Deep Demonstration Tracing: Learning Generalizable Imitator Policy for Runtime Imitation from a Single Demonstration

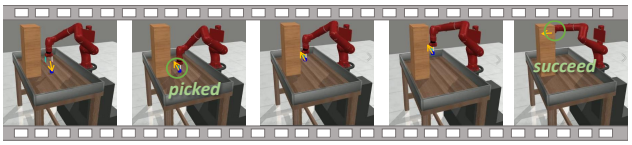
Xiong-Hui Chen [*1,2] **Junyin Ye** [*1,2] **Hang Zhao** [*3,2] **Yi-Chen Li** [1,2] **Xu-Hui Liu** [1,2] **Haoran Shi** [2] **Yu-Yan Xu** [2] **Zhihao Ye** [1,2] **Si-Hang Yang** [1,2] **Yang Yu** [1,2] **Anqi Huang** [4,2] **Kai Xu** [3] **Zongzhang Zhang** [1]

Abstract

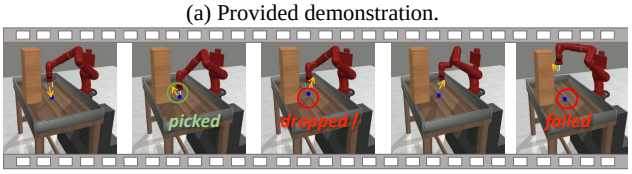
One-shot imitation learning (OSIL) is to learn an imitator agent that can execute multiple tasks with only a single demonstration. In real-world scenario, the environment is dynamic, e.g., unexpected changes can occur after demonstration. Thus, achieving generalization of the imitator agent is crucial as agents would inevitably face situations unseen in the provided demonstrations. While traditional OSIL methods excel in relatively stationary settings, their adaptability to such unforeseen changes, which asking for a higher level of generalization ability for the imitator agents, is limited and rarely discussed. In this work, we present a new algorithm called **D**eep **D**emonstration **T**racing (DDT). In DDT, we propose a demonstration transformer architecture to encourage agents to adaptively trace suitable states in demonstrations. Besides, it integrates OSIL into a meta-reinforcement-learning training paradigm, providing regularization for policies in unexpected situations. We evaluate DDT on a new navigation task suite and robotics tasks, demonstrating its superior performance over existing OSIL methods across all evaluated tasks in dynamic environments with unforeseen changes. The project page is in https://osil-ddt.github.io.

1. Introduction

Humans exhibit the ability to acquire diverse skills through a handful of demonstrations for each task. One-shot imitation learning (OSIL) has emerged as a prominent framework to emulate this learning paradigm, training an imitator agent to execute multiple tasks with only a single demonstration as context at runtime (Duan et al., 2017). Currently, the framework has been extended to imitation with visual inputs (Dasari & Gupta, 2021), multi-modal skill (Shin et al., 2023), and cross-modal task imitation (Li et al., 2021).



(a) Provided demonstration.



(b) Policy trained by a traditional OSIL method.

(c) Policy trained by DDT.

Figure 1: Illustration of OSIL policies under unforeseen changes in Meta-World tasks (Yu et al., 2019). The policy in (b) is trained by traditional OSIL (Dasari & Gupta, 2021). The grasped block may drop by chance due to disturbances that do not exist during demonstration collection.

However, these studies often assume a strong resemblance between the situations encountered during demonstration collection and those encountered when deploying the imitator policy (Duan et al., 2017). Consider a typical scenario: during demonstration collection, an expert guides a car or robot from a starting point to a target point, which may be unseen during training. Upon policy deployment, the task remains the same, i.e., reaching the same target point in a setting characterized by similar surroundings, terrain, deterministic dynamics, etc. In such settings, OSIL can be simplified to find the corresponding states and replay the action sequence. Real-world applications, however, demand a more sophisticated approach, as tasks frequently involve un-

[*]Equal contribution [1]National Key Laboratory for Novel Software Technology, Nanjing University, China & School of Artificial Intelligence, Nanjing University, China [2]Polixir Technologies [3]School of Computer Science, National University of Defense Technology [4]Nanjing University of Science and Technology, Nanjing. Correspondence to: Yang Yu <yuy@nju.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

expected changes after the demonstration is provided. The agent must not only imitate the demonstration but also adapt to unforeseen environmental changes. For autonomous vehicles, one goal is for the vehicle to navigate diverse parking lots directly (Ahn et al., 2022; Kümmerle et al., 2009) based on a human navigation trajectory, necessitating the handling of unexpected obstacles or human beings during parking trajectory imitation. In robot manipulation, one objective is for a robot arm to execute a variety of tasks solely (Dance et al., 2021; Yu et al., 2019) by observing correct operation demonstrations, requiring the agent to adeptly respond to unexpected disturbances.

In this study, we advance the conventional OSIL framework by introducing a pronounced differentiation between the phases of demonstration collection and policy deployment. This modification creates a challenging scenario, requiring the agent to adeptly respond to unexpected changes that surpass the capabilities of standard OSIL methods, rendering them ineffective in such situations, as illustrated in Fig. 1.

To address the challenges posed by unforeseen changes in the environment, we propose a new algorithm named **D**eep **D**emonstration **T**racing (DDT), designed to empower the agent to take reasonable actions in states not visited during demonstrations. In DDT, we introduce a demonstration transformer architecture for constructing the imitator-policy network. Instead of directly taking demonstration as a free context vector to represent the task the agent will face, as previous works do (Duan et al., 2017; Dasari & Gupta, 2021; Shin et al., 2023), we incorporate an inductive bias (Domingos, 2012) for policy learning by leveraging a specifically designed attention structure. This structure encourages the imitator policy to learn to accomplish tasks by adaptively tracing the demonstration in any state. To cultivate decision-making skills in unseen situations, in contrast to the behavior cloning approach adopted by prior studies (Shin et al., 2023; Mandi et al., 2022), we formulate OSIL as a context-based meta-reinforcement-learning task (meta-RL) (Rakelly et al., 2019). The policy interacts with the environment across all tasks based on corresponding demonstrations. Thanks to the trial-and-error learning mechanism of RL and the demonstration transformer architecture, the policy autonomously explores and efficiently optimizes itself to align with expert demonstrations, even in the face of unexpected situations. Lastly, we provide a theoretical analysis of the feasibility of successful imitation in scenarios with limited data coverage, emphasizing the capacity to achieve this even with only a single trajectory.

In our experiments, we establish a comprehensive Valet Parking Assist in Maze (VPA) benchmark for OSIL. The results demonstrate that our DDT algorithm outperforms existing baselines not only in training performance but also in generalization to unseen demonstrations, parking maps, and obstacles. Furthermore, DDT is applied to several robotic tasks and exhibits a clear advantage over baseline methods that struggle in these challenging environments. Notably, our results also suggest that DDT holds the potential for further performance enhancements by scaling up either dataset size or the number of parameters. In summary, our contributions include:

- Advancing the OSIL setting to achieve a higher level of generalization by introducing a substantial differentiation between the period of demonstration collection and policy deployment, supported by the development of a novel demo-navigation benchmark task suite.
- Proposing a demonstration transformer architecture to stimulate the imitator policy to adeptly learn task completion by tracing the demonstration.
- Addressing OSIL as a context-based meta-RL task, facilitating decision-making proficiency in unforeseen situations. Theoretically analyzing the conditions that the imitator is available even with one trajectory.

2. Related Work

We present the related works of OSIL in this section, deferring the comprehensive discussion of related literature to the appendix, including imitation learning (IL) (Sec. C.1), meta-IL (Sec. C.2), the combination of IL and RL (Sec. C.3), and context-based meta-RL (Sec. C.4). The OSIL paradigm (Duan et al., 2017; Dasari & Gupta, 2021) has garnered attention to achieve generalizable imitation through context-based policy models, including extracting task information from state-action sequences (Duan et al., 2017; Yu et al., 2018a) or leveraging video demonstrations (Yu et al., 2018b; Mandi et al., 2022). Duan et al. (2017) advocate for OSIL policies to comprehend the task scenario information based on a single successful demonstration. Subsequently, OSIL processes the current state input and predicts expert actions. Yu et al. (2018b) utilize human video demonstrations to acquire skills in robotic manipulation tasks, placing a specific focus on the capacity to generalize to previously unseen tasks showcased in video demonstrations. Shin et al. (2023) propose a skill-based IL framework enabling not only one-shot IL from a demonstration but also the adaptability of a learned policy to different dynamics. Mandi et al. (2022) employ contrastive learning schemes to effectively extract task information from demonstrations, enhancing cross-domain one-shot imitation capabilities.

These studies presume near-identicality between the environments during demonstration collection and agent deployment. In such scenarios, the primary challenge in OSIL lies in task identification and action inference from observations, where task accomplishment is achievable through the sample replaying of expert action sequences. However, our emphasis in this paper shifts towards OSIL in the presence

of unforeseen changes, where actions are directly observed. Herein, OSIL necessitates the agent to possess the capability to dynamically adapt its behavior in novel and unanticipated situations, departing from the conventional approach of merely repeating the action sequences demonstrated.

3. Problem Formulation

In this section, we give notations, descriptions, and the formal definition of runtime OSIL with unforeseen changes.

Markov Decision Process (MDP): We consider OSIL in an MDP $\mathcal{M}$ (Sutton & Barto, 2018) denoted by a tuple $(\mathcal{S}, \mathcal{A}, T, R, d_0, \gamma)$. In this formulation, $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $T : \mathcal{S} \times \mathcal{A} \to P(\mathcal{S})$ describes a (stochastic) transition process, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a bounded reward function, $d_0 \in P(\mathcal{S})$ is the initial state distribution, and $\gamma \in (0, 1]$ denotes the discount factor. The probability distribution set over X is denoted by $P(X)$. A policy $\Pi : \mathcal{S} \to P(\mathcal{A})$ induces a Markov chain over states within the $\mathcal{M}$. We use $\tau := \{s_0, a_0, \cdots, s_t, a_t\}$ to represent a trajectory, signifying a sequence of state-action pairs for a single episode of the Markov chain, where $s_i \in \mathcal{S}$ and $a_i \in \mathcal{A}$ represent the state and action at timestep i.

Task: We formulate the concept "task" by parameterizing MDPs as $\mathcal{M}_\omega := (\mathcal{S}, \mathcal{A}, T_\omega, R_\omega, d_0, \gamma)$, where ω is the parameter of the MDP $\mathcal{M}_\omega$ in space Ω. We assume that different MDPs only come from T_ω and R_ω and can be defined by ω. Here we consider that we only have the simplest reward function R_ω which can only indicate the ending of trajectories, e.g., c for accomplishing the task, 0 for failure, and $-c$ for dead.

Unforeseen Changes Modeling: We model unexpected changes between the demonstration collection and policy deployment phases by incorporating them into the stochasticity of T_ω. Throughout these two periods, task parameters ω remain consistent, while the agent is exposed to unforeseen states arising from inherent stochasticity. For instance, in autonomous parking tasks, despite the consistent requirement for the agent to park in the same lots (defined by ω) during both phases, the stochastic nature of T_ω captures the random appearance of pedestrians when the agent interacts with the environment.

Expert Demonstration: We use τ_ω to denote an expert demonstration that can accomplish the task with the parameter ω. In OSIL, it is often a prerequisite that the policy conducting the demonstrations should be an expert capable of completing the tasks. Here, we maintain the same setting.

Runtime One-Shot Imitation Learning: The objective of OSIL, the same as the studies in Duan et al. (2017); Dasari & Gupta (2021), is to derive a imitator policy $\Pi(a|s, \tau_\omega)$ which can accomplish the task in $\mathcal{M}_\omega$ for any $\omega \in \Omega$. For policy training, we have pre-collected expert demonstrations $\{\tau_\omega\}$ from $\mathcal{M}_\omega$ in the task set $\mathbb{M}_{\text{train}}$, along with the corresponding simulator of $\mathcal{M}_\omega$ for interacting. For deployment, given any $\omega_{\text{test}} \in \Omega$, we require the policy $\Pi(a|s, \tau_{\omega_{\text{test}}})$ to use one demonstration $\tau_{\omega_{\text{test}}}$ to accomplish the task in $\mathcal{M}_{\omega_{\text{test}}}$ at runtime, i.e., without further fine-tuning.

A fundamental problem of OSIL with unforeseen changes is how can we reconstruct any expert policy as we only know parts of optimal action in the state space and the unforeseen changes will inevitably lead the agent to unseen situations.

4. Deep Demonstration Tracing

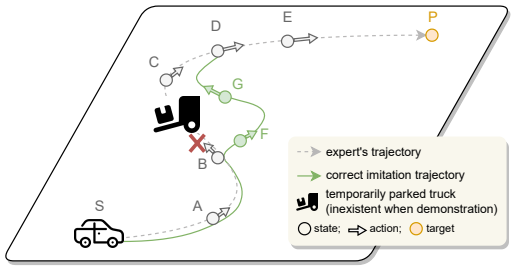


Figure 2: Illustration of how humans achieve OSIL under unforeseen changes.

In this section, we first introduce a motivation example of how humans make decisions in OSIL with unforeseen changes, utilizing a decomposed 3-stage decision-making process (Sec. 4.1). Based on this, we build a novel demonstration-based transformer architecture to stimulate policy learning followed by the 3-stage process, which is in Sec. 4.2. To enable the policy to make correct decisions in unseen situations, we propose a meta-RL-based OSIL solution in Sec. 4.3. Finally, we give a theoretical analysis of the condition for tracing a single demonstration in Sec. 4.4.

4.1. Motivation Example of Demonstration Tracing

Current OSIL studies often assume a strong similarity in the situations the agent would face between the demonstration collection and the imitator policy deployment. However, real-world applications often introduce a variety of unanticipated environmental variations after the demonstration is provided (Nagabandi et al., 2018). Thus, an agent is expected to not only replicate demonstrated actions but also adapt to unseen situations. This is easy for humans but a non-trivial task for current OSIL techniques. We give Fig. 2 to illustrate human decision-making in such settings, which also inspires our methodological framework.

In Fig. 2, the task is to imitate an expert's trajectory from the starting point S to the destination P. Initially, one might replicate the expert's actions to reach point B, but an unforeseen obstacle, a temporarily parked truck near point B,

disrupts this process. This resulted in the observation at B being different from the demonstration, as the truck was not present originally. In this case, humans would generate a detour, e.g., the sequence B-F-G, to bypass the truck and subsequently rejoin the original path at an appropriate juncture, illustrated by G-D in the figure. From this example, we decompose a three-stage human-like decision paradigm for OSIL, called Demonstration Tracing (DT) in this paper:

- **Stage 1**: Identify relevant states within the trajectory based on the current state. For example, for the state at point B, the related states can be B, C, and D.
- **Stage 2**: Analyze the expert's behavior patterns associated with these states. For example, a human would see that the expert drives forward from B, navigating a turn, to reach D and E.
- **Stage 3**: Trace the expert's demonstrations based on the relationship between the current state and the expert's behavior patterns in the demonstrations. For example, from point S to A, since the agent's state is close to the expert's, it tends to repeat the expert's actions; while in point B, since the observation is different from the demonstrations, the policy should use its common sense to avoid obstacles and traceback to the successor states (like the sequence B-F-G).

4.2. Demonstration-based Transformer Architecture

To enhance OSIL, a common strategy involves extracting representations from demonstrations (Duan et al., 2017; Mandi et al., 2022; Dasari & Gupta, 2021), capturing task parameters or relevant information. Subsequently, the downstream imitator policy functions as a context-based policy, generating adaptive action predictions based on these representations. However, direct representation without constraints has been proven to lack robustness in generalization (Luo et al., 2022; Wang et al., 2020). Consequently, numerous OSIL studies have endeavored to mitigate these issues by introducing regularization losses as the auxiliary tasks to stabilize the representations (Mandi et al., 2022; Dasari & Gupta, 2021; Duan et al., 2017).

Conversely, in the field of machine learning, there is ample evidence suggesting that network architectures can be designed to induce specific inductive biases (Domingos, 2012), thereby enhancing predictions on unseen data based on these biases. For instance, Convolutional Neural Networks (CNN) (LeCun et al., 1998) inherently possess an inductive bias through local connectivity and shared weights, proving efficacious for pattern recognition in spatial data. Inspired by this, instead of introducing additional regularization terms, we propose a novel neural network architecture, called demonstration transformer. This architecture constructs the decision-making process of the imitator network aligned with the 3-stage DT process outlined in Sec. 4.1.

Given that this 3-stage DT process is valid across all unseen demonstrations for OSIL, we posit that such a network architecture can implicitly leverage inductive biases and thus naturally enhance the generalizability of the policy network across unseen scenarios and demonstrations.

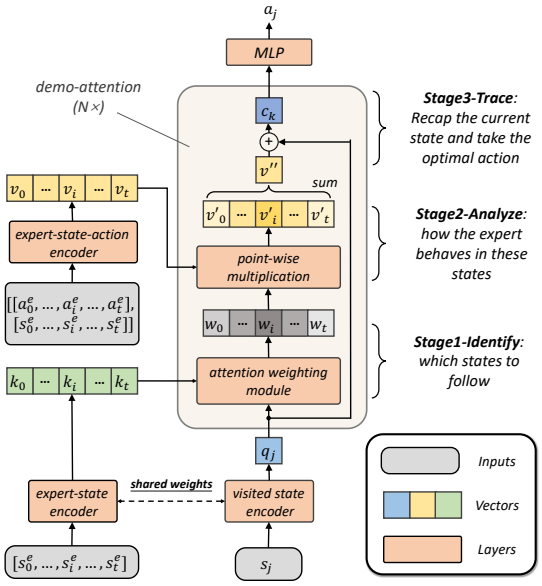


Figure 3: The demonstration transformer architecture for the actor. $[s^e_0, ..., s^e_i, ..., s^e_t]$ denote expert states and $[a^e_0, ..., a^e_i, ..., a^e_t]$ the expert action list. s_j is the visited state of the actor at timestep j. We adopt **q**, **k**, and **v** to denote the query, key, and value vectors of an attention module. $N\times$ denotes an N-layer demo-attention module, which takes the output v'' of the last layer as the input q_j of the next layer. Note that the expert-state encoder and the visited state encoder shared the same weights.

