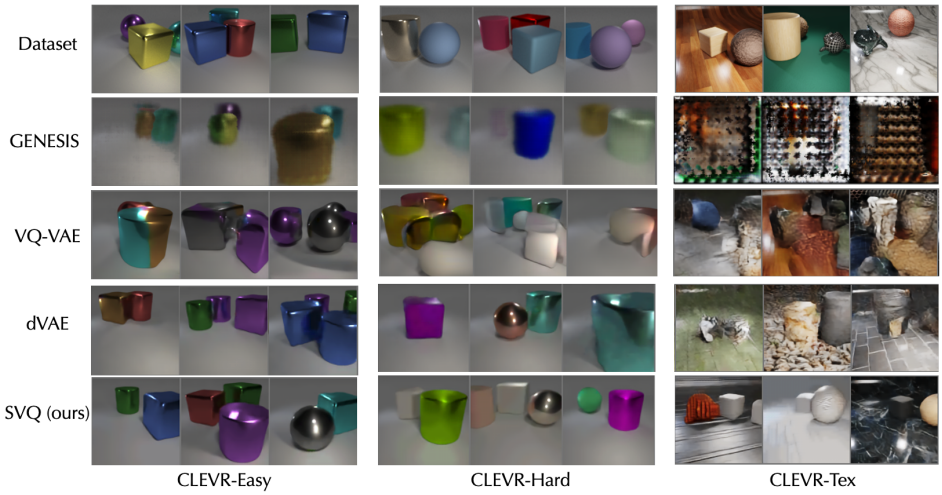

Figure 4: Generated samples for the CLEVR-Easy, CLEVR-Hard, and CLEVR-Tex Datasets.

Table 4: Results for the downstream odd-one-out task. Since all the models can solve the in-distribution (ID) task, we report the number of steps to 98% ID Accuracy and out-of-distribution (OOD) accuracy.

	Steps to 98% (↓)	OOD Acc. % (↑)
dVAE	37,000	26.7
VQ-VAE Indices	77,000	24.0
SysBinder	**27,000**	67.6
SVQ Indices	77,000	46.8
SVQ Codebook	**27,000**	**99.1**

Table 5: Results for the downstream CLEVR-Hard Property Comparison task.

	ID Acc. % (↑)	OOD Acc. % (↑)
dVAE	27.52	19.87
VQ-VAE Indices	24.53	17.74
VQ-VAE Codebook	23.73	18.80
SysBinder	**79.60**	70.09
SVQ Indices	68.21	64.53
SVQ Codebook	75.86	**71.15**

5.2 Downstream Tasks

5.2.1 Odd-One-Out

We first evaluate on a downstream supervised learning task on the 2D Sprites dataset. We modify the dataset by first dividing each image into four quadrants and ensuring exactly one object will be in each quadrant. As in our previous experiments, one object has a single property that is unique from the other objects. The goal of the task is to identify the quadrant of the odd-one-out object. We first pretrain the baseline models on a dataset containing all 12 possible shapes and 7 possible colors. Then, we freeze the underlying model and train a downstream model on top of the learned representations with the supervised objective. The downstream model is trained on a dataset that only contains 9 possible shapes and 4 possible colors. We then evaluate on both the in-distribution (ID) dataset and an out-of-distribution (OOD) dataset that consists of the remaining 3 shapes and 3 colors. In addition to dVAE and VQ-VAE, we use SysBinder as a baseline for this task, to compare its continuous representation with SVQ's discrete representation. For the latent representation of SVQ, we include variants that use the codebook indices (SVQ Indices) and the codebook prototype vectors (SVQ Codebook).

