COMPAS: Compose Actions and Slots in Object-Centric World Models

Abstract

In this paper, we propose a novel approach **COMPAS** (**COMP**ose **A**ctions and **S**lots), which leverages the strengths of state-of-the-art object-centric approach for modeling the dynamics of an environment. Our method encodes the environment's state into symbol-like, object-centric representations, known as slots, where each slot corresponds to an individual object. This approach offers a structured and interpretable way to model complex environments by combining slots with action representations for accurate next-state prediction. The primary contribution of our work is an efficient world model with a dynamics predictor capable of predicting accurate trajectories in action-dependent environments. Additionally, our slot extractor module enhances predictive capabilities by extracting deterministic slots that remain consistent both within a single trajectory and across episodes. This consistency significantly improves the quality of dynamic predictions. We present a comprehensive evaluation of our approach in various environments, demonstrating that our proposed method outperforms competing models in environments with discrete and continuous action spaces.

1 INTRODUCTION

The ability to generalize compositionally is the key to learning faster, solving new problems, and understanding new concepts with limited experience [Lin et al., 2023]. The difficulty in compositional generalization is caused by the so-called binding problem [Greff et al., 2020] – the inability of modern artificial neural networks to dynamically and flexibly bind information distributed over the network, which arises in learning on unstructured input data.

A possible solution to this problem could be the use of symbol-like representations. Such representations can be slot representations [Locatello et al., 2020], where instead of a single latent embedding, the input data is encoded by a set of such embeddings (slots). Slots compete with each other to describe a portion of the input data. Such representations have been successfully used for object-centric tasks such as set property prediction and object detection [Locatello et al., 2020] in an image, learning visual dynamics from video [Wu et al., 2022], image generation [Singh et al., 2022a], and unsupervised object-centric representation learning for real-world data [Seitzer et al., 2023].

World models serve as core components in various embodied machine learning tasks, ranging from robotics and autonomous vehicles to game artificial intelligence and video synthesis [Wu et al., 2023a, Burgard et al., 2016, Ha and Schmidhuber, 2018]. These models aim to represent the environment's complexities and dynamics, enabling agents to understand, predict, and interact with their surroundings effectively. However, creating world models that can handle high-dimensional, continuous, and time-varying data remains challenging. Traditional world models tend to represent the environment as a whole without distinguishing between individual objects [Hafner et al., 2020, Micheli et al., 2023]. Models with holistic state representation struggle to capture the intricate relationships between objects Santoro et al. [2017] and their evolving dynamics over time Oprea et al. [2022]. They often fail to generalize well across different tasks and face scalability challenges when dealing with complex scenes containing multiple interacting objects. In contrast, object-centric world models offer several advantages. First, they provide more interpretable representations, as the state of the world is described in terms of identifiable objects and their properties. Second, they are better suited to handle complex scenes with multiple objects, as changes in one object's state do not necessarily affect the representation of others. Finally, object-centric models are more data-efficient because they use informative, factored representations, rather than relying on a single, monolithic vector that encapsulates all scene information.

Submitted to the 41st Conference on Uncertainty in Artificial Intelligence (UAI 2025). **To be used for reviewing only**.

In this paper, we propose a novel approach **COMPAS** (**COMP**ose **A**ctions and **S**lots) to encode these dynamics, focusing on object-centric world models that integrate actions and slots. The main contributions of our work are:

- A slot extractor for video that ensures consistent slots across both trajectories and episodes.
- A deterministic slot initialization procedure that guarantees slot consistency across episodes. Consistent slot acquisition is demonstrated in Figure 4.
- A temporal consistency block that enforces trajectory consistency in a parallel manner, significantly speeding up inference and training compared to recurrent approaches.
- Stacked slot attention blocks, enabling scalability and more efficient inference and training compared to the original iterative approach.
- An efficient transition model that incorporates causal slot information, slot interactions, and actions to predict next states effectively.

2 RELATED WORKS

2.1 OBJECT-CENTRIC LEARNING

Many modern object-centric representation models are based on the Slot Attention module [Locatello et al., 2020]. It implements an iterative attention mechanism, based on soft k-means clustering, for autoregressive slots refinement. Recent improvements of this method are based on better optimization [Chang et al., 2022], learnable slots initialization [Jia et al., 2023], slots structure augmentation [Singh et al., 2023], improving underlying clustering algorithm [Kirilenko et al., 2024] or simplifying it [Patil et al., 2024]. However, these methods use simple convolutional neural network (CNN) encoders and decoders, which result in worse image reconstructions. Authors of the SLATE [Singh et al., 2022a] proposed to use discrete latent space, which is extracted from the image by dVAE [Van Den Oord et al., 2017]. The computed slots are then passed through a transformer that predicts the latent token. This token is used in the dVAE decoder for image reconstruction. This results in better quality reconstructions than other models.

Classical object-centric models encounter issues with slot permutations. Specifically, when the same image is passed through the model multiple times, the same object may be allocated to a different slot each time. To address this problem, the SAVI [Kipf et al., 2022, Elsayed et al., 2022] and STEVE [Singh et al., 2022b] models propose the use of a predictor model that predicts the initialization of slots for use in Slot Attention for the subsequent frame. This predictor can be implemented as either a Multi-Layer Perceptron (MLP) or a Transformer.

Recent progress in object-centric learning has been closely linked to the adoption of pre-trained image transformers. Notably, DINOSAur [Seitzer et al., 2023] leverages pre-trained DINO transformers [Caron et al., 2021]. The primary training objective in this paradigm is to reconstruct patched-features maps from slots using a MLP as decoder. An extension of this method, VideoSAur, adapts DINOSAur for video data. However, achieving slot consistency within a trajectory through predictor models is challenging due to overfitting and gradient vanishing caused by the recurrent nature of training. Additionally, iterative processing of slot-attention models limit their inference speed and scalability.

2.2 OBJECT-CENTRIC WORLD MODELS

Previous object-centric world models have used other algorithms, than Slot Attention for objects states extraction. SQAIR [Kosiorek et al., 2018] uses a special detection model to detect objects in the scene and track their trajectories through RNNs. SCALOR [Jiang et al., 2020] and SILOT [Crawford and Pineau, 2020] are able to scale the number of objects in SQAIR due to parallel inference mechanism. The STOVE [Kossen et al., 2019] introduced a graph neural network (GNN) [Scarselli et al., 2009] for dynamics prediction and per-object interactions.

