Predictability-Based Curiosity-Guided Action Symbol Discovery

Burcu Kilic
Department of Computer Engineering
Bogazici University
Istanbul, Turkey
burcu.kilic@std.bogazici.edu.tr

Alper Ahmetoglu
Intelligent Robot Lab
Brown University
Providence, Rhode Island, US
aahmetog@cs.brown.edu

Emre Ugur
Department of Computer Engineering
Bogazici University
Istanbul, Turkey
emre.ugur@bogazici.edu.tr

***Abstract*—Discovering symbolic representations for skills is essential for abstract reasoning and efficient planning in robotics. Previous neuro-symbolic robotic studies mostly focused on discovering perceptual symbolic categories given a pre-defined action repertoire and generating plans with given action symbols. A truly developmental robotic system, on the other hand, should be able to discover all the abstractions required for the planning system with minimal human intervention. In this study, we propose a novel system that is designed to discover symbolic action primitives along with perceptual symbols autonomously. Our system is based on an encoder-decoder structure that takes object and action information as input and predicts the generated effect. To efficiently explore the vast continuous action parameter space, we introduce a Curiosity-Based exploration module that selects the most informative actions - the ones that maximize the entropy in the predicted effect distribution. The discovered symbolic action primitives are then used to make plans using a symbolic tree search strategy in single- and double-object manipulation tasks. We compare our model with two baselines that use different exploration strategies in different experiments. The results show that our approach can learn a diverse set of symbolic action primitives, which are effective for generating plans in order to achieve given manipulation goals.**

***Index Terms*—neuro-symbolic robotics, symbol emergence, intrinsic motivation**

I. Introduction

Humans and animals have the ability to perform abstract reasoning about their environments by learning abstract representations of actions and objects. For this, infants acquire high-level skills primarily by exploring their environment out of intrinsic curiosity, without an extrinsic end goal or rewards. Inspired by human developmental processes, we aim to demonstrate such autonomous exploration and high-level skill discovery in robotic agents. We present a robotic agent that can learn object and action abstractions by predicting the effects of its actions and exploring while trying to reduce its uncertainty in the predictions, in other words, exploring out of curiosity.

Symbolic representations of skills can enhance a robot's ability to reason and plan. Abstracting continuous sensorimotor information into discrete symbolic representations can simplify complex decision-making processes. These abstractions enable a robot to have better generalizable and transferable skills. Symbolic action primitives are utilized in symbolic planning, an effective method for efficiently generating action sequences to reach a specific goal state.

Previous neuro-symbolic robotic approaches [1]–[4] have shown success in learning object categories and relational object symbols from the bottleneck layer of an effect prediction encoder-decoder network. In [5], symbolic planning was performed using Planning Domain Description Language (PDDL) [6] with learned object symbols and a set of predefined abstract actions. However, these studies depend on manually defined high-level skills, and there is no autonomous skill discovery part.

Some prior work on high-level skill learning focused on learning classifiers for preconditions and effects of actions [7] [8]. These approaches depend on labeling preconditions and effects, which limits autonomous skill discovery. [9] learns symbolic actions as distinct visual state transitions. [10] demonstrated learning symbolic representations of actions from provided human demonstrations. [11] has proposed high-level skill learning with a vector quantization method from unlabeled demonstration data. [12] has proposed learning abstract relational representations for states and actions from unannotated trajectory data using relational critical regions. These approaches require giving human demonstration data that is inefficient and not fully autonomous.

To learn skills autonomously, [13] used an effect predictor deep neural network with separate object and action encoders, allowing abstract actions such as *grasp, push and pull* to be discovered. A task-specific exploration module was used to select actions with high effects in the environment since the action space is huge, noisy, and mostly inefficient in exploring fully. However, to detect the size of the effect an action has in the environment, it needs to be directly executed, which is ineffective and biologically unrealistic. An effective exploration strategy should be used such that without extrinsic rewards, only with intrinsic motivation can an agent learn meaningful skills in such a large action space.

