Learning Object-Centric Video Models by Contrasting Sets

Sindy Löwe[2,*], Klaus Greff[1], Rico Jonschkowski[1], Alexey Dosovitskiy[1], and Thomas Kipf[1]

[1]Google Research, Brain Team
[2]UvA-Bosch Delta Lab, University of Amsterdam

Abstract

Contrastive, self-supervised learning of object representations recently emerged as an attractive alternative to reconstruction-based training. Prior approaches focus on contrasting individual object representations (slots) against one another. However, a fundamental problem with this approach is that the overall contrastive loss is the same for (i) representing a different object in each slot, as it is for (ii) (re-)representing the same object in all slots. Thus, this objective does not inherently push towards the emergence of object-centric representations in the slots. We address this problem by introducing a global, set-based contrastive loss: instead of contrasting individual slot representations against one another, we aggregate the representations and contrast the joined sets against one another. Additionally, we introduce attention-based encoders to this contrastive setup which simplifies training and provides interpretable object masks. Our results on two synthetic video datasets suggest that this approach compares favorably against previous contrastive methods in terms of reconstruction, future prediction and object separation performance.

1 Introduction

Object-centric approaches, for which a scene is represented by a set of object variables (called *slots*), can greatly improve generalization to new situations in an environment or video [1–4]. Approaches that explicitly model objects can re-use and transfer learned knowledge about the dynamics of individual objects and their interactions, even if the composition of objects in the scene undergoes significant changes. Recently, contrastive losses have achieved promising results for a variety of image-level tasks [5–7], but have received far less attention for learning object-centric representations. All prior approaches to contrastive object discovery [8–11] rely on per-slot losses, which have a fundamental flaw: slotwise contrastive losses cannot differentiate between representations in which (i) a different object is represented in each slot or in which (ii) the same object is (re-)represented in all slots. As a result, this objective does not inherently enforce diverse slots representations, and needs to rely on additional cues such as object interactions or on explicit regularization.

We expose this issue and propose a solution in terms of a global set contrastive loss (SetCon). Instead of contrasting slot representations directly against one another, we aggregate the set of slots into a global scene representation. This global contrasting approach encourages better coordination between slots, as they are forced to jointly represent the entire scene. We further introduce attention-based encoders to this domain using Slot Attention [12], which simplifies training and provides interpretable object masks. We evaluate our approach on two synthetic video datasets, where we measure performance in terms of the ability of a separately trained decoder to reconstruct either the current time-step or the predicted future time-step, and its ability to separate and locate the different objects present in the scene.

[*]Work done while interning at Google, Contact: `loewe.sindy@gmail.com`

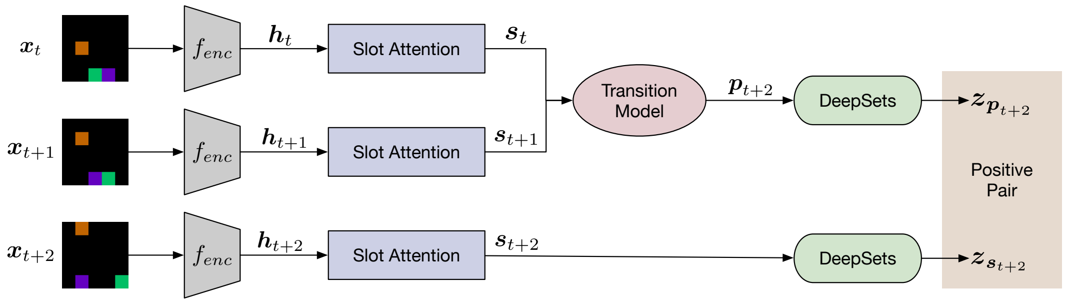


Figure 1: Set Contrastive (SetCon) Slot Attention model.

2 Method

In this section, we outline the four modules (Backbone, Slot Attention, Transition Model and Set Encoder) that make up the SetCon Slot Attention model (Fig. 1) and describe the proposed Set Contrastive loss for learning object-centric video representations.

Backbone We apply an encoding backbone f_{enc} to an input frame $\boldsymbol{x}_t$ at time step t, consisting of a stack of convolutional layers with ReLUs and a linear position embedding: $\boldsymbol{h}_t = f_{\text{enc}}(\boldsymbol{x}_t)$.

