Supplementary Material for: Duplicate Latent Representation Suppression for Multi-object Variational Autoencoders

Li Nanbo
nanbo.li@ed.ac.uk
Robert B. Fisher
rbf@inf.ed.ac.uk

School of Informatics
University of Edinburgh
Edinburgh, UK

A. Implementation Details

Training specifications We refer to Table 1, 2 & 3 to the training configurations of MONet, IODINE and MulMON respectively. Note that 1) for IODINE adn MulMON that use iterative inference modules, we apply LDS per iterative step to compute their ELBOs during training, and 2) for all CompVAEs, we apply LDS only in their training times.

Model Architecture Specifications As discussed in the main paper, we use three existing CompVAE models as our baselines and build our contributions on top of these architectures. It is important to use the same architectures as the that of the original papers. However, we found it difficult to use a latent dimension of 64 as in [5] for the CLEVR-based datasets as it trains too slow, over one week for one run on two RTX2080TI, we thus reduced the dimension of IODINE to 16 for our IODINE. As constructing the proposed *LDS* prior requires no model architecture design and architecture parameter tweaking, we refer to the original papers of MONet [2], IODINE [5], and MulMON [6] for the architecture details.

Table 1: Training Configurations For MONet

TYPE	THE TRAININGS OF MONET^0 AND MONET$^+$
OPTIMIZER	RMSPROP
INITIAL LEARNING RATE η_0	$3e^{-4}$
BATCH SIZE	40 (UNIT: IMAGES)
LEARNING RATE AT STEP s	N/A
TOTAL GRADIENT STEPS	$600k$
GRADIENT-NORM CLIPPING	5.0
LOG-NORMAL LIKELIHOOD STRENGTH	1.0
KL (GAUSSIAN PRIOR) STRENGTH β	0.5
KL (ATTENTION PRIOR) STRENGTH	0.5
LDS (MONET$^+$ ONLY) STRENGTH	0.5

Table 2: Training Configurations of IODINE^0 and IODINE$^+$

TYPE	THE TRAININGS OF IODINE^0 AND IODINE$^+$
OPTIMIZER	ADAM
INITIAL LEARNING RATE η_0	$1e^{-4}$
BATCH SIZE	8
LEARNING RATE AT STEP s	$\star\max\{0.1\eta_0 + 0.9\eta_0 \cdot (1.0 - s/1e^6), 0.1\eta_0\}$
TOTAL GRADIENT STEPS	$600k$
GRADIENT-NORM CLIPPING	5.0
INFERENCE ITERATIONS [5]	5
LOG-NORMAL LIKELIHOOD STRENGTH	1.0
KL (GAUSSIAN PRIOR) STRENGTH β	1.0
LDS (IODINE$^+$ ONLY) STRENGTH	1.0

$\star$: SAME SCHEDULER AS GQNS'.

Table 3: Training Configurations of MulMON^0 and MulMON$^+$

TYPE	THE TRAININGS OF MULMON^0 AND MULMON$^+$
OPTIMIZER	ADAM
INITIAL LEARNING RATE η_0	$2e^{-4}$
BATCH SIZE	8
LEARNING RATE AT STEP s	$\star\max\{0.1\eta_0 + 0.9\eta_0 \cdot (1.0 - s/1e^6), 0.1\eta_0\}$
TOTAL GRADIENT STEPS	$600k$
GRADIENT-NORM CLIPPING	5.0
INFERENCE ITERATIONS [5]	5
LOG-NORMAL LIKELIHOOD STRENGTH	1.0
KL (GAUSSIAN PRIOR) STRENGTH β	1.0
LDS (IODINE$^+$ ONLY) STRENGTH	1.0

$\star$: SAME SCHEDULER AS GQNS'.

B. CompVAE Rendering Process

Figure 1 shows the CompVAE rendering process we used to produce all qualitative results presented in this paper. **Importantly, we used softmax functions to compute the compositional probabilities of each components, i.e. the mixing probabilities in Eqn.(1), to render the whole scene, and sigmoid functions to render independent objects**. However, one might also see independent component rendering with other functions in the related literature, e.g. IODINE [5] uses a linear mapping of x_k to render independent components.