This work was supported by the Scientific and Technological Research Council of Turkey (TUBITAK) ARDEB 1001 Program (124E227) and by the INVERSE Project under Grant 101136067, funded by the European Union.

There have been studies on intrinsic motivation and curiosity-driven exploration. VIME [14] rewards agents for actions that give high information gain, RE3 [15] uses the state entropy as an intrinsic reward, and [16] uses the error from the effect prediction model as a reward signal to the action policy network. In our study, we propose to select actions that maximize the entropy–uncertainty– in the effect prediction distribution as a curiosity-based exploration method, and we aim to extract diverse high-level action representations from the bottleneck layer of our effect prediction model.

The aim of this work is to find discrete action primitives that are effective in planning. For this, a predictive encoder-decoder neural network, which takes action parameters and object features as input and generates action effects as output, is proposed. The core idea is to binarize the embeddings in the bottleneck layer, enabling symbolic planning. We also propose to guide the exploration of the robot by a curiosity signal, which depends on the entropy of the neural network output activation. Finally, after a sequence of actions is generated via a tree-based symbolic planner, the continuous parameters of the corresponding action are found using a gradient-based method, which freezes network weights and applies a search in the action parameter space. Our main contributions to this study are:

1) A deep neural network with separate object and action encoders that predicts a Gaussian distribution over the effect, extracting binary discrete symbols from the bottleneck layer and turning the action primitives back to low-level parameters.
2) A Curiosity-Based Exploration module that selects the action resulting in the maximum entropy in the effect prediction model.
3) A Breadth-First Search (BFS)-based symbolic planner that uses the learned action primitives to generate action sequences that reach goal states in single- and double-object manipulation tasks.

II. Proposed Method

We provide the sensorimotor representation, followed by the predictive encoder-decoder network, continuous action parametrization, curiosity-based exploration module, and the planner.

A. Sensorimotor Representation

In the study, each object $o \in \mathbb{R}^4$ is represented as $o = [s_x, s_y, d, t]$ where s_x, s_y, d denotes the object's dimensions in different axes, and t corresponds to the type of the object. A state $s \in \mathbb{R}^{3m}$ is the state of the environment with m objects. Therefore, object features, which are used as the input of the predictive neural network, correspond to the concatenated vectors of all the objects' features.

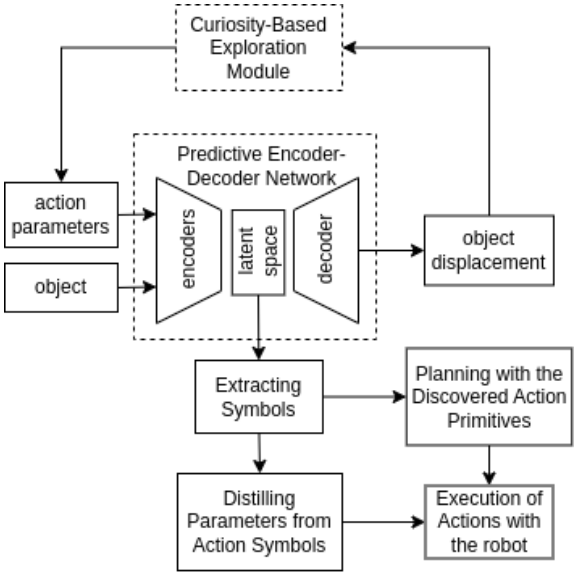

Fig. 1. Overview of the proposed method. The effect prediction model is an encoder-decoder deep neural network that predicts a distribution over the effect. The entropy of the distribution is given to the curiosity-based exploration module to guide the action selection process. The object is randomly initialized in the environment. The symbols are generated via binarization of the bottleneck layer of the effect prediction model.

