Discovering Predictive Relational Object Symbols With Symbolic Attentive Layers

Alper Ahmetoglu, *Member, IEEE*, Batuhan Celik, Erhan Oztop, *Member, IEEE*, and Emre Ugur

***Abstract*—In this letter, we propose and realize a new deep learning architecture for discovering symbolic representations for objects and their relations based on the self-supervised continuous interaction of a manipulator robot with multiple objects in a tabletop environment. The key feature of the model is that it can take a changing number of objects as input and map the object-object relations into symbolic domain explicitly. In the model, we employ a self-attention layer that computes discrete attention weights from object features, which are treated as relational symbols between objects. These relational symbols are then used to aggregate the learned object symbols and predict the effects of executed actions on each object. The result is a pipeline that allows the formation of object symbols and relational symbols from a dataset of object features, actions, and effects in an end-to-end manner. We compare the performance of our proposed architecture with state-of-the-art symbol discovery methods in a simulated tabletop environment where the robot needs to discover symbols related to the relative positions of objects to predict the action's result. Our experiments show that the proposed architecture performs better than other baselines in effect prediction while forming not only object symbols but also relational symbols.**

***Index Terms*—Developmental robotics, learning categories and concepts, deep learning methods.**

I. Introduction

LEARNING the symbolic representation of tasks enables the application of classical AI search techniques to find a solution in the symbolic definition of the task. For well-defined environments, symbolic systems can be manually designed to describe robot-environment interactions. However, such manual designs would only be scalable to a handful of domains and require significant work to adapt to new environments. On the other hand, learning the required symbols for the task from data would be a more scalable and generalizable strategy to achieve truly intelligent robots [1]. Therefore, there is a considerable amount of research on how to convert the sensorimotor experience of a robotic agent into symbolic representations [2].

One prominent strategy for learning the necessary symbols is to partition the precondition and the effect set of the agent's actions and learn classifiers for these partitions [3], [4], [5]. This ensures that the learned symbols are compatible with the actions available to the agent and filters out irrelevant aspects of the environment that the agent cannot manipulate. Learning symbols can also be formalized as compressing the state-space into symbolic state-space with autoencoders [6], [7], [8]. Operators, which are high-level actions manipulating these symbols, can be learned simultaneously or separately. One of the main advantages of this approach is that symbols are learned with deep neural networks, which opens up the possibility of integrating other deep architectures to the learning pipeline, such as convolutional layers to process images, to improve the quality of the learned representations. However, the advantage of deep architectures is usually conditional on the amount of data available, which is especially critical in robotics applications. Therefore, it is important to use deep architectures only for parts of a system that can benefit from them.

Our previous work, DeepSym [9], combines these two motivations: learning preconditions and effects of actions with deep neural networks. In DeepSym, an encoder-decoder network with a discrete bottleneck layer is trained to predict the effect of actions (Fig. 1 – bottom left). However, the network can only handle a fixed number of object interactions, restricting the types of relations that can be learned. This restriction is lifted in a more recent work [10] by introducing a self-attention mechanism [11] to the architecture (Fig. 1 – bottom right). As symbols interact with each other using self-attention, the network can make accurate predictions for related objects (e.g., on top of each other). Although this architecture is effective in making accurate predictions for related objects through the learned multi-object symbols, it does not reveal the explicit relations between objects. Furthermore, as the self-attention layer is applied after discretization, the relational representational capacity of the model is limited by the learned symbols.

In the current work, we explicitly compute discrete self-attention weights from object features and treat them as relational symbols between objects. Using these discrete relations, we fuse object symbols in an aggregation function to produce a single representation for each object, which is then used to

Manuscript received 31 August 2023; accepted 20 December 2023. Date of publication 8 January 2024; date of current version 18 January 2024. This letter was recommended for publication by Associate Editor T. Horii and Editor T. Ogata upon evaluation of the reviewers' comments. This work was supported in part by the TUBITAK The Scientific and Technological Research Council of TurkeyARDEB 1001 Program120E274, in part by Grant-in-Aid for Scientific Research22H03670, in part by the Project JPNP16007 Commissioned by the New Energy and Industrial Technology Development Organization (NEDO), in part by JST, CREST under Grant JPMJCR17A4, and in part by INVERSE Project under Grant 101136067, funded by the European Union. *(Corresponding author: Alper Ahmetoglu.)*

Alper Ahmetoglu, Batuhan Celik, and Emre Ugur are with the Department of Computer Engineering, Bogazici University, Istanbul 34342, Türkiye (e-mail: alper.ahmetoglu@boun.edu.tr; batuhancelik.boun@gmail.com; emreugur@gmail.com).

Erhan Oztop is with the Department of Computer Science, Ozyegin University, Istanbul 34794, Türkiye, and also with OTRI, SISReC, Osaka University, Osaka 565-0871, Japan (e-mail: erhan.oztop@otri.osaka-u.ac.jp).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2024.3350994, provided by the authors.

