PARTS: Unsupervised segmentation with slots, attention and independence maximization - Supplementary Material

Daniel Zoran* Rishabh Kabra* Alexander Lerchner Danilo J. Rezende
DeepMind
London, UK
{danielzoran,rkabra,lerchner,danilor}@deepmind.com

1. Datasets

1.1. Playroom

PLAYROOM [1, 4] is a dataset of trajectories collected by running a pre-trained, embodied agent in a procedurally sampled 3D room generated in Unity [5]. The room is seen from the agent's ego-centric perspective. Transitions between frames are the result of the agent's movement; each action is 10-dimensional representing the agent's degrees of freedom. Objects can be of 34 different types, 10 different colors, and have continuously varying sizes. The environment was made available originally to facilitate certain language tasks at `https://github.com/deepmind/dm_fast_mapping`. We will release our version of the dataset publicly to facilitate further work on this domain.

1.2. CLEVRER

We use the dataset released by [8] at `http://clevrer.csail.mit.edu/`. We ignore the original question answering task for our work. We drop every second frame from the original sequences to make transitions more significant. For the results in Section 4.1, we train PARTS on the first 25 subsampled steps. We also resize the original 320x420 images to 64x64 using bilinear downsampling. We don't use any action information for our experiments on CLEVRER. To evaluate the segmentation performance, we use the ground-truth masks and accompanying scripts released by [9] at `https://github.com/BorealisAI/Spatio-Temporal`.

1.3. Robotics Arm

ROBOTICS ARM is a dataset of real-world videos of a Sawyer robotic arm and uniformly colored cubes captured by a fixed camera. The arm acts according to a policy trained to pick up and stack the objects. Each action is 7-dimensional. The dataset was introduced by [2].

2. Model

We now provide the full model parameters for all experiments. In the ablation experiments (Table 2 in the main paper) we used the same parameters for the baseline models as well (where applicable). This was done to control the performance difference due to architectural changes (attention, priors etc.) rather than leave it to the capacity of different components.

2.1. Hyperparameters

The latent size D_{slot} we use is 64. We train all models with 7 slots (though we instantiate some with 10 slots in Section 5 of the main paper). The output variance σ_x is set to 0.09 in all experiments. We clip the optimizer gradient norm to 5.0 and the internal gradient $\nabla\mathcal{L}$ to a norm of 10.0. All mask logits outputs from the decoder are passed through a $10\tanh(x)$ non linearity to improve stability. Models were trained with batch size 64 with a learning rate of $1e-4$ using the RMSProp optimizer for 1M iterations (ablations were run with batch size 32 for 400k iterations).

2.2. Architecture

Convolutional encoder $\mathcal{E}$. The convolutional encoder is a 5 layer convolutional network with 64, 128, 128, 256, 256 output channels respectively. Kernels are 5×5 in size, the stride is 1, and padding is set to "SAME" such that the size is preserved throughout. We used ReLU activation functions on all layers other than the last one (which is fed directly as input to the slot-attention encoder). The inputs it receives are the observation $\boldsymbol{x}_t$, the log likelihood image $logp(x|z)$ which is output from the decoder.

Slot Attention Encoder SA. We follow the original implementation of [6], including all layer norms, but without unrolling iterations and without using a GRU in the updates. The slot, value and key size are all set to $2D_{slot}$. The input to the slot attention which is used to form the queries is a concatenation along the channel dimension of the current predicted slot posterior parameters $\boldsymbol{\lambda}_t$, the gradient of the loss with respect to them $\nabla_{\boldsymbol{\lambda}_t}\mathcal{L}$ and the last taken action a_{t-1}. The inputs used to form the keys and values are the outputs of the convolutional encoder above. The outputs s_k^{output} from the attention step are used in the parameter update described next.

Posterior parameter update f. We take the output of the slot attention above s_k^{output}, and concatenate the current predicted posterior parameters $\boldsymbol{\lambda}_t$, the gradient $\nabla_{\boldsymbol{\lambda}_t}\mathcal{L}$, the previous action a_{t-1} and pass them through a two layer MLP (with 256 hidden units at each layer) to obtain the final parameter update $\Delta\boldsymbol{\lambda}$ which is added to the current estimate to form the observation-informed posterior for this time step $\hat{\boldsymbol{\lambda}_t}$.

