A. Equivariance and invariance properties

In this section, we verify that the implementations of the marginal distributions $q_\phi(z_k^{1:L} \mid x, z_k^0)$ and $p_\theta(z_k^{1:L})$, the decoder network, and the refinement network all achieve the desired equivariance property. Additionally, we demonstrate that EfficientMORL is also *invariant* to permutations applied to the order of the N inputs $x_n \in \mathbb{R}^D, n = 1, \ldots, N$, where $N = HW$ and H, W are the dimensions of the image. The verification itself is mostly straightforward, as we mainly rely on weight-tying in the neural network architectures to symmetrically process each element of any set-structured inputs.

Definition 1 (Permutation invariance) *Given $X \in \mathbb{R}^{N \times D_1}$, $f : \mathbb{R}^{N \times D_1} \to \mathbb{R}^{N \times D_2}$, and any $N \times N$ permutation matrix Π, we say that f is permutation invariant if*

$$f(\Pi X) = f(X).$$

Definition 2 (Permutation equivariance) *Given $X \in \mathbb{R}^{N \times D_1}$, $f : \mathbb{R}^{N \times D_1} \to \mathbb{R}^{N \times D_2}$, and any $N \times N$ permutation matrix Π, we say that f is permutation equivariant if*

$$f(\Pi X) = \Pi f(X).$$

A.1. Equivariance

Bottom-up posterior Recall that there is an initial *symmetry-breaking* step of sampling K times from $\mathcal{N}(\mu^0, (\sigma^0 I)^2)$ and consider the first layer of the marginal, $q_\phi(\mathbf{z}^1 \mid x, \mathbf{z}^0)$, which is conditioned on samples $\mathbf{z}^0$. First, the samples are used to predict set-structured features $\mathbf{\Theta} \in \mathbb{R}^{K \times D}$ using scaled dot-product set attention (Locatello et al., 2020). By appealing to the proof in Section D.2 of Locatello et al. (2020), all operations within the scaled dot-product set attention (Lines 8-11 of Algorithm 1) preserve permutation equivariance of the features $\mathbf{\Theta}$. Following this, we concatenate $\mathbf{\Theta}$ with itself along the D-dimension and pass it to the DualGRU. The DualGRU uses weight-tying across K to symmetrically process inputs $[\mathbf{\Theta}, \mathbf{\Theta}]$ and previous state $[\boldsymbol{\mu}^0, \boldsymbol{\sigma}^0]$. Next, the two MLPs also use weight-tying across K to output K means and variances. Finally, we sample $\mathbf{z}^1$ from $q_\phi(\mathbf{z}^1 \mid x, \mathbf{z}^0)$. We only sample once from each of the K marginals to generate the input with which to condition $q_\phi(\mathbf{z}^2 \mid x, \mathbf{z}^1)$. Each stochastic layer thereafter is computed identically, and therefore the marginals of the bottom-up posterior are equivariant with respect to permutations applied to their ordering.

The **hierarchical prior** preserves the symmetry of its marginals since it consists of only feed-forward layers applied individually to each element of a sample $\mathbf{z}$ via weight-tying across K. Both considered decoders are permutation invariant since they aggregate the K masks and RGB components with a summation over K, which is a permutation invariant operation (see Section F for decoder details). This ensures that the reconstructed/generated image does not depend on the ordering of the posterior/prior marginals.

The **refinement network** simply consists of feed-forward and recurrent layers with weights tied across K that are again applied individually to each element of the set of posterior parameters $\boldsymbol{\lambda}$.

A.2. Invariance

Due to the use of Slot Attention's scaled dot-product set attention mechanism to process the inputs in each stochastic layer of the posterior, EfficientMORL inherits the property that it is invariant to permutations applied to the order of the $x_1, \ldots, x_N$, $x_n \in \mathbb{R}^D$. An inspection of Line 11 of Algorithm 1 verifies that the N-element input x is aggregated with a permutation invariant summation over N.

B. EfficientMORL vs. GENESIS

In this section, we provide arguments as to why models that learn *unordered* (i.e., permutation equivariant) latents might be preferred over those that learn *ordered* latents. Many of the discussion points here are adapted from Appendix A.3 of Greff et al. (2019). We also empirically show one key advantage using a variant of the Multi-dSprites dataset.

GENESIS (Engelcke et al., 2020) is a generative model for multi-object representation learning that, like MONet, uses sequential attention over the image to discover objects. Its design is motivated by efficiency and the task of unconditional scene generation. It uses an autoregressive prior and autoregressive posterior to induce an ordering on its K object-centric latent variables which enables fast inference and generation. The authors demonstrate that the autoregressive prior also facilitates coherent generation of multi-object scenes.

On biased decomposition However, GENESIS is not equivariant to permutations applied to the K latents. That is, GENESIS infers a *ordered* scene decomposition. Due to the autoregressive prior and posterior, the pixels assigned to the i^{th} latent depends on latents $< i$, which biases the model towards specific decomposition strategies that incorporate global scene information (Figure 11). GENESIS uses a deterministic stick-breaking process to implement sequential attention, which encourages it to learn fixed strategies such as always placing the background in the first or last component. Additionally, this can occasionally lead the model towards learning poor strategies that are overly dependent on color.

On ambiguity Only updating each object-centric latent once makes GENESIS potentially less capable of handling ambiguous and complex cases. EfficientMORL assigns pixels to latents across multiple iterations, which can facilitate disambiguating parts of a scene that are difficult to parse.

On multi-stability GENESIS also does not share the multi-stability property enjoyed by EfficientMORL and IODINE. That is, running inference multiple times with EfficientMORL can produce different scene decompositions, particularly for ambiguous cases, due to randomness in the iterative assignment (see Figure 10 of Greff et al. (2019)).

On sequential extensions Finally, we highlight that IODINE has been demonstrated to be straightforward to extend to sequences, and we expect EfficientMORL could be similarly applied to sequences since it shares with IODINE the use of adaptive top-down refinement. This has been accomplished in a few different ways. One way is to simply pass a new image at each step of iterative inference (Greff et al., 2019). Or, initialize the posterior at each time step with the posterior from the previous time step before performing I refinement steps (Li et al., 2020). In the model-based reinforcement learning setting, a latent dynamics model has been used to predict the initial posterior at each time step using the posterior from the previous time step, followed by I refinement steps (Veerapaneni et al., 2019). Ordered models have been used for sequential data by adding an extra matching step to align the object latents over time, resembling unsupervised multi-object tracking (Watters et al., 2019a).

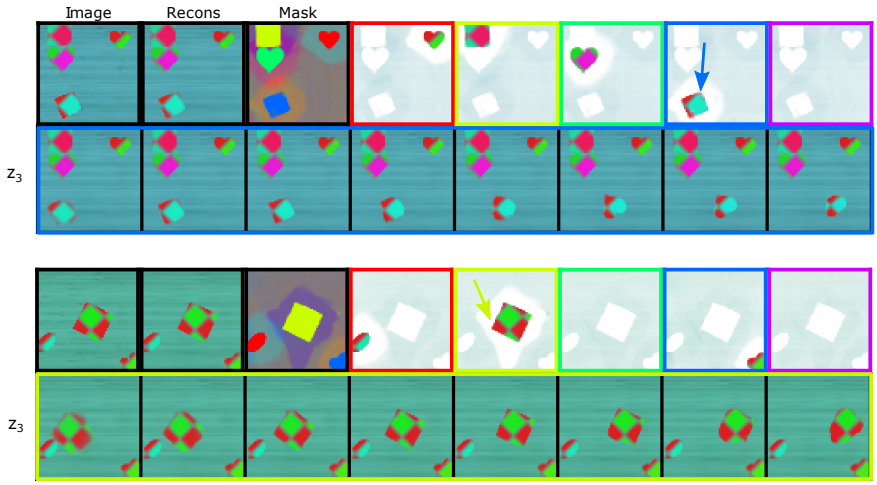
GENESIS-v2 We note that the authors of GENESIS recently released GENESIS-v2 (Engelcke et al., 2021), which appeared after the initial submission of this work. We comment briefly on it here. GENESIS-v2 appears to remove GENESIS's autoregressive posterior and adds a non-parametric *randomized* clustering algorithm to obtain an ordered set of of attention masks. These attention masks are then mapped in parallel to K latents (GENESIS-v2 is able to adjust K on-the-fly via early-stopping of the clustering algorithm). Due to the random initialization of the clustering algorithm, the model cannot learn a fixed sequential decomposition strategy. We highlight that GENESIS-v2 keeps GENESIS's autoregressive prior and that GENESIS-v2 is not equivariant with respect to permutations applied to the set of cluster seeds used to obtain attention masks. Like GENESIS, GENESIS-v2 uses a stick-breaking process such that the last (K^{th}) component is assigned the remaining scope; this suggests that global information may still be leaking into the representations (see Figure 11). The authors note that on some training runs, the model learns to always place the background into the last component.