Slot Attention The backbone is followed by a Slot Attention module [12], $\boldsymbol{s}_t = \text{SlotAttention}(\boldsymbol{h}_t)$, applied to hidden representations $\boldsymbol{h}_t \in \mathbb{R}^{N \times D_{\text{enc}}}$, where N corresponds to the number of pixels in the (flattened) image, i.e. width$\times$height, and D_{enc} is the representation size per pixel. Slot Attention is an iterative attention mechanism that produces K slots $\boldsymbol{s}_t = [\boldsymbol{s}_t^1, ..., \boldsymbol{s}_t^K]$, which can represent individual objects within the input. The two most important steps within each iteration are: (i) slots compete through a softmax attention mechanism that is normalized over the slot dimension and (ii) the final representations are created by aggregating inputs with a weighted mean with the attention matrix $\boldsymbol{a}_t$ acting as weights:

$$\boldsymbol{a}_t = \text{Softmax}\left(\frac{1}{\sqrt{D}} k(\boldsymbol{h}_t) \cdot q(\boldsymbol{c})^T\right) \in \mathbb{R}^{N \times K}, \quad \boldsymbol{u}_t = \text{WeightedMean}\Big(\boldsymbol{a}_t, v(\boldsymbol{h}_t)\Big) \in \mathbb{R}^{K \times D}, \quad (1)$$

where k, q, v are linear, learnable projections that map to a common dimension D. We use a small feedforward network $\boldsymbol{s}_t = f_{\text{ffw}}(\boldsymbol{u}_t)$ to arrive at slot representations $\boldsymbol{s}_t \in \mathbb{R}^{K \times D}$. To simplify training, we do not initialize the slot representations randomly; instead, each slot is initialized with a learned value $\boldsymbol{c} \in \mathbb{R}^{K \times D}$, and we apply only one iteration of the Slot Attention mechanism.

Transition Model We apply a transition model with shared parameters to each of the slot representations $\boldsymbol{s}_t^k$. The transition model learns to predict the representations at the next time-step: $\boldsymbol{p}_{t+2}^k = \boldsymbol{s}_{t+1}^k + f_{\text{transition}}([\boldsymbol{s}_t^k, \boldsymbol{s}_{t+1}^k, \boldsymbol{s}_{t+1}^k - \boldsymbol{s}_t^k])$, where $[\cdot]$ denotes concatenation of vectors. The transition function $f_{\text{transition}}$ consists of three steps: a linear down-projection, LayerNorm [13] and a linear transformation.

Set Encoder We create a global image representation by applying a DeepSets [14] model on the slots and future slot predictions: $\boldsymbol{z}_{\boldsymbol{s}_t} = f_{\text{mlp}}\left(\text{LayerNorm}\left(\sum_{k=1}^K (f_{\text{mlp}}(\boldsymbol{s}_t^k))\right)\right) \in \mathbb{R}^D$. Similarly, $\boldsymbol{z}_{\boldsymbol{p}_t} = \text{DeepSets}(\boldsymbol{p}_t)$. Thus, the loss that is applied on $\boldsymbol{z}$ is applied to the aggregated *set* of slots.

Set Contrastive Loss For the training of our model, we use an inner product between the vector representations as scoring function g and the InfoNCE loss [15]:

$$\mathcal{L}^i = -\mathbb{E}_{\boldsymbol{X}}\left[\log \frac{g(\boldsymbol{z}_{\boldsymbol{p}_t}^i, \boldsymbol{z}_{\boldsymbol{s}_t}^i)}{\sum_{\boldsymbol{z}_{\boldsymbol{s}_{t'}}^j \in \mathcal{B}} g(\boldsymbol{z}_{\boldsymbol{p}_t}^i, \boldsymbol{z}_{\boldsymbol{s}_{t'}}^j) + \sum_{\boldsymbol{z}_{\boldsymbol{p}_{t'}}^j \in \mathcal{B}} g(\boldsymbol{z}_{\boldsymbol{p}_t}^i, \boldsymbol{z}_{\boldsymbol{p}_{t'}}^j)}\right], \quad (2)$$

with $g(\boldsymbol{z}_{\boldsymbol{p}_t}, \boldsymbol{z}_{\boldsymbol{s}_t}) = \exp(\boldsymbol{z}_{\boldsymbol{p}_t}^T \cdot \boldsymbol{z}_{\boldsymbol{s}_t} / \tau)$, where τ is a temperature constant. i, j refer to the samples within the dataset $\boldsymbol{X}$. The positive pair $(\boldsymbol{z}_{\boldsymbol{p}_t}^i, \boldsymbol{z}_{\boldsymbol{s}_t}^i)$ contains the globally aggregated predicted representation and the slot representation of sample i at time-step t. We use both the globally aggregated predicted representations and slot representations $\boldsymbol{z}_{\boldsymbol{p}_{t'}}^j, \boldsymbol{z}_{\boldsymbol{s}_{t'}}^j$ taken from all time-steps of all sequences within the batch $\mathcal{B}$ as negative samples. Since this loss operates on *sets* of representations, we call it the Set Contrastive Loss (SetCon).

