LAVAE: Disentangling Location and Appearance

Andrea Dittadi[1], Ole Winther[1,2,3]
[1]Technical University of Denmark, [2]Copenhagen University Hospital, [3]University of Copenhagen
{adit,olwi}@dtu.dk

Abstract

We propose a probabilistic generative model for unsupervised learning of structured, interpretable, object-based representations of visual scenes. We use amortized variational inference to train the generative model end-to-end. The learned representations of object location and appearance are fully disentangled, and objects are represented independently of each other in the latent space. Unlike previous approaches that disentangle location and appearance, ours generalizes seamlessly to scenes with many more objects than encountered in the training regime. We evaluate the proposed model on multi-MNIST and multi-dSprites data sets.

1 Introduction

Many hallmarks of human intelligence rely on the capability to perceive the world as a layout of distinct physical objects that endure through time – a skill that infants acquire in early childhood [20; 21; 22]. Learning object-based representations of visual scenes, however, is still regarded as an open challenge for artificial systems [1].

Recently, there has been a growing interest in unsupervised learning of disentangled representations [14], which should separate the distinct, informative factors of variations in the data, and contain all the information on the data in a compact, interpretable structure [1]. This notion is highly relevant in the context of visual scene representation learning, where distinct objects should arguably be represented in a disentangled fashion. However, despite recent breakthroughs [3; 9; 11], multi-object scenarios are rarely considered [2; 4; 24].

We propose the Location-Appearance Variational AutoEncoder (LAVAE), a probabilistic generative model that, without supervision, learns structured, compositional, object-based representations of visual scenes. We explicitly model an object's location and appearance with distinct latent variables, unlike in most previous works, thus providing a highly beneficial inductive bias. As in variational autoencoders (VAEs) [12; 18], we parameterize the approximate variational posterior of the latent variables with inference networks that are trained end-to-end with the generative model. Our model learns to correctly count objects and compute a compositional and interpretable representation of the scene. Objects are represented independently of each other, and each object's location and appearance are disentangled. Unlike previous approaches that disentangle location and appearance, LAVAE generalizes seamlessly to scenes with many more objects than in the training regime. We demonstrate these capabilities on multi-MNIST and multi-dSprites data sets similar to those in [4] and [7].

2 Method

Generative model. We propose a latent variable model for images in which the latent space is factored into location and appearance of a variable number of objects. For each image $\mathbf{x}$ with D pixels, the number of objects is modeled by a latent variable $n \leq M$, their locations by n latent variables $\{\mathbf{z}_{\text{loc}}^{(i)}\}_{i=1}^{n}$, and their appearance by $\{\mathbf{z}_{\text{app}}^{(i)}\}_{i=1}^{n}$. The joint distribution of the observed and

Workshop on Perception as Generative Reasoning, NeurIPS 2019, Vancouver, Canada.

latent variables for each data point is:

$$p(\mathbf{x}, \mathbf{z}_{\text{loc}}, \mathbf{z}_{\text{app}}, n) = p(\mathbf{x} \,|\, \mathbf{z}_{\text{loc}}, \mathbf{z}_{\text{app}}) \left(\prod_{i=1}^{n} p\Big(\mathbf{z}_{\text{loc}}^{(i)} \,|\, \{\mathbf{z}_{\text{loc}}^{(j)}\}_{j<i}\Big) p(\mathbf{z}_{\text{app}}^{(i)}) \right) p(n) \tag{1}$$

where we use the shorthand $\mathbf{z}_{\text{app}} = \{\mathbf{z}_{\text{app}}^{(i)}\}_{i=1}^{n}$ and similarly for $\mathbf{z}_{\text{loc}}$.

The generative process can be described as follows. First, the number of objects n is sampled from a categorical distribution $p(n) = \text{Cat}(M+1, \mathbf{a})$ where $\mathbf{a}$ is a learned probability vector of size $M+1$ and n takes values 0 to M. The n location variables are sequentially sampled without replacement from a categorical distribution with D classes:

$$p\Big(\mathbf{z}_{\text{loc}}^{(i)} \,|\, \{\mathbf{z}_{\text{loc}}^{(j)}\}_{j<i}\Big) = \text{Cat}(D, \mathbf{b}^{(i)}), \qquad \mathbf{b}^{(i)} = \frac{1}{D-i+1} \left(\mathbf{1}_D - \sum\nolimits_{j=1}^{i-1} \mathbf{z}_{\text{loc}}^{(j)} \right) \tag{2}$$

where $i = 1, \ldots, n$, and $\mathbf{1}_D$ is a vector of ones of length D. To each $\mathbf{z}_{\text{loc}}^{(i)}$, which is a one-hot representation of an object's location, corresponds a continuous appearance vector $\mathbf{z}_{\text{app}}^{(i)} \sim \mathcal{N}(\mathbf{0}_L, \mathbf{I}_L)$ of size L that describes the object. The likelihood function $p(\mathbf{x} \,|\, \mathbf{z}_{\text{loc}}, \mathbf{z}_{\text{app}})$ is parameterized in a compositional manner. For each image, the visual representation, or *sprite*, of the ith object is generated from $\mathbf{z}_{\text{app}}^{(i)}$ by a shared function f_{sprite}. Each sprite is then convolved with a 2-dimensional Kronecker delta that is the one-hot representation of the object's location. Finally, the resulting n tensors are added together to give the pixel-wise parameters $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_D)$ of the distribution:

$$\boldsymbol{\lambda} = \sum_{i=1}^{n} f_{\text{sprite}}(\mathbf{z}_{\text{app}}^{(i)}) * \mathbf{z}_{\text{loc}}^{(i)} \tag{3}$$

