CAUSALITY-INSPIRED OBJECT-CENTRIC MODELS FOR PLANNING WITH MONTE CARLO TREE SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce COMET (Causality-Inspired Object-Centric Models for Planning with Monte Carlo Tree Search), a model-based reinforcement learning algorithm that performs Monte Carlo Tree Search in a slot-structured latent space. COMET pairs a frozen unsupervised object-centric encoder with a transformer-based world model, in which actions are bound to objects through a novel action–slot fusion mechanism that is used in slot transition prediction. Policy and value heads use causality-inspired attention, modulating token interactions by learned per-slot relevance scores so that decision-making concentrates on task-relevant entities. COMET adds an explicit object-level inductive bias to MuZero-style latent planning. Across eight visually and dynamically diverse tasks from the Object-Centric Visual RL benchmark, ManiSkill, Robosuite, and VizDoom, COMET achieves a higher mean normalized score during the early stages of training compared to object-centric and monolithic baselines.

1 INTRODUCTION

Humans can reason about the consequences of their actions before acting by mentally simulating past experiences or possible future outcomes (Shiffrin et al., 2020). Motivated by this ability, world models have been introduced in reinforcement learning (RL) as a way to imitate the environment and improve learning efficiency (Ha & Schmidhuber, 2018). In model-based reinforcement learning (MBRL), an agent learns a model of the environment dynamics and uses it to generate imagined experiences, thereby reducing the need for real-world interactions.

MBRL methods have achieved strong performance across a wide range of tasks. Notable examples include the Dreamer family of algorithms (Hafner et al., 2020; 2021; 2025), which employ latent world models for long-horizon imagination, approaches based on Model Predictive Path Integral (MPPI) control for planning (Hansen et al., 2022; 2024), and methods that integrate Monte-Carlo Tree Search (MCTS) (Coulom, 2006; Schrittwieser et al., 2020) with learned models (Schrittwieser et al., 2020).

Despite this progress, learning accurate world models remains difficult in environments that are high-dimensional, non-stationary, and composed of multiple interacting objects. One of the challenges for visual environments lies in representing observations effectively. Most existing approaches rely on convolutional neural network (CNN) encoders (LeCun et al., 2015) that produce a single holistic representation of the input image. However, such representations may fail to capture object-level structure and interactions, which are often crucial for decision-making (Santoro et al., 2017). In complex scenes, small but task-relevant objects, dynamic backgrounds, or many irrelevant entities can significantly degrade agent performance (Liang et al., 2024).

Humans, by contrast, perceive the world as composed of discrete entities such as objects (Spelke & Kinzler, 2007), which enables efficient reasoning and planning. Object-centric RL represents the environment as a set of object-level components, where each component corresponds to an individual object. When instance segmentation masks are available, object representations can be extracted using CNN encoders, alternatively, supervised segmentation models (Cheng et al., 2024; Kirillov et al., 2023; Ravi et al., 2025) can be used, though they require annotated data. A large body of work instead focuses on unsupervised object-centric representation learning (Kipf et al., 2020; Lin et al., 2020; Locatello et al., 2020; Elsayed et al., 2022; Singh et al., 2022a;b; Seitzer et al., 2023; Zadaianchuk et al., 2023; Manasyan et al., 2025; Daniel & Tamar, 2024), which discovers structured

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

representations directly from raw images, making it suitable for reinforcement learning without external supervision.

Object-centric MBRL methods with an object-level world model explicitly represent object dynamics and interactions, enabling more focused and interpretable decision-making. Many real-world and simulated environments are inherently object-oriented: the reward is determined by interactions among multiple objects, yet at any given step only a small subset of them is relevant to the current decision. In robotic manipulation, for instance, a robot typically interacts with one object at a time, so actions affect only a few objects while the rest are largely irrelevant. Motivated by this, we hypothesize that explicitly modeling the importance of individual objects for decision-making can improve policy learning. We propose COMET, an object-centric MBRL algorithm based on MCTS, in which the world model maintains disentangled object-centric latents, and the policy and value models are transformers (Vaswani et al., 2017) over these latents with causality-inspired attention. The models process object tokens together with a dedicated target token for action or value prediction, and attention is modulated by learned causality scores to emphasize task-relevant objects.

In summary, our main contributions are as follows:

- We introduce COMET, an MCTS-based object-centric MBRL algorithm that combines frozen object-level representations with a transformer-based world model for planning in an object-structured latent space.
- We propose a novel action-object binding mechanism, where actions are fused with object-centric slots, effectively implementing a learned binding between actions and objects within a unified transformer backbone, enabling object-centric world modeling as well as policy/value prediction.
- We evaluate COMET across a diverse set of object-oriented visual control tasks, including object-centric benchmark environments and robotic manipulation tasks, and show that it shows consistent performance across tasks and, on average, achieves higher sample efficiency than both strong monolithic MCTS-based MBRL methods and object-centric RL baselines.

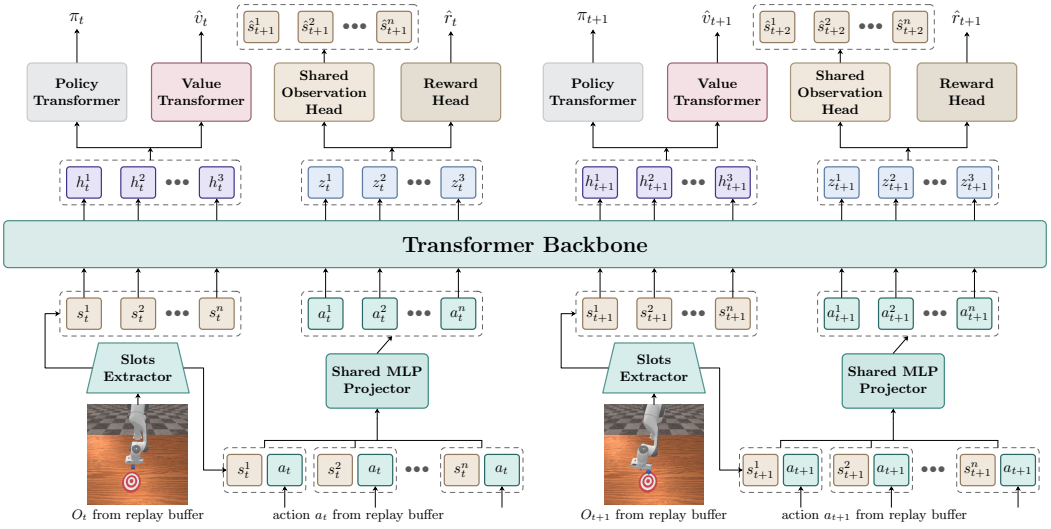

Figure 1: Overview of COMET training. A frozen slot extractor maps observations into slots, which are processed by a transformer backbone to produce latent representations $h_t^1, h_t^2, \ldots, h_t^n$. These latents, together with a learnable target token, are fed into the policy and value transformers to predict the action distribution or value. Next, an action embedding is concatenated with each slot independently and passed through a shared MLP projector, producing slot-conditioned action embeddings $a_t^1, a_t^2, \ldots, a_t^n$. These are processed by a transformer backbone to obtain $z_t^1, z_t^2, \ldots, z_t^n$, which are used to predict the next state (next slots) and reward.

2 Related Work

2.1 Object-Centric Representation Learning

A growing line of research focuses on learning structured object-centric representations directly from raw sensory inputs without manual annotations. Instead of encoding a scene into a single global vector, these methods decompose observations into sets of entities that can be processed independently. A key mechanism is Slot Attention (Locatello et al., 2020), which iteratively assigns a fixed number of latent slots to different parts of the input via competitive cross-attention. Subsequent work extends this idea to sequential data. SAVi (Kipf et al., 2022) and SAVi++ (Elsayed et al., 2022) introduce temporal consistency using motion cues such as optical flow and depth, enabling slots to persist across frames. Other approaches focus on improving reconstruction quality with more expressive generative models. SLATE and STEVE (Singh et al., 2022b) combine discrete latent tokenization (Van Den Oord et al., 2017) with transformer-based decoders and Slot Attention-based grouping. In contrast, DINOSAUR (Seitzer et al., 2023) replaces pixel reconstruction with feature-level objectives using pretrained DINO (Amir et al., 2021) representations to learn semantically meaningful objects. More recent work Slot Contrast (Manasyan et al., 2025) enforces alignment between slots across time by contrasting corresponding object representations, resulting in more robust tracking and reduced slot ambiguity in dynamic scenes.

Not all object-centric models rely on Slot Attention. Deep Latent Particles (DLP) (Daniel & Tamar, 2022) represent images as low-dimensional particles that decouple spatial position and appearance. In a different direction, Artificial Kuramoto Oscillatory Neurons (AKOrN) (Miyato et al., 2025) introduce oscillatory neural dynamics, where synchronized neurons form coherent groups corresponding to objects or parts.

2.2 Object Centric Reinforcement Learning

Recent studies incorporate object-centric representations into model-based reinforcement learning to better capture the compositional structure of environments. COBRA (Watters et al., 2019) learns a transition model over latent slots obtained from MONet (Burgess et al., 2019) and combines it with intrinsic motivation to improve data efficiency. FOCUS (Ferraro et al., 2023) uses an encoder-decoder architecture that segments scenes into object-specific latent variables via learned masks. OC-STORM (Zhang et al., 2025) employs a spatiotemporal transformer to jointly reason over object-centric and pixel-level representations for dynamics modeling. COBRA is limited by the lack of explicit modeling of object interactions, restricting its ability to capture relational dynamics addressed by our method. In contrast, FOCUS and OC-STORM rely on annotated segmentation masks, limiting their applicability in fully unsupervised settings.