The robotic action $a \in \mathbb{R}^{12}$ is a continuous vector concatenating the start, middle, and end points of the robot's trajectory.

$$a = [p_1, p_2, p_3], p_i = [x_i, y_i, z_i, g_i] \in \mathbb{R}^4, i = 1, 2, 3. \quad (1)$$

Here, x_i, y_i, z_i denote the coordinates of the robot relative to the target object, and g_i is a parameter that defines the state of the gripper. The gripper is open if $g_i >= 0.5$ and closed otherwise.

The effect $e \in \mathbb{R}^3$ of an action denotes the change in the target object's absolute position, $e = [\Delta x_t^o, \Delta y_t^o, \Delta z_t^o]$.

B. Discovering Symbols in Predictive Encoder-Decoder Network

1) Extracting Symbols: In our effect prediction model, we aim to extract action and object symbols from the bottleneck layer of the encoder-decoder deep neural network. Specifically, we propose to learn a mapping $\phi_o : \mathbb{R}^4 \rightarrow \mathbb{R}^j$ in the object encoder and a mapping $\phi_a : \mathbb{R}^{12} \rightarrow \mathbb{R}^k$ in the action encoder, discretizing the continuous outputs of these encoders into discrete symbols using a binary step function. After binarization, we obtain discrete representations for both the action and the object:

$$z_o = b(\phi_o(o)) \in \{0,1\}^j, z_a = b(\phi_a(a)) \in \{0,1\}^k. \quad (2)$$

2) Predictive Encoder-Decoder Network Architecture: The core of the architecture is a deep neural network with separate object and action encoders, followed by a single decoder. The network predicts a Gaussian distribution over the effect given action parameters and object features:

$$p(e \mid a, o) = \mathcal{N}(e; \mu(a, o), \sigma^2(a, o)) \tag{3}$$

The action and object encoders have an initial Batch Normalization layer. They consist of 4 hidden layers, the first 3 of which are linear layers with ReLU activation, and the final layer, which is a linear layer with Tanh activation function. The $\tanh$ activation function ensures the embeddings are between -1 and 1, which enables us to binarize the embeddings. It also allows continuous training in a wider range than sigmoid function and has a lower probability of causing vanishing gradients than sigmoid.

The decoder is a 4-layer perceptron with linear layers and ReLU activation in the hidden layers. It takes the concatenated object and action embeddings $z = [z_o, z_a]$. Layer Normalization layer is applied to the concatenated embeddings to ensure they have a stable distribution before further processing. In the final linear layer, the decoder predicts a mean μ and log variance $log(\sigma^2)$ for each of the x, y, and z axes, thereby creating three independent normal distributions over the predicted effect shown as Equation 4. Dropout is applied to all the hidden layers in the model for regularization. The predicted effect as network output is calculated as follows.

$$p(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right). \tag{4}$$

Based on the predicted and observed effects, the negative log-likelihood (NLL) loss is used:

$$\mathcal{L}_{\text{NLL}}(x) = -\log p(x \mid \mu, \sigma^2) = \frac{1}{2}\log(2\pi\sigma^2) + \frac{(x-\mu)^2}{2\sigma^2}. \tag{5}$$

To effectively extract distinct symbols from the latent space, we need to push the embeddings of different actions from each other and pull similar ones together. For this, we used Normalized Temperature-Scaled Cross Entropy (NT-Xent) Loss [17]. The concatenated action and object embeddings are normalized for a batch of N samples. With the normalized embeddings $\tilde{z}_i = \frac{z_i}{\|z_i\|}$ and temperature τ, we define the similarity as follows:

$$s_{ij} = \tilde{z}_i^\top \tilde{z}_j. \tag{6}$$

The NT-Xent loss is then

$$\mathcal{L}_{\text{NT-Xent}} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{\exp\left(\frac{s_{ii}}{\tau}\right)}{\sum_{j=1}^{N} \exp\left(\frac{s_{ij}}{\tau}\right)}, \tag{7}$$

Finally, with a loss coefficient $\lambda = 0.01$, the total loss is defined as

$$\mathcal{L}_{\text{total}} = \lambda\Big(\mathcal{L}_{\text{NLL}} + \mathcal{L}_{\text{NT-Xent}}\Big). \tag{8}$$

After training the effect prediction model, we z_o and z_a are found by binarizing the latent variables from the outputs of the encoders.