B.1. Multi-colored and textured Multi-dSprites

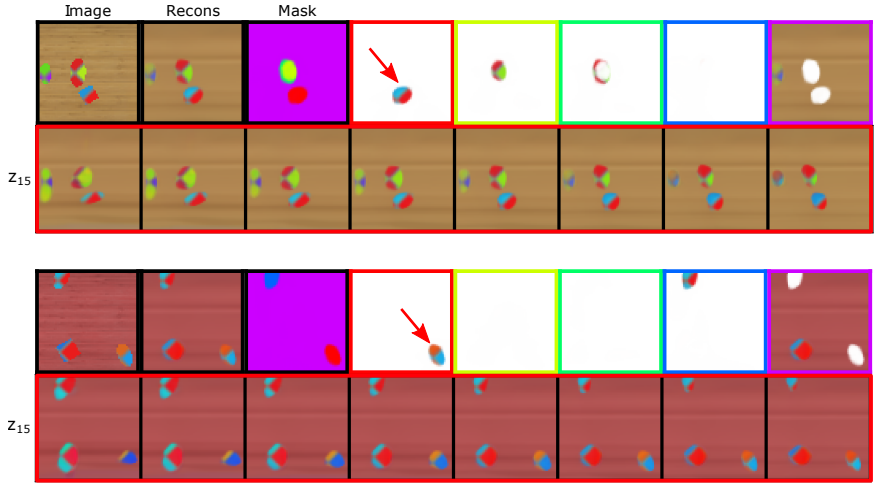
We illustrate how global scene-level information leaks into the object-centric representations in GENESIS in Figure 11. For this, we created a variant of the Multi-dSprites dataset that has as background a simple uniform texture and foreground objects that are multi-colored with a distinctive pattern. Models that successfully decompose these images cannot do so by only grouping using color cues.

We invested considerable effort into tuning GENESIS to solve this dataset. What worked was using the Gaussian image likelihood with $\sigma = 0.1$ (EfficientMORL uses the same) instead of the Gaussian mixture model (see Section F; we tried various values for σ for the mixture model likelihood as well) and decreasing GENESIS's GECO parameter update rate to avoid increasing the KL too much early on. The Gaussian mixture model likelihood consistently caused GENESIS to learn to segment the images purely based on color. We trained both GENESIS and GENESIS-S models, where GENESIS-S uses a single set of latents instead of splitting them into masks and components $[\mathbf{z}^m, \mathbf{z}^c]$. GENESIS converged significantly faster than GENESIS-S and obtained the best results, whereas the latter did not finish converging after 500K steps.

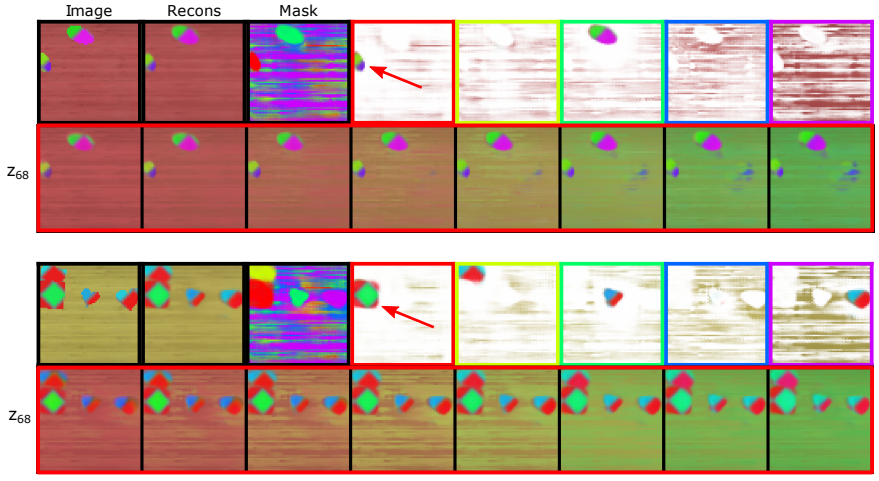
As shown in Figure 11, the object-centric representations learned by EfficientMORL are able to be independently manipulated. GENESIS's autoregressive posterior causes it to incorporate global scene-level information into the representations. We show that manipulating a single dimension of a single object latent can change the entire scene representation.



(a) EfficientMORL

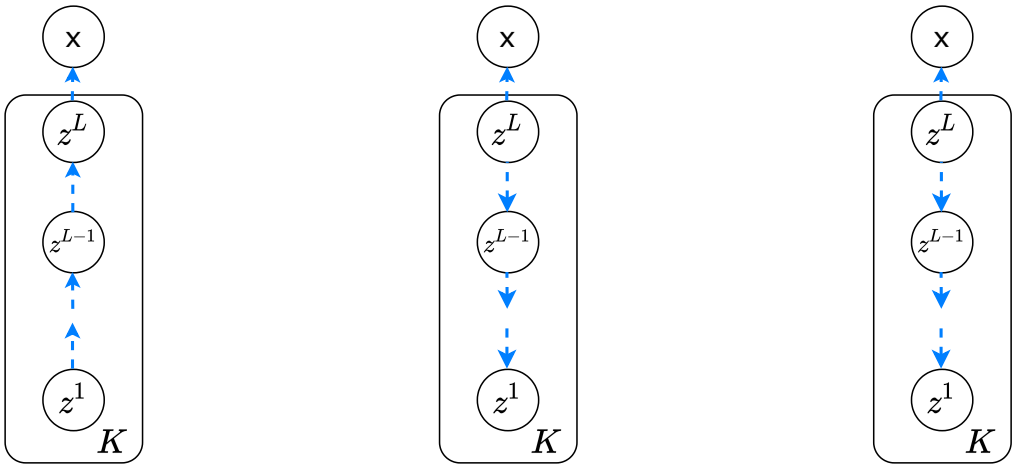


(b) GENESIS-S



(c) GENESIS

Figure 11. **The *ordered* scene decomposition learned by GENESIS leaks global scene information into the object representations**. The second row of each sub-figure shows the reconstructed images generated by varying a single active latent dimension (chosen arbitrarily) of a single component (indicated by color). We show that for GENESIS-S and GENESIS, which have autoregressive posteriors, *all* objects are affected when the first component is manipulated.



(a) Uses $p_\theta(\mathbf{z}^L|\mathbf{z}^{L-1})$ in refinement loss KL (b) Uses $p(\mathbf{z}^L)$ in refinement loss KL (c) Uses $p_\theta(\mathbf{z}^1 \mid \mathbf{z}^2)$ in refinement loss KL

Figure 12. The three variants of the hierarchical prior we explore. a) **EfficientMORL w/ bottom-up prior**, where $p(\mathbf{z}^1)$ is a standard Gaussian and the prior gets more expressive in higher layers, b) **EfficientMORL w/ reversed prior** where $p(\mathbf{z}^L)$, the corresponding prior for the top layer posterior, is a standard Gaussian, and the prior gets more expressive in lower layers, and c) **EfficientMORL w/ reversed prior++** where $p(\mathbf{z}^L)$ is also standard Gaussian but the posteriors at lower layers of the hierarchy are encouraged to match the top layer posterior via the modified refinement KL term.

C. Limitations

For a given image, EfficientMORL requires that K is chosen *a priori*, although any value of K can be chosen. A way to adapt K automatically based on the input could be useful to increase the number of latent variables dynamically if needed.

Related to this, EfficientMORL does not have any explicit mechanism for attending to a subset of objects in a scene. This could prove useful for handling scenes containing many objects and scenes where what constitutes the foreground and background is ambiguous.

We do not expect EfficientMORL to be able to generate *globally coherent* scenes due to the independence assumption in the prior. Identifying how to achieve coherent generation without sacrificing permutation equivariance is an open problem.