Table 4 shows the results of our experiments. Since all models can solve the task when evaluated on the ID dataset, we report the number of steps to reach 98% accuracy on the validation dataset. We find that SysBinder and SVQ Codebook learn the quickest in the ID setting. For the OOD setting, we find that dVAE and VQ-VAE fail completely, not performing better than randomly guessing, showing that the patch-based discrete latent is insufficient for OOD generalization in this task. SysBinder can partially solve the task in the OOD setting, while the SVQ Codebook seems to be able to solve the task, achieving 99% accuracy. This indicates that the compact latent space offered by the discrete code provides better out-of-distribution generalization abilities for this particular task. One possible explanation for this is that since this is an odd-one-out task, the downstream network needs to do

comparisons between the properties of the objects and this may be easier to do with SVQ's codebook vectors that are fixed. SysBinder's continuous latents, on the other hand, offer greater variations for the same concept. This increases the potential for the downstream network to learn spurious correlations in the data, which can negatively impact OOD performance. SVQ Indices is also only able to partially solve the task. This makes sense because in the out-of-distribution case, the model does not have any way of knowing two codebook indices are for the same property value (e.g. if two codebook vectors both correspond to the color blue). Since SVQ Codebook uses the prototype vectors, it does not have this problem because the similarity can be determined by the vector representation.

5.2.2 CLEVR-Hard Property Comparison

For CLEVR-Hard, we construct a downstream task that assigns a number to each image as follows: First, we assign a number for each possible shape, color, and material present in the dataset. Then, for a given image, we identify the maximum number for each of these three properties. Lastly, we sum the max numbers for each of the properties to arrive at one integer label per image. We formulate the problem as a classification problem to correctly identify the number for each image. For example, suppose we have a scene containing a matte red cylinder and a shiny blue sphere. Assume we assign the following numbers to the different property values: matte $= 0$, shiny $= 1$, red $= 5$, blue $= 3$, cylinder $= 4$, sphere $= 6$. Thus the two objects are represented by the numbers $(0, 5, 4)$ and $(1, 3, 6)$. The max numbers for each of the properties is $(1, 5, 6)$ and the final integer label is $1 + 5 + 6 = 12$. Solving this task requires understanding the property values of each object in the scene.

We train the underlying models on the entire dataset consisting of all the possible property values. Then we randomly select 50 objects for an OOD dataset. Since our task relies on knowing the numerical value of each property, the ID dataset we train on may still contain property values of objects in the out-of-distribution dataset, but it will not contain objects where the combination of property values is present in the OOD dataset. Thus, when evaluating on the OOD dataset, we are testing the model on novel *combinations* of property values, even if those property values were individually observed during training. We show the ID and OOD results in Table 5. We see that SVQ outperforms the patch-based methods and performs comparably to SysBinder in both ID and OOD settings. This shows that despite adding a discretization bottleneck, the latents in SVQ are still useful for downstream tasks that rely on the properties of the objects in the scene.

6 Conclusion and Limitations

In this work, we introduced the Semantic Vector-Quantized Variational Autoencoder. Unlike traditional vector quantization methods, our model can obtain semantic neural discrete representations, capturing the rich structure of the objects in a scene. We showed that by training a prior over these semantic discrete tokens, we are able to generate multi-object scenes that follow the underlying data distribution. These semantic discrete representations are also useful for downstream tasks, outperforming the representations from patch-based discretation methods. While our model is only evaluated on static images, an interesting future direction would be to apply our method to videos to predict future frames. This may be fruitful for modeling longer video sequences since SVQ can compress each frame into a set of latents that only depend on the number of objects in the scene.

Limitations. While our method can learn semantic discrete representations and is capable of using these representations to generate images of higher visual fidelity than previous object-centric methods such as GENESIS [40, 41], it is still only shown to work well on synthetic datasets with similar visual complexity as previous work [25]. Although scaling unsupervised object-centric models to more realistic datasets is not a focus of this work, further improving our model so that it can work well on more realistic scenes is an important avenue of future research. Another limitation of our model is that our latent representations are *all* discrete. Although our visual world does consist of many discrete concepts, factors such as position and pose are continuous. It would be interesting to explore ways to combine continuous and discrete factors to better model realistic scenes.

Acknowledgments and Disclosure of Funding

We thank Gautam Singh for insightful discussions and help with the CLEVR datasets. We also thank Sjoerd van Steenkiste for valuable feedback on an earlier version of this paper.

References

[1] Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

[2] Nicholas J Wade. The vision of helmholtz. *Journal of the History of the Neurosciences*, 30(4):405–424, 2021.

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

[4] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and variational inference in deep latent gaussian models. In *International Conference on Machine Learning*, volume 2, 2014.

[5] Stephen E. Palmer. Hierarchical structure in perceptual representation. *Cognitive Psychology*, 9(4):441–474, 1977.

