Diffusion Model-Augmented Behavioral Cloning

Hsiang-Chun Wang [*1] **Shang-Fu Chen** [*1] **Ming-Hao Hsu** [1] **Chun-Mao Lai** [1] **Shao-Hua Sun** [1]

Abstract

Imitation learning addresses the challenge of learning by observing an expert's demonstrations without access to reward signals from environments. Most existing imitation learning methods that do not require interacting with environments either model the expert distribution as the conditional probability $p(a|s)$ (*e.g.*, behavioral cloning, BC) or the joint probability $p(s, a)$ (*e.g.*, implicit behavioral cloning). Despite its simplicity, modeling the conditional probability with BC usually struggles with generalization. While modeling the joint probability can lead to improved generalization performance, the inference procedure can be time-consuming and it often suffers from manifold overfitting. This work proposes an imitation learning framework that benefits from modeling both the conditional and joint probability of the expert distribution. Our proposed diffusion model-augmented behavioral cloning (DBC) employs a diffusion model trained to model expert behaviors and learns a policy to optimize both the BC loss (conditional) and our proposed diffusion model loss (joint). DBC outperforms baselines in various continuous control tasks in navigation, robot arm manipulation, dexterous manipulation, and locomotion. We design additional experiments to verify the limitations of modeling either the conditional probability or the joint probability of the expert distribution as well as compare different generative models.

1 Introduction

Recently, the success of deep reinforcement learning (DRL) (Mnih et al., 2015; Lillicrap et al., 2016; Arulkumaran et al., 2017) has inspired the research community to develop DRL frameworks to control robots, aiming to automate the process of designing sensing, planning, and control algorithms by letting the robot learn in an end-to-end fashion. Yet, acquiring complex skills through trial and error can still lead to undesired behaviors even with sophisticated reward design (Christiano et al., 2017; Leike et al., 2018; Lee et al., 2019). Moreover, the exploring process could damage expensive robotic platforms or even be dangerous to humans (Garcıa & Fernández, 2015; Levine et al., 2020).

To overcome this issue, imitation learning (*i.e.*, learning from demonstration) (Schaal, 1997; Osa et al., 2018) has received growing attention, whose aim is to learn a policy from expert demonstrations, which are often more accessible than appropriate reward functions for reinforcement learning. Among various imitation learning directions, adversarial imitation learning (Ho & Ermon, 2016; Zolna et al., 2021; Kostrikov et al., 2019) and inverse reinforcement learning (Ng & Russell, 2000; Abbeel & Ng, 2004) have achieved encouraging results in a variety of domains. Yet, these methods require interacting with environments, which can still be expensive or unsafe.

On the other hand, behavioral cloning (BC) (Pomerleau, 1989; Bain & Sammut, 1995) does not require interacting with environments. BC formulates imitation learning as a supervised learning problem — given an expert demonstration dataset, an agent policy takes states sampled from the dataset as input and learns to replicate the corresponding expert actions. One can view a BC policy as a discriminative model $p(a|s)$ that models the *conditional probability* of an action a given a state s. Due to its simplicity and training stability, BC has been widely adopted for various applications.

However, BC struggles at generalizing to states unobserved during training (Nguyen et al., 2023). To address this issue, implicit behavioral cloning (IBC) (Florence et al., 2022) aims to model the *joint probability* of the expert state-action pairs $p(s, a)$ with energy-based models. IBC demonstrates superior performance when generalization is required. Yet, imitation learning methods in a similar vein (Ganapathi et al., 2022) that model the *joint probability* of state-action pairs $p(s, a)$ instead of directly predicting actions $p(a|s)$ require time-consuming actions sampling and optimization

[*] Equal contribution [1] National Taiwan University, Taipei, Taiwan. Correspondence to: Shao-Hua Sun <shaohuas@ntu.edu.tw>.

New Frontiers in Learning, Control, and Dynamical Systems Workshop at International Conference on Machine Learning, Honolulu, Hawaii, USA. 2023. Copyright 2023 by the author(s).

Project page: `https://nturobotlearninglab.github.io/dbc` to retrieve a desired action $\arg\max_{a \in \mathcal{A}} p(s, a)$ during inference despite the choice of models.

This work proposes an imitation learning framework that combines both the efficiency of modeling the *conditional probability* and the generalization ability of modeling the *joint probability*. Specifically, we propose to model the expert state-action pairs using a state-of-the-art generative model, a diffusion model, which learns to estimate how likely a state-action pair is sampled from the expert dataset. Then, we train a policy to optimize both the BC objective and the estimate produced by the learned diffusion model. Therefore, our proposed framework not only can efficiently predict actions given states via capturing the *conditional probability* $p(a|s)$ but also enjoys the generalization ability induced by modeling the *joint probability* $p(s, a)$ and utilizing it to guide policy learning.

We evaluate our proposed framework and baselines in various continuous control domains, including navigation, robot arm manipulation, and locomotion. The experimental results show that the proposed framework outperforms all the baselines or achieves competitive performance on all tasks. Extensive ablation studies compare our proposed method to its variants, justifying our design choices, such as different generative models, and investigating the effect of hyperparameters.

2 Related Work

Imitation learning addresses the challenge of learning by observing expert demonstrations without access to reward signals from environments. It has various applications such as robotics (Schaal, 1997), autonomous driving (Ly & Akhloufi, 2020), and game AI (Harmer et al., 2018).

Behavioral Cloning (BC). BC (Pomerleau, 1989; Torabi et al., 2018) formulate imitating an expert as a supervised learning problem. Due to its simplicity and effectiveness, it has been widely adopted in various domains. Yet, it often struggles at generalizing to states unobserved from the expert demonstrations (Ross et al., 2011; Florence et al., 2022). In this work, we augment BC by employing a diffusion model that learns to capture the joint probability of expert state-action pairs.

Adversarial Imitation Learning (AIL). AIL methods aim to match the state-action distributions of an agent and an expert via adversarial training. Generative adversarial imitation learning (GAIL) (Ho & Ermon, 2016) and its extensions (Torabi et al., 2019; Kostrikov et al., 2019; Zolna et al., 2021) resemble the idea of generative adversarial networks (Goodfellow et al., 2014), which trains a generator policy to imitate expert behaviors and a discriminator to distinguish between the expert and the learner's state-action pair distributions. While modeling state-action distributions often leads to satisfactory performance, adversarial learning can be unstable and inefficient (Chen et al., 2020). Moreover, AIL methods require online interaction with environments, which can be costly or even dangerous. In contrast, our work does not require interacting with environments.

Inverse Reinforcement Learning (IRL). IRL methods (Ng & Russell, 2000; Abbeel & Ng, 2004; Fu et al., 2018; Lee et al., 2021) are designed to infer the reward function that underlies the expert demonstrations and then learn a policy using the inferred reward function. This allows for learning tasks whose reward functions are difficult to specify manually. However, due to its double-loop learning procedure, IRL methods are typically computationally expensive and time-consuming. Additionally, obtaining accurate estimates of the expert's reward function can be difficult, especially when the expert's behavior is non-deterministic or when the expert's demonstrations are sub-optimal.

Diffusion Policies. Recently, (Pearce et al., 2023; Chi et al., 2023; Reuss et al., 2023) propose to represent and learn an imitation learning policy using a conditional diffusion model, which produces a predicted action conditioning on a state and a sampled noise vector. These methods achieve encouraging results in modeling stochastic and multimodal behaviors from human experts or play data. In contrast, instead of representing a policy using a diffusion model, our work employs a diffusion model trained on expert demonstrations to guide a policy as a learning objective.

3 Preliminaries

3.1 Imitation Learning

Without loss of generality, the reinforcement learning problem can be formulated as a Markov decision process (MDP), which can be represented by a tuple $M = (S, A, R, P, \rho, \gamma)$ with states S, actions A, reward function $R(S, A) \in (0, 1)$, transition distribution $P(s^{'}|s, a) : S \times A \times S \to [0, 1]$, initial state distribution ρ, and discounted factor γ. Based on the rewards received while interacting with the environment, the goal is to learn a policy $\pi(\cdot|s)$ to maximize the expectation of the cumulative discounted return (*i.e.*, value function): $V(\pi) = \mathbb{E}[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)|s_0 \sim \rho(\cdot), a_t \sim \pi(\cdot|s_t), s_{t+1} \sim P(s_{t+1}|s_t, a_t)]$, where T denotes the episode length. Instead of interacting with the environment and receiving rewards, imitation learning aims to learn an agent policy from an expert demonstration dataset, containing M trajectories, $D = \{\tau_1, ..., \tau_M\}$, where τ_i represents a sequence of n_i state-action pairs $\{s^i_1, a^i_1, ..., s^i_{n_i}, a^i_{n_i}\}$.

3.2 Behavioral Cloning: Modeling Conditional Probability $p(a|s)$

To learn a policy π, behavioral cloning (BC) directly estimates the expert policy π^E with maximum likelihood estimation (MLE). Given a state-action pair (s, a) sampled from the dataset D, BC optimizes $\underset{\theta}{max} \sum_{(s,a)\in D} \log(\pi_\theta(a|s))$, where θ denotes the parameters of the policy π. One can view a BC policy as a discriminative model $p(a|s)$, capturing the *conditional probability* of an action a given a state s. Despite its success in various applications, BC tends to overfit and struggle at generalizing to states unseen during training (Ross et al., 2011; Codevilla et al., 2019; Wang et al., 2022).

3.3 Modeling Joint Probability $p(s, a)$

Aiming for improved generalization ability, implicit behavioral cloning (Florence et al., 2022) and methods in a similar vein (Ganapathi et al., 2022) model the *joint probability* $p(s, a)$ of expert state-action pairs. These methods demonstrate superior generalization performance in diverse domains. Yet, without directly modeling the *conditional probability* $p(a|s)$, the action sampling and optimization procedure to retrieve a desired action $\arg\max_{a\in\mathcal{A}}\ p(s, a)$ during inference is often time-consuming.

Moreover, explicit generative models such as energy-based models (Du & Mordatch, 2019; Song & Kingma, 2021), variational autoencoder (Kingma & Welling, 2014), and flow-based models (Rezende & Mohamed, 2015; Dinh et al., 2017) are known to struggle with modeling observed high-dimensional data that lies on a low-dimensional manifold (*i.e.*, manifold overfitting) (Wu et al., 2021; Loaiza-Ganem et al., 2022). As a result, these methods often perform poorly when learning from demonstrations produced by script policies or PID controllers, as discussed in Section 5.4.

We aim to develop an imitation learning framework that enjoys the advantages of modeling the *conditional probability* $p(a|s)$ and the *joint probability* $p(s, a)$. Specifically, we propose to model the *joint probability* of expert state-action pairs using an explicit generative model ϕ, which learns to produce an estimate indicating how likely a state-action pair is sampled from the expert dataset. Then, we train a policy to model the *conditional probability* $p(a|s)$ by optimizing the BC objective and the estimate produced by the learned generative model ϕ. Hence, our method can efficiently predict actions given states, generalize better to unseen states, and suffer less from manifold overfitting.

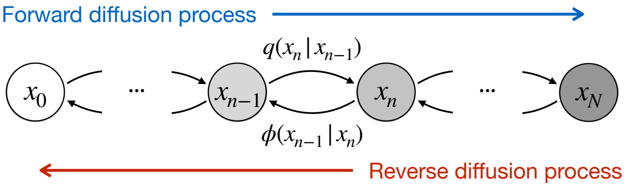


Figure 1. **Denoising Diffusion Probabilistic Model (DDPM).** Latent variables $x_1, ..., x_N$ are produced from the data point x_0 via the forward diffusion process, *i.e.*, gradually adding noises to the latent variables. The diffusion model ϕ learns to reverse the diffusion process by denoising the noisy data to reconstruct the original data point x_0.