C. Planning with the Discovered Action Primitives

Breadth-first tree search (BFS) is used for planning where the branch factor corresponds to the number of discovered action primitives. The goal is to reach a given goal state s_g (within an error threshold) from an initial state s_i by an action sequence π. The states are defined by the absolute positions of all the objects in the environment. The planner iteratively expands the candidate sequences. If the state achieved by an action sequence π is within an error margin of s_g, the sequence is accepted as a solution. If no suitable sequence is identified within a maximum search depth, the algorithm terminates, indicating a failure to generate the plan with the learned action primitives. After finding an action sequence, each action should be executed, and for this, their continuous parameters should be found. In the next section, we will provide the details of the parameter distillation procedure.

D. Distilling Parameters from Discovered Action Symbols

Inspired by [18], we use an optimization-based distillation process to convert the symbols to continuous action parameters. Initially, we start with the set of action parameters that were used to train the effect prediction model. These parameters are the initial estimates for the distilled action parameters. Using the model's trained action encoder, we generate the action embeddings for these action parameters. We freeze the action encoder's weights and optimize only the action parameters using Stochastic Gradient Descent (SGD). The optimization is to minimize the mean squared error (MSE) between the model's encoded action embedding and the target binary symbol. After convergence, action symbol z_a, we select the candidate action parameter with the lowest error as the distilled action primitive. This process inverts the mapping $\phi_a(a)$ and maps the discrete symbols to continuous parameters, which allows the planner to use these primitives during task execution for achieving the given task.

E. Curiosity-Based Exploration Module

The robot interacts in a continuous action space, where efficient exploration is critical. In order to efficiently explore the continuous action space and learn meaningful action primitives, we propose to use a curiosity-based exploration approach based on exploration by selecting actions that maximize entropy in the effect prediction. Our effect prediction model outputs a Gaussian distribution over the effect, and the entropy of this distribution represents the uncertainty of the model regarding the predicted effect of an action. Therefore, by maximizing the entropy of the effect distribution, the method prioritizes actions that the model is uncertain. This way, we aim to accelerate the learning process, encourage the discovery of diverse and effective action primitives, and show how an intrinsic reward signal can make the model learn instead of relying only on extrinsic rewards.

Our proposed algorithm for curiosity-based exploration is given in Algorithm 1. In each exploration step, we uniformly sample a set of candidate action parameters. For each candidate, we forward it to our effect prediction model and