[6] Wolf Singer. Binding by synchrony. *Scholarpedia*, 2:1657, 2007.

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

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

[9] Yi-Fu Wu, Jaesik Yoon, and Sungjin Ahn. Generative video transformer: Can objects be the words? In *International Conference on Machine Learning*, pages 11307–11318. PMLR, 2021.

[10] Jaesik Yoon, Yi-Fu Wu, Heechul Bae, and Sungjin Ahn. An investigation into pre-training object-centric representations for reinforcement learning. *CoRR*, abs/2302.04419, 2023.

[11] Taylor W. Webb, Shanka Subhra Mondal, and Jonathan D. Cohen. Systematic visual reasoning through object-centric relational abstraction. *CoRR*, abs/2306.02500, 2023.

[12] Taylor W. Webb, Shanka Subhra Mondal, and Jonathan D. Cohen. Systematic visual reasoning through object-centric relational abstraction. *CoRR*, abs/2306.02500, 2023.

[13] Andrea Dittadi, Samuele S. Papa, Michele De Vita, Bernhard Schölkopf, Ole Winther, and Francesco Locatello. Generalization and robustness implications in object-centric learning. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 5221–5285. PMLR, 2022.

[14] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 39–48, 2016.

[15] Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building machines that learn and think like people. *CoRR*, abs/1604.00289, 2016.

[16] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6306–6315, 2017.

[17] Ali Razavi, Aäron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 14837–14847, 2019.

[18] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 12873–12883. Computer Vision Foundation / IEEE, 2021.

[19] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in neural information processing systems*, pages 4790–4798, 2016.

[20] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8821–8831. PMLR, 2021.

[21] Gautam Singh, Fei Deng, and Sungjin Ahn. Illiterate DALL-E learns to compose. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[22] Marcelo G Mattar and Máté Lengyel. Planning in the brain. *Neuron*, 110(6):914–934, 2022.

[23] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

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

[25] Gautam Singh, Yeongbin Kim, and Sungjin Ahn. Neural systematic binder. In *The Eleventh International Conference on Learning Representations*, 2023.

[26] Laurynas Karazija, Iro Laina, and Christian Rupprecht. Clevrtex: A texture-rich benchmark for unsupervised multi-object segmentation. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021.

[27] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013.

[28] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *CoRR*, abs/2005.00341, 2020.

[29] Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. Videogpt: Video generation using VQ-VAE and transformers. *CoRR*, abs/2104.10157, 2021.

[30] Yao-Hung Hubert Tsai, Nitish Srivastava, Hanlin Goh, and Ruslan Salakhutdinov. Capsules with inverted dot-product attention routing. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[31] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[32] Nicholas Watters, Loïc Matthey, Christopher P. Burgess, and Alexander Lerchner. Spatial broadcast decoder: A simple architecture for learning disentangled representations in vaes. *CoRR*, abs/1901.07017, 2019.

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

[34] Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved VQGAN. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[35] Robert M. Gray. Vector quantization. *IEEE ASSP Magazine*, 1:4–29, 1984.

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

[37] Klaus Greff, Raphaël Lopez Kaufmann, Rishab Kabra, Nick Watters, Chris Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. *arXiv preprint arXiv:1903.00450*, 2019.

[38] Titas Anciukevicius, Christoph H Lampert, and Paul Henderson. Object-centric image generation with factored depths, locations, and appearances. *arXiv preprint arXiv:2004.00642*, 2020.

[39] Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Neural expectation maximization. In *Advances in Neural Information Processing Systems*, pages 6691–6701, 2017.

[40] Martin Engelcke, Adam R. Kosiorek, Oiwi Parker Jones, and Ingmar Posner. GENESIS: generative scene inference and sampling with object-centric latent representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

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

[42] Julius von Kügelgen, Ivan Ustyuzhaninov, Peter Gehler, Matthias Bethge, and Bernhard Schölkopf. Towards causal generative scene models via competition of experts, 2020.

[43] Yilun Du, Kevin Smith, Tomer Ulman, Joshua Tenenbaum, and Jiajun Wu. Unsupervised discovery of 3d physical objects from video, 2021.