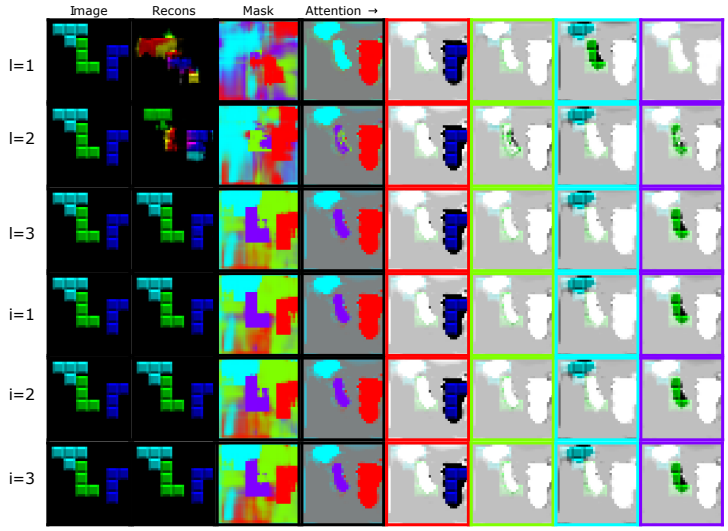
EfficientMORL also inherits similar limitations to IODINE related to handling images containing heavily textured *backgrounds* (see Section 5 of Greff et al. (2019)), as well as large-scale datasets that do not consist of images that represent a dense sampling of the data generating latent factors, which is important for unsupervised learning. EfficientMORL *can* handle textured and multi-colored *foreground* objects assuming a simplistic background (Figure 11a).

D. Additional ablation studies

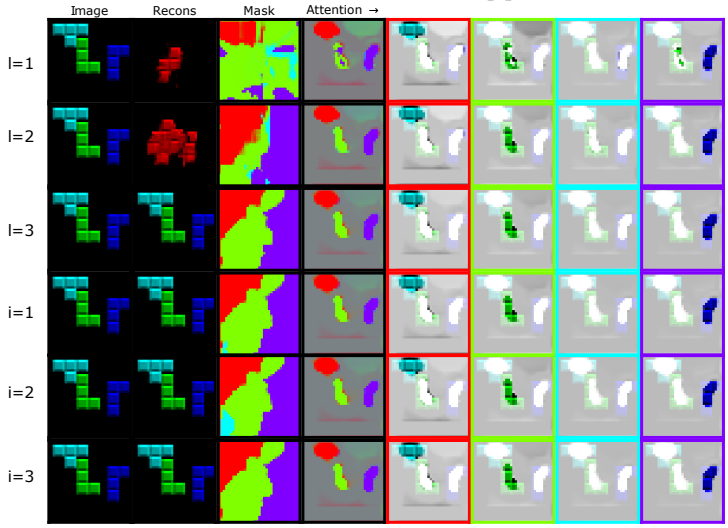
Table 2. Ablation study results. All models use reversed prior++ (Figure 12c) unless specified.

	Env	ARI	MSE ($\times 10^{-4}$)	KL
EfficientMORL w/out DualGRU	CLEVR6	79.7 ± 22.3	$\mathbf{8.1 \pm 1.7}$	1357.4 ± 321.6
EfficientMORL w/out GECO	CLEVR6	79.9 ± 9.9	9.9 ± 2.5	$\mathbf{177.5 \pm 14.7}$
EfficientMORL	CLEVR6	$\mathbf{82.4 \pm 7.9}$	$\mathbf{8.3 \pm 1.1}$	692.7 ± 281.4
EfficientMORL w/out GECO	Tetrominoes	82.2 ± 17.7	68.8 ± 42.2	$\mathbf{45.9 \pm 9.3}$
EfficientMORL w/ bottom-up prior	Tetrominoes	85.7 ± 11.0	17.1 ± 14.6	-
EfficientMORL w/ reversed prior	Tetrominoes	86.9 ± 16.2	22.1 ± 21.4	-
EfficientMORL	Tetrominoes	$\mathbf{97.9 \pm 2.4}$	$\mathbf{7.0 \pm 6.0}$	98.5 ± 21.4

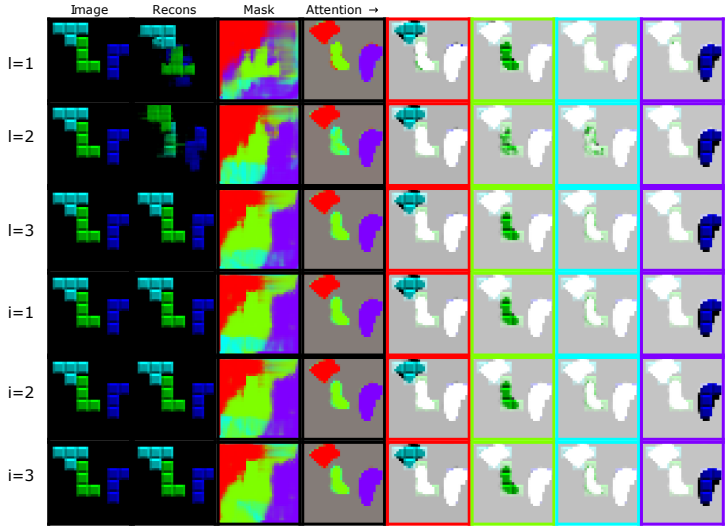
Results are in Table 2. The CLEVR6 results are computed across 10 random seeds for $50k$ steps. At this many steps the model has begun to converge and large differences in final performance can be easily observed. Each of the Tetrominoes results are computed on a validation set of 320 images across 5 random seeds.



(a) EfficientMORL w/ bottom-up prior



(b) EfficientMORL w/ reversed prior



(c) EfficientMORL w/ reversed prior++

Figure 13. Samples from the intermediate posteriors and attention from the ablated hierarchical priors. The reconstructions in the first two rows demonstrate the efficacy of the reversed prior++ at preventing the intermediate posteriors from collapsing.

Hierarchical prior The three hierarchical prior variants are shown in Figure 12. We present the ARI and MSE scores for Tetrominoes in Table 2, where the reversed prior++ consistently achieves the best results. By inspecting samples from the intermediate posteriors of each model (Figure 13, second column, first two rows), we can see that the bottom-up prior and reversed prior learn to leave the posterior at layers $1-2$ close to the initial uncorrelated Gaussian $q(\mathbf{z}^0)$, with the reversed prior exhibiting this most strongly. This limits the expressiveness of the posterior at layer L and on certain runs these models would converge to poor local minima—hence the high standard deviation in ARI and MSE. On the other hand, the reversed prior++ keeps the posterior at layers $1-2$ close to the layer L posterior. The large gap in performance suggests that although Tetrominoes appears to be easy to solve, it may be deceptively challenging as it seems to require fitting a complex, non-Gaussian posterior. Although we do not show the results here, we did try training EfficientMORL w/ reversed prior on CLEVR6 and found that the less expressive posterior was sufficient to achieve comparable results to the reversed prior++.

DualGRU We replaced the DualGRU with a standard GRU of hidden dimension $2D$. The DualGRU regularizes training due to improved parameter efficiency from the block-diagonal design. With a single standard GRU, the model tends to achieve a high KL. Some of these models converged to poor local minima, which is reflected by the lower ARI score.

GECO We trained EfficientMORL without GECO to examine the severity of posterior collapse on CLEVR6 and Tetrominoes (Table 2). We found that posterior collapse did not affect the model on Multi-dSprites and therefore did not use GECO for this environment. Without GECO, the KL is low due to the early collapse of many of the posterior dimensions. For most runs, this leads to poor quality reconstructions that achieve higher MSE and lower ARI scores.

Varying refinement steps We show the test ARI, MSE, and KL for various values of I using the reversed prior++ and bottom-up prior Tetrominoes models (Figure 14). Recall that we held I fixed at three when training on Tetrominoes. The ARI/MSE shows a slight drop when testing with zero refine steps, whereas the KL is noticeably larger at $I = 0$. A single low-dimensional refinement step does not incur a large increase in extra computation (Figure 10) if the low KL posterior is desired at test time.

E. Additional results

E.1. Object decomposition

An extra visualization of a sample from the intermediate and final posteriors for a single scene, showing reconstruction, masks, and attention, is provided in Figure 15.[2]

We also show extra qualitative scene decompositions for CLEVR6 (Figure 16), Multi-dSprites (Figure 17), and Tetrominoes (Figure 18). Each decomposition is sampled from the final posterior after refinement.