Closer to our setting, STICA (Nishimoto & Matsubara, 2026) proposes an object-centric model-based RL framework combining slot-based representations with transformer-based world models and decision modules with causality-inspired attention between object representations. SOLD (Mosbach et al., 2024) learns object-centric latent dynamics directly from pixels without supervision via an action-conditioned slot-based dynamics model and a Slot Aggregation Transformer for policy and value learning. Object-Centric Dreamer (Ugadiarov et al., 2025) (OCDreamer) extends Dreamer by replacing the RSSM with an object-centric RSSM and incorporating GNNs to explicitly model object interactions during prediction and control. Beyond model-based approaches, object-centric representations are also used in model-free RL. OCRL (Yoon et al., 2023) integrates a transformer-based object encoder into PPO (Schulman et al., 2017), enabling flexible use of different object-centric features. Similarly, OC-CA and OC-SA (Stanić et al., 2023) use Slot Attention as a feature extractor and study its generalization across environments.

COMET is an MCTS-based object-centric model-based RL algorithm that performs planning in a slot-structured latent space. The method combines three components: a frozen object-centric encoder that maps visual observations to object slots, a transformer-based world model that predicts future slots and rewards, and policy/value heads equipped with causality-inspired attention. Our implementation builds on the LightZero framework (Niu et al., 2023) and follows the UniZero training pipeline (Pu et al., 2025). Unlike UniZero, which operates on monolithic state embeddings, COMET represents each observation as a set of object-centric slots and performs both dynamics prediction and decision-making over these slots. This design introduces an explicit object-level inductive bias

into MuZero-style latent planning, enabling the model to reason over individual entities and their interactions. As in UniZero, COMET uses the unified transformer backbone implemented using a nanoGPT-based architecture (Karpathy, 2023).

2.3 Slots Extractor

The slots extractor maps an image observation o_t into a set of object-centric latent representations. Specifically, it produces an unordered collection of vectors $\bar{s}_t = \{s_t^1, \ldots, s_t^n\}$, where n is a fixed hyperparameter conventionally defined as the maximum number of objects in the scene plus one slot for the background.

A key challenge of slot-based architectures, which are trained on static image observations, is that the ordering of slots is not guaranteed to be consistent across time steps due to the permutation-invariant and stochastic nature of slot attention. To mitigate this issue and ensure temporal consistency of object representations within an episode, we initialize the slot representations at time $t+1$ using the slots obtained at time t. This encourages stable assignment of slots to underlying objects over time. In contrast, this issue does not arise in video-based object-centric models, where temporal consistency is directly modeled within the architecture. For example, in Slot Contrast, such consistency is handled within the learning pipeline, and no additional slot-initialization mechanism is required, as temporal correspondence is learned end-to-end through the model design.

In our approach, slot extractors are pretrained on observations collected using a random policy and remain frozen during reinforcement learning. We experiment with different object-centric representation models depending on the task, including SLATE, DINOSAUR, and Slot Contrast.

2.4 Object-Centric Token Processing

UniZero uses a transformer backbone based on the nanoGPT architecture. It processes sequences of state and action embeddings arranged alternately in a single sequence. In discrete action spaces, actions are represented as learnable embedding vectors, while in continuous action spaces, actions are passed through a two-layer MLP to produce corresponding embeddings. The transformer processes the sequence in two stages. First, the state embedding z_t is fed into the backbone, producing latent h_t^z, which is then passed to the decision head to model policy and value. Next, the action embedding a_t is processed by the same transformer backbone, yielding latent h_t^a, which is passed through the dynamics head to predict the future state $\hat{z}_{t+1}$ and reward $\hat{r}_t$. UniZero employs standard causal attention masking and learnable absolute positional encodings, with the total sequence length bounded by the context size.

We adapt this architecture for object-centric representations. In object-centric settings, the state is represented as n slots, $\bar{s}_t = \{s_t^1, \ldots, s_t^n\}$. In our approach these slots are fed into the backbone, producing latent representations $\{h_t^1, \ldots, h_t^n\}$, which are then passed to the policy and value networks implemented as transformer modules with causality-inspired attention, as described in Section 2.5. Predicting the next state $\bar{\hat{s}}_{t+1} = \{\hat{s}_t^1, \ldots, \hat{s}_t^n\}$ is non-trivial. Generating n slots from a single action embedding a_t creates a bottleneck, as all object-centric latents must be compressed into a single vector. In our experiments, architectures using this approach perform poorly. To address this, we concatenate the action embedding with each slot, $a_t \oplus s_t^1, \ldots, a_t \oplus s_t^n$, and pass the resulting vectors through a shared MLP projector. This produces slot-conditioned action embeddings $\bar{a}_t = \{a_t^1, \ldots, a_t^n\}$, which are fed into the transformer backbone, yielding latents $\bar{z}_t = \{z_t^1, \ldots, z_t^n\}$. These latents are passed through a shared observation MLP head to predict the next-step object slots $\bar{\hat{s}}_{t+1}$. For reward prediction, $\bar{z}_t$ are summed into a single vector, which is then processed by a reward head implemented as an MLP. The input to the transformer is the sequence of $\bar{s}_t$ and $\bar{a}_t$, naturally decomposed into blocks, each corresponding to a single time step. As in UniZero, each block is augmented with a learnable absolute positional encoding, and the

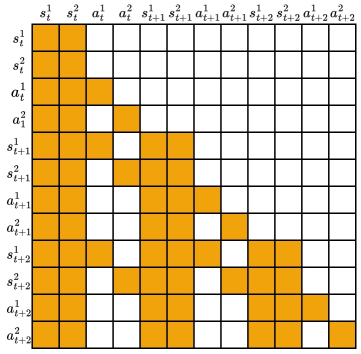

Figure 2: Illustration of the attention mask used in the transformer backbone for a setting with two slots per block across three time steps.

transformer's context size defines the total number of timesteps processed. In our attention mask, each slot embedding attends to all slots within the same block, all slots from previous blocks, and the action embeddings associated with its position. Each action embedding attends to itself as well as to all slots in the current and preceding blocks. The attention mask for the transformer backbone is shown in Figure 2. We view our slot-conditioned action embeddings as closely related to the mechanism of soft action–object binding (Biza et al., 2022), where each slot is influenced by a version of the source action conditioned on the current object state. The overall training pipeline and architecture are illustrated in Figure 1.

2.5 Causality-Inspired Attention

The policy and value networks are implemented as a transformer with the modified attention mechanism introduced in STICA (Nishimoto & Matsubara, 2026). Alongside the latent representations $h_t^1, \ldots, h_t^n$, a learnable target token, specific to the policy or value head, is provided as input; its transformer output is decoded by an MLP to produce the corresponding prediction. To model causal structure, the graph is defined over three groups of objects: the target, causal objects, and non-causal objects:

$$G = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{1}$$

where $G_{i,j} = 1$ indicates that group j exerts a causal influence on group i. Thus, causal objects influence the target ($G_{1,2} = 1$) and one another ($G_{2,2} = 1$), while non-causal objects influence only themselves ($G_{3,3} = 1$).

Since object causality is not known a priori, a causality score $\alpha_t^k \in [0, 1]$ is estimated for each latent h_t^k, denoting the probability that the corresponding object is relevant for the policy or value prediction.

$$W_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \alpha_t^1 & 1 - \alpha_t^1 \\ \vdots & \vdots & \vdots \\ 0 & \alpha_t^n & 1 - \alpha_t^n \end{bmatrix}, \tag{2}$$

whose first row corresponds to the target token and remaining rows to the latent object tokens. The product $W_t G W_t^\top$ lifts G to token-level interactions, encoding the strength of causal influence between every pair of tokens, and is used to modulate scaled dot-product attention:

$$\mathrm{CA}_t = \mathrm{Norm}\left(\mathrm{softmax}\left(\frac{Q_t K_t^\top}{\sqrt{d}}\right) \odot W_t G W_t^\top\right) V_t, \tag{3}$$

where Q_t, K_t, V_t are the query, key, and value matrices, d is the key dimensionality, $\odot$ is element-wise multiplication, and $\mathrm{Norm}(\cdot)$ denotes row-wise normalization. The mechanism thereby concentrates attention on objects that directly influence the target and suppresses irrelevant ones. Although we follow the terminology of STICA and refer to these quantities as causality scores, they should be interpreted as learned object-relevance weights rather than independently identified causal effects.

2.6 Policy and World Model Learning

MuZero-like methods learn a latent model for planning with MCTS rather than using the true environment dynamics (Schrittwieser et al., 2020). The model consists of a representation function, a dynamics function, and a prediction function. The representation function encodes the observation history into a root latent state $x_t = h_\theta(o_{\leq t}, a_{<t})$. The dynamics function predicts imagined transitions and rewards, $\hat{r}_t, x_{t+1} = g_\theta(x_t, a_t)$, and the prediction function outputs a policy prior and value estimate, $\pi_t, \hat{v}_t = f_\theta(x_t)$. MCTS is then performed entirely in latent space. The learned dynamics expands candidate future trajectories, while the prediction function evaluates latent states and provides action priors. After a fixed number of simulations, the visit counts $N(x_t, a_t)$ at the root are normalized to produce an improved policy target p_t

$$p_t = \frac{N(x_t, a_t)^{1/\mathbb{T}}}{\sum_{b_t} N(x_t, b_t)^{1/\mathbb{T}}}, \tag{4}$$

where $\mathbb{T}$ denotes the temperature, which modulates the extent of exploration, the visit count $N(x_t, a_t)$ denotes the number of times action a_t was selected at the root latent state x_t during MCTS. The denominator $\sum_{b_t} N(x_t, b_t)^{1/\mathbb{T}}$ sums over all possible actions b_t from the same state x_t for discrete action spaces, or over sampled candidate actions in continuous-control tasks, where $N(x_t, b_t)$ is the visit count for action b_t. The model is trained end-to-end by unrolling the dynamics for K steps and optimizing policy, value, and reward prediction losses. This framework forms the basis of several MuZero-style algorithms (Schrittwieser et al., 2020; Hubert et al., 2021; Ye et al., 2021; Wang et al., 2024).