[44] Rishabh Kabra, Daniel Zoran, Goker Erdogan, Loic Matthey, Antonia Creswell, Matthew Botvinick, Alexander Lerchner, and Christopher P. Burgess. Simone: View-invariant, temporally-abstracted object representations via unsupervised video decomposition. *arXiv preprint arXiv:2106.03849*, 2021.

[45] Ruixiang Zhang, Tong Che, B. Ivanovic, Renhao Wang, Marco Pavone, Yoshua Bengio, and Liam Paull. Robust and controllable object-centric learning through energy-based models. *arXiv preprint arXiv:2210.05519*, 2022.

[46] SM Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, and Geoffrey E Hinton. Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems*, pages 3225–3233, 2016.

[47] Zhixuan Lin, Yi-Fu Wu, Skand Vishwanath Peri, Weihao Sun, Gautam Singh, Fei Deng, Jindong Jiang, and Sungjin Ahn. Space: Unsupervised object-oriented scene representation via spatial attention and decomposition. In *International Conference on Learning Representations*, 2020.

[48] Jindong Jiang and Sungjin Ahn. Generative neurosymbolic machines. In *Advances in Neural Information Processing Systems*, 2020.

[49] Chang Chen, Fei Deng, and Sungjin Ahn. ROOTS: Object-centric representation and rendering of 3D scenes. *Journal of Machine Learning Research*, 22(259):1–36, 2021.

[50] Fei Deng, Zhuo Zhi, Donghun Lee, and Sungjin Ahn. Generative scene graph networks. In *International Conference on Learning Representations*, 2021.

[51] Zhixuan Lin, Yi-Fu Wu, Skand Vishwanath Peri, Jindong Jiang, and Sungjin Ahn. Improving generative imagination in object-centric world models. In *International Conference on Machine Learning*, pages 4114–4124, 2020.

[52] 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. In *International Conference on Learning Representations (ICLR)*, 2022.

[53] Anand Gopalakrishnan, Kazuki Irie, Jürgen Schmidhuber, and Sjoerd van Steenkiste. Unsupervised learning of temporal abstractions with slot-based transformers. *arXiv preprint arXiv:2203.13573*, 2022.

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

[55] Olivier J Hénaff, Skanda Koppula, Evan Shelhamer, Daniel Zoran, Andrew Jaegle, Andrew Zisserman, João Carreira, and Relja Arandjelović. Object discovery and representation networks. In *ECCV*, pages 123–143. Springer, 2022.

[56] Xudong Wang, Rohit Girdhar, Stella X. Yu, and Ishan Misra. Cut and learn for unsupervised object detection and instance segmentation, 2023.

[57] Xin Wen, Bingchen Zhao, Anlin Zheng, X. Zhang, and Xiaojuan Qi. Self-supervised visual representation learning with semantic grouping. *arXiv preprint arXiv:2205.15288*, 2022.

[58] Daniel Zoran, Rishabh Kabra, Alexander Lerchner, and Danilo J Rezende. Parts: Unsupervised segmentation with slots, attention and independence maximization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10439–10447, 2021.

[59] Eric Crawford and Joelle Pineau. Spatially invariant unsupervised object detection with convolutional neural networks. In *Proceedings of AAAI*, 2019.

[60] Eric Crawford and Joelle Pineau. Exploiting spatial invariance for scalable unsupervised object tracking. *arXiv preprint arXiv:1911.09033*, 2019.

[61] Jindong Jiang, Sepehr Janghorbani, Gerard De Melo, and Sungjin Ahn. Scalor: Generative world models with scalable object representations. In *International Conference on Learning Representations*, 2019.

[62] Daniil Kirilenko, Alexandr Korchemnyi, Alexey Kovalev, and Aleksandr Panov. Quantized disentangled representations for object-centric visual tasks, 2023.

[63] Yanbo Wang, Letao Liu, and Justin Dauwels. Slot-vae: Object-centric scene generation with slot attention. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 36020–36035. PMLR, 2023.

[64] Nicholas Watters, Loic Matthey, Sebastian Borgeaud, Rishabh Kabra, and Alexander Lerchner. Spriteworld: A flexible, configurable reinforcement learning environment. https://github.com/deepmind/spriteworld/, 2019.

[65] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910, 2017.

[66] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[67] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *NeurIPS*, 2022.

