PARTS: Unsupervised segmentation with slots, attention and independence maximization

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

Abstract

From an early age, humans perceive the visual world as composed of coherent objects with distinctive properties such as shape, size, and color. There is great interest in building models that are able to learn similar structure, ideally in an unsupervised manner. Learning such structure from complex 3D scenes that include clutter, occlusions, interactions, and camera motion is still an open challenge. We present a model that is able to segment visual scenes from complex 3D environments into distinct objects, learn disentangled representations of individual objects, and form consistent and coherent predictions of future frames, in a fully unsupervised manner. Our model (named PARTS) builds on recent approaches that utilize iterative amortized inference and transition dynamics for deep generative models. We achieve dramatic improvements in performance by introducing several novel contributions. We introduce a recurrent slot-attention like encoder which allows for top-down influence during inference. We argue that when inferring scene structure from image sequences it is better to use a fixed prior which is shared across the sequence rather than an auto-regressive prior as often used in prior work. We demonstrate our model's success on three different video datasets (the popular benchmark CLEVRER; a simulated 3D Playroom environment; and a real-world Robotics Arm dataset). Finally, we analyze the contributions of the various model components and the representations learned by the model.

1. Introduction

Object-level perception plays a key role in human vision [13]. People interpret scenes as consisting of discrete, independent and coherent objects and a broad set of experiments has shown this ability is developed very early in life [16]. While most present computer vision models lack any explicit object-level structure [7], there is an ongoing effort to create models that would, at least in some sense, represent scenes

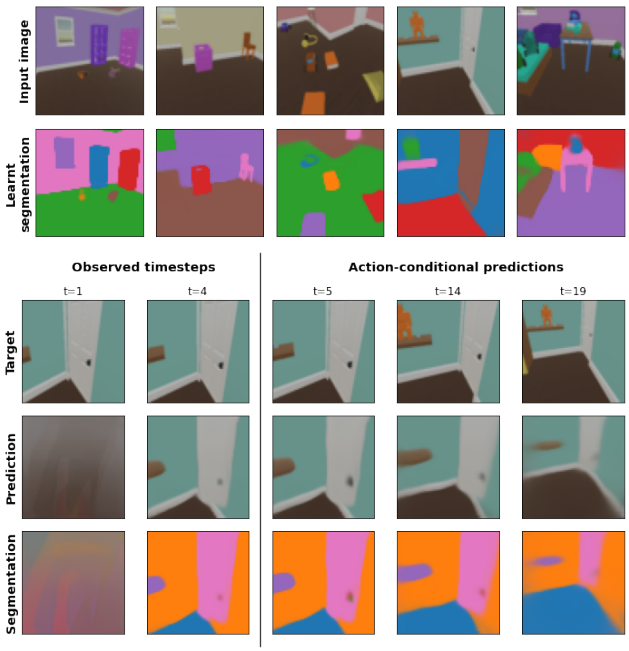

Figure 1. Unsupervised segmentation outputs from PARTS (above). The model is able to extract meaningful segments of complex, cluttered and partially observable scenes. It is also able to predict future frames along with their corresponding segmentation (below).

in terms of their constituent objects [1, 5, 2, 10].

There are good reasons why this is worth pursuing. Object-level representations are inherently combinatorial, allowing complex scenes to be explained by different combinations of simpler elements. Such representations may also allow better generalization by learning general rules related to what objects are, how they move and how they interact. Finally, objects may help in building models that allow planning, take into account actions and their effect on the environment and produce future predictions that are useful and actionable for agents.

Most current models that push in this direction introduce a structural inductive bias in the form of *slots*. These are

a set of latent variables (deterministic or stochastic) which are inferred from an observation and, if things work well, should each capture a different object or entity in the scene. One way to encourage this to happen is through architectural biases such as making the slots "compete" in explaining different parts of the scene (using a softmax, for example), processing each slot independently when decoding or the use of attention which may help in choosing which parts of the scene go to each slot. Probabilistic tools like different auxiliary losses [22] and choice of priors [3] are another way to encourage the emergence of objects in the representations.

When working with image sequences the problem, in some sense, becomes easier. When objects move in the world they usually do it in a coherent, consistent and predictable manner. This makes the problem of inferring which parts of the visual input belong to an object and which objects are in a scene easier. This can also help in avoiding useful, but wrong, heuristics like extracting objects purely by color. Ego-motion, especially if accompanied by action information can also reveal a lot about the structure of a 3D scene – for instance, motion parallax ensures that objects which lie farther from an observer will move differently than nearby objects when the motion is parallel to the camera plane.

While progress has been made in recent years along this front, extracting meaningful objects from complex 3D environments is still a challenge. We introduce **Predict, Attend, Refine with Transformers and Slots (PARTS)** – a model which is able to, unsupervised, extract object-level segmentations and representations of complex 3D environments and make predictions about future observations. We achieve this by extending and improving recently proposed models [5, 18]. We introduce a recurrent slot-attention mechanism which is able to incorporate top-down information and assist in the inference process. We show that the use of transformers and other architectural changes can help performance. Finally, we show how treating frames as *independent* in the generative process (but not in the inference process) can help resulting representations, especially when coupled with expressive pair-wise prediction models such as transformers. We analyse the contribution of each part of the model and study the resulting learned representations.

2. Related work

We first describe two families of models which are directly comparable to our work, and from which we draw our empirical baselines. Then we draw attention to the wider field of neural architectures which deploy slotted mechanisms or model temporal data via objects.

