Restoring Noisy Demonstration for Imitation Learning with Diffusion Models

Shang-Fu Chen[*1] **Co Yong**[*2] **Shao-Hua Sun**[1,3]
[1]Graduate Institute of Communication Engineering, National Taiwan University, Taiwan
[2]Data Science Degree Program, National Taiwan University and Academia Sinica, Taipei, 106, Taiwan
[3]Department of Electrical Engineering, National Taiwan University, Taiwan

***Abstract*—Imitation learning (IL) aims to learn a policy from expert demonstrations and has been applied to various applications. By learning from the expert policy, IL methods do not require environmental interactions or reward signals. However, most existing imitation learning algorithms assume perfect expert demonstrations, but expert demonstrations often contain imperfections caused by errors from human experts or sensor/control system inaccuracies. To address the above problems, this work proposes a filter-and-restore framework to best leverage expert demonstrations with inherent noise. Our proposed method first filters clean samples from the demonstrations and then learns conditional diffusion models to recover the noisy ones. We evaluate our proposed framework and existing methods in various domains, including robot arm manipulation, dexterous manipulation, and locomotion. The experiment results show that our proposed framework consistently outperforms existing methods across all the tasks. Ablation studies further validate the effectiveness of each component and demonstrate the framework's robustness to different noise types and levels. These results confirm the practical applicability of our framework to noisy offline demonstration data.**

***Index Terms*—Imitation Learning, Learning from Noisy Data, Data Restoration, Behavioural Cloning, Diffusion Models**

I. Introduction

IMITATION learning [1]–[13] aims to learn a policy from expert demonstrations and has been applied to various applications, including robotics [8], industrial automation, strategy board games, video games, etc [14]–[19]. Compared to reinforcement learning (RL), acquiring a policy in a trial-and-error manner, which can be unsafe or expensive, imitation learning (IL) algorithms can learn without environmental interactions. Furthermore, while designing sophisticated RL reward functions is often difficult and tedious [20], [21], IL methods learn from expert demonstrations and do not require reward signals.

Despite the wide applicability, most existing imitation learning algorithms assume perfect (*i.e.*, optimal and clean) expert demonstrations, which can be challenging and expensive to collect. Specifically, expert demonstrations often contain imperfections caused by errors from human experts or sensor and control system inaccuracies. For example, the sensors may induce noises due to environmental interference [22], [23], and the control system could perform imperfectly due to steady-state error or control jitter [7], [24], [25]. As a result, learning from noisy expert demonstrations using IL methods while neglecting the noises can significantly limit the performance of acquired policies [3], [9], [26], [27].

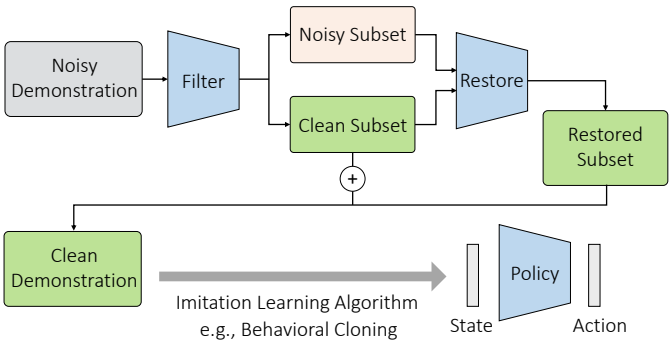


Fig. 1. **Illustration of Diffusion-Model-Based Demonstration Restoration (DMDR)**: We propose a two-stage learning framework that first identifies and filters clean samples from the noisy demonstrations. Then, by learning diffusion models using the clean samples, we restore the remaining noisy samples to provide more reliable demonstrations.

To best leverage expert demonstrations with inherent noises, we propose to (1) filter clean demonstrations from noisy demonstrations, (2) model the clean demonstrations, (3) restore the noisy demonstrations with the learned model, and (4) aggregate the clean and restored demonstrations to learn a policy, as illustrated in Figure 1. In contrast, most existing learning from noisy demonstration methods falls short of implementing this complete pipeline. For example, [27], [28] filter out or give low importance weights on demonstrations determined noisy, failing to extract information from noisy demonstrations; on the other hand, data restoration methods such as [29]–[31] require a known linear degradation model, which is inaccessible for noisy demonstrations in imitation learning. Uniformly restoring entire demonstration sets without separating potentially clean demonstrations can incorrectly modify clean demonstrations and lead to deteriorated learning performance.

This work proposes a filter-and-restore framework, Diffusion-Model-Based Demonstration Restoration (DMDR), for imitation learning from noisy demonstrations. In the demonstration filtering stage, we train autoencoders and perform the local outlier factor [32] using the learned

[*]Equal contribution.
Correspondence to: Shao-Hua Sun <shaohuas@ntu.edu.tw>

embeddings to assign a pseudo label to each data point. In the demonstration restoration stage, we consider the correlation between states and actions and train a pair of conditional diffusion models using the pseudo-labels. One conditional diffusion model aims to restore actions based on the corresponding states, while another diffusion model focuses on the reverse, restoring states based on actions. Then, we aggregate the clean and restored demonstrations and learn a policy using existing IL methods, such as behavioral cloning (BC) [33], [34], implicit behavioral cloning (IBC) [35], and diffusion policy (DP) [36], [37].

We evaluate our proposed framework and existing methods in various domains, including robot arm manipulation, dexterous robotic hand manipulation, and locomotion. The experimental results show that our proposed framework consistently outperforms existing methods across all the tasks. Also, we conduct extensive ablation studies to justify all the components in our filter-and-restore pipeline, including the filtering methods and the restoration settings. The experimental results also confirm that our proposed filter-and-restore pipeline is IL method agnostic, *i.e.*, can be combined with various existing IL methods, including BC, IBC, and DP, and yield improved performance. A noise-type analysis further shows that DMDR maintains considerable advantages under various noise types, including light-tailed, heavy-tailed, biased, and multi-source mixtures. The results highlight DMDR's practical robustness to real-world demonstrations' diverse and complex noise.

II. Related Works

A. Imitation Learning (IL) from expert demonstrations

Imitation learning aims to learn a policy by observing expert demonstrations without reward signals from the environment. Online imitation learning methods use rollouts collected from online interaction to help policy learning. For instance, generative adversarial imitation learning (GAIL) methods [38], [39] learn discriminators that can identify the rollout samples and the expert samples for training a policy that models the expert distribution. Closely related to imitation learning, inverse reinforcement learning (IRL) methods [2], [40], [41] aim to derive the reward function from expert demonstrations for policy learning; Reinforcement Learning from Expert Demonstrations (RLED) [42]–[44] is a paradigm in reinforcement learning that combines expert demonstrations with traditional reinforcement learning techniques to improve learning efficiency and performance. It leverages expert trajectories—sequences of states, actions, and sometimes rewards—to guide the agent's exploration and policy learning.

On the other hand, offline imitation learning methods learn a policy directly from a fixed set of expert demonstrations without environmental interactions. These methods are beneficial when online interactions are expensive, risky, or impractical. Behavior cloning (BC) [34], [45], [46] is a widely studied offline learning approach aimed at imitating expert behaviors through supervised learning; Implicit BC (IBC) [35] learns an energy-based model that takes both states and actions as inputs for better generalization ability; Diffusion Policy [36], [37] employs a diffusion model as a policy to capture multi-modal behaviors that BC struggles to model. However, the effectiveness of these imitation learning methods heavily depends on the quality of the expert demonstrations. Noisy samples in demonstrations can significantly hinder the learning process and the performance of the derived policies [3], [9], [47].

B. Imitation Learning (IL) from Noisy Demonstrations

The issue of noisy demonstrations poses unique challenges to policy learning since the polluted data or trajectories could severely degrade the performance of policy learning. Similar to the case of IL from clean expert demonstrations, recent works have explored online and offline approaches with distinct designs.

For online IL from noisy demonstrations, [48], [49] proposed methods that aim to derive a reward function from suboptimal demonstrations in order to extrapolate better trajectories during inference. Generative adversarial-based approaches [26], [50] assign a confidence or optimality score for training samples to alleviate the interference of the noises. However, in order to evaluate the noisy demonstrations, these online IL methods usually leverage a supplementary dataset with confidence annotations, but such clean and well-annotated data samples are usually hard to collect in real-world cases.

Offline IL methods, which do not require environmental interactions, rely more on the quality of the provided expert demonstrations, making it more challenging to learn effective policies from noisy demonstrations. [51] proposes learning an estimation of the stationary distribution to regularize policy learning and alleviate the disturbance of imperfect demonstrations. [52] extends this approach by imposing additional constraints on optimization to consider the diversity of both states and actions.

Discriminator-based methods [53], [54] assign weights for training samples based on their suboptimality using discriminators in order to emphasize the effect of good data samples while ignoring the bad ones. Behavioral Cloning from Noisy Demonstrations (BCND) [28] assigns weights for training samples based on predictions from previous iterations to seek the major mode of the distribution. However, these methods give low importance weights on demonstrations determined as noisy and fall short of extracting useful information from them.

To address the above issues, we propose a filter-and-restore framework that restores noisy demonstrations to best leverage the noisy demonstrations for policy learning. In our experiments, we compare our proposed framework with BCND to demonstrate the robustness and effectiveness of our approach.

C. Anomaly Detection (Outlier Detection)

Anomaly detection [55] aims to identify samples that deviate from the expected, normal, or typical patterns. Similarly, expert demonstrations represent the *normal* or expected behavior in the domain of imitation learning. Accordingly, we consider corrupted or irrelevant parts of demonstrations as anomalies, such as those caused by sensor glitches, actuator failures, or off-task behaviors.

From the aspect of data characteristics, abnormal samples often stem from rare occurrences or unexpected events, making them inherently diverse and challenging to collect [56]. Therefore, previous literature focuses on learning models to identify the distribution of normality and classify data that deviates from the learned distributions as anomalies.

Classification-based anomaly detection [57]–[62] formulates anomaly detection as a one-class classification problem, where only normal samples are available during training. For instance, [58], [59] utilize a limited set of labeled outliers samples with unlabeled samples to train the classifiers. Alternatively, other works [60]–[62] augment the training data with out-of-distribution or synthetic samples to enhance the classifier's ability to recognize anomalies. Nevertheless, applying these augmentation methods to offline imitation learning is challenging due to the significant differences in behavior among different environments.

On the other hand, reconstruction-based anomaly detection methods utilize autoencoders [63]–[68] to determine outliers based on the reconstruction errors. In these approaches, neural networks are trained to reconstruct input data, with a widely used assumption that deep models tend to learn clean samples faster than noisy samples [69]–[71]. Consequently, samples with higher reconstruction errors are often flagged as anomalies. In this work, we apply a reconstruction-based method to filter outliers for noisy demonstrations

D. Diffusion Models for Data Restoration

Diffusion models have demonstrated remarkable performance in various generation tasks [72]–[82]. Recent works [30], [31], [83] study to extend diffusion models for data restoration tasks, including super-resolution, inpainting, and colorization, etc.