Spatial broadcast decoder $\mathcal{D}$. We use a soft spatial broadcast decoder (following [6]). We use 6 convolutional layers with stride 1, kernel size 1, and 512 output channels each with ReLU activations, other than the last layer which outputs 4 channels and has no activation function. The first 3 are used as the component means and the last channel as the mask logits.

Prediction network $\mathcal{P}$. We follow the same architecture in [2]: a 2-layer transformer (with hidden size of 1024) followed by a slot-wise LSTM with hidden size of 256. The input to the prediction module is the observation-informed posterior from the previous time-step $\hat{\boldsymbol{\lambda}_{t-1}}$ to which we tile and concatenate the last action a_{t-1}. The output is the posterior used to decode and generate the loss.

2.3. Loss

At each time step we use the decoder output to calculate the loss $\mathcal{L}$ using the ELBO calculated with the decoder outputs. We use the loss to calculate the gradient with respect to the posterior parameters as the likelihood image (both used in refinining the posterior parameters).

2.4. Full algorithm

Here we describe the full model algorithm, which is similar to [3, 7], but using all the above modules.

```
Algorithm 1: PARTS inference algorithm - for time-step $t$
Input: observation $\boldsymbol{x}_t$ previous action $a_{t-1}$,
previous posterior parameters $\boldsymbol{\lambda}_{t-1}^N \in \mathbb{R}^{K\times 2D_{\text{slot}}}$, previous core state $\mathbf{h}_{t-1}$
Predict: $\boldsymbol{\lambda}_t^0, \mathbf{h}_t = \mathcal{P}([\hat{\boldsymbol{\lambda}}_{t-1}, a_{t-1}], \mathbf{h}_{t-1})$
for n in {1..N};                                         // N=1 in most experiments
Sample: $\boldsymbol{\mu}, \log\boldsymbol{\sigma} = \boldsymbol{\lambda}_t^{n-1}, \quad \boldsymbol{z}_t \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}) \in \mathbb{R}^{K\times D_{\text{slot}}}$
Decode $\boldsymbol{C}_k, \boldsymbol{m}_k = \mathcal{D}(\boldsymbol{z}_t)$
Loss $\mathcal{L}_t = -\sum_k \boldsymbol{m}_k\mathcal{N}(\boldsymbol{x}_t|C_k, \sigma_x) + D_{\text{KL}}(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}) \mid \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}))$
if training for segmentation or within prediction burn-in period then
    Bottom-up encoding: $\boldsymbol{O}_t = \mathcal{E}(\boldsymbol{x}_t, \log_p(\boldsymbol{x}_t|\boldsymbol{z}_t))$
    Attend: $s_k = [\boldsymbol{\lambda}_t^{n-1}, \nabla_{\boldsymbol{\lambda}_t^{n-1}}\mathcal{L}, a_{t-1}]. \quad s_k^{\text{output}} = SA(s_k, \boldsymbol{O}_t)$ ;        // slot attention
    Refine: $\boldsymbol{\lambda}_t^n = \boldsymbol{\lambda}_t^{n-1} + f([s_k^{\text{output}}, \boldsymbol{\lambda}_t^{n-1}, \nabla_{\boldsymbol{\lambda}_t^{n-1}}\mathcal{L}, a_{t-1}])$ ;        // obtain posterior
else
    $\boldsymbol{\lambda}_t^n = \boldsymbol{\lambda}_t^{n-1}$ ;                                        // no update
end
end for
```

