Few-Shot Neuro-Symbolic Imitation Learning for Long-Horizon Planning and Acting

Pierrick Lorang
Human-Robot Interaction Lab
Tufts University, United States
& Austrian Institute of Technology, Austria
first.last@tufts.edu

Hong Lu
Human-Robot Interaction Lab
Tufts University, United States
first.last663424@tufts.edu

Johannes Huemer
Austrian Institute of Technology, Austria
first.last@ait.ac.at

Patrik Zips
Austrian Institute of Technology, Austria
first.last@ait.ac.at

Matthias Scheutz
Human-Robot Interaction Lab
Tufts University, United States
first.last@tufts.edu

Abstract: Imitation learning enables intelligent systems to acquire complex behaviors with minimal supervision. However, existing methods often focus on short-horizon skills, require large datasets, and struggle to solve long-horizon tasks or generalize across task variations and distribution shifts. We propose a novel neuro-symbolic framework that jointly learns continuous control policies and symbolic domain abstractions from a few skill demonstrations. Our method abstracts high-level task structures into a graph, discovers symbolic rules via an Answer Set Programming solver, and trains low-level controllers using diffusion policy imitation learning. A high-level oracle filters task-relevant information to focus each controller on a minimal observation and action space. Our graph-based neuro-symbolic framework enables capturing complex state transitions, including non-spatial and temporal relations, that data-driven learning or clustering techniques often fail to discover in limited demonstration datasets. We validate our approach in six domains that involve four robotic arms, Stacking, Kitchen, Assembly, and Towers of Hanoi environments, and a distinct Automated Forklift domain with two environments. The results demonstrate high data efficiency with as few as five skill demonstrations, strong zero- and few-shot generalizations, and interpretable decision making. A video of our results is available at this link.

Keywords: Neuro-symbolic, Imitation Learning, Task and Motion Planning, Symbolic Planning, Skill Learning, Human-Robot Interaction

1 Introduction

Teaching robots complex tasks remains a central challenge in robotics and artificial intelligence. Imitation learning has emerged as a prominent solution that enables robots to acquire behaviors from demonstrations [1–5]. However, it typically focuses on short-horizon skills, struggles with distribution shifts over time [6, 7], and generalizes poorly to novel situations.

These limitations are magnified in long-horizon tasks, where successful behavior demands not just precise low-level control but strategic high-level planning. Humans excel by abstracting problems into symbolic representations, facilitating reasoning and generalization. Hierarchical approaches such as Task and Motion Planning (TAMP) [8–10] leverage a similar divide, but traditionally rely on manually crafted symbolic models, making them brittle and laborious to adapt. Previous work has sought to learn either symbolic models [11–24] or low-level controllers [25–30, 30–35] independently. Some more recent work proposes learning both layers from demonstrations [31]. However,

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

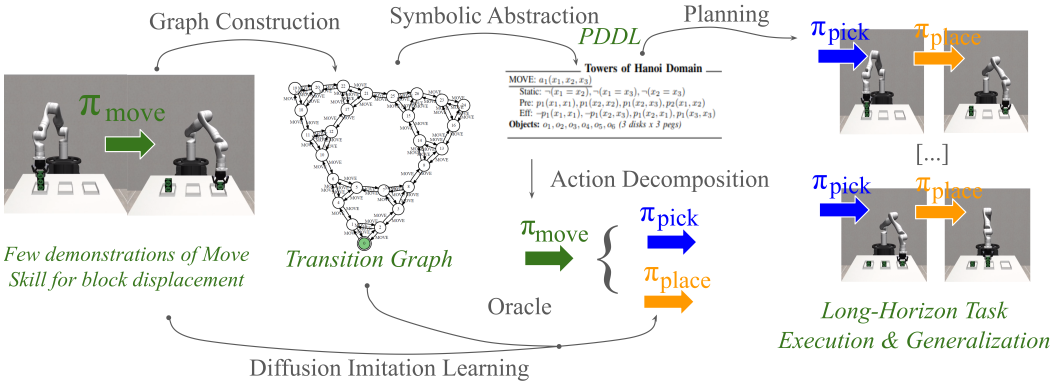

Graph Construction
Symbolic Abstraction
Planning
PDDL
Towers of Hanoi Domain
MOVE: $a_1(x_1, x_2, x_3)$
Static: $\neg(x_1 = x_2), \neg(x_1 = x_3), \neg(x_2 = x_3)$
Pre: $p_1(x_1, x_1), p_1(x_2, x_2), p_1(x_2, x_3), p_2(x_1, x_2)$
Eff: $\neg p_1(x_1, x_1), \neg p_1(x_2, x_3), p_1(x_2, x_1), p_1(x_3, x_3)$
Objects: $o_1, o_2, o_3, o_4, o_5, o_6$ (3 disks x 3 pegs)
π_{move}
Few demonstrations of Move Skill for block displacement
Transition Graph
Action Decomposition
π_{move}
π_{pick}
π_{place}
Oracle
π_{pick}
π_{place}
[...]
π_{pick}
π_{place}
Long-Horizon Task Execution & Generalization
Diffusion Imitation Learning

Figure 1: Our neuro-symbolic framework integrates graph construction, symbolic abstraction, planning, action decomposition, imitation learning and space filtering. Starting with just a few skill demonstrations (left), we construct a transition graph capturing transitions (edges), which are skill transitions between two high-level states (black-box nodes). This graph enables automatic PDDL model extraction via an ASP solver, which powers high-level planning. The oracle decomposes complex tasks into primitive action step policies executed by learned diffusion-based controllers, allowing generalization to novel long-horizon tasks while requiring minimal training data.

these methods typically assume access to symbolic abstractions *a priori*: either through manually designed symbolic state representations, known predicates and object types, or predefined mapping functions from continuous states to symbolic facts.

To our knowledge, no prior work jointly learns low-level control policies and high-level planning models from a few demonstrations without relying on predefined symbolic states, predicates, or lexicon. In our approach, each demonstration is modeled as a transition between two high-level environment states, represented as black-box nodes with visual snapshots. Identical states can be matched by a human based solely on visual comparison, requiring no expertise in symbolic structures or planning. The resulting skills and nodes form a domain structure graph, which is automatically constructed and passed to an ASP solver to yield a PDDL-form symbolic domain. This symbolic representation supports learning data-efficient controls, as well as planning during execution.

Our approach can model object relationships beyond static (position based) spatial predicates, as typically done by clustering approaches [23], and allows for instance time-dependent transitions abstraction—such as waiting for food to cook—by representing them as temporal edges in the graph. A symbolic abstraction is then automatically extracted by using an Answer Set Programming (ASP) solver [13, 14] to discover a Planning Domain Definition Language (PDDL) [36] model consistent with the observed transitions.

Contributions. We present the first *neuro-symbolic imitation learning* framework that jointly learns low-level control policies and high-level symbolic abstractions from few raw demonstrations, without predefined states, predicates, lexicons, or domain knowledge. From these abstractions, an oracle automatically segments demonstrations, filters irrelevant information, and trains neural controllers for each operator. Our method provides: (1) data-efficient learning from few demonstrations, (2) robust generalization to out-of-distribution tasks, and (3) scalable continual learning. We validate it across six domains—Stacking, Kitchen, Assembly, Towers of Hanoi, and two forklift tasks—demonstrating strong data efficiency, broad applicability, robust generalization, and interpretable symbolic plans.

2 Related Work

Prior work on bi-level architectures often assumes access to symbolic representations. Some approaches learn action models from symbolic traces [12, 17, 21, 24, 37], relying on predefined grammars and predicates. Others guide low-level RL with symbolic domains [30, 33, 38–43], but still require manually crafted symbolic states.

Learning skill sequences from demonstrations [44–49] enables long-horizon imitation but lacks modularity, explainability, and data efficiency compared to symbolic planning. Recent approaches extract symbolic abstractions from raw data by clustering to induce predicates [23, 50]; however, such approach still require fifty or more demonstrations. Segmentation [51] helps but cannot fully remove this reliance. SAT- or ASP-based model learning [13, 14] reduces data requirements for symbolic domain acquisition. Prior work typically uses these solvers to learn continuous features that improve TAMP efficiency from a few example plans with embedded continuous data [52]. However, these approaches assume a predefined symbolic domain and solved plans, whereas we jointly learn both the symbolic domain and continuous controllers directly from a handful of raw demonstrations.