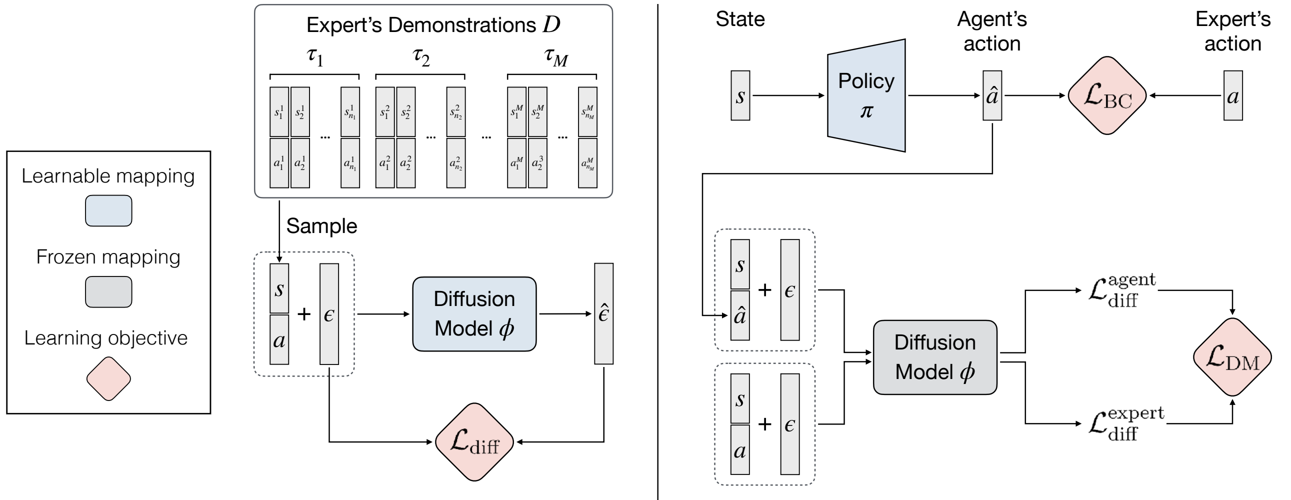
3.4 Diffusion Models

As described in the previous sections, this work aims to combine the advantages of modeling both the *conditional probability* $p(a|s)$ and the *joint probability* $p(s, a)$. To this end, we leverage diffusion models to model the *joint probability* of expert state-action pairs. The diffusion model is a recently developed class of generative models and has achieved state-of-the-art performance on various tasks (Sohl-Dickstein et al., 2015; Nichol & Dhariwal, 2021; Dhariwal & Nichol, 2021).

In this work, we utilize Denoising Diffusion Probabilistic Models (DDPMs) (J Ho, 2020) to model expert state-action pairs. Specifically, DDPM models gradually add noise to data samples (*i.e.*, concatenated state-action pairs) until they become isotropic Gaussian (*forward diffusion process*), and then learn to denoise each step and restore the original data samples (*reverse diffusion process*), as illustrated in Figure 1. In other words, DDPM learns to recognize a data distribution by learning to denoise noisy sampled data. More discussion on diffusion models can be found in the Section E.

4 Approach

Our goal is to design an imitation learning framework that enjoys both the advantages of modeling the *conditional probability* and the *joint probability* of expert behaviors. To this end, we first adopt behavioral cloning (BC) for modeling the *conditional probability* from expert state-action pairs, as described in Section 4.1. To capture the *joint probability* of expert state-action pairs, we employ a diffusion model which learns to produce an estimate indicating how likely a state-action pair is sampled from the expert state-action pair distribution, as presented in Section 4.2.1. Then, we propose to guide the policy learning by optimizing this estimate provided by a learned diffusion model, encouraging the policy to produce actions similar to expert actions, as discussed in Section 4.2.2. Finally, in Section 4.3, we introduce the framework that combines the BC loss and


Learnable mapping
Frozen mapping
Learning objective
Expert's Demonstrations D
τ_1
τ_2
τ_M
Sample
s
a
+
ϵ
Diffusion Model ϕ
$\hat{\epsilon}$
$\mathcal{L}_{\text{diff}}$
State
Agent's action
Expert's action
Policy π
$\hat{a}$
$\mathcal{L}_{\text{BC}}$
$\mathcal{L}^{\text{agent}}_{\text{diff}}$
$\mathcal{L}^{\text{expert}}_{\text{diff}}$
$\mathcal{L}_{\text{DM}}$

(a) Learning a Diffusion Model

(b) Learning a Policy with the Learned Diffusion Model

Figure 2. **Diffusion Model-Augmented Behavioral Cloning.** Our proposed method DBC augments behavioral cloning (BC) by employing a diffusion model. (a) **Learning a Diffusion Model**: the diffusion model ϕ learns to model the distribution of concatenated state-action pairs sampled from the demonstration dataset D. It learns to reverse the diffusion process (*i.e.*, denoise) by optimizing $\mathcal{L}_{\text{diff}}$ in Eq. 2. (b) **Learning a Policy with the Learned Diffusion Model**: we propose a diffusion model objective $\mathcal{L}_{\text{DM}}$ for policy learning and jointly optimize it with the BC objective $\mathcal{L}_{\text{BC}}$. Specifically, $\mathcal{L}_{\text{DM}}$ is computed based on processing a sampled state-action pair (s, a) and a state-action pair $(s, \hat{a})$ with the action $\hat{a}$ predicted by the policy π with $\mathcal{L}_{\text{diff}}$.

our proposed diffusion model loss, allowing for learning a policy that benefits from modeling both the *conditional probability* and the *joint probability* of expert behaviors. An overview of our proposed framework is illustrated in Figure 2.

4.1 Behavioral Cloning Loss

The behavioral cloning (BC) model aims to imitate expert behaviors with supervision learning. BC learns to capture the conditional probability $p(a|s)$ of expert state-action pairs. Given a sampled expert state-action pair (s, a), a policy π learns to predict an action $\hat{a} \sim \pi(s)$ by optimizing

$$\mathcal{L}_{\text{BC}} = d(a, \hat{a}), \tag{1}$$

where $d(\cdot, \cdot)$ denotes a distance measure between a pair of actions. For example, we can adapt the mean-square error (MSE) loss $||a - \hat{a}||^2$ for most continuous control tasks.

4.2 Learning a Diffusion Model and Guiding Policy Learning

Instead of directly learning the conditional probability $p(a|s)$, this section discusses how to model the joint probability $p(s, a)$ of expert behaviors with a diffusion model in Section 4.2.1 and presents how to leverage the learned diffusion model to guide policy learning in Section 4.2.2.

4.2.1 Learning a Diffusion Model

We propose to model the joint probability of expert state-action pairs with a diffusion model ϕ. Specifically, we create a joint distribution by simply concatenating a state vector s and an action vector a from a state-action pair (s, a). To model such distribution by learning a denoising diffusion probabilistic model (DDPM) (J Ho, 2020), we inject noise $\epsilon(n)$ into sampled state-action pairs, where n indicates the number of steps of the Markov procedure, which can be viewed as a variable of the level of noise. Then, we train the diffusion model ϕ to predict the injected noises by optimizing

$$\begin{aligned} \mathcal{L}_{\text{diff}}(s, a, \phi) &= ||\hat{\epsilon}(s, a, n) - \epsilon(n)||^2 \\ &= ||\phi(s, a, \epsilon(n)) - \epsilon(n)||^2, \end{aligned} \tag{2}$$

where $\hat{\epsilon}$ is the noise predicted by the diffusion model ϕ. Once optimized, the diffusion model can *recognize* the expert distribution by perfectly predicting the noise injected into state-action pairs sampled from the expert distribution. On the other hand, predicting the noise injected into state-action pairs sampled from any other distribution should yield a higher loss value. Therefore, we propose to view $\mathcal{L}_{\text{diff}}(s, a, \phi)$ as an estimate of how well the state-action pair (s, a) fits the state-action distribution that ϕ learns from.

4.2.2 Learning a Policy with Diffusion Model Loss

A diffusion model ϕ trained on the expert distribution can produce an estimate $\mathcal{L}_{\text{diff}}(s, a, \phi)$ indicating how well a state-action pair (s, a) fits the expert distribution. We propose to leverage this signal to guide a policy to imitate the expert. Specifically, given a state-action (s, a) sampled from D, the π predicts an action given the state $\hat{a} \sim \pi(s)$ by optimizing

$$\mathcal{L}^{\text{agent}}_{\text{diff}} = \mathcal{L}_{\text{diff}}(s, \hat{a}, \phi) = ||\hat{\epsilon}(s, \hat{a}, n) - \epsilon||^2. \tag{3}$$

Intuitively, the policy learns to predict actions that are indistinguishable from the expert actions for the diffusion model conditioning on the same set of states.

We hypothesize that learning a policy to optimize Eq. 3 can be unstable, especially for state-action pairs that are not well-modeled by the diffusion model, which yield a high value of $\mathcal{L}_{\text{diff}}$ even with expert state-action pairs. Therefore, we propose to normalize the agent diffusion loss $\mathcal{L}^{\text{agent}}_{\text{diff}}$ with an expert diffusion loss $\mathcal{L}^{\text{expert}}_{\text{diff}}$, which can be computed with expert state-action pairs (s, a) as follows:

$$\mathcal{L}^{\text{expert}}_{\text{diff}} = \mathcal{L}_{\text{diff}}(s, a, \phi) = ||\hat{\epsilon}(s, a, n) - \epsilon||^2. \tag{4}$$

We propose to optimize the diffusion model loss $\mathcal{L}_{\text{DM}}$ based on calculating the difference between the above agent and expert diffusion losses:

$$\mathcal{L}_{\text{DM}} = max(\mathcal{L}^{\text{agent}}_{\text{diff}} - \mathcal{L}^{\text{expert}}_{\text{diff}}, 0). \tag{5}$$

4.3 Combining the Two Objectives

Our goal is to learn a policy that benefits from both modeling the conditional probability and the joint probability of expert behaviors. To this end, we propose to augment a BC policy that optimizes the BC loss L_{BC} in Eq. 1 by jointing optimizing the proposed diffusion model loss L_{DM} in Eq. 5, which encourages the policy to predict actions that fit the expert joint probability captured by a diffusion model. To learn from both the BC loss and the diffusion model loss, we train the policy to optimize

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{BC}} + \lambda \mathcal{L}_{\text{DM}}, \tag{6}$$

where λ is a coefficient that determines the importance of the diffusion model loss relative to the BC loss. We analyze the effect of the coefficient in Section 5.7.1.

5 Experiments

We design experiments in various continuous control domains, including navigation, robot arm manipulation, dexterous manipulation, and locomotion, to compare our proposed framework (DBC) to its variants and baselines.

5.1 Experimental Setup

This section describes the environments, tasks, and expert demonstrations used for learning and evaluation. More details can be found in Section A.

Navigation. To evaluate our method on a navigation task, we choose MAZE, a maze environment proposed in (Fu et al., 2020) (maze2d-medium-v2), as illustrated in Figure 3a. This task features a point-mass agent in a 2D maze learning to navigate from its start location to a goal location by iteratively predicting its x and y acceleration. The agent's beginning and final locations are chosen randomly. We collect 100 demonstrations with 18,525 transitions using a controller.

Robot Arm Manipulation. We evaluate our method in a robot arm manipulation domain with two 7-DoF Fetch tasks: FETCHPICK and FETCHPUSH, as illustrated in Figure 3c and Figure 3b. FETCHPICK requires picking up an object from the table and lifting it to a target location; FETCHPUSH requires the arm to push an object to a target location. We use the demonstrations provided in Lee et al. (2021) for these tasks. Each dataset contains 10k transitions (303 trajectories for FETCHPICK and 185 trajectories for FETCHPUSH).