Iterative amortized inference through time. IODINE was the first sophisticated model to use a refinement network to decompose scenes into objects. Its sequential extension (hereafter, S-IODINE) keeps objects naturally aligned over time (provided they remain in view) as it refines their posterior estimates. By virtue of its symmetry-breaking sampling of initial object states, it also has the advantage of yielding multi-modal, multi-stable segmentations. Models that build on IODINE, like OP3 [18], Zablotskaia et al. [22], and PARTS, inherit the same advantages.

While IODINE can be applied to image sequences, it lacks an explicit dynamics model – a component that was introduced later in OP3. OP3 used a feed-forward graph neural network to explicitly model pairwise interactions between objects, whereas we take an implicit, recurrent approach. A more crucial difference between OP3 and PARTS is that OP3 updates its prior at every time-step to be the latest posterior. As we show later, while this makes sense if the model's goal is to generate and reconstruct pixels well, it hurts representation learning and segmentation performance. This is especially true when coupled with a rich and expressive dynamics model.

Recursive segmentation with added temporal dependence. Models like ViMON [20] and OAT [2] belong to a family of object-centric models based on MONet [1]. They employ an attention network to infer objects autoregressively from any given frame. However, they lack temporal coherence across time-steps. Instead, ViMON achieves object-slot consistency (i.e. alignment of slots over time) using slotwise recurrence in its inference procedure, and by mildly conditioning its attention network with the previous segmentation. ViMON lacks a dynamics model, which prohibits dealing with occlusions and interactions.

OAT is the state of the art among models based on MONet. It segments each observation independently over time. The inferred objects then need to be aligned across time-steps. To do so, OAT relies on a "memory" of previously seen objects, which is also useful in dealing with partial observability such as the disappearance and reappearance of objects over time.

We take inspiration from the architecture of the dynamics model and training regime used by OAT, which is capable of rolling out accurately over long sequences despite a weak temporal inductive bias. But while OAT predicts *latents* and uses an auxiliary latent loss, we predict *posterior parameters* and achieve better performance from a standard ELBO loss.

Other work. Among other unsupervised generative models for videos, SQAIR [11] is a highly expressive, structured-representation model which can cope with object dynamics, interaction, and the appearance of new objects. But due to the limitations of its additive image compositing, it mostly works on 2D binarized videos of sprites on a black background. To compare subparts of our model, we find that Recurrent Independent Mechanisms [4] was devised with similar desiderata as our own transition model–namely a combination of independent object dynamics (implemented by slot-wise recurrence) followed by sparse interaction between objects (implemented using key-value attention).

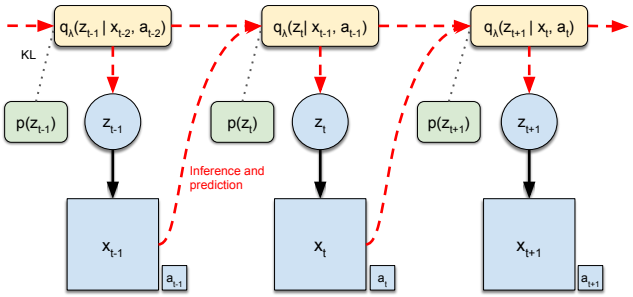

Figure 2. **Inference and generative model.** Our model attempts to infer a slotted latent variable for each time-step in a sequence of frames (and actions). We first try to predict and infer the current posterior parameters based on past observations. This posterior is used to sample the latent at the current time step, which then gets decoded to calculate the ELBO loss. With iterative amortized inference, we improve our estimate of the posterior using the loss and other decoder outputs. The prior we use is a spherical Gaussian, assuming frame, slot and feature independence. See text for details.

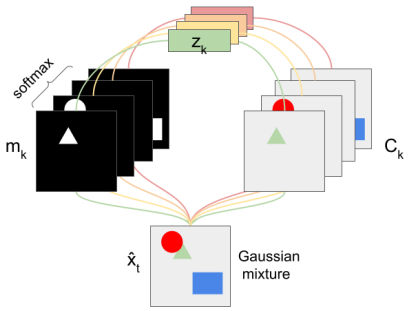

Figure 3. The **generative process**. Each slot $\boldsymbol{z}_k$ in a K-slotted latent variable $\boldsymbol{z}$ is decoded independently into a mean image $\boldsymbol{C}_k$ and a mask logits image $\hat{\mathbf{m}}_k$ using a spatial broadcast decoder. The mask logits images are softmax across the slots, akin to a soft occlusion process, to produce mask image $\boldsymbol{m}_k$ for each slot. These are then used as mixing weights for a per-pixel Gaussian mixture image likelihood model.

3. Model

The purpose of the model is to infer a useful, object-level representation of a 3D scene given a sequence of frames $\{\boldsymbol{x}_t\}_{t=1}^T$. For this, we use a latent-variable model where the frames are *independent* from each other. That is, its likelihood is of the form $p_\theta(\boldsymbol{x}_1, .., \boldsymbol{x}_T) = \prod_{t=1}^T p_\theta(\boldsymbol{x}_t)$. Counter-intuitively, all the dependency between frames is captured by the inference mechanism, not the generative model itself as illustrated in Figure 2. This is in contrast to recently proposed models [18, 22]. We will elaborate on this choice of model architecture in Section 5. Each frame $\boldsymbol{x}$ is modeled independently with a *slotted* latent variable $\boldsymbol{z} \in R^{K \times D_{\text{slot}}}$ and a Gaussian mixture pixel likelihood model [1, 5]. The inference works by a combination of prediction, top-down attention, bottom-up encoding and iterative amortized inference [14].

