Diffusion Imitation from Observation

Bo-Ruei Huang Chun-Kai Yang Chun-Mao Lai Dai-Jie Wu Shao-Hua Sun
Department of Electrical Engineering, National Taiwan University

Abstract

Learning from observation (LfO) aims to imitate experts by learning from state-only demonstrations without requiring action labels. Existing adversarial imitation learning approaches learn a generator agent policy to produce state transitions that are indistinguishable to a discriminator that learns to classify agent and expert state transitions. Despite its simplicity in formulation, these methods are often sensitive to hyperparameters and brittle to train. Motivated by the recent success of diffusion models in generative modeling, we propose to integrate a diffusion model into the adversarial imitation learning from observation framework. Specifically, we employ a diffusion model to capture expert and agent transitions by generating the next state, given the current state. Then, we reformulate the learning objective to train the diffusion model as a binary classifier and use it to provide "realness" rewards for policy learning. Our proposed framework, Diffusion Imitation from Observation (DIFO), demonstrates superior performance in various continuous control domains, including navigation, locomotion, manipulation, and games. Project page: https://nturobotlearninglab.github.io/DIFO

1 Introduction

Learning from demonstration (LfD) [26, 46, 51, 63, 86] aims to acquire policies that can perform desired skills by imitating expert trajectories represented as sequences of state-action pairs, eliminating the necessity of reward functions. Recent advancements in LfD have enabled the deployment of reliable and robust learned policies in various domains, such as robot learning [17, 20, 28, 30], strategy games [22, 47, 71], and self-driving [7, 8, 62, 64]. LfD's dependence on accurately labeled actions remains a substantial limitation, particularly in scenarios where obtaining expert actions is challenging or costly. Moreover, most LfD methods assume that the demonstrator and imitator share the same embodiment, inherently preventing cross-embodiment imitation.

To address these issues, learning from observation (LfO) methods [66, 76, 85] seek to imitate experts from state-only sequences, thereby removing the need for action labels and allowing learning from experts with different embodiments. Schmidt and Jiang [66], Torabi et al. [75], Yang et al. [82] proposed learning inverse dynamic models (IDMs) that can infer action labels from state sequences and subsequently reformulate LfO as LfD. Nevertheless, acquiring sufficiently aligned data with the expert's data distribution to train IDMs remains an unresolved challenge. On the other hand, adversarial imitation learning (AIL) [32, 76, 81] employs a generator policy learning to imitate an expert, while a discriminator differentiates between the data produced by the policy and the actual expert data. Despite its simplicity in formulation, AIL methods can be brittle to learn and are often sensitive to hyperparameters [2, 13].

Recent works have explored leveraging diffusion models' ability in generative modeling and achieved encouraging results in imitation learning and planning [29, 31, 44]. For example, diffusion policies [6, 49] learn to denoise actions with injected noises conditioned on states, allowing for modeling multimodal expert behaviors. Moreover, Chen et al. [5] proposed to model expert state-action pairs with a diffusion model and then provide gradients to train a behavioral cloning policy to improve

Correspondence to: Shao-Hua Sun <shaohuas@ntu.edu.tw>

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

its generalizability. Nevertheless, these works require action labels, fundamentally limiting their applicability to learning from observation.

In this work, we introduce Diffusion Imitation from Observation (DIFO), a novel adversarial imitation learning from observation method that employs a diffusion model as a discriminator to provide rewards for policy learning. Specifically, we design a diffusion model that learns to capture expert and agent state transitions by generating the subsequent state conditioning on the current state. We reformulate the denoising objective of diffusion models as a binary classification task, allowing for the diffusion model to distinguish expert and agent transitions. Then, provided with the "realness" rewards from the diffusion model, the policy imitates the expert by producing transitions that look indistinguishable from expert transitions.

We compare our method DIFO to various existing LfO methods in various continuous control domains, including navigation, locomotion, manipulation, and games. The experimental results show that DIFO consistently exhibits superior performance. Moreover, DIFO demonstrates better data efficiency. The visualized learned reward function and generated state distributions verify the effectiveness of our proposed learning objective for the diffusion model.

2 Related work

Learning from demonstration (LfD). LfD approaches imitate experts from collected demonstrations, consisting of state and action sequences. Behavioral cloning (BC) [51, 70] formulates LfD as a supervised learning problem by learning a state-to-action mapping. Inverse reinforcement learning (IRL) [1, 45, 60] extracts a reward function from demonstrations and uses it to learn a policy through reinforcement learning. In contrast, this work aims to learn from state-only demonstrations, requiring no action label.

Learning from observation (LfO). LfO [11, 73] learns from state-only demonstrations, *i.e.*, state sequences, making it suitable for scenarios where action labels are unobservable or costly to obtain, and allowing for learning from experts with a different embodiment. To tackle LfO, one popular direction is to learn an inverse dynamics model (IDM) for an agent that can recover an action for a pair of consecutive states [66, 75, 82]. However, there is no apparent mechanism to efficiently collect tuples of state, next state, and action that align with the expert state sequences, which makes it difficult to learn a good IDM. On the other hand, adversarial imitation learning from observation (AILfO) [23, 37, 54, 76] resemble the idea of generative adversarial networks (GANs) [19], where an agent generator policy is rewarded by a discriminator learning to distinguish the expert state transitions from the agent state transitions. Despite the encouraging results, the AILfO trainings are often brittle and sensitive to hyperparameters [2, 13]. Recent works also use generative models to predict state transitions and use the prediction to guide policy learning using log-likelihood [12], ELBO [84], or conditional entropy [25]. However, these methods depend highly on the accuracy of the generative models. In contrast, our work aims to improve the sample efficiency and robustness of AILfO by employing a diffusion model as a discriminator.

Learning from video (LfV). Extending from LfO, LfV specifically considers learning from image-based states, *i.e.*, videos, by leveraging recent advancements in computer vision, *e.g.*, multi-view learning [69], image and video comprehension and generation [3, 12, 16, 33, 43, 65, 68], foundation models [9, 42], and optical flow and tracking [31, 80]. Yet, these methods are mostly specifically designed for learning from videos, and cannot be trivially adapted for vectorized states.

Diffusion models. Diffusion models are state-of-the-art generative models capable of capturing and generating high-dimensional data distributions [27, 72]. Diffusion models have been widely adopted for generating images [56, 61], videos [4], 3D structures [52], and speech [41, 53]. Recent works also have explored using the ability to model multimodal distributions of diffusion models for LfD [5, 6, 35, 49, 79], where expert demonstrations could exhibit significant variability [39]. Our work aims to employ the capability of diffusion models for improving AIRLfO.

3 Preliminary

3.1 Learning from observation

Consider environments represented as a Markov decision process (MDP) defined as a tuple $(\mathcal{S}, \mathcal{A}, r, \mathcal{P}, \rho_0, \gamma)$ of state space $\mathcal{S}$, action space $\mathcal{A}$, reward function $r(s, a, s')$, transition dynamics

$\mathcal{P}(s'|s,a)$, initial state distribution ρ_0 and discounting factor γ. We define a policy $\pi(a|s)$ that takes actions from state inputs and generates trajectories $\tau = (s_0, a_0, s_1, \ldots, s_{|\tau|})$. The policy is trained to maximize the sum of discounted rewards $\mathbb{E}_{(s_0,a_0,\ldots,s_{|\tau|})\sim\pi}\left[\sum_{i=0}^{|\tau|-1}\gamma^i r\left(s_i, a_i, s_{i+1}\right)\right]$.

In imitation learning, the environment rewards cannot be observed. Instead, a set of expert demonstrations $\tau_E = \{\tau_0, \ldots, \tau_N | \tau_i \sim \pi_E\}$ is given, which generated by unknown expert policy π_E. We aim to learn the agent policy π_A to generate a similar trajectory distribution with expert demonstrations. Moreover, in the learning from observation (LfO) setting, where expert action labels are absent, agents learn exclusively from state-only observations represented by sequences $\tau = (s_0, s_1, \ldots, s_{|\tau|})$. We use the LfO setting in this work.

Inverse reinforcement learning (IRL). One of the general approaches to imitation learning is IRL. This approach learns a reward function r from transitions, *i.e.*, (s,a) in LfD or (s,s') in LfO, that maximizes the reward of expert transitions and minimizes that of agent transitions. The learned reward function can thereby be used for reinforcement learning to train the policy to imitate expert.

3.2 Denoising Diffusion Probabilistic Models

Diffusion models have emerged as state-of-the-art generative models capable of producing high-dimensional data and modeling multimodal distributions. Our work leverages the Denoising Diffusion Probabilistic Model (DDPM) [27], a latent variable model that generates data through a denoising process. The training procedure of the diffusion model consists of forward and reverse processes. In the forward process, Gaussian noise is progressively added to the clean data, following a predefined noise schedule. The process is formulated as $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$, where $\mathbf{x}_0$ is the clean data, $\boldsymbol{\epsilon}$ is the Gaussian noise, t denotes the time step within the whole process with step T and $\bar{\alpha}_t$ is the scheduled noise level at the current time step. Conversely, the reverse process, denoted by $p_\phi(\mathbf{x}_{t-1}|\mathbf{x}_t)$, is designed to reconstruct the original data by estimating the previously injected noise based on the given noise level. This is achieved by optimizing $\mathcal{L} = \mathbb{E}_{t\sim T,\boldsymbol{\epsilon}\sim\mathcal{N}(0,1)}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\phi(\mathbf{x}_t, t)\|^2\right]$, where ϕ denotes the diffusion model.