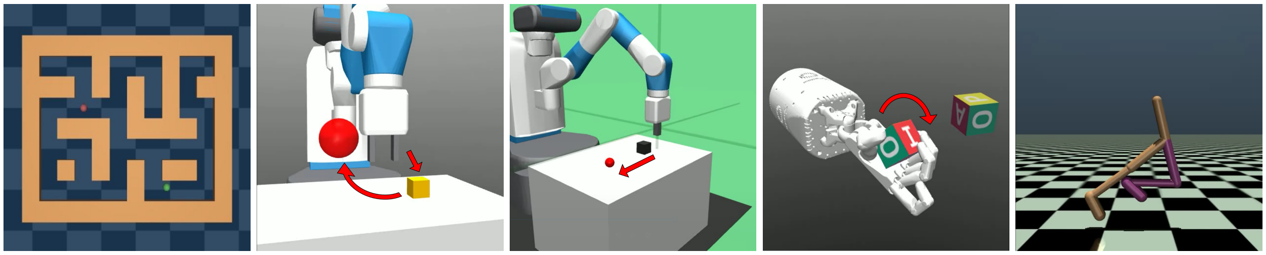
Dexterous Manipulation. In HANDROTATE, we further evaluate our method on a challenging environment proposed in Plappert et al. (2018), where a 24-DoF Shadow Dexterous Hand learns to in-hand rotate a block to a target orientation, as illustrated in Figure 3d. This environment has a high-dimensional state space (68D) and action space (20D). We collected 10k transitions (515 trajectories) from a SAC (Haarnoja et al., 2018) expert policy trained for 10M environment steps.

Locomotion. For locomotion, we leverage the WALKER environment (Brockman et al., 2016), which requires a bipedal agent to walk as fast as possible while maintaining its balance, as illustrated in Figure 3e. We use the demonstrations provided by Kostrikov (2018), which contains 5 trajectories with 5k state-action pairs.

5.2 Baselines

We compare our method DBC with the following baselines.

- **BC** learns to imitate an expert by modeling the conditional probability $p(a|s)$ of the expert behaviors via optimizing the BC loss $\mathcal{L}_{\text{BC}}$ in Eq. 1.
- **Implicit BC (IBC)** (Florence et al., 2022) models expert state-action pairs with an energy-based model. For inference, we implement the derivative-free optimization algorithm proposed in IBC, which samples actions iteratively to select the desired action with the minimum predicted energy. This baseline serves a representative



(a) Maze (b) FetchPick (c) FetchPush (d) HandRotate (e) Walker

Figure 3. **Environments & Tasks. (a) Maze**: A point-mass agent (green) in a 2D maze learns to navigate from its start location to a goal location (red). **(b)-(c) FetchPick and FetchPush**: The robot arm manipulation tasks employ a 7-DoF Fetch robotics arm. FetchPick requires picking up an object (yellow cube) from the table and moving it to a target location (red); FetchPush requires the arm to push an object (black cube) to a target location (red). **(d) HandRotate**: This dexterous manipulation task requires a Shadow Dexterous Hand to in-hand rotate a block to a target orientation. **(e) Walker**: This locomotion task requires learning a bipedal walker policy to walk as fast as possible while maintaining its balance.

Table 1. **Experimental Result.** We report the mean and the standard deviation of success rate (Maze, FetchPick, FetchPush, HandRotate) and return (Walker), evaluated over three random seeds. Our proposed method (DBC) outperforms the baselines on Maze, FetchPick, FetchPush, and HandRotate, and performs competitively against the best-performing baseline on Walker.

Method	Maze	FetchPick	FetchPush	HandRotate	Walker
BC	79.35% ± 5.05%	69.15% ± 5.00%	66.02% ± 6.88%	55.48% ± 3.97%	**7066.61** ± 22.79
Implicit BC	81.43% ± 4.88%	72.27% ± 6.71%	77.70% ± 4.42%	14.52% ± 3.04%	685.92 ± 150.26
Diffusion Policy	73.34% ± 5.30%	74.37% ± 3.80%	86.93% ± 3.26%	58.59% ± 2.85%	6429.87 ± 356.70
DBC (Ours)	**86.99%** ± 2.84%	**88.71%** ± 6.46%	**94.92%** ± 3.09%	**60.34%** ± 4.60%	7057.42 ± 36.19

of the methods that solely model the joint probability $p(s, a)$ of the expert behaviors.

- **Diffusion policy** refers to the methods that learn a conditional diffusion model as a policy (Chi et al., 2023; Reuss et al., 2023). Specifically, we implement this baseline based on Pearce et al. (2023). We include this baseline to analyze the effectiveness of using diffusion models as a policy or as a learning objective (ours).

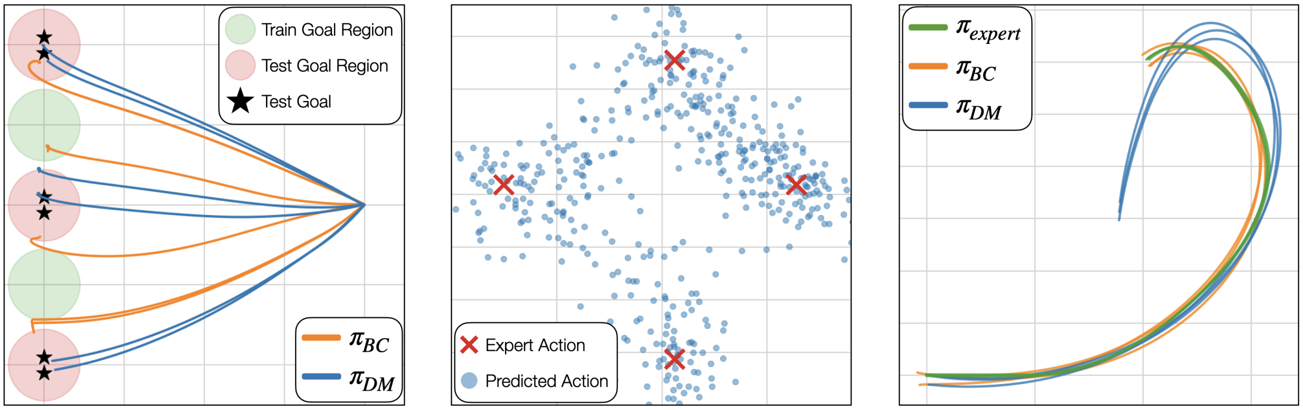
5.3 Experimental Results

We report the experimental results in terms of success rate (Maze, FetchPick, FetchPush, HandRotate), and return (Walker) in Table 1. The details of model architecture can be found in Section B. Training and evaluation details can be found in Section C. Additional analysis and experimental results can be found in Section 5.5 and Section D.

Overall Task Performance. Our proposed method DBC achieves the highest success rates, outperforming our baselines in all the goal-directed tasks (Maze, FetchPick, FetchPush, and HandRotate) and perform competitively in Walker compared to the best-performing baseline (BC). We hypothesize the improvement in the goal-directed tasks can be mostly attributed to the better generalization ability since starting positions and the goals are randomized during evaluation and therefore requires the policy to deal with unseen situation. To verify this hypothesis, we further evaluate the baselines and our method in FetchPick and FetchPush with different levels of randomization in Section 5.5.

Locomotion. Unlike the goal-directed tasks, we do not observe significant improvement but competitive results from DBC compared to the best-performing baseline (BC). We hypothesize that this is because locomotion tasks such as Walker, with sufficient expert demonstrations and little randomness, do not require generalization during inference. The agent can simply follow the closed-loop progress of the expert demonstrations, resulting in both BC (7066.61) and DBC (7057.42) performing similarly to the expert with an average return of 7063.72. On the other hand, we hypothesize that Diffusion Policy performs slightly worse due to its design for modeling multimodal behaviors, which is contradictory to learning from this single-mode simulated locomotion task.

Action Space Dimension. While Implicit BC models the joint distribution and generalizes better, it requires time-consuming actions sampling and optimization during inference. Moreover, such procedure may not scale well to high-dimensional action spaces. Our Implicit BC baseline with a derivative-free optimizer struggles in HandRotate



(a) (b) (c)

Figure 4. **Comparing Modeling Conditional Probability and Joint Probability. (a) Generalization.** We collect expert trajectories from a PPO policy learning to navigate to goals sampled from the green regions. Then, we learn a policy π_{BC} to optimize $\mathcal{L}_{\text{BC}}$, and another policy π_{DM} to optimize $\mathcal{L}_{\text{DM}}$ with a diffusion model trained on the expert distribution. We evaluate the two policies by sampling goals from the red regions, which requires the ability to generalize. π_{BC} (orange) struggles at generalizing to unseen goals, whereas π_{DM} (blue) can generalize (*i.e.*, extrapolate) to some extent. **(b)-(c) Manifold overfitting.** We collect the green spiral trajectories from a script policy, whose actions are visualized as red crosses. We then train and evaluate π_{BC} and π_{DM}. The trajectories of π_{BC} (orange) can closely follow the expert trajectories (green), while the trajectories of π_{DM} (blue) drastically deviates from expert's. This is because the diffusion model struggles at modeling such expert action distribution with a lower intrinsic dimension, which can be observed from poorly predicted actions (blue dots) produced by the diffusion model.

and WALKER environments, whose action dimensions are 20 and 6, respectively. This is consistent with Florence et al. (2022), which reports that the optimizer failed to solve tasks with an action dimension larger than 5. In contrast, our proposed DBC can handle high-dimensional action spaces.

Inference Efficiency. To evaluate the inference efficiency, we measure and report the number of evaluation episodes per second (↑) for Implicit BC (9.92), Diffusion Policy (1.38), and DBC (**30.79**) on an NVIDIA RTX 3080 Ti GPU in MAZE. As a results of modeling the conditional probability $p(a|s)$, DBC and BC can directly map states to actions during inference. In contrast, Implicit BC samples and optimizes actions, while Diffusion Policy iteratively denoises sampled noises, which are both time-consuming. This verifies the efficiency of modeling the conditional probability.

5.4 Comparing Modeling Conditional Probability and Joint Probability

This section aims to empirically identify the limitations of modeling *either* the conditional *or* the joint probability in an open maze environment implemented with (Fu et al., 2020).

Generalization. We aim to investigate if learning from the BC loss alone struggles at generalization (*conditional*) and examine if guiding the policy using the diffusion model loss yields improved generalization ability (*joint*). We collect trajectories of a PPO policy learning to navigate from $(5, 3)$ to goals sampled around $(1, 2)$ and $(1, 4)$ (green), as shown in Figure 4a. Given these expert trajectories, we learn a policy π_{BC} to optimize Eq. 1 and another policy π_{DM} to optimize Eq. 5. Then, we evaluate the two policies by sampling goals around $(1, 1)$, $(1, 3)$, and $(1, 5)$ (red), which requires the ability to generalize. Visualized trajectories of the two policies in Figure 4a show that π_{BC} (orange) fails to generalize to unseen goals, whereas π_{DM} (blue) can generalize (*i.e.*, extrapolate) to some extent. This verifies our motivation to augment BC with the diffusion model loss.

Manifold overfitting. We aim to examine if modeling the joint probability is difficult when observed high-dimensional data lies on a low-dimensional manifold (*i.e.*, manifold overfitting). We collect trajectories from a script policy that executes actions $(0.5, 0)$, $(0, 0.5)$, $(-0.7, 0)$, and $(0, -0.7)$ (red crosses in Figure 4b), each for 40 consecutive time steps, resulting the green spiral trajectories visualized in Figure 4c.

Given these expert demonstrations, we learn a policy π_{BC} to optimize Eq. 1, and another policy π_{DM} to optimize Eq. 5 with a diffusion model trained on the expert distribution. Figure 4b shows that the diffusion model struggles at modeling such expert action distribution with a lower intrinsic dimension. As a result, Figure 4c show that the trajectories of π_{DM} (blue) drastically deviates from the expert trajectories (green) as the diffusion model cannot provide effective loss. On the other hand, the trajectories of π_{BC} (orange) is able to closely follow expert's. This verifies our motivation

Table 2. **FetchPick Generalization Experimental Result.** We report the performance of our proposed framework DBC and the baselines regarding the mean and the standard deviation of the success rate with different levels of noise injected into the initial state and goal locations in FetchPick, evaluated over three random seeds.