While standard MuZero-like methods represent the planning state as a monolithic latent vector or feature map, COMET represents it as a set of object-centric slots, enabling planning and prediction over slot-structured latent space. The joint optimization objective for COMET can be written as

$$\mathcal{L}_{\text{COMET}}(\theta) = \mathbb{E}_{(o_t, a_t, r_t, o_{t+1}, p_t)\sim\mathcal{B}}\Big(\sum_{t=0}^{H-1} \Big(\beta_s \frac{1}{n} \sum_{i=1}^{n} \|\hat{s}^i_{t+1} - s^i_{t+1}\|_2^2 + \beta_r \,\text{CE}(\hat{r}_t, r_t) + \beta_p \,\text{CE}(\pi_t, p_t) + \beta_v \,\text{CE}(\hat{v}_t, v_t) \Big) \Big), \tag{5}$$

where $\mathcal{B}$ is a replay buffer that stores trajectories $\{o_t, a_t, r_t, o_{t+1}, p_t\}$. H denotes the training context length, which corresponds to the rollout length used during training. In COMET, H matches the context window of the transformer backbone. The coefficients $\beta_s, \beta_r, \beta_p, \beta_v$ are constant coefficients used to balance different loss terms, corresponding to next-state prediction, reward prediction, policy prediction, and value prediction, respectively. CE denotes the cross-entropy loss function. Following UniZero, we formulate reward and value prediction as a discrete regression problem in a log-transformed space, optimized by minimizing cross-entropy using v_t and r_t as soft targets. $\hat{s}^i_{t+1}$ denotes the predicted representation of the i-th slot, while s^i_{t+1} denotes the corresponding ground-truth slot obtained from a frozen pretrained object-centric encoder. v_t denotes the bootstrapped n-step TD target, and r_t denotes the target reward.

3 Experimental Setup

3.1 Environments



Figure 3: Visualization of observations and slot-wise attention maps across environments. In each row, the real observation is followed by attention maps for each slot produced by the corresponding model. From top to bottom: SLATE in Object Reaching Task, SLATE in an Object Goal Task, Slot Contrast in Cube Pushing Task, DINOSAUR in Block Lifting Task, and Slot Contrast in Defend The Line Task.

We evaluate our approach on the Object-Centric Visual RL benchmark (Yoon et al., 2023), which includes object-centric environments designed to test perception, interaction, and relational reasoning. The suite consists of Object Goal, Object Interaction, Object Comparison, Property Comparison, and Object Reaching tasks, featuring target objects with distractors and requiring different forms of goal-directed behavior. Across tasks, the agent must identify relevant objects, reason about their properties or relationships, and act under sparse rewards with either discrete or continuous action spaces. We further extend the evaluation to manipulation and control tasks from ManiSkill, Robosuite, and VizDoom. Figure 3 shows examples of observations from the considered environments.

In the ManiSkill framework, we use the Cube Pushing task, where a cube is placed on a tabletop and its initial position is randomly sampled within a small region in front of the agent. The goal is to push the cube into a target region at a fixed offset from its initial position, indicated by a visual marker. The reward is dense and shaped, with pose-based components weighted by $\texttt{pose_reward_coef} = 0.01$ and $\texttt{place_reward_coef} = 0.1$. Episodes are limited to 50 steps.

In Robosuite, we evaluate the Block Lifting task, where a single Panda arm operates in a tabletop environment. A cube is placed on the table at a fixed position, while the robot's initial configuration is randomized at each episode. The objective is to grasp and lift the cube above a predefined height threshold. The task uses a dense reward that encourages gradual progress toward successful lifting, requiring stable grasping and vertical manipulation. Episodes are limited to 125 steps.

Finally, in VizDoom we use the Defend the Line scenario, where the agent is placed on one side of a rectangular map, while melee and ranged monsters spawn on the opposite side and continuously move toward it. Monsters are eliminated with a single shot and respawn after a delay; over time they deal increasing damage. The agent has limited ammunition, and the episode ends when the agent dies. The reward is +1 for killing a monster and -1 for death.

3.2 Mean Normalized Score

The performance and sample efficiency of object-centric RL depend on the quality of learned representations, which are strongly affected by visual complexity. Although agents can sometimes compensate for imperfect representations, this usually reduces sample efficiency and final performance, and different architectures vary in their robustness across environments. Therefore, a key goal is to design object-centric RL agents that maintain stable performance across visually diverse tasks under limited interaction budgets.

Our task suite spans environments with varying visual complexity, ranging from relatively simple settings (e.g., 2D shapes on a monochrome background) to more challenging ones (e.g., ManiSkill, Robosuite, and VizDoom). To quantify performance under a fixed interaction budget, we compute the average performance across environments. To this end, we introduce a normalized score that aggregates an agent's performance across a set of visually and dynamically diverse tasks into a single interpretable metric. Because evaluation metrics (e.g., cumulative reward and success rate) differ in scale and nature across tasks, raw scores must be normalized prior to aggregation. We normalize each agent's performance relative to the best-performing method in each environment. Let $\mathcal{E}$ denote the set of environments, $\mathcal{T}_e$ the time-step budget for environment $e \in \mathcal{E}$, $\mathcal{M}$ the set of RL agents (algorithms) we evaluate, $\tau = t/\mathcal{T}_e \leq 1$ the normalized time step, and $F(m, e; t)$ the evaluation metric (success rate or cumulative reward) achieved by method $m \in \mathcal{M}$ on environment e after t training steps. We define the normalized score $\mathcal{S}(m, e; \tau)$ for method m in environment e, and the mean normalized score $\hat{\mathcal{S}}(m; \tau)$ that aggregates performance across environments:

$$\mathcal{S}(m, e; \tau) = \frac{F(m, e; t)}{\max_{m', t'} F(m', e; t')} \leq 1, \quad \hat{\mathcal{S}}(m; \tau) = \frac{1}{|\mathcal{E}|} \sum_{e \in \mathcal{E}} \mathcal{S}(m, e; \tau). \tag{6}$$

4 Experiments

We compare COMET against a model-free, object-centric PPO baseline OCRL (Yoon et al., 2023) that uses a transformer encoder to pool object-centric representations. As object-centric MBRL baselines, we use OCDreamer. For completeness, we also compare against SOLD, but report its results only on tasks with continuous action spaces, since it is applicable only to that setting. For

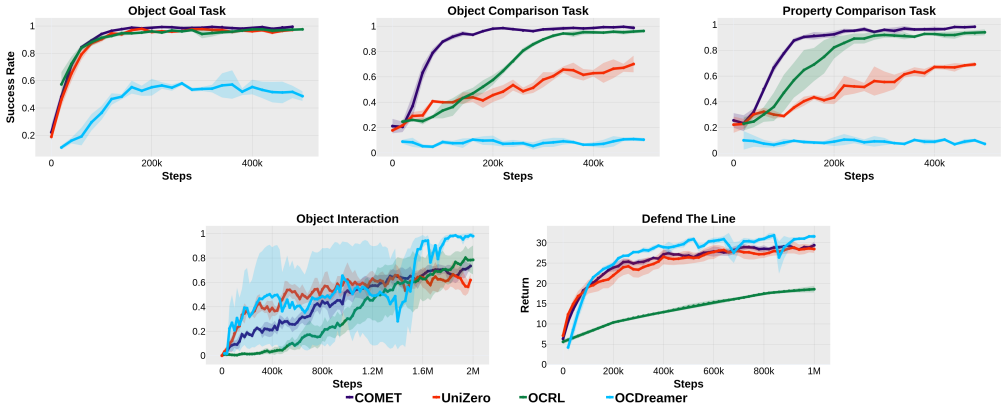

Figure 4: Success rate averaged over 30 episodes and three seeds for COMET and baselines in tasks with discrete action space. Shaded areas indicate standard deviation. Exponential smoothing with coefficient 0.5 is applied.

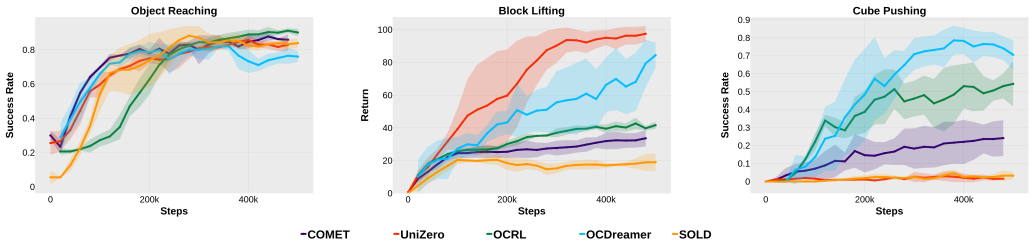

Figure 5: Success rate averaged over 30 episodes and three seeds for COMET and baselines in tasks with continuous action space. Shaded areas indicate standard deviation. Exponential smoothing with coefficient 0.5 is applied.