In this work, we combine the abstraction of the planning domain based on ASP together with continuous controllers learning from few skill demonstrations, enabling scalable long-horizon task solving with minimal supervision.

3 Preliminaries

Symbolic Planning. Symbolic planning builds upon a formal domain description $\sigma = \langle \mathcal{E}, \mathcal{F}, \mathcal{S}, \mathcal{O} \rangle$, where $\mathcal{E}$ is a set of entities, $\mathcal{F}$ a set of boolean or numerical predicates over entities, $\mathcal{S}$ a set of symbolic states formed by grounded predicates, and $\mathcal{O}$ a set of operators. Each operator $o \in \mathcal{O}$ is defined by preconditions ψ and effects ω over predicates. A grounded operator $\hat{o}$ binds objects to parameters and can be applied if its preconditions hold, updating the state according to its effects. A planning task $T = (\mathcal{E}, \mathcal{F}, \mathcal{O}, s_0, s_g)$ seeks a plan $\mathcal{P} = [o_1, \ldots, o_{|\mathcal{P}|}]$ that transitions from initial state s_0 to goal state s_g [36].

Imitation Learning (IL). IL aims to learn a policy $\pi(\tilde{s})$ from expert demonstrations $\{(\tilde{s}_t, a_t, \tilde{s}_{t+1})\}_{t=0}^{T}$, where $\tilde{s}_t$ is a continuous state, a_t the expert action, and $\tilde{s}_{t+1}$ the resulting state. The policy minimizes the mean squared error between predicted and expert actions, defined as $L(\pi) = \frac{1}{T} \sum_{t=0}^{T} \|\pi(\tilde{s}_t) - a_t\|^2$. Unlike reinforcement learning, IL avoids exploration and reward engineering, enabling more data-efficient learning of complex behaviors from demonstrations.

Neuro-Symbolic Architecture. Neuro-symbolic architectures combine symbolic reasoning with neural control. A planner solves a STRIPS task $T = \langle \mathcal{E}, \mathcal{F}, \mathcal{O}, s_0, s_g \rangle$ to produce a plan $\mathcal{P} = [o_1, \ldots, o_{|\mathcal{P}|}]$, where each operator o_i is refined into a neural skill $\pi_i \in \Pi$. Each skill π_i interacts with the environment to realize the operator's effects ω_i, transitioning the system from a state s to a new state s'. This layered approach enables flexible execution in continuous spaces while maintaining high-level task abstraction.

Problem Formulation. We consider a dataset of object-space skill demonstrations $\mathcal{D} = \{\tilde{\tau}_0, \ldots, \tilde{\tau}_{|\mathcal{D}|}\}$, where each trajectory $\tilde{\tau}_i = \{(\tilde{s}_t, a_t, \tilde{s}_{t+1})\}_{t=0}^{|\tilde{\tau}_i|-1}$ captures state-action transitions over objects $\varepsilon \in \mathcal{E}$. Each demonstration also provides two images: one of the initial state v and one of the final state v'. Human input is limited to two forms: (1) demonstrate individual lifted skills, with skill labels shared across objects (2) match high-level states through image comparison. No symbolic annotations, object types, or semantics are assumed.

4 Neuro-Symbolic Imitation Learning

After receiving the set of raw demonstration trajectories $\mathcal{D}$, our neuro-symbolic agent aims to learn robust, generalizable solutions for long-horizon Task and Motion Planning (TAMP) problems. Each trajectory is mapped to a node transition $\tau_i^{node} = \zeta(\tilde{\tau}_i) = (n_{\text{start}}, l, n_{\text{end}})$ where n_{start} and n_{end} denote abstract high-level states at the beginning and end of the skill demonstration, and l is a human-assigned label describing the transition between them. This process structures the skill demonstrations into a task-space abstraction (Fig. 1). The nodes correspond to abstract high-level states (whose symbolic representation remains undiscovered), and the edges represent transitions induced by action operators. A skill typically transitions the environment between two high level states, i.e., performs a node transition, providing a task-space abstraction for the agent. Such skill abstraction can be represented as an edge (n, l, n') in a graph G, while n and n' can be paired to their respective visual snapshots v and v', capturing the beginning and the end of the skill.

To leverage this structure, we adopt a bi-level neuro-symbolic learning approach combining imitation learning and symbolic planning. At the symbolic level, the agent constructs a graph from node transitions to infer operators $\mathcal{O}$ and predicates $\mathcal{F}$ that abstract the task-space transitions. At the skill level, the agent learns neural policies $\pi_i \in \Pi$ that realize each operator $o_i \in \mathcal{O}$ by imitating corresponding segments of the demonstration trajectories.

Inspired by the options framework in Hierarchical Reinforcement Learning [53], we further decompose each skill into sequential action steps. Action steps are automatically identified through consistent sequential action space patterns. For instance, a *MOVE* operator in a *Pick & Place* task decomposes into *reach pick*, *pick*, *reach drop*, and *drop* stages, each constrained to simpler action spaces (e.g., end-effector position or gripper aperture). This decomposition reduces the complexity of individual policies and restricts their action spaces, simplifying learning (see Fig. 2, and [53]).

After acquiring the symbolic operators and their associated neural skills, planning proceeds by specifying an initial and a goal node translated within a PDDL planning problem. A classical planner, MetricFF [54], computes an abstract plan $\mathcal{P} = [o_1, \ldots, o_{|\mathcal{P}|}]$ mapping operators to their corresponding neural skills *(Alg. 1.line 10)*. Execution unfolds by sequentially invoking the associated policies π_i, each internally organized into action-step sub-policies $\pi_{i,j}$ executed until a learned termination condition, modeled by a learned function approximation π_{term}, is met *(Alg. 1.lines 11-17)*.

This hierarchical and modular framework enables the agent to generalize beyond the demonstrations to unseen tasks, adapt to different object configurations, and robustly solve complex, long-horizon problems with limited training data.

4.1 Learning Symbolic Structures from Sparse Demonstrations

Our method constructs a symbolic graph from unordered demonstrations with minimal human input *(Alg. 1.line 1)*. When the agent reaches a new high-level state—known at that moment as a black-box node n' distinct from the current state n—the human (1) assigns a transition label l, and (2) links n' to existing nodes by matching visual snapshots v and v' paired with each node. This process adds an edge (n, l, n') to the evolving graph $G = \langle V, E, L \rangle$.

After collecting demonstrations, we compute a minimal bisimulation $\bar{G}$ of G (Alg. 1, line 2) to eliminate redundant structure (Figs. 8a–8c). This keeps the graph compact, reducing both the search space for domain learning and the effort required for annotation. Formally, for two labeled graphs $G_1 = \langle V_1, E_1, L_1 \rangle$ and $G_2 = \langle V_2, E_2, L_2 \rangle$, a bisimulation relation $R \subseteq V_1 \times V_2$ satisfies: If $(s_1, s'_1) \in E_1$ labeled by l, then there exists $(s_2, s'_2) \in E_2$ with the same label and $(s'_1, s'_2) \in R$, and vice versa. If such an R exists, G_1 and G_2 are bisimilar.

To extract first-order symbolic representations from the graph, we adopt the ASP-based framework of [13, 14] *(Alg. 1.line 3)*. The goal is to find the simplest planning instance $P = \langle \sigma, I \rangle$, where σ is the domain theory and $I = \langle \mathcal{E}, \text{Init}, \text{Goal} \rangle$ defines instance-specific objects and grounded states. Although Init and Goal need **not** be logically specified during demonstrations, they serve to maintain consistency within the symbolic notation.

Each P defines a labeled graph $G(P)$ where nodes correspond to symbolic states and labeled edges represent action transitions. We solve for σ such that $G(P)$ is isomorphic to the input graph G, extending naturally to multiple demonstrations $G_1, \ldots, G_k$ by learning a shared domain σ and separate instances $P_i = \langle \sigma, I_i \rangle$. We assume graphs are complete and noise-free, and that action labels provide no structural or predicate-level information. Finally, the learned domain σ is expressed in PDDL, enabling classical planners to operate over the abstracted symbolic space.