4 Approach

We propose Diffusion Imitation from Observation (DIFO), a novel learning from observation framework integrating a diffusion model into the AIL framework, which is illustrated in Figure 1. Specifically, we utilize a diffusion model to model expert and agent state transitions; then, we learn an agent policy to imitate the expert via reinforcement learning by using the diffusion model to provide rewards based on how "real" agent state transitions are.

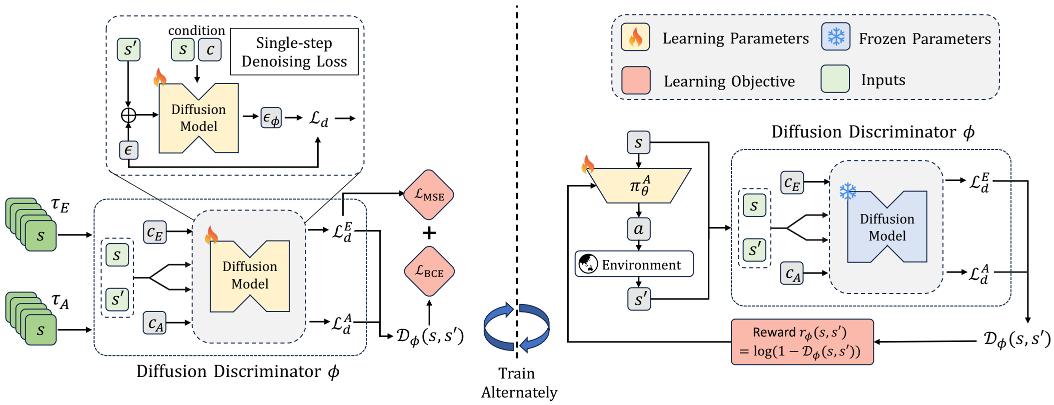
4.1 Modeling expert transitions via diffusion model

Motivated by the recent success in using diffusion models for generative modeling, we use a conditional diffusion model to model expert state transitions. Specifically, given a state transition $(\mathbf{s}, \mathbf{s}')$, the diffusion model conditions on the current state $\mathbf{s}$ and generates the next state $\mathbf{s}'$. We adopt DDPM [27] and define the reverse process as $p_\phi(\mathbf{s}'_{t-1}|\mathbf{s}'_t, \mathbf{s})$, where $t \in T$ and ϕ is the diffusion model, which is trained by minimizing the denoising MSE loss:

$$\mathcal{L}_d(\mathbf{s}, \mathbf{s}') = \mathbb{E}_{t\sim T,\boldsymbol{\epsilon}\sim\mathcal{N}(0,1)}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\phi(\boldsymbol{s'}_t, t|\boldsymbol{s})\|^2\right], \tag{1}$$

where ϵ denotes the noise sampled from a Gaussian distribution and ϵ_ϕ denotes the noise predicted by the diffusion model. Once the diffusion model is trained, we can generate an expert next state conditioned on any given state by going through the diffusion generation process.

State-distance reward. To train a policy π to imitate the expert from a given state $\mathbf{s}$, we can first sample an action from the policy and obtain the next state $\mathbf{s}_\pi{}'$ by interacting with the environment. Next, we generate a predicted next state $\mathbf{s}_\phi{}'$ using the diffusion model. Then, to bring the state distribution of the policy closer to the expert's, we can optimize the policy using reinforcement learning by setting the distance of the two next states $d(\mathbf{s}_\pi{}', \mathbf{s}_\phi{}')$ as a reward, where d denotes some distance function that evaluates how close two states are. However, a good distance function varies



(a) Learning diffusion discriminator

(b) Learning policy with diffusion reward

Figure 1: **Diffusion Imitation from Observation (DIFO).** We propose Diffusion Imitation from Observation (DIFO), a novel adversarial imitation learning from observation framework employing a conditional diffusion model. **(a) Learning diffusion discriminator.** In the *discriminator step* the diffusion model learns to model a state transition $(\mathbf{s}, \mathbf{s}')$ by conditioning on the current state $\mathbf{s}$ and generates the next state $\mathbf{s}'$. With the additional condition on binary expert and agent labels (c_E/c_A), we construct the diffusion discriminator to distinguish expert and agent transitions by leveraging the single-step denoising loss as a likelihood approximation. **(b) Learning policy with diffusion reward.** In the *policy step*, we optimize the policy with reinforcement learning according to rewards calculated based on the diffusion discriminator's output $\log(1 - \mathcal{D}_\phi(\mathbf{s}, \mathbf{s}'))$.

from one domain to another. Moreover, predicting the diffusion model next state ${\mathbf{s}_\phi}'$ can be very time-consuming since it requires T denoising steps.

Denoising reward. We aim to provide rewards for policy learning while avoiding choosing distance function and going through the diffusion generation process. To this end, we take inspiration from Li et al. [38], which shows that the denoising loss approximates the evidence lower bound (ELBO) of the likelihood. Our key insight is to leverage the denoising loss calculated from a state and the policy next state $\mathcal{L}_d(\mathbf{s}, {\mathbf{s}_\pi}')$, or $\mathcal{L}_d$ in short, as an indicator of how well the policy next state fits the expert distribution. That said, a low $\mathcal{L}_d$ means that the policy produces a next state close to the expert next state, while a high $\mathcal{L}_d$ means that the diffusion model does not recognize this policy next state. Hence, we can use $-\mathcal{L}_d$ as reward to learn a policy to imitate the expert by taking actions to produce next states that can be recognized by the diffusion model. Note that this denoising reward can be computed using a single denoising step.

4.2 Diffusion model as a discriminator

The previous section describes how we can use the denoising loss as a reward for policy learning via reinforcement learning. However, the policy can learn to exploit a frozen diffusion model by discovering states that lead to a low denoising loss while being drastically different from expert states. To mitigate this issue, we incorporate principles from the AIL framework by training the diffusion model to recognize both the transitions from the expert and agent. To this end, we additionally condition the model on a binary label $c \in \{c_E, c_A\}$, where c_E represents the expert label and c_A represents the agent label, both implemented as one-hot encoding, resulting in the following denoising losses given a state transition $(\mathbf{s}, \mathbf{s}')$:

$$\mathcal{L}_d^E(\mathbf{s}, \mathbf{s}') = \mathbb{E}_{t \sim T, \boldsymbol{\epsilon} \sim \mathcal{N}(0,1)} \left[\left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\phi(\boldsymbol{s}'_t, t | \boldsymbol{s}, c_E) \right\|^2 \right], \tag{2}$$

$$\mathcal{L}_d^A(\mathbf{s}, \mathbf{s}') = \mathbb{E}_{t \sim T, \boldsymbol{\epsilon} \sim \mathcal{N}(0,1)} \left[\left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\phi(\boldsymbol{s}'_t, t | \boldsymbol{s}, c_A) \right\|^2 \right]. \tag{3}$$

With this formulation and an optimized diffusion model, an expert transition should yield a low $\mathcal{L}_d^E$ and a high $\mathcal{L}_d^A$, while an agent transition should yield a high $\mathcal{L}_d^E$ and a low $\mathcal{L}_d^A$. Thus, we construct a diffusion discriminator that can determine if a transition is close to expert as follows:

$$\mathcal{D}_\phi(\mathbf{s}, \mathbf{s}') = \sigma(\lambda_\sigma(\mathcal{L}_d^A(\mathbf{s}, \mathbf{s}') - \mathcal{L}_d^E(\mathbf{s}, \mathbf{s}'))), \tag{4}$$

where σ is the sigmoid function for normalization and λ_σ is a hyperparameter to control the sensitivity. To turn this diffusion discriminator as a binary classifier to classify agent and expert transitions, we train it to optimize the binary cross entropy (BCE):

$$\mathcal{L}_{\text{BCE}} = \mathbb{E}_{(\mathbf{s},\mathbf{s}')\sim\tau_E}\left[\log(1-\mathcal{D}_\phi(\mathbf{s},\mathbf{s}'))\right] + \mathbb{E}_{(\mathbf{s},\mathbf{s}')\sim\pi_A}\left[\log(\mathcal{D}_\phi(\mathbf{s},\mathbf{s}'))\right]. \tag{5}$$

By optimizing $\mathcal{L}_{\text{BCE}}$, online interactions with the agent are leveraged as negative samples. Given expert transitions, the model should minimize $\mathcal{L}_d^E$ and maximize $\mathcal{L}_d^A$, resulting in a higher score closer to 1. Conversely, when the input is sampled from the agent, the model aims to maximize $\mathcal{L}_d^E$ and minimize $\mathcal{L}_d^A$, outputting a lower score closer to 0. The higher the score is, the more likely a transition is expert. Hence, we can learn a policy to imitate the expert using $\mathcal{D}_\phi$ as rewards. In contrast to MLP binary discriminators used in existing AIL works like GAIL, which maps high-dimensional inputs to a one-dimensional logit, our diffusion discriminator learns to predict high-dimensional noise patterns. This is inherently more challenging to overfit, addressing one of the key instabilities in GAIL.

