Robotics: Science and Systems 2025
Los Angeles, CA, USA, June 21–25, 2025

Bilevel Learning for Bilevel Planning

Bowen Li*†, Tom Silver‡, Sebastian Scherer*, and Alexander Gray†
*Carnegie Mellon University, †Centaur AI Institute, ‡Princeton University

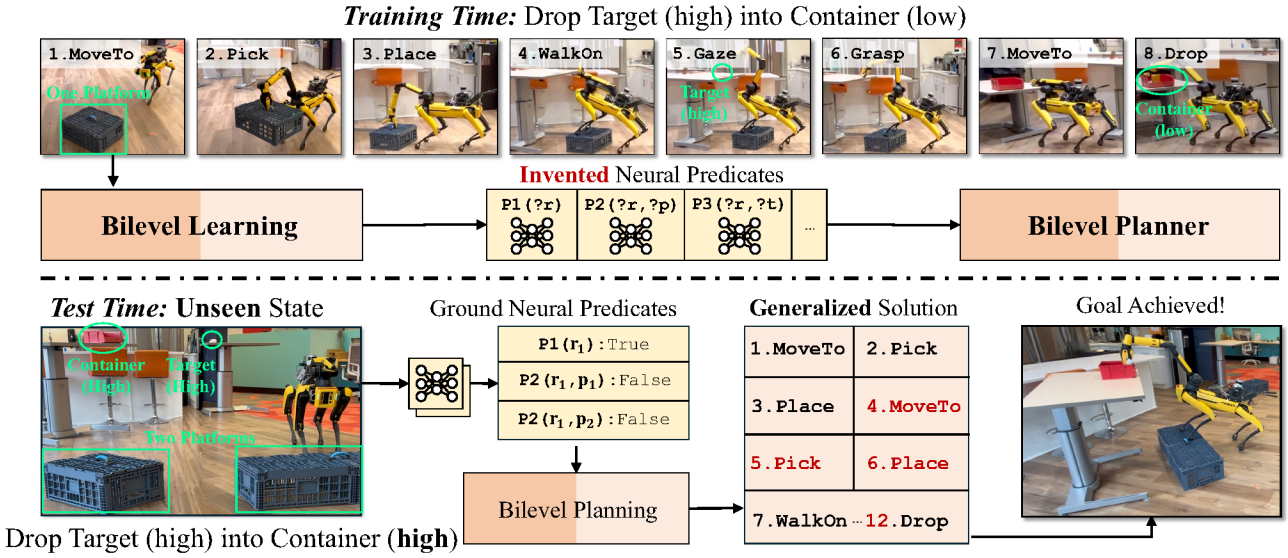

Fig. 1: (Top) Our bilevel learning framework invents **neural** predicates from training demonstrations (with one platform), which enable the learning of a hybrid bilevel planner. (Bottom) The invented predicates realize **zero-shot** compositional generalization over objects (with two platforms), where a longer solution with different action compositions is generated. Continuous action parameters are omitted for simplicity.

***Abstract*—A robot that learns from demonstrations should not just imitate what it sees—it should understand the high-level concepts that are being demonstrated and generalize them to new tasks. Bilevel planning is a hierarchical model-based approach where predicates (relational state abstractions) can be leveraged to achieve compositional generalization. However, previous bilevel planning approaches depend on predicates that are either hand-engineered or restricted to very simple forms, limiting their scalability to sophisticated, high-dimensional state spaces. To address this limitation, we present IVNTR, the first bilevel planning approach capable of learning neural predicates directly from demonstrations. Our key innovation is a neuro-symbolic bilevel learning framework that mirrors the structure of bilevel planning. In IVNTR, symbolic learning of the predicate "effects" and neural learning of the predicate "classifiers" alternate, with each providing guidance for the other. We evaluate IVNTR in six diverse robot planning domains, demonstrating its effectiveness in abstracting various continuous and high-dimensional states. While most existing approaches struggle to generalize (with $< 35\%$ success rate), our IVNTR achieves an average success rate of 77% on unseen tasks. Additionally, we showcase IVNTR on a mobile manipulator, where it learns to perform real-world mobile manipulation tasks and generalizes to unseen test scenarios that feature new objects, new states, and longer task horizons. Our findings underscore the promise of learning and planning with abstractions as a path towards high-level generalization. Project website: https://jaraxxus-me.github.io/IVNTR/.**

†Work was partly done during internship at Centaur AI Institute. Correspondence to {bowenli2,basti}@andrew.cmu.edu.

I. Introduction

Imitation learning has made significant recent strides [1–5], but generalization remains an open challenge, especially when new tasks require recomposing high-level concepts that are only implicit in the training data [6, 7]. In Figure 1, a robot has seen demonstrations of stepping onto a platform to grasp an object, and other demonstrations of dropping the object into a container. Now, faced with a new task where the container is also elevated, the robot should first move the two platforms appropriately, step onto one of them to grasp the object, and finally step onto the other to drop the object. Note that the platform arrangements must be completed before grasping the target, since the robot cannot move platforms with its hand full. This kind of learning and reasoning requires compositional generalization (new objects); sequential generalization (new and longer action sequences); and long-horizon planning with continuous state and action spaces with sparse feedback (goals). In sum, the robot should not just *imitate* demonstrations, but also *understand and leverage* the high-level concepts within the low-level states that are being demonstrated.

One promising direction to address these challenges is to learn and plan with *abstractions* [8–14]. In this work, we continue a line of recent inquiry on learning abstractions for *bilevel planning* [15–21]. In bilevel planning, continuous low-level states are mapped into a symbolic relational state space defined by *predicates* such as `Viewable(?robot,?target)` or

`On(?robot,?platform)`. Planning proceeds jointly in the symbolic high-level space and the continuous low-level space. The key idea is that this hybrid planning can be more efficient and effective than reasoning solely in the low-level space.

The performance of bilevel planning depends substantially on the predicates used to define the abstract state space [17]. To avoid the need for a human engineer to manually define predicates for every new domain, recent work has considered *learning predicates* from data [17, 21–25]. Broadly, three approaches have emerged. The most direct one relies on human feedback (labels or guidance) during the predicate learning process [24–26], which is labor-intensive and does not guarantee useful abstractions for planning [17]. The second approach *invents* predicates with surrogate objectives that are easy to optimize, *e.g.*, reconstruction loss [27–29] or bisimulation [12, 30]. While these methods simplify learning, they complicate planning due to the mismatch between the surrogate objectives and the actual planning goals [17]. The third approach directly invents predicates for efficient planning, making planning "easy" but learning "hard," as objectives like *total-planning-time* and *expected-planning-success* are difficult to optimize [17]. To address this, previous works have used program synthesis with classical grammars [17] and foundation model-based techniques [21, 31]. However, in both cases, the predicates are invented from programmatic and pre-defined classifiers, which are limited in flexibility and scalability.

Our main contribution is b**I**le**V**el lear**N**ing from **TR**ansitions (IVNTR), the first approach capable of learning *neural* predicates that are optimized for efficient and effective bilevel planning. Since directly incorporating the planning objective into network training is challenging, our IVNTR instead constructs a candidate neural predicate pool, which is later subselected [17]. The key insight behind our approach is to center learning around the *effects* of predicates, which provide two major benefits: (1) they enable the derivation of supervision labels for *transition* pairs, yielding a well-structured learning objective for training the neural network; and (2) the inherent sparsity of predicate effects, combined with neural learning signals, facilitates efficient symbolic learning of their structure. To this end, IVNTR presents a novel bilevel learning framework, inspired by the structure of bilevel planning itself. Similar to the alternation between high-level symbolic search and low-level neural sampling in bilevel planning, IVNTR interleaves symbolic effect learning and neural classifier learning in an iterative process. In each iteration, the symbolic learning proposes a candidate predicate effect across different actions, which provides labels for neural learning on transition pairs. Once the neural classifier converges, its validation loss guides the symbolic learning to propose the next candidate that could minimize the loss in the new iteration. This iterative bilevel learning ultimately yields a compact set of neural predicates, which are then selected to optimize the planning objective [17]. The final set of invented predicates seamlessly integrates into operator and sampler learning frameworks [17, 32], ultimately forming a fully functional bilevel planner.

To evaluate the effectiveness of IVNTR, we conduct extensive experiments across six diverse robot planning domains. These domains feature a wide range of low-level state representations, from SE(2) and SE(3) poses to high-dimensional point clouds. Furthermore, as shown in Figure 1, by leveraging relational predicates and AI planning, IVNTR zero-shot generalizes to tasks with unseen entity compositions. Finally, we deploy IVNTR on a quadruped mobile manipulator (Boston Dynamics Spot) for two long-horizon mobile manipulation tasks. The learned predicates successfully abstract complex continuous states into representations compatible with the AI planner, while also providing actionable guidance for the samplers. We believe IVNTR represents a pivotal step towards learning high-level abstractions from sophisticated low-level states.

II. Problem Formulation

We propose a method that uses an offline demonstration dataset to learn planning *abstractions* that generalize to test tasks with unseen objects and action compositions. In this section, we describe the formal problem setting. We follow the notation system introduced in previous work [17]; see Appendix A for a complete notation glossary.

Planning problems are defined within a certain *planning domain* $\langle \Lambda, \mathcal{C}, f, \Psi_{\text{g}}, \Psi_{\text{sta}} \rangle$ with a task distribution $\mathcal{T}$, where we can sample a *planning task* $T \sim \mathcal{T} = \langle \mathcal{O}, \mathbf{x}_0, g \rangle$.