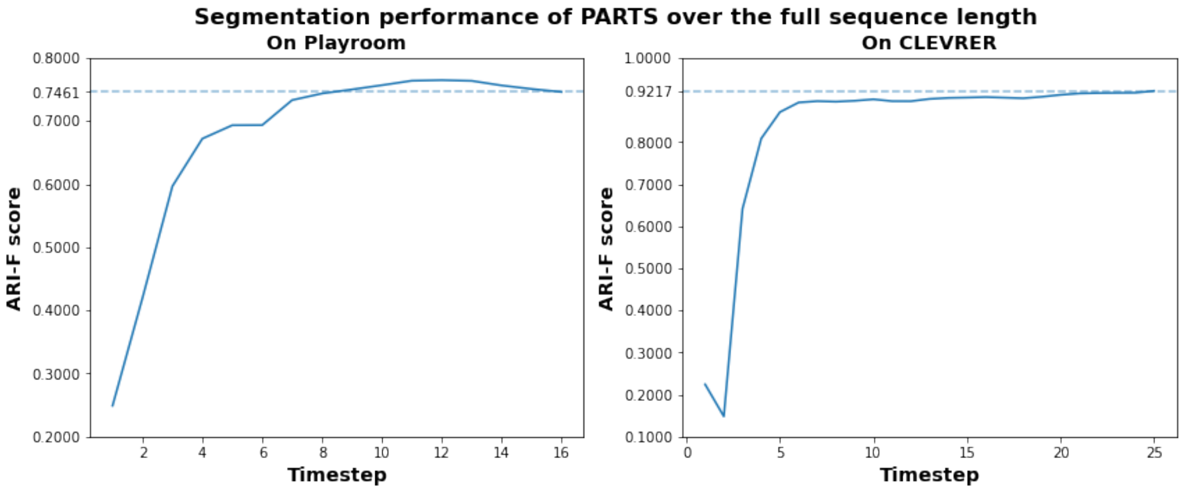

Figure 1. Improvement of segmentation performance over time.

3. More Results

3.1. Segmentation

To shed more light on the segmentation results (Section 4.1 in the main paper), we plot the segmentation score over time to illustrate the effect of refinement in Figure 1. We also provide more visualizations of the model's final reconstruction and segmentation outputs in Figures 2 and 3. Finally, we analyze the model in two settings different from how it was trained to showcase its generalization and consistency: in Figure 4, we show the model's improved performance on cluttered scenes when it is initialized with more slots. In Figure 5, we unroll the model on longer sequences than it was trained on to ensure slot consistency and stability. Note that this is distinct from the prediction/roll-out setting described in Section 4.2.

3.2. Prediction/Unroll

In Section 4.2 of the main paper, we described a training regime where the model observes a number of burn-in steps (where refinement is enabled), but is forced to make predictions for subsequent steps without any observation (i.e. no refinement). The ELBO is optimized as usual for all steps. To illustrate the benefit of this regime, we attach two videos where PARTS is unrolled even longer than in the main paper, for a total of 40 time-steps. The first video (**prediction_no_burn-in_baseline.gif**) is from a model which was trained for 16 steps with no burn-in (i.e. similar to the models in Section 4.1). The second video (**prediction_trained_with_burn-in.gif**) is from a model which was trained for 8 burn-in steps followed by 8 prediction steps.

The videos contain 4 columns each: the target image, decoded reconstruction, ground-truth segmentation, and decoded segmentation respectively. There is a red border in the videos around those frames when the model is receiving observations for refinement. Subsequent frames (without the red border) involve predictions from the transition module alone. The second model clearly performs better: it is able to predict the table in the room persists from a brief glimpse. It justifiably fails to predict the window on the wall because it never sees it during the burn-in period. The baseline model, on the other hand, fails completely after it stops receiving observations.

3.3. Latent Traversals

We attach videos containing a superset of latent traversals visualized in the main paper (Figure 9). The first video (**latent_traversals_playroom.gif**) is on PLAYROOM while the second (**latent_traversals_robotics_arm.gif**) is on ROBOTICS ARM. On PLAYROOM, we initialize the model with 10 slots instead of the usual 7.

The videos show reconstructed images at different traversal points over time, with perturbations spaced linearly from -2.0 to 2.0. Each frame shows different slots being traversed along the rows and latent dimensions along the columns. We can only visualize a subset of the 64 latents the model was trained on, but do visualize all the slots. Note that the last row (separated by a white line from the rows above) shows all slots being traversed at once (rather than slotwise per-latent traversals). This helps illustrate the cross-slot factors such as viewpoint described in Section 5.3.