[68] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

A Additional Experimental Results

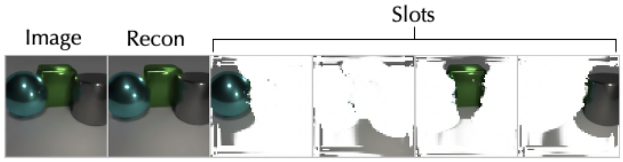

Figure 5: Sample scene we use in our codebook analysis.

A.1 Codebook Analysis

Latent Traversal. In this section, we qualitatively analyze the codebook for a sample scene. Figure 5 shows the sample we will use in our analysis. First, we run the image through the pretrained SVQ encoder to obtain a set of semantic discrete latents. Each latent represents one block from one slot and is provided by a prototype vector in the corresponding codebook for that block. To investigate the effect of traversing through the codebook, we replace each block with a different code in the codebook while keeping all other latents fixed. We then reconstruct the scene with the SVQ decoder and dVAE, essentially generating a new image that only differs from the original image by one discrete latent.

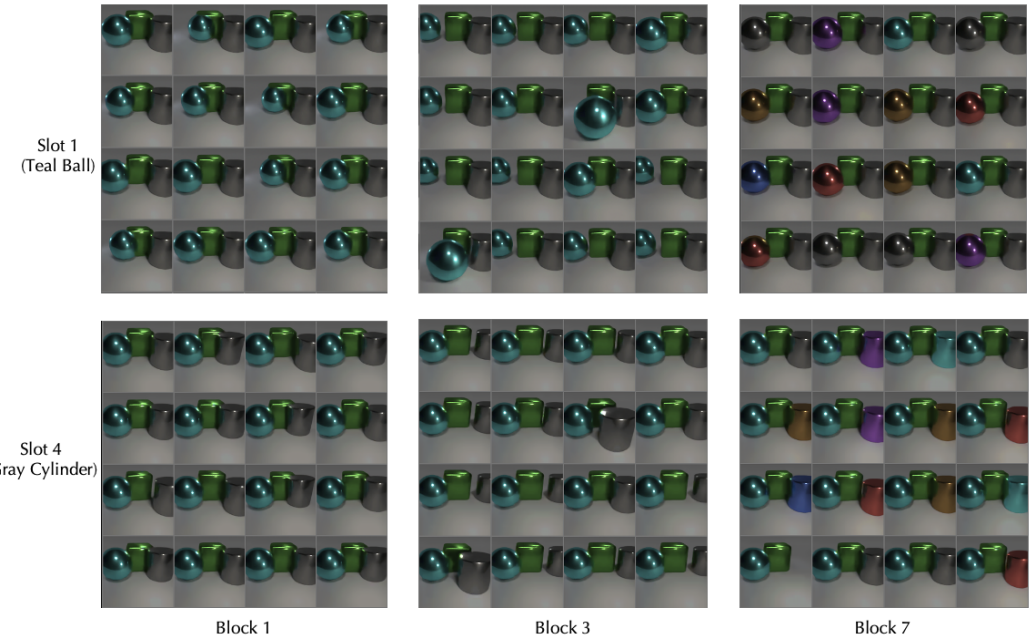

Figure 6: Latent traversal changing one latent in one block at a time while keeping all other latents fixed. The image is then reconstructed with the single changed latent.

Figure 6 shows the results for several sample blocks for the first slot (which corresponds to the teal ball) and the fourth slot (which corresponds to the gray cylinder). For each block, we choose the same set of 16 prototype vectors to display. First, we see that the slots are disentangled at the object level—changing one block in one slot does not affect the other objects. We also see that the different blocks specialize in different factors. Block 1 corresponds to the left and right placement of the object. Block 3 also corresponds to the placement of the object, but seems to also control the forward and backward placement of the object, as well as the size of the object. We notice that in this particular case, the factors of position and size are not completely disentangled. This may be because in this scene, the size depends on the placement of the object (e.g. closer objects are bigger). Block 7 controls the color of the object. We see that the same prototype vector seems to produce the same color, although there are some inconsistencies such as the disappearing cylinder in the bottom left.

The color also seems to be cleanly disentangled from the other factors—changing the color does not affect other factors like shape, size, or position.