One of the notable world models is the Generative Structured World Model (G-SWM) [Lin et al., 2020]. It treats foreground objects and background separately by encoding them into two different latent vectors. For next state prediction, it uses two separate RNN, one for foreground latent vector, the other for background. One of the limitations of this model is that it doesn't consider actions taken in the environment when predicting future trajectories.

Another important approach is contrastive learning of structured world models (C-SWM) [Kipf et al., 2020]. It uses contrastive loss function on slots level, instead of basic image reconstruction loss. The model predicts the trajectory using GNN. Some notable improvements of the C-SWM are negative sampling [Biza et al., 2021], which improves the loss function by either selecting negative samples from different time steps in the same episode or the same time step in different episodes. Another improvement is two types of action attention [Biza et al., 2022]. Soft attention uses simple self-attention with a single head of transformers [Vaswani et al., 2017]. Hard attention calculates the expectation of all possible assignments to objects and takes the index of the object with the highest probability and maps the action to that object. The quality of predictions in models utilizing contrastive loss is significantly influenced by the quality of negative samples [Biza et al., 2021]. The necessity for data deduplication imposes additional constraints on the dataset sampling process, complicating it further in complex, real-world environments. However, since our model employs only Mean Squared Error (MSE) loss for training, it does

not impose strict requirements on the datasets.

The Slotformer [Wu et al., 2023b] uses ether STEVE or SAVI to extract slots from a sequence of images. The slots are then passed through a transformer to predict future frames of the video. This approach has achieved better results in dynamics prediction than previous models. Further improvement is the OCVP [Villar-Corrales et al., 2023] which decouples slots interaction and causal dynamics into 2 attention blocks. However, the main limitation of this method is that, unlike our approach, this model cannot work efficiently in action-dependent environments. Since taken action can drastically change environment or agent state, predicting the future based only previous states may prove to be challenging.

3 METHOD

The COMPAS world model consists of two interconnected components (as shown in figure 1): the slots extractor and the transition model. The purpose of the slots extractor is to generate object-centric slot representations that remain consistent both within a trajectory and across episodes. Consistency ensures that each slot corresponds to a specific object in the scene, maintaining its association without permuting with other slots over time or across episodes. The transition model predicts the next time step based on a context window of length T containing previous slot representations and the actions taken.

3.1 SLOTS REPRESENTATION EXTRACTION

Contrary to iterative approaches used in the classical slot-attention algorithm (or its variations), we utilize sequentially stacked blocks. This design enables increased parallelization, scalability, and faster inference and training. The overall architecture of the slots extractor module is presented in Figure 1.

Firstly, we initialize slots by duplicating a trainable token of size D for $T \times N$ times, resulting in a slots initialization $\hat{S}_t^{init}$. T represents the starting time window dimension and N is the number of slots. By adding a trainable slots position embedding p_s, which encodes each token with specified slot position and sinusoidal time position embeddings p_t to the corresponding slots, we obtain the slots initialization at time step t:

$$S_t^{init} = \hat{S}_t^{init} + p_s + p_t \tag{1}$$

The slot's positional encoding remains constant across time steps, but differs for each slot. Similarly, the time positional encoding is the same for all slots within a single time step, but varies between time steps. This deterministic initialization, combined with a trainable token and slot position embeddings, ensures slot consistency across episodes.

After initialization, we feed S_t^{init} into the slots-extraction block. To generalize notation, let S_t^l represent the slots from the l-th layer of the slots-extractor block, where $S_t^0 = S_t^{init}$. This block extracts slot representations from encoder features $F_t \in \mathbb{R}^{K \times J}$, where K is the number of patches, and J denotes the encoder feature dimensions. We utilize a modified version of the shared key slot attention [Patil et al., 2024]. The encoder features F_t are extracted from input image frames using a pre-trained DINO [Caron et al., 2021] encoder:

$$\begin{aligned} Q &= \text{Linear}(S_t^{l-1}) \\ K &= \text{Linear}(F_t) \\ W &= \text{Softmax}\left(\frac{KQ}{\tau}, \text{axis} = \text{slots}\right) \\ R &= W^T K \\ \tilde{S}_t^l &= \text{MLP}\,(R) + S_t^{l-1} \end{aligned} \tag{2}$$

Object-centric models for video, such as the SAVI or Videosaur models, implement slot trajectory consistency through a recurrent predictor model, which predicts slot initialization for the next time step based on the slots from the previous time step. However, this approach may be less performant and is prone to gradient decay. To address these issues and improve training and inference speed, we introduce a temporal consistency block. This block processes slots in parallel and is implemented as a transformer encoder layer with a mask M_{temp}. The mask promotes attention to the same slots from previous time steps causally, while ignoring other slots. In other words, the i-th slot $\tilde{s}_{t,i}^l \in \tilde{S}_t^l$ can only attend to the same slot $\tilde{s}_m^{i,l}$ from a previous time step m ($m < t$). The visualization of the matrix is shown on the Figure 3.

By feeding the sequence of extracted slots on the l-th layer in a time window of size T, $\left\{\tilde{S}_t^l\right\}_{t=1}^T$, into the temporal consistency block, we obtain the final slots:

$$\left\{S_t^l\right\}_{t=1}^T = \text{TransfEncLayer}(\left\{\tilde{S}_t^l\right\}_{t=1}^T, \text{mask} = M_{\text{temp}}) \tag{3}$$

Assuming that the slots extractor model comprises L layers, sequentially implementing these operations results in the final slots: $\{S_t\}_{t=1}^T = \left\{S_t^L\right\}_{t=1}^T$

Since the context length of the model is constrained to a fixed time window T, which is typically small, challenges arise regarding the scalability of the model for longer videos. To address this issue, the slots-extractor module is trained