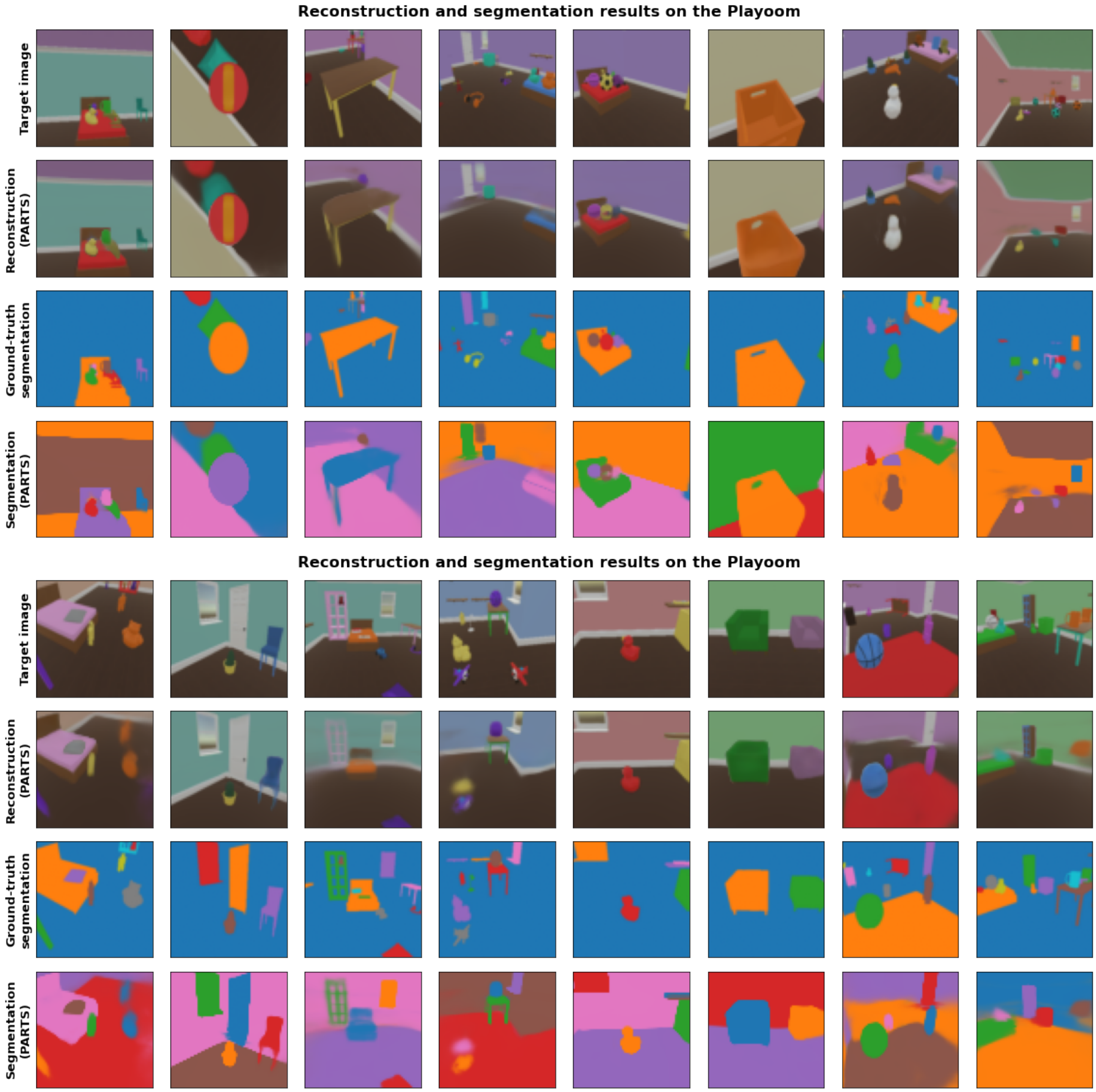
Reconstruction and segmentation results on the Playoom
Target image
Reconstruction (PARTS)
Ground-truth segmentation
Segmentation (PARTS)
Reconstruction and segmentation results on the Playoom
Target image
Reconstruction (PARTS)
Ground-truth segmentation
Segmentation (PARTS)

Figure 2. PARTS reconstruction and segmentation visualizations compared to target ground-truth on PLAYROOM data. Each column shows the final step of a length-16 sequence. We use the same model we used in Section 4.1. It was trained and evaluated with refinement enabled on every time-step.