4.3 Diffusion Imitation from Observation

We present Diffusion Imitation from Observation (DIFO) an adversarial imitation learning from observation framework that trains a policy and a discriminator in turns. In the *discriminator step*, the discriminator learns to classify expert and agent transition by optimizing $\mathcal{L}_{\text{BCE}}$. Furthermore, to ensure the diffusion loss of expert data is optimized so that it approximates the ELBO, the diffusion model also optimizes $\mathcal{L}_d^E$ by sampling from expert demonstrations. [1]

$$\mathcal{L}_{\text{MSE}} = \mathbb{E}_{t\sim T,\boldsymbol{\epsilon}\sim\mathcal{N}(0,1),(\mathbf{s},\mathbf{s}')\sim\tau_E}\left[\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_\phi(\boldsymbol{s'}_t, t|\boldsymbol{s}, c_E)\right\|^2\right], \tag{6}$$

resulting in the overall objective:

$$\mathcal{L}_D = \lambda_{\text{MSE}}\mathcal{L}_{\text{MSE}} + \lambda_{\text{BCE}}\mathcal{L}_{\text{BCE}}, \tag{7}$$

where λ_{MSE} and λ_{BCE} are hyperparameters adjusting the importance of each term. In the *policy step*, to provide the policy rewards based on the "realness" $\mathcal{D}_\phi$ of the agent transitions, we adopt the GAIL reward function [24]:

$$r_\phi(\mathbf{s},\mathbf{s}') = \log(1-\mathcal{D}_\phi(\mathbf{s},\mathbf{s}')), \tag{8}$$

where $\mathcal{D}_\phi$ is computed with a single denoising step. We justify the feasibility of sampling only one denoising step in Section 5.8. We can optimize the policy using any RL algorithm. The DIFO framework is illustrated in Figure 1 and the algorithm is presented in Appendix A.

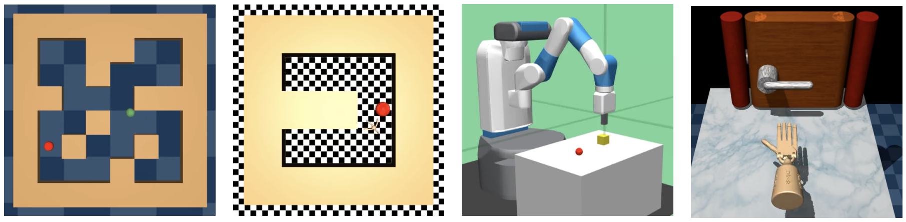
5 Experiments

5.1 Environments

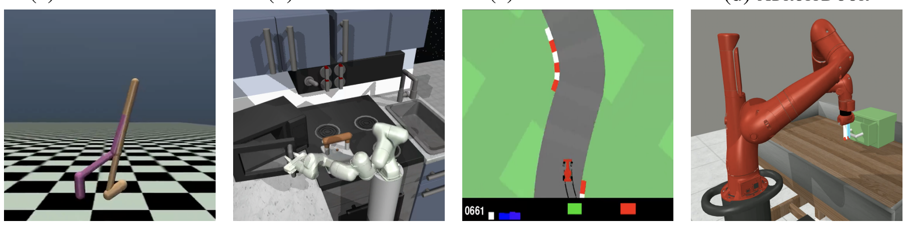
In this section, we introduce environments, tasks, and how expert demonstrations are collected. All environment trajectories, except CarRacing, are fixed-horizon to prevent biased information about success [32]. Further details can be found in Appendix B.

- **PointMaze**: A navigation task for a 2-DoF agent with the medium maze, see Figure 2a. A point agent is trained to navigate from an initial position to a goal. The goal and initial position of the agent are randomly sampled. The agent observes its position, velocity, and goal position. The agent applies linear forces in the x and y directions to navigate the maze and reach the goal. We collect 60 demonstrations (36 000 transitions) using a controller from Fu et al. [14].
- **AntMaze**: A task containing both locomotion and navigation, which presents a significantly more challenging variant of the **PointMaze**, as shown in Figure 2b. The quadruped ant learns to navigate from an initial position to a goal by controlling the torque of its legs, where both the goal and initial position of the ball are also randomly sampled. Notice that this environment serves as a high-dimensional state space task with 29-dimension state space. We use 100 demonstrations (7000 transitions) from Minari [50].

[1]We experiment with optimizing $\mathcal{L}_{\text{MSE}}$ with agent data ($\mathcal{L}_d^A$), leading to unstable training (see Appendix C).



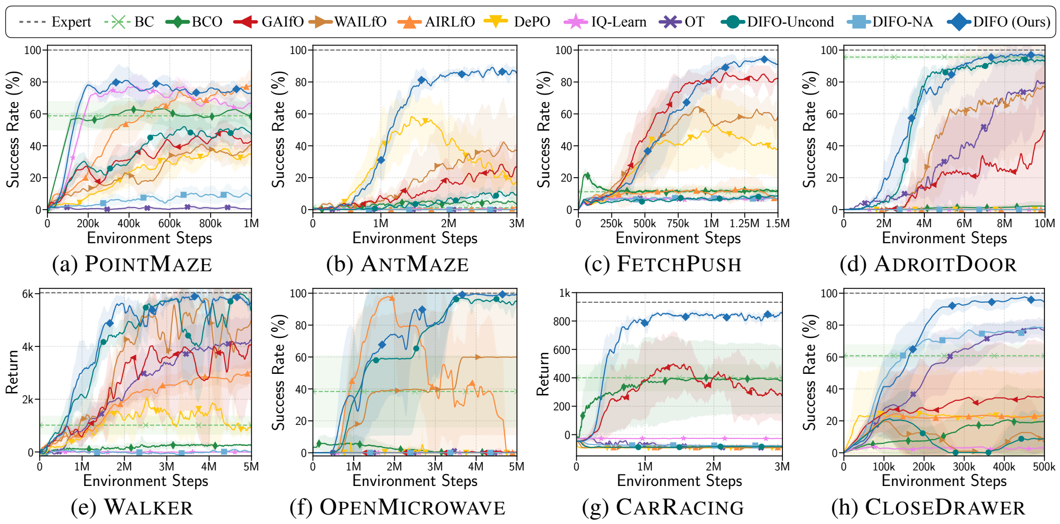
(a) POINTMAZE (b) ANTMAZE (c) FETCHPUSH (d) ADROITDOOR



(e) WALKER (f) OPENMICROWAVE (g) CARRACING (h) CLOSEDRAWER

Figure 2: **Environments & tasks. (a) POINTMAZE**: A point agent (green) is trained to navigate to the goal (red). **(b) ANTMAZE**: A high-dimensional locomotion navigation task for an 8-DoF quadruped ant to navigate to the goal (red). **(c) FETCHPUSH**: A manipulation task to move a block (yellow) to the target (red). **(d) ADROITDOOR**: A high-dimension manipulation task to undo the latch and swing the door open. **(e) WALKER**: A locomotion task for a 6-DoF hopper to maintain at the highest speed while keeping balance. **(f) OPENMICROWAVE**: A manipulation task to control the robot arm to open the microwave with joint space control. **(g) CARRACING**: An image-based task to control the car to complete the track in the shortest time. **(h) CLOSEDRAWER**: An image-based manipulation task to control the robot arm to close the drawer.

- **FETCHPUSH**: The goal is to control a 7-DoF Fetch robot arm to push a block to a target position on a table, see Figure 2c. Both the block and target positions are randomly sampled. The robot is controlled by small displacements of the gripper in XYZ coordinates, which has a 28-dimension state space and a 4-dimension action space. We generate 50 demonstrations (2500 transitions) using an expert policy trained by SAC [21].
- **ADROITDOOR**: A manipulation task to undo the latch and swing the door open, see Figure 2d. The position of the door is randomly placed. It is based on the Adroit manipulation platform [34], with 39-dimension state space and 28-dimension action space containing all the joints. It serves as a high-dimensional state and action space task. We use 50 demonstrations (10 000 transitions) from the dataset released by Fu et al. [14].
- **WALKER**: A locomotion task of a 6-DoF Walker2D in MuJoCo [74], as shown in Figure 2e. The goal is to walk forward by applying torques on the six hinges. Initial joint states are added with uniform noise. We generate 1000 transitions using an expert policy trained by SAC [21].
- **OPENMICROWAVE**: A manipulation task to control a 9-DoF Franka robot arm to open the microwave door, as shown in Figure 2f. The environment has a 59-dimension state space and a 9-dimension continuous action space to control the angular velocity of each joint. It serves as a high-dimensional state and action space task. We use 5 demonstrations (300 transitions) from the dataset released by Fu et al. [14].
- **CARRACING**: An image-based control task aimed at directing a car to complete a track as quickly as possible. Observations consist of top-down frames, as shown in Figure 2g. Tracks are generated randomly in every episode. The car has continuous action space to control the throttle, steering, and breaking. We generate 340 transitions using an expert policy trained by PPO [67].
- **CLOSEDRAWER**: An image-based manipulation task from Meta-World [83] requires the agent to control a Sawyer robot arm to close a drawer. Observations consist of fixed perspective frames, as shown in Figure 2h. The robot has continuous action space to control the gripper in XYZ coordinates, and the initial poses of the robot and the drawer are randomized in every episode. We generate 100 transitions using a scripted policy.



(a) POINTMAZE (b) ANTMAZE (c) FETCHPUSH (d) ADROITDOOR

(e) WALKER (f) OPENMICROWAVE (g) CARRACING (h) CLOSEDRAWER

Figure 3: **Learning performance and efficiency.** We evaluate all the methods with five random seeds and report their success rates in POINTMAZE, ANTMAZE, FETCHPUSH, ADROITDOOR, OPENMICROWAVE, and CLOSEDRAWER, and their returns in WALKER, and CARRACING. The standard deviation is shown as the shaded area. Our proposed method, DIFO, demonstrates more stable and faster learning performance compared to the baselines.