Method	**Noise Level**				
	1	1.25	1.5	1.75	2
BC	86.78% ± 4.68%	69.15% ± 5.00%	54.42% ± 3.89%	43.49% ± 4.68%	36.64% ± 3.85%
Implicit BC	89.40% ± 4.85%	72.27% ± 6.71%	46.32% ± 5.49%	34.60% ± 4.78%	25.84% ± 4.16%
Diffusion Policy	76.04% ± 3.12%	74.37% ± 3.80%	69.22% ± 5.23%	56.95% ± 4.63%	53.93% ± 4.49%
DBC (Ours)	**97.59%** ± 1.53%	**88.71%** ± 6.46%	**78.76%** ± 10.84%	**69.36%** ± 12.72%	**62.62%** ± 14.01%

Table 3. **FetchPush Generalization Experimental Result.** We report the performance of our proposed framework DBC and the baselines regarding the mean and the standard deviation of the success rate with different levels of noise injected into the initial state and goal locations in FetchPush, evaluated over three random seeds.

Method	**Noise Level**				
	1	1.25	1.5	1.75	2
BC	94.07% ± 4.45%	82.52% ± 5.46%	66.02% ± 6.88%	48.85% ± 8.65%	34.82% ± 7.13%
Implicit BC	85.95% ± 8.39%	83.99% ± 6.06%	77.70% ± 4.42%	70.33% ± 6.06%	56.98% ± 11.74%
Diffusion Policy	97.92% ± 1.10%	93.02% ± 2.36%	86.93% ± 3.26%	74.50% ± 3.66%	65.84% ± 3.81%
DBC (Ours)	**99.83%** ± 0.23%	**99.38%** ± 0.78%	**94.92%** ± 3.09%	**87.48%** ± 5.04%	**78.43%** ± 7.41%

to complement modeling the joint probability with modeling the conditional probability (*i.e.*, BC).

5.5 Generalization Experiments in FetchPick and FetchPush

This section further investigates the generalization capabilities of the policies learned by our proposed framework and the baselines. To this end, we evaluate the policies by injecting different noise levels to both the initial state and goal location in FetchPick and FetchPush. Specifically, we parameterize the noise by scaling the 2D sampling regions for the block and goal locations in both environments. We expect all the methods to perform worse with higher noise levels, while the performance drop of the methods with better generalization ability is less significant. In this experiment, we set the coefficient λ of DBC to 0.5 in FetchPush and 0.1 in FetchPick. The results are presented in Table 2 for FetchPick and Table 3 for FetchPush.

Overall Performance. Our proposed framework DBC consistently outperforms all the baselines with different noise levels, indicating the superiority of DBC when different levels of generalization are required.

Performance Drop with Increased Noise Level. In FetchPick, DBC experiences a performance drop of 35.8% when the noise level increase from 1 to 2. However, BC and Implicit BC demonstrate a more significant performance drop of 57.8% and 71.1%, respectively. Notably, Diffusion Policy initially performs poorly at a noise level of 1 but demonstrates its robustness with a performance drop of only 29.1% when the noise level increases to 2. On the other hand, in FetchPush, DBC experiences a performance drop of 21.4% when the noise level increase from 1 to 2, while all the baselines have a more significant performance drop: BC (63%), Implicit BC (33.7%), and Diffusion Policy (32.8%). This demonstrates that our proposed framework not only generalizes better but also exhibits greater robustness to noise compared to the baselines.

5.6 Comparing Different Generative Models

Our proposed framework employs a diffusion model (DM) to model the joint probability of expert state-action pairs and utilizes it to guide policy learning. To justify our choice, we explore using other popular generative models to replace the diffusion model in Maze. We consider energy-based models (EBMs) (Du & Mordatch, 2019; Song & Kingma, 2021), variational autoencoder (VAEs) (Kingma & Welling, 2014), and generative adversarial networks (GANs) (Goodfellow et al., 2014). Each generative model learns to model expert state-action pairs. To guide policy learning, given a predicted state-action pair $(s, \hat{a})$ we use the estimated energy of an EBM, the reconstruction error of a VAE, and the discriminator output of a GAN to optimize a policy with or without the BC loss. More details on learning generative models and utilizing them to guide policy learning can be found in Section C.4.

Table 4 compares using different generative models to model the expert distribution and guide policy learning. All the generative model-guide policies can be improved by adding the BC loss, justifying our motivation to complement modeling the joint probability with modeling the conditional

Table 4. **Comparing Different Generative Models.** We compare using different generative models to model the expert state-action pair distribution and guide policy learning in MAZE. The performance of learning a policy only from the loss provided by a generative model is reported in the "without BC" column; the "with BC" column presents the performance of optimizing a policy using both the generative model loss and the BC loss. The results show that guiding a policy with a diffusion model yields the best performance, which justifies our choice of generative models. Moreover, combining the generative model loss with the BC loss leads to improved performance of all the generative models, which verifies our motivation of modeling both conditional and joint probability.

Method	without BC	with BC
BC	N/A	79.35% ± 5.05%
EBM	49.09% ± 15.15%	80.00% ± 4.06%
VAE	48.47% ± 7.57%	82.31% ± 5.84%
GAN	50.29% ± 8.27%	71.64% ± 5.50%
DM	**53.51**% ± 4.20%	**86.99**% ± 2.84%

probability. With or without the BC loss, the diffusion model-guided policy achieves the best performance compared to other generative models, verifying our choice of the generative model.

5.7 Ablation Study

In this section, we investigate the effect of the diffusion model loss coefficient λ (Section 5.7.1) and examine the effect of the normalization term $\mathcal{L}_{\text{diff}}^{\text{expert}}$ in the diffusion model loss $\mathcal{L}_{\text{DM}}$ (Section 5.7.2).

5.7.1 Effect of the Diffusion Model Loss Coefficient λ

We examine the impact of varying the coefficient of the diffusion model loss λ in Eq. 6 in MAZE. The result presented in Table 5 shows that $\lambda = 5$ yields the best performance. A higher or lower λ leads to worse performance, demonstrating how modeling the conditional probability ($\mathcal{L}_{\text{BC}}$) and the joint probability ($\mathcal{L}_{\text{DM}}$) can complement each other.

5.7.2 Effect of the Normalization Term $\mathcal{L}_{\text{diff}}^{\text{expert}}$

We aim to investigate whether normalizing the diffusion model loss $\mathcal{L}_{\text{DM}}$ with the expert diffusion model loss $\mathcal{L}_{\text{diff}}^{\text{expert}}$ yields improved performance in MAZE. We train a variant of DBC where only $\mathcal{L}_{\text{diff}}^{\text{agent}}$ in Eq. 3 instead of $\mathcal{L}_{\text{DM}}$ in Eq. 5 is used to augment BC. This variant learning from an unnormalized diffusion model loss achieves an average success rate of 80.20%, worse than the full DBC (86.99%). This justifies the effectiveness of the proposed normalization term $\mathcal{L}_{\text{diff}}^{\text{expert}}$ in $\mathcal{L}_{\text{DM}}$.

Table 5. **Effect of the Diffusion Model Loss Coefficient λ.** We experiment with different values of the diffusion model loss coefficient λ in MAZE, each evaluated over three random seeds. A λ that is too hig or too lower leads to worse performance, demonstrating how modeling the conditional probability ($\mathcal{L}_{\text{BC}}$) and the joint probability ($\mathcal{L}_{\text{DM}}$) can complement each other.

λ	Success Rate
1	85.40% ± 4.37%
2	85.64% ± 3.69%
5	**86.99**% ± 2.84%
10	85.46% ± 4.47%
20	85.17% ± 2.61%

6 Conclusion

We propose an imitation learning framework that benefits from modeling both the conditional probability $p(a|s)$ and the joint probability $p(s, a)$ of the expert distribution. Our proposed diffusion model-augmented behavioral cloning (DBC) employs a diffusion model trained to model expert behaviors and learns a policy to optimize both the BC loss and our proposed diffusion model loss. Specifically, the BC loss captures the conditional probability $p(a|s)$ from expert state-action pairs, which directly guides the policy to replicate the expert's action. On the other hand, the diffusion model loss models the joint distribution of expert's state-action pairs $p(s, a)$, which provides an evaluation of how well the predicted action aligned with the expert distribution. DBC outperforms baselines or achieves competitive performance in various continuous control tasks in navigation, robot arm manipulation, dexterous manipulation, and locomotion. We design additional experiments to verify the limitations of modeling either the conditional probability or the joint probability of the expert distribution as well as compare different generative models. Ablation studies investigate the effect of hyperparameters and justify the effectiveness of our design choices.

Acknowledgement

This work was supported by the National Taiwan University and its Department of Electrical Engineering, Graduate Institute of Communication Engineering, and College of Electrical Engineering and Computer Science. The authors also appreciate the Yushan Fellow Program by the Ministry of Education, Taiwan.

References

Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*, 2004.

Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 2017.

Bain, M. and Sammut, C. A framework for behavioural cloning. In *Machine Intelligence 15*, 1995.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

Chen, M., Wang, Y., Liu, T., Yang, Z., Li, X., Wang, Z., and Zhao, T. On computation and generalization of generative adversarial imitation learning. In *International Conference on Learning Representations*, 2020.

Chi, C., Feng, S., Du, Y., Xu, Z., Cousineau, E., Burchfiel, B., and Song, S. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, 2017.

Codevilla, F., Santana, E., López, A. M., and Gaidon, A. Exploring the limitations of behavior cloning for autonomous driving. In *International Conference on Computer Vision*, 2019.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. *Neural Information Processing Systems*, 2021.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real NVP. In *International Conference on Learning Representations*, 2017.

Du, Y. and Mordatch, I. Implicit generation and modeling with energy based models. In *Neural Information Processing Systems*, 2019.

Florence, P., Lynch, C., Zeng, A., Ramirez, O. A., Wahid, A., Downs, L., Wong, A., Lee, J., Mordatch, I., and Tompson, J. Implicit behavioral cloning. In *Conference on Robot Learning*, 2022.

Fu, J., Luo, K., and Levine, S. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Ganapathi, A., Florence, P., Varley, J., Burns, K., Goldberg, K., and Zeng, A. Implicit kinematic policies: Unifying joint and cartesian action spaces in end-to-end robot learning. In *International Conference on Robotics and Automation*, 2022.

Garcıa, J. and Fernández, F. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 2015.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of International Conference on Machine Learning (ICML)*, 2018.

Harmer, J., Gisslén, L., del Val, J., Holst, H., Bergdahl, J., Olsson, T., Sjöö, K., and Nordin, M. Imitation learning with concurrent actions in 3d games. In *IEEE Conference on Computational Intelligence and Games*, 2018.

Ho, J. and Ermon, S. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, 2016.

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

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations*, 2015.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Kostrikov, I. Pytorch implementations of reinforcement learning algorithms. https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail, 2018.

Kostrikov, I., Agrawal, K. K., Dwibedi, D., Levine, S., and Tompson, J. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *International Conference on Learning Representations*, 2019.

Lee, Y., Sun, S.-H., Somasundaram, S., Hu, E. S., and Lim, J. J. Composing complex skills by learning transition policies. In *Proceedings of International Conference on Learning Representations*, 2019.

Lee, Y., Szot, A., Sun, S.-H., and Lim, J. J. Generalizable imitation learning from observation via inferring goal proximity. In *Neural Information Processing Systems*, 2021.

Leike, J., Krueger, D., Everitt, T., Martic, M., Maini, V., and Legg, S. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*, 2018.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

Loaiza-Ganem, G., Ross, B. L., Cresswell, J. C., and Caterini, A. L. Diagnosing and fixing manifold overfitting in deep generative models. *Transactions on Machine Learning Research*, 2022.