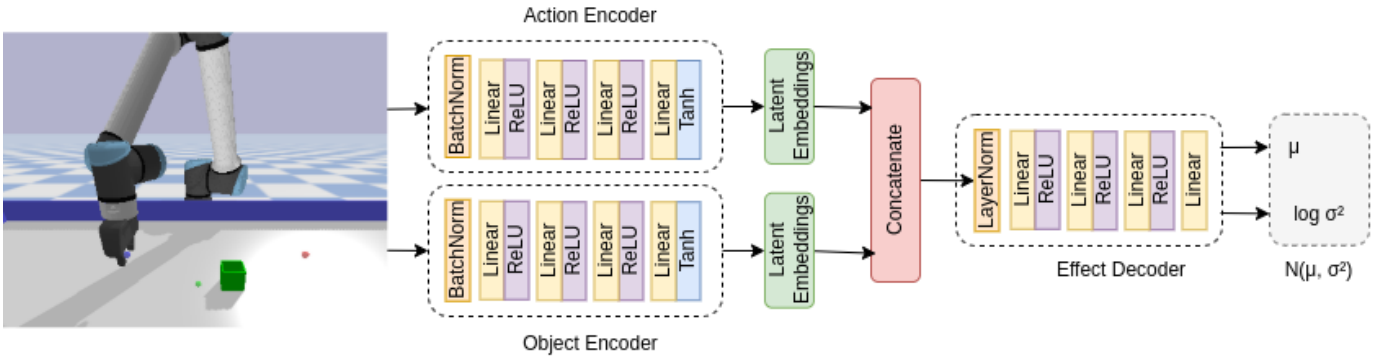

Fig. 2. Overview of the proposed effect prediction model. The first image shows the PyBullet environment with an example object. The blue, green, and red dots show the start, middle, and end points of the robot's trajectory, respectively. The action and object encoders generate latent embeddings, which are then concatenated. The decoder uses these combined embeddings to predict a Gaussian distribution over the object displacement caused by the action. The symbols are then extracted from the latent embeddings with the given method in Section II-D.

obtain a distribution over the predicted effects. The entropy of the distributions for each dimension is calculated, the mean entropy of all dimensions is found, and the candidate action with the maximum mean entropy is selected. We follow a greedy policy, selecting this action a to execute and save its effects to train the effect prediction model further.

Algorithm 1 Curiosity-Based Exploration

Require: Object parameters $o \in \mathbb{R}^4$
Ensure: Selected action $a^* \in \mathbb{R}^{12}$

```
Set number of candidate actions: N
for i = 1 to N do
    Uniformly sample action parameters
    a^(i) ~ U([−0.05, 0.05]^12)
    Create normal distributions for the effect by forwarding the model
    p(e | a^(i), o) = N(e; μ(a^(i), o), σ²(a^(i), o))
    for j = 1, 2, 3 do
        Calculate entropy for each axes x, y, z
        H_j(a^(i), o) = (1/2) log(2πe σ_j²(a^(i), o))
    end for
    Get the mean entropy
    H̄(a^(i), o) = (1/3) Σ_{j=1}^{3} H_j(a^(i), o)
end for
Select the action with the maximum entropy
a* = arg max_{i∈{1,...,N}} H̄(a^(i), o)
return a*
```

III. Experiments

The experiments are performed using a UR10 manipulation robot in a PyBullet simulation environment. In this section, we detail the experimental setup, evaluation, and comparisons with planning.

A. Experimental Setup

The encoders of our network have 4 hidden layers with 128 nodes per layer. tanh activation function is used to constrain the embeddings within the range [-1, 1]. The object encoder's output dimension was set to 2-bits, allowing up to $2^2 = 4$ distinct object categories, while the action encoder produces 3-bit output, allowing at most $2^3 = 8$ distinct action primitives. The concatenated embeddings are normalized using Layer Normalization and processed through the decoder, which has 4 hidden layers consisting of 128 nodes per layer. The model is trained with a mini-batch size of 512, a learning rate of 1e-5, and gradient clipping. The loss is the sum of $\mathcal{L}_{\text{NLL}}$ and $\mathcal{L}_{\text{NT-Xent}}$ with a loss coefficient $\lambda = 0.01$.

During each step of our Curiosity-Based Exploration, an object with a random size and type (hollow or non-hollow) is initialized. Using Algorithm 1, from a set of 2000 randomly sampled candidate action parameters, the action sequence a^* that maximizes the effect prediction entropy is selected and executed. The displacement of the object is saved as the effect. At every 512 steps, the effect prediction model is trained with the gathered (a, o, e) tuples for 10 update epochs. The entire exploration process takes 10,000 steps. After exploration, distilled parameters for all action symbols are obtained with the algorithm in Section II-D. The parameters are executed on the robot to annotate the learned high-level action primitives. Then, using the learned high-level action primitives, we perform single- and multi-object plannings in Section III-E.

B. Baselines

To measure the performance of our Curiosity-Based Model, we compare our findings with two baseline models: Random Exploration Model and Active Learning Model [13]. The Random Exploration Model is based on executing randomly sampled action parameters in randomly generated environments. The Active Learning Model is based on training the effect prediction model with only the actions that have high effects on the objects in the environment. It is a task-specific approach to avoid noise in the training dataset. Both approaches predict the effect directly instead of distribution, hence overlooking the uncertainties in the predicted effects.