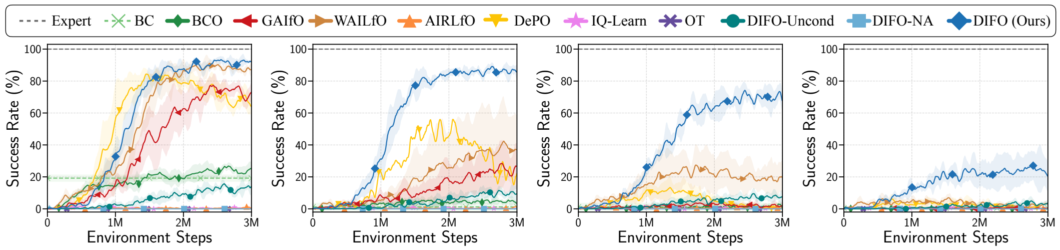
5.2 Baselines and variants

We compare our method DIFO with the following baselines:

- **Behavioral Cloning (BC)** [51] learns a state-to-action mapping using supervised learning without any interaction with the environment. Note that BC is the only baseline having privileged access to ground truth action labels.
- **Behavioral Cloning from Observation (BCO)** [75] first learns an inverse dynamic model through self-supervised exploration, and uses it to reconstruct action from state-only observation. BCO then uses these action labels to perform behavioral cloning.
- **Generative Adversarial Imitation from Observation (GAIfO)** [76], trains a GAIL MLP discriminator taking state transitions (s, s') as input, instead of state-action pairs (s, a).
- **Wasserstein Adversarial Imitation from Observation (WAIfO)** is a LfO variant of WAIL [81], taking (s, s') as input. WAIL replaces the learning objective of the discriminator from Jensen-Shannon divergence (GAIL) to Wasserstein distance.
- **Adverserial Inverse Reinforcement Learning from Observation (AIRLfO)** is a LfO variant of AIRL [13]. AIRL modifies the discriminator output to disentangle task-relevant information from transition dynamics. Similarly to GAIfO, AIRLfO takes (s, s') as input instead of (s, a).
- **Decoupled Policy Optimization (DePO)** [40] decouples the policy into a high-level state planner and an inverse dynamics model, utilizing embedded decoupled policy gradient and generative adversarial training.
- **Inverse soft-Q Learning for Imitation (IQ-Learn)** [15] directly learns a policy in Q-space from demonstrations without explicit reward construction. We use the state-only setting for LfO.
- **Optimal Transport (OT)** [48] derives a proxy reward function for RL by measuring the distance between probability distributions. We use the state-only setting for LfO.

In addition to the existing methods, we also compare DIFO with its variants:

- **DIFO-Non-Adversarial (DIFO-NA)** follows the method introduced in Section 4.1, which first pretrains a conditional diffusion model on expert demonstrations, and simply takes the denoising reward $-\mathcal{L}_d(s, s')$ for policy training.



(a) 200 Trajectories (b) 100 Trajectories (c) 50 Trajectories (d) 25 Trajectories

Figure 4: **Data efficiency.** We vary the amount of available expert demonstrations in ANTMAZE. Our proposed method DIFO consistently outperforms other methods when the number of expert demonstrations decreases, highlighting the data efficiency of DIFO.

- **DIFO-Unconditioned (DIFO-Uncond)** removes the condition on s, and denoises both s and s'. It is optimized only with $\mathcal{L}_{\text{BCE}}$. Namely, replacing the MLP discriminator with a diffusion discriminator from GAIfO. It serves as a baseline showing the effect of network architecture.

5.3 Experimental results

We report the success rates in POINTMAZE, ANTMAZE, FETCHPUSH, and ADROITDOOR, and return in WALKER and CARRACING of all the methods in Figure 3. Each method is reported with the mean value and standard deviation with five random seeds for all the tasks. BC's performance is shown as horizontal lines since BC does not leverage environmental interactions. The expert's performance (gray horizontal lines) in goal-directed tasks, *i.e.*, POINTMAZE, ANTMAZE, FETCHPUSH, ADROITDOOR, is 100%. More details of training and evaluation can be found in Appendix G.

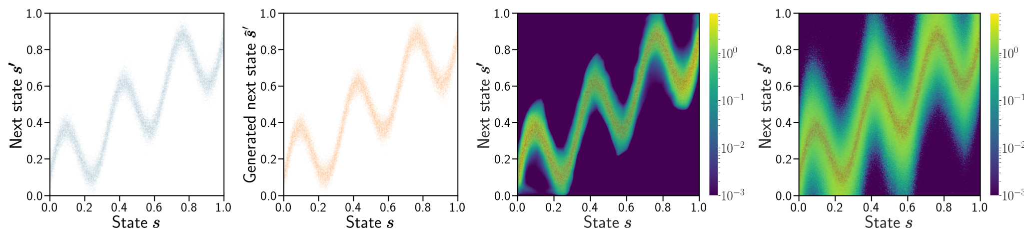
Our proposed method DIFO consistently outperforms or matches the performance of the best-performing baseline in all the tasks, highlighting the effectiveness of integrating a conditional diffusion model into the AIL framework. In ANTMAZE, ADROITDOOR, and CARRACING, DIFO outperforms the baselines and converges significantly faster, indicating its efficiency in modeling expert behavior and providing effective rewards even in high-dimensional state and action spaces. Moreover, DIFO presents more stable training results, with relatively low variance compared to other AIL methods. Notably, although BC has access to action labels, it still fails in most tasks with more randomness. This is because BC relies solely on learning from the observed expert dataset, unlike the LfO methods that utilize online interaction with environments, BC is susceptible to covariate shifts [36, 58, 59] and requires a substantial amount of expert data to achieve coverage of the dataset. The result indicates the significance of online interactions. OT only successfully learns in environments like ADROITDOOR, WALKER, and CLOSEDRAWER, where trajectory variety is limited. OT computes distances at the trajectory level rather than the transition level, which requires monotonic trajectories, making it struggle in tasks with diverse trajectories. In contrast, our method generates rewards at the transition level, allowing us to identify transition similarities even when facing substantial trajectory variability.

Variants of DIFO, *i.e.*, DIFO-Uncond, and DIFO-NA, perform poorly in most tasks. DIFO-NA learns poorly in most of the tasks except CLOSEDRAWER, underscoring diffusion loss could be a reasonable metric for the discriminator while it is still necessary to model agent online interaction data to prevent the diffusion model from being exploited by the policy. On the other hand, DIFO-Uncond performs comparably to other AIL baselines but shows instability across different tasks, this highlighting the importance of modeling transitions using a diffusion model.

We also verify DIFO's capability to model stochastic distribution in Appendix D.

5.4 Data efficiency

To investigate the data efficiency, *i.e.*, how much expert data is required for learning, we vary the number of expert trajectories in ANTMAZE and report the performance of all the methods in Figure 4. Specifically, we use 200, 100, 50, and 25 expert trajectories, each containing 14 000, 7000, 3500, and 1750 transitions, respectively.



(a) Expert (s, s') (b) Generated $(s, \widehat{s}')$ (c) GAIfO reward (d) DIFO reward

Figure 6: **Reward function visualization and generated distribution on SINE. (a)** The expert state transition distribution. **(b)** The state transition distribution generated by the DIFO diffusion model. **(c-d)** The visualized reward functions learned by GAIfO and DIFO, respectively. DIFO produces smoother rewards outside of the expert distribution, allowing for facilitating policy learning.

The results demonstrate that DIFO learns faster compared to all the baselines with various amounts of demonstrations, highlighting its sample efficiency. Specifically, as the number of demonstrations decreases from 200 to 50, DIFO's performance drops modestly from an 80% success rate to 70%, whereas WAILfO, the best-performing baseline when given 200 expert trajectories, experiences a substantial decline to a 20% success rate. Furthermore, when the number of demonstrations is reduced to 25, all other baselines fail to learn, with success rates nearing zero. In contrast, DIFO maintains a success rate of around 20%, underscoring its superior data efficiency. This data efficiency highlights DIFO's potential for real-world applications, where collecting expert demonstrations can be costly.

5.5 Generating data using diffusion models

To investigate whether the DIFO diffusion model can closely capture the expert distribution, we generate trajectories with the diffusion model in POINTMAZE. Specifically, we take a trained diffusion discriminator of DIFO and autoregressively generate a sequence of next states starting from an initial state sampled in the expert dataset. We visualize four pairs of expert trajectories and the corresponding generated trajectories in Figure 5.

The results show that our diffusion model can accurately generate trajectories similar to those of the expert. It is worth noting that the diffusion model can generate trajectories that differ from the expert trajectories while still completing the task, such as the example on the bottom right of Figure 5, where the diffusion model produces even shorter trajectories than the scripted expert policy. Additional expert trajectories and the corresponding generated trajectories are presented in Appendix E.

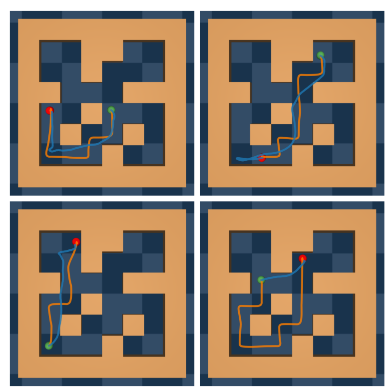


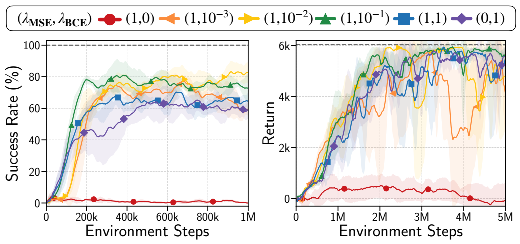
Figure 5: **Generated trajectories under POINTMAZE.** The green point marks the initial state. The red point marks the goal. The blue trace represents the generated trajectory and the orange trace represents the corresponding expert trajectory.