Table 1: MSE and ARI scores ($\times 1\mathrm{e}{-2}$; mean $\pm$ standard error for 3 seeds) for unsupervised object discovery in multi-object datasets. *we omit ARI on the GridWorld dataset as the score is not well behaved on single pixel segmentation masks.

	Loss	Encoder	Future Prediction $\boldsymbol{p}_t$		Slot Representation $\boldsymbol{s}_t$	
			MSE	ARI	MSE	ARI
GridWorld	Reconstruction	FM-MLP	0.828 $\pm$ 0.367	*	0.014 $\pm$ 0.004	*
		Slot Attention	3.104 $\pm$ 1.056	*	0.450 $\pm$ 0.383	*
	Slotwise	FM-MLP	5.507 $\pm$ 2.576	*	3.685 $\pm$ 2.758	*
		Slot Attention	5.123 $\pm$ 2.338	*	4.199 $\pm$ 3.079	*
	SetCon	FM-MLP	3.139 $\pm$ 0.127	*	0.525 $\pm$ 0.134	*
		Slot Attention	**1.214** $\pm$ 0.245	*	**0.072** $\pm$ 0.064	*
Bouncing Balls	Reconstruction	FM-MLP	2.708 $\pm$ 1.073	68.4 $\pm$ 15.0	0.784 $\pm$ 0.495	66.6 $\pm$ 14.5
		Slot Attention	4.077 $\pm$ 0.663	92.8 $\pm$ 2.2	0.142 $\pm$ 0.042	85.2 $\pm$ 0.5
	Slotwise	FM-MLP	8.229 $\pm$ 1.128	39.0 $\pm$ 10.0	4.518 $\pm$ 0.574	35.2 $\pm$ 5.1
		Slot Attention	7.394 $\pm$ 1.241	14.3 $\pm$ 12.2	2.866 $\pm$ 0.639	16.8 $\pm$ 9.5
	SetCon	FM-MLP	9.334 $\pm$ 0.891	49.6 $\pm$ 10.5	5.331 $\pm$ 0.472	38.7 $\pm$ 11.1
		Slot Attention	**5.541** $\pm$ 0.811	**86.8** $\pm$ 2.2	**0.807** $\pm$ 0.033	**75.3** $\pm$ 3.8

3 Related Work

Object Discovery Most unsupervised approaches to object discovery or object-centric representation learning focus on models trained with reconstruction losses in pixel space [1, 12, 16–28]. Out of these approaches, Tagger [16], NEM [18], R-NEM [1], IODINE [21], MONET [22], GENESIS [23], and Slot Attention [12] are most closely related to our approach, as they use a set of generic embeddings in the form of slots together with a segmentation mask to represent objects, but different from our approach they rely on a decoder back into pixel space for training. Relying on a decoder together with a loss in pixel space for training can bias the learning process to depend heavily on object size and pose challenges in scenes containing complex textures.

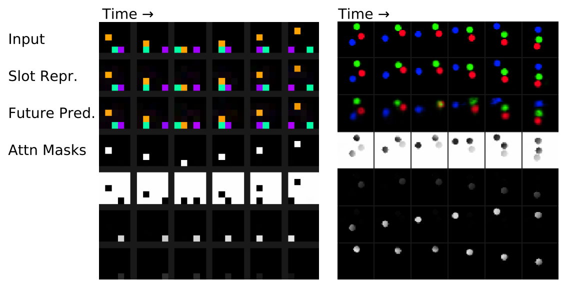
Contrastive Learning Contrastive learning [29–33] has enjoyed increasing popularity in learning image representations [15, 34, 35], with recent methods achieving similar performance as supervised approaches on ImageNet [5–7]. A recent line of work [8–11], starting with the C-SWM model [8], explores the use of contrastive objectives for object discovery. Similar to our model, these approaches use transformations from temporal data as learning signal for discovering individual objects, as common transformations on static images (such as rotation or flipping) are likely unsuitable for this task. Different from our SetCon model, these approaches apply a contrastive loss solely on the slot level, which does not establish communication between slots and hence can suffer from failure modes where objects are ignored or represented multiple times in multiple slots.

