SPACE: SPATIALLY PARALLEL ATTENTION AND COMPONENT EXTRACTION FOR SCENE DECOMPOSITION

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a generative latent variable model for unsupervised scene decomposition. Our model, SPACE, provides a unified probabilistic modeling framework to combine the best of previous models. SPACE can explicitly provide factorized object representation per foreground object while also decomposing background segments of complex morphology. Previous models are good at either of these, but not both. With the proposed parallel-spatial attention, SPACE resolves the scalability problem of previous methods and thus makes the model applicable to scenes with a much larger number of objects without performance degradation. Besides, the foreground/background distinction of SPACE is more effective and intuitive than other methods because unlike other methods SPACE can detect static objects that have been difficult to detect as foreground. In experiments on Atari and 3D-Rooms, we show that SPACE achieves the above properties consistently in all experiments in comparison to SPAIR, IODINE, and GENESIS.

1 INTRODUCTION

One of the unsolved key challenges in machine learning is unsupervised learning of structured representation for a visual scene containing many objects with occlusion, partial observability, and complex background. When properly decomposed into meaningful symbolic entities such as objects and spaces, this structured representation brings many advantages of symbolic representation that contemporary deep learning with continuous vector representation has not been successful. The advantages include sample efficiency of downstream tasks such as a deep reinforcement learning agent (Mnih et al., 2013), ability of visual variable binding (Sun, 1992) for reasoning and causal inference over the relationship between the objects and agents in a scene, and compositionality and transferability for generalization. Recently, several methods have been proposed for this problem, unsupervised object-oriented scene decomposition, and these can be categorized into two approaches: *mixture-scene* models and *spatial-attention* models.

In the mixture-scene models (Greff et al., 2017; 2019; Burgess et al., 2019; Engelcke et al., 2019), a scene is explained by a mixture of K component images. The main benefit is to provide flexible segmentation maps for objects and background segments with complex morphology for which spatial attention with bounding boxes may have difficulties. However, this approach has limitations in scalability to scenes with many objects, and thus currently has been applied to scenes with only less than ten components. This is because, to obtain a jointly complete scene, a component needs to refer to other components, and thus inference is inherently performed sequentially either by an RNN generating a component per step or by iterative inference (Marino et al., 2018). Besides, because each component corresponds to a full-scale image, important physical features of objects like position and scale are only implicitly encoded in the scale of a full image, and thus further disentanglement is required to extract these useful features.

The spatial-attention models (Eslami et al., 2016; Crawford & Pineau, 2019) show the opposite properties to mixture-scene models. It detects objects using spatial attention and thus can obtain not only the appearance representation but also fully disentangled explicit geometric representation of an object such as position and scale. Such features grounded on the semantics of physics should be useful in many ways (e.g., sample efficiency, interpretability, geometric reasoning and inference, and

transferability). However, representing areas such as complex background segments that have too flexible morphology to be captured by spatial attention (e.g., based on rectangular bounding boxes) is the main limitation of this approach. It also shows scalability issues as objects are processed sequentially.

In this paper, we propose a method, called *Spatially Parallel Attention and Component Extraction* (SPACE), that combines the best of both approaches. SPACE learns to process foreground objects, which can be captured efficiently by bounding boxes, by using parallel spatial-attention while decomposing the remaining area that includes both morphologically complex objects and background segments by the component mixture. Thus, SPACE provides object-wise disentangled representation of foreground objects along with explicit properties like *where* per object while also providing decomposed representations of complex background components. Besides, we resolve the scalability issue of existing spatial attention methods by developing fully parallel foreground-object processing. In experiments on 3D-room scenes and Atari game scenes, we quantitatively and qualitatively compare the representation of SPACE to other models and show that SPACE combines the benefits of both approaches in addition to significant speed-ups due to the parallel foreground processing.

The contributions of the paper are as follows. First, we introduce a model that unifies the benefits of two existing approaches in a principled framework of probabilistic latent variable modeling. Second, we introduce a spatially parallel multi-object processing module and demonstrate that it can significantly mitigate the scalability problem of previous methods. Third, we demonstrate SPACE can detect static objects (e.g., the key in Montezuma's revenge) that always appear in the same position in all training images and thus are typically treated as a part of the background in other models. This problem has been a key challenge in unsupervised object detection. Lastly, we provide extensive analysis on and comparisons to previous models.

2 THE PROPOSED MODEL: SPACE

In this section, we describe our proposed model, Spatially Parallel Attention and Component Extraction (SPACE). The main idea of SPACE, presented in Figure 1, is to propose a unified probabilistic generative model that combines the benefits of the spatial-attention models and mixture-scene models.

2.1 GENERATIVE PROCESS

SPACE assumes that a scene $\mathbf{x}$ is decomposed into two latents: foreground $\mathbf{z}^{\text{fg}}$ and background $\mathbf{z}^{\text{bg}}$. The foreground is further decomposed into a set of independent foreground objects $\mathbf{z}^{\text{fg}} = \{\mathbf{z}_i^{\text{fg}}\}$ and the background is also decomposed further into a sequence of background segments $\mathbf{z}^{\text{bg}} = \mathbf{z}_{1:K}^{\text{bg}}$. The foreground is first generated and then the generation of the background is conditioned on the foreground. The image distributions of the foreground objects and the background components are combined together into the following pixel-wise mixture model to produce the complete image distribution.

$$p(\mathbf{x}|\mathbf{z}^{\text{fg}}, \mathbf{z}^{\text{bg}}) = \alpha \underbrace{p(\mathbf{x}|\mathbf{z}^{\text{fg}})}_{\text{Foreground}} + (1-\alpha) \sum_{k=1}^{K} \pi_k \underbrace{p(\mathbf{x}|\mathbf{z}_k^{\text{bg}})}_{\text{Background}}. \tag{1}$$

Here, the foreground mixing probability α is computed as $\alpha = f_\alpha(\mathbf{z}^{\text{fg}})$. This way, the foreground is given precedence in assigning its own mixing weight and the remaining is apportioned to the background. The mixing weight assigned to the background is further sub-divided among the K background components. These weights are computed as $\pi_k = f_{\pi_k}(\mathbf{z}_{1:k}^{\text{bg}})$ and $\sum_k \pi_k = 1$. With these notations, the complete generative model can be described as follows.

$$p(\mathbf{x}) = \iint p(\mathbf{x}|\mathbf{z}^{\text{fg}}, \mathbf{z}^{\text{bg}}) p(\mathbf{z}^{\text{bg}}|\mathbf{z}^{\text{fg}}) p(\mathbf{z}^{\text{fg}}) d\mathbf{z}^{\text{fg}} d\mathbf{z}^{\text{bg}} \tag{2}$$

We now describe the foreground and background models in more detail.

Foreground. SPACE implements $\mathbf{z}^{\text{fg}}$ as a structured latent. In this structure, an image is treated as if it were divided into $H \times W$ cells and each cell is tasked with modeling at most one (nearby) object

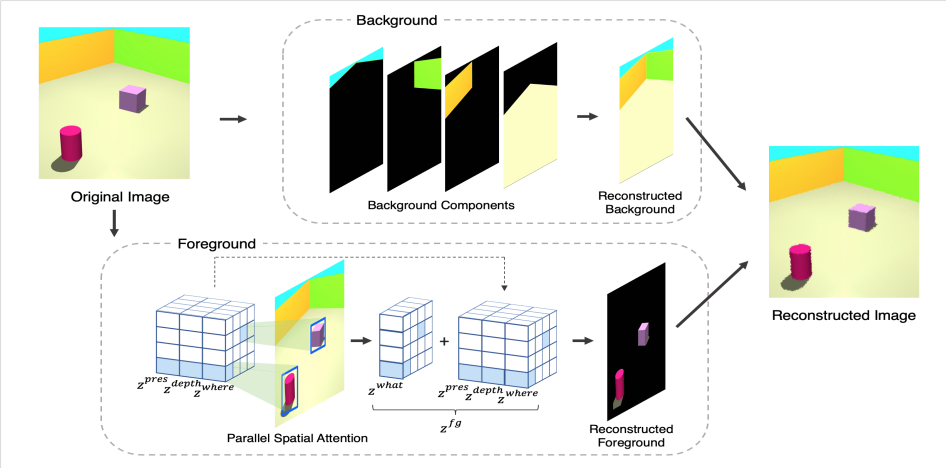

Figure 1: Illustration of the SPACE model. SPACE consists of a foreground module and a background module. In the foreground module, the input image is divided into a grid of $H \times W$ cells (4×4 in the figure). An image encoder is used to compute the z^{where}, z^{depth}, and z^{pres} for each cell in parallel. z^{where} is used to identify proposal bounding boxes and a spatial transformer is used to attend to each bounding box in parallel, computing a z^{what} encoding for each cell. The model selects patches using the bounding boxes and reconstructs them using a VAE from all the foreground latents z^{fg}. The background module segments the scene into K components (4 in the figure) using a pixel-wise mixture model. Each component consists of a set of latents $z^{\text{bg}} = (z^m, z^c)$ where z^m models the mixing probability of the component and z^c models the RGB distribution of the component. The components are combined to reconstruct the background using a VAE. The reconstructed background and foreground are then combined using a pixel-wise mixture model to generate the full reconstructed image.

in the scene. This type of structuring has been used in (Redmon et al., 2016; Santoro et al., 2017; Crawford & Pineau, 2019). Similar to SPAIR, in order to model an object, each cell i is associated with a set of latents $(\mathbf{z}_i^{\text{pres}}, \mathbf{z}_i^{\text{where}}, \mathbf{z}_i^{\text{depth}}, \mathbf{z}_i^{\text{what}})$. In this notation, $\mathbf{z}^{\text{pres}}$ is a binary random variable denoting if the cell models any object or not, $\mathbf{z}^{\text{where}}$ denotes the size of the object and its location relative to the cell, $\mathbf{z}^{\text{depth}}$ denotes the *depth* of the object to resolve occlusions and $\mathbf{z}^{\text{what}}$ models the object appearance and its mask. These latents may then be used to compute the foreground image component $p(\mathbf{x}|\mathbf{z}^{\text{fg}})$ which is modeled as a Gaussian distribution $\mathcal{N}(\mu^{\text{fg}}, \sigma^2_{\text{fg}})$. In practice, we treat σ^2_{fg} as a hyperparameter and decode only the mean image μ^{fg}. In this process, SPACE reconstructs the objects associated to each cell having $\mathbf{z}^{\text{pres}} = 1$. For each such cell, the model uses the $\mathbf{z}_i^{\text{what}}$ to decode the object glimpse and its mask and the glimpse is then positioned on a full-resolution canvas using $\mathbf{z}_i^{\text{where}}$ via the Spatial Transformer Network (Jaderberg et al., 2015). Using the object masks and $\mathbf{z}_i^{\text{depth}}$, all the foreground objects are combined into a single foreground mean-image μ^{fg} and the foreground mask α (See Appendix D for more details).