Λ is a finite set of object *types* $\lambda \in \Lambda$. For example, the Climb-Transport domain depicted in Figure 2 has three object types: $\Lambda = \{\text{robot}(\mathtt{r}), \text{platform}(\mathtt{p}), \text{target}(\mathtt{t})\}$. Each type is associated with a set of *features* that characterize the state of an object of that type.[1] For example, robot has features "BasePose", "HandPose", and "GripperOpenPercent", among others. A specific *task* T is characterized by objects $\mathcal{O} = \{\mathtt{o}_1, \mathtt{o}_2, \cdots, \mathtt{o}_N\}$, each associated with one type in Λ. Objects are fixed within tasks but vary between tasks. The state of a task $\mathbf{x} \in \mathcal{X}$ is defined by an assignment of feature values to all objects in the task. For simplicity of exposition, we assume that a state with N objects can be represented as a matrix $\mathbf{x} \in \mathbb{R}^{N \times K}$ for some domain-specific constant K; however, we show in experiments that our approach can be applied to more sophisticated object-centric state representations as well.

The action space for a domain is characterized by a set of M *parametrized controllers* $\mathcal{C} = \{\mathtt{C}_1, \mathtt{C}_2, \cdots, \mathtt{C}_M\}$, each of which has an object type signature $(\lambda_1, \lambda_2, \cdots, \lambda_v)$ and a continuous parameter space Ω. For example, in Figure 2, `MoveToReach` has type signature $(\text{robot}, \text{platform})$, and continuous parameters $\Omega = \text{SE}(2)$ defining an offset 2D pose for the robot relative to the platform. A *ground action* is a controller with fully specified parameters, e.g., $\texttt{MoveToReach}(\mathtt{r}_1, \mathtt{p}_1, \omega)$ for a certain $\omega \in \Omega$. We use underline notation to represent grounding: $\underline{\mathtt{C}}$ is a certain ground action. A *lifted action* is controller with object parameter placeholders, which are typically prefixed with ?, e.g., $\texttt{MoveToReach}(?\text{r}, ?\text{p}, \cdot)$. States and actions are related through a known transition function $f(\mathbf{x}, \underline{\mathtt{C}}) \mapsto \mathbf{x}'$, which the robot can use to plan.

[1] Unlike previous work [17], we do not assume that features are scalars; high-dimensional images and point clouds are also allowed.

A *predicate* ψ is defined by an object type signature $(\lambda_1, \lambda_2, \ldots, \lambda_u)$ and a classifier $\theta_\psi : \mathcal{X} \times \mathcal{O} \to \{\text{True}, \text{False}\}$, where $\theta_\psi(\mathbf{x}, (\mathtt{o}_1, \ldots, \mathtt{o}_u))$ evaluates the truth value of a ground predicate based on the continuous features of the input objects. For example, the predicate `In` has type signature $(\text{target}, \text{target})$ and a classifier that uses the poses and shapes of two targets to determine whether one is "in" the other. A *ground predicate* $\underline{\psi}$ has fully specified objects. For simplicity, we denote $\theta_{\underline{\psi}}(\mathbf{x}) \triangleq \theta_\psi(\mathbf{x}, (\mathtt{o}_1, \ldots, \mathtt{o}_u))$. A *lifted predicate* has placeholders for objects, e.g., `In(?t,?t)`.

Following previous work [17], we assume that a small set of *goal predicates* Ψ_G is known and used to characterize task goals. In particular, a goal g is defined by a set of ground predicates that must evaluate to True in a state for the goal to be satisfied. For example, the goal in Figure 2 has only one ground predicate, $\mathtt{In}(\mathtt{t}_1, \mathtt{t}_2)$. In this work, we make an additional assumption that any relevant *static predicates* Ψ_{sta} are known. A predicate is static if its evaluation never changes within a task (see Appendix E for examples). Conversely, a predicate is *dynamic* if its evaluation could change within a task; examples are provided later in Definition 1.

A solution to a task is a plan $\pi = [\underline{\mathtt{C}}_1, \underline{\mathtt{C}}_2, \cdots, \underline{\mathtt{C}}_H]$, that is, a sequence of H ground actions such that successive application of the transition model $\mathbf{x}_i = f(\mathbf{x}_{i-1}, \underline{\mathtt{C}}_i)$ on each $\underline{\mathtt{C}}_i \in \pi$, starting from $\mathbf{x}_0$, results in a final state $\mathbf{x}_H$ where g holds. For instance, the plan depicted in Figure 1 bottom right finally leads to the state where $\mathtt{In}(\mathtt{t}_1, \mathtt{t}_2)$ holds. In sum, to generate the plan π for a task, in each state, the robot needs to predict: (1) the action class $\mathtt{C} \in \mathcal{C}$, (2) the objects as the discrete parameters, and (3) the continuous parameter $\omega \in \Omega$.

During training, the robot has access to an offline demonstration dataset $\mathcal{D} = \{(T_i, \pi_i)\}_{i=1}^{B}$, which consists of B task and solution pairs sampled from the task distribution $\mathcal{T}^{\text{train}}$. Note that since the transition function f is known and deterministic, we can also recover the intermediate states from $\mathbf{x}_0$. During test time, the robot is required to solve held-out tasks sampled from a different *test* distribution $\mathcal{T}^{\text{test}}$. In practice, for the sake of evaluating generalization, test tasks typically contain new and more objects than training tasks. For example, as shown in Figure 2, all training tasks only have 1 platform, but during test, there are 2 platforms. The difference in object compositions could result in different lengths of plans with a different action order, requiring the method to *generalize* by understanding the implicit concepts present in the training demonstrations.

III. Bilevel Planning

In this work, we propose a method for learning predicates that can be used for *bilevel planning*. We now provide a brief review of bilevel planning and refer to other references for a more in-depth discussion [15–17, 19–21, 24, 32, 33].

Bilevel planning uses relational abstractions to achieve sequential and compositional generalization. The two principal abstractions are *predicates* (state abstractions) and *operators* (action abstractions). Bilevel planning also uses relational *samplers* to generate possible ground actions from operators. The key idea is that planning jointly in the abstract transition system and the low-level transition system can be much more efficient than planning in the low-level transition space only.

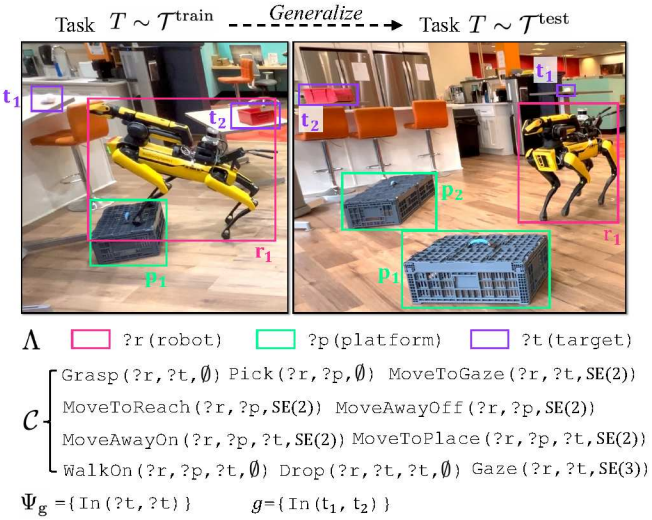

Fig. 2: The Climb-Transport domain is presented as a running example. We have displayed one typical training and one test task on the top. The types, actions, and provided predicates are shown at the bottom.

Definition 1 (Operator)**.** The *operator* for a parametrized controller $\mathtt{C}$ is a tuple, $\mathtt{Op}^{\mathtt{C}} = \langle \mathtt{Var}, \mathtt{Pre}, \mathtt{Eff}^+, \mathtt{Eff}^- \rangle$, where `Var` is a tuple of object placeholders matching the type signature of $\mathtt{C}$, and $\mathtt{Pre}, \mathtt{Eff}^+, \mathtt{Eff}^- \subseteq \Psi$, respectively *preconditions*, *add effects*, and *delete effects*, are each a set of lifted predicates defined with variables in `Var`. Ψ is a predicate set.

For example, the operator for $\mathtt{Grasp}(?\mathtt{r}, ?\mathtt{t})$ could be:

$$\begin{aligned}\mathtt{Pre} &= \{\mathtt{HandEmpty}(?\mathtt{r}), \mathtt{HandSees}(?\mathtt{r}, ?\mathtt{t})\},\\ \mathtt{Eff}^+ &= \{\mathtt{Holding}(?\mathtt{r}, ?\mathtt{t})\},\\ \mathtt{Eff}^- &= \{\mathtt{HandEmpty}(?\mathtt{r}), \mathtt{HandSees}(?\mathtt{r}, ?\mathtt{t})\}.\end{aligned}$$

Given a task $T = \langle \mathcal{O}, \mathbf{x}_0, g \rangle$, bilevel planning (Figure 3b) starts by using predicates to generate an *abstract state* consisting of all ground predicates with objects $\mathcal{O}$ whose classifiers evaluate to True in $\mathbf{x}_0$. The initial ground predicates, together with the operator set and goal g, can then be input to an AI planner [34] to generate a plan *skeleton*, $\bar{\pi}$ with partially grounded actions with unspecified continuous parameters, $\bar{\mathtt{C}}$. To refine the actions in this skeleton $\bar{\mathtt{C}} \in \bar{\pi}$ into fully grounded $\underline{\mathtt{C}}$ with the continuous parameters ω, bilevel planning leverages *samplers*.

Definition 2 (Sampler)**.** The *sampler* $\eta^{\mathtt{C}}$ for a planning operator $\mathtt{Op}^{\mathtt{C}}$ with v object placeholders is a conditional distribution $\omega \sim \eta^{\mathtt{C}}(\cdot \mid \mathbf{x}, (\mathtt{o}_1, \ldots, \mathtt{o}_v))$ that proposes continuous action parameters for $\mathtt{C}((\mathtt{o}_1, \ldots, \mathtt{o}_v), \cdot)$ given a state $\mathbf{x}$.