4 Experiments

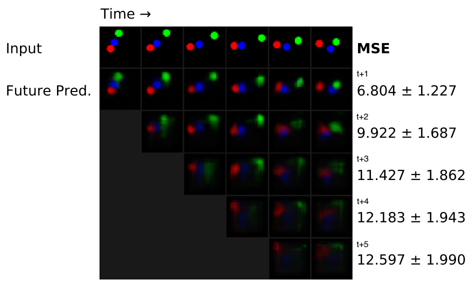
In this section, we first describe the datasets we use in our experiments, outline our evaluation methods and describe our results. Additional details about our experiments can be found in Appendix A.

Datasets We evaluate our method on two synthetic video datasets: Multi-Object GridWorld and Bouncing Balls. Both datasets describe the movement of three visually distinct objects on a two-dimensional, black background, and reuse the same three objects throughout all sequences. In the **Multi-Object GridWorld**, colored pixels represent objects and are restricted to move in four directions: up, down, left and right. Pixels pass through each other, overlapping one another in random order. In the **Bouncing Balls** dataset, colored balls move in continuous space, bouncing against one another. In both datasets, the objects are reflected by image boundaries.

Evaluation We evaluate the performance of our models by training an IODINE spatial broadcast decoder [21, 36] on top of the learned representations without propagating the gradients to the encoder and transition model. In line with previous work [12, 21], this decoder decodes each slot individually into four channels, representing RGB colors and an unnormalized alpha mask. For the final output, the alpha masks are normalized across slots and used to combine the slotwise reconstructions. We apply



(a)



(b)

Figure 2: Qualitative Results. **(a)** Object separation and reconstruction quality for the SetCon Slot Attention model. **(b)** Reconstruction performance (MSE) when predicting up to five time-steps into the future (from top to bottom) on bouncing balls.

this decoder both on the slot representations $\boldsymbol{s}_t$ and the predictions $\boldsymbol{p}_t$ created by the transition model, and measure two metrics: **1) Mean Squared Error (MSE)** between the reconstruction created by the decoder and the input $\boldsymbol{x}_t$ of the model. **2) Adjusted Rand Index (ARI)** [37, 38] between the normalized alpha masks of the decoder and the ground truth object segmentation (excl. background). ARI measures clustering similarity, and ranges from 0 (random) to 1 (perfect match).

Baselines We compare the SetCon Slot Attention model against several model variants, where we combine three different losses with two different encoder architectures. Next to the SetCon loss, we apply a slotwise contrastive loss similar to previous work [8–11]. For this loss, we replace the set representations $\boldsymbol{z}_{\boldsymbol{s}_t}, \boldsymbol{z}_{\boldsymbol{p}_t}$ in Eq. (2) with the slot and transition model representations $\boldsymbol{s}_t, \boldsymbol{p}_t$, and contrast against the respective representations across samples and time-steps in the batch, so that each slot is paired with the slot of the same index in a different time-step or sample. Additionally, we compare these two losses against the "ground-truth" approach of using the reconstruction error of the decoder for training the entire model. Next to the Slot Attention model [12], we also test a C-SWM-like model [8] for creating object-centric representations, which we coin FeatureMap MLP (FM-MLP). After the encoding backbone, it creates object representations by applying a linear layer with dimensionality corresponding to the number of objects and treating these features as slots.

Results As shown in Table 1, SetCon with Slot Attention outperforms all other contrastive approaches across all metrics, and approaches the performance of the reference models trained with the reconstruction loss. For the FM-MLP model, we do not observe a significant performance difference between the SetCon and the slotwise contrastive loss on the Bouncing Balls dataset. In all other settings (FM-MLP model on the GridWorld, Slot Attention on both datasets), the SetCon loss consistently outperforms the slotwise contrastive loss, indicating that it does indeed enforce better coordination between slots and thus an improved object-centric representation. Fig. 2a shows qualitative results of SetCon with Slot Attention. It becomes apparent from the attention masks (*Attn Masks*) that the model learns to attend to different objects within the different slots[1].

Future Prediction In Fig. 2b, we investigate whether our model can predict up to 5 time-steps into the future by applying the transition model iteratively without encoding a new input frame. We find that prediction quality can degrade quickly, as the model has only been trained for single-step prediction and has no explicit constraint to enforce the slot and transition model representations $\boldsymbol{s}_t, \boldsymbol{p}_t$ to lie nearby in latent space. This is an interesting avenue for future work.