We illustrate the demonstration transformer architecture in Fig. 3. In particular, demonstration transformer utilizes the attention mechanism (Vaswani et al., 2017) and uses the following three major modules to mimic the DT process: (1) **Stage 1: Identify** which state to follow. Attention weighting is a standard module in the attention architecture (Vaswani et al., 2017), which outputs the similarity weights of the items in the key vector **k** compared with the query vector **q**. Specifically, one popular implementation is $\mathbf{w} = \mathrm{softmax}(\mathbf{q}\mathbf{k}^\intercal/\sqrt{d_k})$, where d_k is the feature dimension of **k**, and $\mathbf{q}\mathbf{k}^\intercal$ is to compute the dot products of the query with the keys in all timesteps. The dot-product operation of **k** and **q** makes states with higher similarity output a larger attention weight. We utilize this architecture and let the representation of expert states be **k** and the visited state representation be **q**, to regularize the policy and determine the expert state to follow before decision-making; (2) **Stage 2: Analyze** how the expert behave in these states. The attention weighting is followed by a point-wise multiplication

to compute v'', i.e., $v'' = \sum_i v_i w_i$. Each value vector $\mathbf{v}$ is a representation of the corresponding expert state-action pair. The point-wise multiplication applies the attention weight w_i to the representation of state-action $[s_i^e, a_i^e]$ for each timestep i to extract expert's behavior patterns for a_j computation in the next stage; (3) **Stage 3: Trace** the demonstration by recapping the current state and take the optimal action based on the expert behavior patterns. To trace the demonstration effectively, the architecture integrates the current state representation q_j with the weighted expert state-action pairs representations as the expert behavior patterns v''. This is accomplished via a fusion layer that combines the agent's observed state s_j with the expert behavior patterns v'' to form a composite representation c_0. Formally, $c_0 = q_j \oplus v''$, where $\oplus$ denotes a fusion operation, such as concatenation followed by a non-linear transformation or point-wise adding operation. This composite is repeated N times to encapsulate both the current and the expert behavior patterns, i.e., $\forall k \in \{1, 2, \cdots, N\}, c_k = c_{k-1} \oplus v''$, enabling the agent to infer the optimal action a_j based on its related expert behavior patterns.

In summary, demonstration transformer employs a 3-stage process that mirrors human's 3-stage DT process. By integrating an attention mechanism with a specialized network structure, the demonstration transformer framework enhances the agent's ability to generalize from demonstrations to unseen situations.

4.3. Context-based Meta-RL for OSIL

The mere existence of demonstration transformer does not inherently guarantee the adherence to the DT process for decision-making. Conventional OSIL approaches are predominantly based on training with behavior cloning losses (Mandi et al., 2022; Shin et al., 2023), which also fails to guarantee robust decision-making capabilities in unseen states. Drawing inspiration from methodologies that integrate IL with RL through a stationary imitation reward (Ciosek, 2022), we incorporate OSIL into a context-based meta-RL (CbMRL) framework. Within this framework, we can utilize the trial-and-error learning mechanism of RL to allow the imitation policy to systematically explore the state space and effectively achieve decision-making proficiency in unseen states.

However, the stationery reward constructed in (Ciosek, 2022) has some ill-posedness in the OSIL scenario. In this paper, we design an imitation reward in a similar way and empirically fix the ill-posedness of the original imitation reward. We leave the full discussion in App. B and summarize the constructed OSIL reward function as follows:

$$R_{\text{OSIL}}(s, a) := W_{\text{OSIL}}((\bar{s}, \bar{a}), (s, a)) + \alpha R_\omega(s, a), \quad (1)$$

where W_{OSIL} is a stationary imitation function designed for OSIL that indicates the quality of the imitator policy to conduct a in s, $(\bar{s}, \bar{a})$ is the nearest expert state-action pair: $(\bar{s}, \bar{a}) = \arg\min_{(s', a') \in \mathcal{T}} d(s, s')^2$. The selected action $\bar{a}$ corresponds to the action associated with state $\bar{s}$ in the transition pair, and α is a rescale coefficient. $d(\cdot, \cdot)$ measures the distance between two inputs, which can be customized differently for different tasks and is L2 distance in this work. R_ω is the simple task-specific reward mentioned in Sec. 3.

Algorithm 1 Deep Demonstration Tracing

Input: A task set $\mathbb{M}_{\text{train}}$, and a demonstration set $\{\tau_{\omega_i}\}_{i=1}^{|\mathbb{M}_{\text{train}}|}$ for each task $\mathcal{M}_{\omega_i} \in \mathbb{M}_{\text{train}}$
Process:
Initialize the imitator policy Π, and a replay buffer $\mathcal{B}$
for $1, 2, 3, \cdots$ **do**
  Sample a task $\mathcal{M}_\omega$ from the distribution $P(\mathbb{M}_{\text{train}})$ and select the corresponding τ_ω from $\{\tau_{\omega_i}\}_{i=1}^{|\mathbb{M}_{\text{train}}|}$
  for $j = 1, 2, 3, \cdots, H$ **do**
    Sample an action $a_j \sim \Pi(\cdot|s_j, \tau_\omega)$
    Rollout one step: get the next state $s_{j+1} \sim \mathcal{M}_\omega(\cdot|s_j, a_j)$ and the reward $r_j = R_{\text{OSIL}}(s_j, a_j)$
    Add $(s_j, a_j, r_j, s_{j+1}, \tau_\omega)$ to $\mathcal{B}$
  end for
  Use SAC to update Π with batch samples from $\mathcal{B}$
end for

From the standard context-based meta-RL framework, the imitator policy Π can be dissected into a context-based policy π and a task-information extractor ϕ, denoted as $\Pi := \pi(a|s, \phi(\tau_\omega))$. The extractor ϕ is tasked with processing τ_ω to derive task representation denoted by latent variable $z \in \mathcal{Z}$. Meanwhile, π uses the state inputs along with these latent variables to adaptively decide the action for each specific task. The standard objective for optimizing the extractor ϕ and policy π is defined as maximizing the expected discounted sum of rewards over the training task set $\mathbb{M}_{\text{train}}$, formulated as:

$$\max_{\Pi} \mathbb{E}_{\mathcal{M}_\omega \sim P(\mathbb{M}_{\text{train}})} \left[\mathbb{E}_{\mathcal{M}_\omega, \Pi} \left[\sum_{i=0}^{\infty} \gamma^i R_{\text{OSIL}}(s_i, a_i) \right] \right],$$

where $\mathbb{M}_{\text{train}}$ represents the set of training tasks and $\mathbb{E}_{\mathcal{M}_\omega, \Pi}$ denotes the expectation over trajectories sampled from task $\mathcal{M}_\omega$ using policy Π. The imitator policy Π is trained to take optimal actions for all tasks sampled from a distribution $P(\mathbb{M}_{\text{train}})$. If the task set $\mathbb{M}_{\text{train}}$ is representative of the task space Ω, we can assert that the optimally trained policy Π^* will perform correctly on the training set when deployed.

In this work, rather than explicitly modeling π and ϕ, the demonstration transformer architecture is employed as an integrated implementation of the context-based policy and task-information extractor: Π, which is the major dif-

ference compared with the standard context-based meta-RL framework. Finally, we apply the Soft Actor-Critic (SAC) (Haarnoja et al., 2018) for policy optimization and named the whole algorithm **D**eep **D**emonstration **T**raining (DDT) (see Alg. 1). In line 7, we can just store the index of τ_ω in $\{\tau_{\omega_i}\}_{i=1}^{|\mathbb{M}_{\text{train}}|}$ for reduce storage cost. Please refer to App. D for its detailed implementations.

4.4. Theoretical Analysis

In this study, our emphasis revolves around OSIL with a primary focus on a single demonstration. This section aims to illustrate that, under mild conditions, a solitary demonstration is adequate for achieving favorable results.

In the absence of complete coverage of the state-action space, we can construct ill-posed problems such that the unified policy $\Pi(a|s, \tau_\omega)$ cannot imitate successfully (Ross & Bagnell, 2010). The inherent issue arises when the agent deviates from the expert trajectory; it encounters immediate failure and struggles to recover. However, it is important to note that this scenario nearly apply to real-world applications. To capture the distinctive characteristics of real-world tasks, we introduce the concept of *recoverable MDP set*.

Definition 4.1 (Ω-recoverable MDP set)**.** For an MDP set $\mathbb{M} := \{\mathcal{M}_\omega \mid \omega \in \Omega\}$, if there exists a unified goal-conditioned policy $\beta(a|s, g)$, $\forall \mathcal{M}_\omega \in \mathbb{M}$, $\forall \tau_\omega \in \mathcal{T}_w$, we have $\forall s \in \mathcal{S}, \exists g_j \in \tau_\omega$, $\beta(a|s, g_j)$ can reach g_j from $s = s_i$ within finite timesteps, where $\mathcal{S}$ is the state space, i and j denote the timestep of states in τ and $j > i$, then $\mathbb{M}$ is called an Ω-recoverable MDP set.

This kind of MDP set encapsulates situations where, despite deviations from the expert trajectory, the agent has the capability to recover successfully. The assumption has been applied in traditional imitation learning algorithms (Ross et al., 2011) and is practical in many applications, for example, in the task of navigation for parking, we might meet unexpected obstacles and pedestrians in the processing of imitation, which do not exist in the demonstrations. However, for any parking lot, the behaviors to handle the situations are consistent: executing avoidance until the state is safe, then tracing back to the demonstration.

Proposition 4.2 (1-demo imitator availability)**.** *If* $\mathbb{M} := \{\mathcal{M}_\omega \mid \omega \in \Omega\}$ *is an* Ω*-recoverable MDP set, there exists at least a unified imitator policy* $\Pi(a|s, \mathcal{T}_\omega)$ *that can accomplish any task in* $\mathbb{M}$ *only given one corresponding demonstration, i.e.,* $|\mathcal{T}_\omega| = 1$.

The core is the unified goal-conditioned policy β defined in Def. 4.1. The motivation behind β is that, whatever the task we would like to imitate is, and whatever the unexpected changes in the environment will lead the agent to, the behaviors of coming back to the states in the demonstrations are general and consistent. In DDT, β is implicitly implemented in the demonstration transformer architecture and learned by meta-RL.

5. Experiments

In our experiments, we establish a demo-navigation benchmark for OSIL, focusing on navigating diverse, complex mazes without global map information. We present this benchmark in Sec. 5.1 and our experiment setup in Sec. 5.2. In Sec. 5.3, we evaluate our method from various perspectives, including training performance, and various abilities to unseen demonstrations/unexpected situations. We verify the effects of demonstration transformer in Sec. 5.4 and provide ablation studies in Sec. 5.5. We highlight the potential improvements of DDT through scaling up dataset size or parameters in Sec. 5.6. We apply DDT in various complex tasks to show the robustness of our method in other challenges of OSIL in Sec. 5.7.

5.1. Benchmark for OSIL with Unforeseen Changes

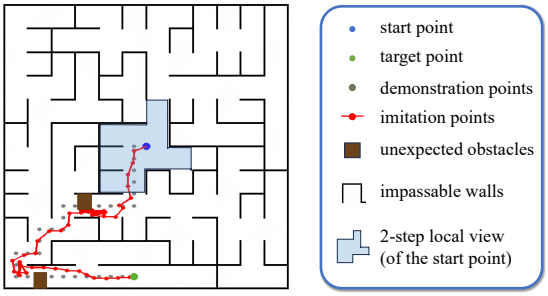


Figure 4: Illustration of the VPAM. The imitation points are provided by our DDT method.

We created a challenging benchmark, named Valet Parking Assist in Maze (VPAM), to assess OSIL's performance for unforeseen changes. This navigation benchmark is inspired by a popular and practical real-world application in autonomous driving, called Valet Parking Assist (VPA) (Heimberger et al., 2017), where the motivation and further details are given in App. E. An illustrative example of VPAM is provided in Fig. 4. In VPAM, a point agent needs to travel from a start point to a target position in a maze, guided by expert demonstrations. The maze and target positions vary between episodes. Obstacle and wall information is directly accessible. Obstacles may be inexistent. The agent relies on its l-step local views to make decisions, where l is a configuration for task setup and its current coordinate is optionally provided. In our experiment, the local view is calculated using 8 rays, each within 5 step length. Without leveraging demonstrations, *finding routes to target positions is impossible due to the lack of global map information*. Besides, for each episode, rectangular obstacles are randomly generated on the path to the target, which may not exist during expert demonstrations. Consequently, the agent cannot blindly follow the demonstration actions without considering the

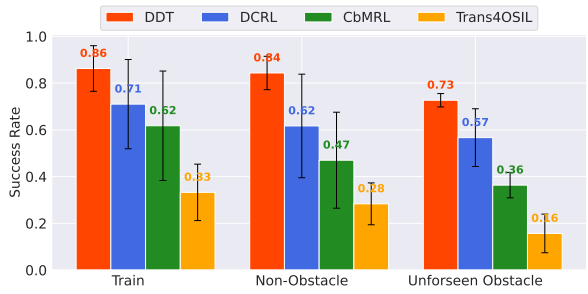


Figure 5: Illustration of the imitation policies' performance deployed among different settings. The black bars denote the standard error among tasks with three seeds.

current situation to reach the target.

5.2. Experiment Setup

Our primary focus is the OSIL capabilities beyond the collected demonstrations. Within VPAM, we create eight tasks by varying three factors: (1) single-map versus multi-map navigation; (2) the presence or absence of obstacles; and (3) whether agent coordinates are provided. For each task, we gather demonstrations targeting different points. To validate the generalization capabilities, we reserve a portion of new demonstrations in each map for testing. In multi-map settings, we separately create new maps to collect demonstrations for evaluation. In Sec. 5.7, we apply DDT in several robotics environments, including Meta-World (Yu et al., 2019), Gymnasium (Towers et al., 2023), and a robot manipulation task (Pang et al., 2023) based on MuJoCo (Todorov et al., 2012). More details are available in App. E.

Baselines We compare DDT with three primary context-based learning approaches that also take demonstrations as inputs: (1) **DCRL** (Dance et al., 2021) embeds demonstrations with standard Transformer and trains policies with task-specific rewards for further improving the expert behavior via RL; (2) **Trans4OSIL** (Dasari & Gupta, 2021) uses Transformer to extract representations from demonstrations and adopts BC for policy reconstruction. (3) **CbMRL** (OpenAI et al., 2019) trains policies only with environment rewards. Demonstrations are embedded with a multi-layer GRU (Cho et al., 2014), which is also the implementation outlined in Alg. 1. All methods are trained for the same duration with the same parameters to ensure fairness. We implement the Transformer of DCRL and Trans4OSIL with the same parameters and layers as our demonstration transformer architecture.

5.3. One-Shot Imitation Ability in Unseen Situations

We present the performance of all methods across various settings with coordinates in Fig. 5 and Fig. 6, while summarizing all experimental results in App. F.

Upon performance on the training set (see the left group in Fig. 5), all algorithms achieve relatively commendable performance in imitating seen demonstrations. Notably, our algorithm, DDT, displayed a distinct performance even in

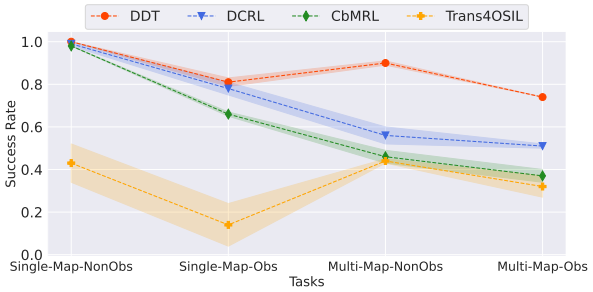


Figure 6: Illustration of the imitation policies' training performance among different settings. The colored areas denote the standard error among the three seeds.

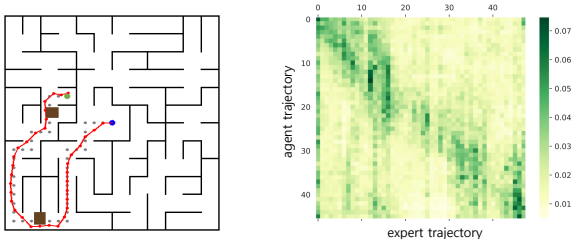
the training task. We attribute this to the integration of the demonstration transformer architecture. This architecture conferred an additional training efficiency boost by implicitly introducing prior knowledge of how OSIL was achieved, facilitating easier adaptation across various tasks and settings of differing complexities. This assertion is further corroborated in Fig. 6, where it is evident that for tasks of minimal complexity, such as `Single-Map-NonObs`, the performance of DCRL and CbMRL was comparable to DDT, with no significant differences. However, as task complexity increased, the performance of DCRL and CbMRL deteriorated significantly, unlike DDT, which remained stable. On another note, Trans4OSIL, employing Behavioral Cloning (BC) directly with action labels, demonstrated robustness across different tasks, albeit with notably inferior performance compared to DDT and other RL-based baselines. For the latter, we postulate that, unlike applications in traditional Trans2OSIL scenarios involving robotic arms, this task presented dead states, i.e., hitting walls or obstacles, where slight imitation errors could lead to failure, making the effects of compounding errors of BC (Ross et al., 2011) larger. Policies trained within the RL paradigm handle this issue.

Given the varying performance levels across algorithms during training, to evaluate the stability of DDT in the face of unforeseen obstacles, we calculated the performance degradation percentages for the four algorithms transitioning from `Train` to `Unforeseen Obstacle` conditions based on Fig. 5. The percentages are as follows: -15%, -20%, -33%, and -52%. It is observable that DDT exhibited at least **5%** better performance retention compared to the baseline algorithms, robustly demonstrating DDT's effective imitation in the presence of unforeseen changes. This robustness, undeniably the core advantage of our algorithm, is attributed to the introduction of the meta-RL mechanism for handling unforeseen changes. This advantage can also be validated by the counterexample that Trans4OSIL consistently struggles with obstacle navigation.

In addition, there is a line of work (Johns, 2021; Valassakis

et al., 2022; Wen et al., 2022) in the robotics learning community that employ parameter-free methods from the perspective of decision-making: Johns (2021); Valassakis et al. (2022) retrieve actions corresponding to matched states (the nearest neighbor) in the single demonstration, while Wen et al. (2022) replays manipulation demonstrations collected from objects of the same category after identifying the pose of the target object. We also make comparisons with these methods as a deployment method. The results, which are provided in App. H.4, show that due to the presence of noise in the target environment, the methods struggle to complete tasks effectively even in scenarios without obstacles, and the results are worse in scenarios with obstacles.

5.4. Demonstration-Attention Mechanism in DDT



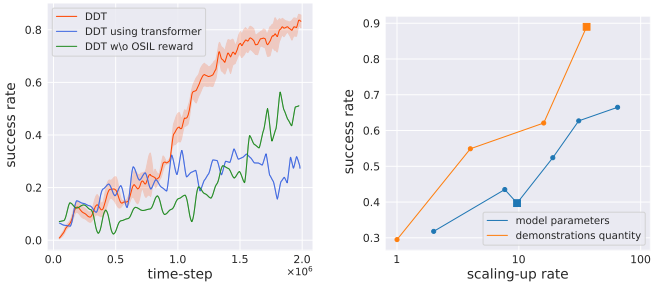
(a) Trajectory of DDT. (b) Attention score.

Figure 7: (a) A trajectory generated by DDT; (b) The attention score map corresponding to (a). The horizontal and vertical axis represent the agent's trajectory index. The deeper color in a row represents the higher attention score.

To further verify that demonstration transformer stimulates the agent making decisions based on the discrepancy between the current state and the states in demonstrations, we visualize the agent trajectory in a randomly generated map with unforeseen obstacles in Fig. 7(a) and depict attention scores during the decision-making process in Fig. 7(b). The attention scores are the product of the query of current state and the keys associated with demonstration states. It is evident that higher attention values are predominantly concentrated on the diagonal, demonstrating our algorithm's ability to identify which state to follow. We provided more visualizations in App. K. Additionally, a corresponding video recording rollouts generated by our DDT method is provided in our project page.