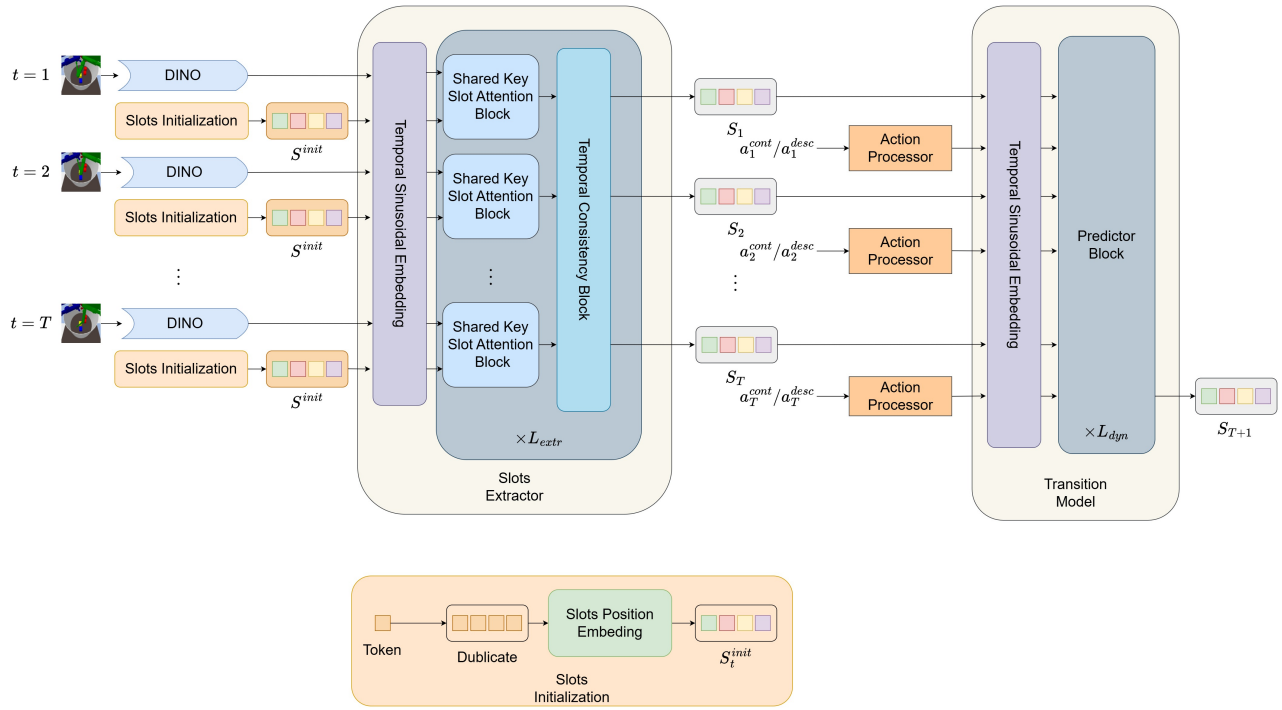

Figure 1: The proposed method begins by inputting a observations trajectory of size T into a frozen DINO encoder to extract feature representations. Deterministic slot initializations are generated by duplicating trainable tokens to match the desired number of slots N, followed by adding trainable slot position encodings, resulting in slot initializations $S^{\text{init}}t$. These initializations, along with the feature representations, are passed into a slot extractor model, which incorporates temporal sinusoidal positional encoding. The slot extractor model consists of two blocks: a shared key-slot attention block that maps each slot to the corresponding observation feature patch and a temporal consistency block that ensures slot consistency across the trajectory. After processing through Lextr layers, the slots extractor outputs slot representations S_t at each time step t. The transition model then predicts the next state based on actions within the time window T, handling both discrete (a_t^{desc}) and continuous (a_t^{cont}) actions. Actions are processed into vector representations via the action processing block. Then extracted slots and processed actions are embedded with temporal position encodings, and passed into predictor block, which is shown on the figure 2. After L_{dyn} layers in the transition model, the next slots S_{T+1} are obtained at the T-th step from the features output.

in an autoregressive manner. The initialization of the next slots, S_{T+1}^{init}, is performed as described in Equation 1. Subsequently, the slots are extracted using the procedure detailed in Equations 2.

The extracted slots, $\tilde{S}_{T+1}^{l}$, are then appended to the shifted time window of previously extracted slots, $\{S_t\}_{t=2}^{T}$, resulting in the sequence $\left\{S_2, S_3, \ldots, \tilde{S}_{T+1}^{l}\right\}$. These slots are then fed into the temporal consistency block described in equation 3. By sequentially processing them through all L layers, the final slots, S_{T+1}, are obtained. This procedure can be repeated for any desired horizon beyond the initial context length T.

The training approach follows the methodology outlined in DINOSAUr [Seitzer et al., 2023]. After extracting the slots S_t across the entire training trajectory $T + Z$, where Z represents the prediction horizon for autoregressive extraction, these slots are processed through a shared MLP decoder to generate feature reconstructions $\hat{F}_t$. The training objective minimizes the Mean Squared Error (MSE) loss: $\mathcal{L}_{\text{extr}} = \frac{1}{T+Z} \sum_{t=1}^{T+Z} \|F_t - \hat{F}_t\|^2$

3.2 TRANSITION MODEL

The transition model is inspired by OCVP. During the early phases of training the slots extractor, slots often lack useful information, which can push the transition model away from the optimal solution or cause it to become stuck in a local minimum. To address these challenges and accelerate the training process, the transition model is trained separately from the slots extractor on precomputed slots.

At the start, slots $S_t \in \mathbb{R}^{N \times D}, t = 0, \ldots, T$ within a time window of size T are projected into the transition model's dimension. Since the transition model operates with the same dimensions D as the slots:

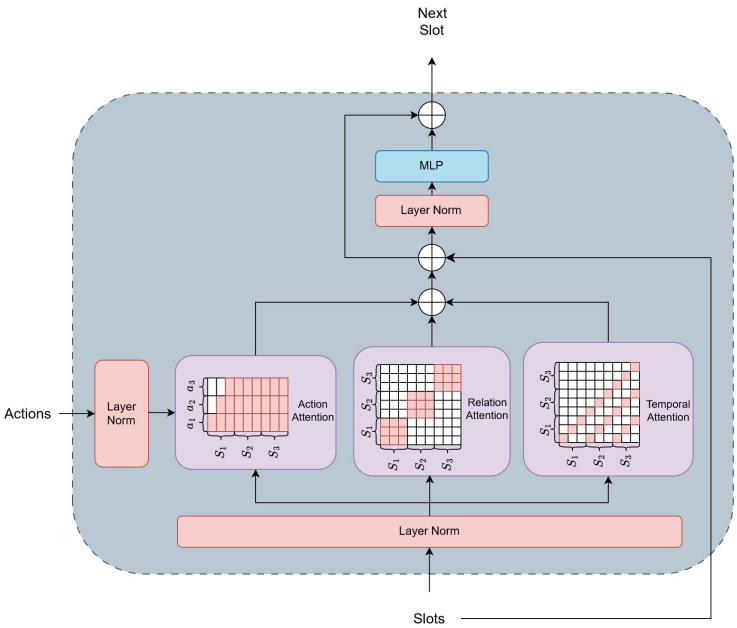

Figure 2: Architecture of the dynamcis predictor block

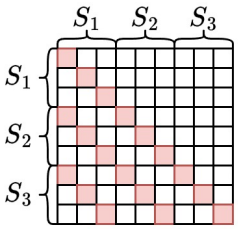

Figure 3: Attention mask used in temporal consistency block.