Inference model. The approximate posterior for a data point $\mathbf{x}$ has the following form:

$$q(\mathbf{z}_{\text{app}}, \mathbf{z}_{\text{loc}}, n \,|\, \mathbf{x}) = q(n \,|\, \mathbf{x}) \prod_{i=1}^{n} q\Big(\mathbf{z}_{\text{app}}^{(i)} \,|\, \mathbf{z}_{\text{loc}}^{(i)}, \mathbf{x}\Big) q\Big(\mathbf{z}_{\text{loc}}^{(i)} \,|\, \{\mathbf{z}_{\text{loc}}^{(j)}\}_{j<i}, \mathbf{x}\Big) \tag{4}$$

The lower bound (ELBO) to the marginal log likelihood is the sum of the ELBO for each data point:

$$\log p(\mathbf{x}) \geq \mathbb{E}_q \left[\log p(\mathbf{x} \,|\, \mathbf{z}_{\text{loc}}, \mathbf{z}_{\text{app}}) \right] + \mathbb{E}_q \left[\log \frac{p(\mathbf{z}_{\text{loc}}, \mathbf{z}_{\text{app}}, n)}{q(\mathbf{z}_{\text{loc}}, \mathbf{z}_{\text{app}}, n \,|\, \mathbf{x})} \right] \tag{5}$$

where the first term is the likelihood and the second is the negative KL (derivation in Appendix A).

Two inference networks compute the appearance and location feature maps, $\mathbf{h}_{\text{app}}$ and $\mathbf{h}_{\text{loc}}$, both having size D like the input. The number of objects n is then sampled from a categorical distribution parameterized by a function of the location features $f_{\text{count}}(\mathbf{h}_{\text{loc}})$. Object locations are obtained by sampling n times without replacement from a categorical distribution with logits $\mathbf{h}_{\text{loc}}$. The vector at location $\mathbf{z}_{\text{loc}}^{(i)}$ in the feature map $\mathbf{h}_{\text{app}}$ represents the appearance parameters for object i, i.e. the mean and log variance of $q(\mathbf{z}_{\text{app}}^{(i)} \,|\, \mathbf{z}_{\text{loc}}^{(i)}, \mathbf{x})$. The overall inference process can be summarized as follows:

$$q(n \,|\, \mathbf{x}) = \text{Cat}(n;\, M+1, f_{\text{count}}(\mathbf{h}_{\text{loc}})) \tag{6}$$

$$q\Big(\mathbf{z}_{\text{loc}}^{(i)} \,|\, \{\mathbf{z}_{\text{loc}}^{(j)}\}_{j<i}, \mathbf{x}\Big) = \text{Cat}(\mathbf{z}_{\text{loc}}^{(i)};\, D, \mathbf{b}^{(i)}) \qquad i = 1, \ldots, n \tag{7}$$

$$\boldsymbol{\mu}_{\text{app}}^{(i)},\ \boldsymbol{\sigma}_{\text{app}}^{2\,(i)} = \mathbf{h}_{\text{app}}[\mathbf{z}_{\text{loc}}^{(i)}] \qquad i = 1, \ldots, n \tag{8}$$

$$q(\mathbf{z}_{\text{app}}^{(i)} \,|\, \mathbf{z}_{\text{loc}}^{(i)}, \mathbf{x}) = \mathcal{N}(\mathbf{z}_{\text{app}}^{(i)};\, \boldsymbol{\mu}_{\text{app}}^{(i)}, \text{diag}(\boldsymbol{\sigma}_{\text{app}}^{2\,(i)})) \qquad i = 1, \ldots, n \tag{9}$$

where by $\mathbf{v}[\mathbf{e}_k]$ we denote the kth element of $\mathbf{v}$ (with $\mathbf{e}_k$ a standard basis vector), and the probability vector for location sampling at each step is computed iteratively:

$$\mathbf{b}_0^{(i)} = \text{softmax}(\mathbf{h}_{\text{loc}}) \cdot \left(\mathbf{1}_D - \sum\nolimits_{j=1}^{i-1} \mathbf{z}_{\text{loc}}^{(j)} \right), \qquad \mathbf{b}^{(i)} = \mathbf{b}_0^{(i)} \cdot ||\mathbf{b}_0^{(i)}||_1^{-1} \tag{10}$$

Table 1: Quantitative results on multi-MNIST and multi-dSprites *test sets*. The log likelihood lower bound is estimated with 100 importance samples. Note that the MNIST-10k dataset is a more complex task than MNIST-1k because the model has to capture a larger variation in appearance (see Appendix B). The object count accuracy is measured as the percentage of images for which the inferred number of objects matches the ground truth (which is not available during training).

	multi-MNIST-1k		multi-MNIST-10k		multi-dSprites	
	$-\log p(\mathbf{x}) \leq$	n acc.	$-\log p(\mathbf{x}) \leq$	n acc.	$-\log p(\mathbf{x}) \leq$	n acc.
LAVAE	$\mathbf{42.2} \pm 1.1$	$\geq 99.9\%$	$\mathbf{60.4} \pm 2.8$	$\geq 99.7\%$	$\mathbf{36.4} \pm 6.4$	$\geq 99.7\%$
baseline	51.6 ± 0.6	—	64.6 ± 0.8	—	45.8 ± 0.7	—

3 Results

We evaluate our model on multi-MNIST and multi-dSprites data sets consisting of 200k images with 0 to 3 objects in each image, of which 190k are used for training, and the remaining 10k for validation. Examples with more than 3 objects are never used for training. See Appendix B for more details on the data sets. In all experiments we use the same architecture for LAVAE (implementation details in Appendix C).