5.5. Ablation Studies

We conduct ablation studies on the demonstration transformer architecture and OSIL rewards in multi-map imitation tasks without obstacles and test in unseen maps, which is depicted in Fig. 8(a). Two DDT variants are constructed: (1) DDT using Transformer, replacing the demonstration transformer with a standard transformer; (2) DDT w/o OSIL reward, where DDT learns with only the ending reward R_ω. Replacing demonstration transformer with a Transformer



(a) Results of DDT variants (b) Scaling up results

Figure 8: (a) Learning curves of DDT variants; (b) Asymptotic performance of DDT under varying demonstration quantities and model parameters, with each unit on the x-axis representing 60 demonstrations or 0.6 million parameters. The x-axis is on a logarithmic scale. *Square markers* depict the performance of the default DDT parameters. The raw results are in App. I.

significantly reduces the asymptotic performance, emphasizing the crucial role of network architecture in enhancing DDT's imitation ability. The OSIL reward is also vital for improving learning efficiency, as OSIL reward enhance the policy learning signals. More ablation studies are in App. H.

5.6. Potential Improvement when Scaling Up

Inspired by recent advances in large decision-making models (Gu et al., 2023; Zhang, 2023), we investigate the potential for further performance improvement when scaling up. Specifically, we train DDT policies with varying demonstration quantities and model parameters in multi-map imitation tasks with obstacles. We only provide coordinates for the former and test the trained policies on unseen maps. Results are visualized in Fig. 8(b), revealing a log-linear performance increase with rising data volume or model parameters. Notably, an increase in model parameters results in approximately $2\times$ performance improvement in Tab. 8. *The results strongly indicate the potential of enhancement with scaling up, prompting further investigation of DDT as a skeleton for generalist agents (Reed et al., 2022).*

5.7. Apply DDT in Other Challenging Tasks

We evaluate the generalization ability of our method to imitate unseen demonstrations across a range of challenging tasks. We deploy DDT in 4 tasks from a well-known benchmark **Meta-World** (Yu et al., 2019) with self-built disturbance. Then we apply DDT in robot tasks requiring **complex planning** (Pang et al., 2023), involving object grasping, stacking, and collecting in clutter environments; **complex controller** with the Reacher and Pusher in (Towers et al., 2023), which feature diverse variables for control, including location, velocity, angular velocity, etc. We also test in our VPAM benchmark by omitting coordinates, pre-

senting a **partially observable** decision-making problem where local views within a single state do not suffice for imitation. Details about the tasks are available in App. E. We summarize the average results in Tab. 1 and Tab. 2. Please refer to Tab. 9 and Tab. 10 in App. F for complete results.

Table 1: Performance comparison on Meta-World.

Tasks	W/o Disturbance		W/ Disturbance	
Demo	seen	unseen	seen	unseen
DDT	**1.00±0.00**	**0.85±0.14**	**0.83±0.15**	**0.61±0.20**
DCRL	0.73±0.42	0.37±0.27	0.23±0.30	0.12±0.14
Trans4OSIL	0.14±0.21	0.11±0.12	0.00±0.00	0.00±0.00
CbMRL	0.70±0.32	0.39±0.36	0.08±0.07	0.10±0.06

Performance under Disturbance in Meta-world: From the result in Meta-World (Tab. 1), baseline methods mainly succeeded in imitating previously seen trajectories, showing a significant performance drop for unseen samples, even without disturbances. This drop was exacerbated in scenarios with disturbances, where baselines nearly failed to operate. Conversely, our method was effective in all scenarios, underscoring its generalizability for OSIL.

Table 2: Performance comparison on various tasks.

Tasks	Complex Planning		Complex Controller		Partially Observable	
Demo	seen	unseen	seen	unseen	seen	unseen
DDT	**0.91±0.10**	**0.76±0.11**	**0.97±0.01**	**0.95±0.01**	**0.67±0.16**	**0.67±0.04**
DCRL	0.10±0.14	0.23±0.33	0.77±0.12	0.69±0.19	0.20±0.05	0.03±0.02
Trans4OSIL	0.15±0.12	0.09±0.08	0.42±0.22	0.33±0.25	0.06±0.00	0.01±0.01
CbMRL	0.24±0.33	0.16±0.23	0.91±0.01	0.86±0.01	0.15±0.00	0.03±0.01

Complex Decision-making Tasks: Other experiments are summarized in Tab. 2. Conducted with *uniform parameters and network architecture* without task-specific tuning, our approach demonstrated excellent training performance and generalization across different tasks. This underscores the method's versatility. We attribute it to the demonstration transformer architecture and the Meta-RL framework's inherent suitability for OSIL tasks. Notably, our method does experience a reduction in training performance in partially observable scenarios, aligning with our expectations due to the inherent challenges of such settings. Future work could incorporate visual OSIL techniques (Mandi et al., 2022) for addressing partially observable problems in DDT, potentially enhancing its problem-solving capabilities. Nevertheless, DDT maintained commendable generalization from training to testing within its learned expertise.

Training on Mixed-Up Environments We validate our algorithm's performance on mixed-up demonstrations from different environments to enhance the significance of our algorithm. We collect demonstrations from varying numbers of environments, with selected environment details provided in App. H. We report the performance of these demonstrations according to the number of environments designed. The results are summarized in Tab. 3. The results reveal the strong adaptability of our method for learning tasks across diverse environments, consistently performing well regardless of the number of environments involved. We also test our algorithm on learning four environments individually, achieving a 0.85 success rate, to better illustrate the robustness of DDT.

Table 3: Performance on mixed-up demonstrations.

Environment number	2	4	8
Seen Demos.	1.00	1.00	1.00
Unseen Demos.	0.84	0.88	0.90

Deploying on Unseen Environments We finally discuss that generalizing tasks to entirely different environments, e.g., different tasks, embodiments, or camera views, rather than unexpected situations, would be somewhat beyond the scope. However, we provide this part of the experimental results to better showcase the potential of DDT. We select the eight objects-manipulation tasks in Meta-World for training, including Button Press Topdown, Button Press Topdown Wall, Button Press Wall, Door Open, Faucet Close, Drawer Close, Window Open, and Window Close. The details are provided in App. H. Then we test and record the generalization performance on three unseen environments, Button Press, Door Close, and Reach without fine-tuning. It can be observed that even when the agent is trained without corresponding task environments, our algorithm still achieves satisfactory success rates by consciously imitating demonstrations.

Table 4: Performance on unseen environments.

Environment	Button Press	Door Close	Reach
Performance	0.78	1.00	0.75

6. Discussion and Future Work

We propose a problem of one-shot imitation learning with unforeseen changes after demonstration collections and a practical *Deep Demonstration Tracing* (DDT) algorithm. DDT leverages a specialized demonstration-based attention architecture to encourage agents to adaptively trace suitable states in demonstrations. We apply DDT to both demo-navigation tasks and robotics tasks. The results demonstrate that DDT outperforms previous OSIL methods with large margins both on training and unseen-tasks testing.

We believe that OSIL with unforeseen changes is a valuable topic for the community, which makes OSIL algorithms generalizable to more potential real-world applications. Besides, our scaling-up experiments also highlight the potential of DDT in solving larger-scale problems, e.g., generalist imitator agents, which will be in our future work.

Acknowledgements

This work is supported by the National Science Foundation of China (61921006). The authors thank anonymous reviewers for their helpful discussions and suggestions for improving the article.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Ahn, J., Kim, M., and Park, J. Vision-based autonomous driving for unstructured environments using imitation learning. *arXiv preprint arXiv:2202.10002*, 2022.

Chen, J., Yuan, B., and Tomizuka, M. Deep imitation learning for autonomous driving in generic urban scenarios with enhanced safety. In *International Conference on Intelligent Robots and Systems*, pp. 2884–2890, 2019.

Chen, X., Yu, Y., Li, Q., Luo, F., Qin, Z. T., Shang, W., and Ye, J. Offline model-based adaptable policy learning. In *Advances in Neural Information Processing Systems*, pp. 8432–8443, 2021.

Chen, X., He, B., Yu, Y., Li, Q., Qin, Z. T., Shang, W., Ye, J., and Ma, C. Sim2rec: A simulator-based decision-making approach to optimize real-world long-term user engagement in sequential recommender systems. In *IEEE International Conference on Data Engineering*, 2023a.

Chen, X., Luo, F., Yu, Y., Li, Q., Qin, Z., Shang, W., and Ye, J. Offline model-based adaptable policy learning for decision-making in out-of-support regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45 (12):15260–15274, 2023b.

Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. In *Workshop on Syntax, Semantics and Structure in Statistical Translation*, pp. 103–111, 2014.

Ciosek, K. Imitation learning by reinforcement learning. In *International Conference on Learning Representations*, 2022.

Dance, C. R., Perez, J., and Cachet, T. Demonstration-conditioned reinforcement learning for few-shot imitation. In *International Conference on Machine Learning*, pp. 2376–2387, 2021.

Dasari, S. and Gupta, A. Transformers for one-shot visual imitation. In *Conference on Robot Learning*, pp. 2071–2084, 2021.

Domingos, P. A few useful things to know about machine learning. *Communications of the ACM*, pp. 78–87, 2012.

Duan, Y., Andrychowicz, M., Stadie, B., Jonathan Ho, O., Schneider, J., Sutskever, I., Abbeel, P., and Zaremba, W. One-shot imitation learning. *Advances in Neural Information Processing Systems*, pp. 1087–1098, 2017.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pp. 1126–1135, 2017a.

Finn, C., Yu, T., Zhang, T., Abbeel, P., and Levine, S. One-shot visual imitation learning via meta-learning. *Conference on Robot Learning*, pp. 357–368, 2017b.

Florensa, C., Held, D., Geng, X., and Abbeel, P. Automatic goal generation for reinforcement learning agents. In *International Conference on Machine Learning*, pp. 1514–1523, 2018.

Fujimoto, S. and Gu, S. S. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 20132–20145, 2021.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1587–1596, 2018.

Gu, J., Kirmani, S., Wohlhart, P., Lu, Y., Arenas, M. G., Rao, K., Yu, W., Fu, C., Gopalakrishnan, K., Xu, Z., Sundaresan, P., Xu, P., Su, H., Hausman, K., Finn, C., Vuong, Q., and Xiao, T. Rt-trajectory: Robotic task generalization via hindsight trajectory sketches. *arXiv preprint arXiv:2311.01977*, 2023.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pp. 1856–1865, 2018.

Heimberger, M., Horgan, J., Hughes, C., McDonald, J., and Yogamani, S. K. Computer vision in automated parking systems: Design, implementation and challenges. *Image and Vision Computing*, 2017.

Hester, T., Vecerík, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Osband, I., Dulac-Arnold, G., Agapiou, J. P., Leibo, J. Z., and Gruslys, A. Deep q-learning from demonstrations. In *AAAI Conference on Artificial Intelligence*, pp. 3223–3230, 2018.

Johns, E. Coarse-to-fine imitation learning: Robot manipulation from a single demonstration. In *2021 IEEE international conference on robotics and automation*, pp. 4613–4619. IEEE, 2021.

Kümmerle, R., Hähnel, D., Dolgov, D., Thrun, S., and Burgard, W. Autonomous driving in a multi-level parking structure. In *International Conference on Robotics and Automation*, pp. 3395–3400, 2009.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, pp. 2278–2324, 1998.

Li, J., Lu, T., Cao, X., Cai, Y., and Wang, S. Meta-imitation learning by watching video demonstrations. In *International Conference on Learning Representations*, 2021.

Luo, F., Jiang, S., Yu, Y., Zhang, Z., and Zhang, Y. Adapt to environment sudden changes by learning a context sensitive policy. In *AAAI Conference on Artificial Intelligence*, pp. 7637–7646, 2022.

Mandi, Z., Liu, F., Lee, K., and Abbeel, P. Towards more generalizable one-shot visual imitation learning. In *International Conference on Robotics and Automation*, pp. 2434–2444, 2022.

Nagabandi, A., Clavera, I., Liu, S., Fearing, R. S., Abbeel, P., Levine, S., and Finn, C. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018.

Nagabandi, A., Clavera, I., Liu, S., Fearing, R. S., Abbeel, P., Levine, S., and Finn, C. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *International Conference on Learning Representations*, 2019.

Nair, S., Savarese, S., and Finn, C. Goal-aware prediction: Learning to model what matters. In *International Conference on Machine Learning*, pp. 7207–7219, 2020.

Ng, A. Y. and Russell, S. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, pp. 663–670, 2000.

OpenAI, Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., Schneider, J., Tezak, N., Tworek, J., Welinder, P., Weng, L., Yuan, Q., Zaremba, W., and Zhang, L. Solving Rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

Pan, Y., Cheng, C., Saigol, K., Lee, K., Yan, X., Theodorou, E. A., and Boots, B. Agile autonomous driving using end-to-end deep imitation learning. In *Robotics: Science and Systems*, 2018.

Pang, J.-C., Yang, S.-H., Chen, X.-H., Yang, X., Yu, Y., Ma, M., Guo, Z., Yang, H., and Huang, B. Object-oriented option framework for robotics manipulation in clutter. In *International Conference on Intelligent Robots and Systems*, pp. 1230–1237, 2023.

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. Sim-to-Real transfer of robotic control with dynamics randomization. In *International Conference on Robotics and Automation*, pp. 1–8, 2018.

Pomerleau, D. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, pp. 88–97, 1991.

Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Robotics: Science and Systems*, 2018.

Rakelly, K., Zhou, A., Finn, C., Levine, S., and Quillen, D. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International Conference on Machine Learning*, pp. 5331–5340, 2019.

Reed, S. E., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., Eccles, T., Bruce, J., Razavi, A., Edwards, A., Heess, N., Chen, Y., Hadsell, R., Vinyals, O., Bordbar, M., and de Freitas, N. A generalist agent. *Transactions on Machine Learning Research*, 2022.

Ross, S. and Bagnell, D. Efficient reductions for imitation learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 661–668, 2010.

Ross, S., Gordon, G. J., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 627–635, 2011.

Shin, S., Lee, D., Yoo, M., Kim, W. K., and Woo, H. One-shot imitation in a non-stationary environment via multi-modal skill. In *International Conference on Machine Learning*, pp. 31562–31578, 2023.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction (Second Edition)*. 2018.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Towers, M., Terry, J. K., Kwiatkowski, A., Balis, J. U., Cola, G. d., Deleu, T., Goulão, M., Kallinteris, A., KG, A., Krimmel, M., Perez-Vicente, R., Pierré, A., Schulhoff,

S., Tai, J. J., Shen, A. T. J., and Younis, O. G. Gymnasium, 2023.

Valassakis, E., Papagiannis, G., Di Palo, N., and Johns, E. Demonstrate once, imitate immediately (dome): Learning visual servoing for one-shot imitation learning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 8614–8621. IEEE, 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Wang, K., Kang, B., Shao, J., and Feng, J. Improving generalization in reinforcement learning with mixture regularization. In *Advances in Neural Information Processing Systems*, pp. 7968–7978, 2020.

Wen, B., Lian, W., Bekris, K. E., and Schaal, S. You only demonstrate once: Category-level manipulation from single visual demonstration. *ArXiv*, abs/2201.12716, 2022.

Xie, F., Chowdhury, A., Kaluza, M. C. D. P., Zhao, L., Wong, L. L. S., and Yu, R. Deep imitation learning for bimanual robotic manipulation. In *Advances in Neural Information Processing Systems*, pp. 2327–2337, 2020.

Xu, T., Li, Z., and Yu, Y. Error bounds of imitating policies and environments. In *Advances in Neural Information Processing Systems*, 2020.

Yeh, J., Chung, C., Su, H., Chen, Y., and Hsu, W. H. Stage conscious attention network (SCAN): A demonstration-conditioned policy for few-shot imitation. In *AAAI Conference on Artificial Intelligence*, pp. 8866–8873, 2022.

Yu, T., Abbeel, P., Levine, S., and Finn, C. One-shot hierarchical imitation learning of compound visuomotor tasks. *arXiv preprint arXiv:1810.11043*, 2018a.

Yu, T., Finn, C., Xie, A., Dasari, S., Zhang, T., Abbeel, P., and Levine, S. One-shot imitation from observing humans via domain-adaptive meta-learning. *arXiv preprint arXiv:1802.01557*, 2018b.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. *arXiv preprint arXiv:1910.10897*, 2019.

Zhang, D., Wu, Z., Chen, J., Zhu, R., Munawar, A., Xiao, B., Guan, Y., Su, H., Hong, W., Guo, Y., Fischer, G. S., Lo, B., and Yang, G. Human-robot shared control for surgical robot based on context-aware sim-to-real adaptation. In *2022 International Conference on Robotics and Automation, ICRA'22*, pp. 7694–7700. IEEE, 2022.

Zhang, W. Large decision models. In *International Joint Conference on Artificial Intelligence*, pp. 7062–7067, 2023.

A. Demonstration Quantity Requirements for Imitator Learning

Without further assumptions on task space Ω, it is always easy to construct ill-posed problems that it is impossible for a unified imitator policy $\Pi(a|s, \mathcal{T}_\omega)$ to reconstruct all of the expert policies unless the expert demonstration set $\mathcal{T}$ does cover the whole state-action space. However, in many applications, it is unnecessary for Π to imitate policies for any $\mathcal{M}$. Here we give one practical task set that enables imitator learning (OSIL) through only one demonstration.

Definition A.1 (Ω-recoverable MDP set)**.** For an MDP set $\mathbb{M} := \{\mathcal{M}_\omega \mid \omega \in \Omega\}$, if there exists a unified goal-conditioned policy $\beta(a|s, g)$, $\forall \mathcal{M}_\omega \in \mathbb{M}$, $\forall \tau_\omega \in \mathcal{T}_w$, we have $\forall s \in \mathcal{S}, \exists g_j \in \tau_\omega$, $\beta(a|s, g_j)$ can reach g_j from $s = s_i$ within finite timesteps, where $\mathcal{S}_0$ is the set of the initial states, i and j denote the timestep of states in τ and $j > i$, then $\mathbb{M}$ is an Ω-recoverable MDP set.

Ω-recoverable MDP set depicts the similarity of the tasks in $\mathbb{M}$ through the demand of the policy $\beta(a|s, g)$. It means that although the transition process and the initial distribution are stochastic and different among $\mathbb{M}$, there exists a goal-conditioned policy β that for any $\mathcal{M}_\omega$, we can guide the agent turn back to some states in the demonstrations. For example, different navigation tasks will have similar decisions in similar traffic conditions even in different terrains. Thus even if the vehicle has to veer off the demonstrations for handling some unexpected situations, it usually can turn back after some timesteps.

Based on the definition, we give an $\mathbb{M}$ formulation that can find a unified imitator policy Π from one demonstration.

Proposition A.2 (1-demo imitator availability)**.** *If $\mathbb{M} := \{\mathcal{M}_\omega \mid \omega \in \Omega\}$ is a Ω-recoverable MDP set, there exists at least a unified imitator policy $\Pi(a|s, \mathcal{T}_\omega)$ that can accomplish any task in $\mathbb{M}$ only given one corresponding demonstration, i.e., $|\mathcal{T}_\omega| = 1$.*

Proof. Since R_ω in $\mathcal{M}_\omega$ is an ending reward function of trajectories, given any expert demonstration τ_ω, we know:

$$R_\omega(s, a) = \begin{cases} c, & s = s_t, \\ 0, & (s, a) \in \tau_\omega \text{ and } s \neq s_t, \\ \text{unkown}, & \text{otherwise.} \end{cases}$$

That is, any policy can accomplish the task in $\mathcal{M}_\omega$ if it can reach the last state s_t of τ_ω, where c is the reward for accomplishing the task.