Luo, C. Understanding diffusion models: A unified perspective. *arXiv preprint arXiv:2208.11970*, 2022.

Ly, A. O. and Akhloufi, M. Learning to drive by imitation: An overview of deep behavior cloning methods. *IEEE Transactions on Intelligent Vehicles*, 2020.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

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

Nguyen, T., Zheng, Q., and Grover, A. Reliable conditioning of behavioral cloning for offline reinforcement learning. *arXiv preprint arXiv:2210.05158*, 2023.

Nichol, A. Q. and Dhariwal, P. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, 2021.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Osa, T., Pajarinen, J., Neumann, G., Bagnell, J. A., Abbeel, P., Peters, J., et al. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 2018.

Pearce, T., Rashid, T., Kanervisto, A., Bignell, D., Sun, M., Georgescu, R., Macua, S. V., Tan, S. Z., Momennejad, I., Hofmann, K., and Devlin, S. Imitating human behaviour with diffusion models. In *International Conference on Learning Representations*, 2023.

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

Pomerleau, D. A. Alvinn: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems*, 1989.

Popov, V., Vovk, I., Gogoryan, V., Sadekova, T., Kudinov, M., and Wei, J. Diffusion-based voice conversion with fast maximum likelihood sampling scheme. In *International Conference on Learning Representations*, 2022.

Reuss, M., Li, M., Jia, X., and Lioutikov, R. Goal-conditioned imitation learning using score-based diffusion policies. *arXiv preprint arXiv:2304.02532*, 2023.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International Conference on Machine Learning*, 2015.

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

Schaal, S. Learning from demonstration. In *Advances in Neural Information Processing Systems*, 1997.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*. PMLR, 2015.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Song, Y. and Kingma, D. P. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Torabi, F., Warnell, G., and Stone, P. Behavioral cloning from observation. In *International Joint Conference on Artificial Intelligence*, 2018.

Torabi, F., Warnell, G., and Stone, P. Generative adversarial imitation from observation. *ICML*, 2019.

Wang, L., Fernandez, C., and Stiller, C. High-level decision making for automated highway driving via behavior cloning. *IEEE Transactions on Intelligent Vehicles*, 2022.

Wu, Q., Gao, R., and Zha, H. Bridging explicit and implicit deep generative models via neural stein estimators. In *Neural Information Processing Systems*, 2021.

Xu, J., Li, Z., Du, B., Zhang, M., and Liu, J. Reluplex made more practical: Leaky relu. In *2020 IEEE Symposium on Computers and communications (ISCC)*, pp. 1–7. IEEE, 2020.

Zolna, K., Reed, S., Novikov, A., Colmenarejo, S. G., Budden, D., Cabi, S., Denil, M., de Freitas, N., and Wang, Z. Task-relevant adversarial imitation learning. In *Conference on Robot Learning*, 2021.

Appendix

A Environment & Task Details

A.1 MAZE

Description. A point-maze agent in a 2D maze learns to navigate from its start location to a goal location by iteratively predicting its x and y acceleration. The 6D states include the agent's two-dimensional current location and velocity, and the goal location. The start and the goal locations are randomized when an episode is initialized.

Evaluation. We evaluate the agents with 100 episodes and three random seeds and compare our method with the baselines regarding the average success rate and episode lengths, representing the effectiveness and efficiency of the policy learned by different methods. An episode terminates when the maximum episode length of 400 is reached.

Expert Dataset. The expert dataset consists of the 100 demonstrations with $18,525$ transitions provided by Lee et al. (2021).

A.2 FETCHPICK & FETCHPUSH

Description. FETCHPICK requires a 7-DoF robot arm to pick up an object from the table and move it to a target location; FETCHPUSH requires the robot arm to push an object to a target location. Following the environment setups of Lee et al. (2021), a 16D state representation consists of the angles of the robot joints, the robot arm poses relative to the object, and goal locations. The first three dimensions of the action indicate the desired relative position at the next time step. For FETCHPICK, the fourth dimension of action specifies the distance between the two fingers of the gripper.

Evaluation. We evaluate the agents with 100 episodes and three random seeds and compare our method with the baselines regarding the average success rate and episode lengths. An episode terminates when the agent completes the task or the maximum episode length is reached, which is set to 50 for FETCHPICK and 120 for FETCHPUSH.

Expert Dataset. The expert dataset of FETCHPICK consists of 303 trajectories ($10k$ transitions) while the expert dataset of FETCHPUSH consists of 185 trajectories ($10k$ transitions) provided by Lee et al. (2021).

A.3 HANDROTATE

Description. HANDROTATE Plappert et al. (2018) requires a 24-DoF Shadow Dexterous Hand to in-hand rotate a block to a target orientation. The 68D state representation consists of the joint angles and velocities of the hand, object poses, and the target rotation. The 20D action indicates the position control of the 20 joints, which can be controlled independently. HANDROTATE is extremely challenging due to its high dimensional state and action spaces. We adapt the experimental setup used in Plappert et al. (2018) and Lee et al. (2021), where the rotation is restricted to the z-axis and the possible initial and target z rotations are set within $[-\frac{\pi}{12}, \frac{\pi}{12}]$ and $[\frac{\pi}{3}, \frac{2\pi}{3}]$, respectively.

Evaluation. We evaluate the agents with 100 episodes and three random seeds and compare our method with the baselines regarding the average success rate and episode lengths. An episode terminates when the agent completes the goal or the maximum episode length of 50 is reached.

Expert Dataset. To collect expert demonstrations, we train a SAC Haarnoja et al. (2018) policy using dense rewards for $10M$ environment steps. The dense reward given at each time step t is $R(s_t, a_t) = d_t - d_{t+1}$, where d_t and d_{t+1} represent the angles (in radian) between current and the desired block orientations before and after taking the actions. Following the training stage, the SAC expert policy achieves a success rate of 59.48%. Subsequently, we collect 515 successful trajectories ($10k$ transitions) from this policy to form our expert dataset for HANDROTATE.

A.4 WALKER

Description. WALKER requires an agent to walk toward x-coordinate as fast as possible while maintaining its balance. The 17D state consists of angles of joints, angular velocities of joints, and velocities of the x and z-coordinate of the top. The 6D action specifies the torques to be applied on each joint of the walker avatar.

Evaluation. We evaluate each learned policy with 30 episodes and three random seeds and compare our method with the baselines regarding the average returns of episodes and episode lengths. The return of an episode is accumulated from

Table 6. **Model Architectures.** We report the architectures used for all the methods on all the tasks.

Method	Models	Component	MAZE	FETCHPICK	FETCHPUSH	HANDROTATE	WALKER
BC	Policy π	# Layers	3	2	2	3	3
		Input Dim.	6	16	16	68	17
		Hidden Dim.	256	1024	1024	1024	256
		Output Dim.	2	4	3	20	6
Implicit BC	Policy π	# Layers	2	2	2	2	2
		Input Dim.	8	20	19	88	23
		Hidden Dim.	1024	1024	1024	512	1024
		Output Dim.	1	1	1	1	1
Diffusion Policy	Policy π	# Layers	5	5	5	5	5
		Input Dim.	8	20	19	88	23
		Hidden Dim.	256	1200	1024	2100	1200
		Output Dim.	2	4	3	20	6
DBC	DM ϕ	# Layers	5	5	5	5	5
		Input Dim.	8	20	19	88	23
		Hidden Dim.	128	1024	1024	2048	1024
		Output Dim.	8	20	19	88	23
	Policy π	# Layers	3	2	2	3	3
		Input Dim.	6	16	16	68	17
		Hidden Dim.	256	1024	1024	512	256
		Output Dim.	2	4	3	20	6

all the time steps of an episode. An episode terminates when the agent is unhealthy (*i.e.*, ill conditions predefined in the environment) or the maximum episode length (1000) is reached.

Expert Dataset. The expert dataset consists of 5 trajectories with $5k$ state-action pairs provided by Kostrikov (2018).

B Model Architecture

This section describes the model architectures used for all the experiments. Section B.1 presents the model architectures of BC, Implicit BC, Diffusion Policy, and our proposed framework DBC. Section B.2 details the model architectures of the EBM, VAE, and GAN used for the experiment comparing different generative models.

B.1 Model Architecture of BC, Implicit BC, Diffusion Policy, and DBC

We compare our DBC with three baselines (BC, Implicit BC, and Diffusion Policy) on various tasks in Section 5.3. We detail the model architectures for all the methods on all the tasks in Table 6. Note that all the models, the policy of BC, the energy-based model of Implicit BC, the conditional diffusion model of Diffusion Policy, the policy and the diffusion model of DBC, are parameterized by a multilayer perceptron (MLP). We report the implementation details for each method as follows.

BC. The non-linear activation function is a hyperbolic tangent for all the BC policies. We experiment with BC policies with more parameters, which tend to severely overfit to expert datasets, resulting in worse performance.

Implicit BC. The non-linear activation function is ReLU for all energy-based models of Implicit BC. We empirically find that Implicit BC prefers shallow architectures in our tasks, so we set the number of layers to 2 for the energy-based models.

Diffusion Policy. The non-linear activation function is ReLU for all the policies of Diffusion Policy. We empirically find that Diffusion Policy performs better with a deeper architecture. Therefore, we set the number of layers to 5 for the policy. In most cases, we use a Diffusion Policy with more parameters than the total parameters of DBC consisting of the policy and the diffusion model.

DBC. The non-linear activation function is ReLU for the diffusion models and is a hyperbolic tangent for the policies. We apply batch normalization and dropout layers with a 0.2 ratio for the diffusion models on FETCHPICK and FETCHPUSH.

B.2 Model Architecture of EBM, VAE, and GAN

We compare different generative models (*i.e.*, EBM, VAE, and GAN) on MAZE in Section 5.6, and we report the model architectures used for the experiment in this section.

Energy-Based Model. An energy-based model (EBM) consists of 5 linear layers with ReLU activation. The EBM takes a concatenated state-action pair with a dimension of 8 as input; the output is a 1-dimensional vector representing the estimated energy values of the state-action pair. The size of the hidden dimensions is 128.

Variational Autoencoder. The architecture of a variational autoencoder consists of an encoder and a decoder. The inputs of the encoder are a concatenated state-action pair, and the outputs are the predicted mean and variance, which parameterize a Gaussian distribution. We apply the reparameterization trick (Kingma & Welling, 2014), sample features from the predicted Gaussian distribution, and use the decoder to produce the reconstructed state-action pair. The encoder and the decoder both consist of 5 linear layers with LeakyReLU Xu et al. (2020) activation. The size of the hidden dimensions is 128. That said, the encoder maps an 8-dimensional state-action pair to two 128-dimensional vectors (*i.e.*, mean and variance), and the decoder maps a sampled 128-dimensional vector back to an 8-dimensional reconstructed state-action pair.

Generative Adversarial Network. The architecture of the generative adversarial network consists of a generator and a discriminator. The generator is the policy model that predicts an action from a given state, whose input dimension is 6 and output dimension is 2. On the other hand, the discriminator learns to distinguish the expert state-action pairs (s, a) from the state-action pairs produced by the generator $(s, \hat{a})$. Therefore, the input dimension of the discriminator is 8, and the output is a scalar representing the probability of the state-action pair being "real." The generator and the discriminator both consist of three linear layers with ReLU activation, and the size of the hidden dimensions is 256.

C Training and Inference Details

We describe the details of training and performing inference in this section, including computation resources and hyperparameters.