3.1. Generative process

The generative process starts with K independent D_{slot} dimensional latent variables $\boldsymbol{z}_k \in \mathbb{R}^{D_{\text{slot}}}$ (we refer to these simply as *slots*). Each of these slot variables would ideally correspond to a single entity or object in the scene when the model is trained. These slots are *independently* decoded using a decoder $\mathcal{D}_\theta$ (with trainable parameters θ) into a component mean image $\boldsymbol{C}_k \in \mathbb{R}^{H \times W \times 3}$ and a mask logits image $\hat{\mathbf{m}}_k \in \mathbb{R}^{H \times W \times 1}$. We then take the softmax of the mask logits (across slots) to produce the final K mask images $\boldsymbol{m}_k$. The softmax step can be thought of as a soft occlusion process and has been used in several recent works. Using these masks and mean images we can arrive at the model's mixture likelihood model of a frame $\boldsymbol{x}$ given the latents:

$$\boldsymbol{C}_k, \hat{\mathbf{m}}_k = \mathcal{D}_\theta(\boldsymbol{z}_k) \quad \boldsymbol{m}_k = \text{Softmax}(\hat{\mathbf{m}})_k \tag{1}$$

$$p_\theta(\boldsymbol{x}|\boldsymbol{z}) = \prod_i \sum_{k=1}^K \boldsymbol{m}_k \mathcal{N}(x_i|\boldsymbol{C}_k, \sigma_x) \tag{2}$$

where σ_x is a fixed hyperparameter and the pixels x_i are assumed to be independent given the latents. See Figure 3 for a summary.

Following [12] we use a soft version of the spatial broadcast decoder [19]. A coordinate grid the size of the output image is generated and then linearly projected to the same dimensionality as the slot latents to form a spatial basis. For each slot we then take the corresponding latent, tile it across space and add to it the spatial basis. The result is then decoded with a size-preserving convolutional neural network to a 4-channel image – the first three channels are used as the component mean image and the last one as the mask logits image. The prior for the latent in each frame is Gaussian, with zero mean and unit variance – assuming both slots and features are independent. The prior is also independent of the time-step and past latents: $p(\boldsymbol{z}) = \prod_{k=1}^K \mathcal{N}(\boldsymbol{z}_k|\mathbf{0}, \boldsymbol{I})$

3.2. Inference

At every frame $\boldsymbol{x}_t$ we would like to infer the corresponding slotted latent $\boldsymbol{z}_t$ using the posterior distribution $p(\boldsymbol{z}_t|\boldsymbol{x}_t)$. As is often the case, obtaining this distribution is intractable [15, 9] and we use an approximate posterior $q_{\boldsymbol{\lambda}_t}(\boldsymbol{z}_t|\boldsymbol{x}_t)$. We assume a diagonal Gaussian posterior parameterized by parameters $\boldsymbol{\lambda}_t := \boldsymbol{\mu}_t, \log \boldsymbol{\sigma}_t$.

Our inference works by employing an encoder $\mathcal{E}_\phi$ which combines bottom-up encoding and top-down attention together with a prediction model $\mathcal{P}$ which provides initial posterior parameters for the (yet to be observed) frame. Figure 4 depicts an overview of the encoder, prediction and inference mechanism.

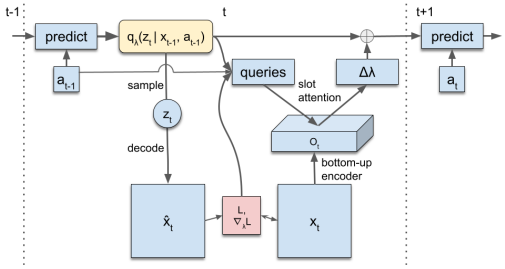

Figure 4. Our **inference mechanism** and **encoder architecture**. At each time step, the refined posterior parameters from the last time-step are passed into the prediction model, along with the last taken action. This produces the approximate posterior parameters for the current time-step. The latent $\boldsymbol{z}_t$ is sampled and decoded and the output is used to calculate the ELBO with the current observed $\boldsymbol{x}_t$. The frame $\boldsymbol{x}_t$ is passed through a conv-net to produce a spatial output $\emptyset_t$. This is attended to by a slot-attention mechanism which generates its queries from the current predicted posterior, and the gradient of the loss w.r.t to the posterior parameters. The output of the slot-attention is used to update the current posterior parameters which will be fed into the prediction model at the next time-step, repeating the process.

Much like [6, 5] we employ iterative amortized inference [14] to estimate the approximate posterior parameters. Iterative amortized inference works by progressively refining the posterior estimate. Specifically, our system *first* predicts an initial guess for the posterior parameters $\boldsymbol{\lambda}_t^0$ given the last obtained posterior $\boldsymbol{\lambda}_{t-1}$ and, if available, the last taken action a_{t-1} (see below for details of the prediction model). The predicted posterior parameters $\boldsymbol{\lambda}_t^0$ are used to sample the latent $\boldsymbol{z}_t \sim q_{\boldsymbol{\lambda}_t^0}$ which is decoded to calculate a loss $\mathcal{L}$ with the decoder output, likelihood model and current observation $\boldsymbol{x}_t$. Using this loss and observation, the posterior parameters at each refinement iteration $n \in \{1..N\}$ $\boldsymbol{\lambda}_t^n$ are updated with the output of the encoder network $\mathcal{E}_\phi$ (with trainable parameters ϕ) to produce the posterior for time step t:

$$\boldsymbol{\lambda}_t^n = \boldsymbol{\lambda}_t^{n-1} + \mathcal{E}_\phi(\boldsymbol{\lambda}_t^{n-1}, \boldsymbol{x}_t, a_{t-1}, \nabla_{\boldsymbol{\lambda}_t^{n-1}}\mathcal{L}, \log p(\boldsymbol{x}_t|\boldsymbol{z}_t)) \tag{3}$$

where the last term is calculated by taking the gradient of the loss $\mathcal{L}$ with respect to the current posterior parameters $\nabla_{\boldsymbol{\lambda}_t}\mathcal{L}$. Note that unless otherwise stated, as we use a *single* refinement step per frame ($N = 1$). The latter point has two subtle implications – the combination of the encoder and prediction model is in essence the refinement network as it appears in [5], and the posterior is always estimated using only *past* observations.

3.2.1 Prediction

The first step our inference model takes is predicting posterior parameters $\boldsymbol{\lambda}_t^0 \in \mathbb{R}^{K \times 2D_{\text{slot}}}$ for the current time step given the previous refined posterior estimates $\boldsymbol{\lambda}_{t-1}^N$ and the last taken action a_{t-1} (if available). This is done using a prediction network $\mathcal{P}_{phi}$. We concatenate the action to all slots in the input and use a transformer [17] followed by a slot-wise LSTM [2]. This expressive architecture allows modeling of complicated behaviour such as self and pairwise dynamics of objects, the effect of actions on slots and also allows the slots to share information regarding global information such as viewpoint (see Section 5).

3.2.2 Bottom-up encoding

Our bottom-up encoder is a simple convolutional neural network which receives as input the current frame x_t and, one of the decoder outputs, the per pixel log likelihood image $\log p(\boldsymbol{x}_t|\boldsymbol{z}_t)$. The log likelihood image and the input image are concatenated along the channel axis and then passed through a size preserving CNN (with strides 1) to produce the bottom-up encoding $\boldsymbol{O} \in \mathbb{R}^{H \times W \times \times C}$ where C is the number of channels in the last output layer of the network.

3.2.3 Recurrent, top-down slot attention

We take inspiration from the recently proposed "slot-attention" mechanism [12]. The original formulation is a core attention step where several heads "compete" to explain each data point (pixel) in the input. This core is unrolled in a closed loop for several iterations, forming an essentially bottom-up process.

We dissociate the core attention step from the iterations and incorporate it into the iterative amortized inference framework. This allows the inputs to our refinement network to be *much* simpler (compared to IODINE and OP3), and improves results substantially. It also allows for top-down influence on the attention, allowing queries based on current context. See Section 5 for an analysis on the value of this influence.

To elaborate on the querying mechanism, the current posterior parameters $\boldsymbol{\lambda}_t$ are concatenated with other top-down information – the last action a_{t-1} and the gradient of the loss with respect to the predicted posterior $\nabla_{\boldsymbol{\lambda}_t}\mathcal{L}$ – along the channel axis (tiling across slots if necessary). This slotted input $\boldsymbol{s} \in \mathbb{R}^{\boldsymbol{K}} \times \boldsymbol{C_q}$ is used to create a set of K queries $\boldsymbol{Q}$ by linearly mapping each slot such that:

$$\boldsymbol{q}_k = \boldsymbol{W}_q \boldsymbol{s}_k \tag{4}$$

The bottom-up input $\boldsymbol{O}$ is linearly mapped (per pixel ij) into two tensors $\boldsymbol{K}$ and $\boldsymbol{V}$ corresponding to keys and values: $\boldsymbol{k}_{ij} = \boldsymbol{W}_K \boldsymbol{O}_{ij}$ and $\boldsymbol{v}_{ij} = \boldsymbol{W}_V \boldsymbol{O}_{ij}$. Note that while the keys and values tensors are purely a function of the bottom-up input, the queries are shaped by top-down influence which depends on the current state of the model and loss to be optimized. At every pixel location we calculate the inner product between the key at that location and each query, producing K logits which are then softmax-ed across the

slot dimension, resulting in K attention weights per location:

$$\text{attn}_{ijk} = \text{Softmax}(\frac{1}{\sqrt{D_q}}\boldsymbol{k}_{ij}^T\boldsymbol{Q})_k \tag{5}$$

Taking the softmax across the slot dimension means that the attention weights sum to one at every pixel location. These attention weights are then used to produce a slotted output by multiplying the resulting attention weights for each slot with the value tensor $\boldsymbol{V}$ and summing spatially to produce the output:

$$\boldsymbol{s}_k^{\text{output}} = \frac{\sum_{ij} \text{attn}_{ijk}\boldsymbol{v}_{ij}}{\sum_{ij} \text{attn}_{ijk}} \tag{6}$$

As a final step we pass these slot outputs (concatenating other flat inputs such as the loss gradient and action) through an MLP (applied independently to each slot) obtaining the refinement $\Delta\boldsymbol{\lambda}_t$ which is added to the current posterior parameter estimate $\boldsymbol{\lambda}_t^(n-1)$ resulting in the current posterior $\boldsymbol{\lambda}_t^n$. With a single refinement step these posterior parameters are passed directly as input to the prediction model and do not get decoded.