SPACE imposes a prior distribution on these latents as follows:

$$p(\mathbf{z}^{\text{fg}}) = \prod_{i=1}^{H \times W} p(\mathbf{z}_i^{\text{pres}}) \left(p(\mathbf{z}_i^{\text{where}}) p(\mathbf{z}_i^{\text{depth}}) p(\mathbf{z}_i^{\text{what}}) \right)^{\mathbf{z}_i^{\text{pres}}} \tag{3}$$

Here, only $\mathbf{z}_i^{\text{pres}}$ is modeled using a Bernoulli distribution while the remaining are modeled as Gaussian.

Background. To model the background, SPACE implements $\mathbf{z}_k^{\text{bg}}$, similar to GENESIS, as $(\mathbf{z}_k^m, \mathbf{z}_k^c)$ where $\mathbf{z}_k^m$ models the mixing probabilities π_k of the components and $\mathbf{z}_k^c$ models the RGB distribution $p(\mathbf{x}|\mathbf{z}_k^{\text{bg}})$ of the kth background component as a Gaussian $\mathcal{N}(\mu_i^{\text{bg}}, \sigma^2_{\text{bg}})$. The following prior is imposed upon these latents.

$$p(\mathbf{z}^{\text{bg}}|\mathbf{z}^{\text{fg}}) = \prod_{k=1}^{K} p(\mathbf{z}_k^c|\mathbf{z}_k^m) p(\mathbf{z}_k^m|\mathbf{z}_{<k}^m, \mathbf{z}^{\text{fg}}) \tag{4}$$

2.2 INFERENCE AND TRAINING

Since we cannot analytically evaluate the integrals in equation 2 due to the continuous latents $\mathbf{z}^{\text{fg}}$ and $\mathbf{z}^{\text{bg}}_{1:K}$, we train the model using a variational approximation. The true posterior on these variables is approximated as follows.

$$p(\mathbf{z}^{\text{bg}}_{1:K}, \mathbf{z}^{\text{fg}}|\mathbf{x}) \approx q(\mathbf{z}^{\text{fg}}|\mathbf{x}) \prod_{k=1}^{K} q(\mathbf{z}^{\text{bg}}_k|\mathbf{z}^{\text{bg}}_{<k}, \mathbf{z}^{\text{fg}}, \mathbf{x}) \tag{5}$$

This is used to derive the following ELBO to train the model using the reparameterization trick and SGD (Kingma & Welling, 2013).

$$\begin{aligned}\mathcal{L}(\mathbf{x}) = \mathbb{E}_{q(\mathbf{z}^{\text{fg}}, \mathbf{z}^{\text{bg}}_{1:K}|\mathbf{x})} \log \left[p(\mathbf{x}|\mathbf{z}^{\text{fg}}, \mathbf{z}^{\text{bg}})\right] - D_{\text{KL}}(q(\mathbf{z}^{\text{fg}}|\mathbf{x}) \parallel p(\mathbf{z}^{\text{fg}})) \\ - \mathbb{E}_{q(\mathbf{z}^{\text{fg}}|\mathbf{x})} \sum_{k=1}^{K} \mathbb{E}_{q(\mathbf{z}^{\text{bg}}_{<k}|\mathbf{x}, \mathbf{z}^{\text{fg}})} D_{\text{KL}}(q(\mathbf{z}^{\text{bg}}_k|\mathbf{z}^{\text{bg}}_{<k}, \mathbf{z}^{\text{fg}}, \mathbf{x}) \parallel p(\mathbf{z}^{\text{bg}}_k|\mathbf{z}^{\text{bg}}_{<k}, \mathbf{z}^{\text{fg}}))\end{aligned} \tag{6}$$

See Appendix B for the derivation of the ELBO and the related details.

Parallel Inference of Cell Latents. SPACE uses mean-field approximation when inferring the cell latents, so $\mathbf{z}^{\text{fg}}_i = (\mathbf{z}^{\text{pres}}_i, \mathbf{z}^{\text{where}}_i, \mathbf{z}^{\text{depth}}_i, \mathbf{z}^{\text{what}}_i)$ for each cell does not depend on other cells.

$$q(\mathbf{z}^{\text{fg}}|\mathbf{x}) = \prod_{i=1}^{HW} q(\mathbf{z}^{\text{pres}}_i|\mathbf{x}) \left(q(\mathbf{z}^{\text{where}}_i|\mathbf{x}) q(\mathbf{z}^{\text{depth}}_i|\mathbf{x}) q(\mathbf{z}^{\text{what}}_i|\mathbf{z}^{\text{where}}_i, \mathbf{x})\right)^{\mathbf{z}^{\text{pres}}_i}. \tag{7}$$

As shown in Figure 1, this allows each cell to act as an independent object detector, spatially attending to its own local region in parallel. This is in contrast to inference in SPAIR, where each cell's latents auto-regressively depend on some or all of the previously traversed cells in a row-major order i.e., $q(\mathbf{z}^{\text{fg}}|\mathbf{x}) = \prod_{i=1}^{HW} q(\mathbf{z}^{\text{fg}}_i|\mathbf{z}^{\text{fg}}_{<i}, \mathbf{x})$. However, this method becomes prohibitively expensive in practice as the number of objects increases. While Crawford & Pineau (2019) claim that these lateral connections are crucial for performance since they model dependencies between objects and thus prevent duplicate detections, we challenge this assertion by observing that 1) due to the bottom-up encoding conditioning on the input image, each cell should have information about its nearby area without explicitly communicating with other cells, and 2) in (physical) spatial space, two objects cannot exist at the same position. Thus, the relation and interference between objects should not be severe and the mean-field approximation is a good choice in our model. In our experiments, we verify empirically that this is indeed the case and observe that SPACE shows comparable detection performance to SPAIR while having significant gains in training speeds and efficiently scaling to scenes with many objects.

Preventing Box-Splitting. If the prior for the bounding box size is set to be too small, then the model could split a large object by multiple bounding boxes and when the size prior is too large, the model may not capture small objects in the scene, hence, causing a trade-off between the prior values of the bounding box size. To alleviate this problem, we found it helpful to introduce an auxiliary loss which we call the *boundary loss*. In the boundary loss, we construct a boundary of thickness b pixels along the borders of each glimpse. Then, we restrict an object to be inside this boundary and penalize the model if an object's mask overlaps with the boundary area. Thus, the model is penalized if it tries to split a large object by multiple smaller bounding boxes. A detailed implementation of the boundary loss is mentioned in Appendix C.

3 RELATED WORKS

Our proposed model is inspired by several recent works in unsupervised object-oriented scene decomposition. The Attend-Infer-Repeat (AIR) (Eslami et al., 2016) framework uses a recurrent neural network to attend to different objects in a scene and each object is sequentially processed one at a time. An object-oriented latent representation is prescribed that consists of 'what', 'where', and 'presence' variables. The 'what' variable stores the appearance information of the object, the 'where' variable represents the location of the object in the image, and the 'presence' variable controls how many steps the recurrent network runs and acts as an interruption variable when the model decides that all objects have been processed.

Since the number of steps AIR runs scales with the number of objects it attends to, it does not scale well to images with many objects. Spatially Invariant Attend, Infer, Repeat (SPAIR) (Crawford & Pineau, 2019) attempts to address this issue by replacing the recurrent network with a convolutional network. Similar to YOLO (Redmon et al., 2016), the locations of objects are specified relative to local grid cells rather than the entire image, which allow for spatially invariant computations. In the encoder network, a convolutional neural network is first used to map the image to a feature volume with dimensions equal to a pre-specified grid size. Then, each cell of the grid is processed *sequentially* to produce objects. This is done sequentially because the processing of each cell takes as input feature vectors and sampled objects of nearby cells that have already been processed. SPAIR therefore scales with the pre-defined grid size which also represents the maximum number of objects that can be detected. Our model uses an approach similar to SPAIR to detect foreground objects, but importantly we make the foreground object processing fully parallel to scale to large number of objects without performance degradation. Works based on Neural Expectation Maximization (Van Steenkiste et al., 2018; Greff et al., 2017) do achieve unsupervised object detection but do not explicitly model the *presence*, *appearance*, and *location* of objects. These methods also suffer from the problem of scaling to images with a large number of objects.

For unsupervised mixture-scene models, several recent models have shown promising results. MONet (Burgess et al., 2019) leverages a deterministic recurrent attention network that outputs pixel-wise masks for the scene components. A variational autoencoder (VAE) (Kingma & Welling, 2013) is then used to model each component. IODINE (Greff et al., 2019) approaches the problem from a spatial mixture model perspective and uses amortized iterative refinement of latent object representations within the variational framework. GENESIS (Engelcke et al., 2019) also uses a spatial mixture model which is encoded by component-wise latent variables. Relationships between these components are captured with an autoregressive prior, allowing complete images to be modeled by a collection of components.

4 EVALUATION

We evaluate our model on two datasets: 1) an Atari (Bellemare et al., 2013) dataset that consists of random images from a pretrained agent playing the games, and 2) a generated 3D-room dataset that consists of images of a walled enclosure with a random number of objects on the floor. In order to test the scalability of our model, we use both a small 3D-room dataset that has 4-8 objects and a large 3D-room dataset that has 18-24 objects. Each image is taken from a random camera angle and the colors of the objects, walls, floor, and sky are also chosen at random. Additional details of the datasets can be found in the Appendix E.