SOLD, we use the SAVi encoder from the original implementation; examples of its attention maps on continuous tasks are provided in Appendix L. For OCRL, OCDreamer, and COMET, we use SLATE on the Object Goal, Object Interaction, Property Comparison, Object Property, and Object Reaching tasks; DINOSAUR on Block Lifting; and Slot Contrast on Cube Pushing and Defend the Line. Examples of attention maps produced by these encoders are shown in Figure 3. Pretrained SLATE is used as described in Appendix E, while all other encoders are trained on the collected data (Appendix D) and kept frozen thereafter. Hyperparameters for SLATE, Slot Contrast, and DINOSAUR are provided in Appendices E, G, and F, respectively. As a monolithic MBRL baseline, we use UniZero. For all baselines, we adopt the original hyperparameters specified in their respective publications and official repositories. Hyperparameters for COMET are provided in Appendix H. For the Cube Pushing task, we reduce `discount_factor` to 0.925 for both COMET and UniZero. True and predicted trajectory rollouts for all tasks are presented in Appendix J, and causality-score visualizations for the policy and value transformers across all tasks are presented in Appendix K.

Figures 4 and 5 report training curves for COMET and the baselines across all tasks. The set of top-performing algorithms changes across tasks. COMET achieves faster convergence and higher final performance than the baselines on Object Comparison and Object Property, but does not outperform the competing methods on Block Lifting and Cube Pushig tasks. To account for variability in task difficulty and performance scales, we normalize the results per task for each algorithm as described in (6). The results in Table 1 demonstrate that COMET achieves a higher mean normalized score across a visually diverse set of tasks at early training stages.

In visually simple tasks, such as Object Goal, Object Comparison, and Property Comparison, COMET leverages strong object-centric representations and a structure with a single target object

Table 1: Mean normalized score (6) versus normalized steps. Normalization parameters are listed in Appendix A. The normalized score is averaged over all considered tasks — Object Goal, Object Interaction, Object Comparison, Property Comparison, Object Reaching, Block Lifting, Cube Pushing, and Defend The Line. The best result in each row is shown in **bold**. COMET demonstrates the highest mean normalized score across the eight visually and dynamically diverse tasks during early training. SOLD is excluded from the table, as it is not applicable to tasks with a discrete action space.

Normalized step	COMET (ours)	OC-Dreamer	OCRL	UniZero
0.33	**0.633**	0.445	0.505	0.552
0.66	**0.727**	0.537	0.701	0.671
1.00	0.758	0.627	**0.778**	0.705

and multiple distractors. Using causality-inspired attention, which benefits from high-quality representations, COMET accurately identifies task-relevant objects by assigning them higher causality scores, as illustrated in Appendix K, enabling it to outperform the baselines. In more dynamically complex tasks, such as Defend the Line and Object Interaction, COMET achieves performance comparable to the baselines. We attribute this behavior to task-specific factors: in Defend the Line, most objects are relevant for prediction, while in Object Interaction, the agent must rely on object-pushing mechanics, so the goal is achieved indirectly through deeper causal chains than those captured by our causal object attention mechanism. For tasks that are challenging in both visual complexity and control, such as Block Lifting and Cube Pushing, COMET achieves moderate performance. This is likely due to limitations in the object-centric representation model, which sometimes merges the cube and the background into a single slot. OCDreamer demonstrates better performance on these tasks, indicating that it is more robust to imperfect representations in such settings. Curves for the mean normalized score are provided in Appendix A, the results of the ablation study on action-slot binding and slot pre-initialization are provided in Appendix B, and the results of the additional experiment on online fine-tuning of the object-centric encoder are provided in Appendix C.

5 Limitations & Future Work

Despite advances in unsupervised object-centric representation learning, current methods still struggle to reliably segment complex, cluttered real-world scenes, especially under occlusion or ambiguous boundaries, limiting their applicability in unconstrained settings. Additionally, transformer-based approaches scale poorly due to quadratic self-attention costs with increasing object slots, restricting scalability in multi-object scenes. Future work will focus on extending these methods to realistic, open-ended environments, such as household tasks, and on improving causal attention mechanisms that dynamically prioritize relevant objects. Enhancing these mechanisms for not only policy and value estimation but also transition dynamics could improve efficiency, relational reasoning, and scalability in complex, multi-object settings.

6 Conclusion

In this work, we introduced COMET, an MCTS-based object-centric model-based reinforcement learning method that combines structured object-centric representations with a transformer-based world model. COMET employs an action-slot binding mechanism that fuses object-centric slots with actions, enabling transition modeling for slots within a unified transformer backbone. By leveraging a causal attention mechanism, the policy and value models focus on task-relevant object representations, improving both the effectiveness and interpretability of decision-making. Experimental results across visually diverse discrete and continuous control tasks show that COMET achieves a higher mean normalized score during the early stages of training compared to object-centric and monolithic baselines.

AI use statement

(This section is **required** and does not count toward the page limit.)

In this work, generative AI tools were used to implement methods only in the form of writing software code via a code-assistant AI agent. We did not use generative AI tools to generate synthetic datasets, propose or refine hypotheses, design or provide feedback on research methodology or experiments, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, interpret results. Formulating mathematical claims, providing critical ingredients for proving them, and assisting in writing proofs are not applicable to this work. In addition, we used generative AI tools to create or modify scientific figures, create or edit software code, draft parts of the paper, summarize or analyze existing literature, discover research topics or identify gaps, search for information, edit the paper to improve readability, identify relevant literature, format references, suggest a structure for the paper, and propose a title and keywords. All AI-assisted work was reviewed: the LLM-generated code was reviewed by two authors, and the text of the paper was proofread by three authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Shir Amir, Yossi Gandelsman, Shai Bagon, and Tali Dekel. Deep vit features as dense visual descriptors. *arXiv preprint arXiv:2112.05814*, 2021.

Ondrej Biza, Robert Platt, Jan-Willem van de Meent, Lawson LS Wong, and Thomas Kipf. Binding actions to objects in world models. *arXiv preprint arXiv:2204.13022*, 2022.

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.

Ho Kei Cheng, Seoung Wug Oh, Brian Price, Joon-Young Lee, and Alexander Schwing. Putting the object back into video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3151–3161, 2024.

Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pp. 72–83. Springer, 2006.

Tal Daniel and Aviv Tamar. Unsupervised image representation learning with deep latent particles. In *International Conference on Machine Learning*, pp. 4644–4665. PMLR, 2022.

Tal Daniel and Aviv Tamar. DDLP: Unsupervised object-centric video prediction with deep dynamic latent particles. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=Wqn8zirthg.

Gamaleldin 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. *Advances in Neural Information Processing Systems*, 35:28940–28954, 2022.

Stefano Ferraro, Pietro Mazzaglia, Tim Verbelen, and Bart Dhoedt. FOCUS: Object-centric world models for robotic manipulation. In *Intrinsically-Motivated and Open-Ended Learning Workshop @NeurIPS2023*, 2023.

David Ha and Jürgen Schmidhuber. World models. *CoRR*, abs/1803.10122, 2018. URL http://arxiv.org/abs/1803.10122.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0oabwyZbOu.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025.

Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Oxh5CstDJU.

Nicklas A Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In *International Conference on Machine Learning*, pp. 8387–8406. PMLR, 2022.

Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Mohammadamin Barekatain, Simon Schmitt, and David Silver. Learning and planning in complex action spaces. In *International Conference on Machine Learning*, pp. 4476–4486. PMLR, 2021.

Andrej Karpathy. nanogpt: The simplest, fastest repository for training/finetuning medium-sized gpts. https://github.com, 2023.

Thomas Kipf, Elise van der Pol, and Max Welling. Contrastive learning of structured world models. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1gax6VtDB.

Thomas Kipf, Gamaleldin Fathy 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*, 2022.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4015–4026, 2023.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

Anthony Liang, Jesse Thomason, and Erdem Bıyık. Visarl: Visual reinforcement learning guided by human saliency. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2907–2912. IEEE, 2024.

Zhixuan Lin, Yi-Fu Wu, Skand Vishwanath Peri, Weihao Sun, Gautam Singh, Fei Deng, Jindong Jiang, and Sungjin Ahn. Space: Unsupervised object-oriented scene representation via spatial attention and decomposition. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkl03ySYDH.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in neural information processing systems*, 33:11525–11538, 2020.

Anna Manasyan, Maximilian Seitzer, Filip Radovic, Georg Martius, and Andrii Zadaianchuk. Temporally consistent object-centric learning by contrasting slots. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 5401–5411, 2025.

Takeru Miyato, Sindy Löwe, Andreas Geiger, and Max Welling. Artificial kuramoto oscillatory neurons. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=nwDRD4AMoN.

Malte Mosbach, Jan Niklas Ewertz, Angel Villar-Corrales, and Sven Behnke. Sold: Reinforcement learning with slot object-centric latent dynamics. *arXiv preprint arXiv:2410.08822*, 2024.

Yosuke Nishimoto and Takashi Matsubara. Object-centric world models for causality-aware reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 24585–24593, 2026.

Yazhe Niu, Yuan Pu, Zhenjie Yang, Xueyan Li, Tong Zhou, Jiyuan Ren, Shuai Hu, Hongsheng Li, and Yu Liu. Lightzero: A unified benchmark for monte carlo tree search in general sequential decision scenarios. *Advances in Neural Information Processing Systems*, 36:37594–37635, 2023.