Figure 1 qualitatively shows the performance of LAVAE on multi-MNIST-10k test images. The inferred location of all objects in an image are summarized by a black canvas where white pixels indicate object presence. For each of those locations, the model infers a corresponding appearance latent variable. Each digit on the right is generated by f_{sprite} from one of these appearance variables.

Samples from the prior are shown in Figure 2: note that the number of generated objects is consistent with the training set, since the prior $p(n)$ is learned. Quantitative evaluation results are shown in Table 1 and compared with a VAE baseline. The inferred object count was correct on almost all images of *all test sets*—even with a held-out number of objects—and across all tested random seeds.

Appendix D presents qualitative results on multi-dSprites, and test results on an additional multi-MNIST test set with out-of-distribution object numerosity.

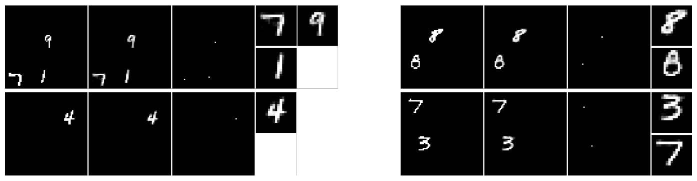

Figure 1: **Inference and reconstruction** on test images. For each image, from left to right: input, reconstruction, summary of inferred locations, sprites generated from each appearance latent variable.

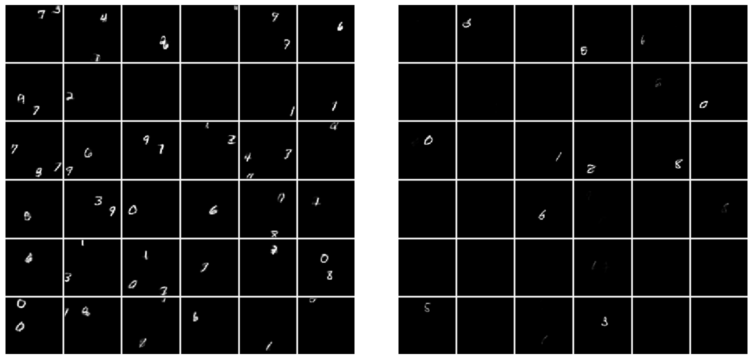

Figure 2: **Samples** generated by LAVAE (left) from its prior $p(\mathbf{z}_{\text{loc}} \mid n)p(\mathbf{z}_{\text{app}} \mid n)p(n)$, where $p(n)$ is learned from data, and by the baseline (right) from its prior $p(\mathbf{z})$.

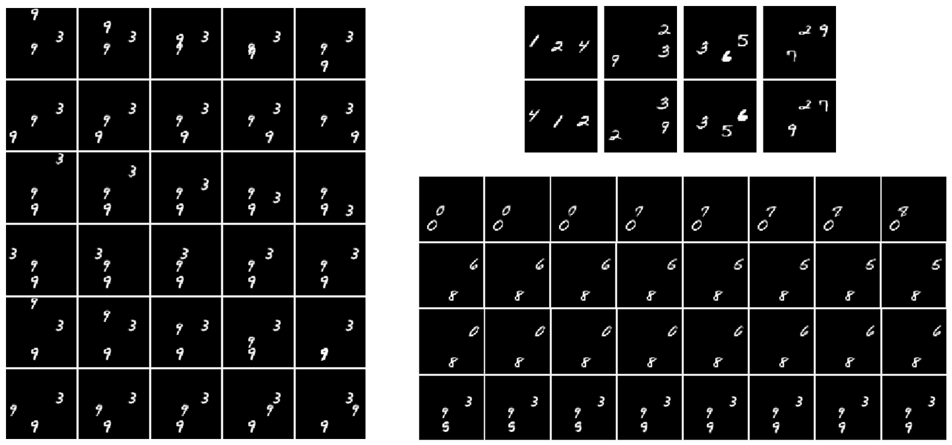

Figure 3: **Disentanglement experiments** on test images. Objects are represented independently of each other, and their location and appearance are disentangled by design. **Left:** Latent traversal on one of the 7 location variables. **Top right:** Reordering the sequence $\{\mathbf{z}_{\text{loc}}^{(i)}\}_i$ or equivalently of $\{\mathbf{z}_{\text{app}}^{(i)}\}_i$ leads to objects being swapped (top row: original reconstruction; bottom row: swapped objects). **Bottom right:** In each row, latent traversal on one appearance variable along one dimension.