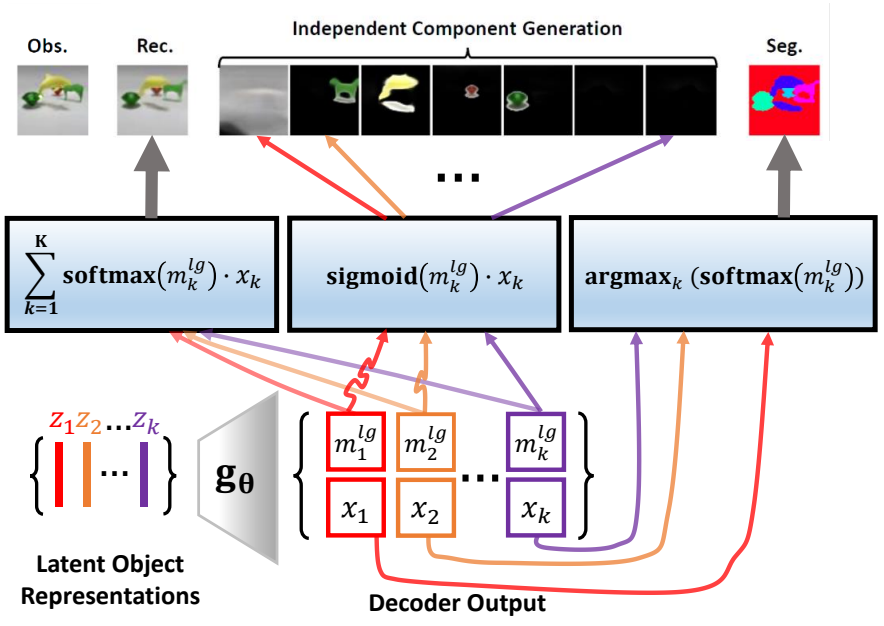

Figure 1: Overview of a CompVAE rendering process. The rendering process starts by inputting a set of inferred latent object representations (**Bottom left**) into the generator network g_θ. The generator g_θ outputs a raw mask ($m_k^{lg} \in \mathbb{R}^{H \times W \times 1}$) and a color pool ($x_k \in \mathbb{R}^{H \times W \times 3}$) (**Bottom middle**). The decoder output is then passed into three different functions (**Middle row**) to get different render results (**Top row**). All computations are defined pixel-wise but executed in parallel.

C. Additional Results

C.1 Abalation Study

The ablation study focuses on two hyperparameters: 1) the standard deviation σ used in the *LDS* prior (see Section 3.2 of the main paper) and 2) the number of object slots K. The former relates to the precision of the similarity measure and the latter determines the size of the similarity matrix constructed in the *LDS* computation, i.e. it relates to the scalability of *LDS*. We do the ablation study with only MONet and on only the CLE-MV dataset for computation efficiency. We select 4 different σ to train MONet and compare their performance on the scene reconstruction and the scene factorization tasks. Figure 2 shows no significant performance loss in tasks by changing σ from the default value, 0.1, to other values. A future investigation will be further increasing σ until it is sufficiently close to a uniform distribution and thus breaks the *LDS* prior. Moreover, the performance might get boosted in some cases. For the object-slot quantity K, we first train MONet with $K = 7$ and $K = 9$ respectively and test them with 7,9, 11, 15 object slots. Figure 2 shows: 1) the models trained with $K = 7$ and $K = 9$ have very similar performance in both tasks and 2) testing with a different K does not cause a significant performance drop.