$$S_{\text{prep},t} = \text{Linear}(S_t) + p_t \tag{4}$$

The transition model supports both continuous and discrete actions. Continuous actions $a_t^{\text{cont}} \in \mathbb{R}^B$ are projected into the same dimension as the transition model:

$$\hat{a}_t \in \mathbb{R}^D = \text{Linear}(a_t^{\text{cont}}) \tag{5}$$

For discrete actions, we use trainable embeddings for each possible action $a_t^{\text{desc},i} \in A_t^{\text{desc}} = \{a_t^{\text{desc},1}, a_t^{\text{desc},2}, \ldots, a_t^{\text{desc},J}\}$:

$$\hat{a}_t = \text{EmbeddingsDictionary}(a_t^{\text{desc},i}) \tag{6}$$

To incorporate temporal information, we add sinusoidal time embeddings p_t to each slot and action at time step t:

$$\begin{aligned} s^i_{\text{prep},t} &= s^i_t + p_t, \quad s^i_{\text{prep},t} \in S_{\text{prep},t} \\ a_t &= \hat{a}_t + p_t \end{aligned} \tag{7}$$

Subsequently, actions and slots are passed into an action attention block, relation attention block, and temporal attention, in parallel as in original OCVP layer. The architecture of predictor block is presented at figure 2

Action attention is implemented as a simple multi-head cross-attention. In this layer, actions a_t serve as keys and values, while slots S_t act as queries.

After projecting the features from the final time step into slot dimensions, we obtain the predicted features for the next time step, denoted as S'_{T+1}. By shifting the input slot time window and appending S'_{T+1} as the last element, along with obtaining $\hat{a}_{T+1}$, we can autoregressively predict future states for any horizon L.

The training objective for the COMPAS dynamics model is defined as the L_2 reconstruction loss between the ground truth slots $s_{t,i} \in S_t$ and the model's predictions $s'_{t,i} \in S'_t$:

$$L_s = \frac{1}{L \cdot N} \sum_{t=T+1}^{L} \sum_{i=1}^{N} ||s'_{t,i} - s_{t,i}||^2 \tag{8}$$

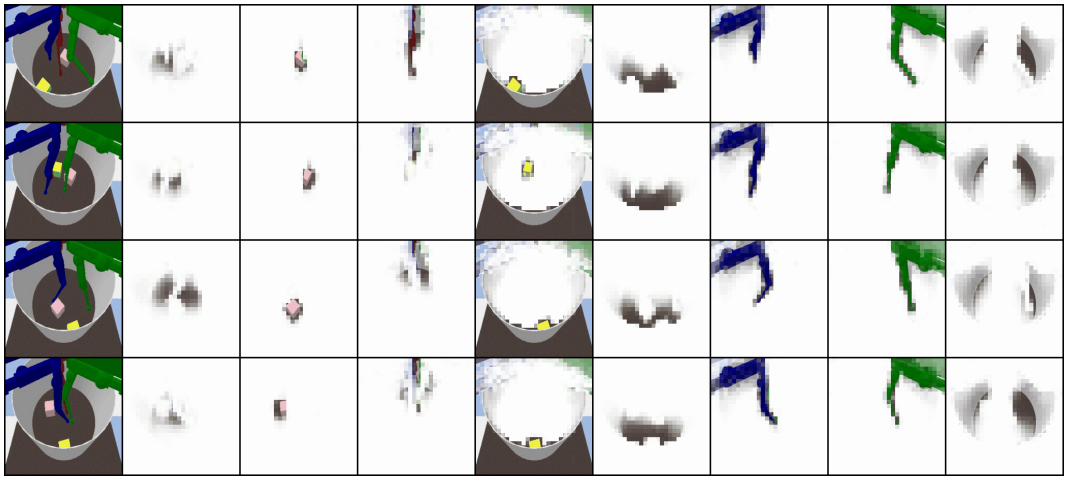

Figure 4: Episode consistent slots visualization extracted by COMPAS slots extractor. First column ground truth image, next columns – slots visualizations.

4 EXPERIMENTS

4.1 ENVIRONMENTS AND DATASETS

To evaluate the effectiveness of our proposed model, we perform experiments using trajectories sampled from three environments: Vizdoom [Wydmuch et al., 2019], Causal World [Ahmed et al., 2020] and RoboSuite [Zhu et al., 2022]. By conducting experiments in the selected environments, we aim to demonstrate the versatility and scalability of our model.

We primarily focus on comparing the predictive accuracy of our model against recent state-of-the-art (SOTA) methods, such as Slotformer and OCVP. Additionally, we include comparisons with the GNN transition model from CSWM, despite it being outdated and relying on a lower-quality slot extraction procedure.

ViZDoom. ViZDoom is a first-person perspective environment built upon the classic Doom video game engine. It offers a visually detailed and dynamic setting for experiments, combining elements such as partial observability and complex enemy behavior. We use the Defend the Line task within ViZDoom to evaluate our model's ability to operate in a discrete action space effectively. In this task, the agent is placed in a rectangular arena, starting at the center of one of the longer walls. On the opposite wall, three melee-only monsters and three shooting monsters are spawned. Enemies can be killed with a single shot. After being defeated, each monster respawns after a delay with increased durability, posing a progressively greater challenge. The discrete action space for this task consists of four actions: no action, turn left, turn right, and shoot.

We have collected datasets using a uniform random policy, consisting of 10,000 training episodes, each with 100 time steps, and 1,000 evaluation episodes of the same length. The observations consist of RGB images with a resolution of 224x224.

Causal World. At the other end of the complexity spectrum, we use the Causal World environment, which presents a much more complicated and challenging scenario. Causal World is a physically simulated environment with three robotic arms interacting with objects. It provides a continuous action space and requires the model to understand complex physical interactions and the impact of nuanced actions on the state of the world. In the Causal World environment, we consider two tasks, both with an action space of size nine. In the Push task, the goal of an agent is to push one block towards a goal position with a specific orientation. In the Reach task, the agent needs to reach target block from 4 blocks overall by manipulator. For the Reach and Push tasks, we follow the same data collection scheme as for the ViZDoom environment. Observations in the Causal World are rendered as images of size 224x224.