4 Related work

Our work builds on recent advances in probabilistic generative modelling, in particular variational autoencoders (VAEs) [12; 18]. One of the methods closest to our work in spirit is Attend Infer Repeat (AIR) [4], which performs explicit object-wise inference through a recurrent network that iteratively attends to one object at a time. A limitation of this approach however is that it has not been shown to generalize well to a larger number of objects. Closely related to our work is also the multi-entity VAE [16], in which multiple objects are independently modelled by different latent variables. The inference process does not include an explicit attention mechanism, and uses instead a spatial KL map as a proxy for object presence. Each object's latent is decoded into a full image, and these are aggregated by an element-wise operation, thus the representation of each object's location and appearance are entangled. In the same spirit, the recently proposed generative models MONet [2] and IODINE [7] learn without supervision to segment the scene into independent and interpretable object-based parts. Although these are more flexible than AIR and can model more complex scenes, the representations they learn of object location and appearance are not disentangled.

Other unsupervised approaches to visual scene decomposition include Neural Expectation Maximization [6; 24], which amortizes the classic EM for a spatial mixture model, and Generative Query Networks [5], that learn representations of rich 3D scenes but do not factor them into objects and need point-of-view information during training. Methods following the vision-as-inverse-graphics paradigm [17; 26] usually learn structured object-centered representations by making strong assumptions on the latent codes or by exploiting the true generative model [13; 23; 25]. Finally, supervised approaches include semantic image segmentation, where acquiring labels for training is typically expensive, and the focus is not on learning structured representations [8; 10; 19].

5 Conclusion

We presented LAVAE, a probabilistic generative model for unsupervised learning of structured, compositional, object-based representations of visual scenes. We follow the amortized stochastic variational inference framework, and approximate the latent posteriors by inference networks trained end-to-end with the generative model. On multi-MNIST and multi-dSprites data sets, LAVAE learns without supervision to correctly count and locate all objects in a scene. Thanks to the structure of the generative model, objects are represented independently of each other, and the location and appearance of each object are completely disentangled. We demonstrate this in qualitative experiments, where we manipulate location or appearance of single objects in the latent space. Our model naturally generalizes to visual scenes with more objects than encountered during training.

References

[1] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[2] C. P. Burgess, L. Matthey, N. Watters, R. Kabra, I. Higgins, M. Botvinick, and A. Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

[3] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180, 2016.

[4] S. A. Eslami, N. Heess, T. Weber, Y. Tassa, D. Szepesvari, G. E. Hinton, et al. Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems*, pages 3225–3233, 2016.

[5] S. A. Eslami, D. J. Rezende, F. Besse, F. Viola, A. S. Morcos, M. Garnelo, A. Ruderman, A. A. Rusu, I. Danihelka, K. Gregor, et al. Neural scene representation and rendering. *Science*, 360 (6394):1204–1210, 2018.

[6] K. Greff, S. van Steenkiste, and J. Schmidhuber. Neural expectation maximization. In *Advances in Neural Information Processing Systems*, pages 6691–6701, 2017.

[7] K. Greff, R. L. Kaufmann, R. Kabra, N. Watters, C. Burgess, D. Zoran, L. Matthey, M. Botvinick, and A. Lerchner. Multi-object representation learning with iterative variational inference. *arXiv preprint arXiv:1903.00450*, 2019.

[8] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[9] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2(5):6, 2017.

[10] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 11–19, 2017.

[11] H. Kim and A. Mnih. Disentangling by factorising. *arXiv preprint arXiv:1802.05983*, 2018.

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

[13] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum. Deep convolutional inverse graphics network. In *Advances in neural information processing systems*, pages 2539–2547, 2015.

[14] F. Locatello, S. Bauer, M. Lucic, S. Gelly, B. Schölkopf, and O. Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. *arXiv preprint arXiv:1811.12359*, 2018.

[15] L. Matthey, I. Higgins, D. Hassabis, and A. Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

[16] C. Nash, S. A. Eslami, C. Burgess, I. Higgins, D. Zoran, T. Weber, and P. Battaglia. The multi-entity variational autoencoder. *NIPS Workshops*, 2017.

[17] T. Poggio, V. Torre, and C. Koch. Computational vision and regularization theory. *Nature*, 317 (6035):314–319, 1985.

[18] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[19] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[20] E. S. Spelke. Principles of object perception. *Cognitive science*, 14(1):29–56, 1990.

[21] E. S. Spelke. Where perceiving ends and thinking begins: The apprehension of objects in infancy. In *Perceptual development in infancy*, pages 209–246. Psychology Press, 2013.

[22] E. S. Spelke and K. D. Kinzler. Core knowledge. *Developmental science*, 10(1):89–96, 2007.

[23] Y. Tian, A. Luo, X. Sun, K. Ellis, W. T. Freeman, J. B. Tenenbaum, and J. Wu. Learning to infer and execute 3d shape programs. *arXiv preprint arXiv:1901.02875*, 2019.

[24] S. Van Steenkiste, M. Chang, K. Greff, and J. Schmidhuber. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. *arXiv preprint arXiv:1802.10353*, 2018.

[25] J. Wu, J. B. Tenenbaum, and P. Kohli. Neural scene de-rendering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 699–707, 2017.

[26] A. Yuille and D. Kersten. Vision as bayesian inference: analysis by synthesis? *Trends in Cognitive Sciences*, 10(7):301–308, 2006.