Baselines. We compare our model against two mixture-scene models (IODINE and GENESIS) and one spatial-attention model (SPAIR). Since SPAIR does not have an explicit background component, we add an additional VAE for processing the background. Additionally, we test against two implementations of SPAIR: one where we train on the entire image using a 16×16 grid and another where we train on random 32×32 pixel patches using a 4×4 grid. We denote the former model as SPAIR and the latter as SPAIR-P. SPAIR-P is consistent with the SPAIR's alternative training regime on Space Invaders demonstrated in Crawford & Pineau (2019) to address the slow training of SPAIR on the full grid size because of its sequential inference. Lastly, for performance reasons, unlike the original SPAIR implementation, we use parallel processing for rendering the objects from their respective latents onto the canvas[1] for both SPAIR and SPAIR-P. Thus, our SPAIR implementation can be seen as a stronger baseline in terms of the speed than the original SPAIR. The complete details of the architecture used is given in Appendix D.

4.1 QUALITATIVE COMPARISON OF INFERRED REPRESENTATIONS

In this section, we provide a qualitative analysis of the generated representations of the different models. For each model, we performed a hyperparameter search and present the results for the best settings of hyperparameters for each environment. Figure 2 shows sample scene decompositions of our baselines from the 3D-Room dataset and Figure 3 shows the results on Atari. Note that

[1]It is important to note that the worst case complexity of rendering is $\mathcal{O}(hw \times HW)$, (where (h, w) is the image size) which is extremely time consuming when we have large image size and/or large number of objects.

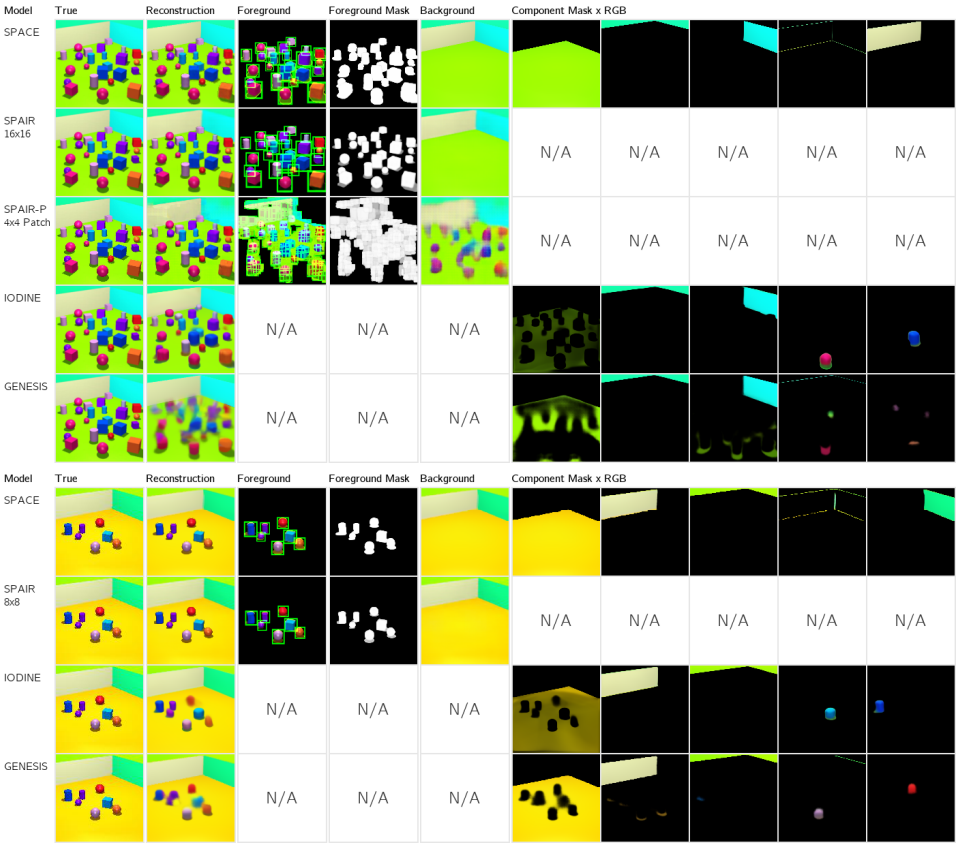

Figure 2: Qualitative comparison between SPACE , SPAIR, SPAIR-P, IODINE and GENESIS for the 3D-Room dataset.

SPAIR does not use component masks and IODINE and GENESIS do not separate foreground from background, hence the corresponding cells are left empty. Additionally, we only show a few representative components for IODINE and GENESIS since we ran those experiments with larger K than can be displayed. More qualitative results of SPACE can be found in Appendix A.

IODINE & GENESIS. In the 3D-Room environment, IODINE is able to segment the objects and the background into separate components. However, it occasionally does not properly decompose objects (in the Large 3D-room results, the orange sphere on the right is not reconstructed) and may generate blurry objects. GENESIS is able to segment the background walls, floor, and sky into multiple components. It is able to capture blurry foreground objects in the Small 3D-Room, but is not able to cleanly capture foreground objects with the larger number of objects in the Large 3D-Room. In Atari, for all games, both IODINE and GENESIS fail to capture the foreground properly. We believe this is because the objects in Atari games are smaller, less regular and lack the obvious latent factors like color and shape as in the 3D dataset, which demonstrates that detection-based approaches are more appropriate in this case.

SPAIR & SPAIR-P. SPAIR is able to detect tight bounding boxes in both 3D-Room and most Atari games (it does not work as well on dynamic background games, which we discuss below). SPAIR-P, however, often fails to detect the foreground objects in proper bounding boxes, frequently uses multiple bounding boxes for one object and redundantly detects parts of the background as foreground objects. This is a limitation of the patch training as the receptive field of each patch is limited to a 32×32 glimpse, hence prohibiting it to detect objects larger than that and making it difficult to distinguish the background from foreground. These two properties are illustrated well in Space Invaders, where it is able to detect the small aliens, but it detects the long piece of background ground on the bottom of the image as foreground objects.

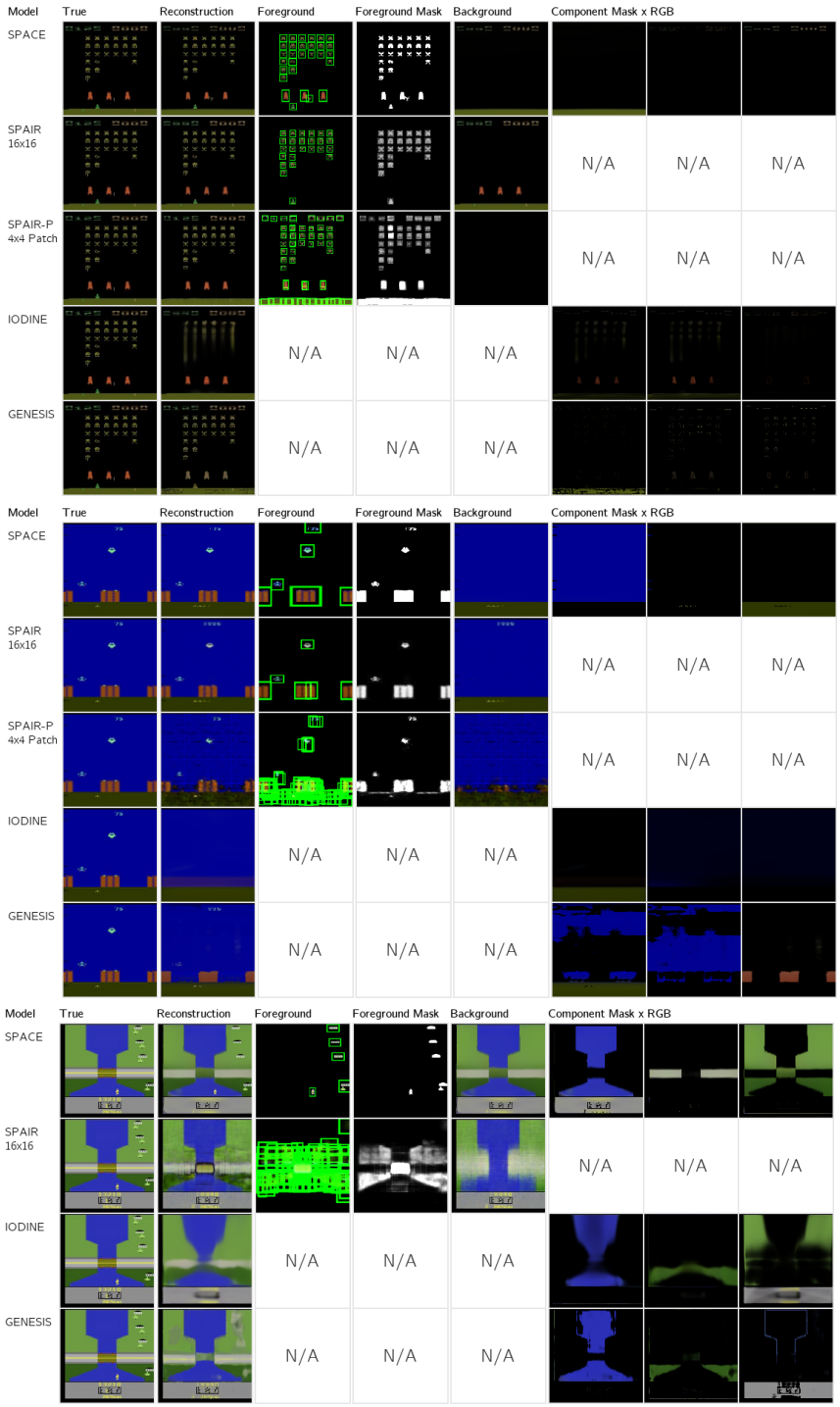

Figure 3: Qualitative comparison between SPACE , SPAIR, IODINE and GENESIS for Space Invaders, Air Raid, and River Raid.

SPACE. In 3D-Room, SPACE is able to accurately detect almost all objects despite the large variations in object positions, colors and shapes, while producing a clean segmentation of the background walls, ground, and sky. This is in contrast to the SPAIR model, while being able to provide a similar foreground detection quality, encodes the whole background into a single component, which makes the representation less disentangled and the reconstruction more blurry. Similarly in Atari, SPACE

consistently captures all foreground objects while producing clean background segmentation across many different games.