Digital Object Identifier 10.1109/LRA.2024.3350994

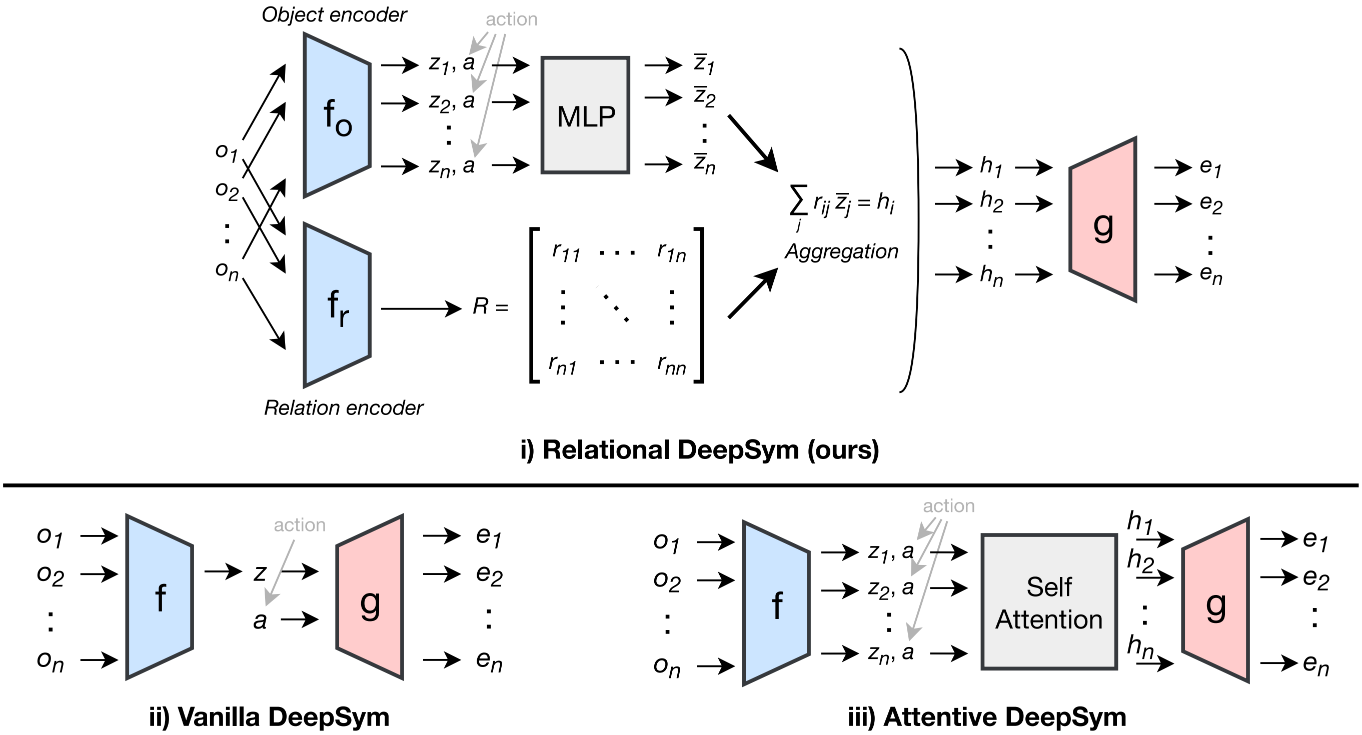

Fig. 1. Proposed model is shown in the top panel. The object and the relational encoders take object features as input and process them in parallel. The object encoder outputs an object symbol z_i for the object o_i, and relational encoder outputs the query vector q_i and the key vector k_i which are used as in (3) to calculate relational symbols. For comparison, we also provide high-level outlines of [9] and [10] in the bottom panel in (ii) and (iii), respectively.

predict the observed, potentially multi-object, effect. This results in a more powerful architecture, which we named Relational DeepSym, that can explicitly output the relations between a varying number of objects while enjoying other properties of DeepSym. Our experiments in a simulated tabletop scenario show that (1) Relational DeepSym achieves lower errors than [9] and [10] for different numbers of objects and actions, (2) learns not only object symbols but also relational symbols.

II. Related Work

Early symbol grounding studies in robotics (e.g., [12], [13]) assumed the existence of manually defined symbols that were effective in plan generation. These studies collected data from interactions of agents and robots and learned sensor-to-symbol mappings to ground the pre-defined symbols in the sensorimotor experience of the robot. In these studies, transition rules, which are connected by symbolic preconditions and effects, were defined, and the continuous experience of the robot was used to map the manually defined symbolic predicates to the continuous perceptual space of the robots. Recently, [14] proposed a deep neural network architecture based on Convolutional Variational Auto-Encoders to discover visual features that are well-suit for pre-defined recognition and interaction tasks. [15] used Multimodal Latent Dirichlet Allocation (MLDA) to learn the mapping between multi-modal sensory experience and preconditions and post-conditions of actions of a robot. We argue that pre-defining symbols in unknown and changing environments is not possible, and as stated by [16], symbols should instead "be formed in relation to the experience of agents, through their perceptual/motor apparatuses, in their world and linked to their goals and actions".

Unsupervised discovery of discrete symbols and rule learning from the continuous sensorimotor experience of embodied agents has been recently studied in robotics in order to equip robots with advanced reasoning and planning capabilities [1], [2]. [17] investigated the discovery of sub-symbolic neural activations that facilitate resource economy and fast learning in skill transfer but did not address high-level reasoning with discrete symbols. [3], [18] discovered symbols that were directly used as predicates in precondition and post-condition fields of action descriptors, represented in Problem Domain Definition Language (PDDL). This encoding allowed for making deterministic and probabilistic plans in 2-dimensional agent environments. The same architecture was extended to a real-world robotic environment in [4], where symbols representing absolute global states were learned and used for planning. [19], on the other hand, learned egocentric symbolic representations that enabled the agents to transfer the previously learned symbols to novel environments directly. [20] considers learning symbols from object-centric observations, allowing for a transfer between tasks that share the same types of objects. Effect clustering techniques and SVM classifiers were used to discretize the continuous sensorimotor experience of the agents in these works.