C.1 Computation Resource

We conducted all the experiments on the following three workstations:

- M1: ASUS WS880T workstation with an Intel Xeon W-2255 (10C/20T, 19.25M, 4.5GHz) 48-Lane CPU, 64GB memory, an NVIDIA RTX 3080 Ti GPU, and an NVIDIA RTX 3090 Ti GPU
- M2: ASUS WS880T workstation with an Intel Xeon W-2255 (10C/20T, 19.25M, 4.5GHz) 48-Lane CPU, 64GB memory, an NVIDIA RTX 3080 Ti GPU, and an NVIDIA RTX 3090 Ti GPU
- M3: ASUS WS880T workstation with an Intel Xeon W-2255 (10C/20T, 19.25M, 4.5GHz) 48-Lane CPU, 64GB memory, and two NVIDIA RTX 3080 Ti GPUs

C.2 Hyperparamters

We report the hyperparameters used for all the methods on all the tasks in Table 7. We use the Adam optimizer (Kingma & Ba, 2015) for all the methods on all the tasks and use linear learning rate decay for all policy models.

C.3 Inference Details

This section describes how each method infers an action $\hat{a}$ given a state s.

BC & DBC. The policy models of BC and DBC can directly predict an action given a state, *i.e.*, $\hat{a} \sim \pi(s)$, and are therefore more efficient during inference as described in Section 5.3.

Implicit BC. The energy-based model (EBM) of Implicit BC learns to predict an estimated energy value for a state-action pair during training. To generate a predicted $\hat{a}$ given a state s during inference, it requires a procedure to sample and optimize actions. We follow Florence et al. (2022) and implement a derivative-free optimization algorithm to perform inference.

Table 7. **Hyperparameters.** This table reports the hyperparameters used for all the methods on all the tasks. Note that our proposed framework (DBC) consists of two learning modules, the diffusion model and the policy, and therefore their hyperparameters are reported separately.

λ	Hyperparameter	Maze	FetchPick	FetchPush	HandRotate	Walker
BC	Learning Rate	1e-4	1e-5	1e-5	5e-6	1e-4
	Batch Size	128	128	128	128	128
	# Epochs	2000	5000	5000	5000	2000
Implicit BC	Learning Rate	1e-4	5e-6	1e-4	1e-5	1e-4
	Batch Size	128	512	512	512	128
	# Epochs	10000	15000	15000	5000	10000
Diffusion Policy	Learning Rate	2e-4	1e-5	1e-5	1e-4	1e-4
	Batch Size	128	128	128	128	128
	# Epochs	20000	15000	15000	30000	10000
DBC (Ours)	Diffusion Model Learning rate	1e-3	1e-4	1e-4	3e-5	2e-4
	Diffusion Model Batch Size	128	128	128	128	1024
	Diffusion Model # Epochs	8000	10000	10000	10000	8000
	Policy Learning Rate	1e-4	1e-5	2e-5	1e-4	1e-4
	Policy Batch Size	128	128	128	128	128
	Policy # Epochs	2000	5000	5000	5000	2000
	λ	5	0.1	0.2	1	0.05

The algorithm first randomly samples N_s vectors from the action space as candidates. The EBM then produces the estimated energy value of each candidate action and applies the Softmax function on the estimated energy values to produce a N_s-dimensional probability. Then, it samples candidate actions according to the above probability and adds noise to them to generate another N_s candidates for the next iteration. The above procedure iterates N_{iter} times. Finally, the action with maximum probability in the last iteration is selected as the predicted action $\hat{a}$. In our experiments, N_s is set to 1000 and N_{iter} is set to 3.

Diffusion Policy. Diffusion Policy learns a conditional diffusion model as a policy and produces an action from sampled noise vectors conditioning on the given state during inference. We follow Pearce et al. (2023); Chi et al. (2023) and adopt Denoising Diffusion Probabilistic Models (DDPMs) J Ho (2020) for the diffusion models. Once learned, the diffusion policy π can "denoise" a noise sampled from a Gaussian distribution $\mathcal{N}(0, 1)$ given a state s and yield a predicted action $\hat{a}$ using the following equation:

$$a_{n-1} = \frac{1}{\sqrt{\alpha_n}}(a_n - \frac{1-\alpha_n}{\sqrt{1-\bar{\alpha}_n}}\pi(s, a_n, n)) + \sigma_n z, \tag{7}$$

where α_n, $\bar{\alpha}_n$, and σ_n are schedule parameters, n is the current time step of the reverse diffusion process, and $z \sim \mathcal{N}(0, 1)$ is a random vector. The above denoising process iterates N times to produce a predicted action a_0 from a sampled noise $a_N \sim \mathcal{N}(0, 1)$. The number of total diffusion steps N is 100 in our experiment, which is the same for the diffusion model in DBC.

C.4 Comparing Different Generative Models

Our proposed framework employs a diffusion model (DM) to model the joint probability of expert state-action pairs and utilizes it to guide policy learning. To justify our choice of generative models, we explore using other popular generative models to replace the diffusion model in Maze. Specifically, we consider energy-based models (EBMs) (Du & Mordatch, 2019; Song & Kingma, 2021), variational autoencoders (VAEs) (Kingma & Welling, 2014), and generative adversarial networks (GANs) (Goodfellow et al., 2014). Each generative model learns to model the joint distribution of expert state-action pairs. For fair comparisons, all the policy models learning from learned generative models consists of 3 linear layers with ReLU activation, where the hidden dimension is 256. All the policies are trained for 2000 epochs using the Adam optimizer (Kingma & Ba, 2015), and a linear learning rate decay is applied for EBMs and VAEs.

C.4.1 Energy-Based Model

Model Learning. Energy-based models (EBMs) learn to model the joint distribution of the expert state-action pairs by predicting an estimated energy value for a state-action pair (s, a). The EBM aims to assign low energy value to the real expert state-action pairs while high energy otherwise. Therefore, the predicted energy value can be used to evaluate how well a state-action pair (s, a) fits the distribution of the expert state-action pair distribution.

To train the EBM, we generate N_{neg} random actions as negative samples for each expert state-action pair as proposed in Florence et al. (2022). The objective of the EBM E_ϕ is the InfoNCE loss Oord et al. (2018):

$$\mathcal{L}_{\text{InfoNCE}} = \frac{e^{-E_\phi(s,a)}}{e^{-E_\phi(s,a)} + \Sigma_{i=1}^{N_{neg}} e^{-E_\phi(s,\tilde{a}_i)}}, \tag{8}$$

where (s, a) indicates an expert state-action pair, $\tilde{a}_i$ indicates the sampled random action, and N_{neg} is set to 64 in our experiments. The EBM learns to separate the expert state-action pairs from the negative samples by optimizing the above InfoNCE loss.

The EBM is trained for 8000 epochs with the Adam optimizer (Kingma & Ba, 2015), with a batch size of 128 and an initial learning rate of 0.0005. We apply learning rate decay by 0.99 for every 100 epoch.

Guiding Policy Learning. To guide a policy π to learn, we design an EBM loss $\mathcal{L}_{\text{EBM}} = E_\phi(s, \hat{a})$, where $\hat{a}$ indicates the predicted action produced by the policy. The above EBM loss regularizes the policy to generate actions with low energy values, which encourage the predicted state-action pair $(s, \hat{a})$ to fit the modeled expert state-action pair distribution. The policy learning from this EBM loss $\mathcal{L}_{\text{EBM}}$ achieves a success rate of 49.09% in MAZE as reported in Table 4.

We also experiment with combining this EBM loss $\mathcal{L}_{\text{EBM}}$ with the $\mathcal{L}_{\text{BC}}$ loss. The policy optimizes $\mathcal{L}_{\text{BC}} + \lambda_{\text{EBM}}\mathcal{L}_{\text{EMB}}$, where λ_{EBM} is set to 0.1. Optimizing this combined loss yields a success rate of 80.00% in MAZE as reported in Table 4.

C.4.2 Variational Autoencoder

Model Learning. Variational autoencoders (VAEs) model the joint distribution of the expert data by learning to reconstruct expert state-action pairs (s, a). Once the VAE is learned, how well a state-action pair fits the expert distribution can be reflected in the reconstruction loss.

The objective of training a VAE is as follows:

$$\mathcal{L}_{\text{vae}} = ||\hat{x} - x||^2 + D_{\text{KL}}(\mathcal{N}(\mu_x, \sigma_x)||\mathcal{N}(0, 1)), \tag{9}$$

where x is the latent variable, *i.e.*, the concatenated state-action pair $x = [s, a]$, and $\hat{x}$ is the reconstruction of x, *i.e.*, the reconstructed state-action pair. The first term is the reconstruction loss, while the second term encourages aligning the data distribution with a normal distribution $\mathcal{N}(0, 1)$, where μ_x and σ_x are the predicted mean and standard deviation given x.

The VAE is trained for $100k$ update iterations with the Adam optimizer (Kingma & Ba, 2015), with a batch size of 128 and an initial learning rate of 0.0001. We apply learning rate decay by 0.5 for every $5k$ epoch.

Guiding Policy Learning. To guide a policy π to learn, we design a VAE loss $\mathcal{L}_{\text{VAE}} = max(\mathcal{L}_{\text{vae}}^{\text{agent}} - \mathcal{L}_{\text{vae}}^{\text{expert}}, 0)$, similar to Eq. 5. This loss forces the policy to predict an action, together with the state, that can be well reconstructed with the learned VAE. The policy learning from this VAE loss $\mathcal{L}_{\text{VAE}}$ achieves a success rate of 48.47% in MAZE as reported in Table 4.

We also experiment with combining this VAE loss $\mathcal{L}_{\text{VAE}}$ with the $\mathcal{L}_{\text{BC}}$ loss. The policy optimizes $\mathcal{L}_{\text{BC}} + \lambda_{\text{VAE}}\mathcal{L}_{\text{VAE}}$, where λ_{VAE} is set to 1. Optimizing this combined loss yields a success rate of 82.31% in MAZE as reported in Table 4.

C.4.3 Generative Adversarial Network

Adversarial Model Learning & Policy Learning. Generative adversarial networks (GANs) model the joint distribution of expert data with a generator and a discriminator. The generator aims to synthesize a predicted action $\hat{a}$ given a state s. On the other hand, the discriminator aims to identify expert the state-action pair (s, a) from the predicted one $(s, \hat{a})$. Therefore, a learned discriminator can evaluate how well a state-action pair fits the expert distribution.

While it is possible to learn a GAN separately and utilize the discriminator to guide policy learning, we let the policy π be the generator directly and optimize the policy with the discriminator iteratively. We hypothesize that a learned discriminator may be too selective for a policy training from scratch, so we learn the policy π with the discriminator D to improve the policy and the discriminator simultaneously.

The objective of training the discriminator D is as follows:

$$\mathcal{L}_{\text{disc}} = BCE(D(s,a),1) + BCE(D(s,\hat{a}),0) = -log(D(s,a)) - log(1 - D(s,\hat{a})), \tag{10}$$

where $\hat{a} = \pi(s)$ is the predicted action, and BCE is the binary cross entropy loss. The binary label $(0,1)$ indicates whether or not the state-action pair sampled from the expert data. The generator and the discriminator are both updated by Adam optimizers using a 0.00005 learning rate.

To learn a policy (*i.e.*, generator), we design the following GAN loss:

$$\mathcal{L}_{\text{GAN}} = BCE(D(s,\hat{a}),1) = -log(D(s,\hat{a})). \tag{11}$$

The above GAN loss guides the policy to generate state-action pairs that fit the joint distribution of the expert data. The policy learning from this GAN loss $\mathcal{L}_{\text{GAN}}$ achieves a success rate of 50.29% in MAZE as reported in Table 4.

We also experiment with combining this GAN loss $\mathcal{L}_{\text{GAN}}$ with the $\mathcal{L}_{\text{BC}}$ loss. The policy optimizes $\mathcal{L}_{\text{BC}} + \lambda_{\text{GAN}}\mathcal{L}_{\text{GAN}}$, where λ_{GAN} is set to 0.2. Optimizing this combined loss yields a success rate of 71.64% in MAZE as reported in Table 4.