3.3. Loss

We use the standard Evidence Lower Bound (ELBO) as the loss at each step:

$$\mathcal{L}_t(\boldsymbol{x}_t, a_{t-1}) = \\ \mathbb{E}_{\boldsymbol{z}\sim q}\left[\log p_\theta\left(\boldsymbol{x}_t \mid \boldsymbol{z}\right)\right] - \beta D_{\text{KL}}(q_{\boldsymbol{\lambda}_t}(\boldsymbol{z}|\boldsymbol{x}_t, a_{t-1}) \mid p(\boldsymbol{z})) \tag{7}$$

where β is set to 0.5 in all experiments. We also take the gradient of this loss with respect to posterior parameters $\boldsymbol{\lambda}_t$ as part of the inference inputs as noted above. For the optimizer loss we sum the individual time-step losses across time with uniform weights $\frac{1}{T}$ such that the total loss optimized is: $\mathcal{L}_{\text{total}} = \frac{1}{T}\sum_{t=0}^{T}\mathcal{L}_t$

4. Experiments and Results

In the following subsections, we first showcase the model in its default regime to illustrate its best-case inference and segmentation performance (Section 4.1). Next, we train and evaluate the model to optimize its predictive performance when rolled out without observations (Section 4.2). Finally, we ablate the model to show the importance of each component, also deriving the OP3 and S-IODINE baseline models as special cases (Section 4.3).

Our experiments are based on three varied video datasets: The PLAYROOM [8] (aka 3D Room in [2]) is a 3D environment built in Unity where an agent can move and interact with toys, furniture, and everyday objects. Second, CLEVRER [21] is a dataset designed to test causal understanding based on moving, colliding, and newly introduced objects. Finally, ROBOTICS ARM [2] is a dataset of real-world trajectories of a ROBOTICS ARM manipulating colored cubes in an attempt to stack them. Each of these datasets has been used in prior work, thereby facilitating comparison. Further details on the datasets are in Appendix.

	S-IODINE	OP3	OAT	PARTS
Segmentation (ARI-F ↑)	0.4461	0.4104	0.6053	**0.7349**
Reconstruction (MSE ↓)	0.0044	0.0041	0.0025	**0.0023**

Table 1. Summary of main results on the PLAYROOM dataset. All metrics are computed on the last time-step of a minibatch of 64 sequences.

We train our models with an RMSProp optimizer. All models are trained with $K = 7$ slots, but since the number of parameters is independent of number of slots, we can always re-instantiate the model with a different number (we do so for latent traversals below). The initial state for the posterior parameters and all recurrent cores is all zeros. We employ gradient clipping to a global norm of 5.0. When we unroll for relatively long sequences, we stop all state gradients every 4 steps, effectively training with Truncated Backpropagation Through Time. This is to prevent overfitting to a specific length so that the model can be unrolled for various lengths for inference or prediction. Full training details with hyper-parameters can be found in the Appendix.

4.1. Segmentation

We trained PARTS (unrolling the full predict → decode → refine loop) on length-25 sequences for CLEVRER and length-8 sequences for ROBOTICS ARM. On CLEVRER, we did not condition the dynamics model on any actions. For ROBOTICS ARM, we used the 7-dimensional movement of the robotic arm as input actions to the dynamics model. Figure 5 shows a qualitative sample of our segmentation results. More detailed segmentation figures, including ground-truth object masks and results on the PLAYROOM, are available in the Appendix.

We can also measure our segmentation performance on datasets where we have ground-truth segmentation masks. This precludes the ROBOTICS ARM dataset which contains unlabelled real-world frames. On the remaining datasets, we compute the Adjusted Rand Index on all foreground components (ARI-F) on the last frame of each sequence in a minibatch. On CLEVRER, PARTS achieves an ARI-F score of 0.921, which is on par with [22] with a similar number of time-steps. On the PLAYROOM, PARTS scores 0.7349, which is significantly better than [2]. Note that the ablation study in Table 2 reports a lower ARI-F score for PARTS on the PLAYROOM; this is because the ablation models were trained for fewer iterations and with lower batch sizes to compare their relative performance without the computation cost of training them to their maximum performance.

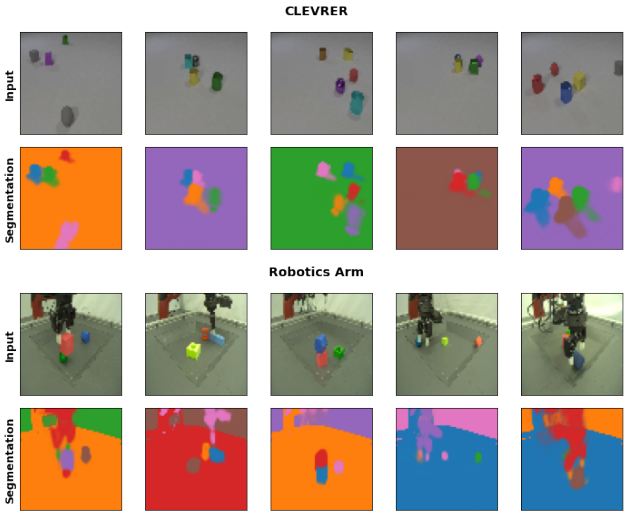

Figure 5. **Segmentations** learnt on the CLEVRER (above) and ROBOTICS ARM (below) datasets. We show the last time-step of a length-25 sequence for CLEVRER and length-8 sequence for ROBOTICS ARM. On **CLEVRER**, PARTS learns to associate each object's shadow with its segment despite intricate lighting. On **ROBOTICS ARM**, the model cleanly segments the robotic arm, separating it from the base of the arm whose position remains constant.