Yuan Pu, Yazhe Niu, Zhenjie Yang, Jiyuan Ren, Hongsheng Li, and Yu Liu. Unizero: Generalized and efficient planning with scalable latent world models. *Transactions on Machine Learning Research*, 2025.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollar, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Ha6RTeWMd0.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. *Advances in neural information processing systems*, 30, 2017.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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.

Richard M. Shiffrin, Danielle S. Bassett, Nikolaus Kriegeskorte, and Joshua B. Tenenbaum. The brain produces mind by modeling. *Proceedings of the National Academy of Sciences*, 117(47): 29299–29301, 2020. doi: 10.1073/pnas.1912340117. URL https://www.pnas.org/doi/abs/10.1073/pnas.1912340117.

Gautam Singh, Fei Deng, and Sungjin Ahn. Illiterate DALL-e learns to compose. In *International Conference on Learning Representations*, 2022a. URL https://openreview.net/forum?id=h0OYV0We3oh.

Gautam Singh, Yi-Fu Wu, and Sungjin Ahn. Simple unsupervised object-centric learning for complex and naturalistic videos. *Advances in Neural Information Processing Systems*, 35:18181–18196, 2022b.

Elizabeth S Spelke and Katherine D Kinzler. Core knowledge. *Dev. Sci.*, 10(1):89–96, January 2007.

Aleksandar Stanić, Yujin Tang, David Ha, and Jürgen Schmidhuber. Learning to generalize with object-centric agents in the open world survival game crafter. *IEEE Transactions on Games*, 2023.

Leonid Ugadiarov, Vitaliy Vorobyov, and Aleksandr Panov. Object-centric dreamer. In *International Conference on Artificial Neural Networks*, pp. 153–165. Springer, 2025.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Shengjie Wang, Shaohuai Liu, Weirui Ye, Jiacheng You, and Yang Gao. Efficientzero v2: Mastering discrete and continuous control with limited data. In *International Conference on Machine Learning*, pp. 51041–51062. PMLR, 2024.

Nicholas Watters, Loic Matthey, Matko Bosnjak, Christopher P Burgess, and Alexander Lerchner. Cobra: Data-efficient model-based rl through unsupervised object discovery and curiosity-driven exploration. *arXiv preprint arXiv:1905.09275*, 2019.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. *Advances in neural information processing systems*, 34:25476–25488, 2021.

Jaesik Yoon, Yi-Fu Wu, Heechul Bae, and Sungjin Ahn. An investigation into pre-training object-centric representations for reinforcement learning. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 40147–40174, 2023.

Andrii Zadaianchuk, Maximilian Seitzer, and Georg Martius. Object-centric learning for real-world videos by predicting temporal feature similarities. *Advances in neural information processing systems*, 36:61514–61545, 2023.

Weipu Zhang, Adam Jelley, Trevor McInroe, and Amos Storkey. Objects matter: object-centric world models improve reinforcement learning in visually complex environments. *arXiv preprint arXiv:2501.16443*, 2025.

A MEAN NORMALIZED SCORE

Table 2: Normalization parameters. Notation is consistent with Equation (6).

Task	Metric, m	Normalization Value, $\max_{m',t'} F(m', e; t')$	Timestep Budget, $\mathcal{T}_e$
Object Goal	Success Rate	1	500 k
Object Interaction	Success Rate	1	2 M
Object Comparison	Success Rate	1	500 k
Property Comparison	Success Rate	1	500 k
Object Reaching	Success Rate	1	500 k
Block Lifting	Cumulative Reward	104.6	500 k
Cube Pushing	Success Rate	1	500 k
Defend The Line	Cumulative Reward	33.6	1 M

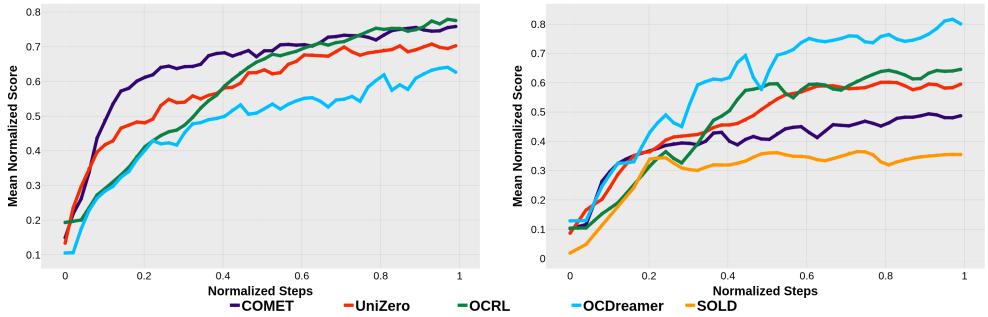

Figure 6: Curves for the mean normalized score (6). Normalization parameters are listed in Table 2. **Left:** the normalized score, averaged over all considered tasks — Object Goal, Object Interaction, Object Comparison, Property Comparison, Object Reaching, Block Lifting, Cube Pushing, and Defend The Line — for all algorithms except SOLD. **Right:** the normalized score, averaged over the continuous-control tasks compatible with SOLD — Object Reaching, Block Lifting, and Cube Pushing.

Table 3: Mean normalized score (6) versus normalized steps. Normalization parameters are listed in Table 2. The normalized score is averaged over the continuous-control tasks compatible with SOLD — Object Reaching, Block Lifting, and Cube Pushing.

Normalized step	COMET (ours)	OC-Dreamer	OCRL	SOLD	UniZero
0.33	0.399	**0.601**	0.368	0.301	0.421
0.66	0.462	**0.747**	0.597	0.739	0.599
1.00	0.499	**0.798**	0.651	0.352	0.601

B ABLATION STUDY

We conduct an ablation study on the contributions of the action-slot binding mechanism and slot pre-initialization to the performance of COMET. Figure 7 shows that COMET outperforms its variant without action-slot binding. The learning curves in Figure 8 show that, in the absence of slot pre-initialization, COMET's learning progress slows significantly. These experiments demonstrate that action-slot binding and slot pre-initialization are key components of the proposed method.

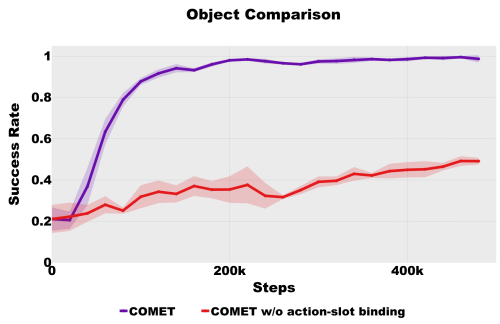

Figure 7: Ablation of action-slot binding on the Object Comparison task. COMET shows slow progress without action-slot binding, indicating that this component is key to the proposed method. Success rate is averaged over 30 episodes and three seeds; shaded areas indicate the standard deviation. Exponential smoothing with a coefficient of 0.5 is applied.

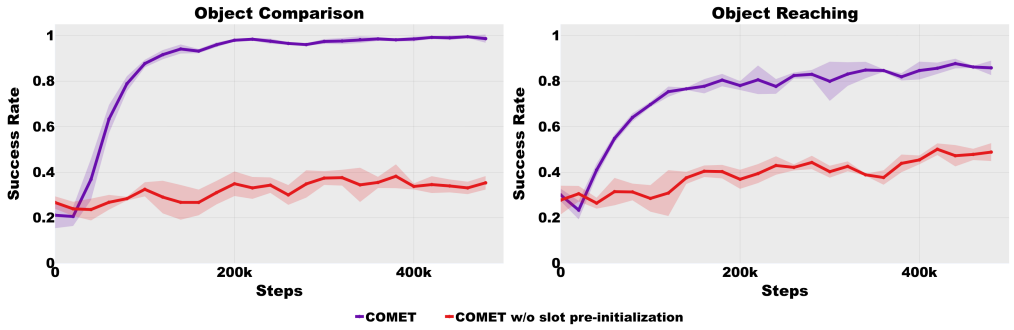

Figure 8: Ablation of slot pre-initialization on the Object Comparison and Object Reaching tasks. COMET shows slow or no progress without it, indicating that this component is key to the proposed method. Success rate is averaged over 30 episodes and three seeds; shaded areas indicate the standard deviation. Exponential smoothing with a coefficient of 0.5 is applied.

C Online Fintetuning of Object-Centric Encoder

Distribution mismatch between data collected by a uniform random policy and data seen by a close-to-optimal policy is a valid limitation of the frozen-encoder design. We run adaptation experiments in which we fine-tune the object-centric encoder along with the world model on replay-buffer data, on two modified tasks with distribution shift:

- Object Comparison task with a new shape (5-pointed star) and new color (cyan).
- Object Reaching where blue/green cubes are replaced with cyan/orange cubes.

The results of the experiments are demonstrated on the Figure 9. Online training of the object-centric encoder recovers most of the lost performance under the shift between the original and modified tasks, because the encoder adapts to unseen data. These experiments show that the object-centric encoder in COMET can be fine-tuned along with the world model as the distribution of observations shifts, while the quality of the policy improves.