Since $\mathbb{M} := \{\mathcal{M}_\omega \mid \omega \in \Omega\}$ is a Ω-recoverable MDP set, there exists a unified goal-conditioned policy $\beta(a|s, g)$, $\forall \mathcal{M}_\omega \in \mathbb{M}$, $\forall \tau_\omega \in \mathcal{T}_w$, we have $\forall s \in \mathcal{S}, \exists g_j \in \tau_\omega$, $\beta(a|s, g_j)$ can reach g_j from $s = s_i$ within finite timesteps. We can construct a unified imitator policy by (1) searching a $g_j \in \tau_\omega$ that can be reached by $\beta(a|s, g_j)$ from current state s_i within finite timesteps, where $j > i$; (2) executing $\beta(a|s, g_j)$ until reaching g_j; (3) repeat (1) and (2) to the end. When deployed, for any $\mathcal{M}_\omega$, in the beginning, $s_0 \sim d_0$, thus the agent will reach one of the state $s_i \in \tau_\omega$ after finite timesteps, where $i > 0$. Since $s_i \in \tau_\omega$, following $\beta(a|s, g_j)$, the agent will arrive another state $s_j \in \tau_\omega$. The process will be repeated until the agent reaches the last state s_t. Once $R_\omega(s_t, \cdot) = c$, the task is accomplished. □

Although we focus on 1-demo imitator availability, note that the 1-demo imitator availability can be extended to the "n-demo" case by extending Ω-recoverable MDP set to " $\mathcal{T}_\Omega$-recoverable MDP set".

Fig. 9 gives a vehicle navigation illustration for the proposition, where all tasks in $\mathbb{M}$ ask the vehicle to reach some locations based on its coordinates and local views. We first consider a simple case in which the initial state is deterministic and is the same as the first state in τ_ω. In this case, even if a truck might be parked unexpectedly (states unvisited in the demonstrations), relying on the local-view information, for any τ_ω, we have a unified goal-conditioned policy $\beta(a|s, g)$, i.e., closing to some of the successor expert states without collision, that can drive the vehicle to be close to the locations in τ_ω. With policy β, there exists at least a unified imitator policy $\Pi(a|s, \mathcal{T}_\omega)$ that can accomplish any task in $\mathbb{M}$ only given one corresponding demonstration: repeatedly traces a reachable successor state $g_j \in \tau_\omega$ and uses β to guide the agent until reaching the ending state. In the following, we consider a counter-example where the agent state can be put to unrecoverable states, e.g., the square point in Fig. 9, for some unforeseen reasons. In this case, if the local view is limited and cannot reach the location of entrances and the entrance might exist *either* in A or B, it is impossible to construct a unified goal-conditioned policy $\beta(a|s, g)$ since in the square point, the correct way to trace back to the demonstrations is agnostic (can be in left or right).

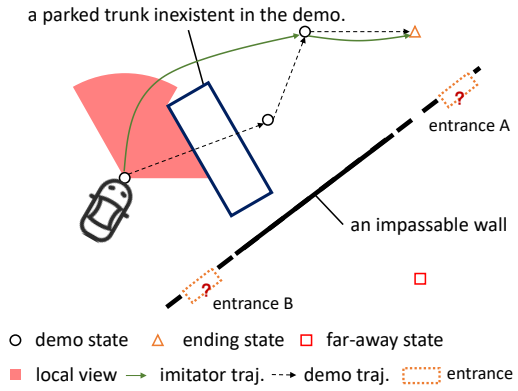


Figure 9: A vehicle navigation example. "traj." is the abbreviation of "trajectory".

Note that the trace-backable property relies on the information we have from the states, e.g., with global map information in the state space, the unified goal-conditioned policy can be constructed by planning a trajectory in the map then the above task set is trace-backable.

B. OSIL Reward from Demonstrations

A theoretical analysis in (Ciosek, 2022) shows that, for deterministic experts, IL can be done by RL with a constructed stationary reward: $R_{\text{int}}(s,a) = \mathbb{I}[(s,a) \in \mathcal{T}]$, where $\mathbb{I}[\cdot]$ denotes the indicator function and $\mathcal{T}$ is the expert demonstration. In practice, the constructed reward function:

$$R_{\text{IL}}(s,a) = 1 - \min_{(s',a')\in\mathcal{T}} d_{\ell_2}((s,a),(s',a'))^2, \tag{2}$$

which is a practical imitation reward R_{int} that can also imitate the experts in several benchmark tasks. Here $d_{\ell_2}(\cdot,\cdot)$ denotes the ℓ_2 distance of two normalized vectors.

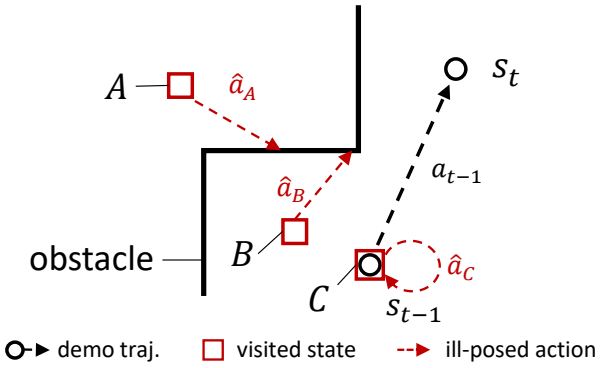


Figure 10: Illustration of the ill-posedness of R_{IL}. A, B, and C denote states, and red dashed arrows ($\hat{a}_A$, $\hat{a}_B$, and $\hat{a}_C$) denote the corresponding ill-posed sub-optimal actions to earn more the cumulative R_{IL} rewards. The agent fails on hitting the obstacle. s_t is the last state, also the target state for task completion.

Inspired by this, we propose to construct a stationary imitator reward R_{OSIL} to embed IL into the RL process, i.e., replacing the reward function R_ω in Alg. 1 (Line 7) with R_{OSIL}. First, we observe that R_{IL} and R_{int} can reconstruct the expert policy only when we have a diverse enough dataset which covers the state-action space. When only with one demonstration, the reward function will be ill-posed in three aspects. We depict that based on the illustration in Fig. 10: (1) A state: if the minimum-distance tuple in Eq. 2 is far away from the visited state, e.g., (s_{t-1}, a_{t-1}) in Fig. 10, the action that reduces the ℓ_2-norm between the next state and s_{t-1} might ignore the impassable terrains between states and finally hit the obstacle; (2) B state: even if the action to reduce the ℓ_2-norm between the next state of B and s_{t-1} is correct to go back to the

demonstration, to reduce the ℓ_2-norm between actions a_{t-1} and the current action at the same time, the derived action might be biased by a_{t-1} and finally lead to an unsafe state; (3) C state: even if the state perfectly matches the one in the demonstration, the agent still has the potential to stay where it is until it reaches the maximum episode length, as R_{IL} might be greater than 0. To handle the above problems, we construct a new imitator reward function $R_{\text{OSIL}}(s, a)$ via:

$$R_{\text{OSIL}}(s,a) := \underbrace{1 - \min\left\{d(\bar{s},s)^2 + \frac{d(\bar{a},a)^2}{\exp(d(\bar{s},s)^2)}, \eta\right\}}_{W_{\text{OSIL}}} + \alpha R_\omega(s,a), \tag{3}$$

where $(\bar{s}, \bar{a}) = \arg\min_{(s',a')\in\mathcal{T}} d(s,s')^2$, η is a hyperparameter that clips the distance penalty calculated based on the too-far state pairs into a fixed constant, $\alpha = 1/(c(1-\gamma))$ is a rescale coefficient, and c is the reward for accomplishing the task defined in R_ω.

$R_{\text{OSIL}}(s, a)$ uses a clipping term η to make the imitation rewards based on the too-far state pairs invalidated to avoid the potential misleading (to handle the "A-state" case). A reweighting item $1/\exp(d(\bar{s}, s)^2)$ is used for the action's distance computation to adaptively adjust the weight of rewards on action matching. This is a heuristic reweighting term to avoid the agent overly penalizing for not strictly following the expert action when its current state is far from the demonstration states and chooses to turn back (to handle the "B-state" case). The necessity of the reweighting term stems from its pivotal role in preventing undesired behaviors in situations where the agent strictly adheres to the demonstrated actions due to state bias. By incorporating the reweighting term, we ensure that the agent does not blindly follow the demonstrations, thereby reducing the risk of unintended consequences. α rescales the ending rewards, which makes the discount on delay to get the ending reward larger than the bonus of repeatedly collecting the immediate rewards, i.e., $\alpha c > 1 - \epsilon + \gamma\alpha c$, where ϵ denotes a larger-than-zero penalty contributed by the second item in Eq. 3 (to handle the "C-state" case).

Note that although we give several tricks to make R_{OSIL} give reasonable rewards in the state-action space, it is still inevitable to output ill-posed rewards in some corner cases. Hence, the ending reward is essential, as it helps the agent focus more on task completion rather than repeatedly collecting R_{IL} rewards. The large coefficient α on the task-specific reward R_ω makes the policies always focus on completing the tasks rather than repeatedly collecting R_{IL} rewards. In this situation, R_{OSIL} just serves as a crucial signal by providing a dense reward, enabling the agent to closely follow the demonstrations and accomplish tasks effectively during the early stages. We leave a theoretical-grounded reward function design as future work.

C. Related Work

C.1. Imitation Learning

Imitation learning (IL) focuses on training a policy with action labels from expert demonstrations. There are two mainstream approaches for IL, namely behavior cloning (BC) (Pomerleau, 1991; Ross & Bagnell, 2010; Xu et al., 2020) and inverse reinforcement learning (IRL) (Ng & Russell, 2000). The former BC converts IL into a supervised paradigm by minimizing the action probability discrepancy with Kullback Leibler (KL) divergence between the actions of the imitating policy and the demonstration actions. The latter IRL fashion learns the hidden reward function behind the expert policy to avoid the impact of compounding errors.

Since IL can learn directly from already collected data, it is widely adopted by complex domains like game playing (Ross & Bagnell, 2010), autonomous driving (Chen et al., 2019; Pan et al., 2018), and robot manipulation (Xie et al., 2020). Although achieving impressive performances, we observe that in many applications, what humans require is the ability to perform many different tasks out of the box, through very limited demonstrations of corresponding tasks, instead of imitating from scratch based on a mass of demonstrations. Adapting the trained policy to unseen tasks is beyond the capability of pure IL, which is designed for single-task learning.

C.2. Meta-Imitation Learning

Meta-IL includes few-shot meta-IL and one-shot meta-IL. Few-shot meta-IL aims to get a generalizable policy that can complete new tasks with only a few expert trajectories. The mainstream solutions utilize model-agnostic meta-learning (MAML) (Finn et al., 2017a) to learn initial task parameters and fine-tune them via a few steps of gradient descent to satisfy new task needs (Finn et al., 2017b; Li et al., 2021; Yu et al., 2018b). However, these approaches need extra computation infrastructure for gradient update and determining a suitable amount of fine-tuning steps before deployment (Finn et al., 2017a). One-shot meta-IL achieves generalizable imitation through context-based policy models (Dasari & Gupta, 2021;

Duan et al., 2017; Mandi et al., 2022), such as Transformer (Vaswani et al., 2017), that take demonstrations as input. The core idea is to extract representations of demonstrations through these powerful fitting abilities of neural networks, then use BC to reconstruct the imitation policy. However, the demonstrations for imitation are limited, the inevitable prediction errors on unseen states and the compounding errors of BC (Ross et al., 2011) hurt the capacities of these methods, especially in generalizing to new tasks (Mandi et al., 2022).

C.3. Combination of Imitation Learning and Reinforcement Learning

We are not the first study to combine IL and RL. Previous studies have combined these for different proposes: Hester et al. (2018) leverage small sets of demonstrations for deep Q-learning which massively accelerates the learning process. Rajeswaran et al. (2018) use demonstrations to reduce the sample complexity of learning the dexterous manipulation policy and enable natural and robust robot movement. Fujimoto & Gu (2021) add BC to the online RL algorithm TD3 (Fujimoto et al., 2018) for advanced offline RL performance. Our method extends the idea of combining IL and RL to handle OSIL with unforeseen changes.

C.4. Context-based Meta Reinforcement Learning

Besides IL, context-based policy models are also widely used in meta-RL. Building a representative context enables a single agent of learning meta-skills and identifying new tasks. Goal-conditioned RL (Florensa et al., 2018; Nair et al., 2020) is the most direct way to build a context-based meta-policy, which scales a single agent to a diverse set of tasks by informing the agent of the explicit goal contexts, e.g., the target to go or the object to pick. The demonstrations can be regarded as an informative "goal" for IL tasks. The demonstration sequence not only tells the agent which task to accomplish but also the way to accomplish it.

Some other works collect interaction trajectories from the environment for understanding the task identity. (Peng et al., 2018; OpenAI et al., 2019; Chen et al., 2023a; Nagabandi et al., 2019; Chen et al., 2021; Luo et al., 2022; Chen et al., 2023b; Zhang et al., 2022) use a end-to-end architecture for environment-parameter representation and adaptable policy learning. A recurrent neural network is introduced for environment-parameter representation, then the context-aware policy takes actions based on the outputs of RNN and the current states. (Rakelly et al., 2019) share the same end-to-end architecture and design a new neural network to represent the probabilistic latent contexts of the environment parameters. Instead of collecting trajectories from the environment for identifying the task, we mine the information from the static expert trajectories to identify the expert policy which can accomplish the task.

Demonstration-conditioned RL (DCRL) (Dance et al., 2021) takes sub-optimal demonstrations as input and seeks to further improve demonstration behavior via RL. Yeh et al. (2022) adopt a similar idea to solve unseen compound robot tasks that contain multiple stages by retrieving from demonstrations. Instead of taking demonstration as the base for policy improvement, OSIL aims to fully utilize the demonstrations to imitate the expert policy for each task.

D. Implementation Details

D.1. Achitecture Details

We give the related hyper-parameters of the architecture in Tab. 5.

In demonstration transformer architecture, we introduce three encoders for expert actions, expert states, and visited states respectively, where the encoders of expert states and visited states share the same weights. The detailed architecture is shown in Fig. 11.

Table 5: DDT Hyper-parameters.

Parameter	Value
learning rate (λ)	$5 \cdot 10^{-5}$
discount (γ)	0.99
replay buffer size	10^5
number of hidden units per layer	256
number of samples per minibatch	256
optimizer	RMSprop
Actor	
encoder layer number (K)	3
cross-attention layer number (N)	6
embedding dimension	128
Critic	
encoder layer number (K)	4
cross-attention layer number (N)	4
embedding dimension	128
OSIL Rewards for VPAM	
rescale coefficient (α)	100
penalty threshold (η)	2
OSIL Rewards for Robotics tasks	
rescale coefficient (α)	200
penalty threshold (η)	2

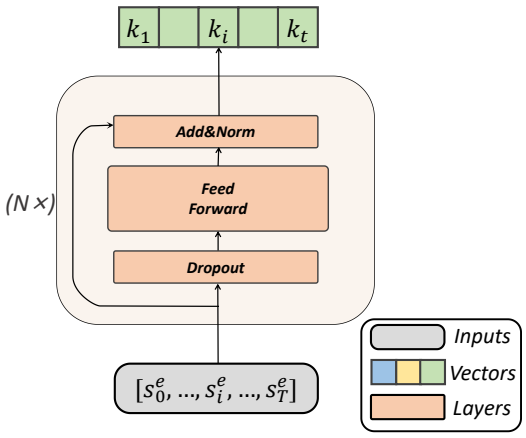


Figure 11: The encoder architecture employed in the demonstration transformer. As an example, we consider the input sequence $[s_0^e, ..., s_i^e, ..., s_t^e]$. However, it is worth noting that this architecture can also accommodate $[a_0^e, ..., a_i^e, ..., a_t^e]$ as inputs. The encoder leverages the Transformer backbone, which incorporates three layers: dropout, feedforward, and add&norm. These layers are organized using the residual connection mechanism. The input sequence passes through N stacked blocks, converting it into key vectors.

D.2. Training Details

Algorithm 2 Demo-Attention Actor-Critic for OSIL

Input: A task set $\mathbb{M}_{\text{train}}$, and a demonstration set $\{\mathcal{T}_{\omega_i}\}$ for each task $\mathcal{M}_{\omega_i} \in \mathbb{M}_{\text{train}}$
Process:
Initialize Actor Π_ψ, Critic Q_θ and a replay buffer $\mathcal{B}$
for $1, 2, 3, ...$ **do**
  Sample a task $\mathcal{M}_\omega$ from the sampling strategy $P(\mathbb{M}_{\text{train}})$
  Get the *single* expert trajectory τ_ω since $|\mathcal{T}_\omega| = 1$
  for $j = 1, 2, 3, ..., H$ **do**
    Sample an action $a_j \sim \Pi_\psi(a|s_j; \tau_\omega)$
    Rollout one step $s_{j+1} \sim \mathcal{M}_\omega(s|s_j, a_j)$, get the reward $r_j = R_{\text{OSIL}}(s_j, a_j)$
    Add $(s_j, a_j, r_j, s_{j+1}, \tau_\omega)$ to $\mathcal{B}$
  end for
  for each update step **do**
    Update Critic $\theta \leftarrow \theta - \lambda \nabla J_Q(\theta)$
    Update Actor $\psi \leftarrow \psi - \lambda \nabla J_\Pi(\psi)$
  end for
end for

We use the SAC (Haarnoja et al., 2018) algorithm to update the demonstration-transformer actor and demonstration-transformer critic. The goal of SAC also maximizes the expected entropy return beyond the objective of a standard RL agent which maximizes the expected sum of rewards:

$$J(\Pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t, a_t) \sim \rho_\Pi}[r(s_t, a_t) + \alpha \mathcal{H}(\Pi(\cdot\,|s_t))], \tag{4}$$

where $\mathcal{H}(\Pi(\cdot\,|s_t))$ is the entropy value of the policy distribution. For learning the maximum entropy, a policy alternates between policy evaluation and policy improvement. For policy evaluation of a fixed policy, we can obtain its soft state value function by iteratively applying the Bellman update:

$$V(s_t) = \mathbb{E}_{a_t \sim \Pi}[Q(s_t, a_t; \tau_\omega) - \alpha \log \Pi(a_t|s_t; \tau_\omega)]. \tag{5}$$

And we can execute critic updates through collected buffer data and the objective:

$$J_Q(\theta) = \mathbb{E}_{(s_t, a_t, \tau_\omega) \sim \mathcal{D}} \left[\frac{1}{2} \left(Q_\theta\left(s_t, a_t; \tau_\omega\right) - \hat{Q}\left(s_t, a_t\right) \right)^2 \right], \tag{6}$$

with

$$\hat{Q}(s_t, a_t) = r_t + \gamma \mathbb{E}_{s_{t+1}}[V(s_{t+1})], \tag{7}$$

and we can execute policy improvement through collected buffer data and the objective:

$$J_\Pi(\psi) = \mathbb{E}_{s_t \sim \mathcal{B}} \mathrm{D}_{\mathrm{KL}} \left(\Pi_\psi(\cdot\,|s_t; \tau_\omega) \,\middle\|\, \frac{\exp\left(Q_\theta(s_t, \cdot\,; \tau_\omega)\right)}{Z_\theta(s_t)} \right), \tag{8}$$

where the partition function $Z_\theta(s_t)$ normalizes the distribution. We adopt one policy (actor) network, two Q-networks (critic), and two target Q-networks for SAC training. Each network consists of one demonstration-based attention module for task-information extraction and projects the task embedding into actions.