A KL divergence

The KL in Eq. (5) can be expanded as follows:

$$
\begin{aligned}
D_{\mathrm{KL}}\big(q(\mathbf{z}_{\mathrm{loc}}, \mathbf{z}_{\mathrm{app}}, n \,|\, \mathbf{x}) \,\|\, p(\mathbf{z}_{\mathrm{loc}}, \mathbf{z}_{\mathrm{app}}, n)\big) &= \mathbb{E}_q \left[\log \frac{q(\mathbf{z}_{\mathrm{app}} \,|\, \mathbf{z}_{\mathrm{loc}}, n, \mathbf{x}) q(\mathbf{z}_{\mathrm{loc}} \,|\, n, \mathbf{x}) q(n \,|\, \mathbf{x})}{p(\mathbf{z}_{\mathrm{app}} \,|\, n) p(\mathbf{z}_{\mathrm{loc}} \,|\, n) p(n)}\right] \\
= \mathbb{E}_{q(\mathbf{z}_{\mathrm{loc}}, \mathbf{z}_{\mathrm{app}}, n \,|\, \mathbf{x})} \left[\log \frac{q(\mathbf{z}_{\mathrm{app}} \,|\, \mathbf{z}_{\mathrm{loc}}, n, \mathbf{x})}{p(\mathbf{z}_{\mathrm{app}} \,|\, n)}\right] &+ \mathbb{E}_{q(n \,|\, \mathbf{x})} \left[\log \frac{q(n \,|\, \mathbf{x})}{p(n)}\right] \\
+ \mathbb{E}_{q(\mathbf{z}_{\mathrm{loc}}, n \,|\, \mathbf{x})} \left[\log \frac{q(\mathbf{z}_{\mathrm{loc}} \,|\, n, \mathbf{x})}{p(\mathbf{z}_{\mathrm{loc}} \,|\, n)}\right] & \\
= \mathbb{E}_{q(\mathbf{z}_{\mathrm{loc}} \,|\, n, \mathbf{x}) q(n \,|\, \mathbf{x})} \left[\sum_{i=1}^{n} D_{\mathrm{KL}}\Big(q(\mathbf{z}_{\mathrm{app}}^{(i)} \,|\, \mathbf{z}_{\mathrm{loc}}^{(i)}, \mathbf{x}) \,\Big\|\, p(\mathbf{z}_{\mathrm{app}}^{(i)})\Big)\right] &+ D_{\mathrm{KL}}\big(q(n \,|\, \mathbf{x}) \,\|\, p(n)\big) \\
+ \mathbb{E}_{q(n \,|\, \mathbf{x}) q(\mathbf{z}_{\mathrm{loc}} \,|\, n, \mathbf{x})} \left[\sum_{i=1}^{n} \left(\log q\Big(\mathbf{z}_{\mathrm{loc}}^{(i)} \,|\, \big\{\mathbf{z}_{\mathrm{loc}}^{(j)}\big\}_{j<i}, \mathbf{x}\Big) - \log p\Big(\mathbf{z}_{\mathrm{loc}}^{(i)} \,|\, \big\{\mathbf{z}_{\mathrm{loc}}^{(j)}\big\}_{j<i}\Big)\right)\right] &
\end{aligned}
$$

where all expectations can be estimated by Monte Carlo sampling.

B Data sets

Multi-MNIST. Each image in the multi-MNIST data set consists of a number of statically binarized MNIST digits scattered at random (avoiding overlaps) onto a black canvas of size 64×64. The digits are first rescaled from their original size (28×28) to 15×15 by bilinear interpolation, and finally binarized by rounding. When generating images, digits are picked from a pool of either 1k or 10k MNIST digits. We call the two resulting data sets *multi-MNIST-1k* and *multi-MNIST-10k*, depending on the number of MNIST digits used for generation. We independently model each pixel as a Bernoulli random variable parameterized by the decoder:

$$p(\mathbf{x} \,|\, \mathbf{z}_{\mathrm{loc}}, \mathbf{z}_{\mathrm{app}}) = \prod_{d=1}^{D} p(x_d \,|\, \mathbf{z}_{\mathrm{loc}}, \mathbf{z}_{\mathrm{app}}) = \prod_{d=1}^{D} \mathrm{Bernoulli}(x_d;\, \lambda_d)\,, \tag{11}$$

where the parameter vector $\boldsymbol{\lambda}$ defined in Section 2 is the pixel-wise mean of the Bernoulli distribution.

Multi-dSprites. The multi-dSprites data set is generated similarly to the multi-MNIST ones, by scattering a number of sprites on a black canvas of size 64×64. Here, the sprites are simple shapes in different colors, sizes, and orientations, as in the dSprites data set [9; 15]. The shape of each sprite is randomly chosen among square, ellipse, and triangle. The maximum sprite size is 19×19, and each sprite's scale is randomly chosen among 6 linearly spaced values in $[0.6, 1.0]$. The orientation angle of each sprite is uniformly chosen among 40 linearly spaced values in $[0, 2\pi)$. The color is uniformly chosen among the 7 colors such that the RGB values are saturated (either 0 or 255), and at least one of them is not 0. This means that each color component of each pixel is a binary random variable and can be modelled independently as in the multi-MNIST case (leading to $3D$ terms in the likelihood, instead of D).