5 Conclusion

Our results indicate that global, set-based contrasting (SetCon) can be an effective alternative learning paradigm for object-centric video models. While our exploration focuses on extremely simplistic environments and merely scratches the surface of this class of models, we are hopeful that further exploration will resolve current limitations for predicting further into the future, for disambiguating object instances with the same appearance, and will ultimately facilitate effective modeling of visually rich and diverse scenes, without having to rely on pixel-based losses.

[1]Note that on the GridWorld data the model seems to represent the third object within the "background" slot.

Acknowledgments

We would like to thank Jean-Baptiste Cordonnier, Francesco Locatello, Aravindh Mahendran, Thomas Unterthiner, Jakob Uszkoreit and Dirk Weissenborn for helpful discussions, and Marvin Ritter and Andreas Steiner for technical support.

References

[1] 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.

[2] Tejas D Kulkarni, Ankush Gupta, Catalin Ionescu, Sebastian Borgeaud, Malcolm Reynolds, Andrew Zisserman, and Volodymyr Mnih. Unsupervised learning of object keypoints for perception and control. In *Advances in Neural Information Processing Systems*, pages 10723–10733, 2019.

[3] Chen Sun, Per Karlsson, Jiajun Wu, Joshua B Tenenbaum, and Kevin Murphy. Stochastic prediction of multi-agent interactions from partial observations. *arXiv preprint arXiv:1902.09641*, 2019.

[4] Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B Tenenbaum. CLEVRER: Collision events for video representation and reasoning. *arXiv preprint arXiv:1910.01442*, 2019.

[5] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[7] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020.

[8] Thomas Kipf, Elise van der Pol, and Max Welling. Contrastive learning of structured world models. *arXiv preprint arXiv:1911.12247*, 2019.

[9] Evan Racah and Sarath Chandar. Slot contrastive networks: A contrastive approach for representing objects. *arXiv preprint arXiv:2007.09294*, 2020.

[10] Qian Huang, Horace He, Abhay Singh, Yan Zhang, Ser-Nam Lim, and Austin Benson. Better set representations for relational reasoning. In *Advances in Neural Information Processing Systems*, 2020.

[11] Anonymous. Systematic evaluation of causal discovery in visual model based reinforcement learning. In *Submitted to International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=gp5Uzbl-9C-`. under review.

[12] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *arXiv preprint arXiv:2006.15055*, 2020.

[13] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[14] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in Neural Information Processing Systems*, pages 3391–3401, 2017.

[15] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[16] Klaus Greff, Antti Rasmus, Mathias Berglund, Tele Hao, Harri Valpola, and Jürgen Schmidhuber. Tagger: Deep unsupervised perceptual grouping. In *Advances in Neural Information Processing Systems*, pages 4484–4492, 2016.

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

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

[19] Charlie Nash, SM Ali Eslami, Chris Burgess, Irina Higgins, Daniel Zoran, Theophane Weber, and Peter Battaglia. The multi-entity variational autoencoder. In *NIPS Workshops*, 2017.

[20] Adam Kosiorek, Hyunjik Kim, Yee Whye Teh, and Ingmar Posner. Sequential attend, infer, repeat: Generative modelling of moving objects. In *Advances in Neural Information Processing Systems*, pages 8606–8616, 2018.

[21] 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 *International Conference on Machine Learning*, pages 2424–2433, 2019.

[22] 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.

[23] Martin Engelcke, Adam R Kosiorek, Oiwi Parker Jones, and Ingmar Posner. GENESIS: Generative scene inference and sampling with object-centric latent representations. *arXiv preprint arXiv:1907.13052*, 2019.

[24] Karl Stelzner, Robert Peharz, and Kristian Kersting. Faster attend-infer-repeat with tractable probabilistic models. In *International Conference on Machine Learning*, pages 5966–5975, 2019.

[25] Eric Crawford and Joelle Pineau. Spatially invariant unsupervised object detection with convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3412–3420, 2019.

[26] Jindong Jiang, Sepehr Janghorbani, Gerard de Melo, and Sungjin Ahn. Scalable object-oriented sequential generative models. *arXiv preprint arXiv:1910.02384*, 2019.

[27] Zhixuan Lin, Yi-Fu Wu, Skand Vishwanath Peri, Weihao Sun, Gautam Singh, Fei Deng, Jindong Jiang, and Sungjin Ahn. SPACE: Unsupervised object-oriented scene representation via spatial attention and decomposition. *arXiv preprint arXiv:2001.02407*, 2020.