Note that unlike the deterministic operators, samplers are usually stochastic and may fail in certain steps. Thus, bilevel planning alternates between the AI planner and samplers, using the predicate classifiers θ_Ψ as "guidance" in each step to compensate for potential sampling failures.

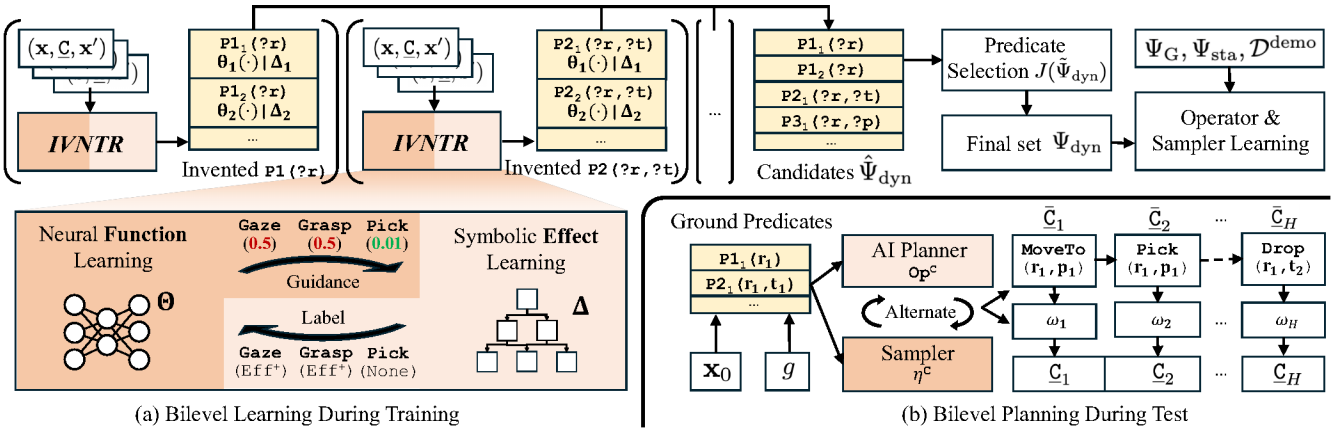

Fig. 3: (a) Overview of IVNTR during training. Given transition pairs in the continuous space, IVNTR invents neural predicates with different arguments parallelly, resulting in a candidate set. A subset that minimizes the planning objective $J(\cdot)$ is selected from the candidates, which serves as the final Ψ_{dyn}. With the complete predicate set, sampler and operator learning can be achieved. (b) Bilevel planning with operators and samplers during test. Compositional ground predicates serve as inputs to the AI planner and provide guidance to the samplers.

Assuming we have a complete predicate set, previous work has studied the problem of learning *operators* [32] and *samplers* [16, 19] from the demonstration dataset $\mathcal{D}$. Since the predicates, learned operators, and samplers are *relational*, they can be generally applied to held-out test tasks sampled from $\mathcal{T}^{\text{test}}$. However, with an insufficient predicate set—for example, with only Ψ_{G} and Ψ_{sta}—bilevel planning can be intractably slow [17]. We next introduce the IVNTR framework, which closes this gap by automatically inventing dynamic predicates for efficient bilevel planning.

IV. Methodology

The problem of inventing dynamic predicates Ψ_{dyn} can be decomposed into *symbolic learning*—how many predicates should be invented, and with what type signatures—and *classifier learning*, determining θ_ψ for each invented predicate $\psi \in \Psi_{\text{dyn}}$. Previous approaches [17, 21] address these problems via a "define-then-select" two-stage pipeline. In the first stage, for each predicate candidate $\hat{\psi}$ with known variable types, its classifer is pre-defined via program synthesis [17] or pre-trained foundation models [21]. These candidates form a large predicate pool $\hat{\Psi}_{\text{dyn}}$. In the second stage, to subselect predicates from the pool, each candidate predicate set $\tilde{\Psi}_{\text{dyn}} \subseteq \hat{\Psi}_{\text{dyn}}$ is scored with a function $J(\tilde{\Psi}_{\text{dyn}})$ that measures both planning *efficiency* and *effectiveness*. A key limitation of this "define-then-select" pipeline is that the classifiers within $\hat{\Psi}_{\text{dyn}}$ are restricted to a relatively simple set. Scaling to more general classifiers, e.g., neural networks, is nontrivial, since $J(\tilde{\Psi}_{\text{dyn}})$ is highly non-differentiable. To address this, we propose IVNTR, a "learn-then-select" approach that leverages *bilevel learning*.

As depicted in Figure 3 (a), given the domain types Λ, IVNTR enumerates all possible typed variable compositions (with maximum input arity). Since the input features for the predicate classifier depend on its argument types, IVNTR invents predicates with different arguments group by group parallelly. For the invention of each group, IVNTR draws inspiration from bilevel planning itself, where planning alternates between the symbolic level and the low level. Similarly, IVNTR interleaves *symbolic learning* and *neural learning*, with each providing guidance for the other. Specifically, symbolic learning proposes *effect vectors* that represent the add and delete effects for candidate predicates across all operators. Neural learning uses these effect vectors to provide supervision for classifier learning. The validation loss in neural learning feeds back into symbolic learning, and the process repeats. In this section, we describe these steps in detail via the exemplar predicate group $\psi \in \Psi^{\texttt{Var}}$, with the typed variables $\texttt{Var} = (?\texttt{r}, ?\texttt{t})$.

A. Effect Vectors as Supervision for Neural Learning

Suppose we had access to the symbolic components of a predicate ψ, but did not yet know its classifier θ_ψ. Suppose further that we had knowledge of all appearances of ψ in the effect sets $(\texttt{Eff}^+, \texttt{Eff}^-)$ for each operator $\texttt{Op}^{\texttt{c}}$. We now describe how this knowledge—which we do not actually have, but which will be suggested by symbolic learning—can be used for supervised learning of the classifier θ_ψ.

Recall that we have access to demonstrations $\mathcal{D}$, and for each demonstrated task T, we can recover the solution trajectory, $[\mathbf{x}_0, \underline{\mathtt{c}}_0, \mathbf{x}_1, \underline{\mathtt{c}}_1, \cdots, \underline{\mathtt{c}}_{H-1}, \mathbf{x}_H]$. If we knew the initial state ground predicates $\underline{\psi}$ in $\mathbf{x}_0$, then we could immediately recover all ground predicates for all the states in the trajectories by chaining together the operator effects. Then, a simple binary classification framework could easily address our neural learning problem. However, we do not have access to the initial state ground predicates—we only have access to operator effects—so we do not have direct knowledge of the abstract states in the demonstration. Nonetheless, we can still provide supervision for neural learning by leveraging the ground predicates that are added, deleted, or stay unchanged in each *transition pair*. We provide this supervision by way of *predicate effect vectors*, including the *lifted effect vector* for a domain, and the *ground effect vector* for a transition.

Definition 3 (Lifted Effect Vector)**.** The *lifted effect vector* for

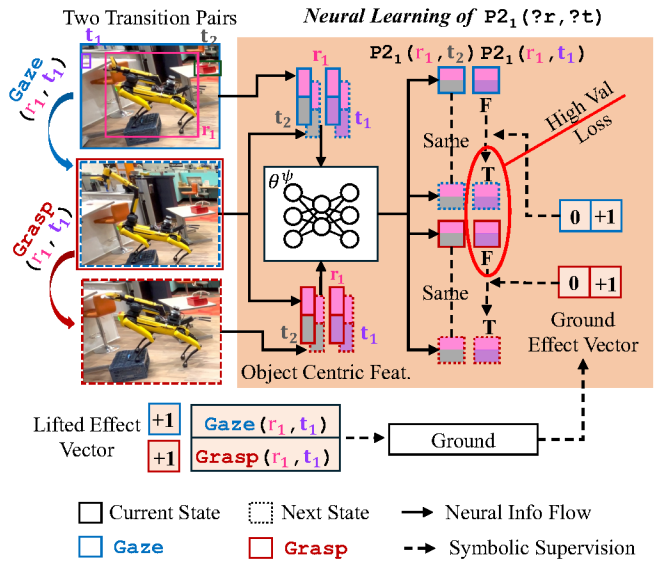

Fig. 4: Detailed neural learning process for predicate $\texttt{P2}_1(?\texttt{r}, ?\texttt{t})$. From the demonstration dataset, we display two transition pairs (one for each action, in total four states) on the left. The neural network takes object centric features as input, predicting ground predicates (in total eight values). At the bottom, we display an example lifted effect vector for action Grasp, Gaze as $\Delta_i^\psi = [+1, +1]$. With the ground effect vector, supervisions can be derived on the predicted values. Due to the unreasonable effect supervisions, the intermediate state is labeled as both `True` and `False`, resulting in high validation loss.

predicate ψ is $\Delta^\psi = [\delta_1^\psi, \cdots, \delta_M^\psi] \in \{-1, 0, 1\}^M$ where:

$$\delta_i^\psi = \begin{cases} 1 & \psi \in \texttt{Eff}^+ \text{ for } \texttt{C}_i \\ -1 & \psi \in \texttt{Eff}^- \text{ for } \texttt{C}_i \\ 0 & \text{otherwise.} \end{cases}$$

For example, in Figure 4, the effect vector $[+1, +1]$ specifies that predicate $\psi = \texttt{P2}_1(?\texttt{r}, ?\texttt{t})$ is the add effect for both $\texttt{Gaze}(?\texttt{r}, ?\texttt{t})$ and $\texttt{Grasp}(?\texttt{r}, ?\texttt{t})$[2].

The lifted effect vector is a favorable symbolic representation of a lifted predicate, since its shape doesn't depend on task object compositions and can thus be learned more efficiently, as we will see. However, to train the neural classifier on the transition pairs, we will need to derive supervision on *ground predicates*, which is achieved by the *ground effect vector*.