Algorithm 1 *Neuro-Symbolic Imitation Learning* $(\mathcal{D}, \zeta, \mathcal{E})$

```
Require: A set of raw demonstration trajectories $\mathcal{D} = \{\tilde{\tau}_0, \dots, \tilde{\tau}_{|\mathcal{D}|}\}$
Require: An automatic function $\zeta$ that extracts $\tau^{node} = \zeta(\tilde{\tau})$
Require: A set of entities $\varepsilon \in \mathcal{E}$
    Learning Phase
 1: $\mathcal{G} \leftarrow$ Build_Graph($\mathcal{D}, \zeta$)                ▷ Querying Human input, sec.4.1
 2: $\bar{\mathcal{G}} \leftarrow$ Get_Minimal_Graph($\mathcal{G}$)            ▷ Verify if a Bisimulation exists
 3: $\{\mathcal{F}, \mathcal{O}\} \leftarrow$ Abstract($\bar{\mathcal{G}}$)                  ▷ ASP solver, sec.4.1
 4: $\Pi \leftarrow \{\pi_i, \pi_i$ is mapped to $o_i \ \forall o_i \in \mathcal{O}\}$
 5: $\forall \pi_i \in \Pi, \{\pi_{i,j}, \mathcal{D}_{o_i,j}\} \leftarrow$ Cluster_Action_Steps($\pi_i, \mathcal{D}_{o_i}$)   ▷ Fig. 2, Split Skills into Steps
 6: $\forall \pi_{i,j} \in \Pi$, Obs($\pi_{i,j}$) $\leftarrow$Oracle($\pi_{i,j}$)           ▷ Enforces $\phi(\tilde{s})$, sec.4.2
 7: $\forall \pi_{i,j} \in \Pi$, Train($\pi_{i,j}, \mathcal{D}_{o_i,j}$)              ▷ Diffusion policies, sec.4.3
    Execution Phase
 8: $\{n_0, n_g\} \leftarrow$ QueryTask()     ▷ Query to select start and goal nodes (via paired state picture)
 9: $T = \langle \mathcal{E}, \mathcal{F}, \mathcal{O}, s_0, s_g \rangle$
10: $\mathcal{P} \leftarrow$ Plan($T$)                                  ▷ $\mathcal{P} = \langle o_1, o_2, ..., o_{|\mathcal{P}|} \rangle$
11: for $o_i \in \mathcal{P}$ do
12:     $\pi_i \leftarrow$ mapped($o_i$)                  ▷ Sequential execution of Skills, sec.4.2.Intro
13:     for $\pi_{i,j} \in \pi_i$ do
14:         $\pi_{exec}, \pi_{term} \leftarrow \pi_{i,j}$
15:         Execute($\pi_{exec}, \pi_{term}$)              ▷ Sequential execution of Skill Steps
16:     end for
17: end for
```

4.2 The Oracle - Filtering Skill-Relevant Data

For each skill, symbolic abstractions identify the critical objects and relations needed for state transitions, ensuring that skills receive only the observations relevant to their symbolic operator models *(Alg. 1.line 6)*. Formally, let $o_i \in \mathcal{O}$ be an operator associated with the skill π_i. The operator o_i defines a symbolic transition between states s and s' such that: $s' = o_i(s)$, where the transition is characterized by changes in a subset of object states $\mathcal{E}_i \subseteq \mathcal{E}$, where $\mathcal{E}$ is the full set of objects in the environment. Let us call : $\mathcal{E}_{o_i} = \{\varepsilon_k \in \mathcal{E} \mid \text{symbolic state of } \varepsilon_k \text{ changes under } o_i\}$, the subset of objects relevant to the grounded operator $o_i \in \mathcal{O}$. Consequently, for each skill π_i, we define the filtered observation function γ that maps the full state $\tilde{s}$ to a reduced observation space containing only relevant objects:

$$\gamma(\tilde{s}, o_i) = \tilde{s}(\mathcal{E}_{o_i}). \tag{1}$$

This filtering mechanism ensures that only the relevant objects are considered during skill execution.

To further enhance efficiency and scalability, we define a transformation function α that maps absolute object coordinates to coordinates relative to the agent's end effector. Then the function α can be expressed as: $\alpha(\tilde{s}, \mathcal{E}_{o_i}) = \{\tilde{s}(\varepsilon_k) - \tilde{s}(\text{EE}) \mid \varepsilon_k \in \mathcal{E}_{o_i}\}$, where $\tilde{s}(\varepsilon_k)$ represents the absolute coordinates of object ε_k, and $\tilde{s}_t(\text{EE})$ represents the position of the agent's end effector, i.e., the part that interacts with the objects. This transformation ensures that policies remain invariant to global positioning, improving generalization across different spatial configurations.

We call the function ϕ, which applies the object filtering γ and transformation α, *the Oracle*:

$$\phi(\tilde{s}_t) = \alpha \circ \gamma(\tilde{s}_t, o_i). \tag{2}$$

For task execution, a planner generates a symbolic plan $\mathcal{P} = [o_1, o_2, \dots, o_n]$, where each operator o_i is grounded to a specific subset of objects $\mathcal{E}_{o_i}$ in the environment. Each operator is then executed using a policy π_{exec}, whose observation space is determined by *the Oracle* ϕ:

$$a_t = \pi_{exec}(\phi(\tilde{s}_t)).$$

For example, if the planner grounds a unary operator `pick(.)` to the entity `cube1`, resulting in the grounded operator $o =$ `pick(cube1)`, the filtering mechanism ensures that the execution policy observes only cube1's relevant properties (e.g., position, orientation, grasp affordances) relative to the agent's proprioception.

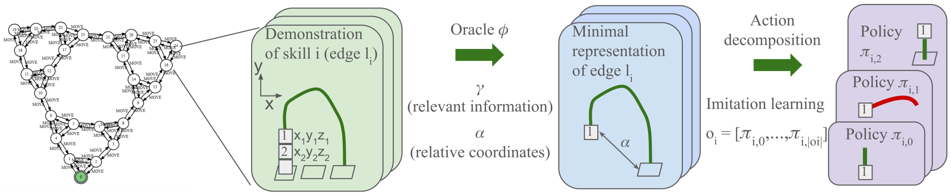

Demonstration of skill i (edge l_i)
y
x
1 $x_1y_1z_1$
2 $x_2y_2z_2$
Oracle ϕ
γ (relevant information)
α (relative coordinates)
Minimal representation of edge l_i
1
α
Action decomposition
Imitation learning
$o_i = [\pi_{i,0},\ldots,\pi_{i,|oi|}]$
Policy $\pi_{i,2}$
Policy $\pi_{i,1}$
Policy $\pi_{i,0}$
1
MOVE

Figure 2: Shown here is an example demonstration of the MOVE operator in the Towers of Hanoi domain, where block 1 is moved off block 2 and placed onto a platform. The agent partitions skill demonstrations into action steps, with an oracle ϕ filtering observations to simplify learning. The demonstration is first collected, then filtered using γ to retain operator-relevant objects (block 1 and platform 3) and α to express coordinates relative to the end-effector. The trajectory is then decomposed into a sequence of simpler action steps. This enables efficient training of low-level controllers, which are sequenced to execute each symbolic operator *(Alg. 1.line 5)*.

The oracle function ϕ is automatically derived from abstractions and requires no human input. It generates a structured perception which reduces observation dimensionality while preserving task-relevant information, improving learning efficiency and scalability. Observation filtering thus serves as a symbolic attention mechanism, dynamically focusing observations on essential information at each execution step.