C Implementation details

The input image $\mathbf{x}$ is fed into a residual network with 4 blocks having 2 convolutional layers each. Every convolution is followed by a Leaky ReLU and batch normalization. A final convolutional layer outputs the feature map $\mathbf{h}_{\mathrm{app}}$ with $2L$ channels. The output of an identical but independent residual network is fed into a 3-layer convolutional network with one output channel that represents the location logits $\mathbf{h}_{\mathrm{loc}}$. The logits are multiplied by a constant factor $\gamma = 4 \approx 1/2\, \log D$. See below for more details about the rescaling of logits.

The number of objects is then inferred by a deterministic function f_{count} of the location logits. This function filters out points in $\mathbf{h}_{\mathrm{loc}}$ that are not local maxima, then counts the number of points above the threshold 4γ. The distribution $q(n \,|\, \mathbf{x}) = \delta(n - f_{\mathrm{count}}(\mathbf{h}_{\mathrm{loc}}))$ deterministically takes the value

$f_{\text{count}}(\mathbf{h}_{\text{loc}})$. We fix the maximum number of objects per image to $M = 10$, although in practice this choice was irrelevant for all our experiments, as long as $M \geq 7$.

The locations $\{\mathbf{z}_{\text{loc}}^{(i)}\}_{i=1}^{n}$ are iteratively sampled from a categorical distribution with logits $\mathbf{h}_{\text{loc}}$, where all the logits of the previously sampled location are set to a low value to prevent them from being sampled again. At the ith step, when the ith object's location is sampled, the corresponding feature vector in $\mathbf{h}_{\text{app}}$ is interpreted as the means and log variances of the L components of the ith appearance variable. When sampling from the categorical distribution, we use Gumbel-Softmax for single-sample gradient estimation. The temperature parameter τ is exponentially annealed from 0.5 to 0.02 in 100k steps. The latent space for each appearance variable has $L = 32$ dimensions, and we linearly anneal a β factor for the appearance KL from 0 to 1 in 100k steps.

The sprite-generating function f_{sprite} is a convolutional network that takes as input a sampled appearance vector $\mathbf{z}_{\text{app}}^{(i)}$. The first part of the network consists of a fully connected layer, a convolutional layer, and a bilinear interpolation. The vector $\mathbf{z}_{\text{app}}^{(i)}$ is then expanded and concatenated to the resulting tensor along the channel dimension. The second part of the network consists of 3 residual blocks with 2 convolutional layers each, followed by a final convolutional layer with a sigmoid activation. Leaky ReLU and batch normalization are used after each layer, except in the last residual block. The size of a generated sprite for the multi-MNIST data set is 17×17, for multi-dSprites it is 21×21. The mean of the Bernoulli output is then computed from these sprites as explained above.

We optimized the model with stochastic gradient descent, using Adamax with batch size 64. The initial learning rate was 1e-4 for the inference networks and 1e-3 for the sprite generation network. In the location inference net, the learning rate was exponentially decayed by a factor of 100 in 75k steps. The number of model parameters is in the range 450k–500k, split almost equally among location inference, appearance inference, and sprite generation.

Training warm-up. In practice we found it beneficial to include a *warm-up period* at the beginning of training, in which an auxiliary loss is added to the negative ELBO. We train a fully-convolutional VAE in parallel with our model, and take the location-wise KL in the latent space as a rough proxy for object location, as suggested in [16]. The additional loss is the squared distance between $\mathbf{h}_{\text{loc}}$ and the KL map. The network inferring object location is therefore initially encouraged to mimic a function that is a (rather rough) approximation of the location, and then it is fine-tuned. Intuitively, we are biasing sampling of $\mathbf{z}_{\text{loc}}$ in favor of information-rich locations, which significantly speeds up and stabilizes training. To stabilize training, we also found it beneficial to randomly force $n = 1$ during training with a probability that is 1 during warm-up and then linearly decays to 0.1.

The auxiliary VAE is implemented as a fully convolutional network, in which the encoder consists of 3 residual blocks with downsampling between blocks, and a final convolutional layer. The decoder loosely mirrors the encoder. The latent variables are organized as a 3D tensor with size $11 \times 11 \times 8$. The auxiliary loss is added to the original training loss for a warm-up phase of 30k steps. In the subsequent 30k, the contribution of this term to the overall loss is linearly annealed to 0.

Rescaling location logits. Assume we have k objects and the logits of the inferred location distribution are either 0 or h. The probability of one of the "hot" locations after softmax is $e^h/(D - k + ke^h) = t$ where t is a constant that should not depend on D, and should be close to $1/k$. Solving for h we get $h = \log t + \log(D - k) - \log(1 - kt)$. If D is large enough and k is small enough, we have $\log(D - k) \approx \log D$, and the constant $\log t$ is relatively small in magnitude. Because of our assumptions on the logits and on t, we can write $\log(1 - kt) \approx 0$, and therefore the high logits h should be approximately proportional to $\log D$. Thus, using the same fully convolutional network architecture for multiple image sizes, the logits should be scaled by a factor proportional to $\log D$. The threshold for counting objects should likewise follow this rule of thumb.