Definition 4 (Ground Effect Vector)**.** Let $\mathcal{O}$ be the object set in a task T, $\underline{\texttt{C}}_i$ be a ground action with objects $\mathcal{O}_{\underline{\texttt{C}}_i} \subseteq \mathcal{O}$, then the ground effect vector $\boldsymbol{t}^{\psi, \underline{\texttt{C}}_i} = [t_1, \cdots, t_P] \in \{-1, 0, 1\}^P$ for predicate ψ grounded on $\mathcal{O}$ is defined as:

$$t_p = \begin{cases} \delta_i^\psi, & \text{if } \mathcal{O}_{\underline{\psi}_p} \subseteq \mathcal{O}_{\underline{\texttt{C}}_i}, \\ 0, & \text{otherwise,} \end{cases}$$

where $\underline{\psi}_p$ denotes the p-th atom with objects $\mathcal{O}_{\underline{\psi}_p}$, among the total of P ground predicates. For example, in Figure 4, ground effects for $\texttt{P2}_1(\texttt{r}_1, \texttt{t}_1)$ will be $+1$ for both ground actions $\texttt{Gaze}(\texttt{r}_1, \texttt{t}_1)$ and $\texttt{Grasp}(\texttt{r}_1, \texttt{t}_1)$, while ground effects for $\texttt{P2}_1(\texttt{r}_1, \texttt{t}_2)$ will be 0, as $(\texttt{r}_1, \texttt{t}_2) \not\subseteq (\texttt{r}_1, \texttt{t}_1)$.

Importantly, this implies that the "value" of the (potentially) non-zero entry of all ground effects from the action class $\texttt{C}_i$ equals δ_i^ψ, while the "position" of the non-zero entry is decided by the object set $\mathcal{O}_{\underline{\texttt{C}}}$ and the predicate grounding (see Appendix B and Appendix C for more explanations).

Now, we are able to train the neural classifier θ_ψ on the transition pairs with supervisions from Δ^ψ. Specifically, consider a transition pair: $(\mathbf{x}, \underline{\texttt{C}}_i, \mathbf{x}')$, we first construct the ground effect vector $\boldsymbol{t}^{\psi, \underline{\texttt{C}}_i}$ using $\delta_i^\psi \in \Delta^\psi$. Then, the following supervised learning objective can be established for θ_ψ:

$$\mathcal{L}(\mathbf{x}, \mathbf{x}', \theta_\psi) = \mathcal{L}_{\text{zero}} + \mathcal{L}_{\text{one}} \tag{1}$$

$$\hat{\mathbf{v}}, \hat{\mathbf{v}}' = \text{Ground}(\mathbf{x}, \theta_\psi), \text{Ground}(\mathbf{x}', \theta_\psi), \tag{2}$$

$$\mathcal{L}_{\text{zero}} = \text{Div}_{\text{JS}}\Big(\hat{\mathbf{v}} \odot \mathbb{I}\big(\boldsymbol{t}^{\psi, \underline{\texttt{C}}_i} = 0\big) \,\Big\|\, \hat{\mathbf{v}}' \odot \mathbb{I}\big(\boldsymbol{t}^{\psi, \underline{\texttt{C}}_i} = 0\big)\Big), \tag{3}$$

$$\mathcal{L}_{\text{one}} = \Big(\text{BCE}\big([\hat{v}_p, \hat{v}'_p], [\tfrac{1-\delta_i^\psi}{2}, \tfrac{1+\delta_i^\psi}{2}]\big)\Big) * \text{abs}(\delta_i^\psi), \tag{4}$$

where $\hat{\mathbf{v}}, \hat{\mathbf{v}}' \in [0, 1]^P$ are the predicted ground predicates by applying θ_ψ on all possible object sets from $\mathcal{O}$. $\text{Div}_{\text{JS}}(\cdot|\cdot)$ denotes the Jensen–Shannon divergence [35] and Eq.(3) tries to minimize the distance between $\hat{\mathbf{v}}$ and $\hat{\mathbf{v}}'$ if the indices with zero values in $\boldsymbol{t}^{\psi, \underline{\texttt{C}}}$. $\hat{v}_p, \hat{v}'_p$ denotes p-th ground predicate, where $\mathcal{O}_{\underline{\psi}_p} \subseteq \mathcal{O}_{\underline{\texttt{C}}_i}$. $\text{BCE}(\cdot, \cdot)$ represents the Binary Cross-Entropy, which tries to directly supervise $\hat{\mathbf{v}}$ and $\hat{\mathbf{v}}'$ if $\delta_i^\psi \neq 0$. Intuitively, $\mathcal{L}_{\text{zero}}$ supervises the ground predicates whose values should stay unchanged in a transition (but we don't know their values). $\mathcal{L}_{\text{one}}$, on the other hand, supervises the ground predicates whose values can be derived based on the effect vectors. Since we have Δ^ψ for all lifted actions, the pipeline can be applied to all ground transition pairs in $\mathcal{D}$, resulting in the global loss,

$$\mathcal{L}^{\mathcal{D}}(\theta_\psi) = \sum_{\texttt{C} \in \mathcal{C}} \mathbb{E}_{(\mathbf{x}, \underline{\texttt{C}}, \mathbf{x}') \sim \mathcal{D}_{\texttt{C}}} \mathcal{L}(\mathbf{x}, \mathbf{x}', \theta_\psi), \tag{5}$$

where $\mathcal{D}_{\texttt{C}}$ denotes the distribution of the grounded transition for action $\texttt{C}$ in the dataset $\mathcal{D}$ (See Figure 4 for the examples of two transitions from different actions). Since $\mathcal{L}$ is fully differentiable with respect to θ, given a effector Δ^ψ and the demonstration dataset, we could leverage the general and standard deep learning frameworks [36, 37] to train a neural classifier θ that minimizes the loss $\mathcal{L}$ for any state representation.

B. Neural Loss as Guidance for Symbolic Learning

To obtain all the classifiers $\theta_{\Psi^{\texttt{Var}}}$ for all the typed predicates $\psi \in \Psi^{\texttt{Var}}$, the problem now becomes finding all the lifted effect vectors $\Delta^\psi \in \Delta^{\Psi^{\texttt{Var}}}$. As defined in Definition 3, the lifted effect vectors live in the discrete world with a finite shape, which motivates us to establish some discrete optimization strategies for it. One insight here is that, despite the large space, only a few effect vectors provide reasonable supervision, with unreasonable ones resulting in high classification error on the validation set after the training converges.

[2]Each predicate here can appear at most once in the effect sets, but this doesn't affect the representation capability of the final predicate set.

A motivating example is depicted in Figure 4, where the proposed effect vector assumes the predicate to be the add effects for both `Gaze` and `Grasp`. In the demonstration, `Grasp` closely follows `Gaze`, making the intermediate state shared in the two transition pairs but with one being the next state and the other being the current state. Then, the intermediate state will be labeled as both `True` and `False`, which is unreasonable and results in high classification error. Thus, our symbolic learning aims to efficiently find all "reasonable" effect vectors,

Definition 5 (Symbolic Learning Objective)**.** Let the demonstrations be split into non-overlapping training and validation sets $\mathcal{D} = \mathcal{D}^{\text{train}} \cup \mathcal{D}^{\text{val}}$, the objective of our symbolic learning is to find a subset of effect vectors $\Delta^{*,\Psi^{\texttt{Var}}} \subset \Delta^{\Psi^{\texttt{Var}}}$,

$$\Delta^{*,\Psi^{\texttt{Var}}} = \left\{ \Delta^{\psi} \in \Delta^{\Psi^{\texttt{Var}}} \mid \mathcal{L}^{\mathcal{D}^{\text{val}}}(\theta_{\psi}) \leq \tau \right\}, \tag{6}$$

where θ^{ψ} is learned from $\mathcal{D}^{\text{train}}$ with supervision derived from Δ^{ψ} using Eq. (5), and $\mathcal{L}^{\mathcal{D}^{\text{val}}}$ denotes the validation loss of classifier θ^{ψ} calculated from Eq. (5), and τ is a given threshold.

Inspired by the fact that a predicate's effects are usually sparse among actions, we propose a tree expansion algorithm for efficiently learning $\Delta^{*,\Psi^{\texttt{Var}}}$. Specifically, as shown in Figure 5 (with $M = 4$ actions), the complete effect vector set, $\Delta^{\Psi^{\texttt{Var}}}$, is formulated into a tree-like structure, with each node $\Delta^{\psi_n} \in \Delta^{\Psi^{\texttt{Var}}}, n > 0$ representing an effect vector. The root Δ^0 of the tree is an "all-zero" effect vector, which is not associated with any potential *dynamic* predicate. The nodes in the $l-$th level represent a vector with l non-zero entries. For each non-leaf node in the $l-$th level, its "children" in the $l+1$-th level have the same non-zero entries with one more non-zero entry. A naive exploration in this tree is to enumerate its nodes and train neural classifiers with supervisions from each of them, which is extremely time-consuming due to the large space. To explore the effect tree more efficiently, IVNTR tries to balance the trade-off between *exploration* and *exploitation* [38, 39].

Specifically, each node Δ^{ψ_n} in the tree is additionally associated with a scalar r_t^n, indicating its value for finding $\Delta^{\psi} \in \Delta^{*,\Psi^{\texttt{Var}}}$ if we expand it at the current t-th iteration. In the t-th iteration, we start by selecting a parent node $\text{Par}(\Delta_t^{\psi})$ with the highest current Upper Confidence bounds applied to Trees (UCT) score [39]. Its child, Δ_t^{ψ} that has the current highest value r_t among all the children is proposed for evaluation (index is neglected for simplicity). The evaluation process is defined as the supervised neural learning process in Section IV-A. After the classifier θ^{ψ} converges, we collect its *action-wise* validation loss $\mathbf{L}_t \in \mathbb{R}^M$ by decomposing Eq. (5) for each action $\texttt{C} \in \mathcal{C}, |\mathcal{C}| = M$. The loss information is then used to update the values of all the nodes in the tree, which helps us select and expand the parent node in the $t+1$-th iteration. The tree expansion terminates if all of the existing nodes are fully expanded, or, if the max iteration has been reached.