4.3 Learning Continuous Control Policies

We learn continuous-space control policies from demonstration data $\mathcal{D}$, where each trajectory $\tilde{\tau}_i$ consists of state-action pairs $(\tilde{s}_t, a_t)$ *(Alg. 1.line 7)*. The objective is to find a policy π minimizing the action prediction loss over demonstrations:

$$\pi^* = \arg\min_{\pi\in\Pi} \sum_{\tilde{\tau}_i\in\mathcal{D}} \sum_{t=0}^{T_i} \mathcal{L}(\pi(\tilde{s}_t), a_t),$$

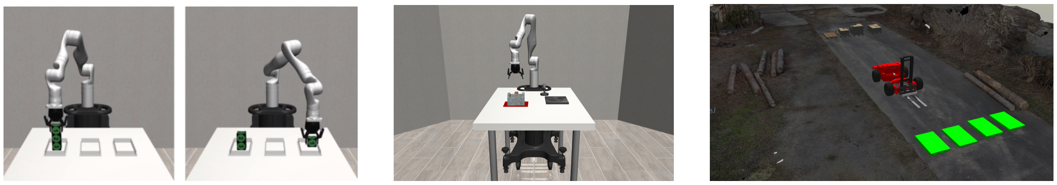
where $\mathcal{L}$ measures action prediction error.

To improve generalization, we preprocess states by filtering observations $\mathcal{E}_{o_i}$ and applying a transformation ϕ to express object positions relative to relevant frames: $\pi^*(\tilde{s}_t) = \pi^*(\phi(\tilde{s}_t))$.

For robustness and sample efficiency, we adopt diffusion policies [55]. A diffusion model $p_\theta(a_t|\phi(\tilde{s}_t))$ learns to generate actions by denoising perturbed expert actions, where ϵ_θ predicts the added noise: $\mathcal{L}_{\text{diff}} = \mathbb{E}_{(\tilde{s}_t,a_t)\sim\mathcal{D},\epsilon\sim\mathcal{N}(0,I)}[\|\epsilon - \epsilon_\theta(a_t + \sigma\epsilon, \phi(\tilde{s}_t))\|^2]$. Diffusion policies capture multi-modal action distributions, avoiding the mode collapse common in direct regression. Our framework remains compatible with any imitation learning algorithm for control. Actions are hierarchically structured into control spaces (e.g., end-effector motion, gripper control), enabling flexible and modular execution across tasks.

5 Evaluation

We evaluate our approach across six environments: four in Robosuite including *Stacking* (involving 3 cubes and randomly generated stacking tasks), *Kitchen*, *Nut Assembly* and our own implementation of the *Towers of Hanoi* problem using numerated cubes, as well as completely different environments in ROS 2 & Gazebo, to perform *Forklift Loading/Unloading tasks)* and *Multiple Pallets Storage tasks* where the agent needs to store multiple pallets at different locations (see Fig. 3). The forklift's articulated kinematics (see appendix) in the latter environments introduce significant control challenges, particularly in fork insertion, as the forward displacement is controlled by the rear of the crawler. Robosuite environments use Cartesian control of the gripper; The ROS 2 & Gazebo Forklift domain uses motion and forks control. The observations consists of the 6D pose of the objects in the scene and the end-effector (i.e., the part that interacts with the objects). The tasks are randomized during evaluation. The demonstrations provided to our framework are short-horizon



(a) Robosuite: *Stacking* & *Hanoi*. (b) Robosuite: *Kitchen*. (c) ROS2/Gazebo: *Pallets Storage*.

Figure 3: Illustrations of some of the simulation domains used for evaluation.

skills, for instance stacking, fork insertion, pallet loading, unloading. Each demonstration has a maximal length of 300 steps, and is blurred with noise in the observation dimensions.

Each agent is evaluated over 30 episodes, and the results are averaged across 5 seeds. Each policy is trained for 8,000 epochs. **Metrics**: we use long-horizon tasks success rate for the main experiments. We also display advancement towards completion as evaluation metrics for the generalization experiments. **Baselines**: we compare our approach (N-S) against three end-to-end neural baselines: (1) **(IL)** Imitation Learning over entire trajectories, (2) **(H-IL)** Hierarchical Imitation Learning with a high-level policy grounding low-level policies, similar to our oracle reasoning system (3) **(H-IL Dense)** A similar Hierarchical Imitation Learning baseline, but with the high-level policy receiving the full trajectory instead of a single point at the beginning of each stacking operation. To ensure fair access to information and adherence to the Markov property, the IL baseline policies and the high-level policies of the H-IL baselines receive absolute rather than relative observations. The low-level policies of the H-IL baselines receive relative observations, as transformed by α in our approach.

Generalization and Fine Tuning

Zero-shot generalization: We evaluate how well our framework scales to harder Hanoi configurations (4cubes×3pegs to 7cubes×5pegs) and spatial shifts (peg displacements up to 10cm) using only the demonstrations on the most basic configuration (3x3) In the Forklift domain, we extend the evaluation to *Multiple Pallets Storage tasks* with multiple pallets and zones, also by performing zero-shot transfer. **Few-shot fine-tuning**: We start from 5 full *MOVE* skill demos, which we incrementally extend over harder Hanoi configurations with 5 additional expert *reach-place* action step demonstrations. We demonstrate that our agent benefits from continually training on new, more complex, demonstration in a curriculum manner (see Fig. 5).

6 Results

Neural Skills – We isolate neural skill learning in the *Forklift Loading/Unloading* and *Stacking* tasks (Fig. 4 *Above Left & Middle*), which emphasize short-horizon control rather than planning. All agents perform similarly in the forklift task due to the absence of planning, a single object with a fixed relative pose, and no skill decomposition. In contrast, our agent begins to outperform baselines in *Stacking* by leveraging the *Oracle* for policy refinement, combining relative observations (α) with symbolic abstractions (γ)—a capability baselines lack. Action space clustering further optimizes skill execution, achieving near 100% success with as few as 5 demonstrations. The lower success rate in the *Forklift Loading/Unloading* domain stems from the vehicle's articulated kinematics: rear-wheel steering complicates precise fork insertion, and learning these dynamics is challenging in the fork's reference frame. Nonetheless, 30 demonstrations are sufficient to overcome these difficulties.

Reasoning Abilities – Reasoning abilities are evaluated in the long-horizon environments (Figs. 4, 5). Baselines completely fail to solve the long-horizon *Towers of Hanoi* task even with 500 full demonstrations, unable to ground sub-goals and diverging into random outputs. In contrast, our method consistently abstracts problem constraints from partial demonstrations, building correct symbolic domains and solving the task reliably. Successes in *Nut Assembly*, *Kitchen* and *Multiple Pallets Storage* further confirms our approach's domain-independent reasoning. Failures occur only when low-level skills fail to achieve expected effects, not due to reasoning errors.

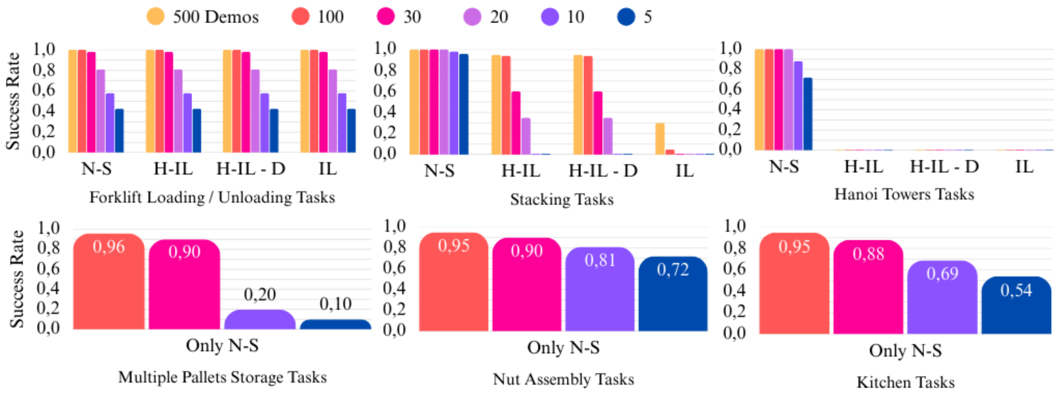


Figure 4: Performance comparison between our *Neuro-Symbolic (N-S)* framework and baseline methods. Our approach achieves high success rates on short—*Stacking & Forklift Pallet Loading/Unloading*—and long-horizon tasks—including *Towers of Hanoi*, *Multiple Pallets Storage*, *Nut Assemly & Kitchen*—even with as few as 5 demonstrations. Our approach is domain agnostic, and works in very different scenarios.