Whereas the previous work addressed learning symbols from given skills, [21], [22], [23], [24], [25] learned a set of skills from a set of symbolic predicates and a collection of demonstrations. [26] considered only the necessary changes in the predicate set for more compact operators. In follow-up work, [27] used a surrogate objective for learning state abstractions that increase planning performance. [5], [28] discovered discrete symbols and used these symbols in order to generate PDDL rules for planning by again combining effect clustering techniques to find discrete effect categories and SVM classifiers to discretize continuous object feature space. These studies used ad-hoc combinations of several machine learning methods. On the other hand, [9] provided a more generic symbol formation engine, which used a novel deep network architecture that runs at the pixel level and relies on purely predictive mechanisms in forming symbols instead of unsupervised clustering techniques. They used an effect predictor encoder-decoder network that took the object image and action as input and exploited a binary bottleneck layer to automatically form object categories. Similar to this work, [6], [7], [8] also exploited deep neural networks with binary bottleneck units to find discrete state and effect symbols and achieve plan generation using these symbols. [29], [30] model multi-object dynamics using graph neural networks and predict object relations with a recurrent architecture. Similarly, [31] learns a latent state-space in a graph structure and a transition function that can predict object relations between objects after an action execution given the graph.

Our work differs from previous research as we propose a new method for learning symbolic representations of objects and relations between them in a unified architecture. Most related to [10] that also uses self-attention to model relational information, our model differs in that it explicitly outputs the relations between objects. In contrast, in [10], the learned relations are opaque to the user.

III. Methods

The problem definition and our assumptions are given in Section III-A, the proposed model is explained in Section III-B, and the differences with previous DeepSym architectures are discussed in Section III-C.

A. Problem Definition

This work deals with the problem of learning symbolic representations of objects and relations between them from continuous state representations collected by a robot to predict the effect of its actions. From a developmental learning perspective, this study starts off with a basic sensorimotor system [32], [33], where the robot can locate objects, and pick and place them on top of each other.

We consider an environment represented by a set of object features $O = \{o_1, \ldots, o_n\}$ where $o_i \in \mathbb{R}^{d_o}$ is a d_o-dimensional continuous-valued vector for the ith object's features and n is the number of objects which can vary through multiple environment instances. Without any loss of generality, we define object features as the combination of (1) the object type (e.g., short block, long block), (2) the position (x, y, z) and the orientation (x, y, z, w) of the object, resulting in a total of 8 dimensions for each object. However, the method can be applied to any other modality type (e.g., images, point clouds) by modifying the networks accordingly as long as the environment state can be partitioned into a set of objects.

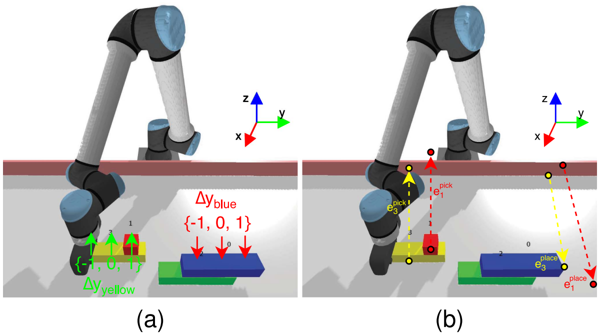

Fig. 2. Example interaction with the environment. Two objects are selected as pick and place targets. Green and red arrows show possible grasp and release locations, respectively. (a) $\texttt{pick-place}(o_3, -1, o_0, 0)$. (b) An example effect.

In our experiments, the robot has a single type of high-level action with different parameterizations: $\texttt{pick-place}(o_i, \Delta y_i, o_j, \Delta y_j)$ where o_i and o_j are the object to be picked up and the target object, respectively, and Δy_i and Δy_j are the y-axis pick and release positions relative to the center of the object (Fig. 2(a)). Δy_i and Δy_j can take discrete values of $\{-1, 0, 1\}$ which correspond to 7.5 cm left, center, and 7.5 cm right of the object center, respectively. This results in a total of $9n^2$ grounded actions (i.e., actions with parameters) for n objects. The robot randomly picks a grounded action, executes it in the environment, and observes the new environment state as $O' = \{o'_1, \ldots, o'_n\}$ where $o'_i \in \mathbb{R}^{d_o}$ is the new object features for the ith object.

The goal is to transform the state vector $O = \{o_1, \ldots, o_n\}$ where each $o_i \in \mathbb{R}^{d_o}$ is a feature vector describing object i into a set of object symbols $Z = \{z_1, \ldots, z_n\}$ and relational symbols $R_k = \{r_{11}^{(k)}, \ldots, r_{nn}^{(k)}\}$ where $z_i \in \{0, 1\}^{d_z}$ is a d_z-dimensional binary vector (i.e., an object symbol) for the ith object and $r_{ij}^{(k)} \in \{0, 1\}$ is a binary value for the kth relation between the ith and jth objects. Once we have a symbolic representation Z, R of a given state O, we can transform the continuously represented interaction data $\{O^{(i)}, a^{(i)}, O'^{(i)}\}_{i=1}^{N}$ into its symbolic counterpart, $\{(Z^{(i)}, R^{(i)}), a^{(i)}, (Z'^{(i)}, R'^{(i)})\}_{i=1}^{N}$, and learn a set of symbolic transition rules $(Z, R) \xrightarrow{a} (Z', R')$ enabling domain-independent planning with AI planners to achieve a goal state [3], [5], [6], [9], [21].

To learn object symbols and relations between objects, we follow the objective in [5], [9], [10] and train a model to predict the effect $E = \{e_1, \ldots, e_n\}$ of the executed action a where $e_i = \delta(o'_i, o_i)$ is defined as the cartesian position difference between o'_i and o_i before and after the execution of the action a. In our experiments, we compare the effect prediction performance of the proposed model with two related models [9], [10] in a simulated tabletop environment.

B. Relational DeepSym

The top panel in Fig. 1 shows a high-level overview of the proposed model. The model consists of four main components:

(1) an object encoder f_o that learns object symbols, (2) a relational encoder f_r that learns relational symbols between objects (3) an aggregation function that combines information from multiple objects by multiplying object symbols with relational symbols, and (4) a decoder g that predicts effect e_i of the executed action a for each object i. As the whole architecture is differentiable and trained in an end-to-end fashion to minimize the effect prediction error, we expect the object and the relational encoders to learn to predict symbols and relations useful for the decoder to predict the effect.