Dynamic Backgrounds. SPACE and SPAIR exhibit some very interesting behavior when trained on games with dynamic backgrounds. For the most static game - Space Invaders, both SPACE and SPAIR work well. For Air Raid, in which the background building moves, SPACE captures all objects accurately while providing a two-component segmentation, whereas SPAIR and SPAIR-P produce splitting and heavy re-detections. In the most dynamic games, SPAIR completely fails because of the difficulty to model dynamic background with a single VAE component, while SPACE is able to perfectly segment the blue racing track while accurately detecting all foreground objects.

Foreground vs Background. Typically, foreground is the dynamic local part of the scene that we are interested in, and background is the relatively static and global part. This definition, though intuitive, is ambiguous. Some of the objects, such as the red shields near the bottom in Space Invaders and the key in Montezuma's Revenge (Figure 5) are detected as foreground objects in SPACE, but are considered background in SPAIR. Though these objects are static[2], they are important elements of the games and should be considered as foreground objects. Similar behavior is observed in Atlantis, where SPACE detects some foreground objects from the middle base that is above the water. We believe this is an interesting property of SPACE and could be very important for providing useful representations for downstream tasks. By using a spatial broadcast network (Watters et al., 2019) which is much weaker when compared to other decoders like sub-pixel convolutional nets (Shi et al. (2016)), we limit the capacity of background module, which favors modeling static objects as foreground rather than background.

Boundary Loss. We notice SPAIR sometimes splits objects into two whereas SPACE is able to create the correct bounding box for the objects (for example, see Air Raid). This may be attributed to the addendum of the auxiliary boundary loss in the SPACE model that would penalize splitting an object with multiple bounding boxes.

4.2 Quantitative Comparison

In this section we compare SPACE with the baselines in several quantitative metrics. We first note that each of the baseline models has a different *decomposition capacity* ($\mathcal{C}$), which we define as the capability of the model to decompose the scene into its semantic constituents such as the foreground objects and the background segmented components. For SPACE, the decomposition capacity is equal to the number of grid cells $H \times W$ (which is the maximum number of foreground objects that can be detected) plus the number of background components K. For SPAIR, the decomposition capacity is equal to the number of grid cells $H \times W$ plus 1 for background. For IODINE and GENESIS, it is equal to the number of components K.

For each experiment, we compare the metrics for each model with similar decomposition capacities. This way, each model can decompose the image into the same number of components. For a setting in SPACE with a grid size of $H \times W$ with K_{SPACE} components, the equivalent settings in IODINE and GENESIS would be with $\mathcal{C} = (H \times W) + K_{\text{SPACE}}$. The equivalent setting in SPAIR would be a grid size of $H \times W$.

As previously shown, SPAIR-P does not work well in many of our environments, so we do not include it in these quantitative experiments. As in the qualitative section, we use the SPAIR implementation with sequential inference and parallel rendering in order to speed up the experiments.

Gradient Step Latency. The leftmost chart of Figure 4 shows the time taken to complete one gradient step (forward and backward propagation) for different decomposition capacities for each of the models. We see that SPAIR's latency grows with the number of cells because of the sequential nature of its latent inference step. Similarly GENESIS and IODINE's latency grows with the number of components K because each component is processed sequentially in both the models. IODINE is the slowest overall with its computationally expensive iterative inference procedure. Furthermore, both IODINE and GENESIS require storing data for each of the K components, so we were unable to run our experiments on 256 components or greater before running out of memory on our 22GB

[2]The pretrained agent used to collect the game frames for Montezuma's Revenge does not reach a state in which it can actually capture the key, hence the position of the key remains constant over time

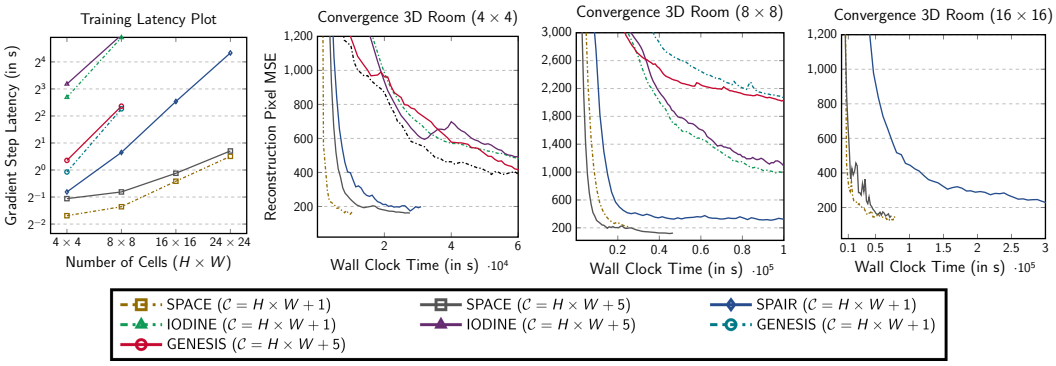

Figure 4: Quantitative performance comparison between SPACE , SPAIR, IODINE and GENESIS in terms of batch-processing time during training, training convergence and converged pixel MSE. Convergence plots showing pixel-MSE were computed on a held-out set during training.

Table 1: Comparison of SPACE with SPACE-WB (SPACE without boundary loss) and the SPAIR baseline with respect to the quality of the bounding boxes in the 3D-Room setting.

Model	Dataset	Avg. Precision IoU Threshold $= 0.5$	Avg. Precision IoU Threshold $\in [0.5 : 0.05 : 0.95]$	Object Count Error Rate
SPACE (16×16)	3D-Room Large	0.8942	0.4608	0.0507
SPACE-WB (16×16)	3D-Room Large	0.8925	0.4428	0.0947
SPAIR (16×16)	3D-Room Large	0.9076	0.4453	0.0427
SPACE (8×8)	3D-Room Small	0.9057	0.5203	0.0330
SPACE-WB (8×8)	3D-Room Small	0.9048	0.5057	0.0566
SPAIR (8×8)	3D-Room Small	0.9036	0.4883	0.0257

GPU. On the other hand, SPACE employs parallel processing for the foreground which makes it scalable to large grid sizes, allowing it to detect a large number of foreground objects without any significant performance degradation. Although this data was collected for gradient step latency, this comparison implies a similar relationship exists with inference time which is a main component in the gradient step.

Time for Convergence. The remaining three charts in Figure 4 show the amount of time each model takes to converge in different experimental settings. We use the pixel-wise mean squared error (MSE) as a measurement of how close a model is to convergence. We see that not only does SPACE achieve the lowest MSE, it also converges the quickest out of all the models.

Average Precision and Error Rate. In order to assess the quality of our bounding box predictions and the effectiveness of boundary loss, we measure the Average Precision and Object Count Error Rate of our predictions. Our results are shown in Table 1. We only report these metrics for 3D-Room since we have access to the ground truth bounding boxes for each of the objects in the scene. All three models have very similar average precision and error rate. Despite being parallel in its inference, SPACE has a comparable count error rate to that of SPAIR. SPACE also achieves better average precision and count error rate compared to its variant without the boundary loss (SPACE-WB), which shows the efficacy of our proposed loss.

From our experiments, we can assert that SPACE can produce similar quality bounding boxes as SPAIR while 1) having orders of magnitude faster inference and gradient step time, 2) converging more quickly, 3) scaling to a large number of objects without significant performance degradation, and 4) providing complex background segmentation.

5 Conclusion

We propose SPACE, a unified probabilistic model that combines the benefits of the object representation models based on spatial attention and the scene decomposition models based on component mixture. SPACE can explicitly provide factorized object representation per foreground object while also decomposing complex background segments. SPACE also achieves a significant speed-up and

thus makes the model applicable to scenes with a much larger number of objects without performance degradation. Besides, the detected objects in SPACE are also more intuitive than other methods. We show the above properties of SPACE on Atari and 3D-Rooms. Interesting future directions are to replace the sequential processing of background by a parallel one and to improve the model for natural images. Our next plan is to apply SPACE for object-oriented model-based reinforcement learning.

REFERENCES

Jonathan T. Barron. Continuously differentiable exponential linear units. *ArXiv*, abs/1704.07483, 2017.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.

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.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *ICLR*, 2016.

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

Martin Engelcke, Adam R. Kosiorek, Oiwi Parker Jones, and Ingmar Posner. Genesis: Generative scene inference and sampling with object-centric latent representations, 2019.

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*, pp. 3225–3233, 2016.

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

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.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Advances in neural information processing systems*, pp. 2017–2025, 2015.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

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

Joseph Marino, Yisong Yue, and Stephan Mandt. Iterative amortized inference. *arXiv preprint arXiv:1807.09356*, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *ArXiv*, abs/1312.5602, 2013.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pp. 4967–4976, 2017.

Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Ron Sun. On variable binding in connectionist networks. *Connection Science*, 4(2):93–124, 1992.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop, neural networks for machine learning. *Tech Report*, 2012. URL `https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Sjoerd Van Steenkiste, Michael Chang, Klaus Greff, and Jürgen Schmidhuber. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. *arXiv preprint arXiv:1802.10353*, 2018.

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

Yuxin Wu and Kaiming He. Group normalization. In *The European Conference on Computer Vision (ECCV)*, September 2018.

Yuxin Wu et al. Tensorpack. `https://github.com/tensorpack/`, 2016.

A ADDITIONAL RESULTS OF SPACE

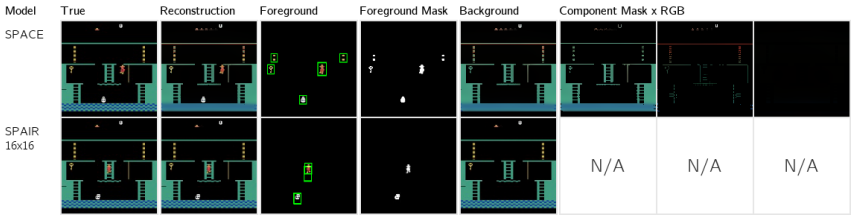

Figure 5: Case illustration of Montezuma's Revenge comparing object-detection behaviour in SPACE and SPAIR.