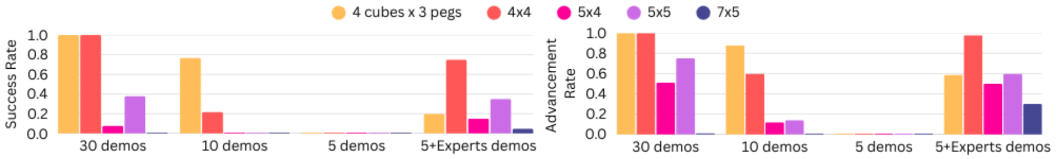


Figure 5: Zero- and few-shot generalization results on Different Hanoi Towers configurations.

Generalization & Fine-Tuning – Zero- and few-shot evaluations (Fig. 5) indicate encouraging generalization to new symbolic domains and spatial shifts. With only 5 demonstrations, performance in the 3×3 setting was limited, but adding a few expert corrections (*5+Experts*) yielded significant gains—outperforming agents trained from scratch with 30 demonstrations while using less data overall. In the more complex 7×5 setting, our agent occasionally demonstrated robust behavior, such as recovering from disturbances without replanning, despite occasional control inaccuracies. These results suggest our neuro-symbolic imitation learning framework generalizes well from limited data and benefits from curriculum learning. Importantly, our approach also addresses the two desiderata identified by Silver et al. [31], in known and unknown settings: (1) our subgoal-conditioned policies and samplers enable reaching diverse low-level states that fulfill the same abstract goal, supporting flexible execution (KD1), and (2) our bilevel symbolic planner can fall back to alternative skill sequences when a plan fails, supporting replanning under abstraction-induced constraints (KD2).

7 Conclusion

Our framework advances explainable, generalizable, and data-efficient long-horizon task execution, addressing key challenges in modern AI. By allowing agents to generalize across diverse tasks and environments, including novel configurations and previously unseen problems, it demonstrates high performance in solving complex reasoning tasks with only a few demonstrations. Furthermore, our approach scales efficiently to more complex tasks with minimal additional supervision, making it a powerful paradigm for human-taught robotics. Importantly, it eliminates the need for direct environment interaction during learning, enabling agents to plan and act effectively without risky exploration or reliance on simulated models or digital twins. Thus, we have demonstrated the utility of our approach in robotics and industrial settings, paving the way for more efficient, scalable, and human-friendly solutions in real-world applications.

7.1 Limitations and Future Work

Our approach has some limitations. Like any neural learning method, imitation learning depends heavily on data quality. Initially, we observed degraded performance due to overly fast demonstrations, which compromised precision; slowing the execution significantly improved success rates for tasks such as pallet insertion and stacking. Effective datasets must strike a balance between diversity for generalization and consistency to support efficient learning. Furthermore, computing the relative pose between objects and the end-effector requires consistently accurate estimation of a valid grasping point. Failing to do so limits the agent's ability to generalize to novel objects, especially after observation filtering by the oracle. Our method relies on an oracle to simplify skill learning, assuming the symbolic solver abstracts information at a sufficiently low level to maintain the Markov property. If not, the learning might not be informed enough, thus leading to the inability to exploit the data for optimal action decision-making.

Second, our framework supports efficient fine-tuning, enabling fast human-taught robotics. When scaling to more objects of known types, the symbolic domain remains valid; failures arise mainly from neural policies encountering out-of-distribution states, which can be corrected with targeted demonstrations for fine-tuning. If the observation or action space changes due to environmental novelties, the agent must either adapt its planning or learn new controllers. In cases where new actions (e.g., screwing), object types (e.g., screwdriver), or predicates (e.g., screwed) are required beyond the original abstraction, performance can usually be restored by extending the graph with a few additional nodes and image pairings. In particular, prior work by Rodriguez et al. [14] (Sec. 7.1–7.2) demonstrates strong robustness of the ASP solver to noisy or missing nodes and edges.

Future work includes integrating real human demonstrations into our neuro-symbolic framework and validating it on real systems. While we currently use simulated data, we notably plan to capture expert demonstrations of pallet manipulation and multiple pallets management with a real forklift. Despite the ROS 2 & Gazebo simulation providing a 1:1 mapping, we want our agent to learn without using the simulation. The main challenge is learning from noisy demonstrations to control the forklift directly. We also aim to demonstrate all the Robosuite scenarios on a real Kinova arm. Additionally, our framework supports future integration of Foundation Models (e.g., CLIP, GPT) to further automate state matching and skill labeling from visual inputs.

Acknowledgments

This work was in part funded by grant N00014-24-1-2024 from the US Office of Naval Research.

References

[1] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne. Imitation learning: A survey of learning methods. *ACM Comput. Surv.*, 50(2):21:1–21:35, Apr. 2017. ISSN 0360-0300. doi:10.1145/3054912.

[2] T. Osa, J. Pajarinen, G. Neumann, J. Bagnell, P. Abbeel, and J. Peters. An algorithmic perspective on imitation learning. *Foundations and Trends in Robotics*, 7:1–179, Nov. 2018. doi:10.1561/2300000053.

[3] B. Fang, S. Jia, D. Guo, M. Xu, S. Wen, and F. Sun. Survey of imitation learning for robotic manipulation. *International Journal of Intelligent Robotics and Applications*, 3, Dec. 2019. doi:10.1007/s41315-019-00103-5.

[4] B. Zheng, S. Verma, J. Zhou, I. Tsang, and F. Chen. Imitation learning: Progress, taxonomies and challenges. (arXiv:2106.12177), Oct. 2022. doi:10.48550/arXiv.2106.12177. URL `http://arxiv.org/abs/2106.12177`. arXiv:2106.12177 [cs].

[5] M. Zare, P. M. Kebria, A. Khosravi, and S. Nahavandi. A survey of imitation learning: Algorithms, recent developments, and challenges. *IEEE Transactions on Cybernetics*, 54(12): 7173–7186, Dec. 2024. ISSN 2168-2275. doi:10.1109/TCYB.2024.3395626.

[6] N. Rajaraman, L. F. Yang, J. Jiao, and K. Ramachandran. Toward the fundamental limits of imitation learning. (arXiv:2009.05990), Sept. 2020. doi:10.48550/arXiv.2009.05990. URL `http://arxiv.org/abs/2009.05990`. arXiv:2009.05990 [cs].

[7] T. Xu, Z. Li, and Y. Yu. Error bounds of imitating policies and environments. (arXiv:2010.11876), Oct. 2020. doi:10.48550/arXiv.2010.11876. URL `http://arxiv.org/abs/2010.11876`. arXiv:2010.11876 [cs].

[8] J. Wolfe, B. Marthi, and S. Russell. Combined task and motion planning for mobile manipulation. *Proceedings of the International Conference on Automated Planning and Scheduling*, 20:254–257, May 2010. ISSN 2334-0843. doi:10.1609/icaps.v20i1.13436.

[9] L. P. Kaelbling and T. Lozano-Pérez. Hierarchical task and motion planning in the now. In *2011 IEEE International Conference on Robotics and Automation*, page 1470–1477, May 2011. doi:10.1109/ICRA.2011.5980391. URL `https://ieeexplore.ieee.org/abstract/document/5980391/`.

[10] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez. Integrated task and motion planning. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(Volume 4, 2021):265–293, May 2021. ISSN 2573-5144. doi:10.1146/annurev-control-091420-084139.

[11] A. Arora, H. Fiorino, D. Pellier, M. Métivier, and S. Pesty. A review of learning planning action models. *The Knowledge Engineering Review*, 33:e20, 2018. ISSN 0269-8889, 1469-8005. doi:10.1017/S0269888918000188.

[12] 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*, 61: 215–289, Jan. 2018. ISSN 1076-9757. doi:10.1613/jair.5575.

[13] B. Blai and G. Hector. *Learning First-Order Symbolic Representations for Planning from the Structure of the State Space*. IOS Press, 2020. doi:10.3233/FAIA200361. URL `https://www.medra.org/servlet/aliasResolver?alias=iospressISBN&isbn=978-1-64368-100-9&spage=2322&doi=10.3233/FAIA200361`.