Denoising Diffusion Restoration Models (DDRM) [30] demonstrate that when the restoration task can be formulated as a linear inverse problem, pre-trained diffusion models can be effectively leveraged for inference given the degradation matrix. This approach allows for the utilization of existing models without the need for retraining or fine-tuning for specific restoration tasks. Building upon DDRM, GibbsDDRM [31] relaxes the requirement of the pre-defined degradation matrix and adopts a learnable linear operator to describe the restoration task instead.

However, existing works in this domain often rely on diffusion models pre-trained on clean datasets, which may not be directly applicable to noisy demonstrations commonly encountered in real-world scenarios. To address this limitation, we propose a two-stage framework. In the first stage, we filter clean samples from noisy demonstrations to create a dataset suitable for training conditional diffusion models. In the second stage, we utilize the derived diffusion models to restore the noisy data effectively.

III. Method

In this paper, we aim to learn from noisy demonstrations without environmental interactions or additional annotations for clean samples. The noisy demonstrations $D = \tau_1, ..., \tau_M$ consists of M trajectories, where each trajectory τ_i comprises a sequence of n_i state-action pairs $s_1^i, a_1^i, ..., s_{n_i}^i, a_{n_i}^i$. For each state and action in the demonstration, there exists a small probability p that random noise is injected, indicating the noise level. Notably, the pollution of states and actions is independent since distinct sensors are typically used for monitoring states and actions in real-world scenarios. To simplify notation, we denote a state-action pair sampled from the entire demonstration dataset as $(s, a) \sim D$, dismissing the trajectory index and the index of a state-action pair within a trajectory.

We propose a *filter–and–restore* framework for imitation learning from noisy demonstrations. Given the noisy demonstrations D, the **Demonstration Filtering** stage (Section III-A) combines an autoencoder with Local Outlier Factor to flag anomalous elements, denoted as s' or a'. In contrast, those who pass the test are marked as $\hat{s}$ or $\hat{a}$, and treated as *clean*. The **Demonstration Restoration** stage (Section III-B) trains conditional diffusion models on these clean samples and then maps each flagged component s' or a' to its denoised counterpart s^* or a^*, resulting in *restored* pairs. After these two steps, we obtain a refined dataset composed of fully clean pairs $(\hat{s}, \hat{a})$ together with partially or fully restored pairs $(s^*, \hat{a})$, $(\hat{s}, a^*)$, and (s^*, a^*), which serves as reliable supervision for downstream policy learning.

A. Demonstration Filtering

In this stage, we aim to filter clean samples from noisy demonstrations by integrating autoencoders and Local Outlier Factor (LOF). The process involves learning global features of the data distribution with autoencoders and then applying LOF to identify potential outliers based on local densities. While demonstrations typically consist of sequences of state-action pairs, in many applications, states and actions represent distinct properties and are usually captured by different sensors. For instance, states commonly denote variables like position or RGB images, whereas actions typically record the torque applied to the joints of robots, as seen in the widely utilized MuJoCo environment [84], Pybullet [85], OpenAI Gym [86], Isaac Gym [87], etc. The noisy demonstrations may arise from control failures that occurred during the data collection process. For instance, performing the desired actions (applying desired torques on joints) could face control system errors, such as steady-state error and control jitter [7], [24], [25], so states and actions would encounter noise perturbation independently. It is worth noting that the goal of this paper is to learn a policy from given noisy demonstrations instead of solving internal control problems directly. As a result, we filter and label state and action samples independently.

1) Anomaly Detection with Autoencoders: We apply a reconstruction-based method to detect and filter potential abnormal samples from noisy demonstrations. As depicted in Figure 2a, we train a pair of autoencoders to capture the majority of states and actions with a reconstruction loss, which can be formulated as follows:

$$\mathcal{L}_{\text{rec}}^{s} = \mathbb{E}_{(s,a)\sim D}\left[\left|\left|\phi(s) - s\right|\right|^2\right], \tag{1}$$

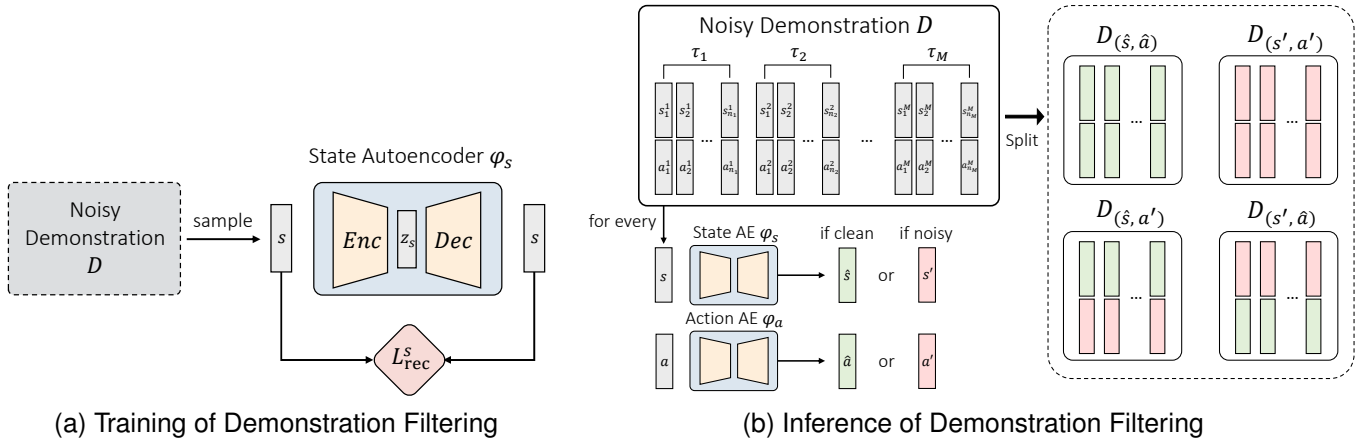


Fig. 2. **Demonstration filtering. (a) Training.** We train the state autoencoder ϕ_s using the reconstruction loss and apply the Local Outlier Factor (LOF) on the feature space, *i.e.*, z_s. We only show the state autoencoder ϕ_s in the figure for illustration, while the action autoencoder ϕ_a follows the identical architecture. **(b) Inference.** We use autoencoders and LOF to individually identify outliers for states and actions. With the predictions of outliers, we filter the noisy demonstrations into four subsets: $D_{(\hat{s},\hat{a})}$, $D_{(s',a')}$, $D_{(\hat{s},a')}$, and $D_{(s',\hat{a})}$, which contains clean state-action pairs, noisy state-action pairs, clean states with noisy actions, and noisy states with clean actions, respectively.

and

$$\mathcal{L}^a_{\text{rec}} = \mathbb{E}_{(s,a)\sim D}\left[||\phi(a) - a||^2\right], \tag{2}$$

where $\phi(s)$ indicates the state autoencoder and $\phi(a)$ indicates the action autoencoder. One can directly adopt L^s_{rec} and L^a_{rec} to filter outliers as reconstruction-based anomaly detection approaches do.

However, in tasks like robot arm manipulation, we find that certain state-action pairs may be infrequent yet crucial for successful execution. For instance, in a scenario where a robot arm needs to grasp an object and arise it to a target location. While most state-action pairs may correspond to routine movements, such as navigating the arm, there are specific instances, such as the action of grasping an object, that occur less frequently but are indispensable for task completion. To prevent discarding essential samples, we further apply the Local Outlier Factor algorithm to identify outliers based on the encoded representations obtained from autoencoders.

2) Combining Autoencoder and Local Outlier Factor: Local Outlier Factor (LOF) is known as an anomaly detection algorithm that measures the local deviation of samples. The algorithm begins by estimating the local density for each sample in the dataset using the k-nearest neighbors (KNN) approach. It calculates the density by considering the distance to each sample's k nearest neighbors. Next, a ratio of local density is computed using the sample and its neighbors, which serves as the LOF score. A larger LOF score indicates that the sample has a lower density than its neighbors, suggesting it may be an outlier. LOF effectively identifies potential outliers in the dataset by analyzing each local region individually.

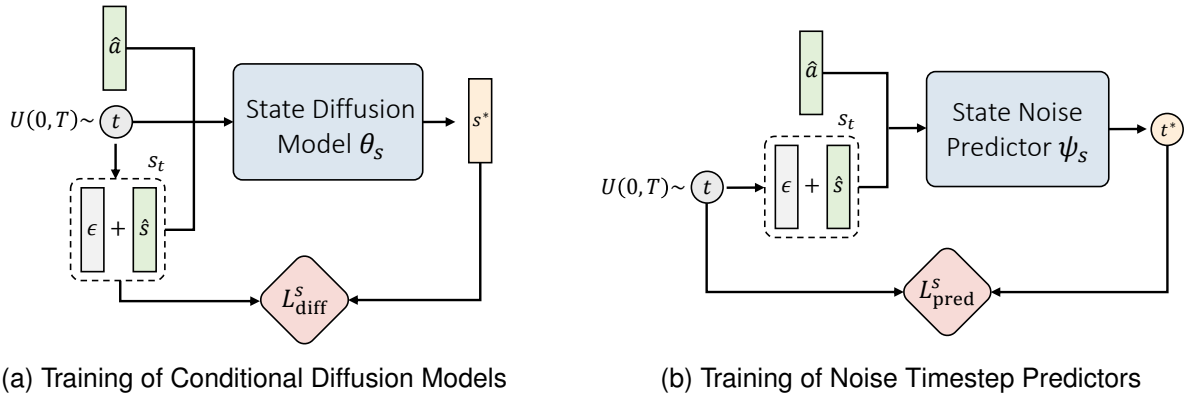
As we mentioned in the previous section, to prevent predicting state-action pairs with unique behaviors as anomalies, we utilize the bottleneck features z_s and z_a from the autoencoders to calculate LOF score for each s and a. These representations capture global behaviors since the autoencoders are trained to minimize reconstruction loss across the entire expert demonstration dataset. LOF then evaluates outliers based on the local density of these representations. By considering the local density of samples' representations, we effectively prevent discarding infrequent samples with useful behaviors, thereby enhancing the robustness of demonstration filtering for imitation learning tasks.

The goal of anomaly detection is to identify samples that differ from normal or typical patterns. Therefore, in anomaly detection literature [32], [57], [88], data that deviate from the majority are typically regarded as anomalies. This assumes that the majority of the training dataset, meaning at least half of the samples, is composed of normal data. Following this assumption, half of the samples with lower LOF scores are labeled as clean ($\hat{s}$ and $\hat{a}$), while the other half is labeled as noisy (s' and a'). This design choice will be further discussed in Section IV-F. As shown in Figure 2b, we filter the dataset into four subsets according to the pseudo-labels: clean state-action pairs $D_{(\hat{s},\hat{a})}$, clean states with noisy actions $D_{(\hat{s},a')}$, noisy states with clean actions $D_{(s',\hat{a})}$, and noisy state-action pairs $D_{(s',a')}$.