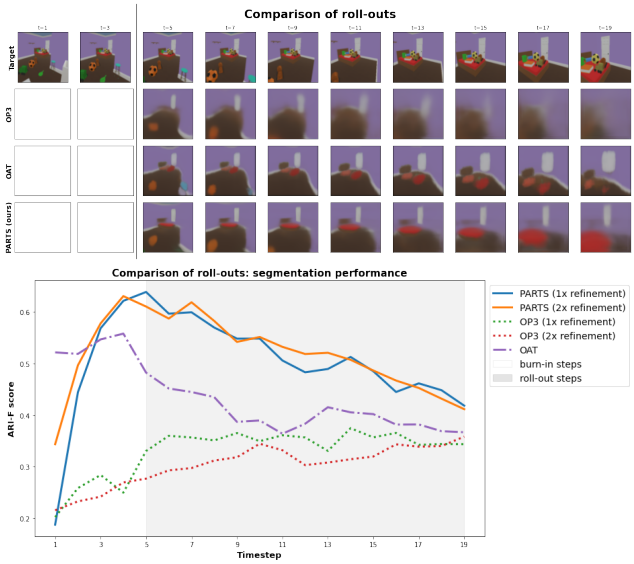

Figure 6. **Comparison of roll-outs** from PARTS and baseline models on a PLAYROOM sequence (above, qualitative) and a consistent batch of sequences (below, quantitative). Each model observes the first four time-steps, and then predicts subsequent time-steps using only the agent's actions in the room.

4.2. Prediction/Unroll

In Figure 6 we show the difference between transition dynamics learnt by PARTS versus comparable models on PLAYROOM. For these results, each model was trained to observe an initial number of *burn-in* frames, and then predict (the slot-wise posterior parameters at) future time-steps using a learned action-conditional dynamics model. This regime was introduced in [2]; we found it beneficial in boosting the performance of the dynamics model. The initial observations help estimate the global properties and contents of the scene. The subsequent time-steps, where only actions are available as inputs, force the dynamics model to output meaningful predictions; it cannot rely on any refinement to correct prediction errors, nor can it serve pass-through predictions if it is to minimize the loss.

During evaluation, we rolled the models out for longer than they were trained to do (15 roll-out steps at evaluation versus 12 in training). Note that OAT alone was trained to unroll for 6 steps rather than 12 and without TBPTT. All models were trained with 4 burn-in steps, observing the same amount of initial information about the scene. From the evaluation results, it is clear that PARTS is capable of significantly more accurate predictions over the same horizon.

4.3. Ablations

We ablate our model, spanning the spectrum from S-IODINE through OP3 to PARTS. This helps show the effect of using top-down slot-attention, a fixed prior, and the transformer plus slotwise LSTM transition model. Adding in all three results in the full PARTS model. We control for network architecture, number of layers and number of units in each layer as much as possible, so the baseline models presented here are larger, deeper and wider than the original models in [5, 18]. All models here share the same decoder and convolutional encoder architecture (up to strides for slot-attention models where it is set to 1 in all layers).

All models were trained on 16-step sequences on PLAYROOM data, including action information for PARTS and OP3. We summarize the ablation results in Table 2. The standard deviations are computed across 4 independent seeds for each row in the table. Given the large number of models, we trained them slightly less than usual (up to 400,000 steps) to save on compute. As a result, the ARI-F scores in Table 2 are lower than mentioned elsewhere (especially Table 1).

5. Analysis

5.1. The effect of a fixed prior

We argue that when interested in representation learning rather the pixel reconstruction, it is more sensible to use a fixed independent prior for all frames in a sequence. The reasoning behind this is twofold – if the prior indeed represents something about the statistical properties we want our latents to have, then it shouldn't matter at what position in the sequence a frame is or what was the posterior of the previous frame. In other words, this assumes a fixed scene throughout the sequence where most of the variance in the

Name	SA	Fix. prior	Actions	TRx	ARI-F (↑) ± std
S-IODINE	✗	✓	✗	✗	0.417 ± .014
OP3	✗	✗	✓	✗	0.425 ± .004
	✓	✓	✗	✗	0.476 ± .009
	✓	✗	✓	✗	0.651 ± .010
	✓	✗	✓	✓	0.554 ± .013
	✓	✓	✓	✗	0.672 ± .014
PARTS	✓	✓	✓	✓	**0.712 ± .009**

Table 2. Ablation analysis on 16-step sequences from the PLAYROOM. We report the ARI-F score measured on the last time-step. We compare PARTS against extensions of IODINE and OP3, augmenting each with the following components to illustrate their contribution: slot attention (SA), a fixed independent prior (Fix. Prior) for each frame in the sequence, an action-conditional transition model (Actions), and our transformer plus slot-wise LSTM transition model (TRx). Adding in all these components yields our model, PARTS, which outperforms all the other variants here. Note that these numbers do not represent the models at their maximum performance (as we trained them for fewer steps for this ablation study), but are meant for relative comparison.

images is due to viewpoint changes. If objects do move in the scene the inference may be able to correct the current scene estimate but the generative model does not handle this case directly.