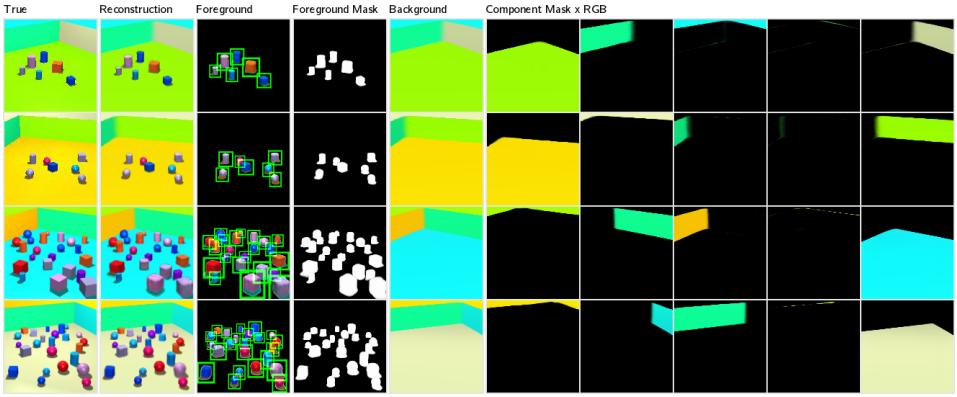

Figure 6: Qualitative demonstration of SPACE trained on the 3D-room dataset.

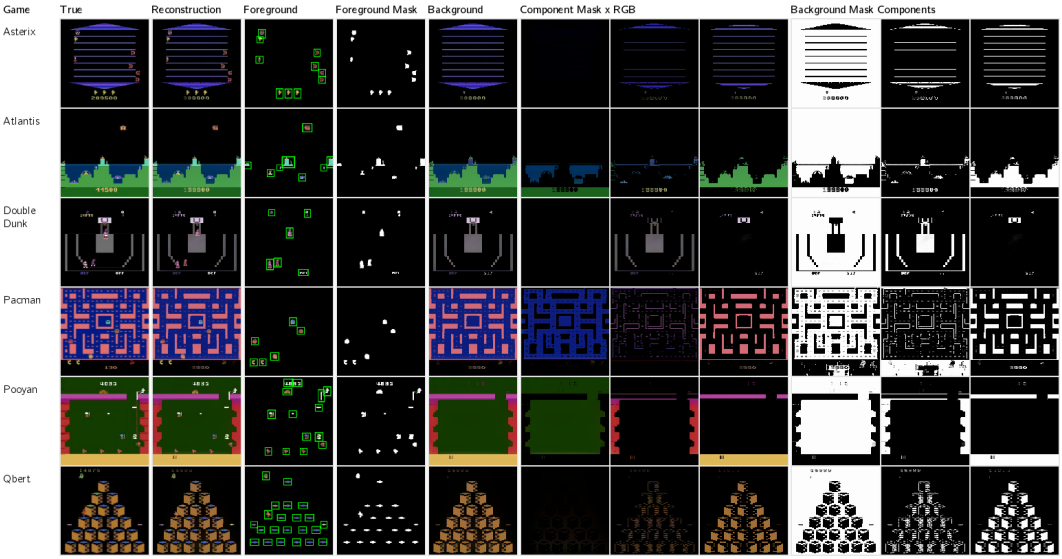

Figure 7: Qualitative demonstration of SPACE trained jointly on a selection of 10 ATARI games. We show 6 games with complex background here.

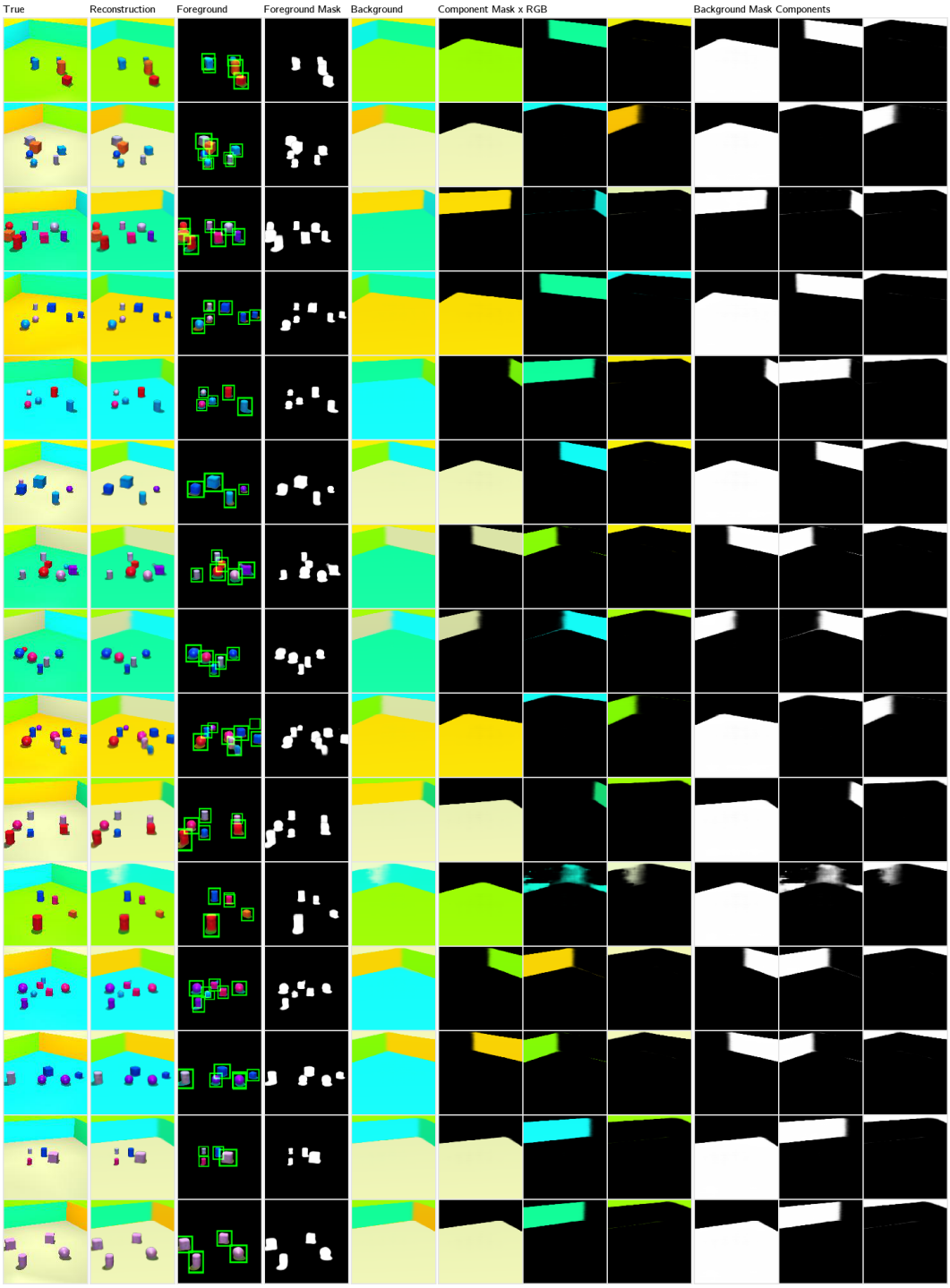

Figure 8: Object detection and background segmentation using SPACE on 3D-Room data set with small number of objects. Each row corresponds to one input image.

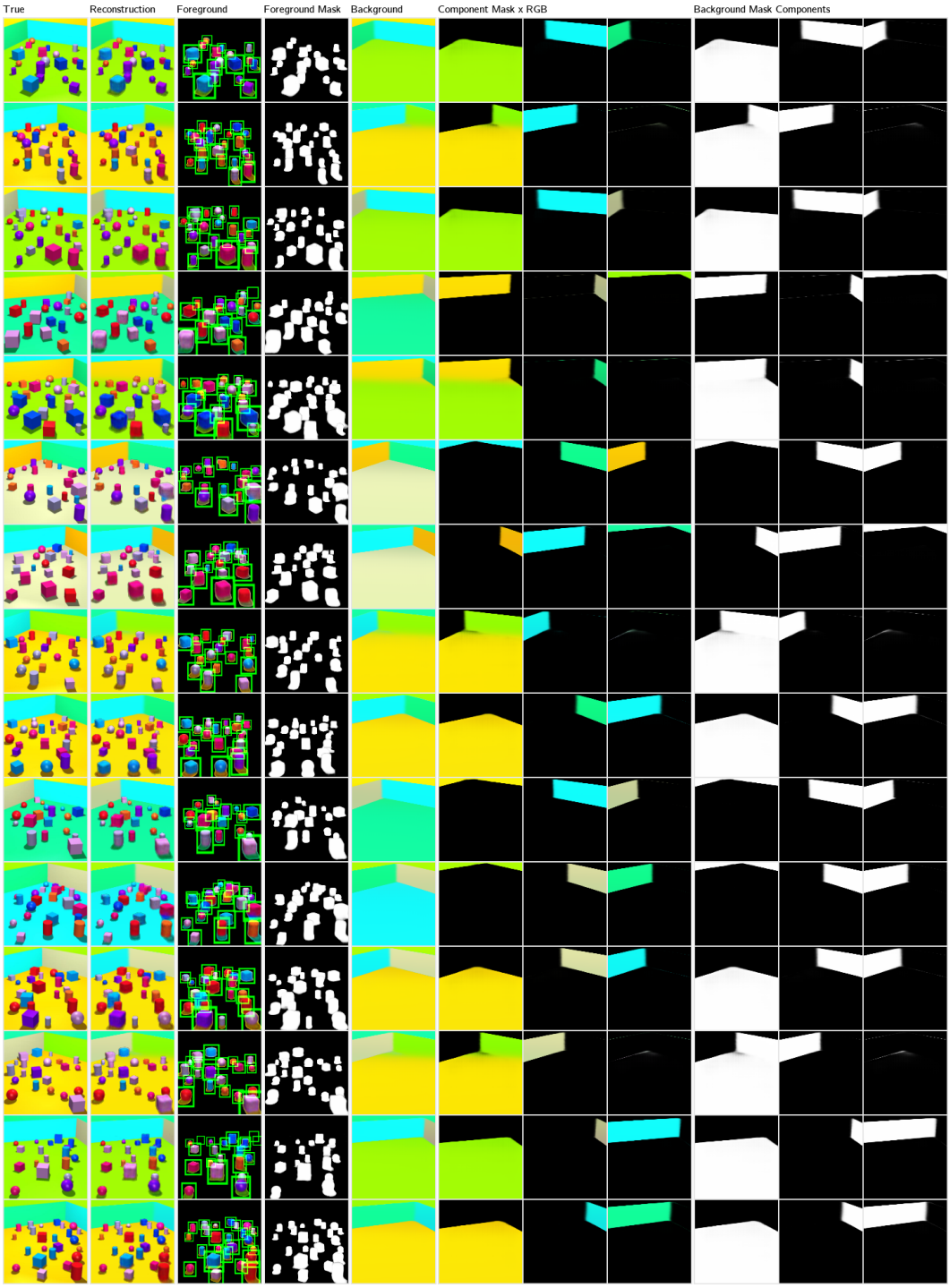

Figure 9: Object detection and background segmentation using SPACE on 3D-Room data set with large number of objects.