B. Demonstration Restoration

In this section, we show how to restore the noisy subsets of demonstrations with diffusion models. Previous works [30], [31] have shown how to use diffusion models for image restorations when the distortion process is known. For instance, Denoising Diffusion Restoration Models (DDRM) [30] utilizes a given degradation matrix for each restoration task to solve the linear inverse problem and GibbsDDRM [31] assumes the distortion can be modeled by a learnable blurring kernel. However, such information is not available in offline imitation learning. To restore demonstrations without a given degradation matrix, we utilize conditional diffusion models to consider the relationships between the corresponding state and action. Moreover, we introduce noise timestep predictors to guide the denoising process of the diffusion models for accurate restoration. We elaborate on the learning of these components in the following subsections.

1) Denoising Diffusion Probabilistic Models (DDPMs): This work uses DDPMs [74] for data restoration. During the training stage, DDPMs gradually add Gaussian noise to each



(a) Training of Conditional Diffusion Models (b) Training of Noise Timestep Predictors

Fig. 3. **Demonstration restoration. (a) Training of conditional Diffusion Models.** We train the state diffusion model θ_s using the condition of clean action $\hat{a}$ to restore the noise-injected state s_t. We employ an identical architecture for the action diffusion model θ_a, which restores the noise-injected action a_t using the clean state $\hat{s}$. **(b) Training of Noise timestep predictors.** To estimate the appropriate noise timestep during inference, we train the state noise predictor ψ_s given a clean action $\hat{a}$. The action noise predictor ψ_a follows the identical architecture.

data sample until it becomes isotropic Gaussian, which is called the forward diffusion process. Then, DDPMs learn to denoise the noise-injected sample to the original data, called the reverse diffusion process. Given a data point x_0 sampled from dataset D, *e.g.*, a state s or an action a sampled from the demonstrations, latent variables $x_1, ..., x_T$ are produced in the forward diffusion process, where T is the number of diffusion steps, and x_T is an isotropic Gaussian. The diffusion model θ learns to reverse the diffusion process by predicting the injected noise ϵ on the sample. The objective of DDPM can be formulated as the following:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{t\sim\mathcal{U}(0,T),x\sim D}\left[||\epsilon - \epsilon_\theta(\alpha_t x_0 + \sigma_t\epsilon, t)||^2\right], \quad (3)$$

where t is sampled from a uniform distribution $\mathcal{U}(0,T)$ and $\sigma_t = \sqrt{1-\alpha_t{}^2}$ is a scalar representing increasing noise schedule. Once the diffusion model θ is learned, one can sample a random noise and use the predicted noise ϵ_θ to compute the next latent variable. The above process is repeated iteratively until a clean sample x_0 is generated.

The reverse process of DDPMs, *i.e.*, step-by-step denoising by predicting the added noise, is aligned with the nature of the signal restoration. From the theoretical aspects, [74], [89] demonstrate that the diffusion model is able to optimize the variational lower-bound of the log-likelihood of the data distribution $p(x_0)$, and outperform other likelihood-based models, such as autoregressive models and variational autoencoders (VAEs); [90] offers an alternative perspective by formulating the diffusion process as a continuous-time stochastic differential equation (SDE) in the limit as $T \to \infty$. This interpretation transforms the problem into denoising score matching, ensuring that the trained model converges to an optimal denoiser. The above evidence motivates us to apply DDPMs to restore expert demonstrations polluted by noise.

2) Learning Conditional Diffusion Models: To consider the correlation between states and actions, we train a pair of conditional DDPMs for states and actions, respectively, using the filtered subset containing clean state-action pairs $D_{(\hat{s},\hat{a})}$. The state diffusion model θ_s aims to restore states based on the corresponding actions, while the action diffusion model θ_a focuses on restoring actions based on states. The state diffusion model θ_s considers the corresponding action to predict the noise-injected states. The objective can be calculated as follows:

$$\mathcal{L}^s_{\text{diff}} = \mathbb{E}_{t\sim\mathcal{U}(0,T),(\hat{s},\hat{a})\sim D_{(\hat{s},\hat{a})}}\left[||\epsilon - \epsilon_{\theta_s}(s_t, \hat{a}, t)||^2\right], \quad (4)$$

where $s_t = \alpha_t\hat{s} + \sigma_t\epsilon$ is the noise-injected state, $\hat{a}$ is the corresponding clean action, and t is the noise timestep sampled from a uniform distribution over diffusion steps. Similarly, we can define the objective for the action diffusion model θ_a as follows:

$$\mathcal{L}^a_{\text{diff}} = \mathbb{E}_{t\sim\mathcal{U}(0,T),(\hat{s},\hat{a})\sim D_{(\hat{s},\hat{a})}}\left[||\epsilon - \epsilon_{\theta_a}(a_t, \hat{s}, t)||^2\right], \quad (5)$$

where $a_t = \alpha_t\hat{a} + \sigma_t\epsilon$ is the noise-injected action, $\hat{s}$ is the corresponding clean state, and t is the sampled noise timestep.

Both conditional diffusion models are trained on the subset of clean pairs $D_{(\hat{s},\hat{a})}$ to ensure the learned models can accurately capture the correct relationships between the states and actions. After learning the diffusion models, we can apply the state diffusion model θ_s on the subset $D_{(s',\hat{a})}$ to restore the noisy states conditioned on the corresponding clean action. Also, the action diffusion model θ_a is applied on $D_{(\hat{s},a')}$ to restore the noisy actions. We discard the noisy state-action pairs from the subset $D_{(s',a')}$ since the polluted state and action can not provide sufficient information for restoration.

3) Learning Noise Timestep Predictors: In the previous section, we have derived diffusion models that can gradually denoise a sampled isotropic noise to a clean state or action with the iterative reverse diffusion process. However, the noisy states and actions are still more informative than isotropic Gaussian noises despite being polluted by noise. To best leverage these samples, we aim to predict a noise timestep t for each noisy sample during the denoising process. With the predicted noise timesteps, the trained diffusion models are able to perform restoration according to the scale of noise on each sample. To this end, we introduce a pair of noise predictors ψ_s and ψ_a to predict the noise timestep for states and actions, respectively.

Similar to the diffusion models, the noise timestep predictors are conditioned on the corresponding state or action. The

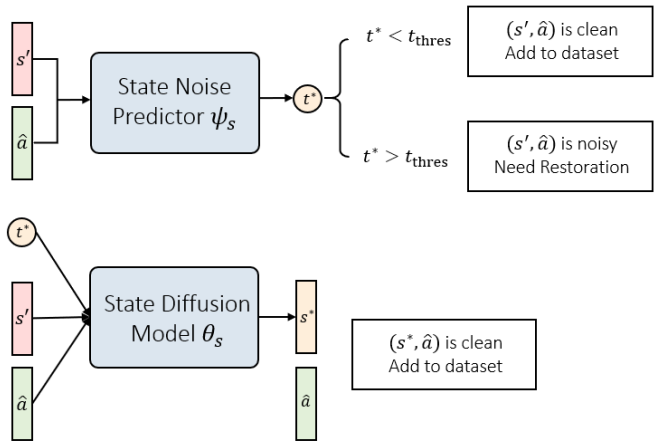


Fig. 4. **Inference of demonstration restoration.** Given a noisy state s' and a clean action $\hat{a}$ from $D_{(s',\hat{a})}$, we first predict the noise timestep t^* for the noisy state. If t^* is less than a predefined threshold t_{thres}, then we append it to the clean subset $D_{(\hat{s},\hat{a})}$ directly. Otherwise, we denoise the noisy state using the state diffusion model θ_s, conditioned on the clean action and the predicted noise timestep. Similarly, noisy actions in $D_{(\hat{s},a')}$ can be restored using the action noise predictor ψ_a and the action diffusion model θ_a.

training objective for the state noise predictor ψ_s is to predict the correct noise timestep for noise-injected states from the clean subset $D_{(\hat{s},\hat{a})}$, which can be calculated as follows:

$$\mathcal{L}^s_{\text{pred}} = \mathbb{E}_{t\sim\mathcal{U}(0,T),(\hat{s},\hat{a})\sim D_{(\hat{s},\hat{a})}}\left[||t - \psi_s(s_t, \hat{a})||^2\right], \quad (6)$$

where $s_t = \alpha_t \hat{s} + \sigma_t \epsilon$ is the noise-injected state, and $\hat{a}$ is the clean action to be served as the condition. Similarly, we can learn the action noise predictor ψ_a by the following equation:

$$\mathcal{L}^a_{\text{pred}} = \mathbb{E}_{t\sim\mathcal{U}(0,T),(\hat{s},\hat{a})\sim D_{(\hat{s},\hat{a})}}\left[||t - \psi_a(a_t, \hat{s})||^2\right], \quad (7)$$

where $a_t = \alpha_t \hat{a} + \sigma_t \epsilon$ is the noise-injected action, and $\hat{s}$ is the clean state to be served as the condition.

4) Restoration with Diffusion Models and Noise Predictors: The restoration of noisy states in $D_{(s',\hat{a})}$ and noisy actions in $D_{(\hat{s},a')}$ follows analogous approaches. Here, we illustrate how to restore the noisy states in $D_{(s',\hat{a})}$ with the trained state noise predictor ψ_s and the state diffusion model θ_s.

As shown in Figure 4, we sample a state-action pair from $D_{(s',\hat{a})}$, where the state s' is noisy, and the action $\hat{a}$ is clean, predicted in the previous filtering stage. To restore the noisy state, we first estimate the noise timestep t^*, which measures the deviation from the major behaviors in the noisy demonstrations. Since we only keep half of the samples in the clean subset during the filtering stage, it is possible that there are clean samples remaining in the noisy subsets as well. Therefore, we use t^* to filter samples by applying a noise timestep threshold t_{thres}. If $t^* < t_{\text{thres}}$, the state is assumed clean and added directly to the clean dataset $D_{(\hat{s},\hat{a})}$. Otherwise, the state s' is restored using $\hat{a}$ and t^* via the state diffusion model, and the restored pair is added to the clean dataset. The noisy actions in $D_{(\hat{s},a')}$ can be restored following a similar procedure.

The effectiveness of the predictors and the noise timestep thresholding are studied in the Section IV-G. While treating all noisy samples as isotropic Gaussian is a viable approach for restoration, our experiments demonstrate that incorporating predictors and noise timestep thresholding enhances the accuracy of diffusion models in restoration tasks.

IV. Experiments

In this section, we evaluate how our Diffusion-Model-Based Demonstration Restoration (DMDR) benefits offline imitation learning with noisy demonstrations. We first introduce the environmental setup for the experiments (Section IV-A) and the baselines (Section IV-B). The experimental results (Section IV-C) show that DMDR is more effective than other baselines on various continuous control tasks. We then do the ablation studies to verify our design choice for both the filtering stage (Section IV-E and Section IV-F) and the restoration stage (Section IV-G). Finally, Sections IV-H to IV-J establish the generalizability and robustness of DMDR by showing consistent performance gains between imitation learning algorithms, diverse types of noise, and different levels of noise.