[28] Rishi Veerapaneni, John D Co-Reyes, Michael Chang, Michael Janner, Chelsea Finn, Jiajun Wu, Joshua Tenenbaum, and Sergey Levine. Entity abstraction in visual model-based reinforcement learning. In *Conference on Robot Learning*, pages 1439–1456. PMLR, 2020.

[29] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546. IEEE, 2005.

[30] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[31] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010.

[32] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in neural information processing systems*, pages 2265–2273, 2013.

[33] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[34] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in neural information processing systems*, pages 766–774, 2014.

[35] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

[36] Nicholas Watters, Loic Matthey, Christopher P Burgess, and Alexander Lerchner. Spatial broadcast decoder: A simple architecture for learning disentangled representations in vaes. *arXiv preprint arXiv:1901.07017*, 2019.

[37] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[38] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1): 193–218, 1985.

[39] Jax Developers. Jax: Autograd and xla, 2020. URL `https://github.com/google/jax`.

[40] Flax Developers. Flax: A neural network library for jax designed for flexibility, 2020. URL `https://github.com/google/flax`.

[41] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[42] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[43] Rishabh Kabra, Chris Burgess, Loic Matthey, Raphael Lopez Kaufman, Klaus Greff, Malcolm Reynolds, and Alexander Lerchner. Multi-object datasets. https://github.com/deepmind/multi_object_datasets/, 2019.

A Implementation Details

We implemented all our experiments in Jax [39] and Flax [40].

A.1 Model

Unless specified differently, all f_{mlp} within our model refer to a two-layer MLP with hidden size of 128 and ReLU activation. Each f_{mlp} in our model has a separate set of trainable parameters.

Encoder We implement the encoder f_{enc} as follows:

$$\boldsymbol{h}_t^{(1)} = f_{\text{mlp}}(\boldsymbol{x}_t) + f_{\text{linear}}(\text{PositionEmbedding}) \tag{3}$$

$$\boldsymbol{h}_t = f_{\text{mlp}}(\text{LayerNorm}(\boldsymbol{h}_t^{(1)})) \tag{4}$$

f_{mlp} consists of two convolutional layers with kernelsize 1x1 with ReLU activation and filter-size $D_{\text{enc}} = 32$. f_{linear} is a single convolutional layers with kernelsize 1x1 and filter-size $D_{\text{enc}} = 32$. PositionEmbedding is a constant array that goes from from 0 to 1 in every image dimension.

Slot Attention For Slot Attention, we followed the implementation provided by the authors[2]. The feedforward network f_{ffw} consists of a GRU [41] and an MLP with skip connections. Following the original implementation of Slot Attention, we use LayerNorm on input features $\boldsymbol{h}_t$ and initial slot representations $\boldsymbol{c}$. Unless specified differently, we use four slots of dimensionality $D = 16$. As we are only considering simplistic environments where each object is assigned a unique, fixed color, we do not need to break symmetries between objects of identical appearance. Thus, we run Slot Attention with slot-specific learnable initializations and only a single iteration of the attention mechanism. We leave using a symmetric treatment of slots with multiple iterations of Slot Attention for future work.

FeatureMap MLP The FeatureMap MLP (FM-MLP) makes use of the same encoder but creates slot representations by applying a linear layer with each feature being interpreted as an individual slot. Each slot is then further processed through an f_{mlp}:

$$\boldsymbol{s}_t = \text{reshape}(f_{\text{linear}}(\boldsymbol{h}_t)) \tag{5}$$

$$\boldsymbol{s}_t = f_{\text{mlp}}(\boldsymbol{s}_t) \tag{6}$$

In our experiments, since we want to model 3 objects and the background, we use four slots and thus f_{linear} has a filter-size of four.

Decoder We use a filter-size of 16 within the decoder on the GridWorld dataset and 32 on the Bouncing Balls dataset. The decoder is trained using the same learning rate as is used for the rest of the model.

A.2 Training

We train all our models for 100,000 steps using Adam [42] and a weight decay of $1\text{e}{-6}$. We set the temperature τ in the score function of the SetCon loss to 0.5. We use a constant learning rate lr, scale it depending on the batch-size b: $lr \cdot \frac{b}{256}$, and tune it: We ran all models with 5 learning rates $lr = (1\text{e}{-4}, 2\text{e}{-4}, 3\text{e}{-4}, 4\text{e}{-4}, 5\text{e}{-4})$ and selected the best result based on their MSE. On the bouncing ball dataset, we select the best model based on its performance on the training set and report the results on a separate test-set.