RoboSuite. We additionally experiment with a more difficult robotic environment, RoboSuite, which uses a Franka Panda robotic arm for manipulation. We choose the Block Lifting task, in which a robotic arm must pick up a red cube placed on the tabletop in front of it and lift it above a certain height. The cube location and the robot's initial states are randomized at the start of each and every episode. The size of the continuous action space for this task is four. The dataset was split into a training set containing 5000 trajectories and a validation set containing 1000 trajectories. Dataset was collected using uniform random policy. In RoboSuite environment, we capture images of size 256x256 from the front-view camera. These images are resized to 224x224 resolution.

Table 1: Comparative Results on the Causal World (Reach) Environment. Mean ± std for 3 Seeds. All values are multiplied by 100 for better visual presentation.

	10 STEPS		30 STEPS		60 STEPS	
MODELS	MSE	$\text{MSE}^{\text{DECODER}}$	MSE	$\text{MSE}^{\text{DECODER}}$	MSE	$\text{MSE}^{\text{DECODER}}$
SLOTFORMER	6.48 ± 0.02	15.35 ± 0.03	7.42 ± 0.12	17.95 ± 0.10	8.22 ± 0.19	19.30 ± 0.19
OCVP	6.32 ± 0.03	15.10 ± 0.04	7.28 ± 0.11	17.20 ± 0.13	8.08 ± 0.18	19.25 ± 0.17
GNN	5.18 ± 0.03	14.48 ± 0.03	6.75 ± 0.10	16.18 ± 0.12	7.88 ± 0.18	18.42 ± 0.18
COMPAS (OUR)	$\mathbf{3.18 \pm 0.01}$	$\mathbf{4.38 \pm 0.01}$	$\mathbf{4.23 \pm 0.07}$	$\mathbf{6.82 \pm 0.07}$	$\mathbf{5.12 \pm 0.15}$	$\mathbf{10.22 \pm 0.15}$

Table 2: Comparative Results on the Causal World (Push) Environment. Mean ± std for 3 Seeds. All values are multiplied by 100 for better visual presentation.

	10 STEPS		30 STEPS		60 STEPS	
MODELS	MSE	$\text{MSE}^{\text{DECODER}}$	MSE	$\text{MSE}^{\text{DECODER}}$	MSE	$\text{MSE}^{\text{DECODER}}$
SLOTFORMER	6.36 ± 0.02	15.33 ± 0.04	7.38 ± 0.10	17.94 ± 0.12	8.07 ± 0.18	19.08 ± 0.20
OCVP	8.65 ± 0.03	16.11 ± 0.05	9.40 ± 0.12	18.98 ± 0.15	10.04 ± 0.19	21.00 ± 0.18
GNN	5.20 ± 0.04	14.50 ± 0.03	6.80 ± 0.11	16.20 ± 0.14	7.90 ± 0.17	18.40 ± 0.19
COMPAS (OUR)	$\mathbf{3.15 \pm 0.01}$	$\mathbf{6.40 \pm 0.01}$	$\mathbf{4.25 \pm 0.08}$	$\mathbf{8.80 \pm 0.08}$	$\mathbf{5.10 \pm 0.16}$	$\mathbf{10.20 \pm 0.16}$

4.2 METRICS

Other papers, such as CSWM Biza et al. [2022], use ranking metrics like HITS@1 and MRR for dynamics prediction evaluation. However, this approach is not entirely representative, as these metrics depend on the scaling of the actual slot values and can yield high scores even if the predictions are inaccurate. To address this, we evaluate models using precomputed slots S_t and decode them back into observations O_t with sizes $H \times W$. If a model can accurately predict the next slots $\hat{S}_t$, the mean squared error (MSE) with the ground truth should be lower compared to competing models. Similarly, the decoder MSE should also be lower, as better dynamics predictions lead to improved reconstructions $\hat{O}_t$. We're calculating metric at certain time step t in order to evaluate quality of prediction at certain horizons.

$$\begin{aligned} \text{MSE}_t &= \frac{1}{N}||S_t - \hat{S}_t||^2 \\ \text{MSE}_t^{\text{decoder}} &= \frac{1}{3(H \times W)}||O_t - \hat{O}_t||^2 \end{aligned} \tag{9}$$

4.3 TRAINING AND EVALUATION SETUP

All transition models trained and evaluated on the precomputed slots, extracted by COMPAS slots extractor. Slots extractor was trained on 120 epochs with batch size 128 on each environmet.

More detailed description is available in Appendix. All configuration and code required for experiments will be available here.

4.4 RESULTS

As highlighted in Table 1, COMPAS demonstrated superior predictive accuracy in the Causal World environment with Reach task. This finding underscores the robustness of our approach in handling complex, physically simulated environments.

Furthermore, COMPAS' performance increased significantly in the more challenging Push task (Table 2). In contrast, non-action-dependent models, such as Slotformer and OCVP, exhibited a substantial performance degradation under these conditions. This comparison highlights the resilience and adaptability of COMPAS in diverse and complex tasks.

On the RoboSuite dataset, the GNN model showed much better results. This improvement is primarily attributed to the lower object density and reduced interaction complexity in this environment. Despite this, COMPAS still outperformed all baseline models on this dataset (Table 3).

On the ViZDoom dataset, COMPAS performed consistently well, achieving better results compared to alternative methods. The results are presented in Table 4.

Overall, COMPAS proves to be a more robust and adaptable

Table 3: Comparative Results on the Robosuite Environment. Mean ± std for 3 Seeds. All values are multiplied by 100 for better visual presentation.

	10 STEPS		30 STEPS		60 STEPS	
MODELS	MSE	$\text{MSE}^{\text{DECODER}}$	MSE	$\text{MSE}^{\text{DECODER}}$	MSE	$\text{MSE}^{\text{DECODER}}$
SLOTFORMER	1.70 ± 0.03	9.83 ± 0.08	1.99 ± 0.11	11.52 ± 0.14	2.65 ± 0.18	12.01 ± 0.20
OCVP	1.82 ± 0.05	10.55 ± 0.07	2.20 ± 0.13	12.19 ± 0.15	2.58 ± 0.19	12.94 ± 0.18
GNN	1.00 ± 0.02	9.46 ± 0.04	1.49 ± 0.12	12.08 ± 0.13	1.89 ± 0.17	12.97 ± 0.19
COMPAS (OUR)	$\mathbf{0.50 \pm 0.01}$	$\mathbf{8.08 \pm 0.03}$	$\mathbf{0.86 \pm 0.11}$	$\mathbf{10.53 \pm 0.12}$	$\mathbf{1.25 \pm 0.16}$	$\mathbf{11.93 \pm 0.17}$

Table 4: Comparative Results on the ViZDoom (Defent the line) Environment. Mean ± std for 3 Seeds. All values are multiplied by 100 for better visual presentation.