E.2. Disentanglement

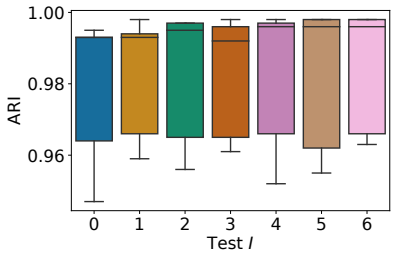
Visualizations We include additional visualizations of the disentanglement learned by our model. We randomly select one object component and multiple latent dimensions to vary. To determine reasonable ranges to linearly interpolate between for each latent dimension, we computed the maximum and minimum values per latent dimension across 100 images. Figure 19 shows one model trained on CLEVR6 and Figure 21 shows one trained on Multi-dSprites. In Figure 20, we visualize one of the ablated Tetrominoes models which did not use GECO and successfully learned to decompose and reconstruct the images, achieving a low final KL.

We emphasize that we did not tune the ELBO to emphasize disentanglement over reconstruction quality in this paper. If desired, EfficientMORL can be trained to achieve a more highly disentangled latent representation, for example by modulating the KL term with a β hyperparameter.

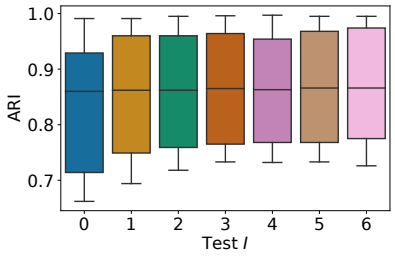
DCI metric We used the `disentanglement_lib` (Locatello et al., 2019) to compute DCI scores (Eastwood & Williams, 2018). DCI measures three quantities, *disentanglement*, *completeness*, and *informativeness* and involves training a regressor (continuous factors) or classifier (discrete factors) to predict the ground truth factors given the extracted representation from the data. We repeat the DCI scores here in Table 3.

We take the following steps to compute DCI for the considered multi-object setting for EfficientMORL and Slot Attention on CLEVR6. For each test image, we extracted $K \times D$ latent codes. For EfficientMORL, these are the means of the final

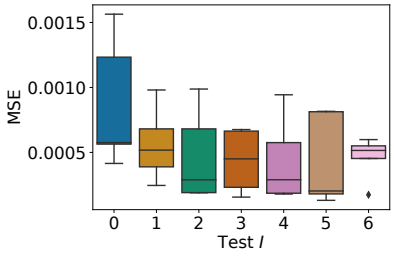
[2]Visualizations are made using a modified script provided by Greff et al. (2019).



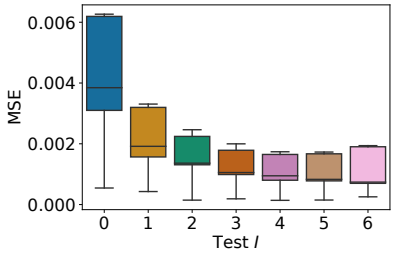
(a) Tetrominoes (w/ reversed prior++) ARI vs. test I.



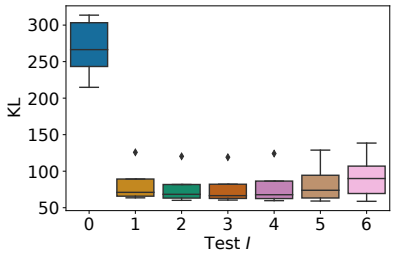
(b) Tetrominoes (w/ bottom-up prior) ARI vs. test I.



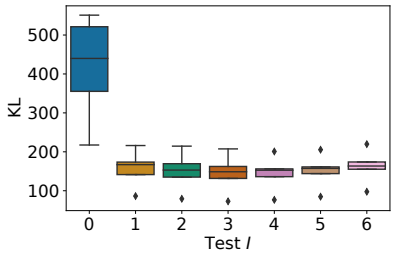
(c) Tetrominoes (w/ reversed prior++) MSE vs. test I



(d) Tetrominoes (w/ bottom-up prior) MSE vs. test I



(e) Tetrominoes (w/ reversed prior++) KL vs. test I



(f) Tetrominoes (w/ bottom-up prior) KL vs. test I

Figure 14. a,c) Training with $I = 3$ fixed and testing with $I = 0$ results in a fractional drop in ARI/MSE with the reversed prior++. A larger gap in KL is again observed between zero and one test refine step, for both the reversed prior++ (e) and bottom-up prior (f). Performance is maintained when increasing I up to 6, as the refinement GRU has learned to exploit sequential information.

posterior distribution. For Slot Attention, it is simply the output slots. We also decode the K masks. Given the ground truth object segmentation, we compute the linear assignment using IOU as the cost function to find the best matching of ground truth object masks to the K predicted object masks.

Finally, we concatenate all pairs of ground truth latent factors and predicted latent codes for all images into a dataset using a random 50/50 train and test split. We use the `disentanglement_lib` to compute the DCI scores, which uses gradient boosted trees. A separate predictor is trained per factor. For CLEVR6, the factors are {x, y, z, rotation, size, material, shape, color} where the first four are continuous values and the last four are discrete values.

E.3. Efficiency

Trainable parameters IODINE has approximately $2.75M$ parameters and EfficientMORL has approximately $666K$ parameters, a 75% decrease.

Timing metric We computed the forward and backward pass timing for the considered models on the same hardware, using 1 RTX 2080 Ti GPU with mini-batch size of 4. We computed a running average across $10K$ passes after a burn in period of about five minutes. We show results for multiple images dimensions in Figure 22.

Refinement curriculum As stated earlier, we used two approaches in our experiments: fixing $I = 3$ throughout training (Tetrominoes), or decreasing I from three to one after $100K$ training steps (CLEVR6 and Multi-dSprites). When applying

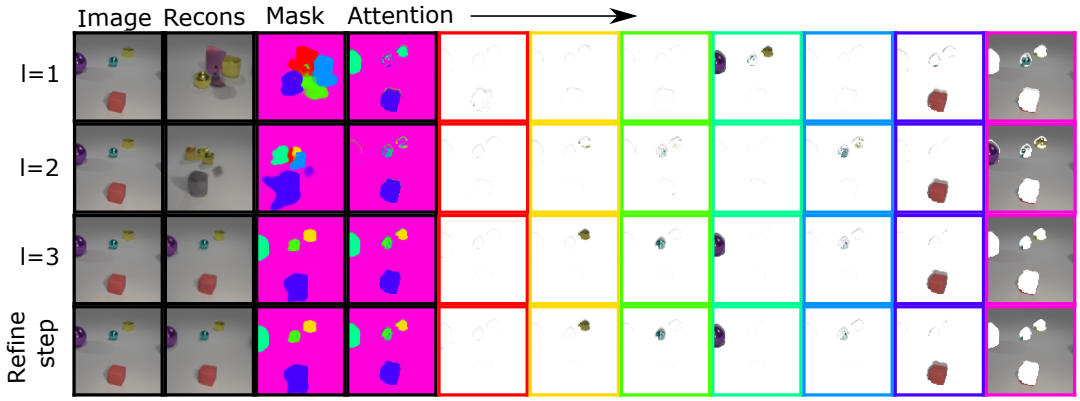


Figure 15. Visualization of reconstructions, masks, and attention from a single sample from the intermediate and final posteriors on CLEVR6.

Table 3. DCI on CLEVR6 (mean ± std dev across 5 runs). Higher is better.

	Disentanglement	Completeness	Informativeness
Slot Attention	0.46 ± 0.01	0.38 ± 0.02	0.34 ± 0.01
EfficientMORL	$\mathbf{0.63} \pm 0.04$	$\mathbf{0.63} \pm 0.06$	$\mathbf{0.46} \pm 0.01$

EfficientMORL to more challenging datasets for which convergence may take a long time, we recommend trying to simply decrement I by one following a reasonable schedule (e.g., every $100K$ steps). We directly decreased I from three to one since EfficientMORL converges relatively quickly on all three of the Multi-object Benchmark datasets.

F. Implementation

Code and data are available at `https://github.com/pemami4911/EfficientMORL`

F.1. Architecture details and hyperparameters

DualGRU The HVAE encoder has two separate GRUs to predict the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}^2$ for each bottom-up inference layer's posterior. This reduces the number of parameters in the encoder which regularizes the model and eases learning; see Section D for an ablation study on its use.