TABLE I
MODEL PREDICTION ERRORS IN X, Y, Z AXES. UNITS ARE IN METERS.

	Curiosity-Based M.	Active Learning M.	Random Exploration M.
x	**0.0843**	0.0917	0.1236
y	**0.0828**	0.1090	0.1406
z	**0.1540**	0.1556	0.2210

C. Model Prediction Error

To evaluate our curiosity-based exploration module, a test dataset of 2400 samples generated with random exploration is used. Each sample includes random action parameters, a randomly generated environment state, and the effect of the action on the target object after execution. Interactions where the total effect is smaller than 0.8 along the three axes were excluded to reduce noise. In our curiosity-driven approach, the model predicts a distribution over the effect, and in this section, we use the mean of the distribution as the predicted effect. In the Random Exploration module and the Active Learning model, the output of the effect prediction model is used as the predicted effect, as they do not produce a distribution. We find the absolute error between this predicted effect and the ground truth effect separately for each dimension. When the model prediction errors are compared, as shown in Table I, our curiosity-based exploration approach has lower errors on all dimensions, meaning that it can provide a better generalization to the unseen data compared to the baseline approaches.

D. Discovered Action Primitives

After the exploration process, to understand and evaluate the diversity of the discovered action primitives, we first visually observe their prototypical executions and report their qualitative performance. To observe the executions of the action primitives, we distill the action parameters with the algorithm given in Section II-D. Some examples of high-level actions learned with our Curiosity-Based model are given in Figure 3. To compare the learned primitives with other baselines, we performed the same experiment with the Active Learning Module and the Random Exploration Module that were explained earlier. The comparison is given in Table II. As shown, our method learned six different action primitives, including different push primitives, grasp, and release actions, whereas the baselines could discover four or three of these primitives. The random exploration module learned less meaningful actions since the continuous action space consists of mostly noisy and null actions. The curiosity-based model performed the best since it allows for selecting novel actions, resulting in a diverse set of high-level skills. With this, we can conclude that our Curiosity-Based Model has learned a richer set of high-level action primitives than the other two baseline models.

E. Planning Performance

In this section, we analyze the planning and plan execution performance of our model and provide a comparison with the baselines. For this, we generated random goal states and ran the planner to generate plans using learned action primitives. We consider single- and double-object planning tasks to verify the effectiveness of the learned action primitives. We generated 100 planning tasks, and for each task, a randomly initialized state, random action parameters, and the resulting state were recorded. In single-object tasks, the environment consists of only one object, and in double-object tasks, the state has two objects, and the planner should decide on the target object to execute an action primitive. Then, with the collected initial and goal state pairs, we perform the planning (with a maximum search depth of 3 and an error threshold of 0.05) in all three models. The success rate is defined as the percentage of plans that successfully reach the goal state. The success rates of our Curiosity-Based model, the Active Learning model, and the Random Exploration model are reported in Table III. As shown, our model outperformed the baseline models, telling us