5.6 Visualized learned reward functions

We aim to visualize and analyze the reward functions learned by DIFO. To this end, we introduce a toy environment SINE in which both the state and action space are 1-dimension with range $[0, 1]$. We generate expert state-only demonstrations by sampling from the distribution $s' = \sin 6\pi s + s + \mathcal{N}(0,\, 0.05^2)$. The sampled expert state transitions are plotted in Figure 6a.

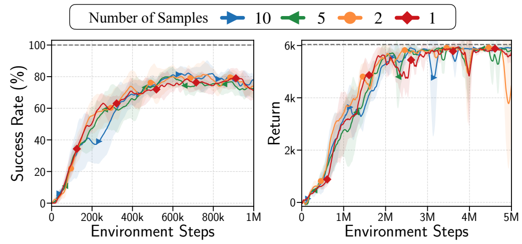
Reconstructed distribution. Given the expert state distribution, we generate a distribution of next states using the diffusion model of DIFO and visualize the distribution in Figure 6b. The generated distribution closely resembles the expert distribution, which again verifies the modeling capability of the conditional diffusion model.

Visualized learned reward functions. We visualize the reward functions learned by GAIfO and DIFO in SINE in Figure 6c and Figure 6d, respectively. The result shows that the reward function



(a) PointMaze (b) Walker

Figure 7: **The effect of λ_{MSE} and λ_{BCE}.** We vary the values of λ_{MSE} and λ_{BCE} in PointMaze and Walker, showcasing DIFO's robustness to hyperparameters and emphasizing the importance of both $\mathcal{L}_{\text{BCE}}$ and $\mathcal{L}_{\text{MSE}}$.



(a) PointMaze (b) Walker

Figure 8: **Different numbers of denoising step samples for reward computation.** We vary the number of denoising step samples to compute rewards. The result indicates the number of samples does not significantly affect the performance.

learned by GAIfO drops dramatically once it deviates from expert distribution, while that of DIFO presents a smoother contour to the region outside the distribution, which allows for bringing a learning agent closer to the expert even when agent's behaviors are far from the expert behavior.

5.7 Ablation study on λ_{MSE} and λ_{BCE}

We hypothesize that both $\mathcal{L}_{\text{MSE}}$ and $\mathcal{L}_{\text{BCE}}$ are important for efficiency learning. To examine the effect of $\mathcal{L}_{\text{MSE}}$ and $\mathcal{L}_{\text{BCE}}$ and verify the hypothesis, we vary the ratio of λ_{MSE} and λ_{BCE} in PointMaze and Walker, including $\mathcal{L}_{\text{BCE}}$ only and $\mathcal{L}_{\text{MSE}}$ only, *i.e.*, $\lambda_{\text{MSE}} = 0$ and $\lambda_{\text{BCE}} = 0$. As shown in Figure 7, the results emphasize the significance of introducing both $\mathcal{L}_{\text{MSE}}$ and $\mathcal{L}_{\text{BCE}}$, since they enable the model to simultaneously model expert behavior ($\mathcal{L}_{\text{MSE}}$) and perform binary classification ($\mathcal{L}_{\text{BCE}}$). Without $\mathcal{L}_{\text{MSE}}$, the performance slightly decreases as it does not modeling expert behaviors. Without $\mathcal{L}_{\text{BCE}}$, the model fails to learn as it does not utilize negative samples, *i.e.*, agent data. Moreover, when we vary the ratio of λ_{MSE} and λ_{BCE}, DIFO maintains stable performance, demonstrating DIFO is relatively insensitive to hyperparameter variations.

5.8 Ablation study on the number of samples for reward computation

To investigate the robustness of our rewards, we conducted experiments with varying numbers of denoising step samples in PointMaze and Walker. We take the mean of losses computed from different numbers of samples, *i.e.*, multiple t, to compute rewards. As presented in the Figure 8, the performance of DIFO is stable under different numbers of samples. As a result, we use a single denoising step sample to compute the reward for the best efficiency. We also investigate the stability of rewards under different numbers of samples in Appendix F.

6 Conclusion

We present Diffusion Imitation from Observation (DIFO), a novel adversarial imitation learning from observation framework. DIFO leverages a conditional diffusion model as a discriminator to distinguish expert state transitions from those of the agent, while the agent policy learns to produce state transitions that are indistinguishable from the expert's for the diffusion discriminator. Experimental results demonstrate that DIFO outperforms existing learning from observation methods, including BCO, GAIfO, WAIfO, AIRLfO, DePO, OT, and IQ-Learn, across continuous control tasks in various domains, including navigation, manipulation, locomotion, and image-based games. The visualization of the reward function learned by DIFO shows that it can generalize well to states unseen from expert state transitions by utilizing agent data. Also, the DIFO diffusion model is able to accurately capture expert state transitions and can generate predicted trajectories that are similar to those of expert's.

Acknowledgement

This work was supported by the National Science and Technology Council, Taiwan (113-2222-E-002-007-). Shao-Hua Sun was supported by the Yushan Fellow Program by the Ministry of Education, Taiwan.

References

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*, 2004.

[2] Paul Barde, Julien Roy, Wonseok Jeon, Joelle Pineau, Chris Pal, and Derek Nowrouzezahrai. Adversarial soft advantage fitting: Imitation learning without policy optimization. In *Neural Information Processing Systems*, 2020.

[3] Chethan Bhateja, Derek Guo, Dibya Ghosh, Anikait Singh, Manan Tomar, Quan Vuong, Yevgen Chebotar, Sergey Levine, and Aviral Kumar. Robotic offline rl from internet videos via value-function pre-training. In *International Conference on Robotics and Automation*, 2023.

[4] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv:2311.15127*, 2023.

[5] Shang-Fu Chen, Hsiang-Chun Wang, Ming-Hao Hsu, Chun-Mao Lai, and Shao-Hua Sun. Diffusion model-augmented behavioral cloning. In *International Conference on Machine Learning*, 2024.

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

[7] Seongjin Choi, Jiwon Kim, and Hwasoo Yeo. Trajgail: Generating urban vehicle trajectories using generative adversarial imitation learning. *Transportation Research Part C: Emerging Technologies*, 2021.

[8] Daniel Coelho, Miguel Oliveira, and Vitor Santos. Rlfold: Reinforcement learning from online demonstrations in urban autonomous driving. In *Association for the Advancement of Artificial Intelligence*, 2024.

[9] Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, et al. Open x-embodiment: Robotic learning datasets and rt-x models. In *International Conference on Robotics and Automation*, 2024.

[10] Rodrigo de Lazcano, Kallinteris Andreas, Jun Jet Tai, Seungjae Ryan Lee, and Jordan Terry. Gymnasium robotics, 2023.

[11] Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *International Conference on Robotics and Automation*, 2017.

[12] Alejandro Escontrela, Ademi Adeniji, Wilson Yan, Ajay Jain, Xue Bin Peng, Ken Goldberg, Youngwoon Lee, Danijar Hafner, and Pieter Abbeel. Video prediction models as rewards for reinforcement learning. In *Neural Information Processing Systems*, 2023.

[13] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018.

[14] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv:2004.07219*, 2020.

[15] Divyansh Garg, Shuvam Chakraborty, Chris Cundy, Jiaming Song, and Stefano Ermon. Iq-learn: Inverse soft-q learning for imitation. In *Neural Information Processing Systems*, 2021.

[16] Dibya Ghosh, Chethan Anand Bhateja, and Sergey Levine. Reinforcement learning from passive data via latent intentions. In *International Conference on Learning Representations*, 2023.

[17] Alessandro Giusti, Jérôme Guzzi, Dan C. Cireşan, Fang-Lin He, Juan P. Rodríguez, Flavio Fontana, Matthias Faessler, Christian Forster, Jürgen Schmidhuber, Gianni Di Caro, Davide Scaramuzza, and Luca M. Gambardella. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 2016.

[18] Adam Gleave, Mohammad Taufeeque, Juan Rocamonde, Erik Jenner, Steven H. Wang, Sam Toyer, Maximilian Ernestus, Nora Belrose, Scott Emmons, and Stuart Russell. imitation: Clean imitation learning implementations. *arXiv:2211.11972*, 2022.

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014.

[20] Abhishek Gupta, Clemens Eppner, Sergey Levine, and Pieter Abbeel. Learning dexterous manipulation for a soft robotic hand from human demonstrations. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016.

[21] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018.

[22] Jack Harmer, Linus Gisslén, Jorge del Val, Henrik Holst, Joakim Bergdahl, Tom Olsson, Kristoffer Sjöö, and Magnus Nordin. Imitation learning with concurrent actions in 3d games. In *IEEE Conference on Computational Intelligence and Games*, 2018.

[23] Peter Henderson, Wei-Di Chang, Pierre-Luc Bacon, David Meger, Joelle Pineau, and Doina Precup. Optiongan: Learning joint reward-policy options using generative adversarial inverse reinforcement learning. In *Association for the Advancement of Artificial Intelligence*, 2018.

[24] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, 2016.

[25] Tao Huang, Guangqi Jiang, Yanjie Ze, and Huazhe Xu. Diffusion reward: Learning rewards via conditional video diffusion. In *European Conference on Computer Vision*, 2024.

[26] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys*, 2017.

[27] A Jain J Ho. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020.

[28] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, 2022.

[29] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, 2022.

[30] Edward Johns. Coarse-to-fine imitation learning: Robot manipulation from a single demonstration. In *International Conference on Robotics and Automation*, 2021.