For learning robust policies, we randomly choose a state from the given demonstration as a start and add a disturbance of $0.1 \times N(0, 1)$ to this state coordinate. We maintain a separate buffer for each demonstration and gather a batch of training data from 5 different buffers. To accelerate the training process, we also add demonstration data which takes 20% of the batch size for joint training. For fair comparison, all the baselines we compared followed the above setting. The detailed hyper-parameters used for our OSIL method training are summarized in Tab. 5.

E. Environment Description

E.1. Valet-Parking-Assist-in-Maze Environment

We create a challenging benchmark, named Valet Parking Assist in Maze (VPAM), to assess OSIL's performance for unforeseen changes. This navigation benchmark is inspired by a popular and practical real-world application in autonomous driving, called Valet Parking Assist (VPA) (Heimberger et al., 2017). After providing a human demonstration, as soon as the driver requests the vehicle in the smartphone app, VPA returns the car to the pick-up area — no need for the driver to spend time looking for the car and maneuvering it out of the parking garage. We summarize the key features of the VPA environment as follows:

- The parking lot is weak in GPS signal, and the coordination is often unreliable.
- The VPA application should work in any parking lot.
- The road would happen to have unexpected roadblocks or passengers.

The development cost of the rule-based policy will increase as more and more configurations are applied. For example, with obstacles and without coordination, it is non-trivial to develop a rule that can follow the demonstration to the target points in any randomly generated maps without collision. We build our VPAM benchmark using the maze as an intuitive navigation testbed. It is easy for us to test the algorithm in unseen tasks by generating many scenarios through the built maze generation algorithm and many demonstrations via the built deep-first-searching policy through the oracle map information.

We include details of our VPAM benchmark, where the maze layout takes a size of 24×24. The maze is generated by randomly traversing all the cells in a Depth-First manner with path width 2. The path in the maze is connected, thereby our environment is Ω-recoverable and 1-demo imitator available, which satisfies our OSIL needs. A point agent needs to travel from a start point to a target position in a maze, guided by expert demonstrations. The maze and target positions vary between episodes, so the trained agent should work well in any simulated parking lot. Some obstacles may appear in the demonstration path to model unexpected roadblocks or passengers. Obstacle and wall information is directly accessible. We fixed the starting point as the center of the map. The agent relies on its l-step local views to make decisions, where l is a configuration for task setup. The current coordinate of the agent is optionally provided for mimicking weak GPS signals in the real parking lot. The expert trajectories can also be obtained by a Depth-First search. We can formulate this environment as a Markov Decision Process, which can be presented as a tuple $(\mathcal{S}, \mathcal{A}, T, R)$.

State space $\mathcal{S}$: The maze state consists of the (x, y) coordinate and the local view of the agent along 8 different directions with an equal interval $\pi/4$. For the simplest task where coordinates are provided and no obstacles exist, the local view length l is set to 1.5; otherwise 5 for observing the surrounding environment changes.

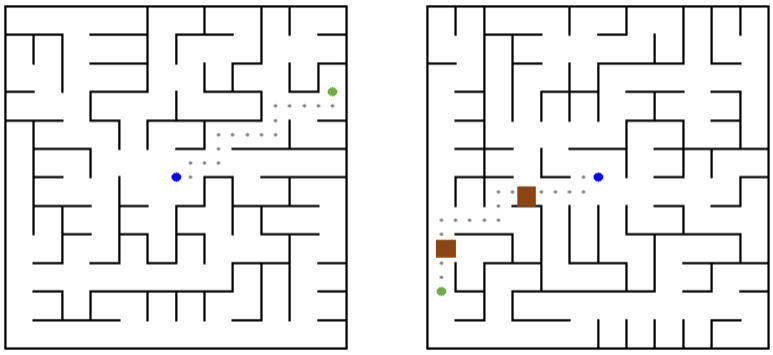
Action space $\mathcal{A}$: The agent is able to take action (Δ_x, Δ_y) which are continuous values in the range of $[-1, 1]$.

Transition function T: When applied with the action (Δ_x, Δ_y) at the coordinate (x, y), the agent translates itself to the $(x + \Delta_x, y + \Delta_y)$ coordinate. Some obstacles, which have lengths in the range of $[1.1, 1.3]$ and widths of 1.35, may appear in the demonstration path. The obstacle will be generated with a fixed probability $p = 0.1$ for each demonstration step and with a maximum number of 4. Once hits the wall or the obstacles, the agent will be dead and the trajectory will be terminated.

Reward function R_ω: We only have a simple reward function R_ω which indicates the ending of trajectories, e.g., c for reaching the target goal, 0 for failure in 50 timesteps, and $-c$ for dead, where ω is the goals we set.

Due to the unavailability of the global map, the agent is expected to follow the demonstration and reconstruct the expert's behavior to reach the goal, as illustrated in Fig. 12(a). Beyond this, some unexpected obstacles may occur, which results in that strictly following the demonstration no longer working, as shown in Fig. 12(b). The agent is expected to learn robust policies that can bypass obstacles and finish the task, based on imitating the given demonstration.

For single-map scenarios, we randomly choose 90% of all demonstrations (290 demos for each map) for training while the left is for evaluation. For multi-map imitation, we generate 240 different maps and only select a small number of 10 training demonstrations from each map. We treat the remaining *new demonstrations* to verify generalization. Besides, we also create 10 *new maps* separately to verify whether the agent trained on the multi-map scenarios works.

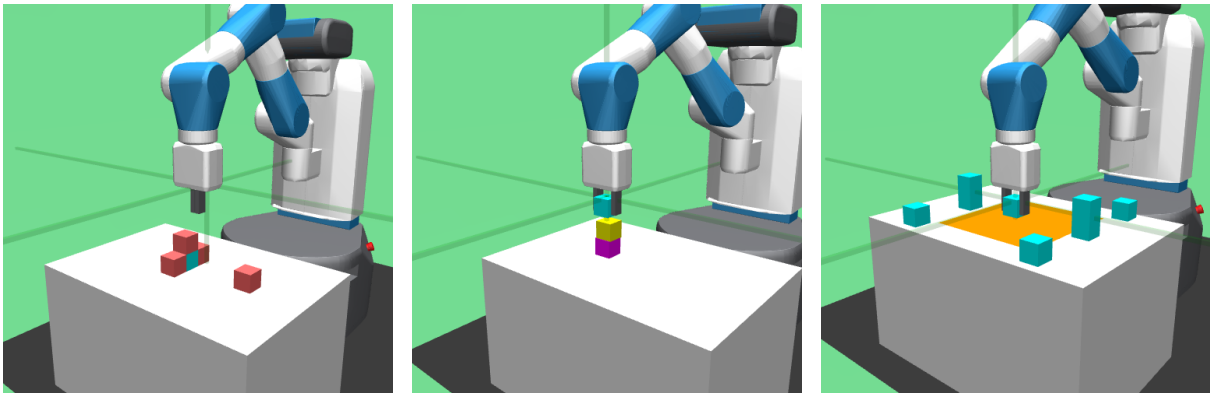


(a) Without obstacle. (b) With obstacles.

Figure 12: Fig. 12(a) shows a demonstration sample on a map without obstacle, where we fix the start position as the center of the map (colored in blue), while the agent is expected to reach the specified goal colored in green. The agent should follow the expert trajectory to achieve the goal due to the unavailability of the global map. Fig. 12(b) shows a demonstration sample on a map with obstacles. Here strictly imitating the expert trajectory cannot well handle unexpected situations, e.g., the agent is blocked by obstacles when it follows the given demonstration. Beyond pure imitation, the agent should also explore the environment to learn robust policies.

E.2. Demo-Manipulation Environment

We introduce details of our robot manipulation environment to verify the imitation ability of our method across different tasks. This environment consists of three types of robotics manipulations, namely object grasping, object stacking, and object collecting. We provide illustrations in Fig. 13, where the workspace is a 50 cm $\times$ 70 cm area. To collect demonstrations, we instruct the robot to execute predefined primitives in sequence. For instance, grasping a single object comprises three primitives: 1) moving the gripper to the object; 2) closing the gripper; 3) moving the gripper to the target. We present the Markov formalization of this environment in the following.



(a) Grasping (b) Stacking (c) Collecting

Figure 13: Various tasks of robot manipulation. (a): Grasp the blocked target object (cyan). (b): Stack the objects. (c): Collect the objects scattered over the desk together to the specified area (yellow).

State space $\mathcal{S}$: The robot manipulation state includes the absolute position of the robot gripper, the absolute positions of the objects, and the relative positions of the gripper fingers.

Action space $\mathcal{A}$: The agent is able to take action $(\Delta_x, \Delta_y, \Delta_z, \Delta_c)$, each of which is continuous value in the range of $[-1, 1]$. The first three dimensions indicate the desired increment in the gripper position at the next timestep, while the last

dimension controls the positions of the gripper fingers.

Transition function T: When applied with the action $(\Delta_x, \Delta_y, \Delta_z)$ at the coordinate (x, y, z), the robot gripper moves to the new coordinate $(x + \Delta_x, y + \Delta_y, z + \Delta_z)$. If $\Delta_c > 0$, the gripper opens; otherwise, it closes. The task is considered failed if any object falls off the desk.

Reward function R: Similar to the valet-parking-assist-in-maze environment, our reward function R is simple and only indicates the end of trajectories. That is, we use c to indicate task accomplishment, $-c$ for failure, and 0 for all other situations. The criteria for accomplishing each task differs. In the object grasping task, the robot needs to grasp the target object without colliding with other objects. In the object stacking task, the robot must stack three blocks together, which are initially placed anywhere on the workspace. Lastly, in the object collecting task, the robot needs to collect all objects scattered over the desk and place them in a specified area.

We randomly generate 300 demonstrations for each task, of which 60 are used for training and the remaining 240 are for testing. To verify the imitation ability of our method across multiple manipulation tasks, we generate 100 demonstrations for each of the three tasks. We randomly select 20 demonstrations from each task for training and leave 240 new demonstrations for the test.

E.3. Pusher and Reacher Environment

We introduce the Pusher and Reacher environments in Gymnasium (Towers et al., 2023), where we each randomly collect 60 demonstrations for training and 240 for testing. For the Pusher task, the state comprises the positions and velocities of the robot's joints (a total of 7), as well as the position of the robot tip arm and the manipulated object. The agent accomplishes the task by taking actions that modify the rotation of each joint and drive the robot to push the object to the specified position. The specific contents of the state space and action space are provided in the following:

Table 6: Details of Pusher observation space.

Num	Observation	Min	Max	Name	Joint	Unit
0	Rotation of the panning the shoulder	-Inf	Inf	r_shoulder_pan_joint	hinge	angle (rad)
1	Rotation of the shoulder lifting joint	-Inf	Inf	r_shoulder_lift_joint	hinge	angle (rad)
2	Rotation of the shoulder rolling joint	-Inf	Inf	r_upper_arm_roll_joint	hinge	angle (rad)
3	Rotation of hinge joint that flexed the elbow	-Inf	Inf	r_elbow_flex_joint	hinge	angle (rad)
4	Rotation of hinge that rolls the forearm	-Inf	Inf	r_forearm_roll_joint	hinge	angle (rad)
5	Rotation of flexing the wrist	-Inf	Inf	r_wrist_flex_joint	hinge	angle (rad)
6	Rotation of rolling the wrist	-Inf	Inf	r_wrist_roll_joint	hinge	angle (rad)
7	Rotational velocity of the panning the shoulder	-Inf	Inf	r_shoulder_pan_joint	hinge	angular velocity (rad/s)
8	Rotational velocity of the shoulder lifting joint	-Inf	Inf	r_shoulder_lift_joint	hinge	angular velocity (rad/s)
9	Rotational velocity of the shoulder rolling joint	-Inf	Inf	r_upper_arm_roll_joint	hinge	angular velocity (rad/s)
10	Rotational velocity of hinge joint that flexed elbow	-Inf	Inf	r_elbow_flex_joint	hinge	angular velocity (rad/s)
11	Rotational velocity of hinge that rolls the forearm	-Inf	Inf	r_forearm_roll_joint	hinge	angular velocity (rad/s)
12	Rotational velocity of flexing the wrist	-Inf	Inf	r_wrist_flex_joint	hinge	angular velocity (rad/s)
13	Rotational velocity of rolling the wrist	-Inf	Inf	r_wrist_roll_joint	hinge	angular velocity (rad/s)
14	x-coordinate of the fingertip of the pusher	-Inf	Inf	tips_arm	slide	position (m)
15	y-coordinate of the fingertip of the pusher	-Inf	Inf	tips_arm	slide	position (m)
16	z-coordinate of the fingertip of the pusher	-Inf	Inf	tips_arm	slide	position (m)
17	x-coordinate of the object to be moved	-Inf	Inf	object (obj_slidex)	slide	position (m)
18	y-coordinate of the object to be moved	-Inf	Inf	object (obj_slidey)	slide	position (m)
19	z-coordinate of the object to be moved	-Inf	Inf	object	cylinder	position (m)

The Reacher environment features a two-jointed robotic arm tasked with moving its end effector, to a randomly positioned target. It involves a continuous action space representing the torques applied at the hinge joints of the arm. The observation space includes details such as the angles of the arm joints, their angular velocities, and the relative position of the fingertip to the target. The action space is the torques applied to the two hinge joints of the robotic arm. More details is in the following:

E.4. Manipulation Tasks in Meta-World Environments

We provide details on manipulation tasks within the Meta-World benchmark (Yu et al., 2019). These environments have been adapted by introducing unexpected disturbances. We chose four types of robotics manipulations for evaluation, namely shelf-place-v2, peg-insert-side-v2, pick-place-hole-v2, and sweep-v2. we use the official script to collect demonstrations. For each environment, we gather 60 demonstrations for training and 240 demonstrations for testing, with randomly generated goals. We present the state space and action space of this environment in the following:

Table 7: Details of Reacher observation space.

Num	Action	Control Min	Control Max	Name	Joint	Unit
0	Rotation of the panning the shoulder	-2	2	r_shoulder_pan_joint	hinge	torque (N m)
1	Rotation of the shoulder lifting joint	-2	2	r_shoulder_lift_joint	hinge	torque (N m)
2	Rotation of the shoulder rolling joint	-2	2	r_upper_arm_roll_joint	hinge	torque (N m)
3	Rotation of hinge joint that flexed elbow	-2	2	r_elbow_flex_joint	hinge	torque (N m)
4	Rotation of hinge that rolls the forearm	-2	2	r_forearm_roll_joint	hinge	torque (N m)
5	Rotation of flexing the wrist	-2	2	r_wrist_flex_joint	hinge	torque (N m)
6	Rotation of rolling the wrist	-2	2	r_wrist_roll_joint	hinge	torque (N m)

State space $\mathcal{S}$: The observation space in Meta-World is represented by a six-part tuple, comprising the 3D Cartesian coordinates of the end-effector, a normalized indicator reflecting the gripper's degree of openness, the 3D position and quaternion orientation of the two objects, and the 3D coordinates of the goal. For each environment, we mask the 3D coordinates of the goal by setting them to 0.

Action space $\mathcal{A}$: The agent can take actions with four dimensions, each ranging from $[-1, 1]$. The first three dimensions represent the desired increment in the gripper position at the next timestep, while the last dimension controls the gripper's openness.

Disturbance To introduce unexpected disturbances, we implement a hardcoded script to disrupt the agent's actions. Specifically, for shelf-place-v2, peg-insert-side-v2, and pick-place-hole-v2, when the object is lifted, we attempt to open the gripper to let the object drop. For sweep-v2, we randomly replace the first three dimensions of the action controlling the gripper's movement with random numbers with a probability of 0.5.

F. Main Results of Various Benchmarks

Tab. 8, 9, and 10 summarize the main outcomes of the VPAM benchmark, Meta-World benchmark (Yu et al., 2019), as well as robot manipulation tasks (Pang et al., 2023) and Gymnasium tasks (Towers et al., 2023), respectively.

Table 8: Success rate comparison on valet-parking-assist-in-maze tasks. The agent needs to imitate demos seen during the training, new demos from seen maps, and demos collected on new maps, namely denoted as "seen", "new_demo", and "new_map" in this table. Our experiment uses 3 random seeds and we **bold** the best scores for each task.