Block Analysis. Next, to further explore the representation captured in the codebook, we visualize the objects that are attended to for different prototype vectors. To achieve this, we run the pretrained SVQ on 1000 images obtaining the semantic discrete latents and slot attention segmentation maps for the objects in the images. Then, for each prototype vector in the codebook, we find and visualize the corresponding slots that are utilizing that code in one of its blocks. Note that unlike [25], we do not need to do any k-means clustering to obtain this visualization since our representations are discrete representations in the codebook. Figures 7 and 8 show sample objects corresponding to three different prototype vectors for block 3 and block 7. We see that block 3 corresponds to object size and block 7 corresponds to object color. These results are consistent with the previous latent traversal experiments. Furthermore, the three prototype vectors we chose for block 7 correspond with the first three latents in Figure 6 (right), showing that these three prototype vectors represent gray, purple, and teal, respectively.

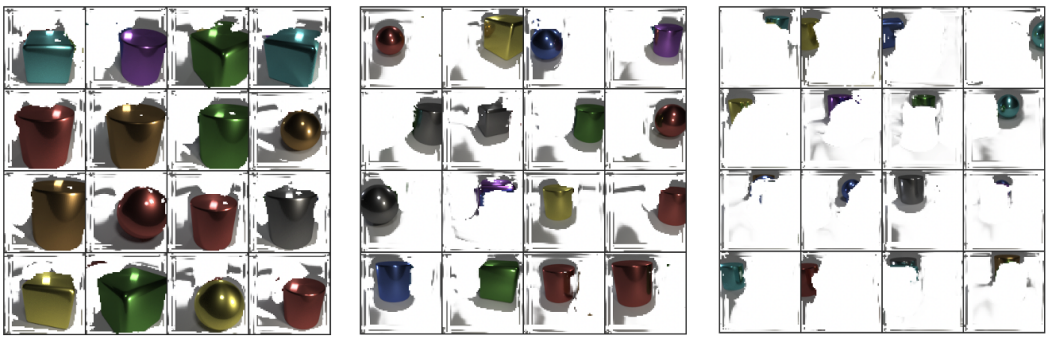

Figure 7: Objects attended to when the latent for block 3 is set to three different prototype vectors.

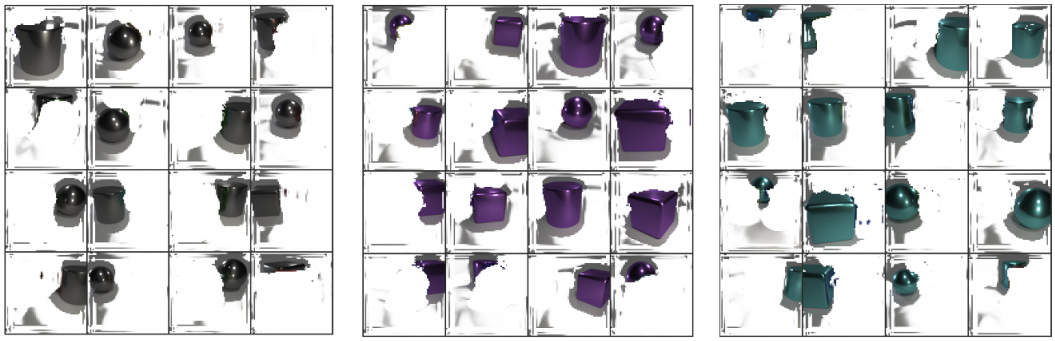

Figure 8: Objects attended to when the latent for block 7 is set to three different prototype vectors.

A.2 Comparison with Slot-Level Quantization

As discussed in Section 3, we hypothesize that slot-level discretization would struggle with complex scenes due to the combinatorial nature of the underlying factors of the objects. We test this hypothesis by running experiments on 2D Sprites and CLEVR-Easy where we set the number of blocks M to 1 and tune the size of the codebook, essentially doing slot-level quantization. In Figures 9, we show the masked attention of each slot on the input image as well as the image reconstruction. We find that with slot-level quantization, the model completely fails on the CLEVR-Easy dataset, unable to cleanly attend to the objects and reconstruct the image. On the 2D sprites dataset, we see that with slot discretization, one slot ends up attending to all the foreground objects and the model still cannot reconstruct the input image correctly. These results point to the importance of our choice to do block-level discretization.