B ELBO DERIVATIONS

In this section, we derive the ELBO for the log-likelihood $\log p(\mathbf{x})$.

$$
\begin{aligned}
\log p(\mathbf{x}) &= \log \iint p(\mathbf{x}|\mathbf{z}^{\text{fg}}, \mathbf{z}^{\text{bg}}_{1:K}) \left[p(\mathbf{z}^{\text{fg}}) \prod_{k=1}^{K} p(\mathbf{z}^{\text{bg}}_k|\mathbf{z}^{\text{bg}}_{<k}, \mathbf{z}^{\text{fg}}) \right] d\mathbf{z}^{\text{bg}}_{1:K} d\mathbf{z}^{\text{fg}} \\
&= \log \mathbb{E}_{q(\mathbf{z}^{\text{fg}}, \mathbf{z}^{\text{bg}}_{1:K}|\mathbf{x})} p(\mathbf{x}|\mathbf{z}^{\text{fg}}, \mathbf{z}^{\text{bg}}_{1:K}) \frac{p(\mathbf{z}^{\text{fg}}) \prod_{k=1}^{K} p(\mathbf{z}^{\text{bg}}_k|\mathbf{z}^{\text{bg}}_{<k}, \mathbf{z}^{\text{fg}})}{q(\mathbf{z}^{\text{fg}}|\mathbf{x}) \prod_{k=1}^{K} q(\mathbf{z}^{\text{bg}}_k|\mathbf{z}^{\text{bg}}_{<k}, \mathbf{z}^{\text{fg}}, \mathbf{x})} \\
&\geq \mathbb{E}_{q(\mathbf{z}^{\text{fg}}, \mathbf{z}^{\text{bg}}_{1:K}|\mathbf{x})} \log \left[p(\mathbf{x}|\mathbf{z}^{\text{fg}}, \mathbf{z}^{\text{bg}}_{1:K}) \frac{p(\mathbf{z}^{\text{fg}}) \prod_{k=1}^{K} p(\mathbf{z}^{\text{bg}}_k|\mathbf{z}^{\text{bg}}_{<k}, \mathbf{z}^{\text{fg}})}{q(\mathbf{z}^{\text{fg}}|\mathbf{x}) \prod_{k=1}^{K} q(\mathbf{z}^{\text{bg}}_k|\mathbf{z}^{\text{bg}}_{<k}, \mathbf{z}^{\text{fg}}, \mathbf{x})} \right] \\
&= \mathbb{E}_{q(\mathbf{z}^{\text{fg}}, \mathbf{z}^{\text{bg}}_{1:K}|\mathbf{x})} \log \left[p(\mathbf{x}|\mathbf{z}^{\text{fg}}, \mathbf{z}^{\text{bg}}) \right] - D_{\text{KL}}(q(\mathbf{z}^{\text{fg}}|\mathbf{x}) \parallel p(\mathbf{z}^{\text{fg}})) \\
&\quad - \mathbb{E}_{q(\mathbf{z}^{\text{fg}}|\mathbf{x})} \sum_{k=1}^{K} \mathbb{E}_{q(\mathbf{z}^{\text{bg}}_{<k}|\mathbf{x}, \mathbf{z}^{\text{fg}})} D_{\text{KL}}(q(\mathbf{z}^{\text{bg}}_k|\mathbf{z}^{\text{bg}}_{<k}, \mathbf{z}^{\text{fg}}, \mathbf{x}) \parallel p(\mathbf{z}^{\text{bg}}_k|\mathbf{z}^{\text{bg}}_{<k}, \mathbf{z}^{\text{fg}}))
\end{aligned}
$$

KL Divergence for the Foreground Latents Under the SPACE 's approximate inference, the $D_{\text{KL}}(q(\mathbf{z}^{\text{fg}}|\mathbf{x}) \parallel p(\mathbf{z}^{\text{fg}}))$, is evaluated as follows.

$$
\begin{aligned}
&D_{\text{KL}}(q(\mathbf{z}^{\text{fg}}|\mathbf{x}) \parallel p(\mathbf{z}^{\text{fg}})) \\
&\quad = \mathbb{E}_{q(\mathbf{z}^{\text{fg}}|\mathbf{x})} \log \frac{q(\mathbf{z}^{\text{fg}}|\mathbf{x})}{p(\mathbf{z}^{\text{fg}})} \\
&\quad = \mathbb{E}_{q(\mathbf{z}^{\text{fg}}|\mathbf{x})} \log \frac{\prod_{i=1}^{HW} q(\mathbf{z}^{\text{pres}}_i|\mathbf{x}) \left(q(\mathbf{z}^{\text{where}}_i|\mathbf{x}) q(\mathbf{z}^{\text{what}}_i|\mathbf{z}^{\text{where}}_i, \mathbf{x}) q(\mathbf{z}^{\text{depth}}_i|\mathbf{x}) \right)^{\mathbf{z}^{\text{pres}}_i}}{\prod_{i=1}^{HW} p(\mathbf{z}^{\text{pres}}_i) \left(p(\mathbf{z}^{\text{where}}_i) p(\mathbf{z}^{\text{what}}_i) p(\mathbf{z}^{\text{depth}}_i) \right)^{\mathbf{z}^{\text{pres}}_i}} \\
&\quad = \sum_{i=1}^{HW} \Bigg[D_{\text{KL}}(q(\mathbf{z}^{\text{pres}}_i|\mathbf{x}) \parallel p(\mathbf{z}^{\text{pres}}_i)) + \mathbb{E}_{q(\mathbf{z}^{\text{pres}}_i|\mathbf{x})} \mathbf{z}^{\text{pres}}_i \Big[D_{\text{KL}}(q(\mathbf{z}^{\text{where}}_i|\mathbf{x}) \parallel p(\mathbf{z}^{\text{where}}_i)) \\
&\quad + \mathbb{E}_{q(\mathbf{z}^{\text{where}}_i|\mathbf{x})} D_{\text{KL}}(q(\mathbf{z}^{\text{what}}_i|\mathbf{x}) \parallel p(\mathbf{z}^{\text{what}}_i)) + D_{\text{KL}}(q(\mathbf{z}^{\text{depth}}_i|\mathbf{x}) \parallel p(\mathbf{z}^{\text{depth}}_i)) \Big] \Bigg]
\end{aligned}
$$

KL Divergence for the Background Latents Under our GENESIS-like modeling of inference for the background latents, the KL term for the background is evaluated as follows.

$$
\begin{aligned}
&D_{\text{KL}}(q(\mathbf{z}^{\text{bg}}_k|\mathbf{z}^{\text{bg}}_{<k}, \mathbf{z}^{\text{fg}}, \mathbf{x}) \parallel p(\mathbf{z}^{\text{bg}}_k|\mathbf{z}^{\text{bg}}_{<k}, \mathbf{z}^{\text{fg}})) \\
&\quad = D_{\text{KL}}(q(\mathbf{z}^{m}_k|\mathbf{z}^{m}_{<k}, \mathbf{z}^{\text{fg}}, \mathbf{x}) \parallel p(\mathbf{z}^{m}_k|\mathbf{z}^{m}_{<k}, \mathbf{z}^{\text{fg}})) + \mathbb{E}_{q(\mathbf{z}^{m}_k|\mathbf{z}^{m}_{<k}, \mathbf{z}^{\text{fg}}, \mathbf{x})} D_{\text{KL}}(q(\mathbf{z}^{c}_k|\mathbf{z}^{m}_k, \mathbf{x}) \parallel p(\mathbf{z}^{c}_k|\mathbf{z}^{m}_k))
\end{aligned}
$$

Relaxed treatment of $\mathbf{z}^{\text{pres}}$ In our implementation, we model the Bernoulli random variable $\mathbf{z}^{\text{pres}}_i$ using the Gumbel-Softmax distribution (Jang et al., 2016). We use the relaxed value of $\mathbf{z}^{\text{pres}}$ in the entire training and use hard samples only for the visualizations.

C BOUNDARY LOSS

In this section we elaborate on the implementation details of the *boundary loss*. We construct a kernel of the size of the glimpse, $gs \times gs$ (we use $gs = 32$) with a boundary gap of $b = 6$ having negative uniform weights inside the boundary and a zero weight in the region between the boundary and the glimpse. This ensures that the model is penalized when the object is outside the boundary. This kernel is first mapped onto the global space via STN Jaderberg et al. (2015) to obtain the global kernel. This is then multiplied element-wise with global object mask α to obtain the *boundary loss map*. The objective of the loss is to minimize the mean of this *boundary loss map*. In addition to the ELBO, this loss is also back-propagated via *RMSProp* (Tieleman & Hinton. (2012)). This loss, due to the boundary constraint, enforces the bounding boxes to be less tight and results in lower average precision, so we disable the loss and optimize only the ELBO after the model has converged well.

D IMPLEMENTATION DETAILS

D.1 ALGORITHMS

Algorithm 1 and Algorithm 2 present SPACE's inference for foreground and background. Algorithm 3 show the details of the generation process of the background module. For foreground generation, we simply sample the latent variables from the priors instead of conditioning on the input. Note that, for convenience the algorithms for the foreground module and background module are presented with `for` loops, but inference for all variables of the foreground module are implemented as parallel convolution operations and most operations of the background module (barring the LSTM module) are parallel as well.