A. Experimental Setup

This paper addresses offline imitation learning using noisy demonstrations D, which consist of sequences of unlabeled (s, a). Since state measurements (s) and action controls (a) are typically derived from different sensors and actuators, we assume they are polluted by independent noise.

To simulate noisy demonstrations, we add noise to expert data (s_e, a_e) as follows:

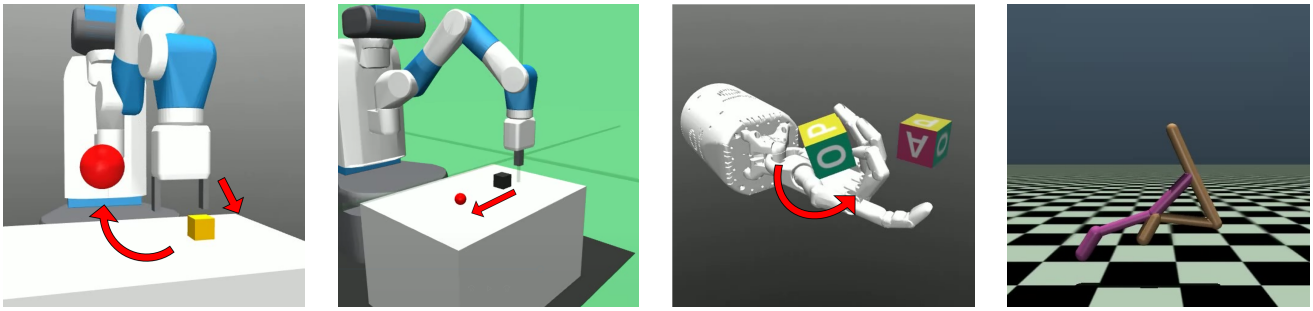
$$s = \begin{cases} s_e + n_s, & \text{with probability } p, \\ s_e, & \text{with probability } 1-p, \end{cases} \quad (8)$$

$$a = \begin{cases} a_e + n_a, & \text{with probability } p, \\ a_e, & \text{with probability } 1-p, \end{cases} \quad (9)$$

Here, noise level p denotes the probability of a state or action being corrupted by noise.

Following previous studies that identify zero-mean Gaussian as the dominant sensor noise [23], [91], [92], we inject $n_s \sim \mathcal{N}(0, \sigma_s^2)$ and $n_a \sim \mathcal{N}(0, \sigma_a^2)$ in the main experiments. Other noise distributions (Uniform, Laplacian, biased, and mixed variants) are analyzed separately in Section IV-I.

All expert data (s_e, a_e) used in our experiments are generated using pre-trained reinforcement learning agents, specifically using Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC) algorithms. These agents are optimized using reward functions provided by the simulation environments, ensuring that the demonstrations reflect expert-level performance. For the simulation environments, we utilized the MuJoCo physics engine [84] and the OpenAI Gym [86] toolkit for our simulations. MuJoCo (Multi-Joint dynamics with Contact) is a physics engine designed for research and development in robotics, biomechanics, graphics, and animation; OpenAI Gym is a toolkit for developing and comparing reinforcement learning algorithms. It provides a variety of environments, including classic control tasks, Atari games, and robotic simulations. Details of model architectures and



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

Fig. 5. **Environments & Tasks. (a)-(b) FETCHPICK and FETCHPUSH**: The robot arm manipulation tasks employ a 7-DoF Fetch robotics arm to pick up/push an object from the table and move it to a target location. **(c) HANDROTATE**: This dexterous manipulation task requires a Shadow Dexterous Hand to in-hand rotate a block to a target orientation. **(d) WALKER**: These locomotion tasks require learning agents to walk as fast as possible while maintaining their balance.

hyperparameters can be found in our supplementary materials in Section VII.

We employ the proposed framework (DMDR) in various continuous control domains. As shown in Figure 5, we illustrate the environments and tasks used in the experiments in the following:

- **Robot Arm Manipulation.** We leverage the FETCHPICK and FETCHPUSH environments to represent the robot arm manipulation tasks. These environments are designed for the Fetch Mobile Manipulator, ensuring that state and action representations align with the real-world robot system to enhance potential real-world applicability. These tasks aim to control a 7-DoF robot arm to interact with an object and achieve a defined target. FETCHPICK (Figure 5a) requires picking up an object from the table and raising it to a target location. On the other hand, FETCHPUSH (Figure 5b) requires pushing an object on the table and moving it to a target location. We use 10k state-action pairs provided by [41] as our expert data for both FETCHPICK and FETCHPUSH, which contains 303 and 185 trajectories, respectively. We set the noise level p to 0.2 to create noisy demonstrations.
- **Dexterous Manipulation.** We leverage the HANDROTATE [93] environment to represent a challenging Dexterous Manipulation task. The task requires controlling a dexterous hand and in-hand rotates a block to a target orientation (Figure 5c). The environment takes 68 dimension states and outputs 20 dimension actions, which is high-dimensional compared to the commonly-used environments in imitation learning. We collect 515 trajectories with 10k state-action pairs as our expert data. We set the noise level p to 0.4 to create noisy demonstrations.
- **Locomotion.** We leverage the WALKER [94] environment to represent the locomotion tasks. This environment requires controlling a bipedal agent to travel toward the x-axis direction as fast as possible while maintaining the balance (Figure 5d). If the agent loses its balance, *e.g.*, the height of the agent is too low, the episode would terminate before the maximum number of steps is reached. We use 20 trajectories with 20k state-action pairs provided by [95] as our expert data. We set the noise level p to 0.2 to create noisy demonstrations.

B. Baselines

To evaluate the effectiveness of the proposed method, we compare our DMDR with other offline imitation learning baselines for noisy demonstrations. Under this problem formulation, the baselines should not require environmental interactions or additional annotations. Therefore, online methods such as [26], [48]–[50], [96] are not applicable and we compare our method with the following baselines:

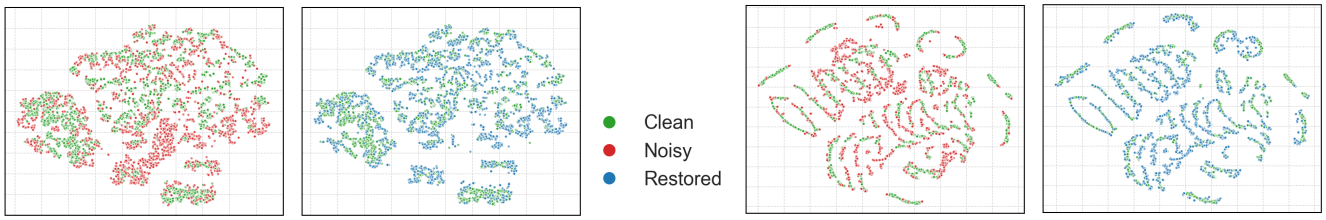
- **BC.** Behavioral Cloning (BC) is a straightforward approach to offline imitation learning. It learns a policy that directly maps from states to actions by imitating the behavior demonstrated in training data using supervised learning. However, BC struggles to deal with noisy demonstrations since the policy tends to overfit noisy data, making it challenging to learn the underlying dynamics accurately.
- **Ensemble BC.** To mitigate the impact of noisy demonstrations, one effective strategy is to employ ensemble techniques. Ensemble BC extends the basic BC approach by training several policies and aggregating their outputs. Ensemble BC reduces the bias and variance of the prediction and thus improves the overall resilience to noise perturbation. Implementation details for Ensemble BC, and a further study for ensemble policies are elaborated in Section VIII of our supplementary material.
- **BCND.** Behavioral Cloning from Noisy Demonstrations (BCND) [28] aims to learn robust policies from noisy demonstrations containing optimal and sub-optimal behaviors. They apply ensemble policies and further design an algorithm to assign weights for each training sample. With the derived weights, the policies are encouraged to capture the behaviors of the major distribution of training samples, which are assumed to be clean and optimal.

C. Quantitative Results

We compare our DMDR with baselines and show the experimental results in Table I. The results are separated into two parts: methods with a single policy and methods that leverage the ensemble technique with multiple policies. We also report the result of DMDR when the ensemble technique is applied (Ensemble DMDR). We evaluate the agents with

TABLE I
OVERALL EXPERIMENT RESULTS

Method	# of policies	FETCHPICK	FETCHPUSH	HANDROTATE	WALKER
BC	1	44.38% ± 11.02%	67.97% ± 4.81%	45.56% ± 6.15%	4456.8 ± 1051.1
DMDR (Ours)	1	**90.52**% ± 4.83%	**79.64**% ± 5.30%	**51.70**% ± 5.85%	**5066.4** ± 886.4
Ensemble BC	5	51.36% ± 6.67%	72.25% ± 3.20%	51.03% ± 4.33%	5170.6 ± 395.2
BCND [28]	5	52.87% ± 12.71%	71.01% ± 17.98%	**54.97**% ± 4.49%	5144.8 ± 739.6
Ensemble DMDR (Ours)	5	**91.80**% ± 2.54%	**82.03**% ± 2.87%	**55.36**% ± 4.43%	**6168.1** ± 284.9



(a) Noisy State (b) Restored State (c) Noisy Action (d) Restored Action

Fig. 6. **Visualization Results on FETCHPICK.** We use t-SNE to project noisy demonstrations from the FETCHPICK environment. The clean, noisy, and restored samples are represented in green, red, and blue, respectively. From Figure 6a and Figure 6b, it can be observed that the restored states (blue) are distributed significantly closer to the clean states (green) than the noisy states (red) prior to the restoration process. Similar results can be observed for the actions by comparing Figure 6c and Figure 6d.

100 episodes and five random seeds on all tasks. We report the average and standard deviation of the success rate for FETCHPICK, FETCHPUSH, and HANDROTATE and return for WALKER.

We observe that the proposed DMDR outperforms baselines whether the ensemble policies are applied or not. The above results verify the effectiveness of our demonstration filtering and restoration process. BC struggles with learning from noisy demonstrations, which suggests that the noisy data points severely hinder performance. By aggregating multiple BC policies, Ensemble BC improves its performance compared to BC, and the standard deviations are lower than those from BC in all tasks, which infers that ensemble policies are more robust to noisy demonstrations. For BCND, it slightly outperforms Ensemble BC, especially in HANDROTATE. However, it produces a large standard deviation in most of the tasks since the weight-assigning mechanism is affected by the noisy samples at the early training stage. The learning curves for these experiments are presented in Section X of the supplementary material.