The object encoder f_o outputs a binary vector z_i for the ith object given its features o_i:

$$z_i = f_o(o_i) \tag{1}$$

To output a discrete vector without removing the differentiability, the activation of the last layer is set to the Gumbel-sigmoid function [34], [35]. The Gumbel-sigmoid function approximates a Bernoulli distribution by injecting noise drawn from Gumbel distribution to the logits, which forces the model to output in the extremities (i.e., either very low or very high values) to send a signal in the presence of noise. In our experiments, f_o is a multi-layer perceptron; however, other differentiable architectures can be used for different modalities (e.g., convolutional layers to process images)

The relational encoder takes object features $\{o_1, o_2, \ldots, o_n\}$ as input and processes them independently to output query and key vectors $\{(q_1, k_1), (q_2, k_2), \ldots, (q_n, k_n)\}$ for each object. Let Q and K be $n \times d$ matrices, each row containing a query vector q_i and a key vector k_i, respectively. The attention weights R, which are relational symbols in our case, are computed as follows:

$$q_i, k_i = f_r(o_i) \quad \forall i \in \{1, 2, \ldots, n\} \tag{2}$$

$$R = \text{GumbelSigmoid}\left(\frac{QK^T}{\sqrt{d}}\right) \tag{3}$$

where d is the dimensionality of the query and key vectors. This is slightly different from the regular self-attention function [11] in which the softmax function is used instead of the Gumbel-sigmoid function. This modification creates two different behaviors: (1) the use of a sigmoid function instead of a softmax function allows multiple attention weights to different objects to be active at the same time (whereas in softmax, attentions compete with each other), and (2) the use of the Gumbel-sigmoid function discretizes the attention weights while preserving differentiability, allowing us to treat the weights as *relational symbols* between objects. As in the object encoder, the relational encoder is a multi-layer perceptron with two different outputs for the query and the key. Note that multiple heads $R_1, R_2, \ldots, R_k$ can be used to model different relations between objects.

In the third step, the aggregation function combines object symbols $\{z_1, z_2, \ldots, z_n\}$, relational symbols $\{R_1, R_2, \ldots, R_k\}$, and the executed action a to produce a single representation for each object. The aggregation function has the following steps:

$$a_i = [(1, \Delta y_i)_{o_i=\text{pick}}, (1, \Delta y_i)_{o_i=\text{place}}] \tag{4}$$

$$\bar{z}_i = \text{MLP}(z_i, a_i) \tag{5}$$

$$H^j = R_j \bar{Z} \tag{6}$$

$$H = (H^1, H^2, \ldots, H^k) \tag{7}$$

for $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, k\}$ where object symbols and the action vector are concatenated in (5), and the aggregation occurs in (6). To concatenate action-specific information to each object symbol z_i in a permutation-invariant way, we define a 4-dimensional vector a_i (4) for each object i in which the first and the third dimensions are set to 1 if object i is the picked or placed object, respectively. For example, consider the action in Fig. 2(a), $\texttt{pick-place}(o_3, -1, o_0, 0)$, which translates as "pick up o_3 from its left and place it on top of o_0". Here, a_3 and a_0 are set to $[1, -1, 0, 0]$ and $[0, 0, 1, 0]$, respectively, and a_1 and a_2 are set to zero vectors.

One can possibly aggregate the input multiple times by applying (6) more than once to model longer effect chains. In our experiments, we use a single aggregation step. Multiple combinations from multiple attention heads are concatenated in (7) to produce a single representation h_i for each object.

As the final step, the decoder takes the aggregated representation h_i as input and predicts the effect $\hat{e}$ for each object for the executed action a. The decoder is a multi-layer perceptron. The predicted effect is then compared with the ground truth effect e to compute the mean squared error:

$$\mathcal{L} = \frac{1}{M} \sum_{j=1}^{M} \sum_{i=1}^{N} (\hat{e}_i^{(j)} - e_i^{(j)})^2 \tag{8}$$

where M is the batch size, and N is the number of objects.

C. Comparison With Related Models

As in DeepSym (Fig. 1 – bottom left), this architecture is also an encoder-decoder architecture with discrete bottleneck layers. The difference is that the information between objects is shared in the aggregation function using the learned attention weights for a more accurate effect prediction for actions involving several objects. In DeepSym, this can only be achieved by fixing the number of input objects, whereas there is no such limitation in the proposed model.

Regarding the architecture in [10] (Fig. 1 – bottom right), the most significant difference is the placement of the self-attention module. In [10], the self-attention module takes object symbols (the encoder's output) as its input and directly outputs the aggregated representation. This restricts the model from learning attention weights only from the learned symbols. In this proposal, attention weights are learned from object features, making relations more general.

The second significant difference is the explicit use of attention weights. In [10], attention weights are used within the self-attention layer as in the original Transformer architecture [11]. However, since attention weights are continuous, they cannot be easily expressed as relational symbols between objects.

IV. Experiments

A. Experiment Setup

1) Environment: We created a tabletop object manipulation environment for our experiments (Fig. 2). The environment consists of a UR10 robot and two to four objects. These objects are either short blocks or long blocks with their physical properties (e.g., size, mass, the friction coefficient) fixed through the interaction phase. The robot has a single type of high-level action: grasping and releasing an object on top of or near another object. We assume that object positions can be recognized by a separate module, and the robot can track the cartesian position change of these objects. What is to be learned is the effect of the executed action on each object in different configurations.

In our experiments, we first collect a fixed-size dataset required for the training by interacting with the environment and then train the model. Note that this procedure can be turned into a buffer-based training where the model training and the data collection are done in parallel, similar to many reinforcement learning setups.