	10 STEPS		30 STEPS		60 STEPS	
MODELS	MSE	$\text{MSE}^{\text{DECODER}}$	MSE	$\text{MSE}^{\text{DECODER}}$	MSE	$\text{MSE}^{\text{DECODER}}$
SLOTFORMER	14.58 ± 0.05	6.51 ± 0.02	20.22 ± 0.12	14.94 ± 0.10	27.39 ± 0.18	30.31 ± 0.20
OCVP	14.10 ± 0.03	6.08 ± 0.03	20.41 ± 0.10	14.89 ± 0.12	26.86 ± 0.20	31.04 ± 0.18
GNN	8.75 ± 0.04	5.20 ± 0.03	15.60 ± 0.11	12.40 ± 0.11	23.10 ± 0.17	32.50 ± 0.19
COMPAS (OUR)	$\mathbf{4.31 \pm 0.01}$	$\mathbf{2.85 \pm 0.01}$	$\mathbf{9.43 \pm 0.08}$	$\mathbf{10.27 \pm 0.08}$	$\mathbf{21.23 \pm 0.16}$	$\mathbf{29.17 \pm 0.16}$

model, particularly in synthetic environments and tasks with discrete action spaces, further cementing its advantages over baseline methods.

5 CONCLUSION AND FUTURE WORKS

Our model demonstrates superior consistency over multiple prediction steps compared to analogous models in the field. This confirms the model's ability to handle different prediction horizons.

The main limitation of our model is the introduction of a new hyperparameter—the context length—which can impact the consistency of the slots. Additionally, the model's performance is influenced by hyperparameters such as the number of slots and the context time window, both of which can significantly affect the quality of slot representations and dynamics predictions. Another challenge is generalization, as the current model is designed to be trained for a specific environment and a single task. Future research could address these issues and improve the model's robustness and adaptability.

Looking to the future, a promising direction is to explore the neuro-symbolic perspectives of object-centric approaches. Neuro-symbolic reasoning combines the strengths of neural networks and symbolic reasoning, allowing models to learn from data while incorporating structured, symbolic knowledge. This could potentially improve the interpretability and robustness of object-centric world models, and allow them to better generalize from learned knowledge. Further improvements could also be made in the integration of actions into the model, potentially through the use of more sophisticated action representations or more advanced mechanisms for binding actions and slots.

In conclusion, our work represents an important step forward in object-centric world models. By integrating action information directly into the model and using an autoregressive space-state model for prediction, we have demonstrated a novel way to encode and predict the dynamics of complex environments. We expect that our work will inspire further research and advances in this exciting area of machine learning.

References

Ossama Ahmed, Frederik Träuble, Anirudh Goyal, Alexander Neitz, Yoshua Bengio, Bernhard Schölkopf, Manuel Wüthrich, and Stefan Bauer. Causalworld: A robotic manipulation benchmark for causal structure and transfer learning. *arXiv:2010.04296*, 2020.

Ondrej Biza, Elise van der Pol, and Thomas Kipf. The impact of negative sampling on contrastive structured world models. *ICML Workshop: Self-Supervised Learning for Reasoning and Perception*, 2021.

Ondrej Biza, Robert Platt, Jan-Willem van de Meent, Law-

son L. S. Wong, and Thomas Kipf. Binding actions to objects in world models. *ICLR 2022 workshop on Objects, Structure and Causality*, 2022.

Wolfram Burgard, Martial Hebert, and Maren Bennewitz. World modeling. *Springer handbook of robotics*, pages 1135–1152, 2016.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

Michael Chang, Tom Griffiths, and Sergey Levine. Object representations as fixed points: Training iterative refinement algorithms with implicit differentiation. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Eric Crawford and Joelle Pineau. Exploiting spatial invariance for scalable unsupervised object tracking. In *AAAI*, pages 3684–3692. AAAI Press, 2020.

Gamaleldin F. Elsayed, Aravindh Mahendran, Sjoerd van Steenkiste, Klaus Greff, Michael C. Mozer, and Thomas Kipf. SAVi++: Towards end-to-end object-centric learning from real-world videos. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Klaus Greff, Sjoerd Van Steenkiste, and Jürgen Schmidhuber. On the binding problem in artificial neural networks. *arXiv:2012.05208*, 2020.

David Ha and Jürgen Schmidhuber. World models. *arXiv:1803.10122*, 2018.

Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *CoRR*, abs/2010.02193, 2020. URL https://arxiv.org/abs/2010.02193.

Baoxiong Jia, Yu Liu, and Siyuan Huang. Improving object-centric learning with query optimization. *ICLR*, 2023.

Jindong Jiang, Sepehr Janghorbani, Gerard de Melo, and Sungjin Ahn. Scalor: Generative world models with scalable object representations. In *ICLR*, 2020.

Thomas Kipf, Gamaleldin F. Elsayed, Aravindh Mahendran, Austin Stone, Sara Sabour, Georg Heigold, Rico Jonschkowski, Alexey Dosovitskiy, and Klaus Greff. Conditional Object-Centric Learning from Video. In *International Conference on Learning Representations (ICLR)*, 2022.

Thomas N. Kipf, Elise van der Pol, and Max Welling. Contrastive learning of structured world models. In *ICLR*, 2020.

Daniil Kirilenko, Vitaliy Vorobyov, Alexey Kovalev, and Aleksandr Panov. Object-centric learning with slot mixture module. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=aBUidW4Nkd.

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.

Jannik Kossen, Karl Stelzner, Marcel Hussing, Claas Voelcker, and Kristian Kersting. Structured object-aware physics prediction for video modeling and planning. *arXiv:1910.02425*, 2019.

Baihan Lin, Djallel Bouneffouf, and Irina Rish. A survey on compositional generalization in applications. *arXiv:2302.01067*, 2023.

Zhixuan Lin, Yi-Fu Wu, Skand Peri, Bofeng Fu, Jindong Jiang, and Sungjin Ahn. Improving generative imagination in object-centric world models. In *ICML*, pages 6140–6149. PMLR, 2020.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention, 2020.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *ICLR*, 2023.