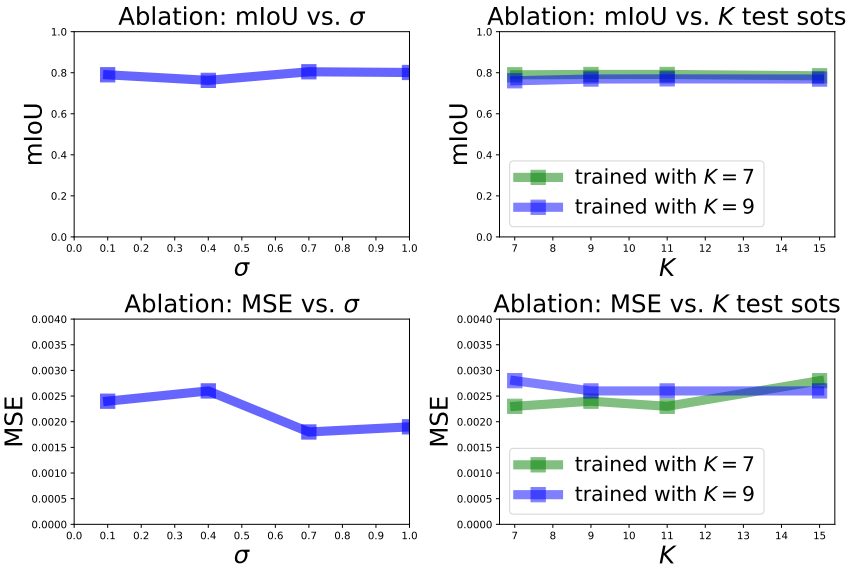

Figure 2: Ablation study results. **Top left** Scene decomposition performance vs. *LDS* prior precision (σ). **Top right** Scene decomposition performance vs. the number of object slots used in training and testing (K). **Bottom left** Scene observation reconstruction performance vs. *LDS* prior precision (σ). **Bottom right** Scene observation reconstruction performance vs. the number of object slots used in training and testing (K).

C.2 GENESIS on the CLE-MV Data

We tested GENESIS [4] on the CLE-MV data to assess how well the inference redundancy problems are handled by the autoregressive model of GENESIS. The experiment was conducted on top of the official implementation of GENESIS [1] with strict abidance of its original hyperparameter configurations. However, as shown in Figure 3, GENESIS failed to factorise CLE-MV scenes correctly—it treats a CLE-MV scene observation (i.e. an image) as a big and flat object that contains all the content. As a result, it produces wrong image segmentation. A possible reason could be that GENESIS represents the autoregressive conditioning of object discovery in the latent space (i.e. $z_k \,|\, z_{1:k-1}$) instead of the image space as that of MONet—a successive object mask conditions directly on all the previous obtained masks (i.e. $m_k \,|\, m_{1:k-1}$). According to [3], this could introduce more severe global information leaking issue. In general, future study is needed to better understand the practical limitations and their causes in GENESIS.

C.3 Real-image Experiments

To demonstrate that the proposed LDS can efficiently perform duplicate suppression on real images, we conducted comparison experiments between CompVAEs that are trained with and without LDS priors on the a collected real-image dataset.

Real-image Dataset We created such dataset by randomly placing $2-4$ cubes (of different colours) on white table top and taking photos with a webcam that is mounted on a moving robot arm. We created 109 scenes in total and for each scene we captured $20-30$

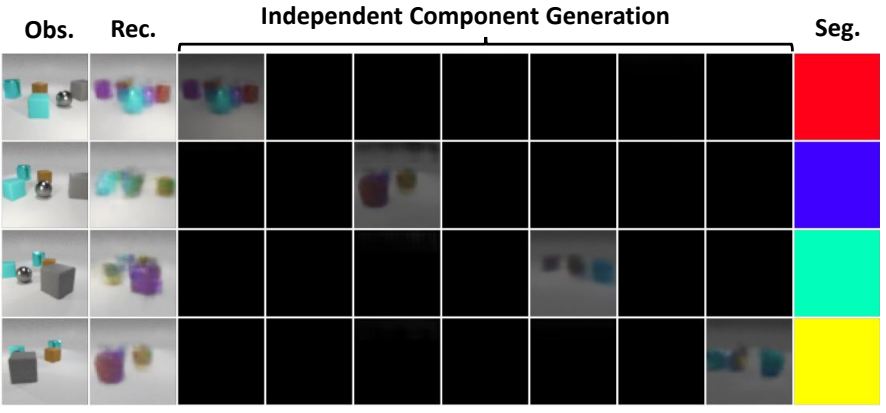

Figure 3: Qualitative results of GENESIS on the CLE-MV dataset.

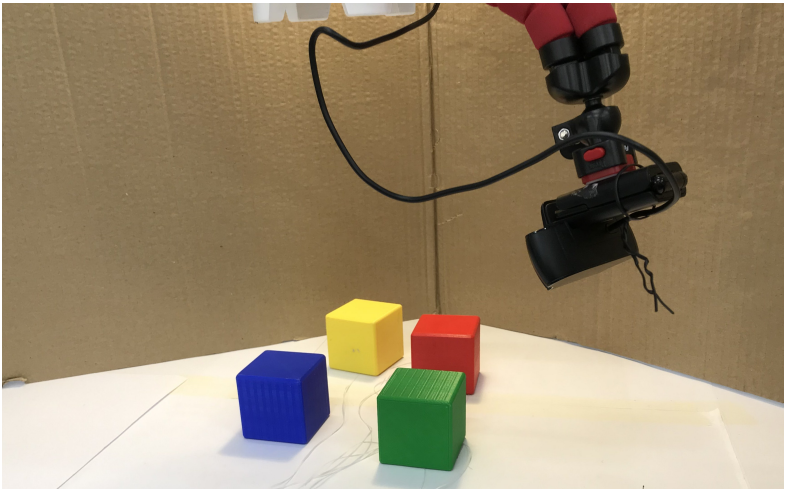

Figure 4: Hardware platform for real-image dataset recording.

images from different viewing angles. We show the hardware platform setup in Figure 4.