To parallelize the computation of the two GRUs we fuse their weights to create a single GRU—the *DualGRU*—that has sparse block-diagonal weight matrices. Specifically, the D-dim output of the scaled dot-product attention set attention is concatenated with itself $[\boldsymbol{\Theta}; \boldsymbol{\Theta}] \in \mathbb{R}^{K \times 2D}$ and then passed as input to the DualGRU. As an example, the DualGRU multiplies the input with the following:

$$\mathbf{W}_{ih} = \begin{bmatrix} \mathbf{W}_{ih}^1 & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_{ih}^2 \end{bmatrix}, \tag{9}$$

where $\mathbf{W}_{ih}^1 \in \mathbb{R}^{2D \times D}$ and $\mathbf{W}_{ih}^2 \in \mathbb{R}^{2D \times D}$. The shape of $\mathbf{W}_{ih}$ is $4D \times 2D$; internally, the DualGRU splits the $K \times 4D$ output of its application to the $K \times 2D$ input into 4 D-dim activations for the two reset and update gates. All other weight matrices $\mathbf{W}_{hh}$, $\mathbf{W}_{in}$, $\mathbf{W}_{hn}$ are similarly defined as block matrices.

In practice, we predict the variance for each Gaussian using the `softplus` (SP) operation. In detail, let $\mathbf{o}$ be a $K \times D$ output of a linear layer. Then,

$$\mathbf{o}^{'} = \texttt{min}(\mathbf{o}, 80 * \mathbf{1}) \tag{10}$$

$$\boldsymbol{\sigma}^2 = \log(1 + \texttt{exp}(\mathbf{o}') + 1e - 5) \tag{11}$$

Image likelihoods Let $P = 3HW$ be the number of pixels, x_p be the random variable for pixel p's value, $y_{k,p} \in \mathbb{R}^3$ be the RGB value for the p^{th} pixel from the $k \in K^{\text{th}}$ component, and $\pi_{k,p} \in [0, 1]$ be the corresponding assignment. The

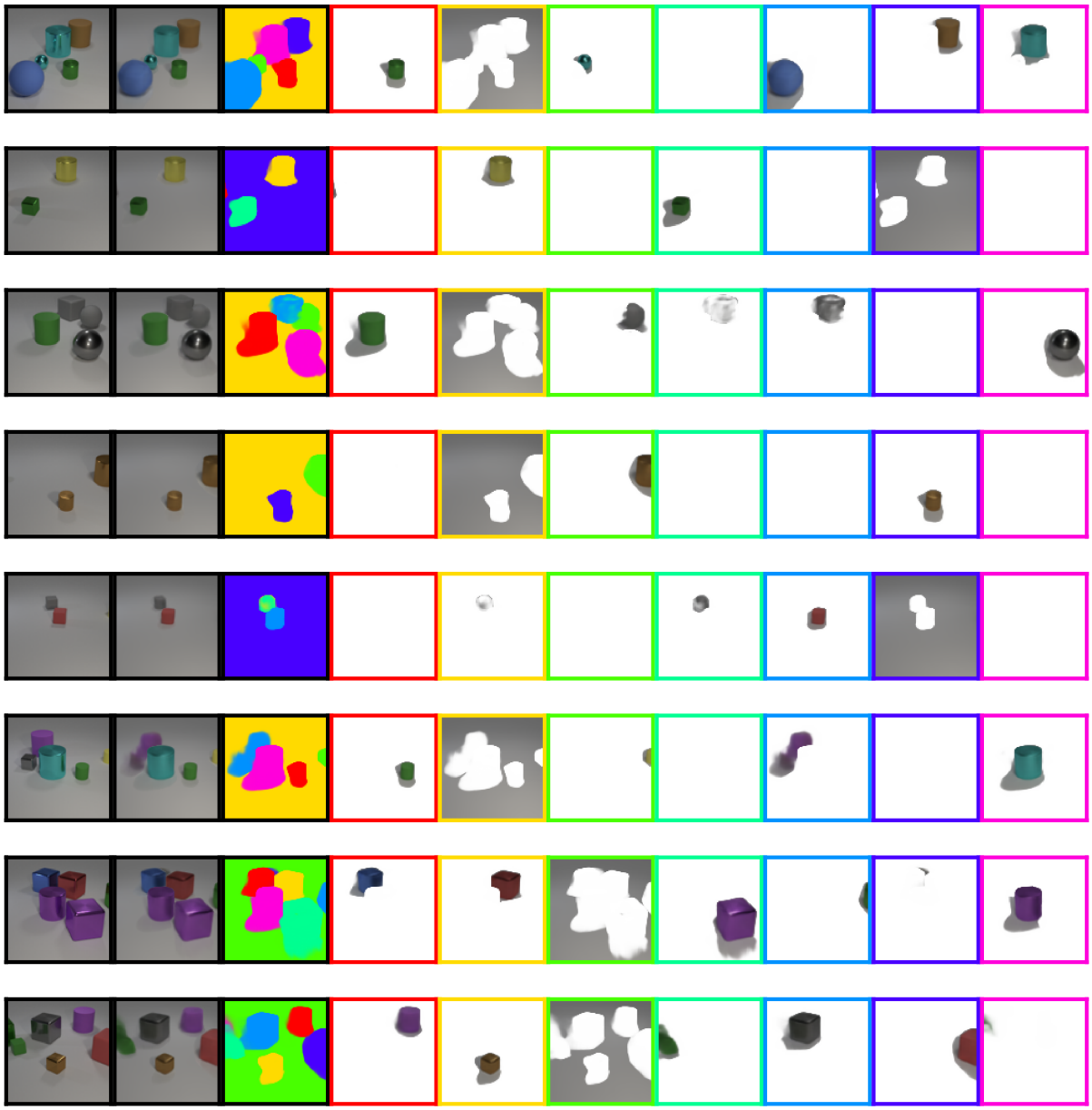


Figure 16. Additional CLEVR6 scene decompositions with $K = 7$, $L = 3$, and $I = 1$. First column is the ground truth image, second column is the reconstruction, third column is the mask, and the remaining columns are the masked components. One example of a failure case is shown in row 6 where the model joins a purple cylinder and an adjacent silver cube.

Figure 17. Additional Multi-dSprites scene decompositions with $K = 6$, $L = 3$, and $I = 1$. First column is the ground truth image, second column is the reconstruction, third column is the mask, and the remaining columns are the masked components.