Baseline implementation. As baseline we use a fully convolutional β-VAE, where the latent variables are organized as a 3D tensor. Being closer in spirit to our method, this leads to a fairer comparison. Furthermore, this inductive bias makes it easier to preserve spatial information. Indeed, this empirically leads to better likelihood and generated samples than a VAE with fully connected layers (and a 1D structure for latent variables), and allows to model more naturally a varying number of objects in a scene. The encoder consists of a convolutional layer with stride 2, followed by 4 residual blocks with 2 convolutional layers each. Between the first two pairs of blocks, a convolutional

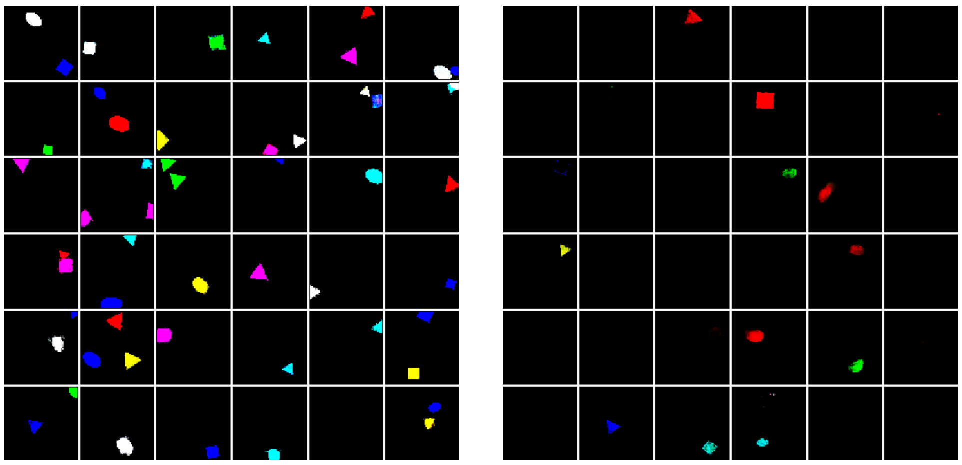
Figure 4: **Generated samples.** Images generated by LAVAE (left) from its prior $p(\mathbf{z}_{\text{loc}} \mid n)p(\mathbf{z}_{\text{app}} \mid n)p(n)$, where $p(n)$ is learned from data, and by the baseline (right) from its prior $p(\mathbf{z})$.

layer with stride 2 reduces dimensionality. The resulting tensor has size $64 \times 8 \times 8$, and a final convolutional layer brings the number of channels to $2d_{\text{VAE}}/8^2$ where d_{VAE} is the number of latent variables. The decoder architecture takes as input a sample $\mathbf{z}$ of size $d_{\text{VAE}}/8^2 \times 8 \times 8$ and outputs the pixel-wise Bernoulli means. Its architecture loosely mirrors the encoder's, with convolutions being replaced by transposed convolutions, except for the last upsampling operation which consists of bilinear interpolation and ordinary convolution. All convolutions and transposed convolutions, except for the ones between two residual blocks, are followed by Leaky ReLU and batch normalization. The last convolutional layer is only followed by a sigmoid nonlinearity. In our experiments we used $d_{\text{VAE}} = 1024$ and we linearly annealed β from 0 to 1 in 100k steps. The number of model parameters is about 1M.

D Additional experiments

D.1 Experiments on multi-dSprites

Figure 4 shows images generated by sampling from the prior in LAVAE and in the VAE baseline, where we can appreciate how our model accurately captures the number and diversity of objects in the data set. Figure 5 shows instead an example of inference on test images. Finally, as we did for multi-MNIST, we performed disentanglement experiments where, starting from the latent representation of a test image, we manipulate the latent variables. In Figure 6 we change the order of the location variables to swap objects, whereas Figure 7 shows the effect of separately altering the location and appearance latent variables of a single object.

D.2 Generalizing to more objects

LAVAE correctly infers the number of objects in images from the 7-object versions of our data sets, despite the fact that it was only trained on images with up to 3 objects. Furthermore, by representing each object independently and disentangling location and appearance of each object, it accurately decomposes 7-object scenes and allows intervention as easily as in images with fewer objects. Figure 8 demonstrates this on the 7-object version of the multi-MNIST-10k data set. Finally, in Figure 9 we show images generated by LAVAE after modifying the prior $p(n)$ to be uniform in $\{4, 5\}$.

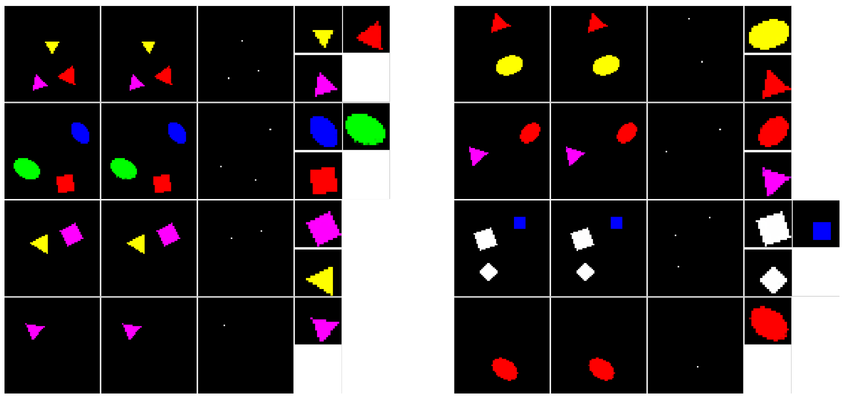

Figure 5: Example of **inference and reconstruction** on multi-dSprites test images. From left to right: input, reconstruction, summary of inferred locations, sprites generated from the inferred appearance latent variables.