References

[1] Josh Abramson, Arun Ahuja, Arthur Brussee, Federico Carnevale, Mary Cassin, Stephen Clark, Andrew Dudzik, Petko Georgiev, Aurelia Guy, Tim Harley, et al. Imitating interactive intelligence. *arXiv preprint arXiv:2012.05672*, 2020. 1

[2] Antonia Creswell, Rishabh Kabra, Chris Burgess, and Murray Shanahan. Unsupervised object-based transition models for 3d partially observable environments. *arXiv preprint arXiv:2103.04693*, 2021. 1, 2

[3] 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*

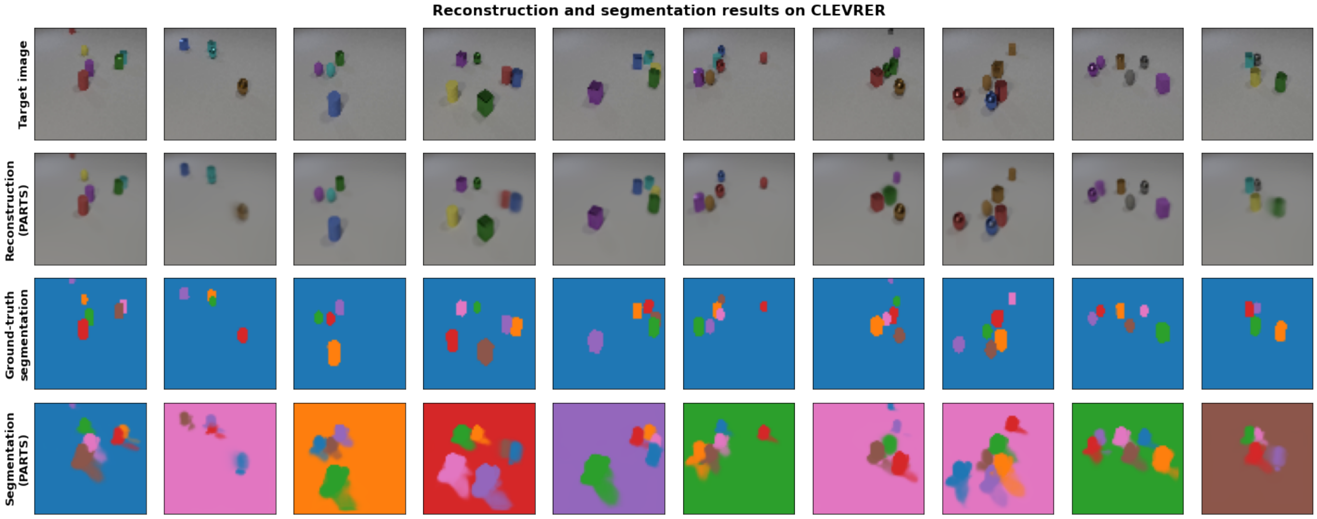
Reconstruction and segmentation results on CLEVRER
Target image
Reconstruction (PARTS)
Ground-truth segmentation
Segmentation (PARTS)

Figure 3. PARTS reconstruction and segmentation visualizations compared to target ground-truth on CLEVRER data. Each column shows the last step of a length-25 sequence. We use the same model we used in Section 4.1. It was trained and evaluated with refinement enabled on every time-step.

on Machine Learning, pages 2424–2433. PMLR, 2019. 2

[4] Felix Hill, Olivier Tieleman, Tamara von Glehn, Nathaniel Wong, Hamza Merzic, and Stephen Clark. Grounded language learning fast and slow. *arXiv preprint arXiv:2009.01719*, 2020. 1

[5] Arthur Juliani, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar, et al. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*, 2018. 1

[6] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 11525–11538. Curran Associates, Inc., 2020. 2

[7] 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. 2

[8] Kexin Yi*, Chuang Gan*, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B. Tenenbaum. Clevrer: Collision events for video representation and reasoning. In *International Conference on Learning Representations*, 2020. 1

[9] Polina Zablotskaia, Edoardo A Dominici, Leonid Sigal, and Andreas M Lehrmann. Unsupervised video decomposition using spatio-temporal iterative inference. *arXiv preprint arXiv:2006.14727*, 2020. 1