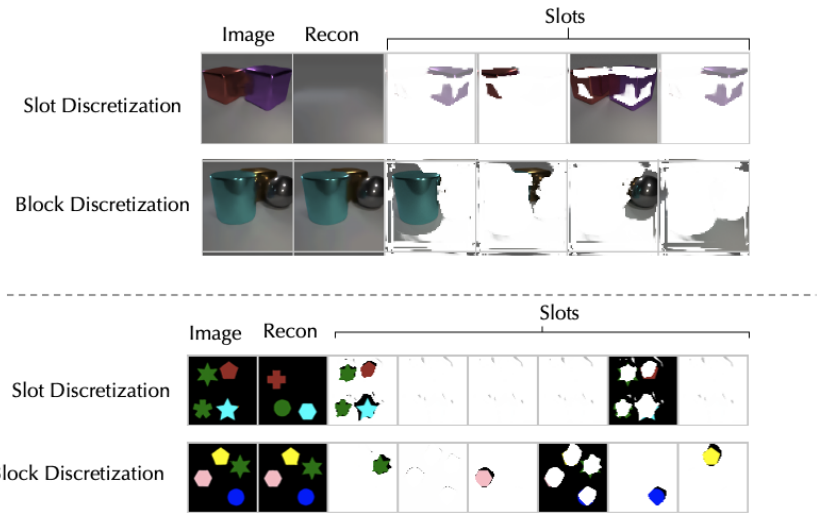

Figure 9: Comparison of slot discretization and block discretization on CLEVR-Easy (top) and 2D sprites (bottom).

B Implementation Details

B.1 Training and Implementation Details.

We use input images of 64x64 resolution for the 2D Sprites datasets and 128x128 for the CLEVR datasets. Each model is trained on NVIDIA Quadro RTX 8000 GPUs with 48GB memory. We also use Flash Attention [67] and half-precision floating-point format when training our models. We train SVQ for 400k iterations which takes around 80 hours for the CLEVR datasets and 50 hours for the 2D datasets. We then train the prior for 1 million iterations which takes around 40 hours. For the 2D Sprites dataset, similar to [10], we first train the underlying models on a dataset of random shapes without any relationship between the objects. We then train the prior models on the odd-one-out datasets.

B.2 Hyperparameters

		Dataset			
Module	Hyperparameter	CLEVR-Easy	CLEVR-Hard	2D Sprites	2D Sprites w/ BG
General	Batch Size	40	40	40	40
	Training Steps	400K	400K	400K	400K
	Image Size	128×128	128×128	64×64	64×64
SVQ	Codebook Dimension	256	128	256	32
	# Blocks	8	16	8	8
	Codebook Size	64	64	64	128
	# Iterations	3	3	3	3
	# Slots	4	4	6	8
	β	50	50	50	50
	Learning Rate	0.0001	0.0001	0.0001	0.0001

Table 6: Hyperparameters of our model used in our experiments.

Table 6 shows the hyperparameters we used for the different datasets in our experiments with SVQ. For the dVAE and Transformer Decoder, we follow the hyperparameters, architecture, and training procedure provided in [25] for CLEVR-Easy and CLEVR-Hard. For the 2D Sprites datasets, we use

the same hyperparameters as we do for CLEVR-Easy for those components. All models are trained with the Adam optimizer [68] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

B.3 Prior Models

For the SVQ and DVAE prior models, we use a transformer architecture with 8 layers, 4 heads, model dimension 192, feedforward dimension 768, and a dropout probability of 0.1. We use a learning rate of 0.0003 and 30,000 warmup steps. For VQ-VAE, we use a 20-layers PixelCNN prior, as proposed in the original paper [16].

B.4 Downstream Models

For the 2D Sprites downstream experiments, we use a transformer architecture with 3 layers, 8 heads, model dimension 192, feedforward dimension 768, and a dropout probability of 0.1 for all models. We use the Adam optimizer with a learning rate of 0.0003.

For the CLEVR-Hard downstream experiments, we use a transformer architecture with 8 layers, 4 heads, model dimension 192, feedforward dimension 768, and a dropout probability of 0.1 for all models. We use the Adam optimizer with a learning rate of 0.0001.