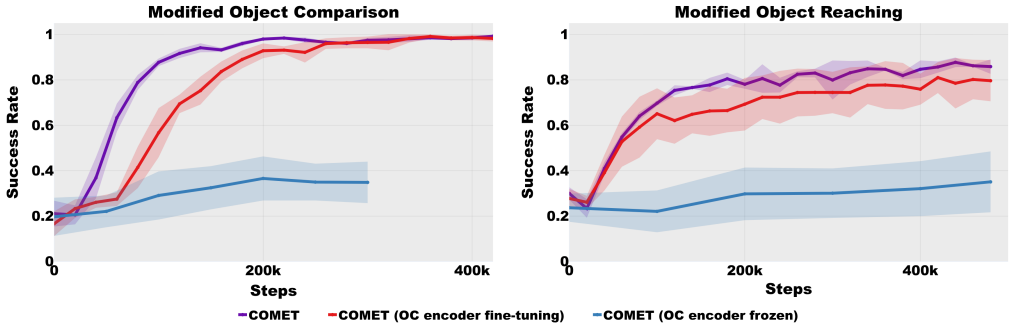

Figure 9: Experiments with frozen / finetuned object-centric encoder on the modified tasks. COMET uses an OC encoder frozen after pre-training on the modified task. COMET (OC encoder frozen) instead uses an encoder pre-trained on the original task and kept frozen. COMET (OC encoder fine-tuning) starts from that same original-task encoder but fine-tunes it online. COMET (OC encoder frozen) shows no significant progress, since its encoder is pre-trained on the original task, which lacks some objects present in the modified task. COMET (OC encoder fine-tuning) learns more slowly than COMET due to online encoder fine-tuning, but still solves the task successfully. Success rate is averaged over 30 episodes and three seeds; shaded areas indicate the standard deviation. Exponential smoothing with a coefficient of 0.5 is applied.

D DATASETS

We collect images to train object-centric representation models using a uniform random policy.

Table 4: Parameters of the collected datasets.

Task	Dataset Size	Source Image Resolution
Block Lifting	300 k	256×256
Cube Pushing	300 k	224×224
Defend The Line	500 k	336×336

E SLATE'S HYPERPARAMETERS

For Object Goal, Object Interaction, Object Comparison, Property Comparison, and Object Reaching tasks we used pretrained SLATE models from the OCRL official repository `https://github.com/jsikyoon/OCRL`.

Table 5: Hyperparameters for SLATE

Learning	Training dataset size	1000000
	Temp. cooldown	1.0 to 0.1
	Temp. cooldown steps	30000
	LR for DVAE	0.0003
	LR for CNN Encoder	0.0001
	LR for Transformer Decoder	0.0003
	LR warm-up steps	30000
	LR half time	250000
	Dropout	0.1
	Clip	0.05
	Batch size	32
	Epochs	100
DVAE	Vocabulary size	4096
CNN Encoder	Hidden size	64
Slot Attention	Iterations	3
	Slot heads	1
	Slot dim.	192
	MLP hidden dim.	192
Transformer Decoder	Layers	4
	Heads	4
	Hidden dim.	192

F DINOSAUR'S HYPERPARAMETERS

Table 6: Hyperparameters for DINOSAUR

Learning	Training dataset size	300000
	Training steps	500000
	Batch size	64
	LR warm-up steps	10000
	Peak LR	0.0004
	Exp. decay half-life	100000
	ViT Architecture	ViT-B
	Feature dim.	768
	Patch size	8
	Gradient norm clipping	1.0
	Image/Crop size	224
	Cropping strategy	Full
	Tokens	784
Decoder	Type	MLP
	Layers	4
	MLP hidden dim.	1024
Slot Attention	Iterations	3
	Slots	5
	Slot dim.	64
	MLP hidden dim.	512

G SLOT CONTRAST'S HYPERPARAMETERS

Table 7: Hyperparameters for Slot Contrast

Learning	Training steps	100000
	Batch size	64
	Training segment length	4
	LR warm-up steps	2500
	Optimizer	Adam
	Peak LR	0.0004
	Exp. decay half-life	100000
	ViT Architecture	DINOv2 Small
	Initialization	FixedLearnedInit
	Patch size	14
	Feature dim.	384
	Gradient norm clipping	0.05
	Image/Crop size	336
	Cropping strategy	Full
	Image tokens	576
Predictor	Type	Transformer
	Layers	1
	Heads	4
Decoder	Type	MLP
Slot Attention	Iterations (first / other frames)	3 / 2
	Slot dim.	64
Loss Parameters	Slot-slot contrastive loss	disabled

H COMET'S HYPERPARAMETERS

Table 8: Hyperparameters for COMET

Planning	Number of MCTS Simulations (sim)	50
	Number of Sampled Actions (K)	20 (Continuous tasks only)
	Inference Context Length	10
	Temperature	0.25
	Dirichlet Noise (α)	0.3
	Dirichlet Noise Weight	0.25
	Coefficient c_1	1.25
	Coefficient c_2	19652
Environment and Replay Buffer	Replay Buffer Capacity	1,000,000
	Sampling Strategy	Uniform
	Reward Clipping	True (Discrete only)
	Data Augmentation	False
	Game Segment Length	400 (Discrete); 100 (Continuous)
Architecture	Number of Backbone Transformer Heads	8 (Discrete); 4 (Continuous)
	Number of Backbone Transformer Layers (N)	2
	Number of Policy/Value Transformer Heads	4
	Number of Policy/Value Transformer Layers (N)	1
	Dropout Rate (p)	0.1
	Activation Function	GELU
	Reward/Value Bins	101 (Continuous); 601 (Discrete)
Optimization	Training Context Length (H)	10
	Replay Ratio	0.25
	Buffer Reanalyze Frequency	0 (Discrete); $1/100000$ (Continuous)
	Batch Size	64
	Optimizer	AdamW
	Learning Rate	1×10^{-4}
	Next Latent State Loss Coefficient	10
	Reward Loss Coefficient	1 (Discrete); 0.1 (Continuous)
	Policy Loss Coefficient	1 (Discrete); 0.1 (Continuous)
	Value Loss Coefficient	0.5 (Discrete); 0.1 (Continuous)
	Policy Entropy Coefficient	1×10^{-4}
	Weight Decay	10^{-4}
	Max Gradient Norm	5
	Discount Factor	0.997 (0.925 in Cube Pushing Task)
	Soft Target Update Momentum	0.05
	Hard Target Network Update Frequency	100
	Temporal Difference (TD) Steps	5

I COMPUTE RESOURCES

In our setup, training COMET for 500k environment steps on a single NVIDIA H100 (80 GB) GPU takes approximately 18 hours on average across different tasks.

J COMET'S TRAJECTORY ROLLOUTS

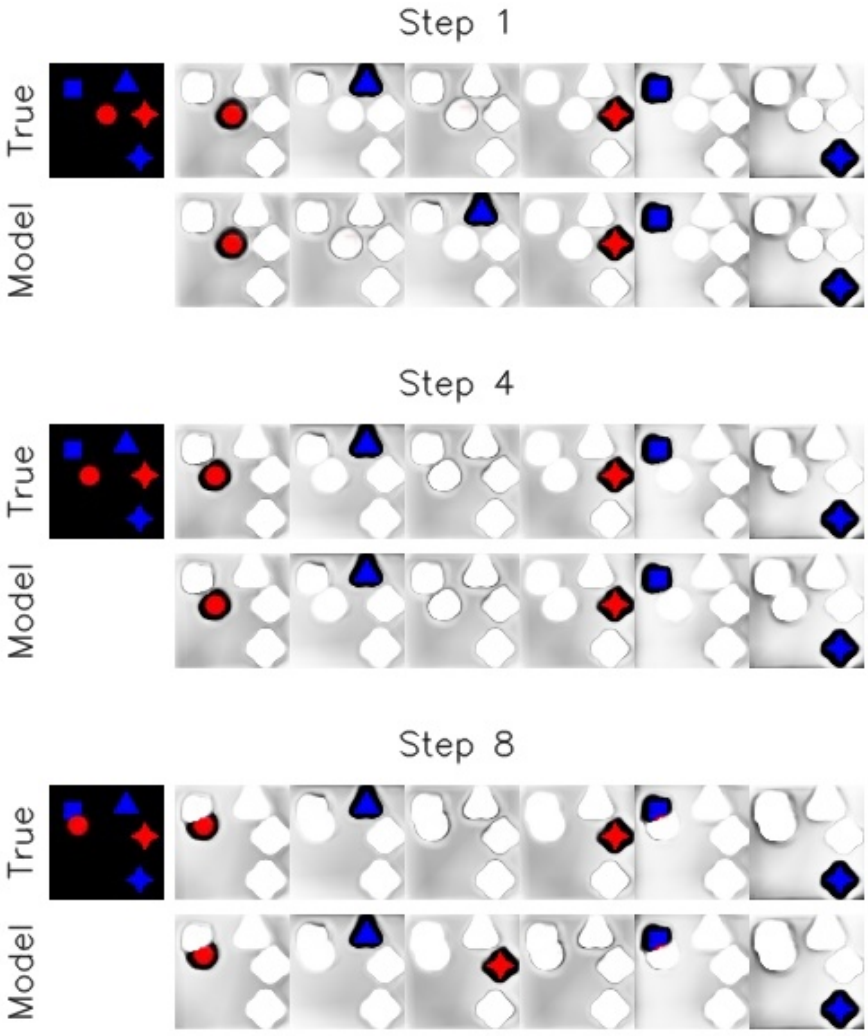

Figure 10: Trajectory rollout generated using COMET's policy for the Object Goal Task. The first row at each time step shows the real observation from the environment along with attention maps produced by the SLATE model over each slot inferred by the SLATE model. The second row shows attention maps produced by the SLATE model for each slot predicted by COMET's dynamics model. COMET's dynamics model correctly predicted object representations.