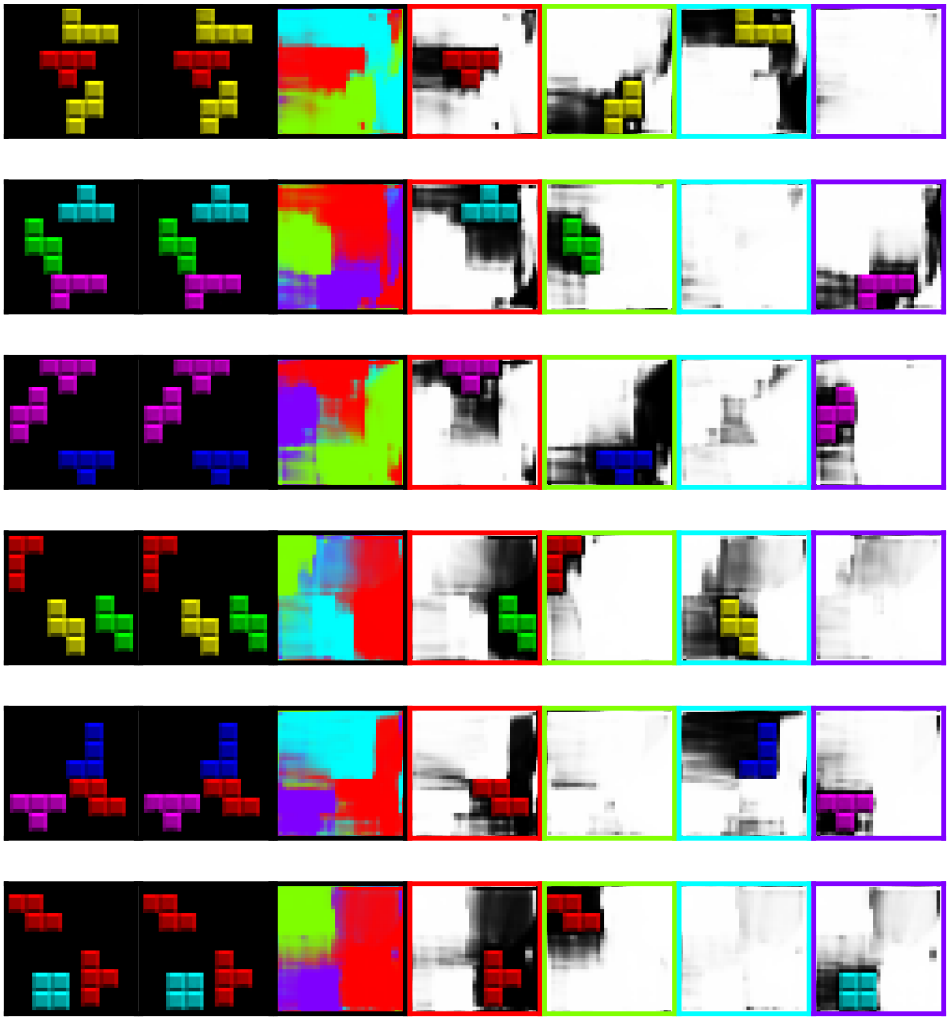


Figure 18. Additional Tetrominoes scene decompositions with $K = 4$, $L = 3$ and $I = 3$. First column is the ground truth image, second column is the reconstruction, third column is the mask, and the remaining columns are the masked components. The Gaussian image likelihood has only a weak inductive bias to encourage the background to be assigned to a single component; the model often learns to split the simple black background across all components.

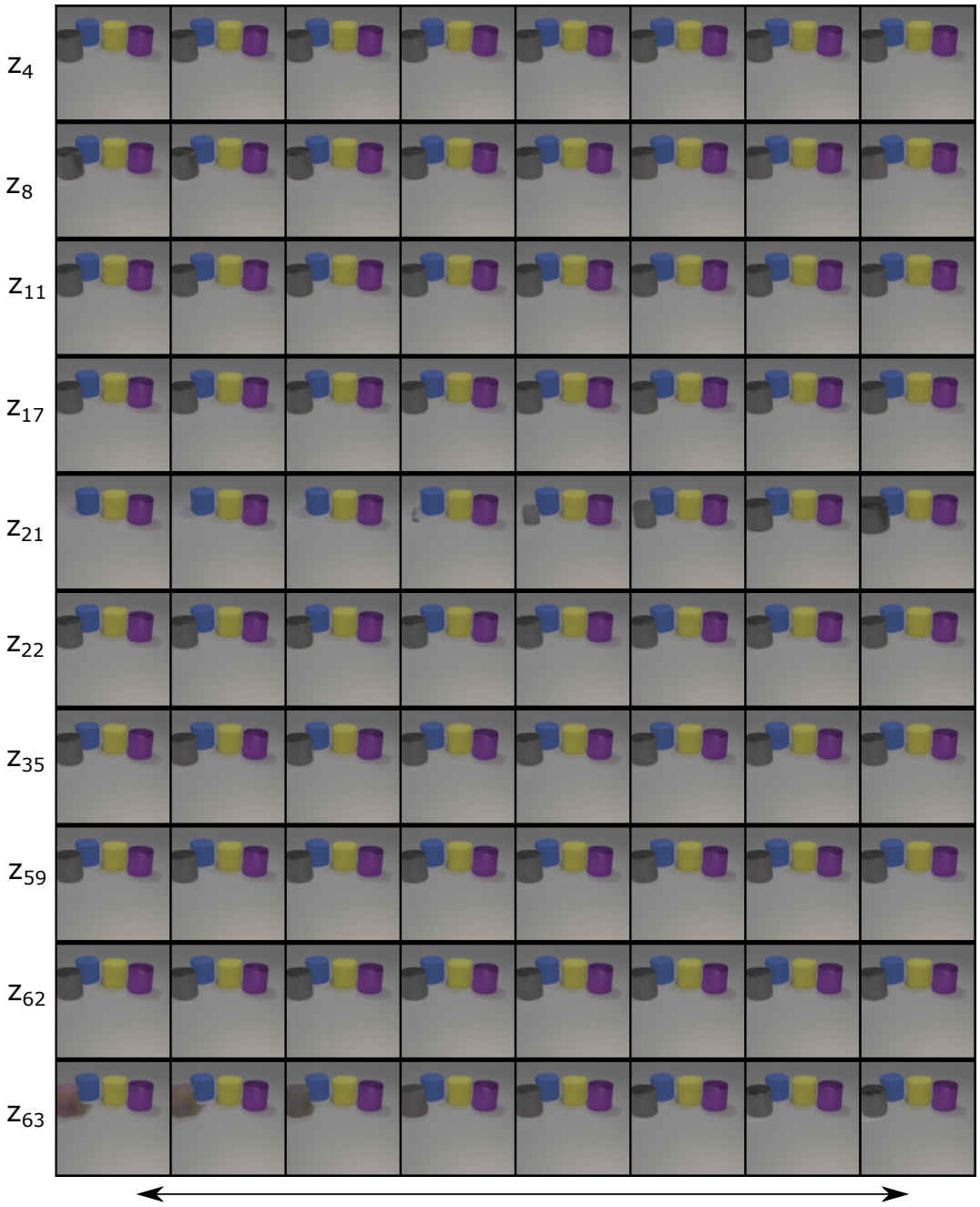


Figure 19. Varying random latent dimensions for CLEVR6. We randomly select latent dimensions from a single object component to vary. Most of the latent dimensions are deactivated and do not change the image.