D. Visualization Results

To demonstrate the effectiveness of our restoration approach intuitively, we conduct a visualization experiment using t-SNE projections with noisy demonstrations from the FETCHPICK environment. As illustrated in Figure 6, clean, noisy, and restored samples are represented in green, red, and blue, respectively. The locations of the clean samples are fixed for easier comparison. From Figure 6a and Figure 6b, it can be observed that the restored states (blue) are distributed significantly closer to the clean states (green) than the noisy states (red) prior to the restoration process. Similar results can be observed for the actions by comparing Figure 6c and Figure 6d. These results confirm that our restoration process effectively recovers distorted states and actions. Consequently, the policy trained on the restored dataset achieves improved performance.

E. Ablation Study for Demonstration Filtering

To evaluate the effectiveness of our demonstration filtering design, we ablate the filtering part while fixing the restoration algorithm and the setting for the policy learning. We compare the filtering approaches listed in the following:

- **Random Filtering.** A naive way to do filtering is to randomly label half the states and actions as clean independently. This baseline serves as a bottom line for filtering methods.
- **Autoencoder (AE).** Two autoencoders are learned for states and actions, respectively. Since autoencoders tend to capture the major behaviors of training data, samples with higher reconstruction losses are considered outliers and labeled as noisy.
- **Local Outlier Factor (LOF).** LOF estimates the local deviation of a data point with respect to its neighbors given a dataset. Specifically, we filter samples by computing the LOF score based on k-nearest neighbors and then identifying a sample as an outlier if the score is large.
- **Ours.** Our method calculates the LOF scores for each data point based on the features derived from the autoencoders. This method considers global representations with the autoencoders and local deviations of samples with the LOF algorithm.

The results of filtering ablation are shown in Table II. We note that employing the naive random filtering method demon-

TABLE II
DEMONSTRATION FILTERING ABLATION STUDY

Method	# of Samples	Success Rate
Noisy Data	10000	45.40% ± 11.33%
Random Filtering	7505	51.20% ± 6.80%
AE	7313	86.00% ± 3.97%
LOF	7311	76.75% ± 27.93%
Ours	7362	**91.80% ± 4.09%**

strates an improvement in success rate alongside a reduction in variance. We hypothesize that the improvement results from the restoration process. Even though the training data for diffusion models and noise predictors are still noisy because of random filtering, the diffusion models can still improve the quality of samples by restoration. The above results strengthen the motivation of our filter-and-restore framework.

Utilizing Autoencoder (AE) or Local Outlier Factor (LOF) individually for filtering demonstrations shows a respectable improvement. However, the autoencoders only capture the majority of all training samples and can only better describe the global behaviors of the demonstrations. On the other hand, the Local Outlier Factor (LOF) only considers local features and is unaware of global behaviors. These restrictions hold back their ability to filter out noisy samples accurately.

In contrast, our proposed approach, which combines AE and LOF, capitalizes on the strengths of both methods. By integrating global and local feature representations, our method surpasses the performance of its components, leading to superior results.

F. Ablation Study for Trusted Data Fraction

In the demonstration filtering stage, we rank all samples by their Local Outlier Factor (LOF) scores and keep the lowest-ranked fraction as trusted data for training the diffusion model. This follows the *majority-clean assumption* widely used in unsupervised anomaly detection, which states that at least half of the dataset is clean. Accordingly, our default choice is to retain the lowest 50 % of the samples. To confirm that this design remains effective in practice, we vary the trusted data fraction τ from 30% to 90% and measure the downstream policy performance on FETCHPICK, keeping all other components of DMDR unchanged.

TABLE III
POLICY PERFORMANCE ON PICK UNDER DIFFERENT TRUSTED DATA FRACTIONS τ.

τ	Discarded samples	Success Rate
90 %	107	53.80% ± 8.35%
80 %	414	70.60% ± 6.03%
70 %	993	84.60% ± 7.23%
60 %	1760	87.80% ± 6.72%
50 %	**2,638**	**90.52% ± 4.83%**
40 %	3760	32.60% ± 4.88%
30 %	4993	3.00% ± 1.41%

The results in Table III show a steady improvement in performance as the trusted data fraction decreases from 90% to 30%, but performance drops sharply when fewer than 50% of the samples are retained. Higher values of τ preserve more data and increase diversity, but also include noisier samples that negatively affect the quality of the learned restorations. On the other hand, lower values of τ remove too many samples, reducing coverage and limiting the downstream policy to learn. The performance peak is exactly at $\tau = 50\%$. This is the largest subset where the clean and noisy separation remains trustworthy, matching the majority-clean assumption that at least half of the dataset is uncontaminated. We keep $\tau = 50\%$ as a robust and task-agnostic setting for all experiments in DMDR.

G. Ablation Study for Demonstration Restoring

To verify the effect of demonstration restoration and evaluate the designs of our restoration method proposed in Section III-B, we compare different strategies that deal with samples from the noisy subsets.

We compare our method with the following baselines and variants on FETCHPICK and WALKER environments:

- **Random forest regressor**: Random forest regressor makes predictions based on the predictions from multiple decision trees. We train the regressor using the clean state-action pairs from $D_{(\hat{s},\hat{a})}$ and use the trained regressor to predict the noisy states or actions given the corresponding clean actions or states.
- **Generation**: Given the diffusion models learned in the filtering stage, this baseline directly generates states/actions based on the corresponding actions/states. We compare our restoration method with this baseline to verify the benefits of restoring noisy samples instead of generating samples from isotropic Gaussian noises directly.
- **Ours w/o predictor**: To verify the effectiveness of our noise timestep predictors, we apply restoration with a fixed noise timestep t for the diffusion model. Given that the total number of diffusion steps T is 100, we set the fixed noise timesteps as 50.
- **Ours w/o t_{thres}**: As described in Section III-B3, the predicted noise timestep t^* from predictors can be used to filter noisy demonstrations by thresholding. To evaluate the above design, we employ a variant of our method that does not apply thresholding and directly restores the noisy samples based on t^*.
- **Ours**: Our restoration method utilizes the predicted noise predictors to predict the noise timestep for each noisy sample and further sets a threshold t_{thres} to filter samples with smaller noise timesteps before restoring them.

The results of restoration ablation are shown in Table IV. The random forest regressor is outperformed by other diffusion-model-based restoration methods, which verifies the effectiveness of using diffusion models for recovering data. We observe that all diffusion-model-based methods perform similarly on FETCHPICK, including the augmentation with diffusion models baseline. The results infer that a well-trained diffusion model can directly generate informative training data

TABLE IV
DEMONSTRATION RESTORATION ABLATION STUDY

Method	FETCHPICK (Success Rate)	WALKER (Return)
Random forest regressor	65.00% ± 12.71%	255.3 ± 48.1
Generation	**89.89**% ± 5.13%	3452.6 ± 1528.8
Ours w/o predictor	88.20% ± 4.82%	2702.9 ± 1281.3
Ours w/o t_{thres}	89.00% ± 4.00%	4261.4 ± 857.2
Ours	**90.52**% ± 4.83%	**5066.4** ± 886.4

from noises in this environment. Therefore, all of the diffusion-model-based methods perform well regardless of whether the noise predictors are included.

On the other hand, our restoration method outperforms all baselines and variants in WALKER. The results indicate that generating data from noise and restoring data without noise predictors can not restore the noisy sample effectively and highlight the importance of utilizing noise predictors and the thresholding method.

H. Imitation Learning Algorithms with DMDR

Using DMDR to restore demonstrations not only benefits policy learning of BC but also other imitation learning algorithms. Here, we evaluate three offline imitation learning algorithms on the FETCHPICK environment to compare the performance when using noisy demonstrations and the demonstrations restored by our DMDR.

- **BC.** Behavioral cloning (BC) is a straightforward baseline that learns a policy to map states to actions directly using the mean square error (MSE) for training.
- **Implicit BC** [35] utilizes an energy-based model (EBM) to train an implicit behavior-cloning policy, which models the expert policy. The training of the energy-based model employs the InfoNCE loss, as described in [97].
- **Diffusion Policy** Diffusion Policy [36], [37] learn a conditional diffusion model using diffusion loss to predict actions given the observed states. During inference, the diffusion model takes the current state as a condition and gradually denoises the action from noise with the reverse diffusion process.

As shown in Table V, all algorithms struggle to learn directly from the noisy demonstrations but can significantly improve using the restored demonstrations from our DMDR, while Implicit BC and Diffusion Policy are even more sensitive to noisy demonstrations than BC. DMDR benefits these imitation learning algorithms by restoring the noisy demonstrations and results in superior performances with more stable training progressing (lower standard deviation), enabling broader applications of real-world scenarios.

I. Ablation Study on Noise Types

To verify that the proposed DMDR framework is not restricted to a single noise model, we repeat the FETCHPICK experiment under various noise types that together cover light-tailed, heavy-tailed, biased, and mixture-sourced noise. All settings keep the overall noise level $p = 0.2$ and fix the noise standard deviation $\sigma_s = \sigma_a = \frac{1}{6}$. The results are shown in Table VI.

TABLE V

Imitation Learning Algorithms with DMDR

Algorithm	Noisy Demo.	Restored Demo.
BC	44.38% ± 11.02%	**90.52**% ± 4.83%
Implicit BC	3.00% ± 3.24%	**44.56**% ± 6.98%
Diffusion Policy	25.80% ± 2.59%	**97.40**% ± 2.19%

- **Unbiased noise.** DMDR maintains high success under Gaussian and Laplacian noise, indicating that the filter-and-restore pipeline is not tied to light-tailed assumptions. However, under uniform noise DMDR (72.0%) is *comparable* to BC (74.8%). LOF relies on spotting points whose local density is much lower than that of their k neighbours [32]. Uniform noise fills the space evenly, so every sample has neighbors at a similar density, which hinders the performance of density-based anomaly detection [98].
- **Biased noise.** To simulate systematic sensor drift, we inject a constant offset of $+0.4$ into every corrupted sample, producing *Gaussian (biased)* and *Uniform (biased)* settings.
  BC struggles with these shifted demonstrations. Our goal is to test whether DMDR can detect displacement and restore demonstrations to the clean manifold. Significant performance improvements confirm DMDR's ability to correct systematic shifts that conventional imitation learning suffers.
- **Multi-source noise.** Real sensor errors rarely come from a single mechanism, so we also evaluate two mixture settings: *Gaussian+Uniform* (half of the corruptions sampled from a zero-mean Gaussian, half from a zero-mean Uniform range) and *Gaussian+Gaussian* (a bimodal Gaussian with peaks at ± 0.4).
  DMDR improves BC's performance *without knowing to underlying noise distribution*. The result confirms that DMDR can handle realistic, multi-sourced noise.

These results confirm that **DMDR generalizes beyond Gaussian noise** and is resilient to a spectrum of realistic noise, with the only limitation arising in the edge case of uniform noise—where density-based anomaly detection performs poorly. This suggests future work to explore outlier detection strategies leveraging temporal or model-based structure.

J. Effect of Noise Level p

To verify that DMDR is robust to varying noise levels, we evaluate its performance under two noise levels, $p = 0.2$ and $p = 0.4$. The results are shown in Figure 7, where each task reports the performance of BC and DMDR under both noise levels.