[31] Po-Chen Ko, Jiayuan Mao, Yilun Du, Shao-Hua Sun, and Joshua B Tenenbaum. Learning to act from actionless videos through dense correspondences. In *International Conference on Learning Representations*, 2024.

[32] Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *International Conference on Learning Representations*, 2019.

[33] Aviral Kumar, Anikait Singh, Frederik Ebert, Mitsuhiko Nakamoto, Yanlai Yang, Chelsea Finn, and Sergey Levine. Pre-training for robots: Offline rl enables learning new tasks from a handful of trials. In *Robotics: Science and Systems*, 2022.

[34] Vikash Kumar. *Manipulators and Manipulation in high dimensional spaces*. PhD thesis, University of Washington, Seattle, 2016.

[35] Chun-Mao Lai, Hsiang-Chun Wang, Ping-Chun Hsieh, Yu-Chiang Frank Wang, Min-Hung Chen, and Shao-Hua Sun. Diffusion-reward adversarial imitation learning. *arXiv:2405.16194*, 2024.

[36] Michael Laskey, Sam Staszak, Wesley Yu-Shu Hsieh, Jeffrey Mahler, Florian T. Pokorny, Anca D. Dragan, and Ken Goldberg. Shiv: Reducing supervisor burden in dagger using support vectors for efficient learning from demonstrations in high dimensional state spaces. In *International Conference on Robotics and Automation*, 2016.

[37] Youngwoon Lee, Andrew Szot, Shao-Hua Sun, and Joseph J. Lim. Generalizable imitation learning from observation via inferring goal proximity. In *Neural Information Processing Systems*, 2021.

[38] Alexander C. Li, Mihir Prabhudesai, Shivam Duggal, Ellis Brown, and Deepak Pathak. Your diffusion model is secretly a zero-shot classifier. In *International Conference on Computer Vision*, 2023.

[39] Yunzhu Li, Jiaming Song, and Stefano Ermon. Infogail: Interpretable imitation learning from visual demonstrations. In *Neural Information Processing Systems*, 2017.

[40] Minghuan Liu, Zhengbang Zhu, Yuzheng Zhuang, Weinan Zhang, Jianye Hao, Yong Yu, and Jun Wang. Plan your target and learn your skills: Transferable state-only imitation learning via decoupled policy optimization. In *International Conference on Machine Learning*, 2022.

[41] Songxiang Liu, Dan Su, and Dong Yu. Diffgan-tts: High-fidelity and efficient text-to-speech with denoising diffusion gans. *arXiv:2201.11972*, 2022.

[42] Yecheng Jason Ma, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. Liv: Language-image representations and rewards for robotic control. In *International Conference on Machine Learning*, 2023.

[43] Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *International Conference on Learning Representations*, 2023.

[44] Utkarsh Aashu Mishra, Shangjie Xue, Yongxin Chen, and Danfei Xu. Generative skill chaining: Long-horizon skill planning with diffusion models. In *Conference on Robot Learning*, 2023.

[45] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, 2000.

[46] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, Jan Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 2018.

[47] Ricardo Palma, Antonio A. Sánchez-Ruiz, Marco Antonio Gómez-Martín, Pedro Pablo Gómez-Martín, and Pedro Antonio González-Calero. Combining expert knowledge and learning from demonstration in real-time strategy games. In *Case-Based Reasoning Research and Development*, 2011.

[48] Georgios Papagiannis and Yunpeng Li. Imitation learning with sinkhorn distances. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2022.

[49] Tim Pearce, Tabish Rashid, Anssi Kanervisto, David Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, and Sam Devlin. Imitating human behaviour with diffusion models. In *International Conference on Learning Representations*, 2023.

[50] Rodrigo Perez-Vicente, Omar Younis, John Balis, and Alex Davey. Minari: A dataset api for offline reinforcement learning, 2023.

[51] Dean A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 1991.

[52] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *International Conference on Learning Representations*, 2023.

[53] Vadim Popov, Ivan Vovk, Vladimir Gogoryan, Tasnima Sadekova, and Mikhail Kudinov. Grad-tts: A diffusion probabilistic model for text-to-speech. In *International Conference on Machine Learning*, 2021.

[54] Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Visual adversarial imitation learning using variational models. In *Neural Information Processing Systems*, 2021.

[55] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 2021.

[56] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2022.

[57] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention*, 2015.

[58] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *International Conference on Artificial Intelligence and Statistics*, 2010.

[59] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, 2011.

[60] Stuart Russell. Learning agents for uncertain environments. In *Annual Conference on Computational Learning Theory*, 1998.

[61] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Neural Information Processing Systems*, 2022.

[62] Tanmay Vilas Samak, Chinmay Vilas Samak, and Sivanathan Kandhasamy. Robust behavioral cloning for autonomous vehicles using end-to-end imitation learning. *SAE International Journal of Connected and Automated Vehicles*, 2021.

[63] Stefan Schaal. Learning from demonstration. In *Advances in Neural Information Processing Systems*, 1997.

[64] Oliver Scheel, Luca Bergamini, Maciej Wolczyk, Błażej Osiński, and Peter Ondruska. Urban driver: Learning to drive from real-world demonstrations using policy gradients. In *Conference on Robot Learning*, 2022.

[65] Karl Schmeckpeper, Oleh Rybkin, Kostas Daniilidis, Sergey Levine, and Chelsea Finn. Reinforcement learning with videos: Combining offline observations with interaction. In *Conference on Robot Learning*, 2020.

[66] Dominik Schmidt and Minqi Jiang. Learning to act without actions. In *International Conference on Learning Representations*, 2024.

[67] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.

[68] Younggyo Seo, Kimin Lee, Stephen L James, and Pieter Abbeel. Reinforcement learning with action-free pre-training from videos. In *International Conference on Machine Learning*, 2022.

[69] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *International Conference on Robotics and Automation*, 2018.

[70] Nur Muhammad Mahi Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. In *Neural Information Processing Systems*, 2022.

[71] Andy Shih, Stefano Ermon, and Dorsa Sadigh. Conditional imitation learning for multi-agent games. In *International Conference on Human-Robot Interaction*, 2022.

[72] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[73] Shao-Hua Sun, Hyeonwoo Noh, Sriram Somasundaram, and Joseph Lim. Neural program synthesis from diverse demonstration videos. In *International Conference on Machine Learning*, 2018.

[74] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *International Conference On Intelligent Robots and Systems*, 2012.

[75] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *International Joint Conference on Artificial Intelligence*, 2018.

[76] Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. In *International Conference on Machine Learning*, 2019.

[77] Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, 2023.

[78] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, Dhruv Nair, Sayak Paul, William Berman, Yiyi Xu, Steven Liu, and Thomas Wolf. Diffusers: State-of-the-art diffusion models, 2022.

[79] Bingzheng Wang, Guoqiang Wu, Teng Pang, Yan Zhang, and Yilong Yin. Diffail: Diffusion adversarial imitation learning. In *Association for the Advancement of Artificial Intelligence*, 2024.

[80] Chuan Wen, Xingyu Lin, John So, Kai Chen, Qi Dou, Yang Gao, and Pieter Abbeel. Any-point trajectory modeling for policy learning. In *Robotics: Science and Systems*, 2023.

[81] Huang Xiao, Michael Herman, Joerg Wagner, Sebastian Ziesche, Jalal Etesami, and Thai Hong Linh. Wasserstein adversarial imitation learning. *arXiv:1906.08113*, 2019.

[82] Chao Yang, Xiaojian Ma, Wenbing Huang, Fuchun Sun, Huaping Liu, Junzhou Huang, and Chuang Gan. Imitation learning from observations by minimizing inverse dynamics disagreement. In *Neural Information Processing Systems*, 2019.

[83] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, 2020.

[84] Xingyuan Zhang, Philip Becker-Ehmck, Patrick van der Smagt, and Maximilian Karl. Action inference by maximising evidence: zero-shot imitation from observation with world models. In *Neural Information Processing Systems*, 2024.

[85] Zhuangdi Zhu, Kaixiang Lin, Bo Dai, and Jiayu Zhou. Off-policy imitation learning from observations. In *Neural Information Processing Systems*, 2020.

[86] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Association for the Advancement of Artificial Intelligence*, 2008.

Appendix

A Pseudocode of DIFO

Algorithm 1 Diffusion Imitation from Observation (DIFO)

Input: Expert demonstrations $\tau_E \sim \pi_E$, policy π_A and diffusion model $\mathcal{D}_\phi$
while π_A not converges **do**
  Sample agent transitions $\tau_i \sim \pi_A$ by interacting with the environment
  Calculate discriminator loss $\mathcal{L}_D$ with Eq. 7
  Update discriminator parameters ϕ with $\nabla \mathcal{L}_D$
  Calculate IRL reward $r_\phi(\mathbf{s}, \mathbf{s}')$ with Eq. 8
  Update π_A with respect to r_ϕ with any RL algorithm
end while

B Environment & task details

The environments we used for our experiments in Section 5 are from Gymnasium [10, 77]. We list names, version numbers, dimensions of state and action spaces in Table 1. All environments we used are continuous action spaces.

Table 1: **Enviroments.** Detailed setting of each Task.