Clearly, the key to more efficient learning lies in the definition and updating strategy of the node values r_t^n. As the sparsity of predicate effects among actions is indicated by the sparsity of non-zero entries in the effect vector, we try to efficiently explore the entries in the effect vectors that **should not** be zero to optimize Eq. (6). To achieve this, we try to leverage the guidance from the "zero-parts" (Eq. (3)) of $\mathbf{L}_t$. Specifically, after the evaluation of the node Δ_t^{ψ} in the $t-$th iteration with $\mathbf{L}_t$, we use the following equations to update and compute r_{t+1}^n for all the nodes in the tree,

$$\begin{aligned} \mathbf{R}_{t+1} &= \mathbf{R}_t \odot \mathbb{I}(\Delta_t^{\psi} \neq 0) + \frac{(\mathbf{R}_t + \mathbf{L}_t)}{2} \odot \mathbb{I}(\Delta_t^{\psi} = 0). \\ r_{t+1}^n &= \text{Mean}\big(\mathbf{R}_{t+1} \odot \mathbb{I}(\Delta^{\psi_n} = 0)\big), \end{aligned} \tag{7}$$

where $\mathbf{R}_t \in \mathbb{R}^M, \mathbf{R}_0 = \mathbf{0}^M$ is a global value vector that stores the information from historical evaluations. $\mathbb{I}(\Delta_t^{\psi} = 0)$ is a binary vector indicating if an entry in the evaluated node Δ_t^{ψ} equals to zero. Here, we only update $\mathbf{R}_t$ with the "zero-parts" of the loss information $\mathbf{L}_t$, which then helps update the node values. The node values intuitively indicate how likely the loss will decrease in its children where there are fewer zeros in the effect vectors. From Eq. (7), we see that the higher loss in the zero entry indexes of Δ^{ψ_n} contributes to its higher value, encouraging the evaluation to prioritize its children, which are likely to decrease the loss. In Appendix D, we additionally introduce some pruning strategy for more efficient expansion.

Finally, we collect all the effect vectors $\Delta^{\psi_n} \in \Delta^{*,\Psi^{\texttt{Var}}}$ with the associated neural classifier θ_{ψ_n} as the outcomes from the bilevel learning of typed predicates $\Psi^{\texttt{Var}}$. As shown in Figure 3 (a), there could exist multiple different vector-classifier pairs for the same typed predicate $\Psi^{\texttt{Var}}$, which are treated as different predicates in the following predicate selection stage.[3]

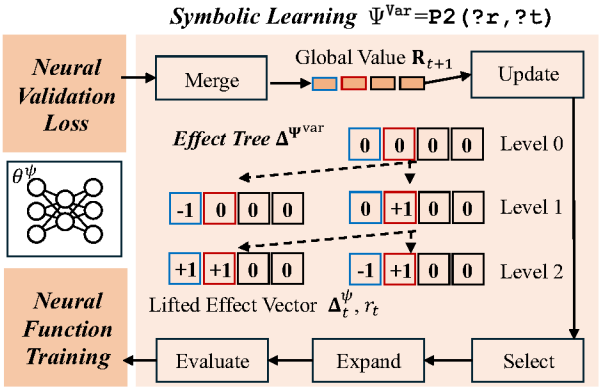

Fig. 5: Detailed symbolic learning process for predicate group `P2(?r,?t)`. With the neural validation loss from the previous iteration, symbolic learning starts by merging the loss into the global value vector (See Eq. (7)). After the node values in the effect tree are updated, we conduct parent node selection and expansion. Among the children nodes, we evaluate the child with the current highest value, via the neural classifer training process described in Figure 4.

C. Predicate Selection

With all possible typed predicates, IVNTR is able to construct the predicate pool $\hat{\Psi}_{\text{dyn}}$ without any classifier predefinition. This strength has made bilevel planning applicable

[3]Following previous work [17], we can also add quantifiers and negations as prefix for each of the invented neural predicates.

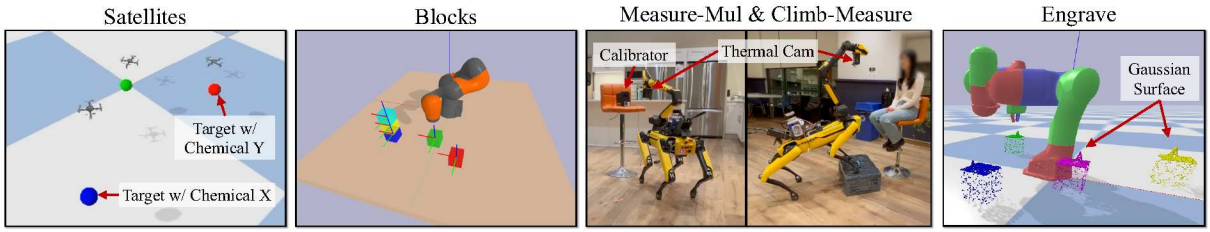

Fig. 6: Visualization of the five domains (excluding Climb-Transport) we have studied in this work. These domains feature various state representations (including the high-dimensional point clouds in the Engrave domain) where our IVNTR can be generally applied.

Domain	Satellites [20]			Blocks [32]			Measure-Mul			Climb-Measure			Climb-Transport			Engrave		
State Space	SE2 ($\mathbb{R}^{3\times5}$)			Vec3 ($\mathbb{R}^{3\times8}$)			SE3 ($\mathbb{R}^{7\times5}$)			SE3 ($\mathbb{R}^{7\times5}$)			SE3 ($\mathbb{R}^{7\times5}$)			PCD ($\mathbb{R}^{1024\times3\times6}$)		
Test Dist	$\mathcal{T}^{\text{train}}$	$\mathcal{T}^{\text{test}}$	$\downarrow$	$\mathcal{T}^{\text{train}}$	$\mathcal{T}^{\text{test}}$	$\downarrow$	$\mathcal{T}^{\text{train}}$	$\mathcal{T}^{\text{test}}$	$\downarrow$	$\mathcal{T}^{\text{train}}$	$\mathcal{T}^{\text{test}}$	$\downarrow$	$\mathcal{T}^{\text{train}}$	$\mathcal{T}^{\text{test}}$	$\downarrow$	$\mathcal{T}^{\text{train}}$	$\mathcal{T}^{\text{test}}$	$\downarrow$
Oracle	100.0	100.0	0.00	100.0	100.0	0.00	100.0	100.0	0.00	90.0	81.6	0.09	91.2	82.0	0.10	100.0	100.0	0.00
IVNTR (Ours)	**99.2**	**93.2**	**0.06**	**100.0**	**82.0**	**0.18**	**90.0**	**88.4**	**0.02**	**91.6**	**65.6**	**0.28**	**79.2**	**53.2**	**0.33**	**98.4**	**79.2**	**0.20**
GNN [40]	74.0	6.0	0.92	82.4	24.0	0.71	2.4	1.2	0.50	20.8	0.7	0.97	48.0	2.0	0.96	0.0	0.0	1.00
Transformer [41]	46.8	1.2	0.97	24.4	7.6	0.69	10.0	2.4	0.76	29.2	0.0	1.00	10.4	0.8	0.92	0.0	0.0	1.00
FOSAE [28]	100.0	34.8	0.65	2.4	0.4	0.83	3.6	1.2	0.67	21.2	0.0	1.00	45.6	0.8	0.98	0.0	0.0	1.00
Grammar [17]	0.0	0.0	1.00	0.0	0.0	1.00	0.0	0.0	1.00	0.0	0.0	1.00	0.0	0.0	1.00	N/A	N/A	N/A
Random	0.0	0.0	1.00	10.6	1.2	0.89	0.0	0.0	1.00	0.0	0.0	1.00	0.0	0.0	1.00	0.0	0.0	1.00

TABLE I: Empirical success rate comparison on six simulated robot planning domains. Among all the methods, IVNTR suffers the least from the generalization challenge due to its relational structure. The scores are obtained from the averaged results over five random seeds.

to complicated and high-dimensional state spaces. Meanwhile, as the predicate classifiers are *relational* and can be seamlessly integrated into an AI planner [34], our approach can naturally achieve compositional generalization.

Yet, not all of the predicates in $\hat{\Psi}_{\text{dyn}}$ are favorable for planning. Therefore, we next try to select a subset $\tilde{\Psi}_{\text{dyn}} \subset \hat{\Psi}_{\text{dyn}}$ that minimizes the score function $J(\tilde{\Psi}_{\text{dyn}}, \mathcal{D})$ [17]. Specifically, with a set of candidate predicates $\tilde{\Psi} = \{\Psi_{\text{G}}, \Psi_{\text{sta}}, \tilde{\Psi}_{\text{dyn}}\}$, we start by grounding all the states $\mathbf{x}$ in the demonstration $\mathcal{D}$, which forms a ground atom dataset. Since we already have the effect vector for each of the predicates in $\tilde{\Psi}_{\text{dyn}}$, the effect sets $(\tilde{\texttt{Eff}}^+, \tilde{\texttt{Eff}}^- \subset \tilde{\Psi}_{\text{dyn}})$ for each operator $\tilde{\texttt{Op}}^{\text{c}}$ can be easily obtained. Then, the precondition set for each operator can be calculated using an intersection strategy [32]. The learned operator set $\tilde{\texttt{Op}}$ is then applied to the ground atom dataset to generate plan *skeletons* $\tilde{\pi}$ for each task, which are compared with the demonstration plan *skeletons* $\bar{\pi}$ for objective calculation [17]. The objective is finally minimized by running a hill-climbing search over $\hat{\Psi}_{\text{dyn}}$, resulting in the desired predicate set Ψ_{dyn}. With the complete set, we are now able to learn the planning *abstractions* (operators and samplers) using standard pipelines [20, 21, 32] as shown in Figure 3 (a). For a more detailed explanation to operator and sampler learning, please see Appendix Appendix K.