2) Data Collection: At each iteration of the exploration process, the robot picks a grounded action a, i.e., a specific parameterization of the action such as `pick-place`(2, −7.5, 1, 0.0), and executes it in the environment to observe effect e_i of action a on each object i. Object features before the execution of the action are recorded as the state vector. Here, object features are object types and poses with respect to the object frame that is going to be picked, which allows models to generalize to different object positions. Effects are the concatenation of (1) the position change of objects after the pick-up action, and (2) the position change of objects after the release action: $e_i = [\delta(o'^{\text{pick}}_i, o^{\text{pick}}_i), \delta(o'^{\text{place}}_i, o^{\text{place}}_i)]$ resulting in a 6-dimensional vector for each object (Fig. 2(b)). Such an effect representation filters out the movement effect of the object from the source location to the target location. In this way, the effect representation models what happens 'immediately after the pick-up' and 'immediately after the release' actions. Object and effect representations might have been selected as raw images as in [9], [10]; however, we opt for a simpler setup to compare different architectures in a controlled environment.

To compare different architectures in different settings, we collected three datasets that contain exactly two, three, or four objects. We combine these datasets to create a fourth dataset that contains a varying number of objects. Each dataset contains $(\{o_1, o_2, \ldots, o_n\}, a, \{e_1, e_2, \ldots, e_n\})$ triplets where n is the number of objects. We collect 120 K samples for two objects, 180 K for three objects, and 240 K for four objects. We use 80% of samples for training, 10% for validation, and 10% for testing.

3) Baselines: We compare our method with [9] and [10]. As the vanilla DeepSym architecture requires a fixed-size input and output, we modified it to make it suitable for our experiments. Namely, a maximum number of objects is determined for a given training session. Then, the input (and the output) vector is reshaped into $[o_{\text{grasped}}, o_{\text{released}}, o_{\text{rest}}]$ where o_{grasped} and o_{released} are the object features of the grasped and released objects, respectively, and o_{rest} is the object features of the remaining objects.

TABLE I
Effect Prediction Results for Different Methods

Dataset	DeepSym	Attentive	Relational
2 objects	2.22 ± 0.56	0.89 ± 0.10	**0.50 ± 0.03**
3 objects	3.06 ± 0.16	2.55 ± 0.09	**1.67 ± 0.02**
4 objects	4.26 ± 0.68	2.75 ± 0.12	**2.00 ± 0.04**
2-4 objects	2.38 ± 0.25	1.86 ± 0.12	**1.35 ± 0.04**

4) Training Details: All architectures are trained with the same hyperparameters throughout the text unless mentioned otherwise. We train models for 4000 epochs with five repetitions with different seeds. Adam optimizer [36] is used with a batch size of 128 and a learning rate of 0.0001. All network components (e.g., encoder, decoder) consist of two hidden layers with 128 hidden units. The number of attention heads for attentive models is set to four. We clip gradients by their norm to 10. Extended experimental details (training logs, layer gradients, and other training options) can be found at Weights & Biases[1] [37].

B. Effect Prediction Results

Firstly, we compare effect prediction results for different datasets in Table I[2]. The reported results are absolute errors summed over all dimensions (three dimensions for the pick-up effect and three dimensions for the release effect). The results show that the proposed method achieves significantly lower errors than others. Moreover, the variance is lower than others, indicating that Relational DeepSym is more robust to different seeds.

Errors increase as the number of objects increases. This is expected since the number of unique effects increases with the number of objects as the robot creates more complex structures in random exploration. Choosing the exploration schedule in a guided way, similar to experience replay in reinforcement learning [39], would be a promising future direction.

To compare the sample efficiency, we train each model on a subset of the full train set that is composed of interactions with two to four objects (432 K samples in total) and evaluate the effect prediction performance on the full test set. Fig. 5(a) shows the prediction errors over five runs for different models as the number of samples increases. Each model is trained for 1000 epochs except for the full train set where we train for 4000 epochs. The results show that the performances of Attentive DeepSym and Relational DeepSym increase in a similar fashion with the increasing sample size. However, the overall performance of Relational DeepSym is better than Attentive DeepSym on all sample sizes. This suggests that a bottleneck (in this case, the discrete representation) with a set of object symbols and relational symbols is more sample efficient than the one with only object symbols. In Attentive DeepSym, the bottleneck is essentially a set of object symbols, and relations are implicitly learned from these symbols (see Fig. 1 – bottom right). On the other hand, in Relational DeepSym, the object and

[1] https://api.wandb.ai/links/alper/xvpcogu1

[2] Averaged over five runs. Units are in centimeters. Welch's t-test [38] shows significant differences ($p < 0.02$ for all cases) between the proposed method and others.