Task	ID	Observation space	Action space	Fixed horizon
PointMaze	PointMaze_Medium-v3	8	2	True
AntMaze	AntMaze_UMaze-v4	31	8	True
FetchPush	FetchPush-v2	31	4	True
AdroitDoor	AdroitHandDoorCustom-v1	39	28	True
Walker	Walker2d-v4	17	6	True
OpenMicrowave	FrankaKitchen-v1	59	9	True
CarRacing	CarRacing-v2	$96 \times 96 \times 3$	3	False
CloseDrawer	N/A	$64 \times 64 \times 3$	4	True

Notably, we fix the horizon to prevent biased information about termination [32] in all environments except CarRacing since the goal of CarRacing is to complete the track as fast as possible instead of goal completion. Notice that though the objective of Walker is also not goal completion, the termination signal provides additional information about tumbling.

Preprocessing. In CarRacing, we preprocess the observation state by applying frame skipping with a factor of 2, resizing, and converting the image to grayscale, resulting in a 64×64 matrix. Finally, we stack 2 frames to form the input state.

We list the details of the expert demonstration datasets we used in Section 5 and how we collect them in Table 2. All of our expert data incorporates stochasticity to enhance diversity in trajectories. We add a small amount of noise to the experts' actions, providing stochasticity and multimodality in expert behaviors.

Table 2: **Expert observations.** Detailed information on collected expert observations in each Task.

Task	# of trajectories	# of transitions	Algorithm or Retrieved Source
PointMaze	60	36 000	D4RL [14]
AntMaze	100	7000	Minari [50]
FetchPush	50	2500	Own SAC expert
AdroitDoor	50	10 000	D4RL [14]
Walker	1	1000	Own SAC expert
OpenMicrowave	5	300	D4RL [14]
CarRacing	1	340	Own PPO expert
CloseDrawer	1	100	Own Script expert

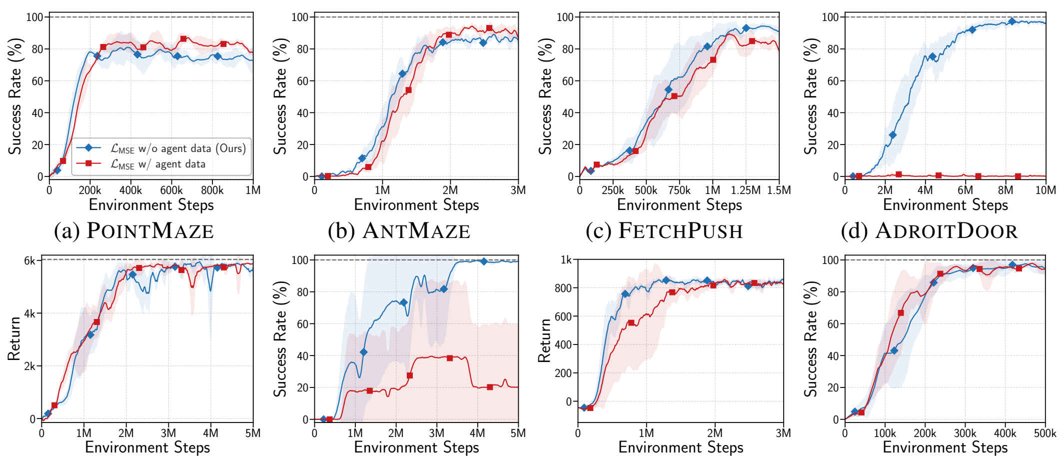
In Section 5.6, we introduce a toy environment SINE. We generate 25 000 state-only transition $(s,\ s')$, with transition function:

$$s' = \sin 6\pi s + s + \mathcal{N}(0,\ 0.05^2) \tag{9}$$

where $\mathcal{N}(\mu,\ \sigma^2)$ is a normal distribution noise with the mean value $\mu = 0$ and the standard deviation $\sigma = 0.05$.

C Optimizing $\mathcal{L}_{\text{MSE}}$ with agent data

Our method optimizes $\mathcal{L}_{\text{MSE}}$ (Eq. 6) to approximate the ELBO only using expert demonstrations. To investigate the effect of optimizing this MSE loss using agent data, we experiment with optimizing $\mathcal{L}_{\text{MSE}}$ with and without agent data on all tasks. The results are reported in Figure 9. We found that optimizing $\mathcal{L}_{\text{MSE}}$ with agent data can lead to slower and unstable convergence, especially in tasks with larger state and action spaces, *e.g.*, ADROITDOOR, where optimizing $\mathcal{L}_{\text{MSE}}$ leads to a 0% success rate. We hypothesize that optimizing $\mathcal{L}_{\text{MSE}}$ leads to unstable training because, during the early stage of training, the agent policy changes frequently and generates diverse transitions. This diversity leads to a consistent distribution shift for the diffusion model, making the diffusion model unstable to learn. As a result, the overall performance can be less stable. Hence, our method is designed to only optimize $\mathcal{L}_{\text{MSE}}$ using expert demonstrations.



(a) POINTMAZE (b) ANTMAZE (c) FETCHPUSH (d) ADROITDOOR

(e) WALKER (f) OPENMICROWAVE (g) CARRACING (h) CLOSEDRAWER

Figure 9: **Optimizing $\mathcal{L}_{\text{MSE}}$ with and without agent data.** We evaluate optimizing $\mathcal{L}_{\text{MSE}}$ with and without agent data in all tasks. The results show that optimizing $\mathcal{L}_{\text{MSE}}$ with agent data (red) leads to slower convergence and less stable performance.

D Stochastic environment

We create a stochastic version of ANTMAZE environment where Gaussian noise is added to the agent's actions before they are applied in the environment. The magnitude of the noise is 0.5, resulting in the actual action taken in the environment would be $action = action + 0.5\mathcal{N}(0, 1)$. Given the action space of this environment is $[-1, 1]$, this represents a high level of stochasticity.

Figure 10 shows that the performance of our method remains robust even under such high stochasticity, indicating our model's ability to adapt to stochastic environments effectively.

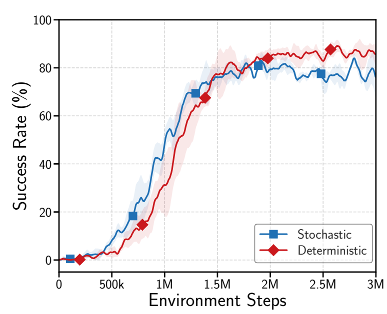


Figure 10: **Stochastic environment.** We apply addition Gaussian noises to actions to create a stochastic ANTMAZE. Our method DIFO maintains robust performance under large stochasticity.

E Full trajectories generations of POINTMAZE

More results of the POINTMAZE trajectory generation experiment in Section 5.5 can be found in Figure 11.

Figure 11: **Full result of generated trajectories under POINTMAZE.** The blue trace represents the generated trajectory and the orange trace represents the corresponding expert trajectory. DIFO can accurately model expert behavior and generate successful trajectories.

F The stability of rewards

To further verify the stability of the rewards under different numbers of denoising step samples, we present the standard deviation to mean ratio of the rewards from 500 computations results in Table 3. The values are averaged from a batch of 4096 transitions. The result shows that sampling a single denoising step is enough to produce a stable reward.

Table 3: **Reward ratio.** Standard deviation to mean rewards ratio over 500 computations, averaged from 4096 transitions. n is the number of denoising step samples to compute the reward.

Learning Progress	$n = 1$	$n = 2$	$n = 5$	$n = 10$
20%	0.323	0.237	0.294	0.246
40%	0.234	0.199	0.206	0.230
60%	0.201	0.175	0.157	0.206
80%	0.157	0.152	0.150	0.190
100%	0.142	0.133	0.145	0.161

G Training details

Our codebase inherits from *Imitation* [18], an open-source imitation learning framework based on *Stable Baselines3* [55]. We then describe the implementation of each baseline:

- GAIfO, AIRLfO, and BC follow the original implementation of *Imitation*.
- IQ-Learn and OT are modified from SAC in *Stable Baselines3* with the loss function borrow from the official implementations [15, 48].
- We implemented our own BCO, WAIfO, and DePO.

Table 4: **Hyperparameters.** The overview of the hyperparameters used for all the methods in every task. We abbreviate "Discriminator" as "Disc." in this table.