Note that before the predicate selection stage, all of the predicates' neural classifiers have been pre-trained using our IVNTR framework, which has avoided the challenge of using the planning objective for learning but still achieved a powerful and adaptive model optimized for planning.

V. Experiments

A. Implementation Details

System and Hardware: All methods are evaluated on a single NVIDIA A100 GPU and an AMD EPYC 7543 32-Core CPU to ensure fairness. Training is conducted on the same hardware as evaluation, with domain-specific details provided in Appendix E. Real-robot experiments are performed using the Boston Dynamics Spot robot equipped with an arm.

Baselines: Since we do not assume access to the complete predicate set, most existing bilevel planning approaches [19, 20, 32] are inapplicable. We attempted the grammar-based approach [17], but it failed to optimize the planning objective in most domains (see Appendix J). Thus, IVNTR is primarily compared to relational neural policy learning methods [28, 40, 41]. Following prior works [16, 17, 20, 32], baselines are trained using standard behavior cloning pipelines and evaluated with a shooting strategy; see Appendix F for details.

Domains: We evaluate the methods across six diverse robot planning domains with varying state representations, as visualized in Figure 6 and summarized in Table I. Below, we provide a high-level overview of these domains. For more implementation details, please refer to Appendix E:

- *Satellites* comes from prior work [20], which involves a group of satellites collaborating to capture sensor readings from targets. States comprise SE2 poses and object attributes. Training scenarios ($\mathcal{T}^{\text{train}}$) include 2 satellites and 2 targets, while test scenarios ($\mathcal{T}^{\text{test}}$) have 3 of each.
- *Blocks:* Inspired by [32], this domain tasks a robot with manipulating 3D blocks to form goal towers. Unlike vanilla Blocks World, the goals here involve "packing" pairs of blocks into two-level towers. $\mathcal{T}^{\text{train}}$ includes 4–5

Planner	Seed/Task	Climb-Measure T0	T1	T2	T3	T4	T5	Mean	Avg.	Climb-Transport T0	T1	T2	T3	T4	T5	Mean	Avg.
Oracle (Human)	S0	0.0	1.0	1.0	1.0	0.5	1.0	0.750	0.833	0.5	0.0	1.0	1.0	1.0	1.0	0.750	0.592
	S1	1.0	1.0	1.0	0.5	1.0	0.5	0.833		1.0	1.0	0.33	0.5	0.5	0.5	0.638	
	S2	1.0	1.0	1.0	1.0	1.0	0.5	0.917		0.5	0.33	0.5	0	0.5	0.5	0.388	
IVNTR (Learned)	S0	1.0	1.0	0.5	0.0	1.0	1.0	0.750	0.778	0.5	1.0	0	0.5	1.0	0.33	0.555	0.546
	S1	1.0	1.0	1.0	1.0	1.0	0.0	0.833		0.33	0.5	1.0	0.33	0	0.5	0.443	
	S2	0.5	1.0	1.0	1.0	0.0	1.0	0.750		0.5	0.5	1.0	1.0	0.5	0.33	0.638	

TABLE II: Success rate comparison of the two planners on the real robot planning tasks sampled from $\mathcal{T}^{\text{test}}$. "Oracle" denotes bilevel planners exhaustively engineered by a human expert. IVNTR is learned from the demonstrations collected in the simulated environment. For each domain, we have tested six tasks (T0∼T5) sampled from three random seeds (S0∼S2). For each planner in each task, we run it at most three times and record the averaged task success rate. Our framework has achieved comparable real robot performance with the Oracle.

Domains	Blocks			Climb-Measure		
Metric	$\mathcal{T}^{\text{train}}$	$\mathcal{T}^{\text{test}}$	$J(\cdot)$ $(\times 10^5)$	$\mathcal{T}^{\text{train}}$	$\mathcal{T}^{\text{test}}$	$J(\cdot)$ $(\times 10^5)$
GT-Vectors	80.0	62.8	121.59	90.4	61.2	2.718
IVNTR	**100.0**	**82.0**	**56.77**	**91.6**	**65.6**	**2.481**

TABLE III: Comparison between the predicates learned with ground-truth (GT)-vectors and using our IVNTR framework. We report the task success rate and the final planning objective.

Domains	Satellites			Measure-Mul		
Sampling with θ^{ψ}	✓	✗	↓	✓	✗	↓
$\mathcal{T}^{\text{train}}$	**99.2**	74.0	0.254	**90.0**	2.8	0.969
$\mathcal{T}^{\text{test}}$	**93.2**	16.8	0.820	**88.4**	1.4	0.984

TABLE IV: Comparison between sampling with and without the invented predicate classifiers. Without using the predicates as step-wise success indicator, the performance drops significantly.

blocks, while $\mathcal{T}^{\text{test}}$ features 6–7 blocks.

- *Measure-Mul:* In this new domain inspired by Satellites, a Spot robot calibrates a thermal camera by aligning it with a calibrator before measuring body temperatures of multiple human targets. States include 6D poses of the robot and the targets. Training distributions ($\mathcal{T}^{\text{train}}$) have 2–3 humans, while test distributions ($\mathcal{T}^{\text{test}}$) include 4.
- *Climb-Measure* is similar to Measure-Mul but with added complexity: calibrators and human targets may be placed at high, initially unreachable locations. The Spot robot must arrange platforms and climb onto them to reach targets. Training ($\mathcal{T}^{\text{train}}$) includes 0–1 platforms, while testing ($\mathcal{T}^{\text{test}}$) requires planning with 2 platforms.
- *Climb-Transport:* Introduced in Figure 2, this domain requires the Spot robot to arrange platforms to grasp a high-placed target, then transport it into a container. Training setups ($\mathcal{T}^{\text{train}}$) feature 0–1 platforms, while testing ($\mathcal{T}^{\text{test}}$) involves 2 platforms.
- *Engrave* features high-dimensional state spaces represented as object-centric point clouds. Similar to Blocks, the goal is to "pack" blocks. However, blocks start with one irregular Gaussian surface that must be "engraved" and "rotated" to create a matching fit. Training distributions ($\mathcal{T}^{\text{train}}$) include 3–4 blocks, while testing ($\mathcal{T}^{\text{test}}$) has 5–6.

Experiment Setup: For each domain, we manually designed an oracle bilevel planner (Oracle) to collect training demonstrations. We report averaged results over five random seeds for all six domains. For each seed in Satellites, Blocks, Measure-Mul, and Engrave, we have collected 500 demonstrations. For Climb-Measure and Climb-Transport, 2000 demonstrations were collected per seed. During test, each seed in each domain includes 50 in-domain tasks ($T \sim \mathcal{T}^{\text{train}}$) and 50 generalization tasks ($T \sim \mathcal{T}^{\text{test}}$). We report the success rate within the same maximum planning time for all the methods. For real-world Climb-Measure and Climb-Transport, a shared map was recorded using Spot's default graph_nav service for simulation and localization. Each domain was tested on 3 random seeds, each with 6 generalized tasks. For manipulation-based actions, we have utilized an off-the-shelf segment anything model (SAM) [42, 43] for computing the grasping pixel.

B. Empirical Results

Simulated Planning Domains: The empirical comparison across the six simulated domains is presented in Table I. Alongside the averaged success rates, we report the performance drop percentage during generalization. IVNTR consistently outperforms all baselines in both $\mathcal{T}^{\text{train}}$ and $\mathcal{T}^{\text{test}}$ across all domains. For complex state representations such as SE3 and PointClouds (PCD), none of the baselines achieve a success rate above 5% on generalized tasks, while IVNTR stably solves over 50% by virtue of its relational structure. In Appendix L, we further demonstrate the potential of IVNTR to abstract high-dimensional RGB image states into symbolic predicates.

Real Robot Planning Tasks: All real robot tasks are sampled from $\mathcal{T}^{\text{test}}$, making IVNTR the only applicable approach. To benchmark performance, a human expert has exhaustively engineered oracle planners for the real robot in the two domains. Each approach attempts each task up to three times, with the average success rate reported in Table II. Despite the simulation-to-reality gap, IVNTR successfully generalizes to held-out tasks, achieving results comparable to the oracle planner. Most real-world deployment failures stem from perception and localization errors, with examples shown in Appendix G.

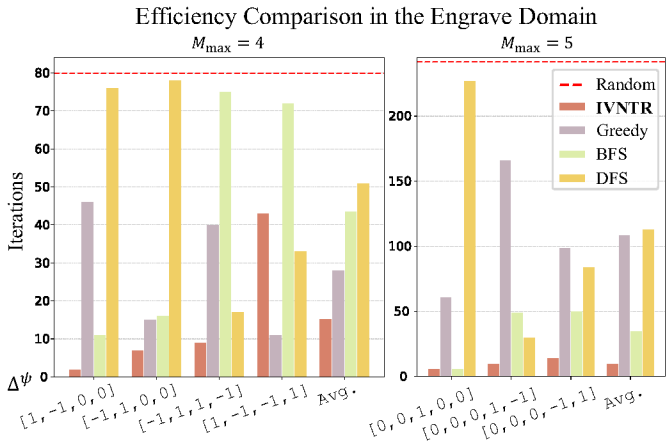

Fig. 7: Comparison between IVNTR with other search strategies in the Engrave domain. We report the number of iterations for each of the algorithm to find the reasonable effect vectors (different predicate could have different maximum search space $M_{\max}$). IVNTR has demonstrated the highest efficiency in finding the desired vectors.