Map Type		Single-Map				Multi-Map					
Obstacle Type		Non-Obstacle		Unforeseen Obstacle		Non-Obstacle			Unforeseen Obstacle		
Demontrations		seen	new_demo	seen	new_demo	seen	new_demo	new_map	seen	new_demo	new_map
Coord	DDT	**1.00±0.00**	**0.94±0.03**	**0.81±0.02**	**0.76±0.02**	**0.90±0.01**	**0.82±0.02**	**0.77±0.02**	**0.74±0.00**	**0.73±0.00**	**0.69±0.01**
	DCRL	0.99±0.01	0.93±0.01	0.78±0.03	0.74±0.03	0.56±0.04	0.46±0.03	0.46±0.04	0.51±0.01	0.50±0.02	0.46±0.02
	OSIL	0.43±0.09	0.16±0.10	0.14±0.10	0.04±0.02	0.44±0.01	0.37±0.01	0.32±0.02	0.32±0.05	0.22±0.03	0.21±0.04
	CbMRL	0.98±0.00	0.76±0.02	0.66±0.01	0.44±0.02	0.46±0.03	0.31±0.02	0.34±0.01	0.37±0.03	0.32±0.03	0.33±0.02
No-Coord	DDT	**0.51±0.19**	**0.71±0.06**	**0.46±0.06**	**0.58±0.04**	**0.83±0.03**	**0.63±0.01**	**0.54±0.05**	**0.50±0.02**	**0.44±0.02**	**0.40±0.03**
	DCRL	0.24±0.03	0.01±0.01	0.15±0.01	0.00±0.00	0.15±0.06	0.05±0.02	0.04±0.02	0.13±0.02	0.04±0.01	0.08±0.01
	OSIL	0.06±0.02	0.00±0.00	0.02±0.02	0.00±0.00	0.06±0.04	0.01±0.01	0.02±0.01	0.02±0.03	0.03±0.02	0.01±0.01
	CbMRL	0.15±0.06	0.02±0.01	0.09±0.02	0.01±0.00	0.14±0.01	0.03±0.01	0.02±0.01	0.10±0.02	0.05±0.02	0.06±0.01

G. Learning Curve

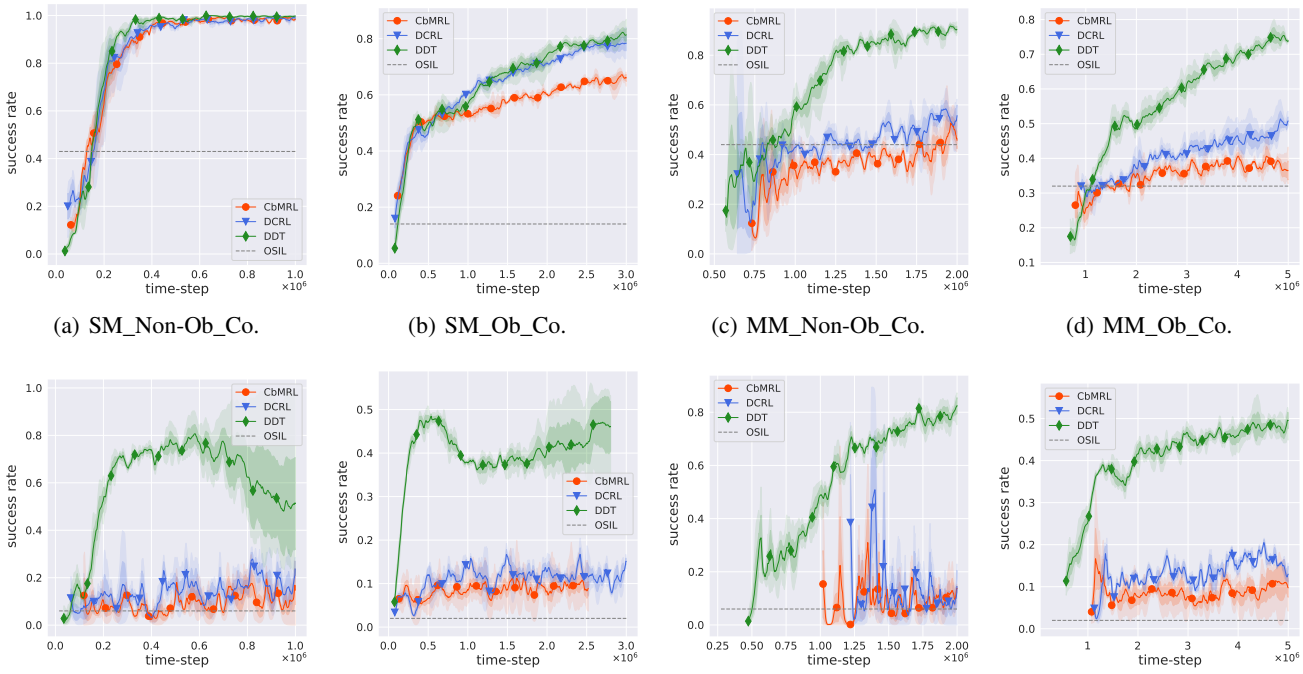
We list the learning curves in this section. Fig. 14, Fig. 15, and Fig. 16 show the learning curves in eight different navigation tasks respectively. Fig. 17 shows the BC loss of TRANS-BC, which is the mean squared error (MSE) loss between expert actions and agent actions. To ensure conciseness in our description, we employ the following abbreviations: "SM" for Single-Map, "MM" for Multi-Map, "Ob" for scenes with obstacles, "Non-Ob" for scenes without obstacles, "Co" for scenes with coordinates and "Non-Co" for scenes without coordinates.

Table 9: Performance comparison on the Meta-World benchmark, including scenarios without disturbance and with disturbance. We **bold** the best scores for each task.

Task		Shelf Place		Peg Insert Side		Pick Place Hole		Sweep	
Demonstration		seen	unseen	seen	unseen	seen	unseen	seen	unseen
No Disturbance	DDT	**1.00**	**0.94**	**1.00**	**0.62**	**1.00**	**0.84**	**1.00**	**1.00**
	DCRL	0.97	0.76	1.00	0.28	0.00	0.00	0.95	0.44
	Trans4OSIL	0.02	0.04	0.04	0.08	0.00	0.00	0.50	0.32
	CbMRL	0.78	0.44	0.84	0.16	0.16	0.00	1.00	0.94
With Disturbance	DDT	**0.79**	**0.44**	**0.92**	**0.70**	**1.00**	**0.90**	**0.61**	**0.40**
	DCRL	0.75	0.34	0.07	0.02	0.00	0.00	0.10	0.10
	Trans4OSIL	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	CbMRL	0.18	0.14	0.02	0.14	0.00	0.00	0.12	0.10

Table 10: Performance comparison on various tasks. We **bold** the best scores for each task.

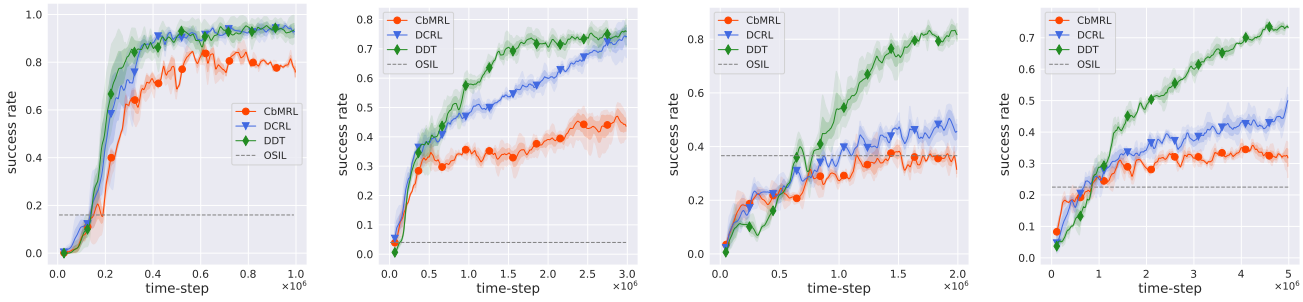
Domain	**Complex Manipulation**						**Complex Control Space**			
Task	Grasping		Stacking		Collecting		Reacher		Pusher	
Demonstration	seen	unseen	seen	unseen	seen	unseen	seen	unseen	seen	unseen
DDT	**0.98**	**0.84**	**0.77**	**0.84**	**0.99**	**0.61**	**0.98**	**0.95**	**0.96**	**0.94**
DCRL	0.30	0.70	0.00	0.00	0.00	0.00	0.65	0.50	0.89	0.87
Trans4OSIL	0.28	0.20	0.00	0.02	0.17	0.06	0.63	0.59	0.20	0.08
CbMRL	0.71	0.49	0.00	0.00	0.00	0.00	0.90	0.87	0.91	0.85



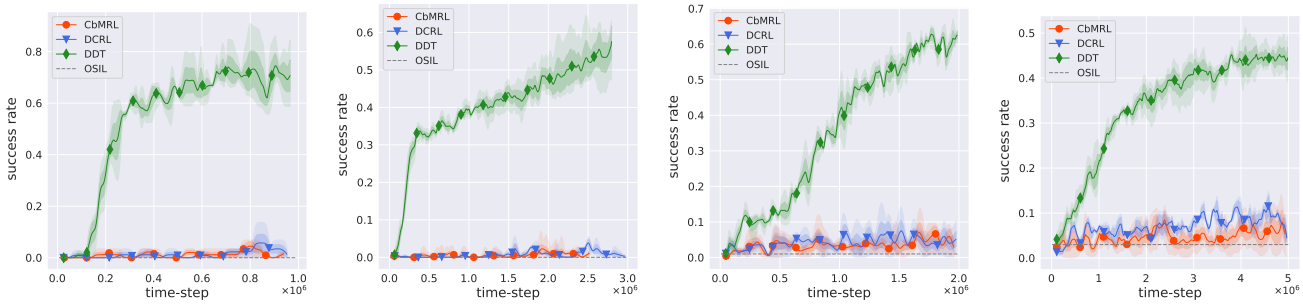
(a) SM_Non-Ob_Co. (b) SM_Ob_Co. (c) MM_Non-Ob_Co. (d) MM_Ob_Co.

(e) SM_Non-Ob_Non-Co (f) SM_Ob_Non-Co. (g) MM_Non-Ob_Non-Co. (h) MM_Ob_Non-Co.

Figure 14: Learning curves on demonstrations seen during the training.

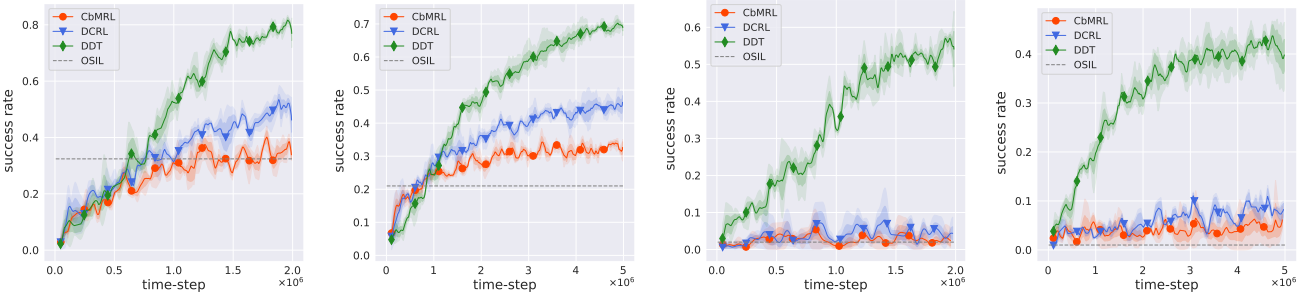


(a) SM_Non-Ob_Co. (b) SM_Ob_Co. (c) MM_Non-Ob_Co. (d) MM_Ob_Co.



(e) SM_Non-Ob_Non-Co. (f) SM_Ob_Non-Co. (g) MM_Non-Ob_Non-Co. (h) MM_Ob_Non-Co.

Figure 15: Learning curves on new demonstrations from seen maps.



(a) MM_Non-Ob_Co. (b) MM_Ob_Co. (c) MM_Non-Ob_Non-Co. (d) MM_Ob_Non-Co.

Figure 16: Learning curves on demonstrations collected from new maps.

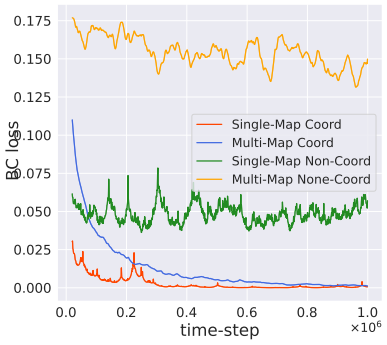


Figure 17: BC Loss of OSIL. Note that both scenes with obstacles and without obstacles use the same set of offline demonstrations, thus there are a total of four curves representing eight tasks.

H. More Ablation Study Results

H.1. The Roles of Three Stages in DDT

Since the three stages of the DDT policy are implicitly embedded, it is non-trivial to adequately verify the roles. In the main manuscript, we have verified the first stage of decision-making via visualizing attention scores in Sec. 5.4. In these experiments, the higher attention values are predominantly concentrated on the relevant states, demonstrating our DDT's ability to identify which state to follow. The provided robot manipulation video also validates that DDT dynamically focuses on demonstration states that should be followed during control tasks. To address the concern further, we included experimental results for the roles of the latter two stages:

Stage2-Analyze: We demonstrate that the agent completes tasks by analyzing expert state-action pairs by randomly dropping out inputs to the state-action encoder. We report the performance based on the keep rate in Tab. 11.

Table 11: Performance across different keep rates.

Keep Rate	1.0	0.8	0.4	0.2	0.1
Performance	0.94	0.92	0.73	0.44	0.28

It is evident that as fewer state-action pairs are retained as input, the success rate decreases. This highlights the importance of utilizing state-action pairs as the value item for the attention net.

Stage3-Trace: We validate our algorithm's decision ability to trace demonstrations, i.e., recap the current state and take the optimal action. We achieve this by reducing the times of fusion operation $\oplus$.

Table 12: Performance across different times of fusion operation.

Times of fusion operation	6	5	4	3	2	1
Performance	0.94	0.91	0.66	0.30	0.06	0.02

In Tab. 12, it is noticeable that as the times of fusion operation decrease, the agent's decision-making ability significantly declines. This further indicates the necessity of our network architecture design.

H.2. Performance in Unseen Environments

In Tab. 13, We provide experimental results of DDT in unseen environments. We select the objects-manipulation tasks in Meta-World for training and testing. The training tasks are combined with Button Press Topdown, Button Press Topdown Wall, Button Press Wall, Door Open, Faucet Close, Drawer Close, Window Open, and Window Close. Then we test and record the generalization performance on three new environments, Button Press, Door Close, and Reach without fine-tuning.

Table 13: Performance in unseen environments.

Environments	Button Press	Door Close	Reach
Performance	0.78	1.00	0.75

The algorithm is trained following the previous hyper-parameters directly. It can be observed that even when the agent is trained without corresponding task environments, our algorithm still achieves satisfactory success rates by consciously imitating demonstrations.

H.3. Performance across Different Demonstration Lengths

Generally, attention mechanisms are known to be sensitive to sequence length to some degree. However, in our experiments, we observed that this sensitivity was not significant. We provide results as proof by statistically analyzing the relationship between demonstration length and success rate in Tab. 14.

Table 14: Success rate across different demonstration lengths.

Length	10∼14	15∼19	20∼24	25∼29	30∼34	35∼39	40∼44	45∼49	50∼54
Performance	0.93	0.96	0.97	0.99	0.98	0.98	0.99	0.93	0.95

H.4. Comparison with Replay-based and Match-based Methods

There is a line of work (Johns, 2021; Valassakis et al., 2022; Wen et al., 2022) in the robotics/robot learning community that also studies one-shot imitation learning, but follows a much simpler replay-based strategy, requires only one demonstration for a category of tasks, and is capable of performing complicated robotics manipulation. The three OSIL baselines all employ parameter-free methods from the perspective of decision-making: (Johns, 2021; Valassakis et al., 2022) retrieve actions corresponding to matched states (the nearest neighbor) in the single demonstration, while (Wen et al., 2022) replays manipulation demonstrations collected from objects of the same category after identifying the target object pose. We make comparisons with these methods, where each test case was evaluated on unseen maps in the valet-parking-assist-in-maze scenario.

Table 15: Comparison with replay-based and match-based methods.

Method	W/o obstacle & W/coordinate	W/ obstacle & W/ coordinate	W/o obstacle & W/o coordiante	W/obstacle & W/o coordinate
Replay-based	0.54	0.32	0.53	0.32
Match-based	0.36	0.21	0.27	0.14
DDT	**0.77**	**0.69**	**0.54**	**0.4**

Here for the replay-based method, we use the ground-truth trajectory to replay the actions. We can observe that due to the presence of noise in the environment, the methods struggle to complete tasks effectively even in scenarios without obstacles and with coordinates provided, and the results are worse in scenarios with obstacles. This highlights that while parameter-free OSIL methods may be feasible for tasks that are repeatable in the same scene, they severely lack generalization ability, let alone generalize to unexpected situations.

H.5. Training on Mixed-Up Demonstrations from Different Environments

We also believe that validating our algorithm's performance on mixed-up demonstrations from all different environments can enhance the significance of our algorithm. We collected mixed-up demonstrations from varying numbers of environments. We report the performance of these demonstrations according to the number of environments designed:

Table 16: Performance on mixed-up demonstrations from different environments.

Envionment number	2	4	8
Seen Demos.	1.00	1.00	1.00
Unseen Demos.	0.84	0.88	0.90

Note:

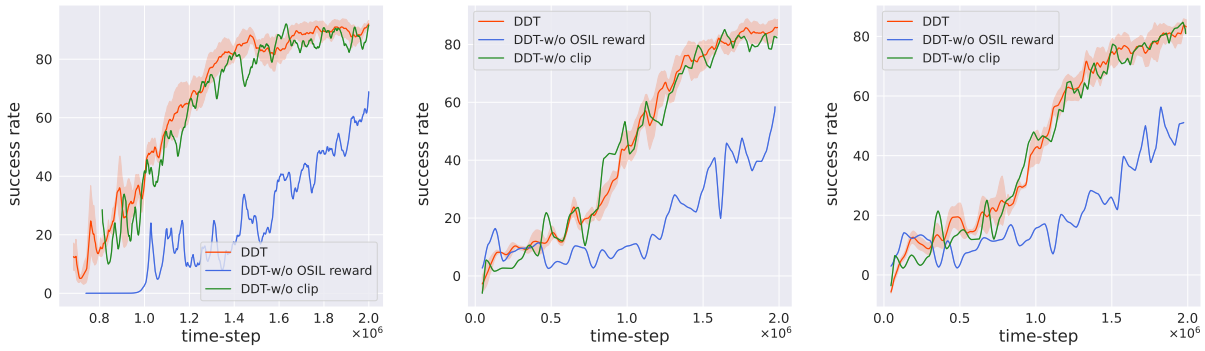
- 2 environments: Sweep, Peg Insert Side;
- 4 environments: With Pick Place Hole, and Shelf Place newly added;
- 8 environments: With Button Press, Dial Turn, Drawe Close, and Window Open newly added.

The results in Tab. 16 reveal the strong adaptability of our method for learning tasks across diverse environments, consistently performing well regardless of the number of environments involved.

H.6. Reward Designs

We conduct ablation studies considering reward design in maze tasks. In particular, for clipping term η in R_{OSIL}, we set η as infinite value (DDT-w/o clip), the results can be found in Fig. 18. The results show that directly removing the clipping function of η, which enhances the probabilities of ill-posedness led by the L2 distance, e.g., a wall obstructing the path

between two states will have a small L2 distance, reduces the sample efficiency of DDT, but the asymptotic performance is still similar, which demonstrates the robustness of DDT to the ill-posedness of the adopted L2 distance.



(a) Seen demonstrations. (b) New demonstrations. (c) New maps.

Figure 18: Learning curves of agents with different implementations of R_{OSIL} reward in the valet-parking-assist-in-maze benchmark. The task is the multi-map imitation without obstacles and with coordinates.

H.7. The Robustness on the Faraway States

In general OSIL scenarios, the task set we face might violate the assumptions defined in Def. A.1, for example, the transition function might, with some probabilities, lead the agent to some states that are "significantly far away" from the expert states. More specifically, in these faraway states, we cannot trace back to the demonstrations just by relying on the current states' information; e.g., in the Maze-navigation benchmark, if the current states' coordinations are far away from the demonstrations, the way back to the demonstrations depends on the walls' location in the maps, which is unseeable to the agent, thus the agent cannot find a unified behavior to back to the demonstration and reach the goal. In this section, we conducted an offset-range test in the maze benchmark to verify the robustness of DDT in faraway states. In particular, we use a DDT policy trained in the setting of multi-map navigation without obstacles and with coordinates provided. When deploying the policy, we generate 100 unseen maps and add positional offsets sampled from a uniform distribution to the initial states. We average the success rate under different ranges of offsets in Fig. 19.

In the maze environment, it is noteworthy that an offset range of initial states larger than could potentially make the agent be separated from the expert trajectory by a wall, which violates the property of 1-demo imitator availability in Prop. 4.2. Correspondingly, the results show that the policy sustains a respectable success rate within an offset of 2.4. After that, expanding the range leads to a nearly linear decrease in success rate. The experiment demonstrated the agent's success in devising a well-performed policy for the scope of tasks with 1-demo imitator availability. Similar results can also be found in other experiments in which there is no exact match between the current and target state in these scenarios.

- In maze settings with obstacles (Figs. 27, 28, 31, and 32), we have observed the agent's remarkable ability to adaptively adjust its behavior when encountering obstacles.
- In robot manipulation tasks (Figs. 33-38), we present a showcase of the robotic arm's proficiency in following a trajectory while optimizing its operational efficiency. Moreover, in the corresponding video, which records rollouts generated by the DDT policy, we can observe the simultaneous activation of multiple expert states through attention mechanisms when an exact match between the current state and the target state is lacking. The video can be found in the supplementary material.