[14] I. D. Rodriguez, B. Bonet, J. Romero, and H. Geffner. Learning First-Order Representations for Planning from Black Box States: New Results. In *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning*, pages 539–548, 11 2021. doi:10.24963/kr.2021/51. URL https://doi.org/10.24963/kr.2021/51.

[15] T. Silver, R. Chitnis, J. Tenenbaum, L. P. Kaelbling, and T. Lozano-Perez. Learning symbolic operators for task and motion planning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 3182–3189, Prague, Czech Republic, Sept. 2021. IEEE. ISBN 978-1-6654-1714-3. doi:10.1109/IROS51168.2021.9635941. URL https://ieeexplore.ieee.org/document/9635941/.

[16] P. Verma, S. R. Marpally, and S. Srivastava. Discovering user-interpretable capabilities of black-box planning agents. (arXiv:2107.13668), May 2022. URL http://arxiv.org/abs/2107.13668. arXiv:2107.13668 [cs].

[17] R. Chitnis, T. Silver, J. B. Tenenbaum, T. Lozano-Perez, and L. P. Kaelbling. Learning neuro-symbolic relational transition models for bilevel planning. (arXiv:2105.14074), June 2022. URL http://arxiv.org/abs/2105.14074.

[18] A. Ahmetoglu, M. Y. Seker, J. Piater, E. Oztop, and E. Ugur. Deepsym: Deep symbol generation and rule learning from unsupervised continuous robot interaction for planning. *Journal of Artificial Intelligence Research*, 75:709–745, Nov. 2022. ISSN 1076-9757. doi:10.1613/jair.1.13754. arXiv:2012.02532 [cs].

[19] T. Silver, R. Chitnis, N. Kumar, W. McClinton, T. Lozano-Perez, L. P. Kaelbling, and J. Tenenbaum. Predicate invention for bilevel planning. (arXiv:2203.09634), Nov. 2022. URL http://arxiv.org/abs/2203.09634. arXiv:2203.09634 [cs].

[20] A. Li and T. Silver. Embodied active learning of relational state abstractions for bilevel planning. (arXiv:2303.04912), June 2023. doi:10.48550/arXiv.2303.04912. URL http://arxiv.org/abs/2303.04912. arXiv:2303.04912 [cs].

[21] N. Kumar, W. McClinton, R. Chitnis, T. Silver, T. Lozano-Pérez, and L. P. Kaelbling. Learning efficient abstract planning models that choose what to predict. (arXiv:2208.07737), Sept. 2023. URL http://arxiv.org/abs/2208.07737. arXiv:2208.07737 [cs].

[22] M. Iii and W. B. *Learning Compositional Abstract Models Incrementally for Efficient Bilevel Task and Motion Planning*. Thesis, Massachusetts Institute of Technology, Feb. 2024. URL https://dspace.mit.edu/handle/1721.1/153869. Accepted: 2024-03-21T19:12:11Z.

[23] 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:2402.11871), Mar. 2024. doi:10.48550/arXiv.2402.11871. URL http://arxiv.org/abs/2402.11871. arXiv:2402.11871 [cs].

[24] E. Umili, E. Antonioni, F. Riccio, R. Capobianco, D. Nardi, and G. D. Giacomo. Learning a symbolic planning domain through the interaction with continuous environments. 2021.

[25] F. Yang, D. Lyu, B. Liu, and S. Gustafson. Peorl: Integrating symbolic planning and hierarchical reinforcement learning for robust decision-making. (arXiv:1804.07779), June 2018. URL http://arxiv.org/abs/1804.07779. arXiv:1804.07779 [cs, stat].

[26] L. Illanes, X. Yan, R. T. Icarte, and S. A. McIlraith. Symbolic plans as high-level instructions for reinforcement learning. *Proceedings of the International Conference on Automated Planning and Scheduling*, 30:540–550, June 2020. ISSN 2334-0843. doi:10.1609/icaps.v30i1.6750.

[27] H. Kokel, A. Manoharan, S. Natarajan, B. Ravindran, and P. Tadepalli. Reprel: Integrating relational planning and reinforcement learning for effective abstraction. *Proceedings of the International Conference on Automated Planning and Scheduling*, 31:533–541, May 2021. ISSN 2334-0843. doi:10.1609/icaps.v31i1.16001.

[28] L. Guan, S. Sreedharan, and S. Kambhampati. Leveraging approximate symbolic models for reinforcement learning via skill diversity. 2022.

[29] J. Balloch, Z. Lin, R. Wright, X. Peng, M. Hussain, A. Srinivas, J. Kim, and M. O. Riedl. Neuro-symbolic world models for adapting to open world novelty. (arXiv:2301.06294), Jan. 2023. URL `http://arxiv.org/abs/2301.06294`. arXiv:2301.06294 [cs].

[30] S. Cheng and D. Xu. League: Guided skill learning and abstraction for long-horizon manipulation. *IEEE Robotics and Automation Letters*, 8(10):6451–6458, Oct. 2023. ISSN 2377-3766. doi:10.1109/LRA.2023.3308061.

[31] T. Silver, A. Athalye, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling. Learning neuro-symbolic skills for bilevel planning. In *Proceedings of The 6th Conference on Robot Learning*, page 701–714. PMLR, Mar. 2023. URL `https://proceedings.mlr.press/v205/silver23a.html`.

[32] K. Acharya, W. Raza, C. Dourado, A. Velasquez, and H. Song. Neurosymbolic reinforcement learning and planning: A survey. *IEEE Transactions on Artificial Intelligence*, PP:1–14, Sept. 2023. doi:10.1109/TAI.2023.3311428.

[33] P. Lorang, S. Goel, Y. Shukla, P. Zips, and M. Scheutz. A framework for neurosymbolic goal-conditioned continual learning in open world environments. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 12070–12077, Oct. 2024. doi:10.1109/IROS58592.2024.10801627. URL `https://ieeexplore.ieee.org/document/10801627`.

[34] P. Lorang, H. Horvath, T. Kietreiber, P. Zips, C. Heitzinger, and M. Scheutz. Adapting to the "open world": The utility of hybrid hierarchical reinforcement learning and symbolic planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, page 508–514, May 2024. doi:10.1109/ICRA57147.2024.10611594. URL `https://ieeexplore.ieee.org/document/10611594`.

[35] S. Goel, P. Lymperopoulos, R. Thielstrom, E. Krause, P. Feeney, P. Lorang, S. Schneider, Y. Wei, E. Kildebeck, S. Goss, M. C. Hughes, L. Liu, J. Sinapov, and M. Scheutz. A neurosymbolic cognitive architecture framework for handling novelties in open worlds. *Artificial Intelligence*, 331:104111, 2024. ISSN 0004-3702. doi:10.1016/j.artint.2024.104111.

[36] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL - The Planning Domain Definition Language, 1998.

[37] R. Chitnis, T. Silver, J. B. Tenenbaum, T. Lozano-Perez, and L. P. Kaelbling. Learning neuro-symbolic relational transition models for bilevel planning. *arXiv preprint arXiv:2105.14074*, 2021.

[38] R. T. Icarte, T. Q. Klassen, R. A. Valenzano, and S. A. McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. *JAIR*, 73:173–208, 2020.

[39] V. Sarathy, D. Kasenberg, S. Goel, J. Sinapov, and M. Scheutz. Spotter: Extending symbolic planning operators through targeted reinforcement learning. In *AAMAS*, 2021.

[40] S. Goel, Y. Shukla, V. Sarathy, M. Scheutz, and J. Sinapov. Rapid-learn: A framework for learning to recover for handling novelties in open-world environments. In *IEEE ICDL*, 2022.

[41] C. Gehring, M. Asai, R. Chitnis, T. Silver, L. Kaelbling, S. Sohrabi, and M. Katz. Reinforcement learning for classical planning: Viewing heuristics as dense reward generators. *Proceedings of the International Conference on Automated Planning and Scheduling*, 32(1):588–596, June 2022. doi:10.1609/icaps.v32i1.19846.

[42] F. Yang, D. Lyu, B. Liu, and S. Gustafson. Peorl: Integrating symbolic planning and hierarchical reinforcement learning for robust decision-making. pages 4860–4866, 07 2018. doi:10.24963/ijcai.2018/675.