Algorithm 1: Foreground Inference

Input: image $\boldsymbol{x}$
Output: foreground mask α, appearance μ^{fg}
$\boldsymbol{e}^{\text{img}}$ = ImageEncoderFg($\boldsymbol{x}$)
for $i \leftarrow 1$ **to** HW **do**
  `/* The following is performed in parallel */`
  $\boldsymbol{\rho}_i$ = ZPresNet($\boldsymbol{e}_i^{\text{img}}$)
  $[\boldsymbol{\mu}_i^{\text{depth}}, \boldsymbol{\sigma}_i^{\text{depth}}] = \text{ZDepthNet}(\boldsymbol{e}_i^{\text{img}})$
  $[\boldsymbol{\mu}_i^{\text{scale}}, \boldsymbol{\sigma}_i^{\text{scale}}] = \text{ZScaleNet}(\boldsymbol{e}_i^{\text{img}})$
  $[\boldsymbol{\mu}_i^{\text{shift}}, \boldsymbol{\sigma}_i^{\text{shift}}] = \text{ZShiftNet}(\boldsymbol{e}_i^{\text{img}})$
  $\mathbf{z}_i^{\text{pres}} \sim \text{Bern}(\boldsymbol{\rho}_i)$
  $\mathbf{z}_i^{\text{depth}} \sim \mathcal{N}(\boldsymbol{\mu}_i^{\text{depth}}, \boldsymbol{\sigma}_i^{\text{depth},2})$
  $\mathbf{z}_i^{\text{scale}} \sim \mathcal{N}(\boldsymbol{\mu}_i^{\text{scale}}, \boldsymbol{\sigma}_i^{\text{scale},2})$
  $\mathbf{z}_i^{\text{shift}} \sim \mathcal{N}(\boldsymbol{\mu}_i^{\text{shift}}, \boldsymbol{\sigma}_i^{\text{shift},2})$
  `/* rescale local shift to global shift */`
  $\mathbf{z}_i^{\text{where}} = [\sigma(\mathbf{z}_i^{\text{scale}}), \text{rescale}(\tanh(\mathbf{z}_i^{\text{shift}}))]$
  `/* Extract glimpses with a Spatial Transformer */`
  $\hat{\boldsymbol{x}}_i$ = ST($\boldsymbol{x}, \mathbf{z}_i^{\text{where}}$)
  $[\boldsymbol{\mu}_i^{\text{what}}, \boldsymbol{\sigma}_i^{\text{what}}]$ =GlimpseEncoder($\hat{\boldsymbol{x}}_i$)
  $\mathbf{z}_i^{\text{what}} \sim \mathcal{N}(\boldsymbol{\mu}_i^{\text{what}}, \boldsymbol{\sigma}_i^{\text{what},2})$
  `/* Foreground mask and appearance */`
  $[\boldsymbol{\alpha}_i^{\text{att}}, \boldsymbol{o}_i^{\text{att}}]$ = GlimpseDecoder($\mathbf{z}_i^{\text{what}}$)
  $\hat{\boldsymbol{\alpha}}_i^{\text{att}} = \boldsymbol{\alpha}_i^{\text{att}} \odot \mathbf{z}_i^{\text{pres}}$
  $\boldsymbol{y}_i^{\text{att}} = \hat{\boldsymbol{\alpha}}_i^{\text{att}} \odot \boldsymbol{o}_i^{\text{att}}$
end
`/* Compute weights for each component */`
$\boldsymbol{w} = \text{softmax}(-c \cdot \sigma(\mathbf{z}^{\text{depth}}) \odot \hat{\boldsymbol{\alpha}}^{\text{att}})$
`/* Compute global weighted mask and foreground appearance */`
$\alpha = \text{sum}(\boldsymbol{w} \odot \hat{\boldsymbol{\alpha}}^{\text{att}})$
$\mu^{\text{fg}} = \text{sum}(\boldsymbol{w} \odot \boldsymbol{y}^{\text{att}})$

D.2 TRAINING REGIME AND HYPERPARAMETERS

For all experiments we use an image size of 128×128 and a batch size of 12 to 16 depending on memory usage. We use the RMSProp (Tieleman & Hinton. (2012)) optimizer with a learning rate of 1×10^{-5} for the foreground module and Adam (Kingma & Ba (2014)) optimizer with a learning rate of 1×10^{-3} for the background module. We use gradient clipping with a maximum norm of 1.0. For Atari games, we find it beneficial to set α to be fixed for the first 1000-2000 steps, and vary the actual value and number of steps for different games. This allows both the foreground as well as the background module to learn in the early stage of training. For 3D Room datasets, we found that the quality of background segmentation would degrade if trained for too long, so we set the background learning rate to zero at appropriate steps.

Algorithm 2: Background Inference

Input: image $\boldsymbol{x}$, initial LSTM states $\boldsymbol{h}_0, \boldsymbol{c}_0$, initial dummy mask $\mathbf{z}_0^m$
Output: background masks π_k, appearance μ_k^{bg}, for $k = 1, \ldots, K$
$\boldsymbol{e}^{\text{img}}$ = ImageEncoderBg($\boldsymbol{x}$)
for $k \leftarrow 1$ **to** K **do**
  $\boldsymbol{h}_k, \boldsymbol{c}_k = \text{LSTM}([\mathbf{z}_{k-1}^m, \boldsymbol{e}^{\text{img}}], \boldsymbol{c}_{k-1}, \boldsymbol{h}_{k-1})$
  $[\boldsymbol{\mu}_k^m, \boldsymbol{\sigma}_k^m] = \text{PredictMask}(\boldsymbol{h}_k)$
  $\mathbf{z}^m \sim \mathcal{N}(\boldsymbol{\mu}_k^m, \boldsymbol{\sigma}_k^{m,2})$
  /* Actually decoded in parallel */
  $\hat{\pi}_k = \text{MaskDecoder}(\mathbf{z}_k^m)$
  /* Stick breaking process as described in GENESIS */
  $\pi_k = \text{SBP}(\hat{\pi}_{1:k})$
  $[\boldsymbol{\mu}_k^c, \boldsymbol{\sigma}_k^c] = \text{CompEncoder}([\pi_k, \boldsymbol{x}])$
  $\mathbf{z}^c \sim \mathcal{N}(\boldsymbol{\mu}_k^c, \boldsymbol{\sigma}_k^{c,2})$
  $\mu_k^{\text{bg}} = \text{CompDecoder}(\mathbf{z}_k^c)$
end

Algorithm 3: Background Generation

Input: initial LSTM states $\boldsymbol{h}_0, \boldsymbol{c}_0$, initial dummy mask $\mathbf{z}_0^m$
Output: background masks π_k, appearance μ_k^{bg}, for $k = 1, \ldots, K$
for $k \leftarrow 1$ **to** K **do**
  $\boldsymbol{h}_k, \boldsymbol{c}_k = \text{LSTM}(\mathbf{z}_{k-1}^m, \boldsymbol{c}_{k-1}, \boldsymbol{h}_{k-1})$
  $[\boldsymbol{\mu}_k^m, \boldsymbol{\sigma}_k^m] = \text{PredictMaskPrior}(\boldsymbol{h}_k)$
  $\mathbf{z}^m \sim \mathcal{N}(\boldsymbol{\mu}_k^m, \boldsymbol{\sigma}_k^{m,2})$
  /* Actually decoded in parallel */
  $\hat{\pi}_k = \text{MaskDecoder}(\mathbf{z}_k^m)$
  /* Stick breaking process as described in GENESIS */
  $\pi_k = \text{SBP}(\hat{\pi}_{1:k})$
  $[\boldsymbol{\mu}_k^c, \boldsymbol{\sigma}_k^c] = \text{PredictComp}(\mathbf{z}_k^m)$
  $\mathbf{z}_k^c \sim \mathcal{N}(\boldsymbol{\mu}_k^c, \boldsymbol{\sigma}_k^{c,2})$
  $\mu_k^{\text{bg}} = \text{CompDecoder}(\mathbf{z}_k^c)$
end

We list out our hyperparameters for 3D large dataset and joint training for 10 static Atari games below. Hyperparameters for other experiments are similar, but are finetuned for each dataset individually. In the tables below, $(m \to n) : (p \to q)$ denotes annealing the hyperparameter value from m to n, starting from step p and till step q.

3D Room Large

Name	Symbol	Value
$\mathbf{z}^{\text{pres}}$ prior prob	$\boldsymbol{\rho}$	1×10^{-10}
$\mathbf{z}^{\text{scale}}$ prior mean	$\boldsymbol{\mu}^{\text{scale}}$	$(-1.0 \to -2.0) : (10000 \to 20000)$
$\mathbf{z}^{\text{scale}}$ prior stdev	$\boldsymbol{\sigma}^{\text{scale}}$	0.1
$\mathbf{z}^{\text{shift}}$ prior	$\boldsymbol{\mu}^{\text{shift}}, \boldsymbol{\sigma}^{\text{shift}}$	$\mathcal{N}(\mathbf{0}, \boldsymbol{I})$
$\mathbf{z}^{\text{depth}}$ prior	$\boldsymbol{\mu}^{\text{depth}}, \boldsymbol{\sigma}^{\text{depth}}$	$\mathcal{N}(\mathbf{0}, \boldsymbol{I})$
$\mathbf{z}^{\text{what}}$ prior	$\boldsymbol{\mu}^{\text{what}}, \boldsymbol{\sigma}^{\text{what}}$	$\mathcal{N}(\mathbf{0}, \boldsymbol{I})$
foreground stdev	σ^{fg}	0.15
background stdev	σ^{bg}	0.15
component number	K	5
gumbel-softmax temperature	τ	2.0
#steps to fix α		N/A
fixed α value		N/A

Joint Training on 10 Atari Games

Name	Symbol	Value
$\mathbf{z}^{\text{pres}}$ prior prob	$\boldsymbol{\rho}$	0.01
$\mathbf{z}^{\text{scale}}$ prior mean	$\boldsymbol{\mu}^{\text{scale}}$	$(-2.0 \to -2.5) : (0 \to 20000)$
$\mathbf{z}^{\text{scale}}$ prior stdev	$\boldsymbol{\sigma}^{\text{scale}}$	0.1
$\mathbf{z}^{\text{shift}}$ prior	$\boldsymbol{\mu}^{\text{shift}}, \boldsymbol{\sigma}^{\text{shift}}$	$\mathcal{N}(\mathbf{0}, \boldsymbol{I})$
$\mathbf{z}^{\text{depth}}$ prior	$\boldsymbol{\mu}^{\text{depth}}, \boldsymbol{\sigma}^{\text{depth}}$	$\mathcal{N}(\mathbf{0}, \boldsymbol{I})$
$\mathbf{z}^{\text{what}}$ prior	$\boldsymbol{\mu}^{\text{what}}, \boldsymbol{\sigma}^{\text{what}}$	$\mathcal{N}(\mathbf{0}, \boldsymbol{I})$
foreground stdev	σ^{fg}	0.20
background stdev	σ^{bg}	0.10
component number	K	3
gumbel-softmax temperature	τ	2.5
#steps to fix α		2000
fixed α value		0.1

D.3 MODEL ARCHITECTURE

Here we describe the architecture of our 16×16 SPACE model. The model for 8×8 grid cells is the same but with a stride-2 convolution for the last layer of the image encoder.