C. Ablation Studies

Comparison to Ground-Truth Effect Vectors: A notable strength of IVNTR is its ability to discover non-ground-truth (GT) predicates. In Table III, we replaced our tree expansion with oracle-derived GT effect vectors, where the performance on the Blocks and Climb-Measure domains are reported. Interestingly, IVNTR minimizes the planning objective more effectively than GT vectors, resulting in its higher accuracy. This outcome highlights the advantage of exploring better high-level abstractions beyond human-engineered ones.

Comparison to Other Search Algorithms: To evaluate the efficiency of our neural-informed tree expansion algorithm, we compared it with alternative search strategies: a greedy approach (Greedy) that flips the zero entry with the highest current loss, breadth-first (BFS) and depth-first (DFS) searches. For the Engrave domain, Figure 7 shows the number of iterations required to find each reasonable effect vector using these methods. Compared to uninformed methods, our IVNTR framework is at least $2\times$ more efficient. The greedy approach exhibits high variance and is generally less reliable. We present comparisons in more domains in Appendix M.

Comparison to Pure High-level Planning: As discussed in Section III, predicates not only enable compositional generalization through planning operators but also serve as indicators of sampler failures in low-level states. To assess the importance of our invented predicates for reliable low-level sampling, we disabled bilevel planning and followed the high-level plan greedily, ignoring predicate-based checks for sampler success. As shown in Table IV, this approach results in performance drops of up to 98.4%, underscoring the critical role of predicates as indicators of low-level state validity.

D. Interpreting Invented Predicates

Predicates play a key role in defining the preconditions and effects of operators, specifying the order of ground actions to complete a task. In Table V, we display part of the precondition and effect sets for high-level actions in the Climb-Transport domain, where invented predicates capture logical relationships among actions. For example, the `Drop` action requires $\texttt{P2}_1$ and $\texttt{P2}_2$, which are the add effects of `Gaze`, `MTGaze`, and `WalkOn`. Similarly, the `Pick` action depends on $\texttt{P1}_1$, the delete effect of `Grasp`, enforcing all `Pick` actions to precede `Grasp`. These relational constraints over objects enable the generation of long-horizon plans that generalize to unseen object compositions. The complete operators are detailed in Appendix H. We also provide visualizations of how predicates act as success indicators to filter out sampler failures in Appendix I.

VI. Related Works

A. Learning Abstractions for Planning

Learning abstractions is essential for reducing the complexity of long-horizon planning in high-dimensional domains. Traditional approaches, such as hierarchical task networks (HTN) [44], heavily rely on hand-designed abstractions. Recent advances have shifted towards data-driven approaches that automatically discover useful abstractions from interactions [18, 30, 45–47] or demonstrations [32, 48–50]. However, these methods struggle to generalize beyond the training environments [21]. Foundation models, such as large language models (LLMs) and vision-language models (VLMs), have been explored for high-level planning with minimal or no demonstrations [21, 25, 51–57]. While these models leverage commonsense knowledge for efficient plan generation, two challenges remain: (1) High-level plans from LLMs [25, 53–55] are difficult to reliably refine in the low-level space [21]. (2) VLM-based methods [13, 21, 52, 57] struggle in domains where images cannot fully capture the state space.

B. Task and Motion Planning

To address these challenges, task and motion planning (TAMP) integrates high-level symbolic planning with low-level motion generation. Traditional TAMP methods [33, 58] rely on manually designed planners [59, 60]. These frameworks inherently support compositional generalization due to their relational structure. In addition, the coupling between high-level and low-level planning can handle failures at either level effectively. However, traditional TAMP requires substantial human effort. Recent advances integrate learning into TAMP [13, 17, 20, 21, 32, 57, 61], forming bilevel planning frameworks. These approaches combine the strengths of TAMP with the scalability of machine learning models. However, most bilevel planners rely on manually engineered state abstractions (predicates), limiting their scalability and flexibility.

C. Predicate Invention for Planning

To automate predicate generation for planning, various approaches have been proposed [14, 17, 21, 24, 25, 27–30]. The most direct approaches [24, 25] rely on domain knowledge, such as human-provided labels [24] or LLM-based oracles [25]. To *invent* predicates, earlier approaches derive "easy" step-wise objectives, such as reconstruction [27–29] or bisimulation [30].

Predicates	$\texttt{P1}_1(?\texttt{r})$	$\texttt{P2}_1(?\texttt{r},?\texttt{t})$	$\texttt{P2}_2(?\texttt{r},?\texttt{t})$	$\texttt{In}(?\texttt{t},?\texttt{t})$
Grasp	Pre | $\texttt{Eff}^-$	Pre	Pre	
Gaze		Pre	$\texttt{Eff}^+$	
MAOff	Pre			
MAOn		Pre	Pre	
MTGaze		$\texttt{Eff}^+$		
WalkOn		$\texttt{Eff}^+$		
MTPlace	Pre			
MTReach				
Pick	Pre			
Drop		Pre	Pre	$\texttt{Eff}^+$

TABLE V: The preconditions, add effects, and delete effects for each of the actions (the variables are neglected for simplicity) with (part of) the invented predicates in the Climb-Transport Domain. `MA` is for `MoveAway` and `MT` for `MoveTo`. The invented predicates have specified some logical constrains over the order of actions.

Among these, LatPlan (FOSAE) [28] learns relational neural abstractions for images by reconstructing states and identifying action spaces for planning, which is the closest work to IVNTR. However, its implicit predicates are not optimized for efficient planning, limiting its applicability to domains with only nullary actions. Recent approaches [17, 21] address this by learning abstractions tailored to fast planners [34]. However, these methods struggle to *learn* predicate classifiers due to non-differentiable objectives. Consequently, they often rely on pre-defined predicate candidates from program synthesis [17] or foundation models [21], which constrains their applicability in more sophisticated and high-dimensional state spaces. Our approach is motivated by the bilevel planning framework [17] but eliminates the need for pre-defining the candidates. Instead, we can learn adaptive neural classifiers for different domains, enabling more flexible and scalable learning based planning.

VII. Limitations and Future Works

In this work, we introduced IVNTR, a bilevel learning framework that invents neural classifiers as relational planning predicates. These predicates enable the learning of relational bilevel planners capable of generating long-horizon plans for unseen object compositions. At the neural level, IVNTR leverages the high-level effects of predicates across actions to provide step-wise supervisions on transition pairs. At the symbolic level, IVNTR captures the sparsity of effects through an informed tree expansion algorithm. By adopting neural classifiers, IVNTR adapts to diverse robot planning domains with continuous and high-dimensional state representations. Additionally, we deployed IVNTR on a mobile manipulator, demonstrating its ability to achieve compositional generalization over objects and actions in long-horizon mobile manipulation tasks.

IVNTR has several limitations that we leave for future work: (1) IVNTR can only invent *dynamic* predicates. *Static* predicates are still assumed as domain-level prior knowledge. (2) Following previous works [19], we have assumed the sparsity of effects; discussions about the general cases can be found in Appendix B. (3) Since IVNTR trains different neural networks in each iteration, the learning time could be prolonged in domains with extreme complexity. Currently, we parallelize the invention of different predicate groups on multiple GPUs for efficiency (see Appendix E). (4) Neural predicates with quantifiers are less reliable due to the prediction errors in the classifiers. Future work could explore probabilistic symbolic planners [62] or hybrid declarative-imperative representations [63] to plan with an inaccurate model. (5) Since the predicates are neural networks, it is hard to interpret their physical meanings. It would also be intriguing for future approaches to make them directly human readable.

Acknowledgment

We acknowledge the support of the Air Force Research Laboratory (AFRL), DARPA, under agreement number FA8750-23-2-1015. We also acknowledge Defence Science and Technology Agency (DSTA) under contract #DST000EC124000205. This work used Bridges-2 at PSC through allocation cis220039p from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program which is supported by NSF grants #2138259, #2138286, #2138307, #2137603, and #213296. We express sincere gratitude to Qinglin Feng for her valuable time in supporting our real-robot experiments and for her intelligence in motivating our Climb-Transport domain. The authors would also like to express sincere gratitude to Jiayuan Mao (MIT), Nishanth Kumar (MIT), Prof. Katia Sycara (CMU), and Prof. Pradeep Ravikumar (CMU) for their valuable feedback, discussions, and suggestions on the early stages of this work. Finally, the authors wish to thank our Spot robot, Spotless, for being reliable throughout our real-world experiments.

References

[1] Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. *arXiv preprint arXiv:2310.17596*, 2023.

[2] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023.

[3] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems (RSS)*, 2023.

[4] Dian Wang, Stephen Hart, David Surovik, Tarik Kelestemur, Haojie Huang, Haibo Zhao, Mark Yeatman, Jiuguang Wang, Robin Walters, and Robert Platt. Equivariant diffusion policy. *arXiv preprint arXiv:2407.01812*, 2024.

[5] Jingyun Yang, Zi-ang Cao, Congyue Deng, Rika Antonova, Shuran Song, and Jeannette Bohg. Equibot: Sim (3)-equivariant diffusion policy for generalizable and data efficient learning. *arXiv preprint arXiv:2407.01479*, 2024.

[6] Jiayuan Mao, Tomás Lozano-Pérez, Josh Tenenbaum, and Leslie Kaelbling. What Planning Problems Can A

Relational Neural Network Solve? In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2024.

[7] Bowen Li, Zhaoyu Li, Qiwei Du, Jinqi Luo, Wenshan Wang, Yaqi Xie, Simon Stepputtis, Chen Wang, Katia P Sycara, Pradeep Kumar Ravikumar, et al. LogiCity: Advancing Neuro-Symbolic AI with Abstract Urban Simulation. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 69840–69864, 2024.

[8] Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. In *AI&M*, 2006.

[9] David Abel, Dilip Arumugam, Lucas Lehnert, and Michael Littman. State abstractions for lifelong reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2018.

[10] George Konidaris, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 16:215–289, 2018.