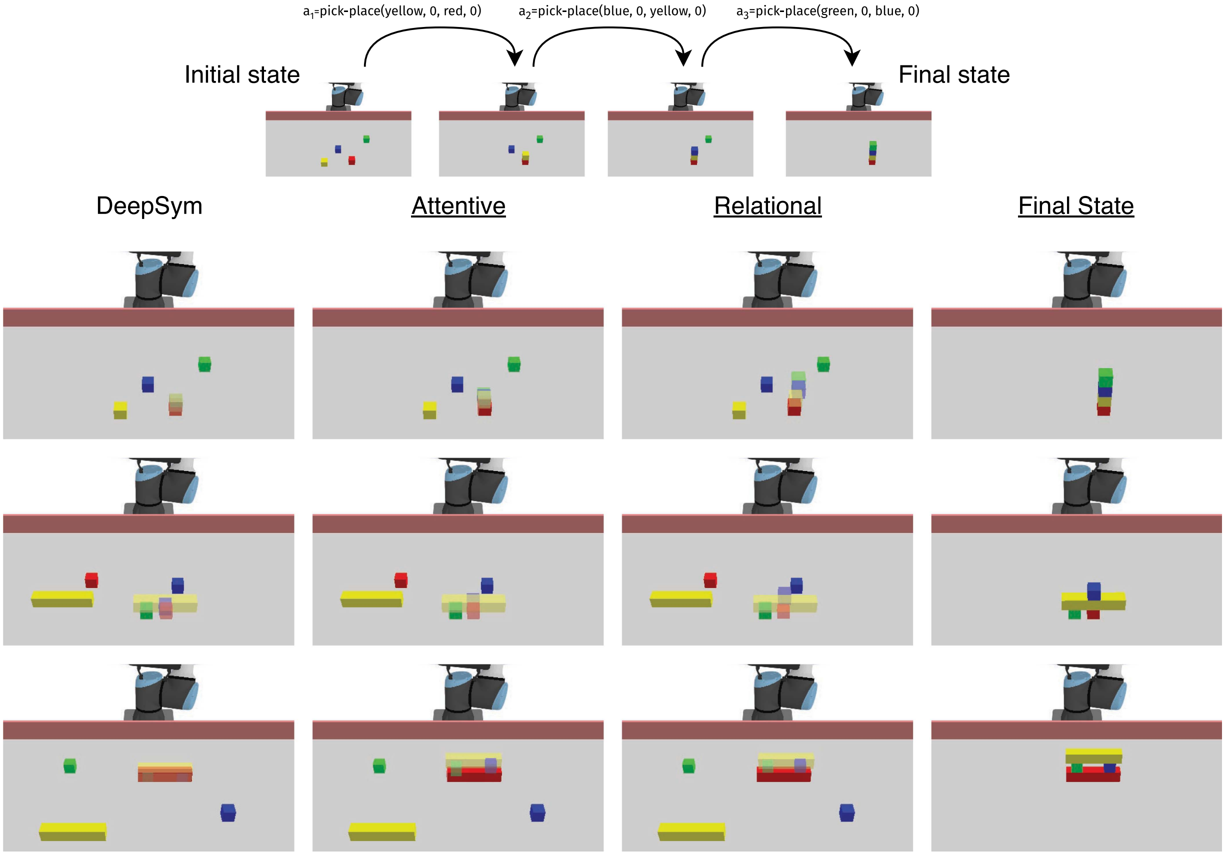
a_1=pick-place(yellow, 0, red, 0)
a_2=pick-place(blue, 0, yellow, 0)
a_3=pick-place(green, 0, blue, 0)
Initial state
Final state
DeepSym
Attentive
Relational
Final State

Fig. 3. Action sequence prediction results for different models. Top – An example action sequence is shown with the initial state (the input to models) in the first image and the ground truth final state after executing the action sequence in the fourth image. Rows 2–4 – The final object positions predicted by each network are shown with a transparent color.

the relational symbols are processed independently from each other (f_o and f_r in Fig. 1 – top) and combined in the aggregation function.

Next, we analyze the effect of the number of attention heads on the prediction performance. Fig. 5(b) shows the prediction errors for Attentive and Relational DeepSym for 1 to 4 attention heads. We see that the performance of Attentive DeepSym remains the same for different numbers of heads. As the discrete bottleneck in Attentive DeepSym is not relations but object symbols, the number of attention heads does not affect the effect prediction accuracy. However, the performance of Relational DeepSym increases with the increasing number of attention heads—the number of relations—since the model capacity increases with multiple relations in the aggregation step. The number of attention heads is a hyperparameter that needs to be tuned for each problem by finding the plateau in the performance, as done in [9] for the dimensionality of object symbols.

C. Action Sequence Prediction

In this section, using the effect predictions $\{\hat{e}_1, \ldots, \hat{e}_n\}$ of models, we predict the next state $\{\hat{o}'_1, \ldots, \hat{o}'_n\}$ by adding the prediction back into the position part of the state vector. Firstly, the predicted pick-up and release effects are summed with the state vector. Then, the movement from the pick-up to the release position is added for objects that are predicted to be picked up. This way, given an initial state, we can predict the final state the robot reaches after executing a sequence of actions. Here, the challenge is to understand what happens when an object is lifted and released on top of another object in the presence of multiple objects.

Fig. 3 shows action sequence prediction examples. Relational DeepSym's predictions are more accurate than others, especially in the z-axis, the most significant axis in these experiments. This shows that the proposed model understands that the presence of an object on top of another object will change the action results.

In Fig. 4, we analyze how models perform as the number of actions increases. We see that Relational DeepSym shows a slightly lower error than others. Errors increase for all models when the number of actions increases. This is an expected result since we add the effect prediction back into the state vector, effectively cascading the error over multiple steps.

D. Comparing Different Activations for Relations

In this section, we compare the performance of the Gumbel-sigmoid activation used for learning relational symbols with sigmoid, softmax, and Gumbel-softmax functions. Although Gumbel-sigmoid is also used for learning object symbols as well, we rather focus and ablate on the relational part. We train

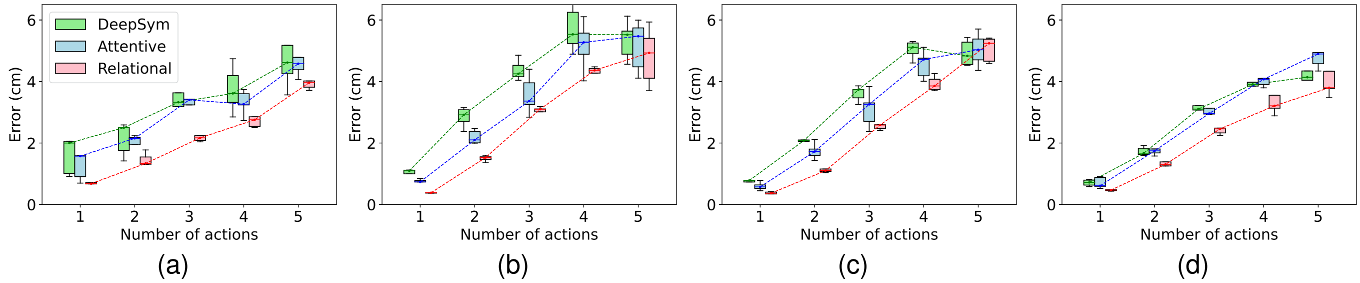

Fig. 4. Prediction errors for different models as the number of actions increases. (a) 2 objects. (b) 3 objects. (c) 3 objects. (d) 2–4 objects.