In general, DMDR consistently outperforms BC in all tasks and both noise levels. For instance, in FETCHPICK, DMDR achieves a significantly higher success rate than BC at both

TABLE VI
SUCCESS RATE ON FETCHPICK UNDER DIFFERENT NOISE TYPES.

Noise type	BC	DMDR
Unbiased noise		
Gaussian	44.38% ± 11.02%	**90.52% ± 4.83%**
Laplacian	54.00% ± 7.21%	**87.40% ± 2.70%**
Uniform	**74.80% ± 6.02%**	**72.00% ± 13.51%**
Biased noise (+0.4 offset)		
Gaussian (biased)	8.80% ± 4.32%	**76.40% ± 19.76%**
Uniform (biased)	0.00% ± 0.00%	**35.80% ± 14.24%**
Mixed-source noise		
Gaussian+Uniform	17.80% ± 12.56%	**82.20% ± 7.19%**
Gaussian+Gaussian	45.40% ± 5.03%	**89.20% ± 7.09%**

$p = 0.2$ and $p = 0.4$. A similar trend is observed in FETCHPUSH and WALKER, where increasing p leads to performance degradation for both methods, but DMDR maintains a clear advantage.

The HANDROTATE task, which operates in a high-dimensional state and action space, presents a more challenging scenario. At p = 0.2, the performance gap between DMDR and BC is smaller, as the corruption may not be enough to show a clear difference. However, at p = 0.4, DMDR achieves better performance than BC, suggesting that it remains effective even under more challenging noise conditions.

These results demonstrate that DMDR is robust not only to different types of noise but also to different levels of noise. This robustness is crucial in real-world applications, where the actual corruption rate is unknown and may vary across tasks and environments.

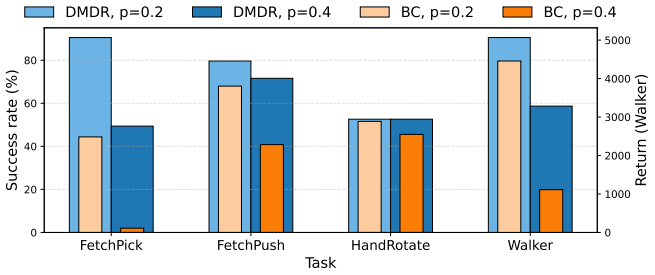


Fig. 7. Sensitivity to noise level p. Success rate for manipulation tasks; episodic return for WALKER.

V. DISCUSSION AND LIMITATION

In this paper, we propose an offline imitation learning framework (DMDR) that enables imitation learning methods to learn from noisy demonstrations. Our DMDR first filters clean samples from the demonstrations and then learns diffusion models to restore the noisy samples. The experiments show that DMDR outperforms baselines for learning from noisy demonstrations in various tasks, including robot arm manipulation, dexterous manipulation, and locomotion. Ablation studies investigate the effect of the filtering method and restoration method. We also show that our DMDR can be applied to various imitation learning algorithms to verify the effectiveness of our proposed method. Most importantly, DMDR remains robust under complex, mixed-source noise without any prior knowledge of the underlying noise distribution.

Despite this broad robustness, one limitation is that density-based anomaly detectors, such as LOF, are restricted in their performance on an edge case when the noise distribution is uniform. Leveraging temporal or model-based structures could be a worthwhile future direction.

Another potential limitation of DMDR is that the diffusion models are trained on the clean subset $D_{(\hat{s},\hat{a})}$. When this subset is small, the restored trajectories may lack diversity and fail to recover the remaining demonstrations. Transitions flagged as outliers, (s', a'), are currently *discarded*, preventing any potentially useful information from reaching the learner. A promising direction is to recycle these rejected transitions so that every sample, clean or noisy, contributes to learning and improves overall data efficiency.

Lastly, our current pipeline does not attempt to filter or correct demonstrations that are sub-optimal yet uncorrupted; extending DMDR to explicitly handle varying levels of optimality is left for future work.

REFERENCES

[1] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.
[2] B. Piot, M. Geist, and O. Pietquin, "Bridging the gap between imitation learning and inverse reinforcement learning," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 8, pp. 1814–1826, 2016.
[3] B. Zheng, S. Verma, J. Zhou, I. W. Tsang, and F. Chen, "Imitation learning: Progress, taxonomies and challenges," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.
[4] T. Zhang, L. Yue, C. Wang, J. Sun, S. Zhang, A. Wei, and G. Xie, "Leveraging imitation learning on pose regulation problem of a robotic fish," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 3, pp. 4232–4245, 2022.
[5] A. Vahabpour, T. Wang, Q. Lu, O. Pooladzandi, and V. Roychowdhury, "Diverse imitation learning via self-organizing generative models," *IEEE Transactions on Neural Networks and Learning Systems*, 2024.
[6] Z. Cheng, L. Liu, A. Liu, H. Sun, M. Fang, and D. Tao, "On the guaranteed almost equivalence between imitation learning from observation and demonstration," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 2, pp. 677–689, 2021.
[7] A. W. A. Ali, F. A. A. Razak, and N. Hayima, "A review on the ac servo motor control systems," *ELEKTRIKA-Journal of Electrical Engineering*, vol. 19, no. 2, pp. 22–39, 2020.
[8] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.
[9] M. Zare, P. M. Kebria, A. Khosravi, and S. Nahavandi, "A survey of imitation learning: Algorithms, recent developments, and challenges," *IEEE Transactions on Cybernetics*, 2024.
[10] S.-F. Chen, H.-C. Wang, M.-H. Hsu, C.-M. Lai, and S.-H. Sun, "Diffusion model-augmented behavioral cloning," in *International Conference on Machine Learning*, 2024.
[11] C.-M. Lai, H.-C. Wang, P.-C. Hsieh, Y.-C. F. Wang, M.-H. Chen, and S.-H. Sun, "Diffusion-reward adversarial imitation learning," in *Neural Information Processing Systems*, 2024.
[12] B.-R. Huang, C.-K. Yang, C.-M. Lai, D.-J. Wu, and S.-H. Sun, "Diffusion imitation from observation," in *Neural Information Processing Systems*, 2024.
[13] C.-H. Yeh, T.-S. Nan, R. Vuorio, W. Hung, H. Y. Wu, S.-H. Sun, and P.-C. Hsieh, "Action-constrained imitation learning," in *International Conference on Machine Learning*, 2025.
[14] B. Fang, S. Jia, D. Guo, M. Xu, S. Wen, and F. Sun, "Survey of imitation learning for robotic manipulation," *International Journal of Intelligent Robotics and Applications*, vol. 3, pp. 362–369, 2019.
[15] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 4693–4700.

[16] L. Le Mero, D. Yi, M. Dianati, and A. Mouzakitis, "A survey on imitation learning techniques for end-to-end autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 14 128–14 147, 2022.

[17] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 661–668.

[18] J. Harmer, L. Gisslén, J. del Val, H. Holst, J. Bergdahl, T. Olsson, K. Sjöö, and M. Nordin, "Imitation learning with concurrent actions in 3d games," in *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2018, pp. 1–8.

[19] F. F. Duarte, N. Lau, A. Pereira, and L. P. Reis, "A survey of planning and learning in games," *Applied Sciences*, vol. 10, no. 13, p. 4529, 2020.

[20] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.

[21] D. Hadfield-Menell, S. Milli, P. Abbeel, S. J. Russell, and A. Dragan, "Inverse reward design," *Advances in neural information processing systems*, vol. 30, 2017.

[22] S.-M. Yang and S.-J. Ke, "Performance evaluation of a velocity observer for accurate velocity estimation of servo motor drives," *IEEE Transactions on Industry Applications*, vol. 36, no. 1, pp. 98–104, 2000.

[23] V. G. Biju, A.-M. Schmitt, and B. Engelmann, "Assessing the influence of sensor-induced noise on machine-learning-based changeover detection in cnc machines," *Sensors*, vol. 24, no. 2, p. 330, 2024.

[24] K. Smeds and X. Lu, "Effect of sampling jitter and control jitter on positioning error in motion control systems," *Precision Engineering*, vol. 36, no. 2, pp. 175–192, 2012.

[25] D. Gao, S. Wang, Y. Yang, H. Zhang, H. Chen, X. Mei, S. Chen, and J. Qiu, "An intelligent control method for servo motor based on reinforcement learning," *Algorithms*, vol. 17, no. 1, p. 14, 2023.

[26] Y.-H. Wu, N. Charoenphakdee, H. Bao, V. Tangkaratt, and M. Sugiyama, "Imitation learning from imperfect demonstration," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6818–6827.

[27] Y. Wang, C. Xu, B. Du, and H. Lee, "Learning to weight imperfect demonstrations," in *International Conference on Machine Learning*. PMLR, 2021, pp. 10 961–10 970.

[28] F. Sasaki and R. Yamashina, "Behavioral cloning from noisy demonstrations," in *International Conference on Learning Representations*, 2020.

[29] H. Chung, B. Sim, D. Ryu, and J. C. Ye, "Improving diffusion models for inverse problems using manifold constraints," *Advances in Neural Information Processing Systems*, vol. 35, pp. 25 683–25 696, 2022.

[30] B. Kawar, M. Elad, S. Ermon, and J. Song, "Denoising diffusion restoration models," *Advances in Neural Information Processing Systems*, 2022.

[31] N. Murata, K. Saito, C.-H. Lai, Y. Takida, T. Uesaka, Y. Mitsufuji, and S. Ermon, "GibbsDDRM: A partially collapsed Gibbs sampler for solving blind inverse problems with denoising diffusion restoration," in *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[32] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 93–104.

[33] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in neural information processing systems*, 1989.

[34] M. Bain and C. Sammut, "A framework for behavioural cloning." in *Machine Intelligence 15*, 1995, pp. 103–129.

[35] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson, "Implicit behavioral cloning," in *Conference on Robot Learning*. PMLR, 2022, pp. 158–168.

[36] T. Pearce, T. Rashid, A. Kanervisto, D. Bignell, M. Sun, R. Georgescu, S. V. Macua, S. Z. Tan, I. Momennejad, K. Hofmann, and S. Devlin, "Imitating human behaviour with diffusion models," in *International conference on learning representations*, 2023.

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

[38] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, 2016.

[39] F. Torabi, G. Warnell, and P. Stone, "Generative adversarial imitation from observation," *arXiv preprint arXiv:1807.06158*, 2018.

[40] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adverserial inverse reinforcement learning," in *International Conference on Learning Representations*, 2018.

[41] Y. Lee, A. Szot, S.-H. Sun, and J. J. Lim, "Generalizable imitation learning from observation via inferring goal proximity," *Advances in neural information processing systems*, vol. 34, pp. 16 118–16 130, 2021.

[42] M. Jing, X. Ma, W. Huang, F. Sun, C. Yang, B. Fang, and H. Liu, "Reinforcement learning from imperfect demonstrations under soft expert guidance," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 04, 2020, pp. 5109–5116.