In the context of iterative inference, the posterior from the last time-step can indeed be a good guess for the posterior at the current time-step, but this, we argue, shouldn't matter to the prior or optimization objective. The issues that may arise from using an auto-regressive prior become more severe in the context of object-centric learning if one lets latents in different slots communicate and share information (as is the case in PARTS, and also OP3). While desirable in many respects (*i.e.* allowing modeling pair-wise object dynamics), such interactions may cause the latent variable to effectively lose its slotted structure and slot independence as time passes in the sequence and the original independent prior becomes negligible in its effect. This is true also if one is interested in disentangled representations, as the features themselves can mix and cease to be independent.

To show this actually happens, we take a PARTS model trained with a fixed prior, as well as a model trained with predictive prior (similar to OP3) and show their resulting segmentation, reconstruction and time-step KL and MSE metrics in Figure 7. The models are both unrolled on a fixed image with action input of 0 (meaning no movement), and unrolled for 32 time steps. As can be seen, the fixed prior model achieves stable segmentation and much higher KL values well into the sequence. The predictive prior model indeed gets better reconstruction error, has much lower KL values (because the KL is much easier to minimize in this case) but the resulting segmentation are far worse.

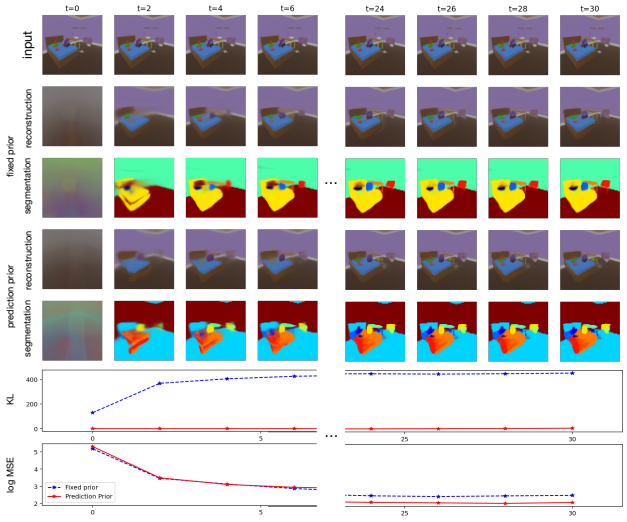

Figure 7. **Fixed vs. Predictive Prior.** We train two models: one is the full PARTS model which uses a fixed, independent prior. The second is identical in every way other than using the output of the predictor as the prior at every step (as in OP3). The two models were tested on a static scene, unrolled with 0 actions for 32 steps. We see the resulting segmentation in the fixed prior model (third row) is much better than the predictive prior version (fifth row). As is to be expected, the predictive prior model achieves lower reconstruction eventually and a much lower KL.

5.2. The effect of top-down information on attention

We test the hypothesis that top-down information is used to shape the attention queries. To do this we train two models, identical in every aspect other than the information used to generate the queries in Equation 4. In one model (NO-TOP-DOWN) we set $\boldsymbol{s}_k$ to be the corresponding slot in $\boldsymbol{\lambda}_t$. No other information is passed in (as would be the case in the original slot-attention model [12]). In the other model (which is the full PARTS model) we concatenate the previous action a_{t-1} and the gradient of the loss w.r.t the posterior parameters as described above. We postulate that this information allows the queries to be shaped more accurately according to the system's objective. Note that the action and gradient information *are* available to both models in the bottom-up direction, post-attention. We measure the ARI scores of both models on PLAYROOM and find a significant difference – 0.72 for NO-TOP-DOWN vs 0.78 for the full PARTS model. The attention patterns are also quite different, as can be seen in Figure 8. We visualize the attention weights by normalizing to visible range and super-imposing them on the observation image $\boldsymbol{x}_t$. Without top-down information the the attention is much less specific, being more spread out across the image. With top-down influence the attention is able to focus much more and results in better segmentation. See supplementary material for videos of this visualization.

5.3. Latent traversals

To better understand the learned representations of our model we perform traversals in latent space. We run trained models on 12-step and 8-step sequences from PLAYROOM and ROBOTICS ARM respectively. We use the inferred latents of the last frame as the origin of the traversals. We then apply fixed perturbations within the range -2.0 to 2.0 to each feature dimension in each slot (separately, so we can observe the effect on each slot independently). On PLAYROOM, we additionally apply the same perturbation to all slots simultaneously (but still to a single feature dimension) to visualize the joint effect.

A subset of results are in Figure 9. On PLAYROOM, we observe some feature dimensions capture basic individual properties of the objects such as color. Other latents seem to be shared across the different slots and tie them in non-trivial ways – the ones portrayed here seem to correspond to camera viewpoint. Such cross-slot factors are in contrast to the independence assumption imposed by the prior. We postulate the model is able to learn them through the interactions in the transformer. Learning such a global latent explicitly is an interesting future direction which may result in more interpretable models. On ROBOTICS ARM, the model learns latents representing object position across the ground plane (third row in Figure 9). We also find latents that change the arm's position and grasp (bottom row). More traversals, including videos, are in the supplementary material.

6. Conclusion

We have introduced PARTS–a model that produces state-of-the-art segmentations and representations of objects in complex 3D scenes without supervision. It employs a recurrent, transformer-based dynamics model to predict latents from prior information (including agent actions), followed by a slot attention-based refinement step, which is informed by the reconstruction of the frame from the current posterior estimate, among other inputs. We have shown how the model can be trained to optimize for segmentation or prediction.