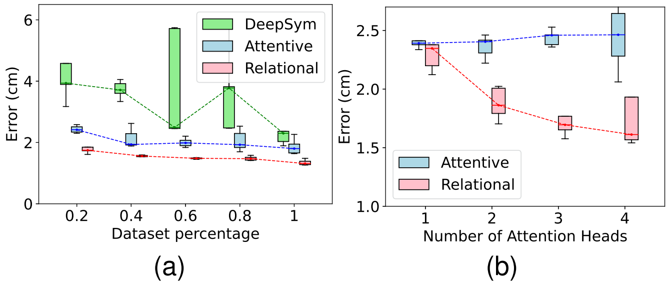

Fig. 5. Prediction errors for different models as (a) the number of samples and (b) attention heads increase.

TABLE II
EFFECT PREDICTION RESULTS WITH DIFFERENT ACTIVATIONS

	w/o rounding	w/ rounding
Gumbel-sigmoid	1.34 ± 0.09	**1.83 ± 0.41**
Gumbel-softmax	1.66 ± 0.16	2.70 ± 0.61
Sigmoid	1.32 ± 0.04	24.03 ± 3.71
Softmax	1.35 ± 0.03	23.26 ± 2.83

a Relational DeepSym model with the same hyperparameters as in Section IV-A except for the activation function used in (3) to compute object-object relations.

We report errors on the test set for different activations in Table II. Since learning a symbolic definition of the environment is a requirement that we want to satisfy to enable domain-independent planning with off-the-shelf AI planners [3], [5], [6], [9], [21], we can only use discrete outputs that are rounded to either 0 or 1 at inference time. As such, we report two different results: (1) without rounding and (2) with rounding. The results show that using Gumbel-sigmoid for learning relational symbols achieves lower errors in terms of effect prediction. This is expected since Gumbel-sigmoid is designed to approximate a Bernoulli distribution, which is in accordance with the distribution of pairwise relations; an object-object relation is either active or inactive.

V. CONCLUSION

In this letter, we proposed a new method to simultaneously learn object symbols and relations between objects in a single architecture. Namely, *discrete* attention weights are computed from object features to model relations between objects. As these weights are discrete, they can be regarded as relational symbols between objects. Such a feature is desirable because it allows us to model the environment with object symbols and relations between objects, which was not available previously [9], [10]. We showed that the proposed model achieves significantly lower errors than others in predicting the effects of (possibly a sequence of) actions on a varying number of objects and produces meaningful symbols that allow us to model the relations between objects for settings where the number of objects can vary.

As the next step, we plan to convert the learned symbols into PDDL operators [40] for domain-agnostic planning with off-the-shelf planners. Rules defined with learned symbols can be generated by a tree learning approach in which the features would be the binding of variable names to the symbol values, and the labels would be unique symbolic effects (unique changes in object symbols and relations). Alternatively, these operators can be learned by partitioning the symbolic dataset as in [22]. Such a conversion will remove the cascading of errors in action sequence prediction and allow for a fast search in the symbolic space by removing the need to use a neural network.

In the current report, we focused on the formation of discrete representation of continuous action-effect relations in a tabletop scenario. One future direction is to test the generalization effectiveness of the obtained discrete relations in more complex manipulation setups. In our experiments, we used the positional change of objects as the effect of an action, which we plan to extend to include orientation difference in $\mathcal{SO}(3)$. Learning action primitives, such as pick and place, together with object symbols is another promising direction that would allow the method to be applied to tasks where actions cannot be given apriori.

REFERENCES

[1] G. Konidaris, “On the necessity of abstraction,” *Curr. Opin. Behav. Sci.*, vol. 29, pp. 1–7, 2019.

[2] T. Taniguchi et al., “Symbol emergence in cognitive developmental systems: A survey,” *IEEE Trans. Cogn. Devlop. Syst.*, vol. 11, no. 4, pp. 494–516, Dec. 2019.

[3] G. Konidaris, L. Kaelbling, and T. Lozano-Perez, “Constructing symbolic representations for high-level planning,” in *Proc. AAAI Conf. Artif. Intell.*, 2014, pp. 1932–1940.

[4] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez, “From skills to symbols: Learning symbolic representations for abstract high-level planning,” *J. Artif. Intell. Res.*, vol. 61, pp. 215–289, 2018.

[5] E. Ugur and J. Piater, “Bottom-up learning of object categories, action effects and logical rules: From continuous manipulative exploration to symbolic planning,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 2627–2633.

[6] M. Asai and A. Fukunaga, "Classical planning in deep latent space: Bridging the subsymbolic-symbolic boundary," in *Proc. AAAi Conf. Artif. Intell.*, 2018, pp. 6094–6101.

[7] M. Asai and C. Muise, "Learning neural-symbolic descriptive planning models via cube-space priors: The voyage home (to strips)," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, 2021, Art. no. 371.

[8] M. Asai, H. Kajino, A. Fukunaga, and C. Muise, "Classical planning in deep latent space," *J. Artif. Intell. Res.*, vol. 74, pp. 1599–1686, 2022.

[9] A. Ahmetoglu, M. Y. Seker, J. Piater, E. Oztop, and E. Ugur, "Deepsym: Deep symbol generation and rule learning for planning from unsupervised robot interaction," *J. Artif. Intell. Res.*, vol. 75, pp. 709–745, 2022.