In conclusion, within our solution scope, i.e., the problem with 1-demo imitator availability defined in Prop. 4.2, the imitator policy works well, even if the current states do not perfectly match the expert states. Besides, the policy still works to some degree in faraway states. However, we argue that without further information, assumptions, or prior knowledge, it is impossible to find a perfect imitator policy when the agent is inevitably to reach some faraway states. We leave the imitator learning problem in this setting as future work.

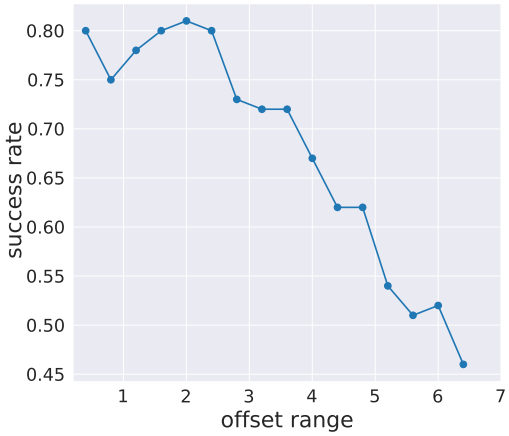
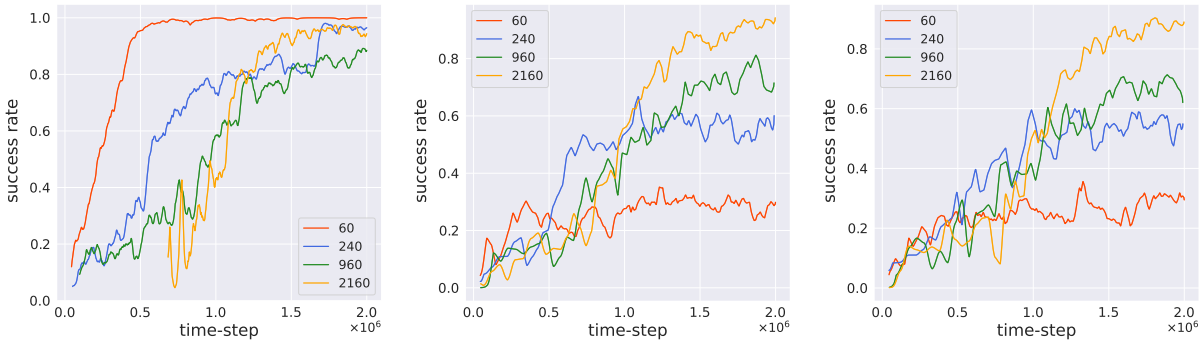


Figure 19: Illustration of DDT in offset-range test. The X-axis is the offset range on the initial states, while the Y-axis shows the corresponding success rate of DDT under various offsets.

I. The Details of the Scaling Up Experiments

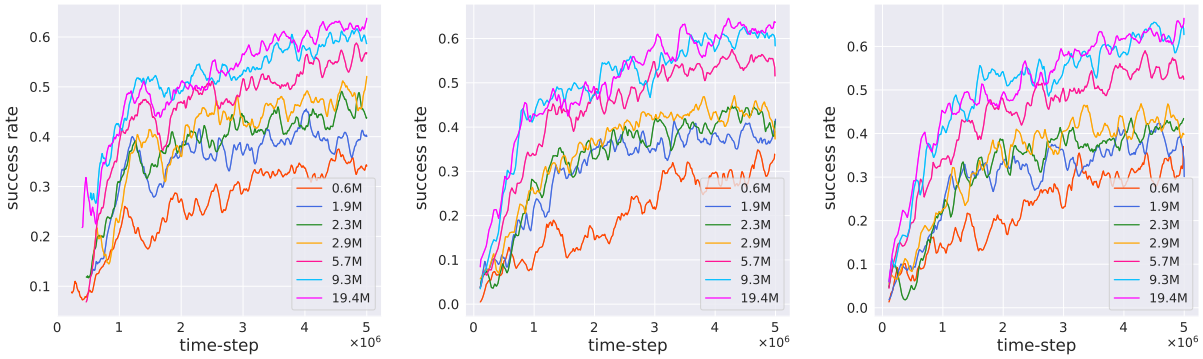
Experiments on Different Demonstration Quantity To investigate the influence of demonstration quantity on model performance, we conducted experiments with four varying quantity settings: 60, 240, 960, and 2160. The results are in Fig. 20. We observed that fewer data leads to quicker initial learning speed and rapid performance improvement. However, a bottleneck emerges when aiming for generalization performance. As demonstration quantity increases, the learning task becomes more difficult, resulting in slower initial learning. Nonetheless, the final model exhibits notably superior performance on new demos and new maps compared to experiments conducted with fewer data.



(a) Seen demos (b) New demos (c) New maps

Figure 20: Learning curves of agents with varying demonstration quantity. Note that the task is the multi-map imitation without obstacles and with coordinates.

Experiments on Different Model Parameters In this experiment, We focused on tuning d_{model}, nhead, $L_{encoder}$ and $L_{decoder}$ for both actor and critic to construct DDT variants with different model parameters, where d_{model} represents the desired number of features in the encoder/decoder inputs, nhead denotes the number of heads in the multi-head attention mechanism, $L_{encoder}$ represents the number of sub-encoder layers within the encoder, and $L_{decoder}$ refers to the number of sub-decoder layers within the decoder. The details of parameters selection are in Tab. 17, and the learning curves are in Fig. 21.



(a) Seen demos (b) New demos (c) New maps

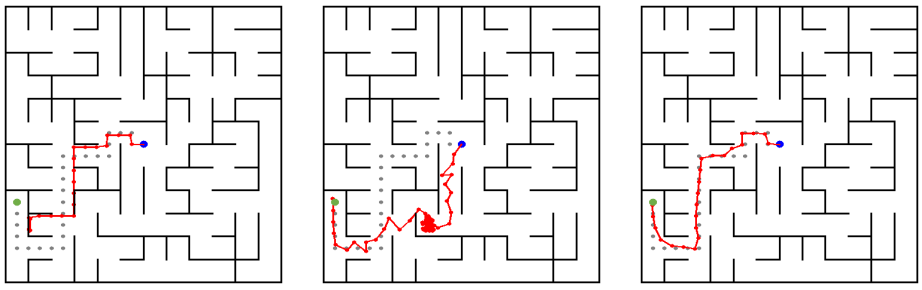
Figure 21: Learning curves of agents with varying model sizes. Note that the task is the multi-map imitation with obstacles and without coordinates.

Table 17: Architecture hyperparameters for DDT variants with different model parameters.

Total Parameters	Actor					Critic				
	d_{model}	nheads	L_{encoder}	L_{decoder}	parameters	d_{model}	nheads	L_{encoder}	L_{decoder}	parameters
0.6M	64	16	3	3	0.4M	64	16	2	2	0.2M
1.9M	64	32	3	6	0.6M	128	16	4	4	1.3M
2.3M	96	32	3	6	1.0M	128	16	4	4	1.3M
2.9M	128	64	3	6	1.6M	128	16	4	4	1.3M
5.7M	192	16	3	6	3.1M	192	16	4	4	2.6M
9.3M	256	16	3	6	5.1M	256	16	4	4	4.2M
19.4M	384	16	3	6	10.7M	384	16	4	4	8.7M

J. Examples of Trajectories with Different Learning Paradigms

The demonstration input not only tells the agent which task to accomplish but the way to accomplish the task. To better illustrate that our DDT method leverages information from demonstrations to understand the task and enables better generalization ability learning by imitating demonstrations, we provide visualized agent behaviors of different learning paradigms in Fig. 22. (a) Trans4OSIL attempts to mimic demonstrations but cannot generalize to states not seen in the demonstrations, failing. (b) DCRL completes the task but does not explicitly imitate the demonstrations. (c) In contrast, our DDT method efficiently completes the task while imitating demonstrations. The results underscore the superior generalization ability compared to other paradigms.

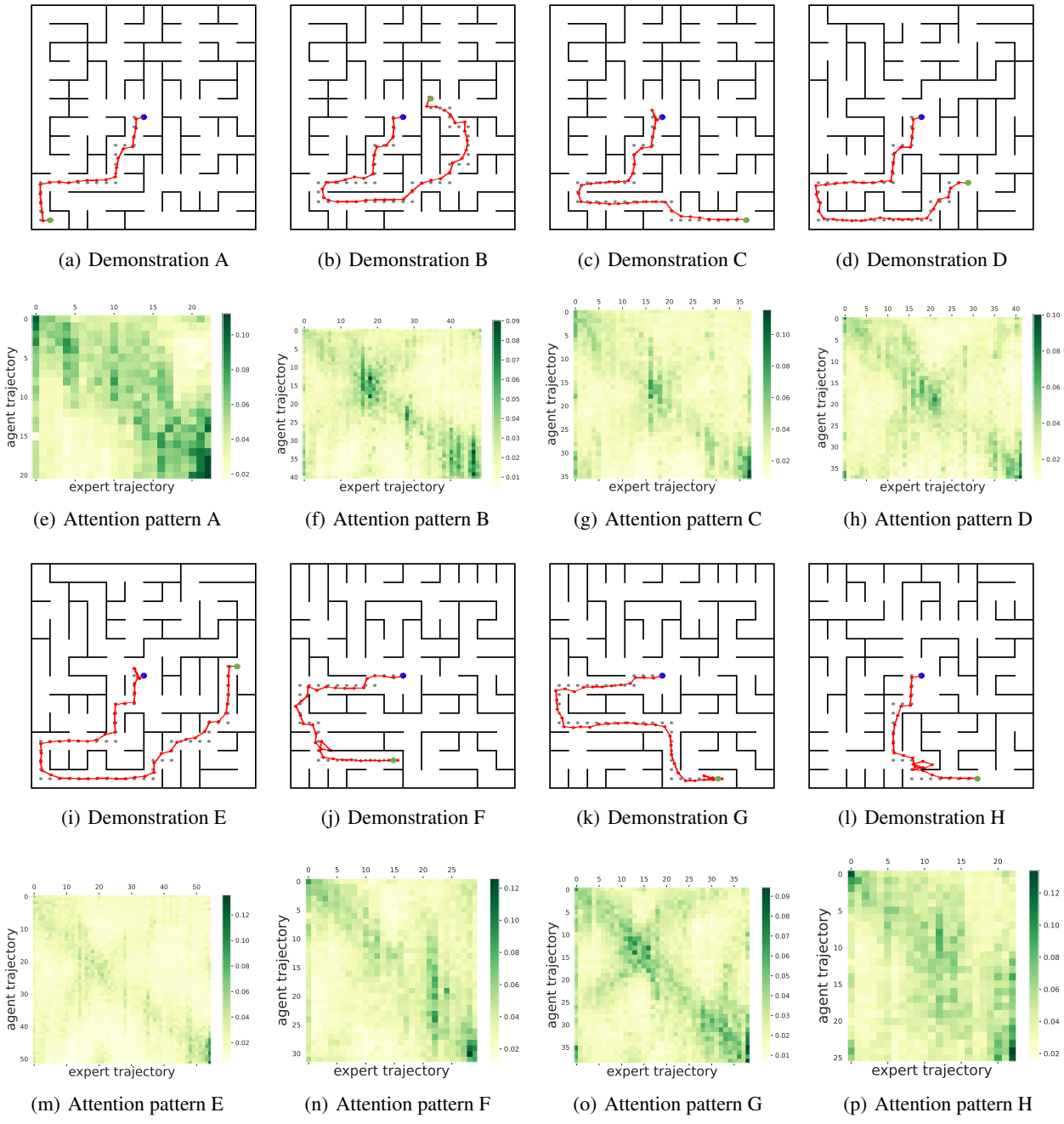


(a) Trans4OSIL (b) DCRL (c) DDT

Figure 22: Visualized trajectories of different learning paradigms.

K. Visualization

In Fig. 23, we give more visualizations. In these tasks, the agent knows its coordinate and is required to imitate demonstrations collected on new maps without obstacles.



(a) Demonstration A (b) Demonstration B (c) Demonstration C (d) Demonstration D

(e) Attention pattern A (f) Attention pattern B (g) Attention pattern C (h) Attention pattern D

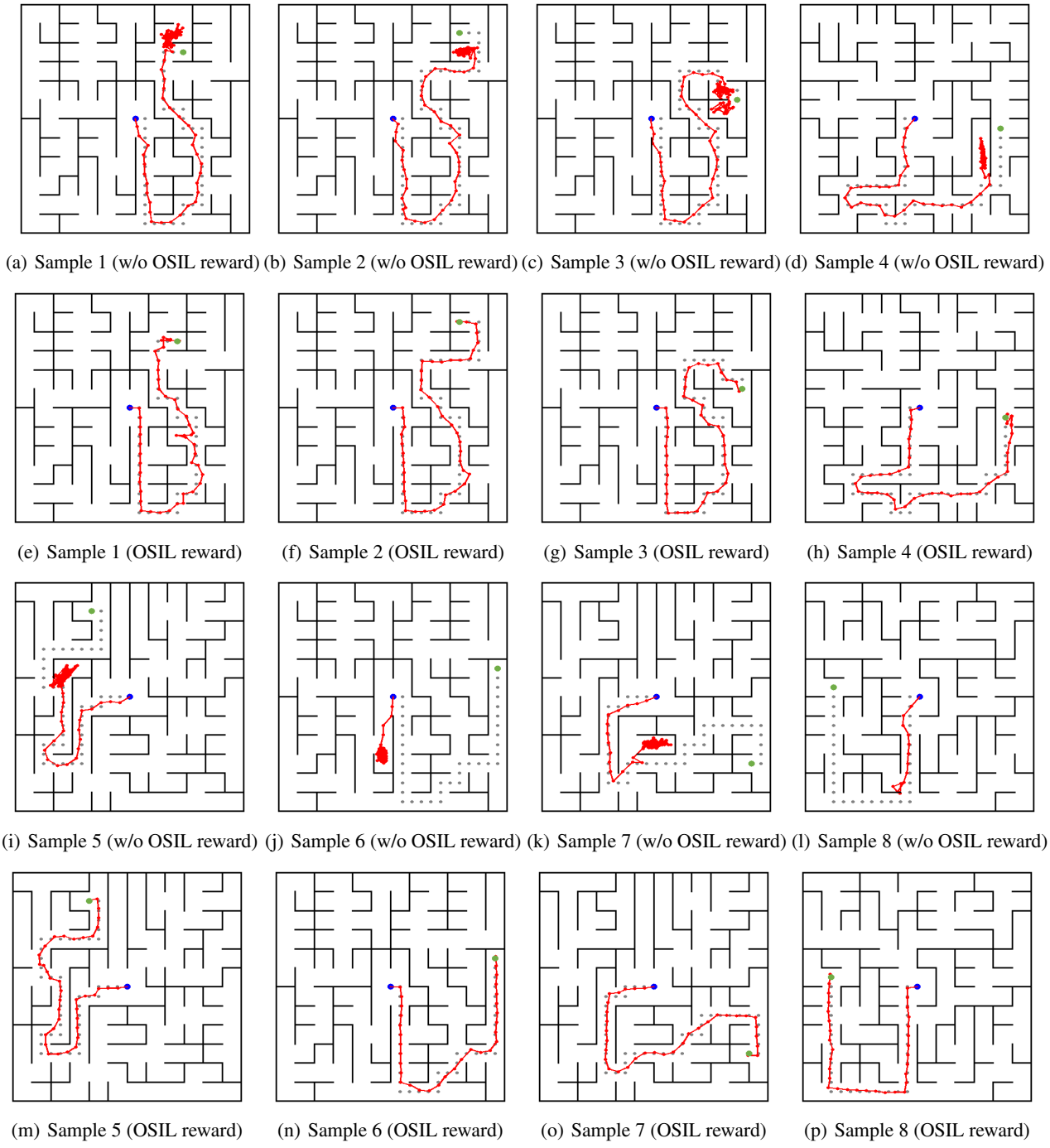
(i) Demonstration E (j) Demonstration F (k) Demonstration G (l) Demonstration H

(m) Attention pattern E (n) Attention pattern F (o) Attention pattern G (p) Attention pattern H

Figure 23: Illustrations of attention patterns in DDT. In Fig.(e)-(h) and Fig.(m)-(p), the horizontal and vertical axis represent the agent's trajectory index. The intensity of color within a row indicates the level of attention allocated by the agent to the corresponding expert state. The imitator agent actively aligns with expert states by leveraging the matched state for decision-making, *with higher attention values predominantly concentrated along the diagonal*.

L. Comparisons of Trajectories of DDT Trained by Different Rewards

In Fig. 24, we provide visualized comparison of trajectories of DDT trained with our OSIL reward and without. In the tasks, the agent knows its coordinate and is required to imitate demonstrations collected on new maps without obstacles. Through random sampling of multiple tasks, we have observed that in specific scenarios, intelligent agents without OSIL reward tend to encounter wall collisions or deviate from the correct path, resulting in losts. This behavior can potentially arise from their inclination to take shortcuts as a means to expedite reaching the goal.



(a) Sample 1 (w/o OSIL reward) (b) Sample 2 (w/o OSIL reward) (c) Sample 3 (w/o OSIL reward) (d) Sample 4 (w/o OSIL reward)

(e) Sample 1 (OSIL reward) (f) Sample 2 (OSIL reward) (g) Sample 3 (OSIL reward) (h) Sample 4 (OSIL reward)

(i) Sample 5 (w/o OSIL reward) (j) Sample 6 (w/o OSIL reward) (k) Sample 7 (w/o OSIL reward) (l) Sample 8 (w/o OSIL reward)

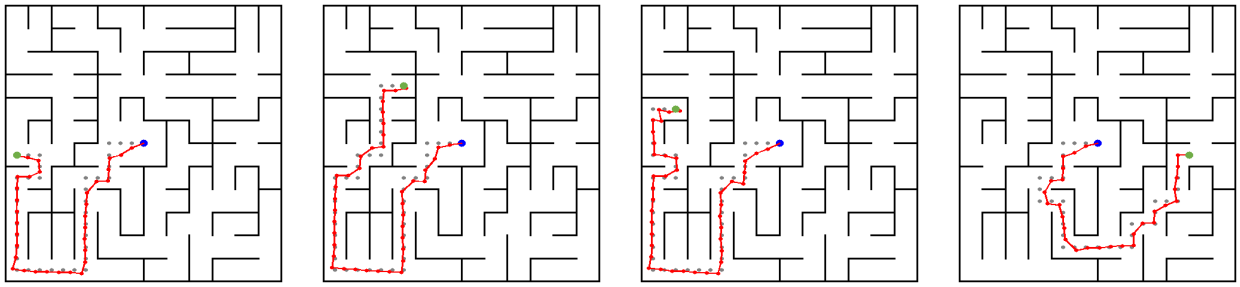
(m) Sample 5 (OSIL reward) (n) Sample 6 (OSIL reward) (o) Sample 7 (OSIL reward) (p) Sample 8 (OSIL reward)

Figure 24: comparison of trajectories of DDT trained with or without the OSIL reward.