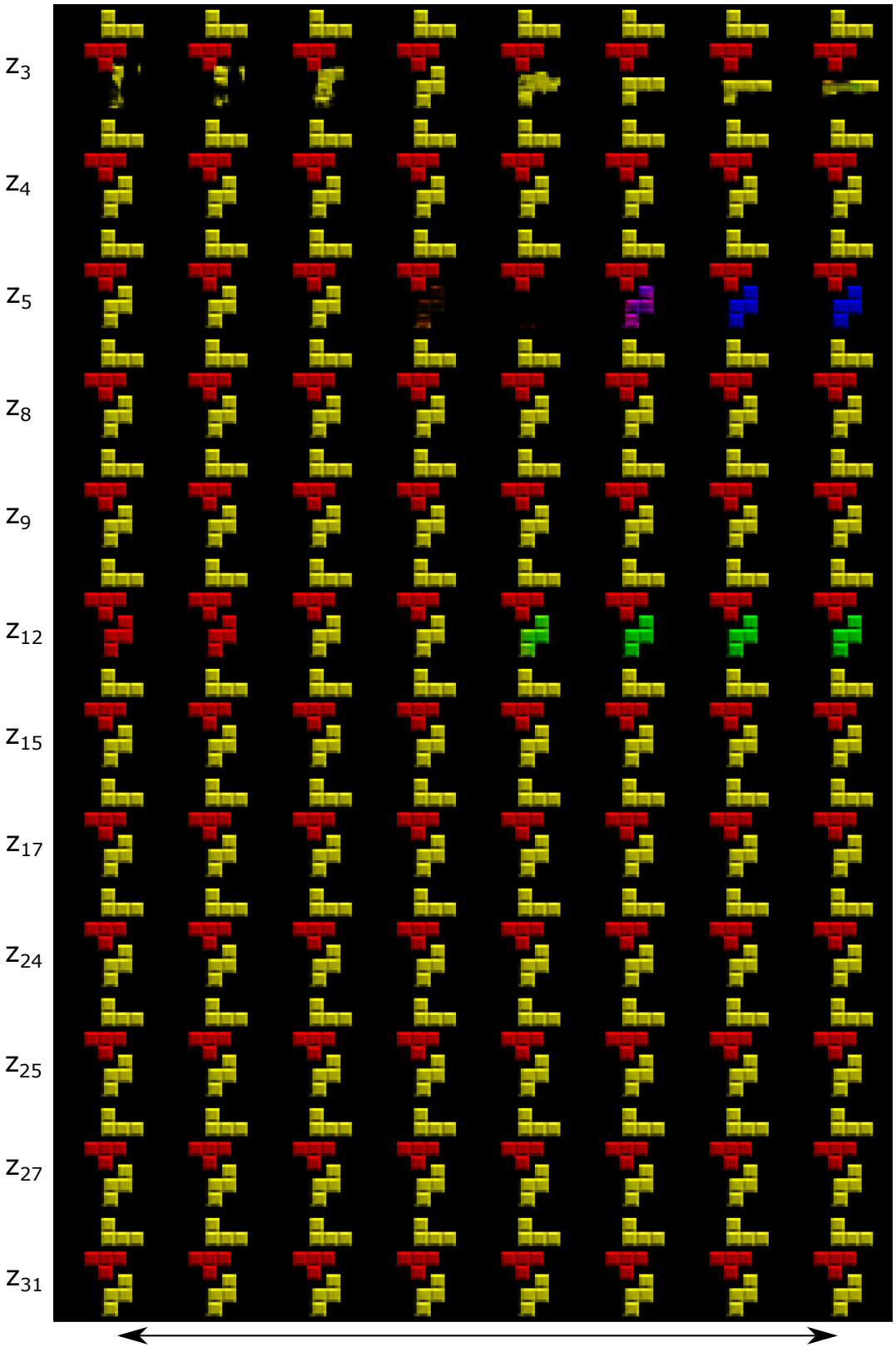


Figure 20. Varying random latent dimensions for Tetrominoes. We randomly select latent dimensions from a single object component to vary. Most of the latent dimensions are deactivated and do not change the image. The model shown here was trained *without* GECO.

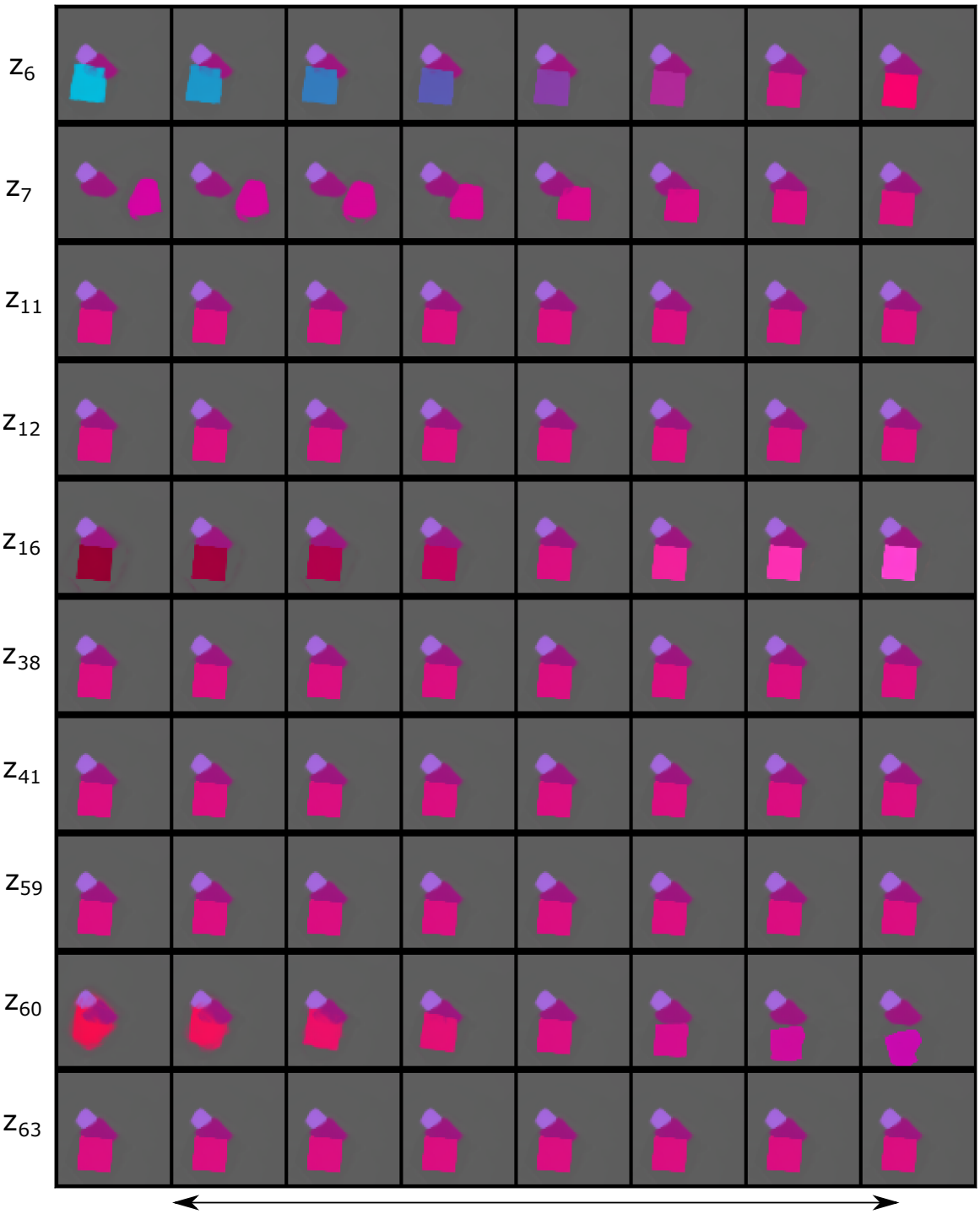
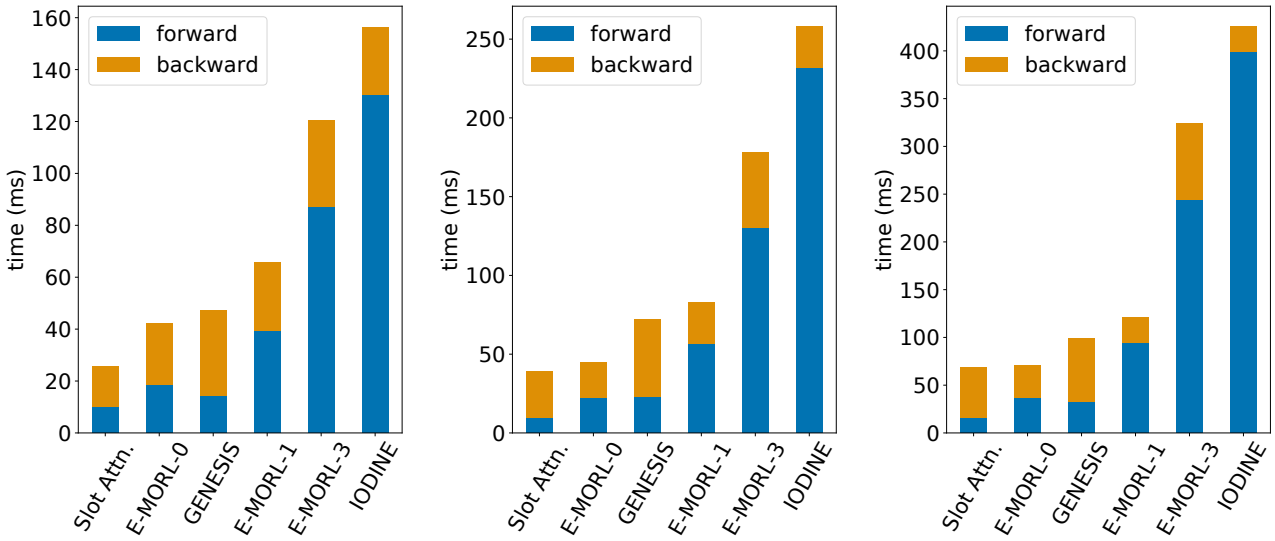


Figure 21. Varying random latent dimensions for Multi-dSprites. We randomly select latent dimensions from a single object component to vary. Most of the latent dimensions are deactivated and do not change the image.



(a) 64×64 images (b) 96×96 images (c) 128×128 images

Figure 22. Forward and backward pass timings

Gaussian image likelihood has x_p distributed according to a Gaussian with mean given by the sum over K of $\pi_{k,p} y_{k,p}$:

$$p_\theta(x \mid \mathbf{z}^L) = \prod_{p=1}^{P} \mathcal{N}(x_p; \sum_{k=1}^{K} \pi_{k,p} y_{k,p}, \sigma^2) \tag{12}$$

with variance σ^2 fixed for all p. We use this formulation for Tetrominoes ($\sigma = 0.3$) and Multi-dSprites ($\sigma = 0.1$).