Results Figure 5 shows that the original MONet^{0*} infers redundant white table components. Although MONet^{+} demonstrates a slight performance drop in handling occlusions (e.g. renders the independent table component worse than MONet^{0*}), it does suppress the duplicate table finding issues of MONet^{0*}. Also, we see that MONet^{+} produces cleaner segmentation results than MONet^{0*}. Compared with synthetic data, real images often come from complex distributions and thus exhibit significant larger pixel variances (due to uncontrolled lighting, materials, etc.), complicating the training of a generative model. This also explains why neither MONet^{0*} nor MONet^{+} model the independent table (always partially occluded) distribution properly. In conclusion, LDS is an effective addition to CompVAEs on real data and can potentially serve as a useful tool in some real applications.

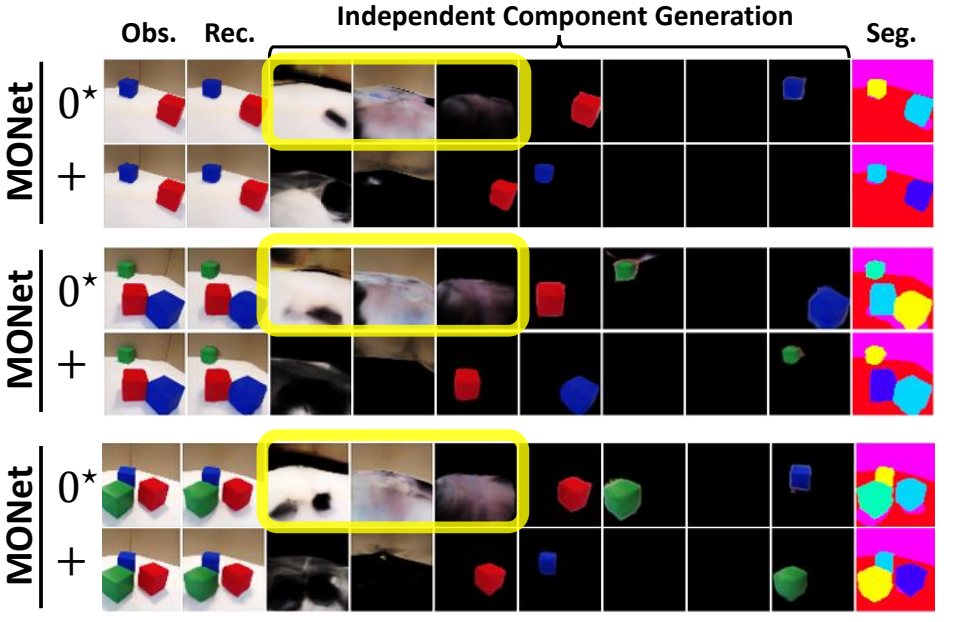

Figure 5: Qualitative results of MONet on real images. Symbols "0*" and "+" tag models that trained with and without LDS respectively. Yellow circles highlight duplicated or partially duplicated components.

References

[1] GENESIS official implementation. https://github.com/applied-ai-lab/genesis.git. Accessed: 2021-07-01.

[2] 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.

[3] Patrick Emami, Pan He, Sanjay Ranka, and Anand Rangarajan. Efficient iterative amortized inference for learning symmetric and disentangled multi-object representations. In *ICML, 2021*.

[4] Martin Engelcke, Adam R. Kosiorek, Oiwi Parker Jones, and Ingmar Posner. GENESIS: Generative Scene Inference and Sampling of Object-Centric Latent Representations. In *ICLR, 2020*.

[5] Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In *ICML, 2424–2433, 2019*.

[6] Li Nanbo, Cian Eastwood, and Robert B Fisher. Learning object-centric representations of multi-object scenes from multiple views. In *NeurIPS, 2020*.