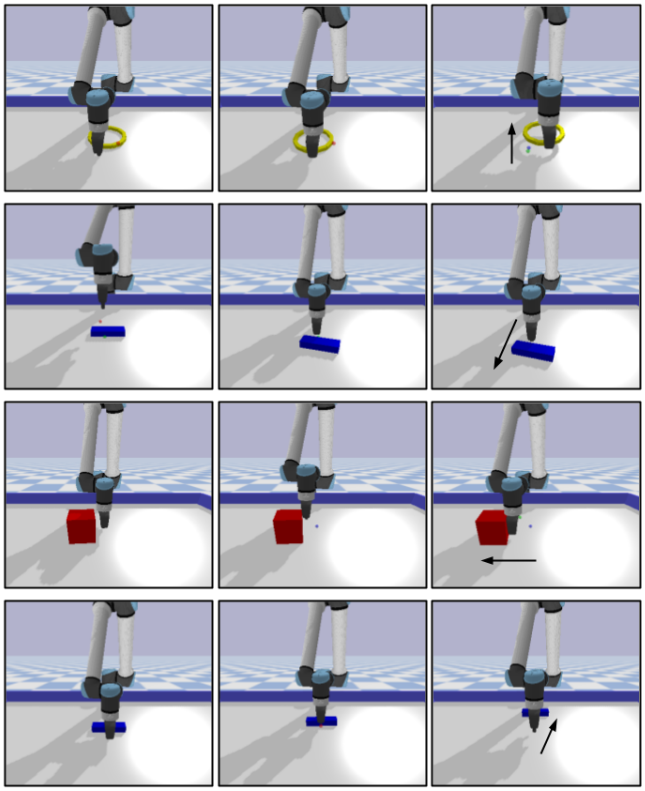

Fig. 3. Example of the learned action primitives in the Curiosity-driven model. From top to bottom; pick up, forward push, left push, pull.

TABLE II
LEARNED ACTION PRIMITIVES

	Curiosity-Based M.	Active Learning M.	Random Exploration M.
right push	+	+	+
left push	+	+	+
forward push	+	+	-
pull	+	+	-
grasp	+	-	+
pick and place	+	-	-

TABLE III
SUCCESSFUL PLAN RATIO OF THE MODELS

	Curiosity-Based M.	Active Learning M.	Random Exploration M.
Single-Object	**82%**	56%	13%
Double-Object	**59%**	38%	9%

that the learned high-level action primitives are more versatile.

Figure 4 shows sample plans generated by our planner with the learned primitives in the Curiosity-Based model. The first row shows a single-object plan consisting of left push and pull primitives, while the second row shows a two-object plan that has pick & place, pull, and left push primitives.

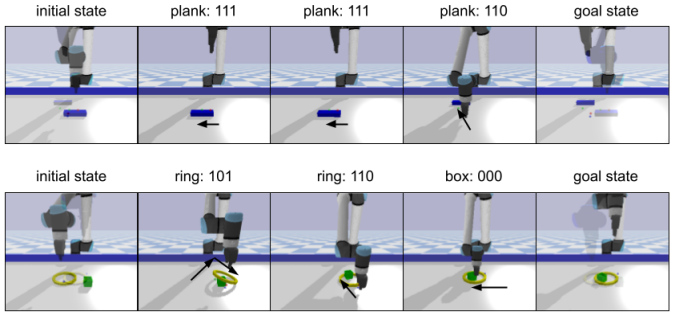

Fig. 4. Example plans generated by BFS with the learned primitives in Curiosity-Based Model. The goal state is shown transparently in the first column, and the initial state is shown so in the last column to emphasize the effect.

IV. CONCLUSIONS

In this study, we introduced a framework for discovering diverse high-level action primitives effective in next-state prediction and planning. We proposed a deep neural network to predict a Gaussian distribution over the effect of an action on an object. To efficiently train the model, we designed a curiosity-based exploration module that selects the most informative actions (that maximize the entropy in the predicted effect distribution). We utilized the learned action and object symbols to perform single- and double-object manipulation tasks with a Breadth-First Search (BFS)-based planner. We showed that when the robot explores its environment using an entropy-based curiosity signal, compared to random exploration module and effect-maximization, our method has better generalizing capability in predicting effects and learns more useful and diverse set of meaningful high-level action primitives. We also showed that, the action primitives found by our method can be effectively used by the symbolic planner in generating plans to achieve various single and paired object tasks, and in executing these plans thanks to our parameter distillation approach.