All modules that output distribution parameters are implemented with either one single fully connected layer or convolution layer, with the appropriate output size. Image encoders are fully convolutional networks that output a feature map of shape $H \times W$, and the glimpse encoder comprises of convolutional layers followed by a final linear layer that computes the parameters of a Gaussian distribution. For the glimpse decoder of the foreground module and the mask decoder of the background module we use the sub-pixel convolution layer (Shi et al. (2016)). On the lines of GENESIS (Engelcke et al. (2019)) and IODINE (Greff et al. (2019)), we adopt Spatial Broadcast Network (Watters et al. (2019)) as the component decoder to decode z_k^c into background components.

For inference and generation of the background module, the dependence of $\mathbf{z}_k^m$ on $\mathbf{z}_{1:k-1}^m$ is implemented with LSTMs, with hidden sizes of 64. Dependence of $\mathbf{z}_k^c$ on $\mathbf{z}_k^m$ is implemented with a MLP with two hidden layers with 64 units per layer. We apply $\mathrm{softplus}$ when computing standard deviations for all Gaussian distributions, and apply $\mathrm{sigmoid}$ when computing reconstruction and masks. We use either Group Normalization (GN) (Wu & He (2018)) and CELU (Barron (2017)) or Batch Normalization (BN) (Ioffe & Szegedy (2015)) and ELU (Clevert et al. (2016)) depending on the module type.

The rest of the architecture details are described below. In the following tables, $\mathrm{ConvSub(n)}$ denotes a sub-pixel convolution layer implemented as a stride-1 convolution and a PyTorch $\mathrm{PixelShuffle}(n)$ layer, and $\mathrm{GN}(n)$ denotes Group Normalization with n groups.

Name	Value	Comment
$\mathbf{z}^{\text{depth}}$ dim	1	
$\mathbf{z}^{\text{pres}}$ dim	1	
$\mathbf{z}^{\text{scale}}$ dim	2	for x and y axis
$\mathbf{z}^{\text{shift}}$ dim	2	for x and y axis
$\mathbf{z}^{\text{what}}$ dim	32	
$\mathbf{z}^{m}$ dim	32	
$\mathbf{z}^{c}$ dim	32	
glimpse shape	$(32, 32)$	for $\boldsymbol{o}^{\text{att}}, \boldsymbol{\alpha}^{\text{att}}$
image shape	$(128, 128)$	

Background Image Encoder

Layer	Size/Ch.	Stride	Norm./Act.
Input	3		
Conv 3×3	64	2	BN/ELU
Conv 3×3	64	2	BN/ELU
Conv 3×3	64	2	BN/ELU
Conv 3×3	64	2	BN/ELU
Flatten			
Linear	64		ELU

Glimpse Encoder

Layer	Size/Ch.	Stride	Norm./Act.
Input	3		
Conv 3×3	16	1	GN(4)/CELU
Conv 4×4	32	2	GN(8)/CELU
Conv 3×3	32	1	GN(4)/CELU
Conv 4×4	64	1	GN(8)/CELU
Conv 4×4	128	2	GN(8)/CELU
Conv 4×4	256	1	GN(16)/CELU
Linear	32 + 32		

Glimpse Decoder

Layer	Size/Ch.	Stride	Norm./Act.
Input	32		
Conv 1×1	256	1	GN(16)/CELU
ConvSub(2)	128	1	GN(16)/CELU
Conv 3×3	128	1	GN(16)/CELU
ConvSub(2)	128	1	GN(16)/CELU
Conv 3×3	128	1	GN(16)/CELU
ConvSub(2)	64	1	GN(8)/CELU
Conv 3×3	64	1	GN(8)/CELU
ConvSub(2)	32	1	GN(8)/CELU
Conv 3×3	32	3	GN(8)/CELU
ConvSub(2)	16	1	GN(4)/CELU
Conv 3×3	16	1	GN(4)/CELU

Background Image Encoder

Layer	Size/Ch.	Stride	Norm./Act.
Input	3		
Conv 3×3	64	2	BN/ELU
Conv 3×3	64	2	BN/ELU
Conv 3×3	64	2	BN/ELU
Conv 3×3	64	2	BN/ELU
Flatten			
Linear	64		ELU

Mask Decoder

Layer	Size/Ch.	Stride	Norm./Act.
Input	32		
Conv 1×1	256	1	GN(16)/CELU
ConvSub(4)	256	1	GN(16)/CELU
Conv 3×3	256	1	GN(16)/CELU
ConvSub(2)	128	1	GN(16)/CELU
Conv 3×3	128	1	GN(16)/CELU
ConvSub(4)	64	1	GN(8)/CELU
Conv 3×3	64	1	GN(8)/CELU
ConvSub(4)	16	1	GN(4)/CELU
Conv 3×3	16	1	GN(4)/CELU
Conv 3×3	1	1	

Component Encoder

Layer	Size/Ch.	Stride	Norm./Act.
Input	3+1 (RGB+mask)		
Conv 3×3	32	2	BN/ELU
Conv 3×3	32	2	BN/ELU
Conv 3×3	64	2	BN/ELU
Conv 3×3	64	2	BN/ELU
Flatten			
Linear	32+32		

Component Decoder

Layer	Size/Ch.	Stride	Norm./Act.
Input	32 (1d)		
Spatial Broadcast	32+2 (3d)		
Conv 3×3	32	1	BN/ELU
Conv 3×3	32	1	BN/ELU
Conv 3×3	32	1	BN/ELU
Conv 3×3	3	1	

D.4 BASELINE SPAIR

Here we give out the details of the background encoder and decoder in training of SPAIR (both full image as well as patch-wise training). The image encoder is same as that of SPACE with the only difference that the inferred latents are conditioned on previous cells' latents as described in Section 2.2.

SPAIR Background Encoder - Full Image training

Layer	Size/Ch.	Norm./Act.
Input	$3 \times 128 \times 128$	
Linear	256	GN(16)/CELU
Linear	128	GN(16)/CELU
Linear	32	

SPAIR Background Decoder

Layer	Size/Ch.	Stride	Norm./Act.
Input	16		
Conv 3×3	256	1	GN(16)/CELU
ConvSub 3×3	128	1	GN(16)/CELU
Conv 3×3	128	1	GN(16)/CELU
ConvSub 3×3	64	1	GN(8)/CELU
Conv 3×3	64	1	GN(8)/CELU
ConvSub 3×3	16	1	GN(4)/CELU
Conv 3×3	16	1	GN(4)/CELU
Conv 3×3	16	1	GN(4)/CELU
Conv 3×3	3	1	

SPAIR Background Encoder For Patch Training

Layer	Size/Ch.	Stride	Norm./Act.
Input	3		
Conv 2×2	16	2	GN(4)/CELU
Conv 2×2	32	2	GN(8)/CELU
Conv 2×2	64	2	GN(8)/CELU
Conv 2×2	128	2	GN(16)/CELU
Conv 2×2	32	2	GN(4)/CELU

SPAIR Background Decoder For Patch Training

Layer	Size/Ch.	Stride	Norm./Act.
Input	16		
Conv 1×1	256	1	GN(16)/CELU
Conv 1×1	2048	1	
ConvSub(4)	128	1	GN(16)/CELU
Conv 3×3	128	1	GN(16)/CELU
Conv 1×1	256	1	
ConvSub(2)	64	1	GN(8)/CELU
Conv 3×3	64	1	GN(8)/CELU
Conv 1×1	256	1	
ConvSub(4)	16	1	GN(4)/CELU
Conv 3×3	16	1	GN(4)/CELU
Conv 3×3	16	1	GN(4)/CELU
Conv 3×3	3	1	

E DATASET DETAILS

Atari. For each game, we sample 60,000 random images from a pretrained agent (Wu et al., 2016). We split the images into 50,000 for the training set, 5,000 for the validation set, and 5,000 for the testing set. Each image is preprocessed into a size of 128×128 pixels with BGR color channels. We present the results for the following games: Space Invaders, Air Raid, River Raid, Montezuma's Revenge.

We also train our model on a dataset of 10 games jointly, where we have 8,000 training images, 1,000 validation images, and 1,000 testing images for each game. We use the following games: Asterix,

Atlantis, Carnival, Double Dunk, Kangaroo, Montezuma Revenge, Pacman, Pooyan, Qbert, Space Invaders.

Room 3D. We use MuJoCo (Todorov et al., 2012) to generate this dataset. Each image consists of a walled enclosure with a random number of objects on the floor. The possible objects are randomly sized spheres, cubes, and cylinders. The small 3D-Room dataset has 4-8 objects and the large 3D-Room dataset has 18-24 objects. The color of the objects are randomly chosen from 8 different colors and the colors of the background (wall, ground, sky) are chosen randomly from 5 different colors. The angle of the camera is also selected randomly. We use a training set of 63,000 images, a validation set of 7,000 images, and a test set of 7,000 images. We use a 2-D projection from the camera to determine the ground truth bounding boxes of the objects so that we can report the average precision of the different models.