[43] H. Liu, Z. Huang, J. Wu, and C. Lv, "Improved deep reinforcement learning with expert demonstrations for urban autonomous driving," in *2022 IEEE intelligent vehicles symposium (IV)*. IEEE, 2022, pp. 921–928.

[44] J. Ramírez, W. Yu, and A. Perrusquía, "Model-free reinforcement learning from expert demonstrations: a survey," *Artificial Intelligence Review*, vol. 55, no. 4, pp. 3213–3241, 2022.

[45] D. Michie, M. Bain, and J. Hayes-Miches, "Cognitive models from subcognitive skills," *IEE control engineering series*, vol. 44, pp. 71–99, 1990.

[46] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 7 2018, pp. 4950–4957.

[47] Q. Xiao, B. Niu, Y. Tan, Z. Yang, and X. Chen, "Generative upper-level policy imitation learning with pareto-improvement for energy-efficient advanced machining systems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 36, no. 3, pp. 5190–5203, 2025.

[48] D. Brown, W. Goo, P. Nagarajan, and S. Niekum, "Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations," in *International conference on machine learning*. PMLR, 2019, pp. 783–792.

[49] D. S. Brown, W. Goo, and S. Niekum, "Better-than-demonstrator imitation learning via automatically-ranked demonstrations," in *Conference on robot learning*. PMLR, 2020, pp. 330–359.

[50] V. Tangkaratt, N. Charoenphakdee, and M. Sugiyama, "Robust imitation learning from noisy demonstrations," in *International Conference on Artificial Intelligence and Statistics*, 2020.

[51] G.-H. Kim, S. Seo, J. Lee, W. Jeon, H. Hwang, H. Yang, and K.-E. Kim, "Demodice: Offline imitation learning with supplementary imperfect demonstrations," in *International Conference on Learning Representations*, 2021.

[52] L. Yu, T. Yu, J. Song, W. Neiswanger, and S. Ermon, "Offline imitation learning with suboptimal demonstrations via relaxed distribution matching," in *Proceedings of the AAAI conference on artificial intelligence*, 2023, pp. 11 016–11 024.

[53] H. Xu, X. Zhan, H. Yin, and H. Qin, "Discriminator-weighted offline imitation learning from suboptimal demonstrations," in *International Conference on Machine Learning*. PMLR, 2022, pp. 24 725–24 742.

[54] W. Zhang, H. Xu, H. Niu, P. Cheng, M. Li, H. Zhang, G. Zhou, and X. Zhan, "Discriminator-guided model-based offline imitation learning," in *Conference on Robot Learning*. PMLR, 2023, pp. 1266–1276.

[55] Y. Li, Z. Chen, D. Zha, K. Zhou, H. Jin, H. Chen, and X. Hu, "Automated anomaly detection via curiosity-guided search and self-imitation learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 6, pp. 2365–2377, 2021.

[56] Y. Zhu, Y. Lai, K. Zhao, X. Luo, M. Yuan, J. Wu, J. Ren, and K. Zhou, "From bi-level to one-level: A framework for structural attacks to graph anomaly detection," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 36, no. 4, pp. 6174–6187, 2025.

[57] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *International conference on machine learning*. PMLR, 2018, pp. 4393–4402.

[58] L. Ruff, R. A. Vandermeulen, N. Görnitz, A. Binder, E. Müller, K.-R. Müller, and M. Kloft, "Deep semi-supervised anomaly detection," *arXiv preprint arXiv:1906.02694*, 2019.

[59] Z. Wang *et al.*, "Inductive multi-view semi-supervised anomaly detection via probabilistic modeling," in *2019 IEEE International Conference on Big Knowledge (ICBK)*. IEEE, 2019.

[60] D. Hendrycks, M. Mazeika, and T. Dietterich, "Deep anomaly detection with outlier exposure," *arXiv preprint arXiv:1812.04606*, 2018.

[61] K. Sohn, C.-L. Li, J. Yoon, M. Jin, and T. Pfister, "Learning and evaluating representations for deep one-class classification," *arXiv preprint arXiv:2011.02578*, 2020.

[62] C.-L. Li, K. Sohn, J. Yoon, and T. Pfister, "Cutpaste: Self-supervised learning for anomaly detection and localization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9664–9674.

[63] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *International conference on learning representations*, 2018.

[64] D. Gong, L. Liu, V. Le, B. Saha, M. R. Mansour, S. Venkatesh, and A. v. d. Hengel, "Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019.

[65] H. Park, J. Noh, and B. Ham, "Learning memory-guided normality for anomaly detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020.

[66] D. S. Tan, Y.-C. Chen, T. P.-C. Chen, and W.-C. Chen, "Trustmae: A noise-resilient defect classification framework using memory-augmented auto-encoders with trust regions," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2021.

[67] X. Kong, W. Zhang, H. Wang, M. Hou, X. Chen, X. Yan, and S. K. Das, "Federated graph anomaly detection via contrastive self-supervised learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 36, no. 5, pp. 7931–7944, 2025.

[68] Z. Huang, B. Zhang, G. Hu, L. Li, Y. Xu, and Y. Jin, "Enhancing unsupervised anomaly detection with score-guided network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 10, pp. 14 754–14 769, 2024.

[69] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," in *International conference on machine learning*. PMLR, 2018, pp. 4334–4343.

[70] J. Li, R. Socher, and S. C. Hoi, "Dividemix: Learning with noisy labels as semi-supervised learning," in *International Conference on Learning Representations*, 2020.

[71] N. Karim, M. N. Rizve, N. Rahnavard, A. Mian, and M. Shah, "Unicon: Combating label noise through uniform selection and contrastive learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 9676–9686.

[72] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[73] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet, "Video diffusion models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 8633–8646, 2022.

[74] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[75] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *International Conference on Learning Representations*, 2021.

[76] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever, "Consistency models," in *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[77] Z. Wang, J. J. Hunt, and M. Zhou, "Diffusion policies as an expressive policy class for offline reinforcement learning," in *The Eleventh International Conference on Learning Representations*, 2023.

[78] B. B. Moser, A. S. Shanbhag, F. Raue, S. Frolov, S. Palacio, and A. Dengel, "Diffusion models, image super-resolution, and everything: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[79] H. Zhao, K. Q. Lin, R. Yan, and Z. Li, "Diffusionvmr: Diffusion model for joint video moment retrieval and highlight detection," *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[80] R. Chen, J. Fan, M. Wu, R. Cheng, and J. Song, "G-diff: A graph-based decoding network for diffusion recommender model," *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[81] H. Huang, L. Sun, B. Du, and W. Lv, "Learning joint 2-d and 3-d graph diffusion models for complete molecule generation," *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[82] S. Lee, S. Heo, and S. Lee, "Dmesh: A structure-preserving diffusion model for 3-d mesh denoising," *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[83] L. Pei, Y. Cao, Y. Kang, Z. Xu, and Q. Liu, "Spatiotemporal imputation of traffic emissions with self-supervised diffusion model," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2024.

[84] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[85] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.

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

[87] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, "Isaac gym: High performance gpu-based physics simulation for robot learning," 2021.

[88] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth ieee international conference on data mining*. IEEE, 2008, pp. 413–422.

[89] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *International conference on machine learning*. PMLR, 2021, pp. 8162–8171.

[90] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," *arXiv preprint arXiv:2011.13456*, 2020.

[91] J. Gabela, A. Kealy, S. Li, M. Hedley, W. Moran, W. Ni, and S. Williams, "The effect of linear approximation and gaussian noise assumption in multi-sensor positioning through experimental evaluation," *IEEE Sensors Journal*, vol. 19, no. 22, pp. 10 719–10 727, 2019.

[92] M. El Helou and S. Süsstrunk, "Blind universal bayesian image denoising with gaussian noise level learning," *IEEE Transactions on Image Processing*, vol. 29, pp. 4885–4897, 2020.

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

[94] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

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

[96] S. Zhang, Z. Cao, D. Sadigh, and Y. Sui, "Confidence-aware imitation learning from demonstrations with varying optimality," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 340–12 350, 2021.

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

[98] G. O. Campos and et al., "On the evaluation of unsupervised outlier detection," *Data Mining and Knowledge Discovery*, 2016.

[99] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International conference on learning representations*, 2015.

[100] K. Brantley, W. Sun, and M. Henaff, "Disagreement-regularized imitation learning," in *International Conference on Learning Representations*, 2019.

SUPPLEMENTARY MATERIALS

VI. ALGORITHM

Our proposed framework DMDR consists of two stages. (1) **Demonstration filtering**: We learn the global features of the expert distribution with a pair of autoencoders ϕ_s and ϕ_a, and then apply Local Outlier Factor (LOF) to filter clean samples based on local densities. The algorithm for demonstration filtering is detailed in Algorithm 1. (2) **Demonstration restoration**: The conditional diffusion models θ_s and θ_a learn to restore state/action while the noise timestep predictors ψ_s and ψ_a aim to predict the noise timestep of the noisy state s' and actions a' for restoration. The algorithm for demonstration restoration is detailed in Algorithm 2.

VII. TRAINING DETAILS

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

A. Model Architecture

We elaborate the model architecture for each component of our DMDR in Table VII. For fair comparisons, the policy architectures of baselines used in this paper follow identical designs.

B. Hyperparamters

We report the hyperparameters of learning the component used for all the methods on all the tasks in Table VIII. We use the Adam optimizer [99] and a batch size of 128 for all the methods on all the tasks. Also, we use linear learning rate decay for all policy models. The number of diffusion steps is 100 for conditional diffusion models, noise predictors, and Diffusion Policies.