We trained all our models on 8 TPU v2 cores.

In the GridWorld dataset, we use a batch-size of 1024 which is divided equally between the 8 TPU v2 cores that we train on. In the Bouncing Balls dataset, we set the global batch-size to 128.

[2]https://github.com/google-research/google-research/tree/master/slot_attention

A.3 Datasets

Multi-Object GridWorld The Multi-Object GridWorld dataset consists of sequences of eight frames, where each frame is made up of a 5x5 black plain. Colored pixels representing objects move deterministically through the scene: each time-step they move exactly one pixel-length to the top, bottom, right or left, and at the image boundary they bounce off in the opposite direction. The starting position, direction and color of each pixel is randomly sampled for each sequence. Objects start at different locations, but might overlap throughout the time-series in which case one of them becomes invisible. Unless otherwise specified, we restrict the number of pixels (i.e. objects) and the set of possible colors to three, and ensure that each color appears exactly once per sequence. We generate this set of colors by selecting equally spaced out hue values in HSV with a random offset and converting them to RGB.

Sequences are generated on the fly throughout training and evaluation. With 25 possible starting positions and 4 possible directions, this gives us a total of 970,200 possible sequences to generate. We evaluate all methods across the 5,000 last training batches, evaluating the performance on each batch before using its result for optimizing the model parameters.

Bouncing Balls The Bouncing Balls dataset consists of 2D sequences with three balls bouncing against one another and the image boundary. We generate 1,000 sequences and use 128 of these for evaluation. Each sequence consists of twelve frames. We generate three colors in the same way as for the Multi-Object GridWorld dataset, and use these across all sequences. Even though the training set is relatively small, we do not observe any overfitting in any of our models.

Note that due to the color selection in both datasets, it is possible to identify the objects by simply clustering the image by color.

We do not apply any random augmentations on both datasets, and solely rely on temporal differences for the contrastive loss. We scale all input values to the interval [-1, 1].

A.4 Evaluation

ARI Score In line with previous works [12, 21–23], we exclude background labels from the ARI evaluation. We use the implementation provided by Kabra et al. [43], available at https://github.com/deepmind/multi_object_datasets.

Future Prediction We train the decoder to decode all $\boldsymbol{p}_{t+k}$ and $\boldsymbol{s}_t$ to the respective inputs $\boldsymbol{x}_{t+k}$ and $\boldsymbol{x}_t$, and measure their reconstruction performance in MSE. We predict up to $k = 5$ steps into the future.

B Additional Experiments

B.1 Additional Results

From the qualitative results of the Slot Attention model when trained with the slotwise contrastive loss on the Bouncing Balls dataset in Fig. 3 it becomes apparent that this objective does not manage to push the individual slots to create diverse representations.

While we tested only one setup of a slotwise loss in our experiments, we expect the described fundamental problem to hold for other versions as well. In our experiments, we tested a slotwise loss as presented by Kipf et al. [8] in which slots are paired based on their index. Alternatively, one could design a slotwise loss in which slots with different indices are matched based on their respective loss. We do not expect such a setup to resolve the failure case of slotwise losses presented in this paper. It does not enforce any communication between the slots within the loss and thus it would still be a feasible solution for the model to learn to represent the same object across all slots.

Fig. 4 shows the normalized alpha masks of the decoder for the SetCon Slot Attention model.

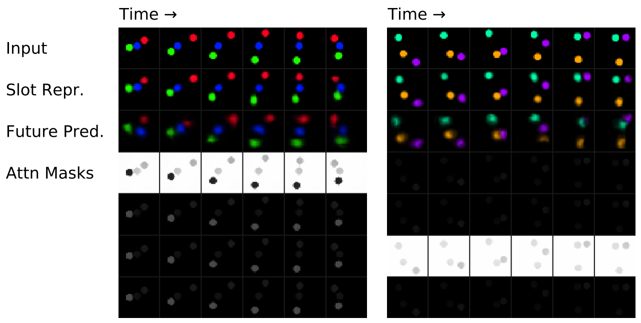


Figure 3: Qualitative Results for the Slot Attention model trained with the slotwise contrastive loss on the Bouncing Balls dataset.