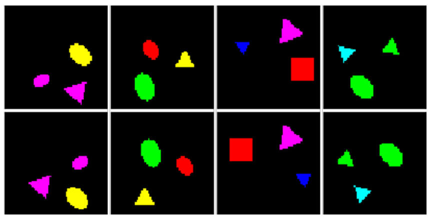

Figure 6: **Object swap.** Changing the order of location (or equivalently appearance) latent variables leads to objects being swapped. Top row: original reconstruction of test image; bottom row: objects are swapped by manipulating the latent variables.

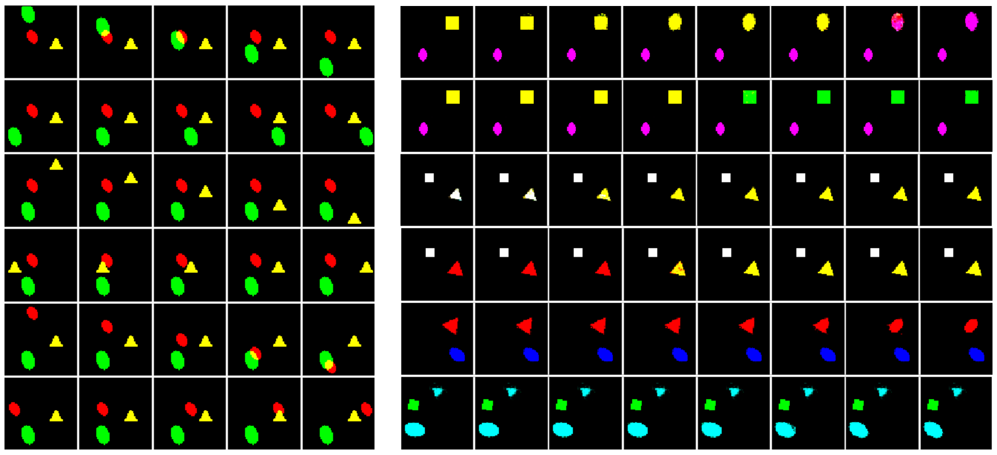

Figure 7: Left: **Location traversal.** Latent traversal on location variables in a test image with 3 objects. Right: **Appearance traversal.** Latent traversal on appearance variables: changing $\mathbf{z}_{\text{app}}^{(i)}$ for some i along one latent dimension corresponds to changing appearance attributes of one specific object. The appearance latent space is only partially disentangled. Here we show examples where a change in one latent dimension leads to a change of a single factor of variation (rows 2, 4, 5, 6). However, in row 1 there is a change both in color and shape, and in row 3 both in color and scale.

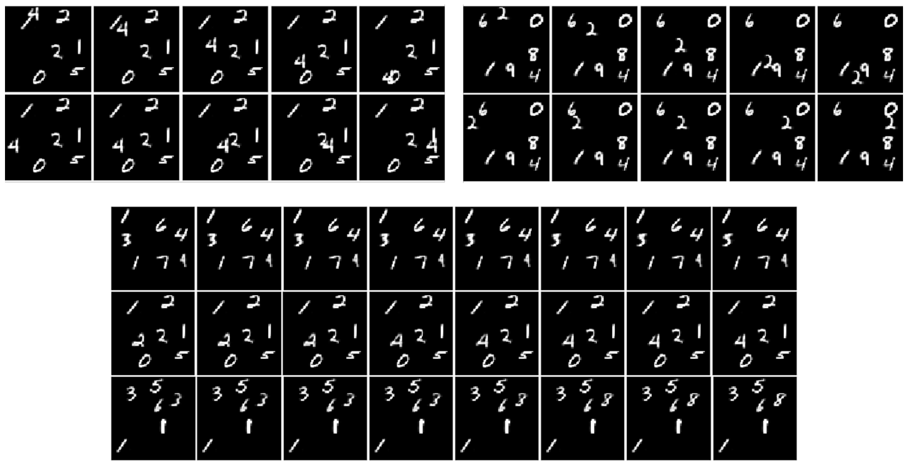

Figure 8: **Disentanglement with more objects.** Latent traversal on location (top) and appearance (bottom) variables, on multi-MNIST-10k test images containing 7 objects. LAVAE can still correctly infer the scene's structure and reconstruct it, allowing intervention on location or appearance of single objects.

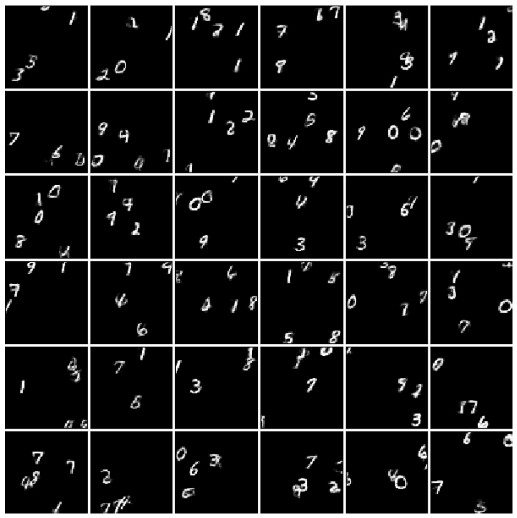

Figure 9: **Generation with out-of-distribution number of objects.** Images generated by LAVAE from a modified prior in which n takes value 4 or 5 with probability $1/2$.