D Qualitative Results and Additional Analysis

This section provides more detailed analyses of our proposed framework and the baselines. We present the qualitative results in Section D.1. Then, we analyze the learning progress and the episode length of goal-directed tasks during inference in Section D.2 and Section D.3, respectively.

D.1 Qualitative Results

Rendered videos of the policies learned by our proposed framework and the baselines can be found at `https://sites.google.com/view/diffusion-behavioral-cloning`. A screenshot of the rendered videos on the web page is presented in Figure 5.

D.2 Learning Progress Analysis

In this section, we analyze the learning progress of all the methods on all the tasks. The training curves are presented in Figure 6. Our proposed framework (DBC) not only achieves the best converged performance but also converges the fastest, demonstrating its learning efficiency.

Since Implicit BC and Diffusion Policy take significantly longer to converge, we set a higher number of training epochs for these two methods (see Table 7), and hence their learning curves are notably longer than BC and DBC.

Note that we make sure the numbers of training epochs for Implicit BC and Diffusion Policy are not less the total number of training epochs for learning both the diffusion model and the policy in DBC, except for Implicit BC in HANDROTATE where training longer does not yield any improvement. This forecloses the possibility of the superior performance of DBC coming from learning with a higher total number of training epochs.

D.3 Episode Length Analysis of Goal-Directed Tasks

In this section, we investigate the efficiency of the learned policies regarding the number of time steps they need to fulfill a task. We compare all the methods regarding average episode lengths over 100 episodes and three random seeds in all goal-directed tasks (MAZE, FETCHPUSH, FETCHPICK, and HANDROTATE). The results are presented in Table 8 and Figure 7 .

Note that Implicit BC and Diffusion Policy take significantly longer to converge, and hence we set a higher number of training epochs for these two methods (see Table 7). As a result, their learning curves are notably longer than BC and DBC.

We observe that our proposed framework DBC results in the shortest episode lengths in MAZE, FETCHPUSH, and FETCHPICK while performing competitively against the best-performing baseline (Diffusion Policy) in HANDROTATE. This indicates that DBC learns an efficient policy that can accomplish tasks quickly.

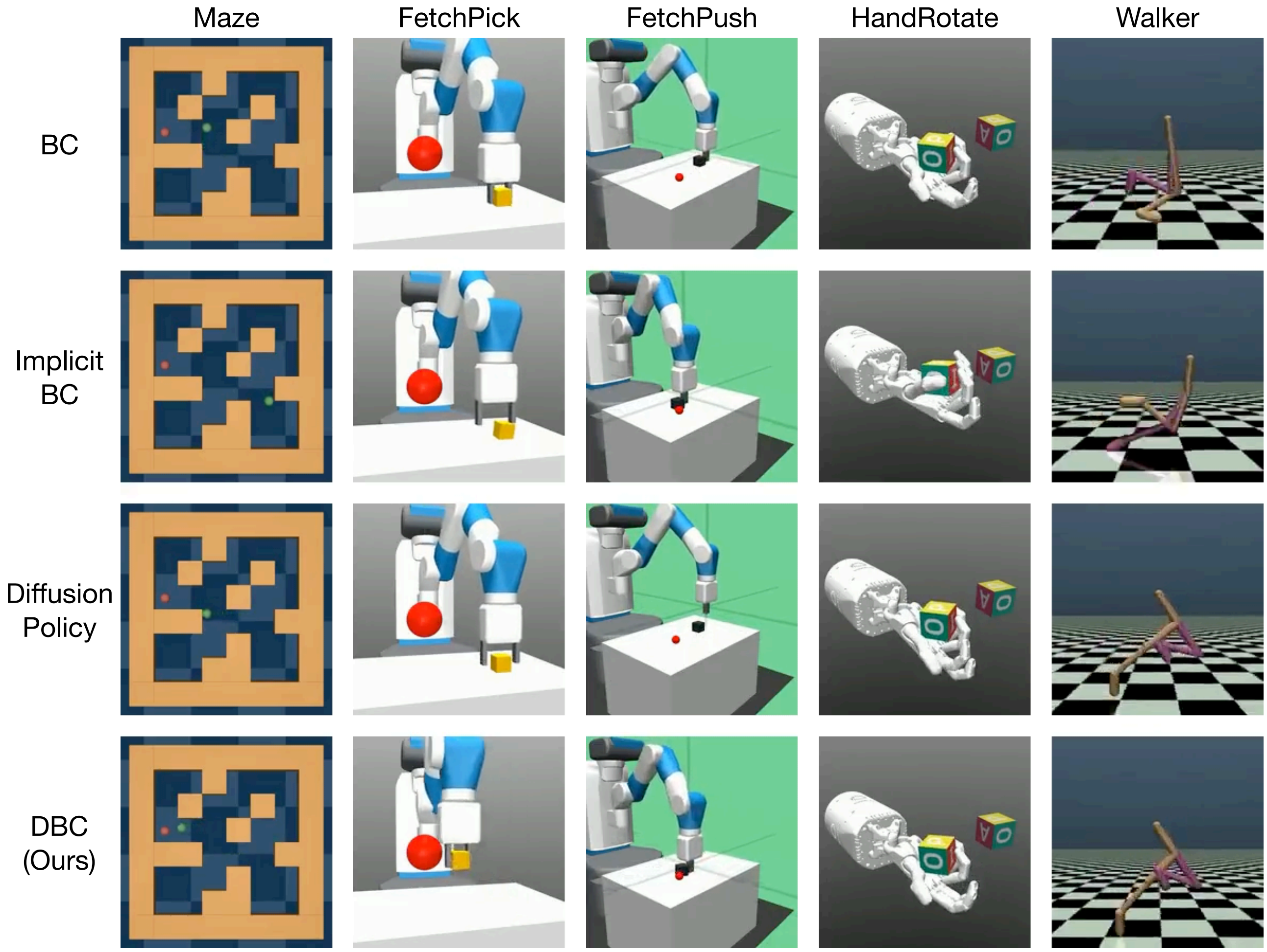


Figure 5. **Qualitative Results.** Rendered videos of the policies learned by our proposed framework and the baselines can be found at `https://sites.google.com/view/diffusion-behavioral-cloning`.

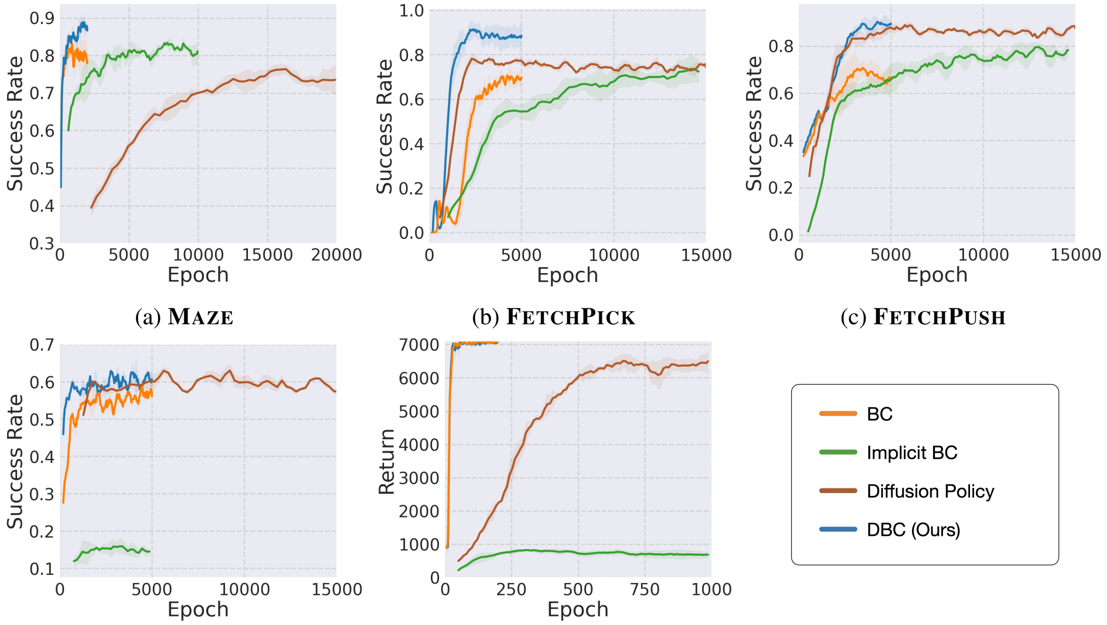
E On the Theoretical Motivation for Guiding Policy Learning with Diffusion Model

This section further elaborates on the technical motivation for leveraging diffusion models for imitation learning. Specifically, we aim to learn a diffusion model to model the joint distribution of expert state-action pairs. Then, we propose to utilize this learned diffusion model to augment a BC policy that aims to imitate expert behaviors.

We consider the distribution of expert state-action pairs as the real data distribution q_x in learning a diffusion model. Following this setup, x_0 represents an original expert state-action pair (s, a) and $q(x_n|x_{n-1})$ represents the forward diffusion process, which gradually adds Gaussian noise to the data in each timestep $n = 1, ..., N$ until x_N becomes an isotropic gaussian distribution. On the other hand, the reverse diffusion process is defined as $\phi(x_{n-1}|x_n) := \mathcal{N}(x_{n-1}; \mu_\theta(x_n, n), \Sigma_\theta(x_n, n))$, where θ denotes the learnable parameters of the diffusion model ϕ, as illustrated in Figure 1.

Our key idea is to use the proposed diffusion model loss $\mathcal{L}_{\text{DM}}$ in Eq. 5 as an estimate of how well a predicted state-action pair $(s, \hat{a})$ fits the expert state-action pair distribution, as described in Section 4.2.2. In the following derivation, we will show that by optimizing this diffusion model loss $\mathcal{L}_{\text{DM}}$, we maximize the lower bound of the agent data's probability under the derived expert distribution and hence bring the agent policy π closer to the expert policy π^E, which is the goal of imitation learning.

As depicted in Luo (2022), one can conceptualize diffusion models, including DDPM (J Ho, 2020) adopted in this work, as a hierarchical variational autoencoder (Kingma & Welling, 2014), which maximizes the likelihood $p(x)$ of observed data points x. Therefore, similar to hierarchical variational autoencoders, diffusion models can optimize the Evidence Lower Bound (ELBO) by minimizing the KL divergence $D_{KL}(q(x_{n-1}|x_n, x_0)||\phi(x_{n-1}|x_n))$. Consequently, this can be viewed



(a) **MAZE**

(b) **FETCHPICK**

(c) **FETCHPUSH**

(d) **HANDROTATE**

(e) **WALKER**

Figure 6. **Learning Progress.** We evaluate the baselines and our proposed method DBC and its variants during the learning process. Since Implicit BC (green) and Diffusion Policy (brown) take significantly longer to converge, we set a higher number of training epochs for these two methods, and hence their learning curves are notably longer than BC (orange) and DBC (blue). Our method demonstrates superior learning efficiency over the baselines.

Table 8. **Episode Length of Goal-Directed Tasks.** We report the mean and the standard deviation of the episode length (↓) on MAZE, FETCHPICK, FETCHPUSH, and HANDROTATE, evaluated over three random seeds. The experiments demonstrate that our proposed method (DBC) outperforms (*i.e.*, finish tasks with fewer time steps) the baselines on MAZE, FETCHPICK, and FETCHPUSH while performing competitively in HANDROTATE.