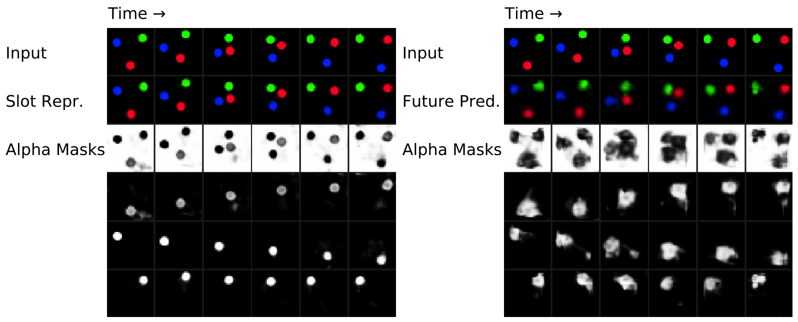


Figure 4: Decoder: Normalized alpha masks when trained on top of the SetCon Slot Attention model.

B.2 Future Prediction

Similarly to Section 4, we test whether our model is capable of predicting 5 time-steps into the future on the GridWorld dataset. As shown in Fig. 5, our approach leads to an overall worse future prediction performance compared to our main result (Table 1). Nonetheless, it achieves a relatively consistent performance across all predicted time-steps, and still performs favorably compared to the single-step prediction of the slotwise contrastive loss. The qualitative example shows that the model does manage to accurately predict the future trajectory of some objects (here: red pixel), but largely fails for others (here: green pixel), explaining the overall worse, but consistent performance.

B.3 Number of Colors

In Fig. 6, we evaluate the performance of the SetCon Slot Attention model when adding more colors to the GridWorld dataset. The plots show that the performance of the model degrades when more colors are added, irrespective of whether we use a fixed number of slots (four), or whether we use

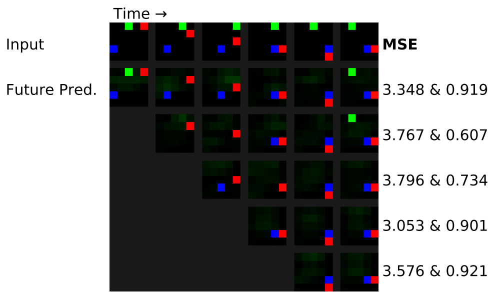


Figure 5: Predicting several time-steps into the future. From top to bottom: predicting further into the future.

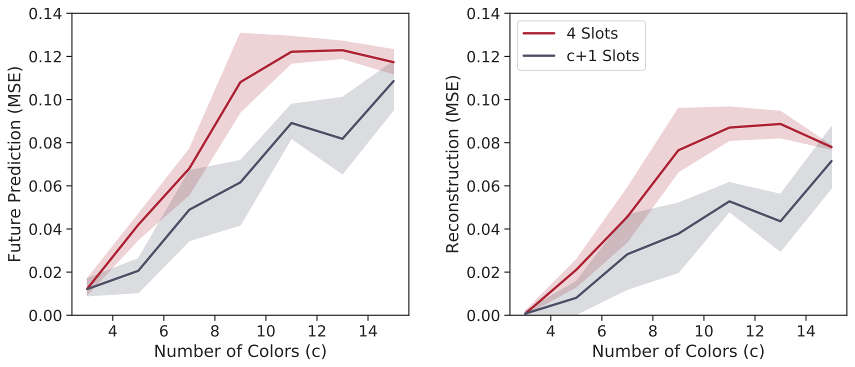


Figure 6: Reconstruction performance (MSE) depending on the number of colors and slots in the GridWorld dataset.

one slot for each color (plus one for the background). This indicates that the SetCon model relies heavily on the color information to separate different objects, and to create a better representation of the scene and its dynamics.

B.4 Random Initialization of Slots

Due to the learned initialization of the slots within the Slot Attention model, our model looses the ability to generalize to scenes with more objects. In this experiment, we want to investigate how our approach performs under randomly initialized slots. For this, we randomly sample from a unit Gaussian distribution $\boldsymbol{\varepsilon} \sim \mathcal{N}(0,1)$ and scale the result using learned parameters $\boldsymbol{\mu}, \boldsymbol{\sigma}$ that are shared across slots: $\boldsymbol{c} = \boldsymbol{\mu} + \boldsymbol{\varepsilon} \cdot \boldsymbol{\sigma}$. Then, we use three iterations within the Slot Attention module to compute our final slot representations $\boldsymbol{s}$.

We find that the performance deteriorates strongly with this approach: we achieve a MSE of 6.016 ± 2.418 on the future prediction $\boldsymbol{p}_t$ and of 3.678 ± 2.367 on the slot representations $\boldsymbol{s}_t$. This suggests that further exploration is necessary to make our approach work for scenes that require decomposition of objects with identical appearance (such as two red objects in one scene).