Figure 11: Trajectory rollout generated using COMET's policy for the Block Lifting Task. The first row at each time step shows the real observation from the environment along with attention maps produced by the DINOSAUR model over each slot inferred by the DINOSAUR model. The second row shows attention maps produced by the DINOSAUR model for each slot predicted by COMET's dynamics mode. COMET's dynamics model correctly predicted object representations. COMET's dynamics model correctly predicted object representations.

K COMET'S CAUSALITY-INSPIRED ATTENTION PROBABILITIES IN POLICY AND VALUE MODELS

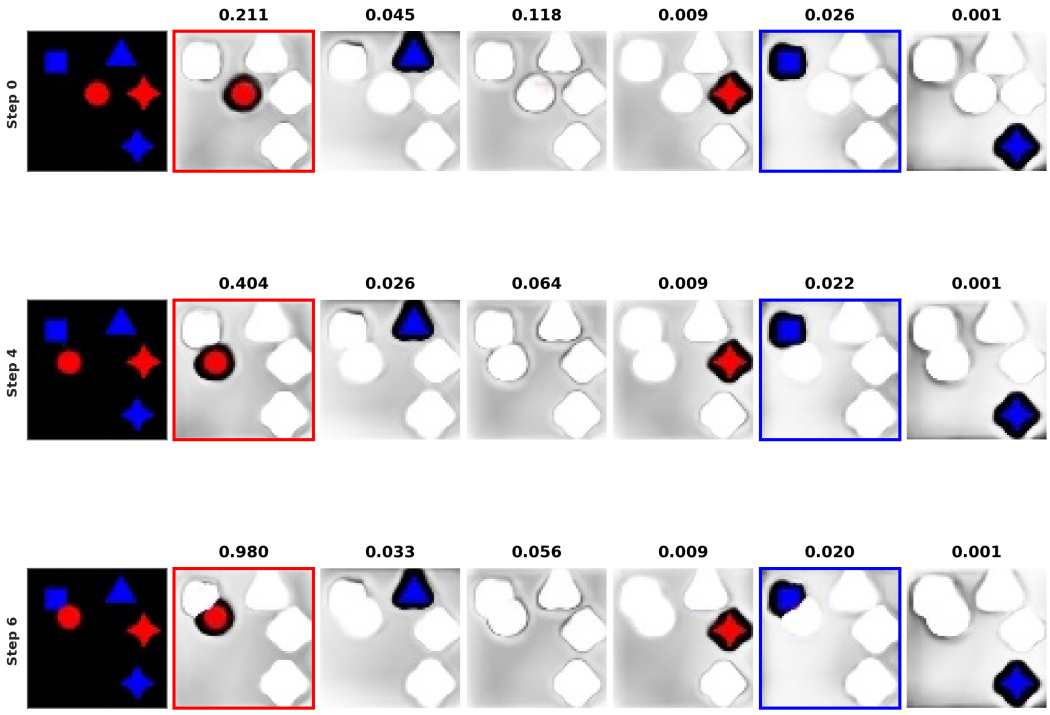

Figure 12: Per-slot causality scores for the value transformer in the Object Goal Task. Each row corresponds to a time step and shows the real observation from the environment together with attention maps produced by the SLATE model for each slot inferred by the SLATE model. The number above each slot denotes its causality score α_t^i, indicating the probability that the corresponding object is causally relevant for value prediction. Red bounding boxes indicate agent-related objects, blue bounding boxes indicate target object. The causality score is highest for the agent object across all time steps.

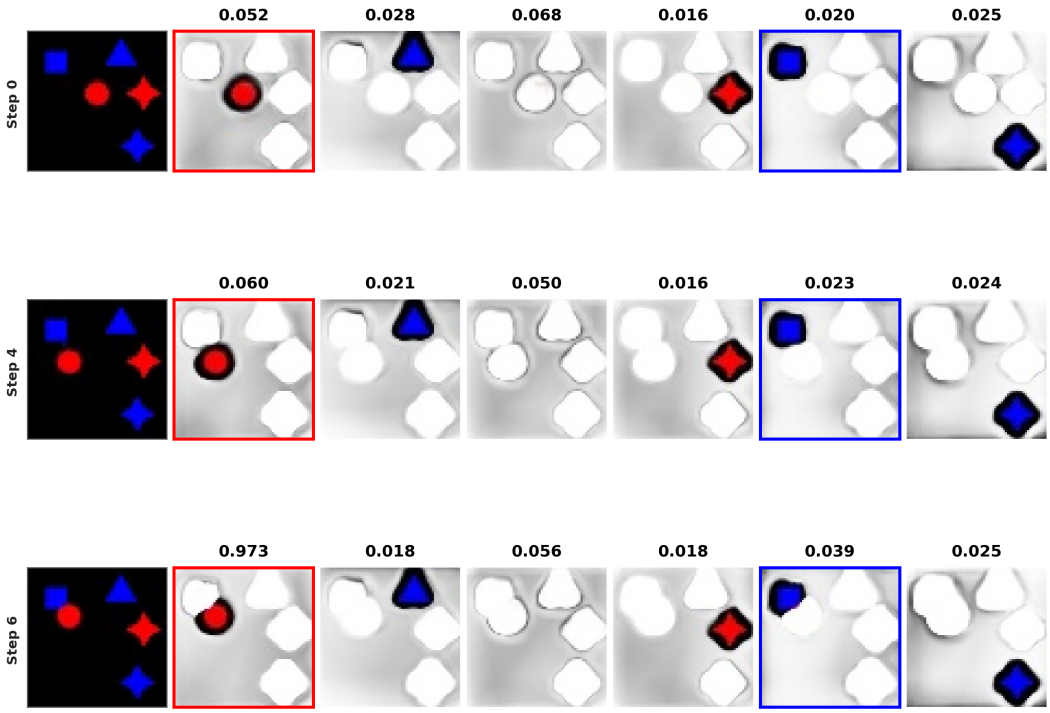

Figure 13: Per-slot causality scores for the policy transformer in the Object Goal Task. Each row corresponds to a time step and shows the real observation from the environment together with attention maps produced by the SLATE model for each slot inferred by the SLATE model. The number above each slot denotes its causality score α_t^i, indicating the probability that the corresponding object is causally relevant for policy prediction. Red bounding boxes indicate agent-related objects, blue bounding boxes indicate target object. The causality score is highest for the agent object across most time steps.

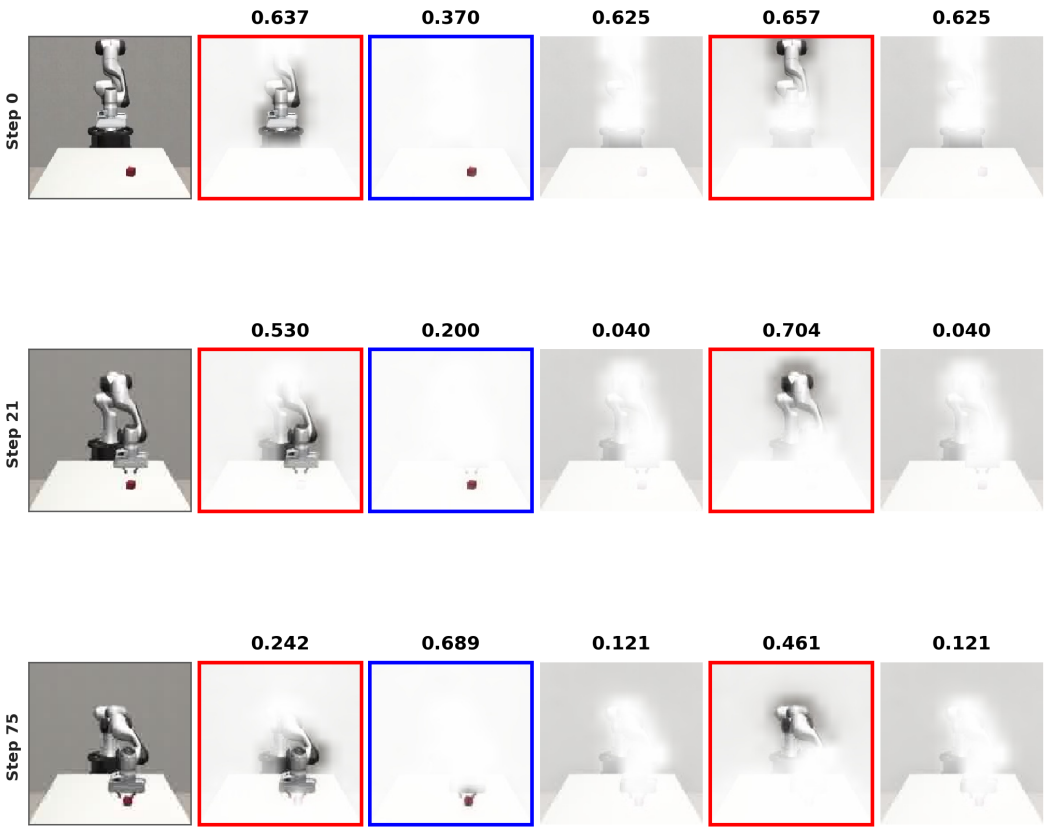

Figure 14: Per-slot causality scores for the value transformer in the Block Lifting Task. Each row corresponds to a time step and shows the real observation from the environment together with attention maps produced by the DINOSAUR model for each slot inferred by the DINOSAUR model. The number above each slot denotes its causality score α_t^i, indicating the probability that the corresponding object is causally relevant for value prediction. Blue bounding boxes indicate target object. The causality score is highest for the target object and and for agent's objects distributed across multiple slots. The causality score is highest for the target and agent objects across most time steps.