[10] A. Ahmetoglu, E. Oztop, and E. Ugur, "Learning multi-object symbols for manipulation with attentive deep effect predictors," 2022, *arXiv:2208.01021*.

[11] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.

[12] K. Mourao, R. P. Petrick, and M. Steedman, "Using kernel perceptrons to learn action effects for planning," in *Proc. Int. Conf. Cogn. Syst.*, 2008, pp. 45–50.

[13] F. Wörgötter, A. Agostini, N. Krüger, N. Shylo, and B. Porr, "Cognitive agents-A procedural perspective relying on the predictability of object-action-complexes (OACs)," *Robot. Auton. Syst.*, vol. 57, no. 4, pp. 420–432, 2009.

[14] A. Dehban, S. Zhang, N. Cauli, L. Jamone, and J. Santos-Victor, "Learning deep features for robotic inference from physical interactions," *IEEE Trans. Cogn. Devel. Syst.*, vol. 15, no. 3, pp. 985–999, Sep. 2023.

[15] F. S. Lay, A. S. Bauer, A. Albu-Schäffer, F. Stulp, and D. Leidner, "Unsupervised symbol emergence for supervised autonomy using multi-modal latent dirichlet allocations," *Adv. Robot.*, vol. 36, no. 1/2, pp. 71–84, 2022.

[16] R. Sun, "Symbol grounding: A new look at an old idea," *Philos. Psychol.*, vol. 13, no. 2, pp. 149–172, 2000.

[17] A. Ahmetoglu, E. Ugur, M. Asada, and E. Oztop, "High-level features for resource economy and fast learning in skill transfer," *Adv. Robot.*, vol. 36, no. 5/6, pp. 291–303, 2022.

[18] G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "Symbol acquisition for probabilistic high-level planning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2015, pp. 3619–3627.

[19] S. James, B. Rosman, and G. Konidaris, "Learning portable representations for high-level planning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 4682–4691.

[20] S. James, B. Rosman, and G. Konidaris, "Autonomous learning of object-centric abstractions for high-level planning," in *Proc. Int. Conf. Learn. Representations*, 2021.

[21] T. Silver, R. Chitnis, J. Tenenbaum, L. P. Kaelbling, and T. Lozano-Pérez, "Learning symbolic operators for task and motion planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 3182–3189.

[22] R. Chitnis, T. Silver, J. B. Tenenbaum, T. Lozano-Perez, and L. P. Kaelbling, "Learning neuro-symbolic relational transition models for bilevel planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 4166–4173.

[23] T. Silver, A. Athalye, J. B. Tenenbaum, T. Lozano-Perez, and L. P. Kaelbling, "Learning neuro-symbolic skills for bilevel planning," in *Proc. Conf. Robot Learn.*, 2022, pp. 701–714.

[24] A. Li and T. Silver, "Embodied active learning of relational state abstractions for bilevel planning," in *Proc. Conf. Lifelong Learn. Agents*, 2023. [Online]. Available: https://arxiv.org/abs/2303.04912

[25] J. Achterhold, M. Krimmel, and J. Stueckler, "Learning temporally extended skills in continuous domains as symbolic actions for planning," in *Proc. Conf. Robot Learn.*, 2023, pp. 225–236.

[26] N. Kumar, W. McClinton, T. Lozano-Pérez, and L. P. Kaelbling, "Overcoming the pitfalls of prediction error in operator learning for bilevel planning," in *Proc. RSS Workshop Learn. Task Motion Plan.*, 2023.

[27] T. Silver et al., "Predicate invention for bilevel planning," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 12120–12129.

[28] E. Ugur and J. Piater, "Refining discovered symbols with multi-step interaction experience," in *Proc. IEEE-RAS 15th Int. Conf. Humanoid Robots*, 2015, pp. 1007–1012.

[29] A. E. Tekden, A. Erdem, E. Erdem, M. Imre, M. Y. Seker, and E. Ugur, "Belief regulated dual propagation nets for learning action effects on groups of articulated objects," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 10556–10562.

[30] A. Tekden, A. Erdem, E. Erdem, T. Asfour, and E. Ugur, "Object and relation centric representations for push effect prediction," 2023. [Online]. Available: http://dx.doi.org/10.2139/ssrn.4386221

[31] Y. Huang, A. Conkey, and T. Hermans, "Planning for multi-object manipulation with graph neural network relational classifiers," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 1822–1829.

[32] E. Ugur, Y. Nagai, E. Sahin, and E. Oztop, "Staged development of robot skills: Behavior formation, affordance learning and imitation with motionese," *IEEE Trans. Auton. Ment. Develop.*, vol. 7, no. 2, pp. 119–139, Feb. 2015.

[33] E. Ugur, Y. Nagai, H. Celikkanat, and E. Oztop, "Parental scaffolding as a bootstrapping mechanism for learning grasp affordances and imitation skills," *Robotica*, vol. 33, no. 5, pp. 1163–1180, 2015.

[34] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," 2016, *arXiv:1611.00712*.

[35] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," 2016, *arXiv:1611.01144*.

[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015. [Online]. Available: http://dblp.uni-trier.de/db/conf/iclr/iclr2015.html#KingmaB14

[37] L. Biewald, "Experiment tracking with weights and biases," 2020. [Online]. Available: https://wandb.ai/

[38] B. L. Welch, "The generalization of 'student's'problem when several different population varlances are involved," *Biometrika*, vol. 34, no. 1/2, pp. 28–35, 1947.

[39] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. 4th Int. Conf. Learn. Representations*, Y. Bengio and Y. LeCun, 2016. [Online]. Available: http://arxiv.org/abs/1511.05952

[40] C. Aeronautiques et al., "PDDL the planning domain definition language," 1998. [Online]. Available: https://api.semanticscholar.org/CorpusID:59656859