M. More Examples of DDT Trajectories

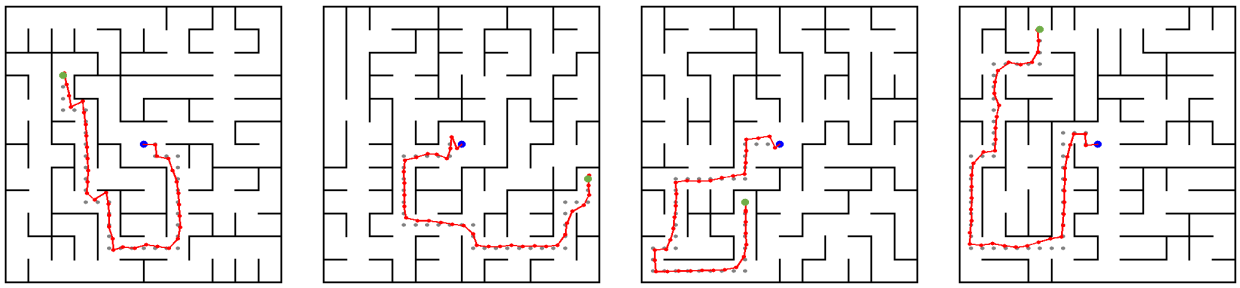
M.1. DDT Trajectories in Maze Environments

In Fig. 25-32, we show the trajectories generated by DDT agents in all of the tasks.



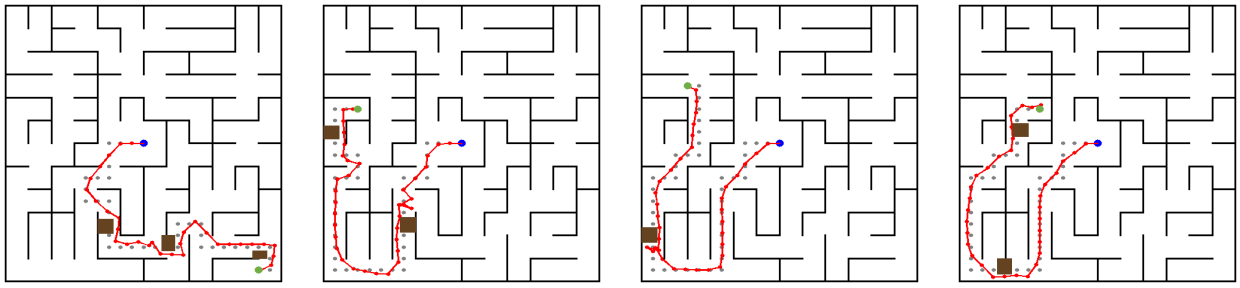
(a) Sample 1 (b) Sample 2 (c) Sample 3 (d) Sample 4

Figure 25: Illustration of trajectories generated by DDT agents where the agents are trained on the single-map setting without obstacles and with coordinates provided.



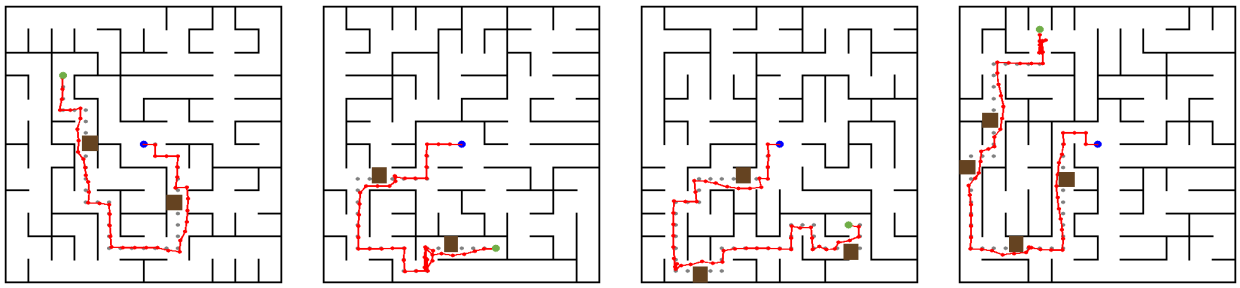
(a) Sample 1 (b) Sample 2 (c) Sample 3 (d) Sample 4

Figure 26: Illustration of trajectories generated by DDT agents where the agents are trained on the multi-map setting without obstacles and with coordinates provided.



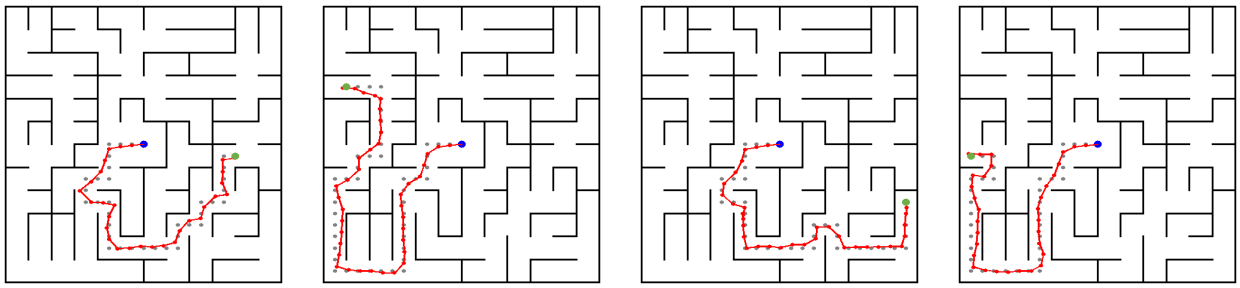
(a) Sample 1 (b) Sample 2 (c) Sample 3 (d) Sample 4

Figure 27: Illustration of trajectories generated by DDT agents where the agents are trained on the single-map setting with obstacles and with coordinates provided.



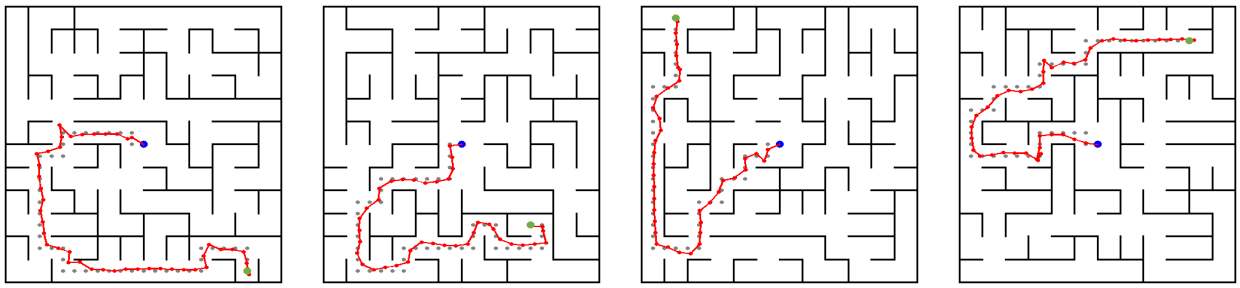
(a) Sample 1 (b) Sample 2 (c) Sample 3 (d) Sample 4

Figure 28: Illustration of trajectories generated by DDT agents where the agents are trained on the multi-map setting with obstacles and with coordinates provided.



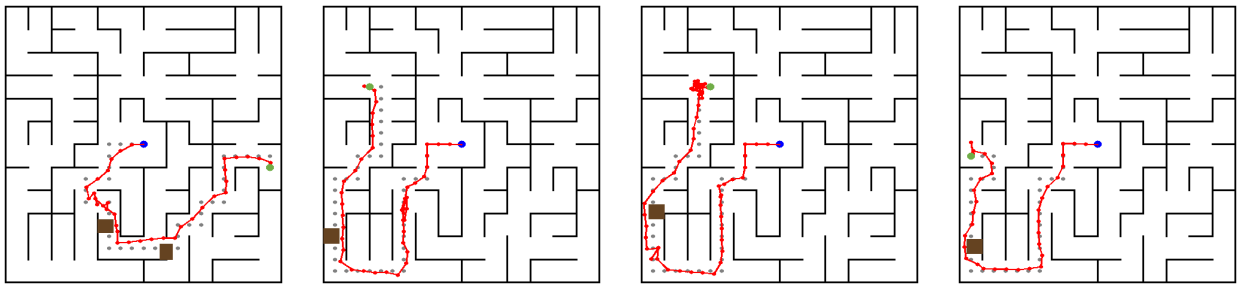
(a) Sample 1 (b) Sample 2 (c) Sample 3 (d) Sample 4

Figure 29: Illustration of trajectories generated by DDT agents where the agents are trained on the single-map setting without obstacles and without coordinates provided.



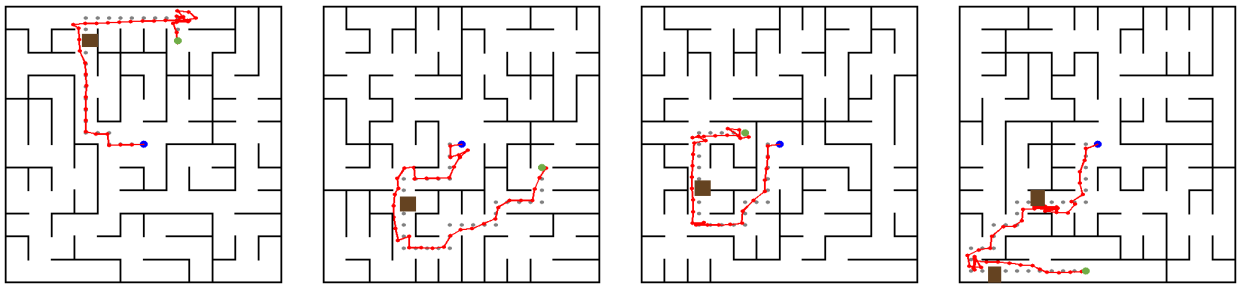
(a) Sample 1 (b) Sample 2 (c) Sample 3 (d) Sample 4

Figure 30: Illustration of trajectories generated by DDT agents where the agents are trained on the multi-map setting without obstacles and without coordinates provided.



(a) Sample 1 (b) Sample 2 (c) Sample 3 (d) Sample 4

Figure 31: Illustration of trajectories generated by DDT agents where the agents are trained on the single-map setting with obstacles and without coordinates provided.

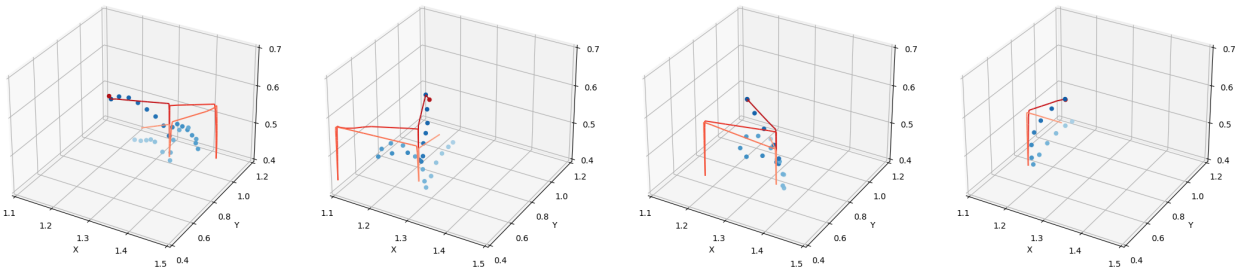


(a) Sample 1 (b) Sample 2 (c) Sample 3 (d) Sample 4

Figure 32: Illustration of trajectories generated by DDT agents where the agents are trained on the multi-map setting with obstacles and without coordinates provided.

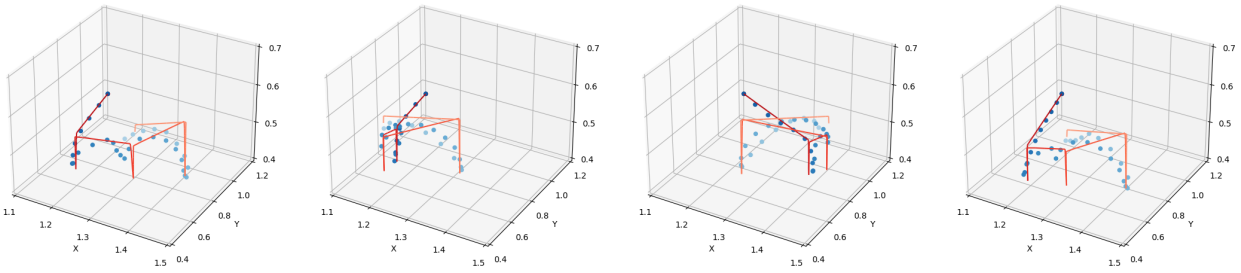
M.2. DDT Trajectories in Robot Manipulation Environments

In Fig. 33-38, we show the trajectories generated by DDT agents in all of the tasks. In all of the figures, the red line represents the expert demonstrations collected by sequentially executing predefined heuristic primitives, *the lighter the latter*. The blue dots are event points denoting the agent trajectory, *the lighter the latter*. The event points generated by our method always distribute around the demonstration trajectories, which demonstrates that the agent actively matches expert states for decision making. Besides, we recorded the corresponding videos in the supplementary material.



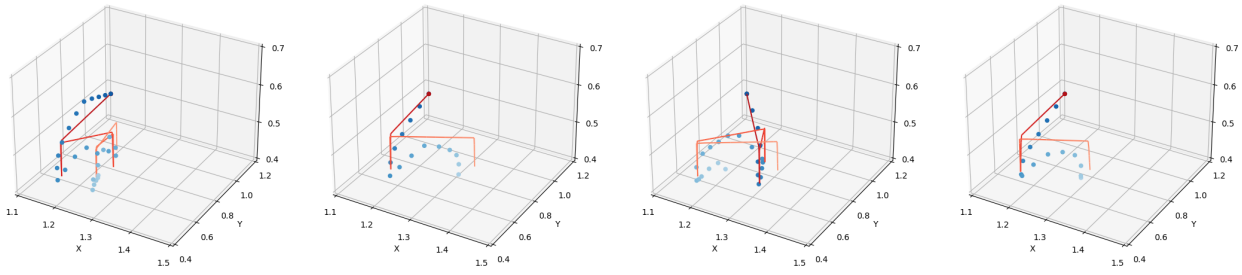
(a) Sample 1 (b) Sample 2 (c) Sample 3 (d) Sample 4

Figure 33: Illustration of trajectories generated by DDT agents and corresponding demonstrations in object-grasping tasks. The agents are trained on object-grasping tasks and tested with unseen object-grasping demonstrations. The robot needs to grasp the target object without colliding with other objects.



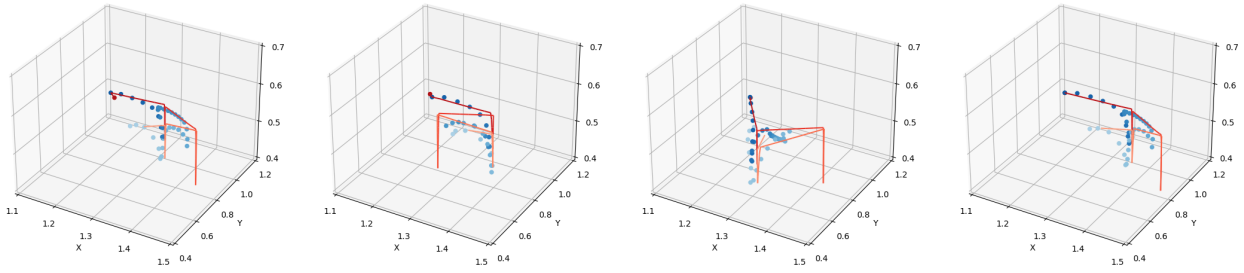
(a) Sample 1 (b) Sample 2 (c) Sample 3 (d) Sample 4

Figure 34: (a)-(d) Robot manipulation trajectories generated by DDT agents. The agents are trained on object-stacking tasks and tested with unseen object-stacking demonstrations. The robot needs to stack three initially placed objects together.



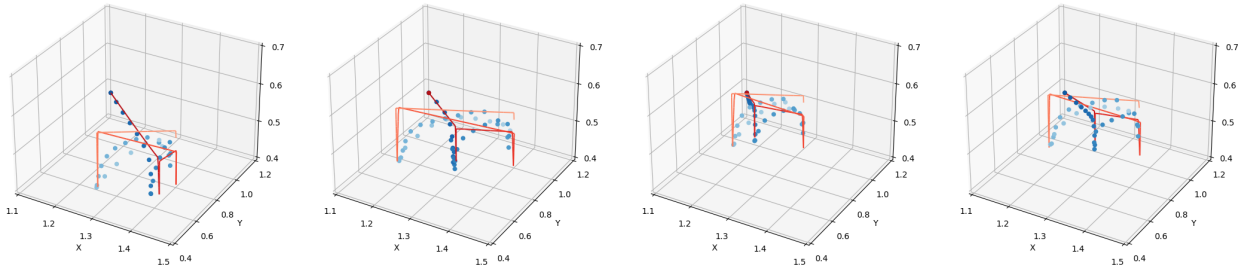
(a) Sample 1 (b) Sample 2 (c) Sample 3 (d) Sample 4

Figure 35: Illustration of trajectories generated by DDT agents and corresponding demonstrations in object-collecting tasks. The agents are trained on object-collecting tasks and tested with unseen object-collecting demonstrations. The robot needs to collect all objects scattered over the desk and place them in the specified area (yellow).



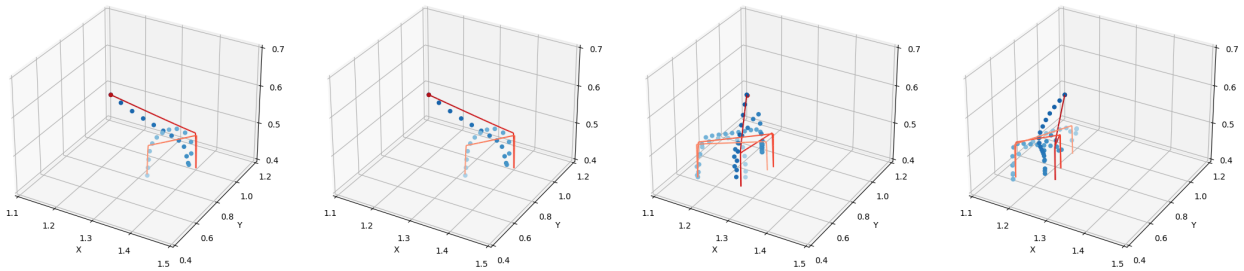
(a) Sample 1 (b) Sample 2 (c) Sample 3 (d) Sample 4

Figure 36: Illustration of trajectories generated by DDT agents trained to imitate three types of manipulation tasks simultaneously. The agents are tested with unseen object-grasping demonstrations.



(a) Sample 1 (b) Sample 2 (c) Sample 3 (d) Sample 4

Figure 37: Illustration of trajectories generated by DDT agents trained to imitate three types of manipulation tasks simultaneously. The agents are tested with unseen object-stacking demonstrations.



(a) Sample 1 (b) Sample 2 (c) Sample 3 (d) Sample 4

Figure 38: Illustration of trajectories generated by DDT agents trained to imitate three types of manipulation tasks simultaneously. The agents are tested with unseen object-collecting demonstrations.