Method	Hyperparameter	PointMaze	AntMaze	FetchPush	AdroitDoor	Walker	CarRacing
RL Algorithm		SAC	SAC	SAC	SAC	PPO	PPO
BC	Batch Size	128	128	128	128	128	128
	Learning Rate	0.0003	0.0003	0.0003	0.0001	0.0003	0.0003
	L2 Weight	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	Training Steps	100 000	500 000	500 000	500 000	500 000	500 000
BCO	Batch Size	128	128	128	128	128	128
	Learning Rate	0.0003	0.0003	0.0003	0.0001	0.0003	0.0001
	L2 Weight	0.001	0.001	0.001	0.001	0.001	0.001
	Entropy Weight	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	α	0.01	0.2	1.0	5.0	1.0	5.0
	# Environment Steps	1 000 000	3 000 000	1 500 000	10 000 000	5 000 000	3 000 000
GAIfO	Batch Size	64	64	64	64	64	64
	Disc. Learning Rate	0.0001	0.0001	0.0001	0.0001	0.00001	0.0001
	Disc. Hidden Dimensions	128	128	128	128	128	128
	Disc. Hidden Layers	5	5	5	4	5	5
	Disc. Buffer Size	1 000 000	1 000 000	1 000 000	1 000 000	1 000 000	1 000 000
	# Environment Steps	1 000 000	3 000 000	1 500 000	10 000 000	5 000 000	3 000 000
WAIfO	Batch Size	64	64	64	64	64	64
	Disc. Learning Rate	0.0001	0.0001	0.0001	0.0001	0.00001	0.0001
	Disc. Buffer Size	1 000 000	1 000 000	1 000 000	1 000 000	1 000 000	1 000 000
	Disc. Hidden Dimensions	128	128	128	128	128	128
	Disc. Hidden Layers	5	5	5	4	5	5
	Regularize Epsilon	0.001	0.001	0.001	0.001	0.001	0.001
	Regularize Function	L_2	L_2	L_2	L_2	L_2	L_2
	Gradient Clipping	1	1	1	1	1	1
	# Environment Steps	1 000 000	3 000 000	1 500 000	10 000 000	5 000 000	3 000 000
AIRLfO	Batch Size	64	64	64	64	64	64
	Disc. Learning Rate	0.0001	0.0001	0.0001	0.00001	0.0001	0.00001
	Disc. Buffer Size	1 000 000	1 000 000	1 000 000	1 000 000	1 000 000	1 000 000
	Disc. Hidden Dimensions	128	128	128	128	128	128
	Disc. Hidden Layers	5	5	5	4	5	5
	# Environment Steps	1 000 000	3 000 000	1 500 000	10 000 000	5 000 000	3 000 000
DePO	Batch Size	64	64	64	64	64	64
	IDM. Batch Size	256	256	256	256	256	256
	IDM. Learning Rate	0.0003	0.0003	0.0003	0.0003	0.0003	0.0003
	# Environment Steps	1 000 000	3 000 000	1 500 000	10 000 000	5 000 000	3 000 000
IQ-Learn	Batch Size	256	256	256	256	256	256
	Actor Learning Rate	3×10^{-5}	3×10^{-5}	3×10^{-5}	3×10^{-5}	3×10^{-5}	3×10^{-5}
	Critic Learning Rate	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}	3×10^{-4}
	Entropy Coefficient	0.01	0.01	0.01	0.01	0.01	0.01
	# Environment Steps	1 000 000	3 000 000	1 500 000	10 000 000	5 000 000	3 000 000
OT	Batch Size	256	256	256	256	256	256
	Reward Scale	100	100	100	100	100	100
	# Expert Samples	10	10	10	10	10	10
	Learning Rate	0.0003	0.0003	0.0003	0.0003	0.0003	0.0003
	# Environment Steps	1 000 000	3 000 000	1 500 000	10 000 000	5 000 000	3 000 000
DIFO (Ours)	Batch Size	64	64	64	64	64	64
	Disc. Learning Rate	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	Disc. Buffer Size	1 000 000	1 000 000	1 000 000	1 000 000	1 000 000	1 000 000
	BCE Weight λ_{BCE}	0.1	0.01	0.01	0.001	0.01	0.1
	MSE Weight λ_{MSE}	1	1	1	1	1	1
	λ_σ	10	10	10	10	10	10
	U-Net Units	(256, 256, 256)	(256, 256, 256)	(256, 256, 256)	(256, 256, 256)	(256, 256, 256)	(256, 256, 256)
	Embedding Dimension	128	128	128	128	32	128
	Denoising Sample Range	(250, 750)	(250, 750)	(250, 750)	(250, 750)	(250, 750)	(250, 750)
	Denoising Timestep	1000	1000	1000	1000	1000	1000
	Logit Scale	10	10	10	10	10	10
	# Environment Steps	1 000 000	3 000 000	1 500 000	10 000 000	5 000 000	3 000 000
DIFO-Uncond	Batch Size	64	64	64	64	64	64
	Disc. Learning Rate	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	Disc. Buffer Size	1 000 000	1 000 000	1 000 000	1 000 000	1 000 000	1 000 000
	BCE Weight λ_{BCE}	0	0	0	0	0	0
	MSE Weight λ_{MSE}	1	1	1	1	1	1
	λ_σ	10	10	10	10	10	10
	U-Net Units	(256, 256, 256)	(256, 256, 256)	(256, 256, 256)	(256, 256, 256)	(256, 256, 256)	(256, 256, 256)
	Embedding Dimension	128	128	128	128	128	128
	Denoising Sample Range	(250, 750)	(250, 750)	(250, 750)	(250, 750)	(250, 750)	(250, 750)
	Denoising Timestep	1000	1000	1000	1000	1000	1000
	Logit Scale	1	1	1	1	1	1
	# Environment Steps	1 000 000	3 000 000	1 500 000	10 000 000	5 000 000	3 000 000
DIFO-NA	Batch Size	64	64	64	64	64	64
	Disc. Learning Rate	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	Disc. Buffer Size	1 000 000	1 000 000	1 000 000	1 000 000	1 000 000	1 000 000
	BCE Weight λ_{BCE}	0.01	0.01	0.01	0.01	0.01	0.01
	MSE Weight λ_{MSE}	1	1	1	1	1	1
	λ_σ	10	10	10	10	10	10
	U-Net Units	(512, 512, 512, 512)	(512, 512, 512, 512)	(512, 512, 512, 512)	(512, 512, 512, 512)	(512, 512, 512, 512)	(256, 256, 256)
	Embedding Dimension	128	256	256	256	256	128
	Denoising Sample Range	(250, 750)	(250, 750)	(250, 750)	(250, 750)	(250, 750)	(250, 750)
	Denoising Timestep	1000	1000	1000	1000	1000	1000
	Logit Scale	10	10	10	10	10	10
	# Environment Steps	1 000 000	3 000 000	1 500 000	10 000 000	5 000 000	3 000 000

G.1 Model architectures

Vector-based observation space. For tasks with 1D vectorized state space, we implement a U-Net with MLP layers as our backbone of the diffusion model. The conditions are applied on every layer.

Image-based observation space. For tasks with image observations, we implemented a 2D U-Net as the backbone of the diffusion model with the *diffusers* package provided by von Platen et al. [78], which originally proposed by Ronneberger et al. [57]. The model contains 3 down-sampling blocks and 3 up-sampling blocks. Each block consists of 2 convolution residual layers, with group normalization applied using 4 groups. The conditions are applied on every layer.

G.2 Hyperparameters

The hyperparameters of the policies and discriminators employed for all methods across all tasks are listed in Table 4 and 5. We use Adam as the optimizer for all the experiments.

Table 5: **SAC & PPO training parameters.**

Method	**Hyperparameter**	h
SAC	Learning Rate	0.0003
	Batch Size	256
	Discount Factor γ	0.99
PPO	Learning Rate	0.0001
	Batch Size	128
	Discount Factor γ	0.99
	Clip	0.001
	GAE λ	0.95
	Value Function Coefficient	0.5
	Entropy Coefficient	0
	Maximum gradient norm	0.6
	# Epochs	5

H Computational resources

We used the workstations listed in Table 6. Our method takes approximately 3 hours for each task. The shortest task is POINTMAZE which takes around 1 hour. The longest task is CARRACING which takes around 6 hours. As for reproducing all results including the baselines, it takes about 2000 GPU hours to run in series.

Table 6: **Computational resources.**

Workstation	CPU	GPU	RAM
Workstation 1	Intel Xeon w7-2475X	NVIDIA GeForce RTX 4090 x 2	125 GiB
Workstation 2	Intel Xeon w5-2455X	NVIDIA RTX A6000 x 2	125 GiB
Workstation 3	Intel Xeon W-2255	NVIDIA GeForce RTX 4070 Ti x 2	125 GiB
Workstation 4	Intel Xeon W-2255	NVIDIA GeForce RTX 4070 Ti x 2	125 GiB

I Limitations

This work presents a novel learning from observation framework, DIFO, integrating a diffusion model into the AIL framework. Despite that DIFO achieves encouraging results in various domains, there are still some limitations. Firstly, our method takes state-only transitions (s, s') as the reward function, while the underlying optimal reward function could be in the form $r(s, a, s')$, where dynamics is involved. This may lead to sub-optimal performance in tasks with delayed effects on actions. Secondly, our method assumes the state spaces of the agent and expert are identical as they share the same model, which limits cross-embodiment applications to some extent. Thirdly, our method highly relies on expert demonstrations, the presence of sub-optimal demonstrations may adversely impact the performance. Finally, due to the learning nature focused on discrimination, DIFO may not incorporate well with environmental rewards even if they are accessible.

J Broader impact

Experimental results demonstrate that DIFO exhibits data efficiency, generalizability, and resistance to noisy environments, thereby enhancing its suitability for real-world applications. Learning from observation enables its deployment in scenarios where action data is costly or inaccessible.

However, our method inherits the nature of Adversarial Imitation Learning. One significant concern is the potential negative impact on safety when deployed in real-world settings, as the exploration process may lead to unsafe actions. Additionally, imitation learning may capture and reinforce bias present in expert demonstrations, causing trapping in sub-optimal behaviors. These issues highlight the necessity of further research focused on reducing these negative impacts.

NeurIPS paper checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our abstract and introduction reflect our paper's contributions and scope. We clearly claim that our novel adversarial imitation learning method, DIFO, integrates a diffusion model into the learning from observation framework. We acknowledge the limitations of existing methods and how DIFO addresses these challenges, aligning well with our paper's contributions and findings. Additionally, we demonstrate that DIFO has better data efficiency and performance in various continuous control tasks.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have discussed the limitations of our method in Section I.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: This paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide detailed implementation and hyperparameter information in Appendix G. Additionally, we fix seeds in all experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We plan to release code, expert datasets, and models recently.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We have reported the experimental setting and details in Section B and Section G.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We have reported the error bars in every figure and discussed the statistical significance of the experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provided the computing resources and computing time in Section H.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and confirm that this paper conforms to the Code of Ethics in every respect.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We have provided the broader impacts in Section J

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We gave credits and cited the diffusers package and existing imitation implementations we built upon in Section G.1.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: This paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.