For future work, it would be worthwhile exploring whether our independent-frames assumption in the generative process can be relaxed to generate coherent, novel videos without sacrificing representation quality or segmentation performance. The datasets used in this work, while non-trivial, are not of real-world scale — it would be interesting to scale the model up to work on natural data. Finally, we look forward to evaluate downstream applications of the structured representations which PARTS can offer.

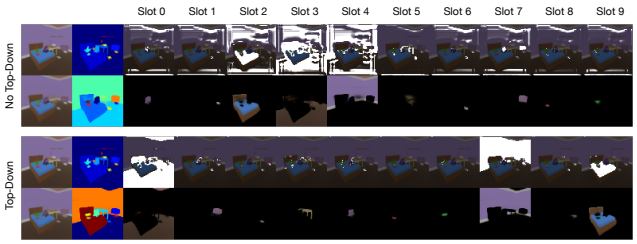

Figure 8. **Top-down information and attention.** We study the effect of top-down influence on resulting attention patterns and performance. We train two PARTS models which differ only in the inputs that are used to generate queries in the slot-attention mechanism. TOP-DOWN receives the posterior parameters along with the last taken action and gradient of the loss w.r.t the posterior. NO-TOP-DOWN receives only the posterior parameters. We superimpose the attention for each slot on the input image (top row in each panel). As can be seen, top-down information focuses the attention on content which directly relates to objects in slots (masked reconstructions for each slot are shown in the bottom row of each panel). See text for more details.

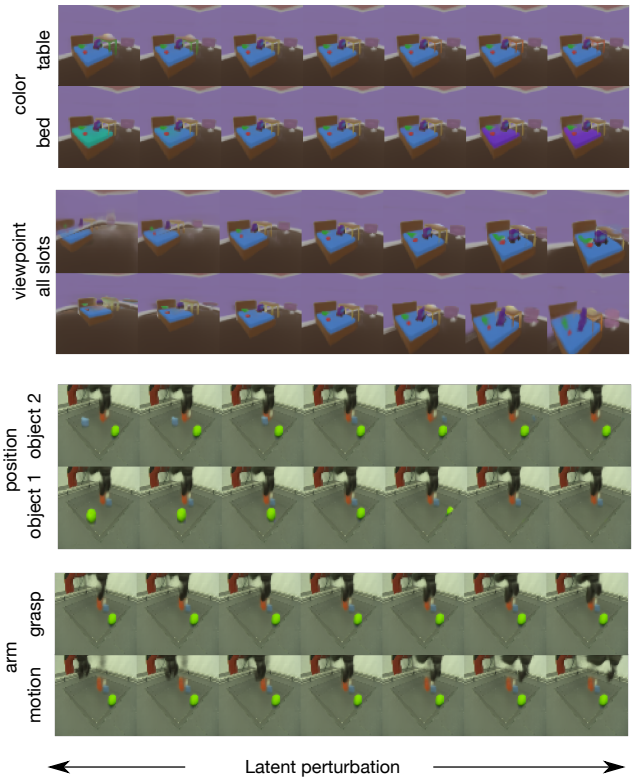

Figure 9. **Latent traversals**. We investigate our learned representations by performing latent traversals. We infer last-step latents on PLAYROOM and ROBOTICS ARM sequences. Then, we perturb a single latent in a single slot (top, third and last traversals) or a single latent in all slots simultaneously (second traversal). On PLAYROOM the model learns latents that change, for example, the color of an object (top). It also learns latents that relate to shared, global information such as viewpoint or camera motion (second) when traversed across all slots at the same time. On ROBOTICS ARM, the model is able to learn a disentangled representation for object position (third) and for arm motion and grasping (bottom). See supplementary material for videos and more examples.

References

[1] 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. 1, 2, 3

[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, 4, 5, 6

[3] 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. 2

[4] Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. Recurrent independent mechanisms. *arXiv preprint arXiv:1909.10893*, 2019. 2

[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 *International Conference on Machine Learning*, pages 2424–2433. PMLR, 2019. 1, 2, 3, 4, 6

[6] Klaus Greff, Antti Rasmus, Mathias Berglund, Tele Hao, Harri Valpola, and Jürgen Schmidhuber. Tagger: Deep unsupervised perceptual grouping. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. 4

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[8] 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. 5

[9] Diederik P Kingma and Max Welling. An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691*, 2019. 3

[10] Thomas Kipf, Elise van der Pol, and Max Welling. Contrastive learning of structured world models. *arXiv preprint arXiv:1911.12247*, 2019. 1

[11] Adam Kosiorek, Hyunjik Kim, Yee Whye Teh, and Ingmar Posner. Sequential attend, infer, repeat: Generative modelling of moving objects. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. 2

[12] 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. 3, 4, 7

[13] Nikos K Logothetis and David L Sheinberg. Visual object recognition. *Annual review of neuroscience*, 19(1):577–621, 1996. 1

[14] Joe Marino, Yisong Yue, and Stephan Mandt. Iterative amortized inference. In *International Conference on Machine Learning*, pages 3403–3412. PMLR, 2018. 3, 4

[15] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014. 3

[16] Elizabeth S. Spelke. Principles of object perception. *Cognitive Science*, 14(1):29–56, 1990. 1

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 4

[18] 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, 3, 6

[19] 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. 3

[20] M. A. Weis, K. Chitta, Y. Sharma, W. Brendel, M. Bethge, A. Geiger, and A. S. Ecker. Unmasking the inductive biases of unsupervised object representations for video sequences. *arXiv*, Jun 2020. 2

[21] 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. 5

[22] 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. 2, 3, 5