Sergiu Oprea, Pablo Martinez-Gonzalez, Alberto Garcia-Garcia, John Alejandro Castro-Vargas, Sergio Orts-Escolano, Jose Garcia-Rodriguez, and Antonis Argyros. A review on deep learning techniques for video prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(6):2806–2826, June 2022. ISSN 1939-3539. doi: 10.1109/tpami.2020.3045007. URL http://dx.doi.org/10.1109/TPAMI.2020.3045007.

Vihang Patil, Andreas Radler, Daniel Klotz, and Sepp Hochreiter. Simplified priors for object-centric learning, 2024. URL https://arxiv.org/abs/2410.00728.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In I. Guyon, U. Von 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. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/e6acf4b0f69f6f6e60e9a815938aa1ff-Paper.pdf.

F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

Maximilian Seitzer, Max Horn, Andrii Zadaianchuk, Dominik Zietlow, Tianjun Xiao, Carl-Johann Simon-Gabriel, Tong He, Zheng Zhang, Bernhard Schölkopf, Thomas Brox, and Francesco Locatello. Bridging the gap to real-world object-centric learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=b9tUk-f_aG.

Gautam Singh, Fei Deng, and Sungjin Ahn. Illiterate dall-e learns to compose. In *ICLR*, 2022a.

Gautam Singh, Yi-Fu Wu, and Sungjin Ahn. Simple unsupervised object-centric learning for complex and naturalistic videos. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022b. URL https://openreview.net/forum?id=eYfIM88MTUE.

Gautam Singh, Yeongbin Kim, and Sungjin Ahn. Neural systematic binder. In *ICLR*, 2023.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *NeurIPS*, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017.

Angel Villar-Corrales, Ismail Wahdan, and Sven Behnke. Object-centric video prediction via decoupling of object dynamics and interactions. In *Internation Conference on Image Processing (ICIP)*, 2023.

Philipp Wu, Alejandro Escontrela, Danijar Hafner, Pieter Abbeel, and Ken Goldberg. Daydreamer: World models for physical robot learning. In *Conference on Robot Learning*, pages 2226–2240. PMLR, 2023a.

Ziyi Wu, Nikita Dvornik, Klaus Greff, Thomas Kipf, and Animesh Garg. Slotformer: Unsupervised visual dynamics simulation with object-centric models. *arXiv:2210.05861*, 2022.

Ziyi Wu, Nikita Dvornik, Klaus Greff, Thomas Kipf, and Animesh Garg. Slotformer: Unsupervised visual dynamics simulation with object-centric models, 2023b.

Marek Wydmuch, Michał Kempka, and Wojciech Jaśkowski. ViZDoom Competitions: Playing Doom from Pixels. *IEEE Transactions on Games*, 11(3):248–259, 2019. doi: 10.1109/TG.2018.2877047. The 2022 IEEE Transactions on Games Outstanding Paper Award.

Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning, 2022.

Supplementary Material

A ARCHITECTURE DETAILS

Tables below describe architecture details and hyperparameters for our experiments.

Table 5: Hyperparameters of COMPAS Slots Extrator Used for Causal World Experiments.

Module	Parameter	Value
	Image Size	224
	Number of train episodes collected	10000
	Number of validation episodes collected	1000
	Number of steps per episode	100
	Number of training epochs	150
	Batch size	128
	Video length	8
DINO	Patch Size	8
DINO	Model Size	S
COMPAS	Slots dim	128
COMPAS	Number of slots	8
COMPAS	Number of layers	4

Table 6: Hyperparameters of COMPAS Slots Extrator Used for ViZDoom Experiments.

Module	Parameter	Value
	Image Size	224
	Number of train episodes collected	10000
	Number of validation episodes collected	1000
	Number of steps per episode	100
	Number of training epochs	150
	Batch size	128
	Video length	8
DINO	Patch Size	8
DINO	Model Size	S
COMPAS	Slots dim	128
COMPAS	Number of slots	8
COMPAS	Number of layers	4

Table 7: Hyperparameters of COMPAS Slots Extrator Used for Robosuite Experiments.

Module	Parameter	Value
	Image Size	224
	Number of train episodes collected	5000
	Number of validation episodes collected	1000
	Number of steps per episode	100
	Number of training epochs	150
	Batch size	128
	Video length	8
DINO	Patch Size	8
DINO	Model Size	S
COMPAS	Slots dim	128
COMPAS	Number of slots	4
COMPAS	Number of layers	4

B REBUTTAL

B.1 SLOT EXTRACTION

We work with a dataset of trajectories. Let $x_1, \ldots, x_t, \ldots$ denote a sequence of images constituting a trajectory. We then extract patch-encoded features from the images using a pre-trained frozen DINO encoder: $\hat{F}_t = \text{DINO}(x_t) \in \mathbb{R}^{K \times J}$ (where K is the number of patches of size J for a single image). Let N_{slot} denote the chosen number of slots. For time steps $1 \leq t \leq T$ within the context window of size T, the slots are initialized deterministically: $z_t^i = \mu_i$, for $1 \leq i < N_{\text{slot}}$, where μ_i are learnable parameters (equation (1) in the paper). Let f_θ denote a slot attention module with parameters θ implementing the neural network layers described in equations (2) and (3) in the paper. The slots for time steps within the context window $1 \leq t \leq T$ are obtained using the initialized values:

$$f_\theta(\hat{F}_t, z_t^1, \ldots, z_t^{N_{\text{slot}}}) \to s_t^1, \ldots, s_t^{N_{\text{slot}}}.$$

The slots for time steps from the autoregressive horizon $T < t \leq T + Z$ are calculated using the values of slots from the previous time step:

$$f_\theta(\hat{F}_t, s_{t-1}^1, \ldots, s_{t-1}^N) \to s_t^1, \ldots, s_t^{N_{\text{slot}}}.$$

Table 8: COMPAS Transition Model Hyperparameters

Environment	Model Dim	Memory length	Num. Layers	Heads	Dropout	Training epochs	Batch Size	Horizon
ViZDoom	128	4	8	4	0.1	100	64	4
Casual World	128	4	8	4	0.1	100	64	4
Robosuite	128	4	8	4	0.1	100	64	4

Table 9: Slotformer Transition Model Hyperparameters

Environment	Model Dim	Memory length	Num. Layers	Heads	Dropout	Training epochs	Batch Size	Horizon
ViZDoom	128	4	8	4	0.1	100	64	4
Casual World	128	4	8	4	0.1	100	64	4
Robosuite	128	4	8	4	0.1	100	64	4