The other image likelihood we consider is a *mixture* of pixel-wise Gaussians (Burgess et al., 2019; Greff et al., 2019; Engelcke et al., 2020):

$$p_\theta(x \mid \mathbf{z}^L) = \prod_{p=1}^{P} \sum_{k=1}^{K} \pi_{k,p} \mathcal{N}(x_p; y_{k,p}, \sigma^2) \tag{13}$$

with the variance σ^2 fixed for all k and all p. As demonstrated in our experiments, this likelihood attempts to segment the background into a single component, which is (perhaps unintuitively) more challenging for the Tetrominoes and Multi-dSprites environments. Therefore, we only use it on CLEVR6 ($\sigma = 0.1$). As evidenced by IODINE, with sufficient hyperparameter tuning it seems possible to use the mixture likelihood for all three environments; however, we wished to minimize hyperparameter optimization to avoid overfitting to these specific environments. We attempted to use $\sigma = 0.1$ for all environments (like IODINE) but found that slightly increasing σ to 0.3 on Tetrominoes helped the model converge more rapidly.

A third image likelihood that has been previously used (van Steenkiste et al., 2020) is a *layered* image representation (i.e., alpha compositing). We did not compare against this one for the following reasons. The standard way to implement layered rendering requires imposing an *ordering* on the layers. This breaks the symmetry of the latent components, which is undesirable. Alternatively, one could treat the ordering as a discrete random variable that must be inferred, but 1) this increases the difficulty of the already-challenging inference problem, and 2) this can lead to the generation of implausible scenes due to uncertainty about the true (unknown) depth ordering.

Bottom-up inference network Following Slot Attention and IODINE, we use latent dimensions of $D = 64$ for CLEVR6 and Multi-dSprites and $D = 32$ for Tetrominoes. Before the first layer, the provided image is embedded using a simple CNN (Locatello et al., 2020):

Image encoder

Type	Size/*Ch.*	Act. Func.
$H \times W$ image	*3*	
Conv 5×5	*64*	ReLU
Conv 5×5	*64*	ReLU
Conv 5×5	*64*	ReLU
Conv 5×5	*64*	ReLU

where each 2D convolution uses stride of 1 and padding of 2. A learned **positional encoding** of shape $H \times W \times 4$ is projected to match the channel dimension 64 with a linear layer and then added to the image embedding. Like Slot Attention, the four dimensions of the encoding captures the cardinal directions of left, right, top, and bottom respectively. This enables extracting spatially-aware image features while processing the image in a permutation invariant manner. The positionally-aware image embedding is flattened along the spatial dimensions to $HW \times 64$ and then processed sequentially with a LayerNorm, 64-dimensional linear layer, a ReLU activation, and another 64-dimensional linear layer. The result is used as the key and value for scaled dot-product set attention.

Each of the L bottom-up stochastic layers share these parameters:

l^{th} layer

Type	Size/*Ch.*	Act. Func.
key (k)	$64 \to D$	None
value (v)	$64 \to D$	None
query (q)	$D \to D$	None
DualGRU	$2D \to 2D$	Sigmoid/Tanh
MLP $(\boldsymbol{\mu})$	$D \to 2D \to D$	ReLU
MLP $(\boldsymbol{\sigma})$	$D \to 2D \to D$	ReLU, softplus

Each of the above operations are applied symmetrically to each of the K elements of $\mathbf{z}^l$. The two D-dimensional outputs of the DualGRU are passed through separate trainable LayerNorm layers before the MLPs. The posterior parameters μ^0 and σ^0 provided to the DualGRU when $l = 1$ are trainable and are initialized to $\mathbf{0}$ and $\mathbf{1}$ respectively.

Hierarchical prior network Each of the $L - 1$ data-dependent layers in the prior shares parameters. These predict the mean and variance of a Gaussian conditional on a D-dimensional random sample $\mathbf{z}$. We use ELU activations (Clevert et al., 2016) here but use ReLU activations in the inference network in the parts of the architecture similar to Slot Attention.

l^{th} layer

Type	Size/*Ch.*	Act. Func.
MLP	$D \to 128$	ELU
Linear $(\boldsymbol{\mu})$	$128 \to D$	None
Linear $(\boldsymbol{\sigma})$	$128 \to D$	softplus

This computation is applied symmetrically to each of the K elements of $\mathbf{z}$.

Refinement The output of the refinement network is $\boldsymbol{\delta\lambda} = [\boldsymbol{\delta\mu}, \boldsymbol{\delta\sigma}]$ which is used to make the additive update to the posterior.

Refinement network

Type	Size/*Ch.*	Act. Func.
$[\boldsymbol{\lambda}, \nabla_{\boldsymbol{\lambda}}\mathcal{L}]$	$4D$	
MLP	$4D \to 128 \to D$	ELU
GRU	$D \to D$	Sigmoid/Tanh
Linear $(\boldsymbol{\delta\mu})$	$D \to D$	None
Linear $(\boldsymbol{\delta\sigma})$	$D \to D$	SP

The two vector inputs to the refinement network are first processed with trainable LayerNorm layers. We compute the update for each of the K elements of the set of posterior parameters $\boldsymbol{\lambda}$ in parallel.

Decoder The spatial broadcast decoder we use is similar to IODINE's except we adopt Slot Attention's positional encoding to mirror the encoding used during bottom-up inference. Each of the K elements of $\mathbf{z}$ are decoded independently in parallel.

Spatial broadcast decoder

Type	Size/*Ch.*	Act. Func.
Input: $\mathbf{z}$	D	
Broadcast	$(H+10) \times (W+10) \times D$	
Pos. Enc.	$4 \rightarrow D$	Linear
Conv 3×3	*64*	ELU
Conv 3×3	*64*	ELU
Conv 3×3	*64*	ELU
Conv 3×3	*64*	ELU
Conv 3×3	*4*	None

Each convolutional layer uses a stride of 1 and no padding. The channel dimension of the positional encoding is again projected to D before being added to the broadcasted $\mathbf{z}$. For Tetrominoes, following Slot Attention we use a lighter decoder that has just three convolutional layers with 5×5 kernels, stride 1 and padding 1, and channel dimension of 32.

The deconvolutional decoder we use for E-MORL-X-S in Figure 10 is identical to Slot Attention's (see Section E.3, Table 6 in their paper).

The four dimensional output of the decoder is split into K masks and RGB images. The masks are normalized over K by a softmax and the RGB outputs are passed through a sigmoid to squash values between zero and one. Images under both considered likelihood models are reconstructed by summing the normalized masks multiplied by the RGB components across K.

F.2. GECO (Rezende & Viola, 2018)

Where needed, we mitigate posterior collapse while simultaneously balancing the reconstruction and KL with GECO. This reformulates the objective as a minimization of the KL subject to a constraint on the reconstruction error. The training loss $\mathcal{L}$ (Equation 8) is modified for GECO:

$$\mathcal{L}^{(L,0)} = D_{KL}\big(q_\phi(\mathbf{z}^{1:L} \mid x) \parallel p_\theta(\mathbf{z}^{1:L})\big) - \zeta\big(C + \mathcal{L}_{\text{NLL}}\big)$$
$$\mathcal{L}^{(L,i)} = D_{KL}\big(q(\mathbf{z}; \boldsymbol{\lambda}^{(L,i)}) \parallel p(\mathbf{z}^L)\big) - \zeta\big(C + \mathcal{L}_{\text{NLL}}^{(L,i)}\big).$$

Here, ζ is a Lagrange parameter that penalizes the model when the reconstruction error is higher than a manually-specified threshold C. Depending on the hierarchical prior variant we replace $p(\mathbf{z}^L)$, e.g., with $p_\theta(\mathbf{z}^1|\mathbf{z}^2)$ for reversed prior++. We use the recommended exponential moving average C_{EMA} with parameter $\alpha = 0.99$ to keep track of the difference between the reconstruction error of the mini-batch and C. The Lagrange parameter is updated at every step with $\zeta' = \zeta -$ 1e-6 C_{EMA}. For numerical stability, we use $\texttt{softplus}(\zeta)$ when computing the GECO update and constrain $\zeta \geq 0.55$ so that $\texttt{softplus}(\zeta)$ is always greater than or equal to one. For CLEVR6, we used $C = -61000$ and for Tetrominoes we used $C = -4500$. GECO was not needed on Multi-dSprites.