C. Computation Resource

We conducted all the experiments on the following three workstations:

- M1: ASUS WS880T workstation with an Intel Xeon W-2255 (10C/20T, 19.3MiB, 3.70GHz) 20-core CPU, 125GB memory, two NVIDIA GeForce RTX 3080 Ti GPUs
- M2: ASUS WS880T workstation with an Intel Xeon W-2255 (10C/20T, 19.3M, 3.7GHz) 20-core CPU, 64GB memory, two NVIDIA RTX 4070 Ti GPUs

Algorithm 1 Demonstration Filtering

Input: Noisy demonstration D
Output: Subsets of demonstration $D_{(\hat{s},\hat{a})}$, $D_{(s',a')}$, $D_{(\hat{s},a')}$, and $D_{(s',\hat{a})}$
// Training
Randomly initialize a pair of autoencoders ϕ_s and ϕ_a
for each autoencoder training iteration **do**
  Sample (s,a) from D
  Update ϕ_s using L^s_{rec} from Eq. 1
  Update ϕ_a using L^a_{rec} from Eq. 2
end for
// Inference
for each state-action pair (s,a) in D **do**
  Calculate the state feature z_s with ϕ_s
  Calculate the action feature z_a with ϕ_a
end for
Apply LOF on $\{z_s\}$ and $\{z_a\}$ separately
Initialize empty sets $D_{(\hat{s},\hat{a})}$, $D_{(s',a')}$, $D_{(\hat{s},a')}$, and $D_{(s',\hat{a})}$
for each state-action pair (s,a) in D **do**
  Label state s based on LOF score from $\{z_s\}$
  Label action a based on LOF score from $\{z_a\}$
  Append (s,a) to the subset with state/action labels
end for
return $D_{(\hat{s},\hat{a})}$, $D_{(s',a')}$, $D_{(\hat{s},a')}$, and $D_{(s',\hat{a})}$

Algorithm 2 Demonstration Restoration

Input: Demonstration subsets $D_{(\hat{s},\hat{a})}$, $D_{(s',\hat{a})}$ and $D_{(\hat{s},a')}$
Output: $D_{(\hat{s},\hat{a})}$ with restored state-action pairs
// Training
Randomly initialize diffusion models θ_s and θ_a
for each diffusion model training iteration **do**
  Sample $(\hat{s},\hat{a})$ from $D_{(\hat{s},\hat{a})}$
  Update θ_s using L^s_{diff} from Eq. 4
  Update θ_a using L^a_{diff} from Eq. 5
end for
for each noise predictor training iteration **do**
  Sample $(\hat{s},\hat{a})$ from $D_{(\hat{s},\hat{a})}$
  Update ψ_s using L^s_{pred} from Eq. 6
  Update ψ_a using L^a_{pred} from Eq. 7
end for
// Inference
for each state-action pair $(s',\hat{a})$ in $D_{(s',\hat{a})}$ **do**
  Calculate the predicted noise timestep t' with ψ_s
  if $t' < t_{\text{thres}}$ **then**
    Append $(s',\hat{a})$ to $D_{(\hat{s},\hat{a})}$
  else
    Restore the noisy state with θ_s and output s^*
    Append $(s^*,\hat{a})$ to $D_{(\hat{s},\hat{a})}$
  end if
end for
for each state-action pair $(\hat{s},a')$ in $D_{(\hat{s},a')}$ **do**
  Calculate the predicted noise timestep t' with ψ_a
  if $t' < t_{\text{thres}}$ **then**
    Append $(\hat{s},a')$ to $D_{(\hat{s},\hat{a})}$
  else
    Restore the noisy state with θ_a and output a^*
    Append $(\hat{s},a^*)$ to $D_{(\hat{s},\hat{a})}$
  end if
end for
return $D_{(\hat{s},\hat{a})}$

TABLE VII
MODEL ARCHITECTURES

Model	Component	FETCHPICK	FETCHPUSH	HANDROTATE	WALKER
State/Action Autoencoder	Input Dim.	16/4	16/4	68/20	17/6
	Hidden Dim.	[512, 128, 32, 128, 512] for all environments			
	Output Dim.	16/4	16/4	68/20	17/6
State/Action Diffusion Model	# Layers	5	5	5	5
	Input Dim.	20	20	88	23
	Hidden Dim.	1024	1024	1024	1024
	Output Dim.	16/4	16/4	68/20	17/6
Noise Predictor	# Layers	5	5	5	5
	Input Dim.	20	20	88	23
	Hidden Dim.	1024	1024	1024	1024
	Output Dim.	1	1	1	1
Policy	# Layers	4	4	4	4
	Input Dim.	16	16	68	17
	Hidden Dim.	512	512	512	512
	Output Dim.	4	4	20	6

TABLE VIII
HYPERPARAMETERS

Method	Hyperparameter	FETCHPICK	FETCHPUSH	HANDROTATE	WALKER
BC	Batch Size	128	128	128	128
	Learning Rate	2e-6	5e-7	5e-5	2e-5
	# Epochs	2000	2000	2000	2000
BCND [28]	Batch Size	128	128	128	128
	Learning Rate	1e-6	2e-6	5e-5	5e-6
	# Epochs	2000	2000	2000	2000
DMDR (Ours)	**Demonstration Filtering**				
	Autoencoder Learning Rate	1e-6	1e-6	1e-6	1e-6
	Autoencoder # Epochs	500	2000	2000	500
	LOF # Neighbors	50	50	10	50
	Demonstration Restoration				
	Diffusion Model Learning Rate	1e-4	1e-4	1e-4	1e-4
	Diffusion Model # Epochs	8000	8000	8000	8000
	Noise Predictor Learning Rate	1e-6	1e-6	1e-6	1e-6
	Noise Predictor # Epochs	2000	2000	2000	2000
	t_{thres}	20	20	20	20
	Policy Learning				
	Learning Rate	5e-6	1e-6	5e-5	2e-5
	# Epochs	2000	2000	2000	2000

VIII. ABLATION STUDY FOR ENSEMBLE METHODS

Ensemble policies learn n policy variants and aggregate the predictions. To improve the robustness using ensemble techniques, randomness among the policies is essential for learning distinct features. A widely adopted method is to sample different training data for each policy. We use BC policies to evaluate three ways to sample the training data that introduce randomness for ensemble techniques.

- **Split.** In [28], they split the demonstration dataset into n parts. Each policy learns from a distinct and disjoint subset of the demonstration data. The method provides diversity in the ensemble with the cost of fewer training data for each policy.
- **Sample with replacement.** In [100], they sample data with replacement from the demonstration for each policy. The number of training data for each policy is identical, but the content of training data is different due to replacement after sampling.
- **Shuffle.** All n policies are trained using the entire demonstration dataset, where the randomness is introduced by shuffling the training data. Policies can capture various features by learning from training data with different orders.

In Table IX, we adopt 3 and 5 policies for the above ensemble approaches and compare them with a single policy baseline. In general, the performances improve when the number of policies increases, except for the Split ensemble method. This could be a result of the lack of training data for each policy since the number of samples decreases as the number of policies increases. We observe that the Shuffle

TABLE IX
ENSEMBLE METHOD ABLATION STUDY

Method	# of policies	FETCHPICK	FETCHPUSH	HANDROTATE	WALKER
Single policy baseline	1	44.38% ± 11.02%	67.97% ± 4.81%	45.56% ± 6.15%	4456.8 ± 1051.1
Split	3	45.99% ± 11.14%	69.48% ± 4.78%	40.14% ± 4.30%	**5232.6** ± 920.1
	5	30.63% ± 6.82%	62.98% ± 5.24%	31.99% ± 5.90%	4672.5 ± 747.7
Sample with replacement	3	**51.49**% ± 7.90%	68.94% ± 2.86%	49.16% ± 3.68%	3750.0 ± 1342.7
	5	48.21% ± 8.09%	71.09% ± 3.91%	49.59% ± 5.07%	4929.5 ± 640.6
Shuffle	3	50.49% ± 9.49%	70.11% ± 3.32%	49.34% ± 3.88%	4952.1 ± 637.0
	5	**51.36**% ± 6.67%	**72.25**% ± 3.20%	**51.03**% ± 4.33%	**5170.6** ± 395.2

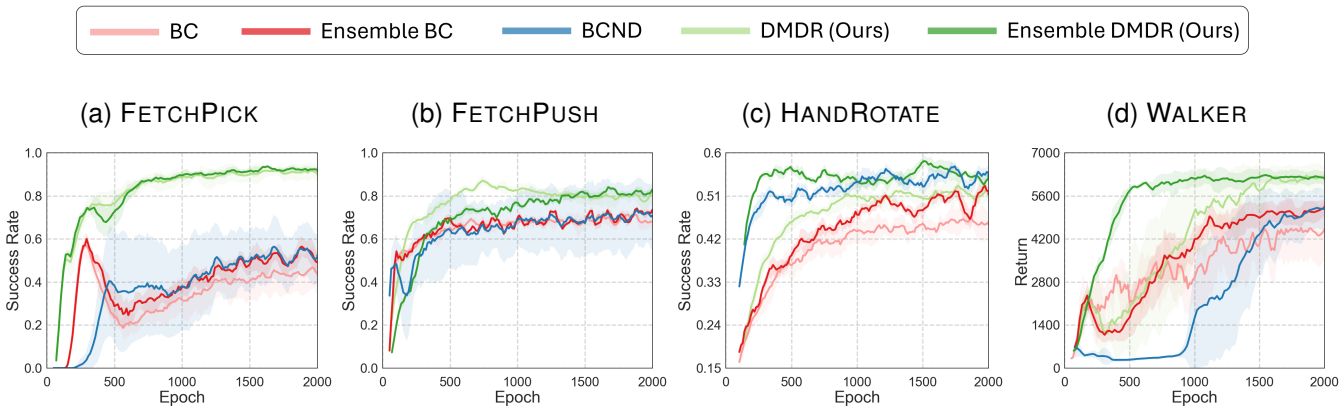


Fig. 8. **Training progress.** We observe that DMDR is more robust and stable during the training, resulting in a lower standard deviation. In contrast, the training of BC is more easily affected by noisy samples, which can be observed on (a) FETCHPICK and (d) WALKER.

ensemble method with 5 policies performs the best on all the tasks. As the Shuffle ensemble method utilizes every accessible training data for each policy, we can infer that the number of training samples is an important factor for these imitation learning tasks. Therefore, we use the Shuffle ensemble method with 5 policies for Ensemble BC, BCND, and Ensemble DMDR in the Table I of our main paper. The above results also motivate the employment of demonstration restoration, which aims to correct and utilize as many noisy samples as possible.

IX. ENVIRONMENTS COMPLEXITIES

In this section, we aim to quantitatively evaluate the complexities of the environments used in this paper. We test the multimodality of each environment since it is more challenging to learn an effective policy if the expert trajectories contain multimodal actions. In imitation learning, multimodality can stem from the nature of the task (e.g., different goals executed in arbitrary orders) or variations in expert demonstrations (e.g., achieving the same goal through different paths). For each environment, we create 10 clusters of states and 10 clusters of actions from expert demonstrations. Then, we measure if the states in the same cluster share actions from the same clusters. Specifically, we determine the predominant action class for each state cluster and compute the proportion of states belonging to that class. This ratio, ranging from 0.1 to 1, indicates whether actions within a state cluster are randomly distributed (1/10) or consistently the same (1/1).

The results of each environment are reported below:

- FETCHPICK: 0.6286
- FETCHPUSH: 0.5482
- HANDROTATE: 0.3035
- WALKER: 0.4240

We observe that robot arm manipulation (FETCHPICK and FETCHPUSH) tasks result in higher majority ratios, which indicates that the expert behaviors are more unimodal. On the other hand, dexterous manipulation (HANDROTATE) tasks and locomotion (WALKER) result in lower majority ratios, which means the demonstrations contain more multimodal paths or behaviors for similar goals. This experiment shows that the environments in this paper exhibit varying degrees of multimodal actions and that the proposed DMDR effectively improves the baseline performance across all environments.

X. TRAINING PROGRESS

We illustrate the training progress of all methods in Figure 8. We observe that DMDR is more robust and stable during the training, resulting in a lower standard deviation. In contrast, BC encounters a performance drop when the training progress is affected by the noisy samples, which can be observed on FETCHPICK and WALKER.