Let g_η denote a decoder network with parameters η. We use it to reconstruct the ground truth DINO features $\hat{F}_t$ from the slot representations:

$$g_\eta(s_t^1, \dots, s_t^N) \to \tilde{F}_t.$$

We assume the multivariate Gaussian statistical model $P(F_t|\text{img}_t; \{\mu_i\}, \theta, \eta)$ with a diagonal identity covariance matrix Σ for the DINO features:

$$P(F_t|x_t; \{\mu_i\}, \theta, \eta) = \frac{1}{(\det \Sigma)^{0.5}(2\pi)^{0.5KJ}} \exp\left(-\frac{1}{2}(F_t - \hat{F}_t)^T \Sigma^{-1}(F_t - \hat{F}_t)\right)$$

We optimize the slot initialization parameters $\{\mu_i\}$ and the parameters of the neural networks f_θ and g_η by maximizing the likelihood of the observed data, assuming independence between time steps. This gives us an expression proportional to the MSE objective $\mathcal{L}_{\text{extr}}$ for slot extraction from the paper:

$$\begin{aligned}
\{\mu_i^*\}, \theta^*, \eta^* &= \arg\max P(\tilde{F}_1|x_1; \{\mu_i\}, \theta, \eta) \cdots P(\tilde{F}_{T+Z}|x_{T+Z}; \{\mu_i\}, \theta, \eta) \\
&= \arg\max \sum_{t=1}^{T+Z} \log P(\tilde{F}_t|x_t; \{\mu_i\}, \theta, \eta) \\
&= \arg\max -\frac{T+Z}{2} KJ \log 2\pi - \sum_{t=1}^{T+Z} \frac{\|\tilde{F}_t(\{\mu_i\}, \theta, \eta) - \hat{F}_t\|^2}{2} \\
&= \arg\min \sum_{t=1}^{T+Z} \|\tilde{F}_t(\{\mu_i\}, \theta, \eta) - \hat{F}_t\|^2
\end{aligned}$$

Table 10: OCVP Transition Model Hyperparameters

Environment	Model Dim	Memory length	Num. Layers	Heads	Dropout	Training epochs	Batch Size	Horizon
ViZDoom	128	4	8	4	0.1	100	64	4
Casual World	128	4	8	4	0.1	100	64	4
Robosuite	128	4	8	4	0.1	100	64	4

Table 11: GNN Transition Model Hyperparameters

Environment	Model Dim	Memory length	Num. Layers	Heads	Dropout	Training epochs	Batch Size	Horizon
ViZDoom	128	4	8	4	0.1	100	64	4
Casual World	128	4	8	4	0.1	100	64	4
Robosuite	128	4	8	4	0.1	100	64	4

When we train the transition model, we work with a dataset of slots extracted from the trajectory images. Let $\hat{S}_1 = (\hat{s}_1^1, \dots, \hat{s}_1^{N_{slots}}), \dots, \hat{S}_t = (\hat{s}_t^1, \dots, \hat{s}_t^{N_{slots}}), \dots$ denote a sequence of slot representations extracted from the sequence of images $x_1, \dots, x_t, \dots$ constituting a trajectory. Let $a_1, \dots, a_t, \dots$ denote a sequence of actions from the same trajectory. We project actions into the internal dimension of size D of the transition model by employing a learnable function $p_\nu(a_t) \to \hat{a}_t \in \mathbb{R}^D$. In the case of a continuous action space, p_ν is a neural network with parameters ν (equation (5) in the paper), while in the case of a discrete action space, p_ν is a set of learnable embeddings for each possible action (equation (6) in the paper).

Let h_ϕ denote the transition model with parameters ϕ (equations (4), (7), and Figure 2 in the paper). It takes a sequence of size T of slots and projected actions as input and predicts the values of slots for the next time step $T+1$:

$$h_\phi(\hat{S}_1, \dots, \hat{S}_T, \hat{a}_1, \dots, \hat{a}_T) \to \tilde{S}_{T+1} = (\tilde{s}_{T+1}^1, \dots, \tilde{s}_{T+1}^{N_{slots}}).$$

For time steps within the prediction horizon, $T+1 < t \leq T+L$, we predict slots in an autoregressive manner. For instance, for time step $t = T+2$, we obtain:

$$h_\phi(\hat{S}_1, \dots, \hat{S}_T, \tilde{S}_{T+1}, \hat{a}_1, \dots, \hat{a}_{T+1}) \to \tilde{S}_{T+2}.$$

We assume the multivariate Gaussian statistical model

$$P(S_{T+1} | S_1, \dots, S_T, \hat{a}_1, \dots, \hat{a}_T; \nu, \phi)$$

with a diagonal identity covariance matrix Σ for the slot representations of the next step:

$$P(S_{T+1} | S_1, \dots, S_T, \hat{a}_1, \dots, \hat{a}_T; \nu, \phi) = \frac{1}{(\det \Sigma)^{0.5} (2\pi)^{0.5 D N_{slots}}} \exp\left(-\frac{1}{2} (S_{T+1} - \hat{S}_T)^T \Sigma^{-1} (S_T - \hat{S}_T) \right).$$

We optimize the parameters of p_ν and h_ϕ by maximizing the likelihood of the observed data, assuming independence between time steps within the prediction horizon. This gives us an expression proportional to the MSE objective L_s (equation (8) in the paper) for slot extraction:

$$\begin{aligned}
\nu^*, \phi^* &= \arg\max P(\tilde{S}_{T+1} | \hat{S}_1, \dots, \hat{S}_T, \hat{a}_1, \dots, \hat{a}_T; \nu, \phi) \cdots P(\tilde{S}_{T+L} | \hat{S}_{L-1}, \dots, \tilde{S}_{T+L-1}, \hat{a}_L, \dots, \hat{a}_{T+L-1}; \nu, \phi) \\
&= \arg\max \sum_{t=T+1}^{T+L} \log P(\tilde{S}_t | \hat{S}_{t-T}, \dots, \tilde{S}_t, \hat{a}_{t-T}, \dots, \hat{a}_t; \nu, \phi) \\
&= \arg\max -\frac{L D N_{slots}}{2} \log 2\pi - \sum_{t=T+1}^{T+L} \frac{\|\tilde{S}_t(\nu, \phi) - \hat{S}_t\|^2}{2} \\
&= \arg\min \sum_{t=T+1}^{T+L} \|\tilde{S}_t(\nu, \phi) - \hat{S}_t\|^2.
\end{aligned}$$