In our work, we focused on the effects of a single object's displacement during exploration and training. Future work can extend this to include the relations between objects [4] to reason more comprehensively in multi-object tasks. Additionally, although the BFS-based planning strategy performed reasonably well in our experiments, it may work slower when new objects or more complex tasks are introduced. In the future, we plan to translate the learned abstractions and rules into Planning Domain Definition Language (PDDL) [6] and use efficient off-the-shelf AI planners [19], [20].

REFERENCES

[1] E. Ugur, A. Ahmetoglu, Y. Nagai, T. Taniguchi, M. Saveriano, and E. Oztop, "Neuro-symbolic robotics," 2025, http://dx.doi.org/10.13140/RG.2.2.25854.09283.

[2] 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," *Journal of Artificial Intelligence Research*, vol. 75, pp. 709–745, 2022.

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

[4] A. Ahmetoglu, B. Celik, E. Oztop, and E. Ugur, "Discovering predictive relational object symbols with symbolic attentive layers," *IEEE Robotics and Automation Letters*, vol. 9, no. 2, pp. 1977–1984, 2024.

[5] A. Ahmetoglu, E. Oztop, and E. Ugur, "Symbolic manipulation planning with discovered object and relational predicates," *IEEE Robotics and Automation Letters*, 2025.

[6] C. Aeronautiques, A. Howe, C. Knoblock, I. D. McDermott, A. Ram, M. Veloso, D. Weld, D. W. Sri, A. Barrett, D. Christianson *et al.*, "Pddl—the planning domain definition language," *Technical Report, Tech. Rep.*, 1998.

[7] E. Ugur and J. Piater, "Bottom-up learning of object categories, action effects and logical rules: From continuous manipulative exploration to symbolic planning," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2627–2633.

[8] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez, "From skills to symbols: Learning symbolic representations for abstract high-level planning," *Journal of Artificial Intelligence Research*, vol. 61, pp. 215–289, 2018.

[9] M. Asai and C. Muise, "Learning neural-symbolic descriptive planning models via cube-space priors: The voyage home (to strips)," *arXiv preprint arXiv:2004.12850*, 2020.

[10] S. R. Ahmadzadeh, A. Paikan, F. Mastrogiovanni, L. Natale, P. Kormushev, and D. G. Caldwell, "Learning symbolic representations of actions from human demonstrations," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 3801–3808.

[11] H. Aktas and E. Ugur, "VQ-CNMP: Neuro-symbolic skill learning for bi-level planning," in *2nd CoRL Workshop on Learning Effective Abstractions for Planning*, 2024. [Online]. Available: https://openreview.net/forum?id=OZ1mOJSJEv

[12] N. Shah, J. Nagpal, P. Verma, and S. Srivastava, "From reals to logic and back: Inventing symbolic vocabularies, actions, and models for planning from raw data," *arXiv preprint arXiv:2402.11871*, 2024.

[13] B. Kilic, A. Ahmetoglu, and E. Ugur, "Learning action primitives in manipulation robots," 2025, [Online]. Available: https://burcukilic.github.io/publications/kilic-siu-2025, unpublished.

[14] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel, "Vime: Variational information maximizing exploration," *Advances in neural information processing systems*, vol. 29, 2016.

[15] Y. Seo, L. Chen, J. Shin, H. Lee, P. Abbeel, and K. Lee, "State entropy maximization with random encoders for efficient exploration," in *International Conference on Machine Learning*. PMLR, 2021, pp. 9443–9454.

[16] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *International conference on machine learning*. PMLR, 2017, pp. 2778–2787.

[17] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PmLR, 2020, pp. 1597–1607.

[18] H. Aktas, U. Bozdogan, and E. Ugur, "Multi-step planning with learned effects of partial action executions," *Advanced Robotics*, vol. 38, no. 8, pp. 562–576, 2024.

[19] J. Hoffmann, "Ff: The fast-forward planning system," *AI magazine*, vol. 22, no. 3, pp. 57–57, 2001.

[20] M. Helmert, "The fast downward planning system," *Journal of Artificial Intelligence Research*, vol. 26, pp. 191–246, 2006.