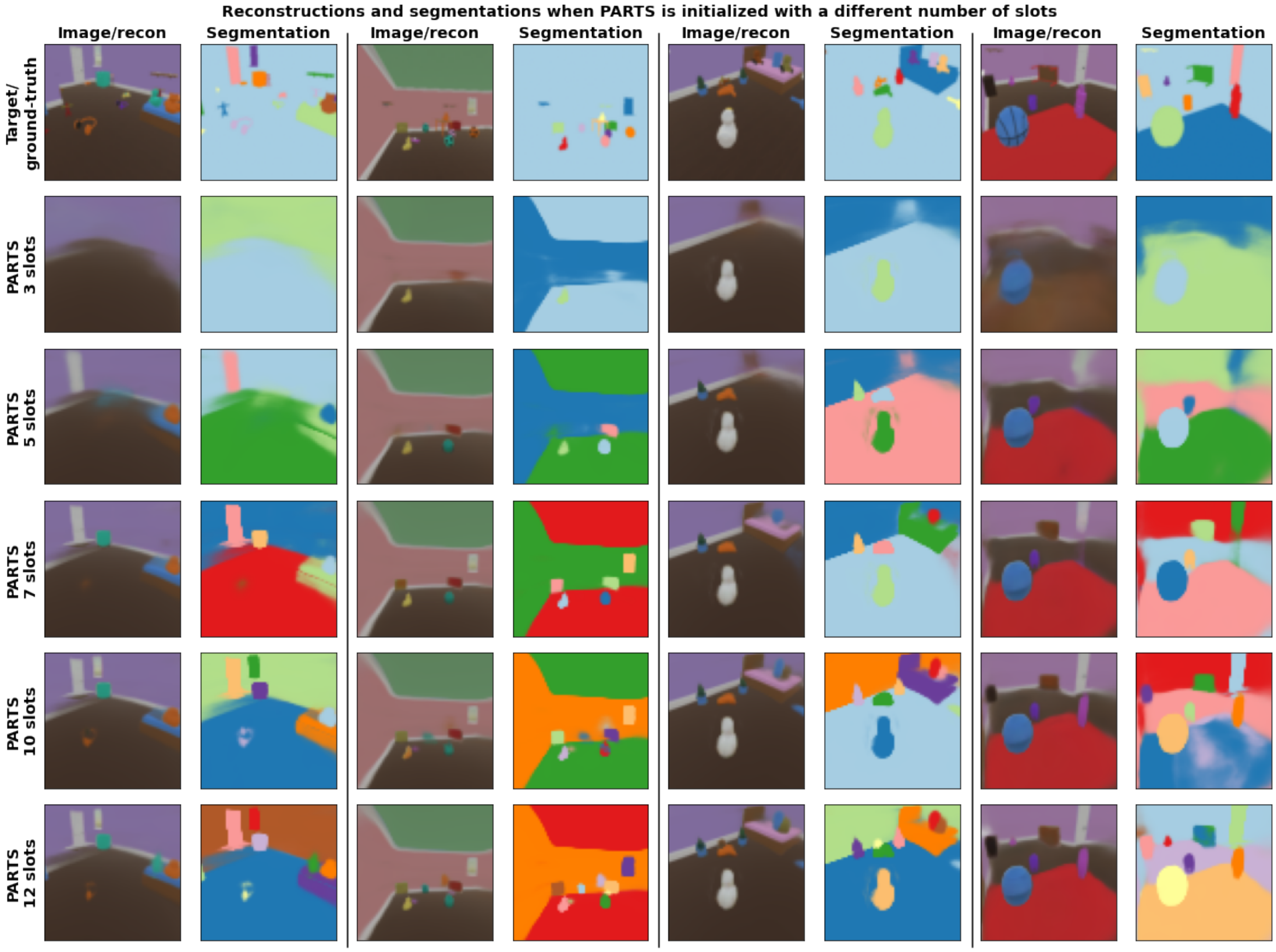

Figure 4. PARTS reconstruction and segmentation visualizations on PLAYROOM data when the model is initialized with a different number of slots than the number it was trained with. We use the same model here as in Section 4.1, which was trained with $K = 7$ slots. Adding more slots helps the model process cluttered scenes more accurately.