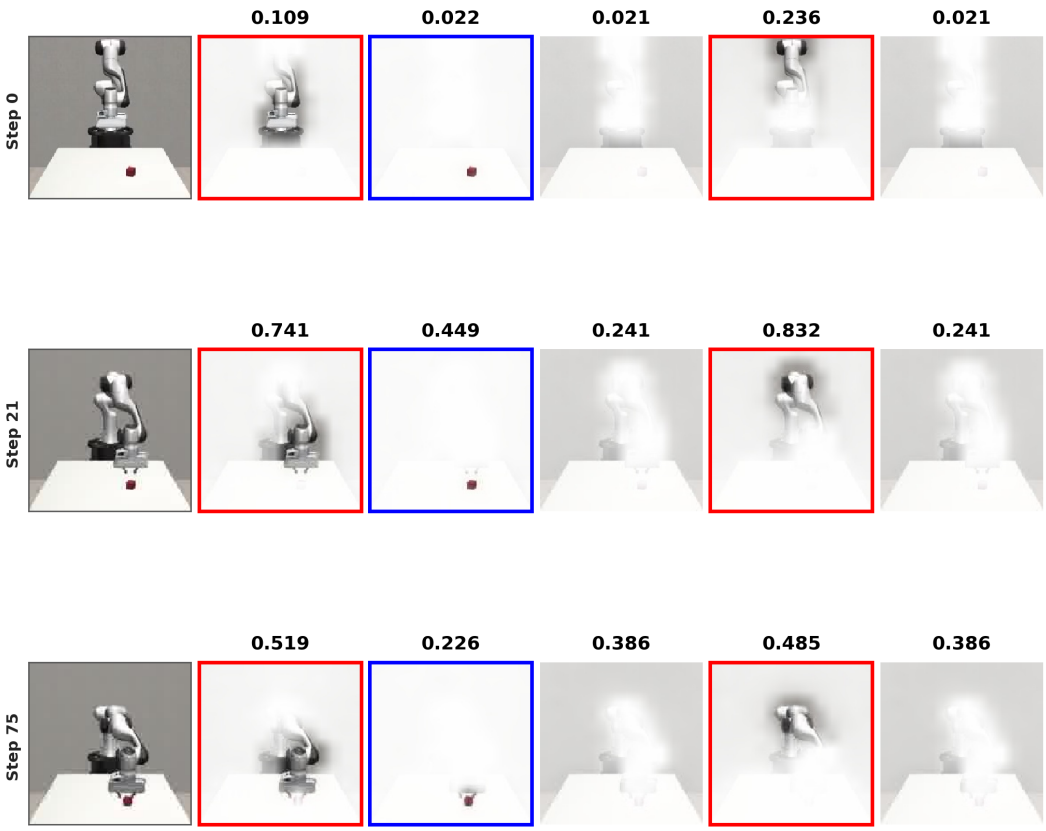

Figure 15: Per-slot causality scores for the policy transformer in the Block Lifting Task. Each row corresponds to a time step and shows the real observation from the environment together with attention maps produced by the DINOSAUR model for each slot inferred by the DINOSAUR model. The number above each slot denotes its causality score α_t^i, indicating the probability that the corresponding object is causally relevant for policy prediction. Red bounding boxes indicate agent-related objects, blue bounding boxes indicate target objects. The causality score is highest for the target and agent objects across most time steps.

L SAVI'S VISUALIZATIONS FOR SOLD

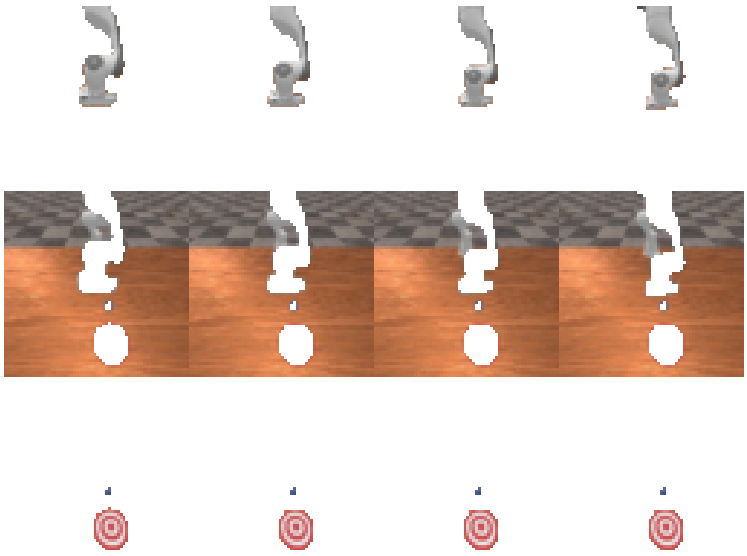

Figure 16: Hard attention maps produced by the SAVI model in the Cube Pushing task.

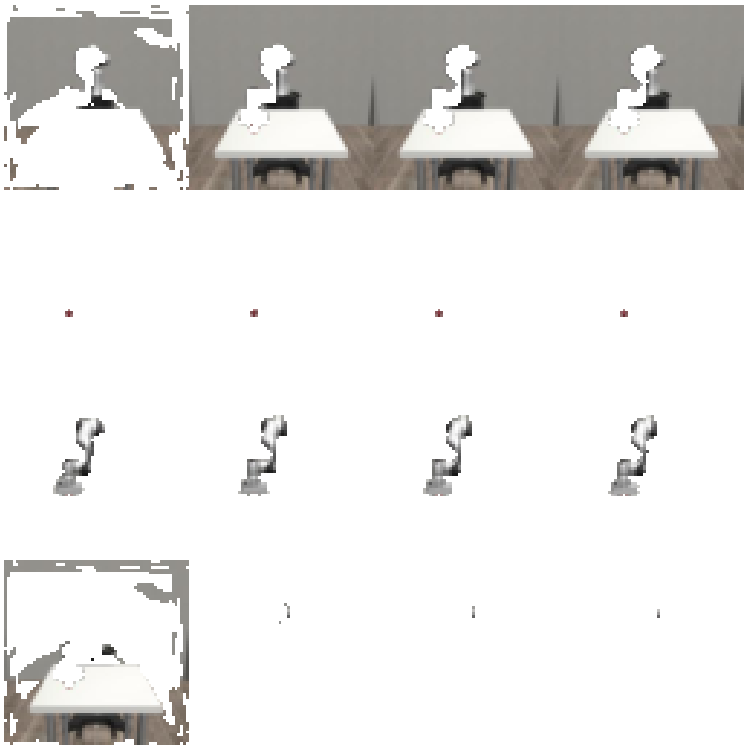

Figure 17: Hard attention maps produced by the SAVI model in the Block Lifting task.

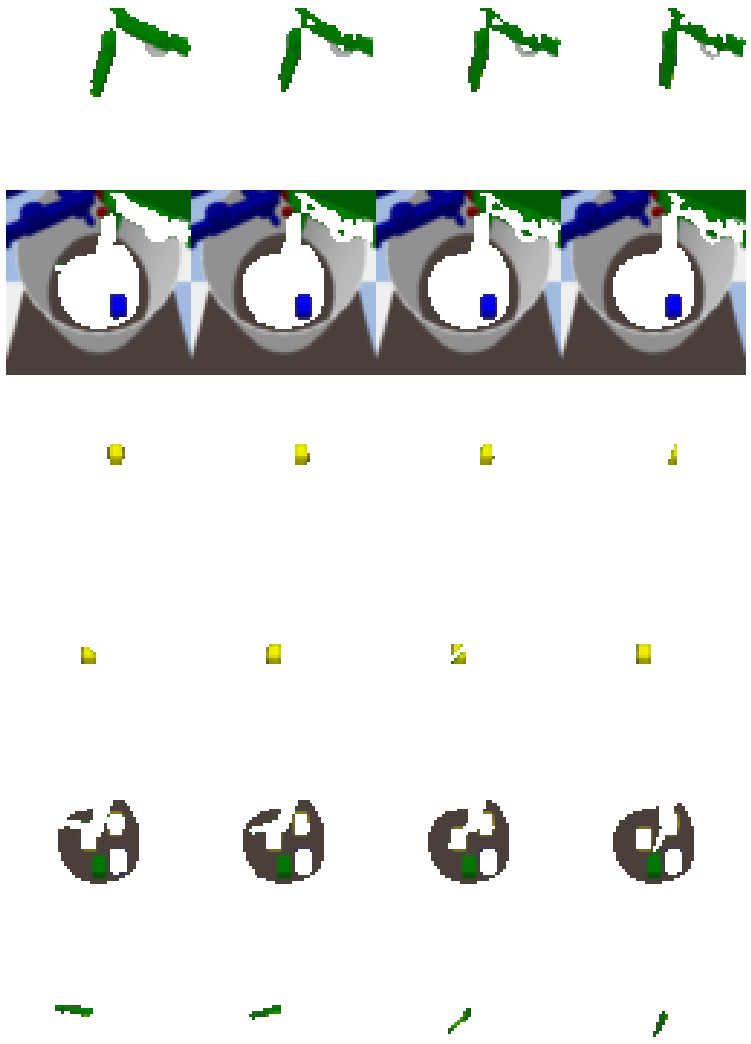

Figure 18: Hard attention maps produced by the SAVI model in the Object Reaching task.