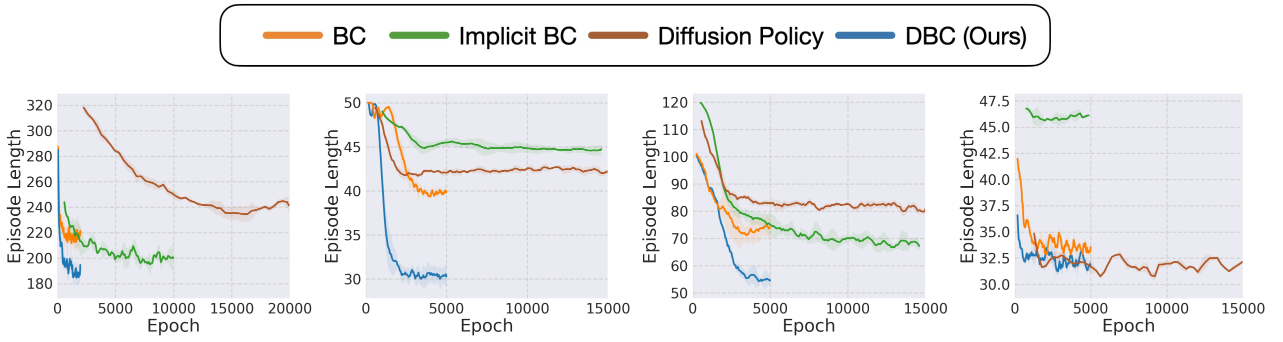
Method	MAZE	FETCHPICK	FETCHPUSH	HANDROTATE
BC	219.95 ± 13.21	39.92 ± 0.65	74.08 ± 5.55	33.79 ± 1.18
Implicit BC	199.91 ± 15.95	44.67 ± 0.65	67.75 ± 3.13	46.13 ± 0.84
Diffusion Policy	241.45 ± 12.47	42.20 ± 0.64	80.93 ± 8.88	**31.95** ± 0.82
DBC (Ours)	**193.12** ± 10.30	**30.22** ± 1.38	**54.58** ± 3.33	**31.97** ± 1.49

as minimizing the KL divergence to fit the distribution of the predicted state-action pairs $(s, \hat{a})$ to the distribution of expert state-action pairs.

According to Bayes' theorem and the properties of Markov chains, the forward diffusion process $q(x_{n-1}|x_n, x_0)$ follows:

$$q(x_{n-1}|x_n, x_0) \sim \mathcal{N}(x_{n-1}; \underbrace{\frac{\sqrt{\alpha_n}(1-\bar{\alpha}_{n-1})x_n + \sqrt{\bar{\alpha}_{n-1}}(1-\alpha_n)x_0}{1-\bar{\alpha_n}}}_{\mu_q(x_n, x_0)}, \underbrace{\frac{(1-\alpha_n)(1-\bar{\alpha}_{n-1})}{1-\bar{\alpha}_n}}_{\Sigma_q(n)}).$$

The variation term $\Sigma_q(n)$ in the above equation can be written as $\sigma_q^2(n)I$, where $\sigma_q^2(n) = \dfrac{(1-\alpha_n)(1-\bar{\alpha}_{n-1})}{1-\bar{\alpha}_n}$. Therefore,



(a) MAZE (b) FETCHPICK (c) FETCHPUSH (d) HANDROTATE

Figure 7. **Episode Length of Goal-Directed Tasks.** We evaluate the baselines and our proposed method regarding the episode length during the learning process. Since Implicit BC (green) and Diffusion Policy (brown) take significantly longer to converge, we set a higher number of training epochs for these two methods, and hence their learning curves are notably longer than BC (orange) and DBC (blue). The average episode length indicates how fast the agent reaches the goal, which can be a measurement of the efficiency of the agent. Our method DBC demonstrates superior efficiency in accomplishing tasks.

minimizing the KL divergence is equivalent to minimizing the gap between the mean values of the two distributions:

$$\begin{aligned}&\arg\min_\theta D_{KL}(q(x_{n-1}|x_n,x_0)||\phi(x_{n-1}|x_n))\\&=\arg\min_\theta D_{KL}(\mathcal{N}(x_{n-1};\mu_q,\Sigma_q(n))||\mathcal{N}(x_{n-1};\mu_\theta,\Sigma_q(n)))\\&=\arg\min_\theta \frac{1}{2\sigma_q^2(n)}[||\mu_\theta-\mu_q||_2^2],\end{aligned}$$

where μ_q represents the denoising transition mean and μ_θ represents the approximated denoising transition mean by the model.

Different implementations adopt different forms to model μ_θ. Specifically, for DDPMs adopted in this work, the true denoising transition mean $\mu_q(x_n, x_0)$ derived above can be rewritten as:

$$\mu_q(x_n,x_0)=\frac{1}{\sqrt{\alpha_n}}(x_n-\frac{1-\alpha_n}{\sqrt{1-\bar{\alpha}_n}}\epsilon_0),$$

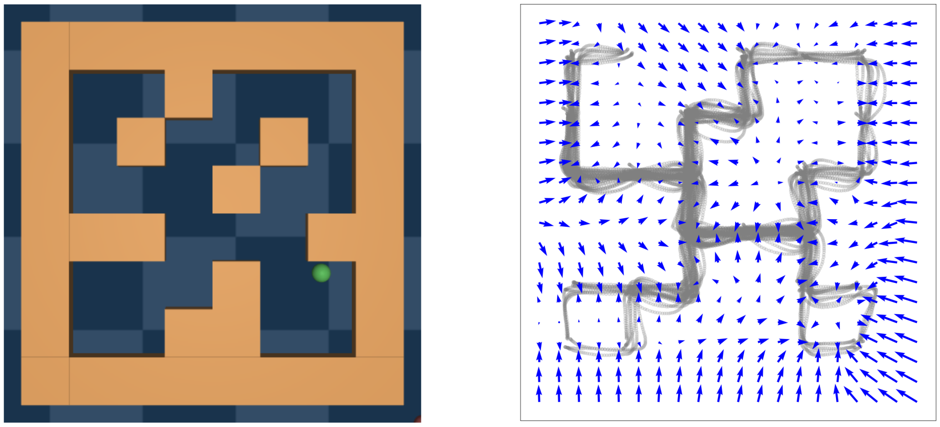
which is referenced from Eq. 11 in J Ho (2020). Hence, we can set our approximate denoising transition mean μ_θ in the same form as the true denoising transition mean:

$$\mu_\theta(x_n,n)=\frac{1}{\sqrt{\alpha_n}}(x_n-\frac{1-\alpha_n}{\sqrt{1-\bar{\alpha}_n}}\hat{\epsilon_\theta}(x_n,n)), \tag{12}$$

as illustrated in Popov et al. (2022). Song et al. (2021) further show that the entire diffusion model formulation can be revised to view continuous stochastic differential equations (SDEs) as a forward diffusion. It points out that the reverse process is also an SDE, which can be computed by estimating a score function $\nabla_x \log p_t(x)$ at each denoising time step. The idea of representing a distribution by modeling its score function is introduced in Song & Ermon (2019). The fundamental concept is to model the gradient of the log probability density function $\nabla_x \log p_t(x)$, a quantity commonly referred to as the (Stein) score function. Such score-based models are not required to have a tractable normalizing constant and can be directly acquired through score matching. The measure of this score function determines the optimal path to take in the space of the data distribution to maximize the log probability under the derived real distribution.

As shown in Figure 8b, we visualized the learned gradient field of a diffusion model, which learns to model the expert state-action pairs in MAZE. Once trained, this diffusion model can guide a policy with predicted gradients (blue arrows) to move to areas with high probability, as proposed in our work.

Essentially, by moving in the opposite direction of the source noise, which is added to a data point x_t to corrupt it, the data point is “denoised”; hence the log probability is maximized. This is supported by the fact that modeling the score function



(a) **Maze Layout**

(b) **Learned Gradient Field**

Figure 8. **Visualized Gradient Field. (a) Maze Layout**: The layout of the medium maze used for MAZE. **(b) Learned Gradient Field**: We visualize the MAZE expert demonstration as a distribution of points by their first two dimensions in gray. The points that cluster densely have a high probability, and vice versa. Once a diffusion model is well-trained, it can move randomly sampled points to the area with high probability by predicting gradients (blue arrows). Accordingly, the estimate $p(s, a)$ of joint distribution modeling can serve as guidance for policy learning, as proposed in this work.

is the same as modeling the negative of the source noise. This perspective of the diffusion model is dubbed diffusion SDE. Moreover, Popov et al. (2022) prove that Eq. 12 is diffusion SDE's maximum likelihood SDE solver. Hence, the corresponding divergence optimization problem can be rewritten as:

$$\begin{aligned}&\arg\min_{\theta} D_{KL}(q(x_{n-1}|x_n, x_0)||\phi(x_{n-1}|x_n))\\&= \arg\min_{\theta} \frac{1}{2\sigma_q^2(n)} \frac{(1-\alpha_n)^2}{(1-\bar{\alpha}_n)\alpha_n}[||\hat{\epsilon}_\theta(x_n, n) - \epsilon_0||_2^2],\end{aligned}$$

where ϵ_θ is a function approximator aim to predict ϵ from x. As the coefficients can be omitted during optimization, we yield the learning objective $\mathcal{L}_{\text{diff}}$ as stated in in Eq. 2:

$$\mathcal{L}_{\text{diff}} = ||\hat{\epsilon}(s, a, n) - \epsilon(n)||^2 = ||\phi(s, a, \epsilon(n)) - \epsilon(n)||^2.$$

The above derivation motivates our proposed framework that augments a BC policy by using the diffusion model to provide guidance that captures the joint probability of expert state-action pairs. Based on the above derivation, minimizing the proposed diffusion model loss (*i.e.*, learning to denoise) is equivalent to finding the optimal path to take in the data space to maximize the log probability. To be more accurate, when the learner policy predicts an action that obtains a lower $\mathcal{L}_{\text{diff}}$, it means that the predicted action $\hat{a}$, together with the given state s, fits better with the expert distribution.

Accordingly, by minimizing our proposed diffusion loss, the policy is encouraged to imitate the expert policy. To further alleviate the impact of rarely-seen state-action pairs (s, a), we propose to compute the above diffusion loss for both expert data (s, a) and predicted data $(s, \hat{a})$ and yield $\mathcal{L}_{\text{diff}}^{\text{expert}}$ and $\mathcal{L}_{\text{diff}}^{\text{agent}}$, respectively. Therefore, we propose to augment BC with this objective: $\mathcal{L}_{\text{DM}} = max(\mathcal{L}_{\text{diff}}^{\text{agent}} - \mathcal{L}_{\text{diff}}^{\text{expert}}, 0)$ This design is justified in Section 5.7.2.

F Limitations

This section discusses the limitations of our proposed framework.

- Since this work aims to learn from demonstrations without interacting with environments, our proposed framework in its current form is only designed to learn from expert trajectories and cannot learn from trajectories produced by the learner policy. Extending our method to incorporate agent data can potentially allow for improvement when interacting environments are possible, which is left for future work.

- The key insight of our work is to allow the learner policy to benefit from both modeling the conditional and joint probability of expert state-action distributions. To this end, we propose to optimize both the BC loss and the proposed diffusion model loss. To balance the importance of the two losses, we introduce a coefficient λ as an additional hyperparameter. While the ablation study conducted in MAZE shows that the performance of our proposed framework is robust to λ, this can potentially increase the difficulty of searching for optimal hyperparameters when applying our proposed framework to a new application.

G Broader Impacts

This work proposes Diffusion Model-Augmented Behavioral Cloning, a novel imitation learning framework that aims to increase the ability of autonomous learning agents (*e.g.*, robots, game AI agents) to acquire skills by imitating demonstrations provided by experts (*e.g.*, humans). However, it is crucial to acknowledge that our proposed framework, by design, inherits any biases exhibited by the expert demonstrators. These biases can manifest as sub-optimal, unsafe, or even discriminatory behaviors. To address this concern, ongoing research endeavors to mitigate bias and promote fairness in machine learning hold promise in alleviating these issues. Moreover, research works that enhance learning agents' ability to imitate experts, such as this work, can pose a threat to job security. Nevertheless, in sum, we firmly believe that our proposed framework can offer tremendous advantages in terms of enhancing the quality of human life and automating laborious, arduous, or perilous tasks that pose risks to humans, which far outweigh the challenges and potential issues.