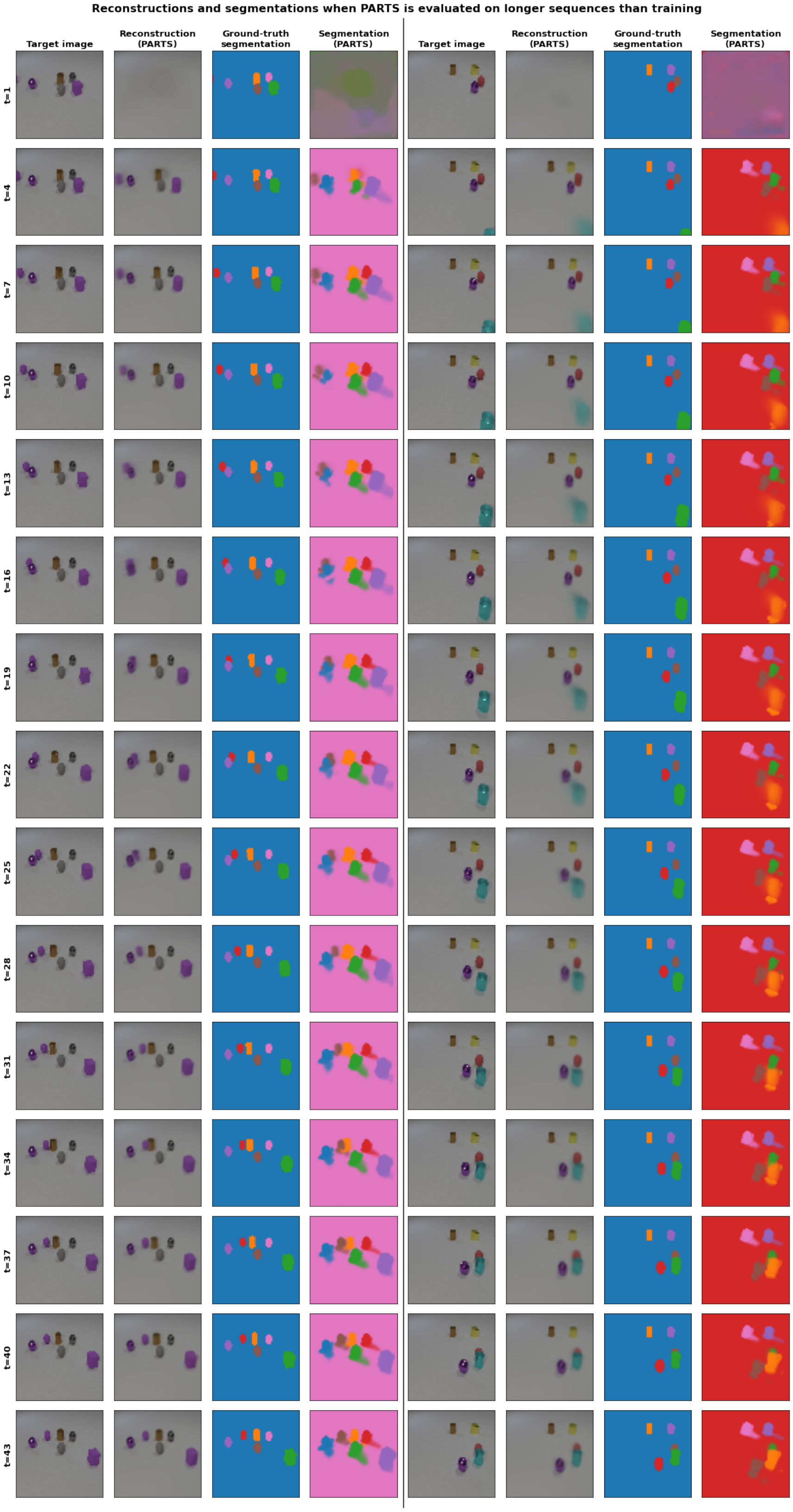

Figure 5. PARTS reconstruction and segmentation visualizations on CLEVRER data. We use the same model here as in Section 4.1, which was trained on length-25 sequences. Here we run the model on length-45 sequences, and visualize every third time-step. The model maintains the precision of its results and a consistent object-slot assignment (despite collisions) over the extended horizon. Even when a shape is ambiguous (e.g. when it is newly introduced in the scene), the model is able to refine its parameters over time.