[11] Lionel Wong, Jiayuan Mao, Pratyusha Sharma, Zachary S Siegel, Jiahai Feng, Noa Korneev, Joshua B Tenenbaum, and Jacob Andreas. Learning Grounded Action Abstractions From Language. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[12] Aidan Curtis, Tom Silver, Joshua B Tenenbaum, Tomás Lozano-Pérez, and Leslie Kaelbling. Discovering State And Action Abstractions For Generalized Task And Motion Planning. In *Proceedings of The AAAI Conference On Artificial Intelligence (AAAI)*, number 5, pages 5377–5384, 2022.

[13] Zhutian Yang, Caelan Garrett, Dieter Fox, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Guiding Long-Horizon Task and Motion Planning with Vision Language Models, 2024.

[14] Naman Shah, Jayesh Nagpal, Pulkit Verma, and Siddharth Srivastava. From Reals to Logic and Back: Inventing Symbolic Vocabularies, Actions and Models for Planning from Raw Data. *arXiv preprint arXiv:2402.11871*, 2024.

[15] Tom Silver, Rohan Chitnis, Joshua Tenenbaum, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Learning Symbolic Operators for Task and Motion Planning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3182–3189, 2021.

[16] Tom Silver, Ashay Athalye, Joshua B. Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Learning Neuro-Symbolic Skills for Bilevel Planning. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2022.

[17] Tom Silver, Rohan Chitnis, Nishanth Kumar, Willie McClinton, Tomás Lozano-Pérez, Leslie Kaelbling, and Joshua B Tenenbaum. Predicate Invention for Bilevel Planning. In *Proceedings of The AAAI Conference on Artificial Intelligence (AAAI)*, volume 37, pages 12120–12129, 2023.

[18] Rohan Chitnis, Tom Silver, Joshua B Tenenbaum, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. GLIB: Efficient Exploration for Relational Model-Based Reinforcement Learning via Goal-Literal Babbling. In *Proceedings of The AAAI Conference on Artificial Intelligence (AAAI)*, volume 35, pages 11782–11791, 2021.

[19] Nishanth Kumar, Tom Silver, Willie McClinton, Linfeng Zhao, Stephen Proulx, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Jennifer Barry. Practice Makes Perfect: Planning To Learn Skill Parameter Policies. In *Proceedings of the Robotics: Science And Systems (RSS)*, 2024.

[20] Nishanth Kumar, Willie McClinton, Rohan Chitnis, Tom Silver, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Learning Efficient Abstract Planning Models That Choose What to Predict. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2023.

[21] Yichao Liang, Nishanth Kumar, Hao Tang, Adrian Weller, Joshua B Tenenbaum, Tom Silver, João F Henriques, and Kevin Ellis. VisualPredicator: Learning Abstract World Models With Neuro-Symbolic Predicates For Robot Planning, 2024.

[22] Johannes Kulick, Marc Toussaint, Tobias Lang, and Manuel Lopes. Active learning for teaching a robot grounded relational symbols. In *IJCAI*, pages 1451–1457. Citeseer, 2013.

[23] George Konidaris, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research (JAIR)*, 2018.

[24] Amber Li and Tom Silver. Embodied Active Learning of Relational State Abstractions for Bilevel Planning. In *Proceedings of the Conference on Lifelong Learning Agents (CoLLAs)*, pages 358–375, 2023.

[25] Muzhi Han, Yifeng Zhu, Song-Chun Zhu, Ying Nian Wu, and Yuke Zhu. InterPreT: Interactive Predicate Learning from Language Feedback for Generalizable Task Planning. *arXiv preprint arXiv:2405.19758*, 2024.

[26] Toki Migimatsu and Jeannette Bohg. Grounding Predicates through Actions. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 3498–3504. IEEE, 2022.

[27] Masataro Asai and Alex Fukunaga. Classical Planning in Deep Latent Space: Bridging the Subsymbolic-Symbolic Boundary. In *Proceedings of The AAAI Conference on Artificial Intelligence (AAAI)*, volume 32, 2018.

[28] Masataro Asai. Unsupervised Grounding of Plannable First-Order Logic Representation From Images. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, volume 29, pages 583–591, 2019.

[29] Masataro Asai and Christian Muise. Learning Neural-Dymbolic Descriptive Planning Models via Cube-Space Priors: the Voyage Home (to STRIPS). In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 2676–2682, 2021.

[30] Philippe Hansen-Estruch, Amy Zhang, Ashvin Nair, Patrick Yin, and Sergey Levine. Bisimulation Makes Analogies in Goal-conditioned Reinforcement Learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 8407–8426, 2022.

[31] Ashay Athalye, Nishanth Kumar, Tom Silver, Yichao Liang, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Predicate Invention from Pixels via Pretrained Vision-Language Models. *arXiv preprint arXiv:2501.00296*, 2024.

[32] Rohan Chitnis, Tom Silver, Joshua B. Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Learning Neuro-Symbolic Relational Transition Models for Bilevel Planning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4166–4173, 2022.

[33] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated Task and Motion Planning. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1):265–293, 2021.

[34] Malte Helmert. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[35] Jianhua Lin. Divergence Measures based on the Shannon Entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991.

[36] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[37] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning Representations by Back-propagating Errors. *nature*, 323(6088):533–536, 1986.

[38] Rémi Coulom. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In *Proceedings of the International Conference on Computers and Games (ICCG)*, pages 72–83, 2006.

[39] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the Game of Go without Human Knowledge. *nature*, 550(7676):354–359, 2017.

[40] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational Inductive Biases, Deep Learning, and Graph Networks. *arXiv preprint arXiv:1806.01261*, 2018.

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All You Need. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

[42] Luca Medeiros. Language Segment-Anything. https://github.com/luca-medeiros/lang-segment-anything, 2024.

[43] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment Anything in Images and Videos. *arXiv preprint arXiv:2408.00714*, 2024.

[44] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical Task and Motion Planning in the Now. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 1470–1477. IEEE, 2011.

[45] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay Policy Learning: Solving Long-Horizon Tasks via Imitation and Reinforcement Learning. In *Proceedings of the Conference on Robot Learning (CoRL)*, pages 1025–1037, 2020.

[46] Soroush Nasiriany, Huihan Liu, and Yuke Zhu. Augmenting Reinforcement Learning with Behavior Primitives for Diverse Manipulation Tasks. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 7477–7484, 2022.

[47] Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li, and Denny Zhou. Neural Logic Machines. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–10, 2019.

[48] Mohit Sharma, Arjun Sharma, Nicholas Rhinehart, and Kris M Kitani. Directed-Info GAIL: Learning Hierarchical Policies from Unsegmented Demonstrations using Directed Information. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[49] Thomas Kipf, Yujia Li, Hanjun Dai, Vinicius Zambaldi, Alvaro Sanchez-Gonzalez, Edward Grefenstette, Pushmeet Kohli, and Peter Battaglia. Compile: Compositional Imitation Learning and Execution. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3418–3428, 2019.

[50] Jiayuan Mao, Tomás Lozano-Pérez, Josh Tenenbaum, and Leslie Kaelbling. PDSketch: Integrated Domain Programming, Learning, and Planning. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 36972–36984, 2022.

[51] Weiyu Liu, Neil Nie, Ruohan Zhang, Jiayuan Mao, and Jiajun Wu. BLADE: Learning Compositional Behaviors from Demonstration and Language. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2024.

[52] Xiaolin Fang, Bo-Ruei Huang, Jiayuan Mao, Jasmine Shone, Joshua B Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Keypoint Abstraction using Large Models for Object-Relative Imitation Learning. *arXiv preprint arXiv:2410.23254*, 2024.

[53] Tom Silver, Soham Dan, Kavitha Srinivas, Joshua B Tenenbaum, Leslie Kaelbling, and Michael Katz. Generalized Planning in PDDL Domains with Pretrained Large Language Models. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 38, pages 20256–20264, 2024.

[54] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-Thought Prompting Elicits Reasoning in Large

Language Models. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[55] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models. In *Proceedings of the Conference on Robot Learning (CoRL)*, pages 540–562, 2023.

[56] Yingdong Hu, Fanqi Lin, Tong Zhang, Li Yi, and Yang Gao. Look Before You Leap: Unveiling the Power of GPT-4v in Robotic Vision-Language Planning. *arXiv preprint arXiv:2311.17842*, 2023.

[57] Nishanth Kumar, Fabio Ramos, Dieter Fox, and Caelan Reed Garrett. Open-World Task and Motion Planning via Vision-Language Model Inferred Constraints. In *CoRL Workshop on Language and Robot Learning: Language as an Interface*, 2024.

[58] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Pddlstream: Integrating Symbolic Planners and Blackbox Samplers via Optimistic Adaptive Planning. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, volume 30, pages 440–448, 2020.

[59] Drew McDermott, Malik Ghallab, Adele E. Howe, Craig A. Knoblock, Ashwin Ram, Manuela M. Veloso, Daniel S. Weld, and David E. Wilkins. PDDL-the Planning Domain Definition Language. 1998.

[60] Sertac Karaman, Matthew R Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime Motion Planning using the RRT. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 1478–1483, 2011.

[61] Nicolas Bougie and Ryutaro Ichise. Skill-based curiosity for intrinsically motivated reinforcement learning. *Machine Learning*, 109, 2020.

[62] Håkan LS Younes and Michael L Littman. PPDDL1. 0: An extension to PDDL for expressing planning domains with probabilistic effects. *Techn. Rep. CMU-CS-04-162*, 2:99, 2004.

[63] Jiayuan Mao, Joshua B Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Hybrid Declarative-Imperative Representations for Hybrid Discrete-Continuous Decision-Making. In *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR)*.

[64] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3D Deep Learning with PyTorch3D. *arXiv:2007.08501*, 2020.

[65] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017.

[66] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.