[43] P. Lorang, H. Lu, and M. Scheutz. Curiosity-driven imagination: Discovering plan operators and learning associated policies for open-world adaptation. Number arXiv:2503.04931. IEEE International Conference on Robotics and Automation (ICRA), Mar. 2025. doi:10.48550/arXiv.2503.04931. URL http://arxiv.org/abs/2503.04931. arXiv:2503.04931 [cs].

[44] S. Manschitz, J. Kober, M. Gienger, and J. Peters. Learning to sequence movement primitives from demonstrations. *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, page 4414–4421, Sept. 2014. doi:10.1109/IROS.2014.6943187.

[45] H. Le, N. Jiang, A. Agarwal, M. Dudik, Y. Yue, and I. I. I. Hal Daumé. Hierarchical imitation and reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, page 2917–2926. PMLR, July 2018. URL https://proceedings.mlr.press/v80/le18a.html.

[46] K. Pertsch, Y. Lee, Y. Wu, and J. J. Lim. Demonstration-guided reinforcement learning with learned skills. July 2021.

[47] A. K. Tanwani, A. Yan, J. Lee, S. Calinon, and K. Goldberg. Sequential robot imitation learning from observations. *The International Journal of Robotics Research*, 40(10–11):1306–1325, Sept. 2021. ISSN 0278-3649. doi:10.1177/02783649211032721.

[48] Y. Zhu, P. Stone, and Y. Zhu. Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation. *IEEE Robotics and Automation Letters*, 7(2):4126–4133, Apr. 2022. ISSN 2377-3766, 2377-3774. doi:10.1109/LRA.2022.3146589.

[49] S. Teng, L. Chen, Y. Ai, Y. Zhou, Z. Xuanyuan, and X. Hu. Hierarchical interpretable imitation learning for end-to-end autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 8(1): 673–683, Jan. 2023. ISSN 2379-8904. doi:10.1109/TIV.2022.3225340.

[50] L. Keller, D. Tanneberg, and J. Peters. Neuro-symbolic imitation learning: Discovering symbolic abstractions for skill learning. (arXiv:2503.21406), Mar. 2025. doi:10.48550/arXiv.2503.21406. URL http://arxiv.org/abs/2503.21406. arXiv:2503.21406 [cs].

[51] J. Loula, K. Allen, T. Silver, and J. Tenenbaum. Learning constraint-based planning models from demonstrations. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 5410–5416. IEEE, 2020. URL https://ieeexplore.ieee.org/abstract/document/9341535/.

[52] A. Curtis, T. Silver, J. B. Tenenbaum, T. Lozano-Pérez, and L. Kaelbling. Discovering state and action abstractions for generalized task and motion planning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(5):5377–5384, 2022. ISSN 2374-3468, 2159-5399. doi:10.1609/aaai.v36i5.20475.

[53] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 1999. ISSN 0004-3702. doi:https://doi.org/10.1016/S0004-3702(99)00052-1.

[54] J. Hoffmann. The metric-ff planning system: Translating“ignoring delete lists”to numeric state variables. *Journal of artificial intelligence research*, 20:291–341, 2003.

[55] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

A Appendix

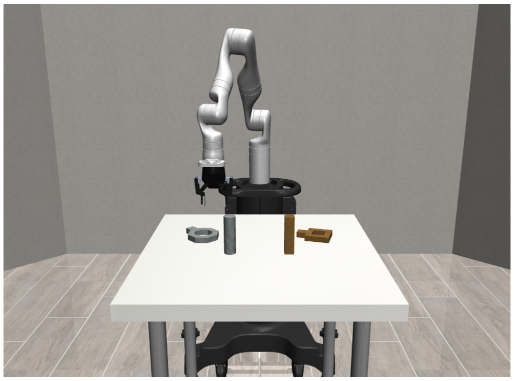


Figure 6: Illustration of the Nut Assembly environment in Robosuite.

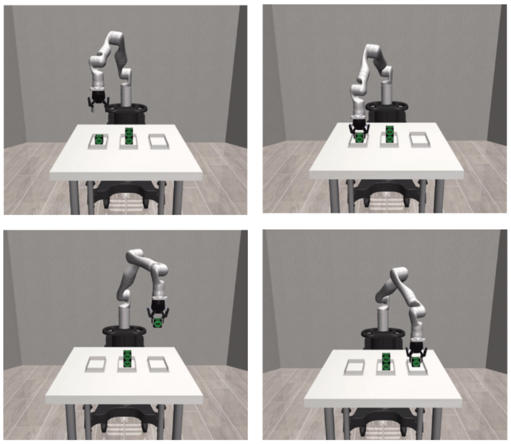


Figure 7: Example of "MOVE" skill decomposition into action steps using the *Stacking* and *Towers of Hanoi* demonstrations. From left to right: *reach-pick*, *pick*, *reach-drop*, and *drop*.

A.1 Symbols Abstraction

We use a script that automatically transcribes the formatted ASP solver output into PDDL. Note that we have access to both the symbolic operators—i.e., the models of edge transitions in the graph—and the full symbolic description of each node using the entities $\mathcal{E}$ and their relations $\mathcal{F}$. Here are the resulting domains from the graph depicted in Figs 8a, 8b, 8d and 8c:

Forklift Multiple Pallets Storage Domain

MOVE: $a_1(x_1, x_2)$
Static: $\neg(x_1 = x_2)$
Pre: $p_2(x_1), p_3(x_2)$
Eff: $\neg p_2(x_1), \neg p_3(x_2), p_3(x_1), p_2(x_2)$
UNLOAD $a_2(x_1, x_2)$
Pre: $p_3(x_2), p_4(x_1)$
Eff: $\neg p_4(x_1), p_1(), p_5(x_1, x_2)$
LOAD: $a_3(x_1, x_2)$
Pre: $p_1(), p_3(x_2), p_5(x_1, x_2)$
Eff: $\neg p_1(), \neg p_5(x_1, x_2), p_4(x_1)$
Objects: o_1, o_2, o_3, o_4 *(2 pallets x 2 locations)*

Towers of Hanoi Domain

MOVE: $a_1(x_1, x_2, x_3)$
Static: $\neg(x_1 = x_2), \neg(x_1 = x_3), \neg(x_2 = x_3)$
Pre: $p_1(x_1, x_1), p_1(x_2, x_2), p_1(x_2, x_3), p_2(x_1, x_2)$
Eff: $\neg p_1(x_1, x_1), \neg p_1(x_2, x_3), p_1(x_2, x_1), p_1(x_3, x_3)$
Objects: $o_1, o_2, o_3, o_4, o_5, o_6$ *(3 disks x 3 pegs)*

Nut Assembly Domain

MOVE: $a_1(x_1, x_2, x_3)$
Pre: $p_1(x_1, x_2)$
Eff: $\neg p_1(x_1, x_2), p_1(x_3, x_2)$
Objects: o_1, o_2, o_3, o_4 *(2 nuts x 2 pegs)*

Kitchen Domain

MOVE: $a_1(x_1, x_2, x_3)$
Static: $p_3(x_2, x_1)$
Pre: $p_1(x_2, x_1)$
Eff: $\neg p_1(x_2, x_1),\ p_1(x_2, x_3)$

TURNON: $a_2()$
Pre: $\neg p_2$
Eff: p_2

TURNOFF: $a_3()$
Pre: p_2
Eff: $\neg p_2$

WAIT: $a_4(x_1)$ *(or COOK)*
Pre: $p_2,\ p_1(o_2, o_3),\ p_1(x_1, o_2)$
Eff: $p_4(x_1)$

Objects: o_1, o_2, o_3, o_4, o_5 *(e.g., bread, pot, stove, serving-area, table)*

A.2 Post-Hoc Interpretation

The output of the ASP Solver does not come with semantics attached, but a human with domain knowledge can interpret such symbols as follows:

Forklift Multiple Pallets Storage Domain

MOVE: $a_1(x_1, x_2)$
Pre: (free_location x_1), (forklift_at x_2)
Eff:(not (free_location x_1)), (not (forklift_at x_2)), (forklift_at x_1), (free_location x_2)
UNLOAD: $a_2(x_1, x_2)$
Pre: (forklift_at x_2), (loaded_pallet x_1)
Eff:(not (loaded_pallet x_1)), (free_forklift), (at x_1 x_2)
LOAD: $a_3(x_1, x_2)$
Pre: (free_forklift), (forklift_at x_2), (at x_1 x_2))
Eff: (not (free_forklift)), (not (at x_1 x_2)), (loaded_pallet x_1)
Objects: o_1, o_2, o_3, o_4 *(2 pallets x 2 locations)*

Towers of Hanoi Domain

MOVE: $a_1(x_1, x_2, x_3)$
Pre: (clear x_1), (clear x_2), (on x_2 x_3), (greater x_1 x_2)
Eff: (not (clear x_1)), (not (on x_2 x_3)), (on x_2 x_1), (clear x_3)

Objects: $o_1, o_2, o_3, o_4, o_5, o_6$ *(3 disks x 3 pegs)*

Nut Assembly Domain

MOVE: $a_1(x_1, x_2, x_3)$
Pre: `(on` x_2 x_1`)`
Eff: `(not (on` x_2 x_1`)), (on` x_2 x_3`)`
Objects: o_1, o_2, o_3, o_4 *(2 nuts x 2 pegs)*

Kitchen Domain

MOVE: $a_1(x_1, x_2, x_3)$
Pre: `(on` x_2 x_1`), (greater` x_2 x_1`)`
Eff: `(not (on` x_2 x_1`)), (on` x_2 x_3`)`

TURNON: $a_2()$
Pre: `(not (stove-on))`
Eff: `(stove-on)`

TURNOFF: $a_3()$
Pre: `(stove-on)`
Eff: `(not (stove-on))`

WAIT: $a_4(x_1)$ *(or COOK)*
Pre: `(stove-on), (on o2 o3), (on` x_1 `o2)`
Eff: `(cooked` x_1`)`

Objects: o_1, o_2, o_3, o_4, o_5 *(e.g., bread, pot, stove, serving-area, table)*

A.3 Skills decomposition

For *Stacking*, *Nut Assembly* and the 3-disk *Towers of Hanoi* domains, the only action label was MOVE. Following Section.4.2.*Intro*, the agent split the policy learning, and broke the MOVE operator down into four action steps: *reach-pick*, *pick*, *reach-drop*, and *drop* all shown in 7.

For the forklift *Loading/Unloading Pallet* and *Multiple Pallets Storage*, the agent did not split the given skills into smaller steps.

A.4 Hardware

All experiments were conducted on a workstation equipped with an NVIDIA RTX 4090 GPU and 64 GB of RAM. Training and evaluation were performed using PyTorch and MuJoCo/ROS2+Gazebo-based environments. On average, training a single policy from 5 to 30 demonstrations took between 20 minutes to 5 hours, depending on the task complexity, the number of demonstrations and the trajectories average length. Learning the symbolic domain on a CPU only system takes from 0 secondes (*Nut Assembly domain*) to an order of magnitude of a day (*Kitchen domain*). Full evaluation runs (including symbolic domain construction, planning, and policy execution) took approximately 1–3 hours per domain. All experiments were executed using a single GPU and did not require large-scale distributed computing.

A.5 Experimental Parameters

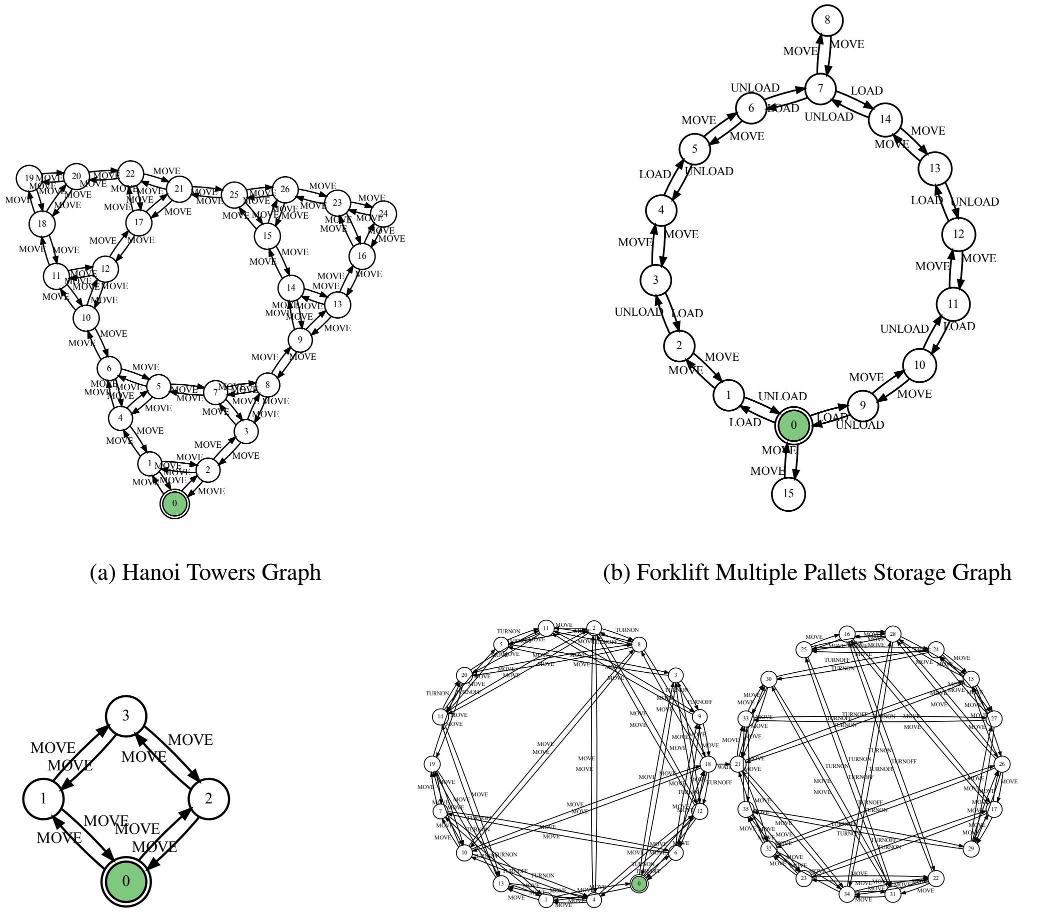
All policies were trained using the Diffusion Policy framework in a low-dimensional setting for all tasks. No keypoint features were used. The training horizon was set to 16, with 4 observation steps and 8 action steps. Policies were trained using a Transformer-based diffusion model with the following key parameters: 8 layers, 4 attention heads, 256-dimensional embeddings, and causal attention enabled. Dropout was set to 0.0 for embeddings and 0.01 for attention.

The DDPMScheduler was used with 100 training timesteps and a squared cosine beta schedule. The model was trained for 8000 epochs with a batch size of 256, a learning rate of 1×10^{-4}, and cosine learning rate scheduling with warmup over 1000 steps. The optimizer used Adam with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and a weight decay of 0.1. Exponential Moving Average (EMA) was applied with an inverse gamma of 1.0 and a cap at 0.9999.

Validation was performed every epoch with a validation split of 2% and up to 90 episodes used during training. Rollouts and checkpoints were saved every 50 epochs, and model sampling occurred every 5 epochs.

For dataset handling, we used a padded low-dimensional zarr-based dataset, with appropriate temporal padding before and after each sequence to match training horizon and step requirements.

For demonstration generation, we used a hand-coded automated script and injected Gaussian noise into all action dimensions—except when explicitly set to zero. The noise had a mean equal to half the current target command and a standard deviation of 30%, promoting trajectory diversity while maintaining task feasibility.



(a) Hanoi Towers Graph

(b) Forklift Multiple Pallets Storage Graph

(c) Nut Assembly Graph

(d) Kitchen Graph (two clusters connected via a unidirectional `WAIT` *(or COOK)* edge).

Figure 8: Task graphs for different environments. *Stacking* and *Forklift Load/Unload Pallets* graphs consist only of two nodes and a connecting edge, as they consider one-skill tasks and do not require planning.