Adapting Diffusion Policies to Human Preferences via Reward-Guided Fine-Tuning

Yuxin Chen[1†], Devesh K. Jha[2], Masayoshi Tomizuka[1], Diego Romeres[2]

***Abstract*— Diffusion policies have recently achieved impressive results in robotic manipulation tasks. However, their rigid reliance on demonstration data makes it difficult to adapt behavior to evolving user preferences or dynamic deployment environments. We propose FDPP (Fine-tuning Diffusion Policy with Human Preference), a simple yet effective method that leverages human preference labels to train a reward function, which is then used to fine-tune pre-trained diffusion policies via reinforcement learning. This approach allows robots to align with new task constraints or personalized objectives while retaining core task competence. We further incorporate Kullback–Leibler (KL) regularization during fine-tuning to prevent overfitting and preserve the original policy distribution. Experiments on diverse robotic tasks demonstrate that FDPP successfully reshapes policy behavior in alignment with human intent without sacrificing task success.**

I. Introduction

Imitation learning from human demonstrations is a powerful method for training robots to perform a wide range of manipulation tasks, such as grasping [1], [2], [3], dexterous manipulation [4], [5], and legged locomotion [6]. Recently, the rapid advancement of generative models has highlighted their remarkable ability to synthesize complex, high-dimensional distributions, offering new opportunities for enhancing policy learning [3]. Among these, diffusion models, a type of generative model that gradually transform random noise into a data sample, have been applied in imitation learning for robotics. These models, referred to as *diffusion policies*, have achieved state-of-the-art performance by leveraging the powerful generative modeling capabilities [3], [7].

However, diffusion policies share common challenges with general imitation learning methods. For example, training a robust diffusion policy [3] for a particular task typically requires 100 to 200 human-collected demonstrations, making the process both time-consuming and sample-inefficient. Additionally, these policies are task-specific, necessitating a new set of demonstrations for each task. The environmental setup during demonstration must also closely resemble the deployment environment in terms of viewpoint, object appearance, and action space [7]. Nevertheless, during real-world deployment, it is common to encounter additional constraints (e.g., avoiding undesired regions during movement) or preferences (e.g., aligning blocks rather than unstable stacking during a block stacking task) that differ from the pre-collected demonstrations. This mismatch between the policy's learned behavior and the task requirements creates a significant challenge. Therefore, effectively adapting a pre-trained diffusion policy to new environments is essential for successful real-world deployment.

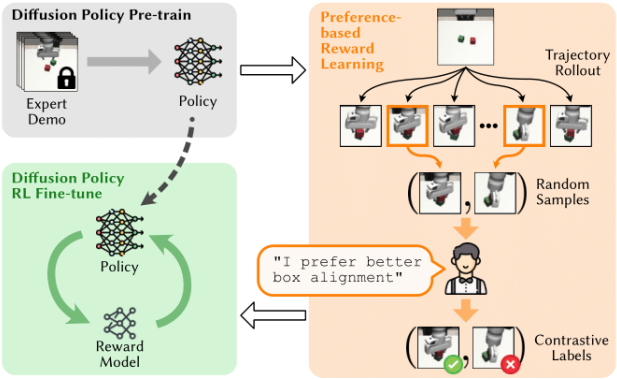


Fig. 1: **Fine-tune Diffusion Policy with Human Preference.** Given a pre-trained diffusion policy, **FDPP** collects trajectory roll-outs and queries human feedback to label pairs of randomly sampled image observations based on human preferences or task specifications. Using these labels, a reward function is trained through preference-based reward learning, which is then used to fine-tune the diffusion policy via reinforcement learning.

Motivated by this challenging problem, we propose fine-tuning the diffusion policy using online reinforcement learning (RL) to better align with human preferences and task specifications. We introduce **F**ine-tuning **D**iffusion **P**olicy with Human **P**reference (**FDPP**), a straightforward yet effective algorithm that learns reward functions through preference-based learning with human labels. The pre-trained policy is then fine-tuned using the learned reward function via RL. An overview of these steps is illustrated in Fig. 1. We evaluate **FDPP** on a variety of robotic tasks with differing preferences and find that it effectively adapts the behavioral distribution of the pre-trained diffusion policy to align with human preferences, without compromising performance on the original task.

In summary, we make the following contributions:

- We propose **FDPP**, a method that fine-tune the pre-trained diffusion policy to align with human preference.
- We investigate how incorporating Kullback–Leibler (KL) regularization into diffusion policy fine-tuning can effectively prevent over-fitting to the reward while preserving the original performance of the pre-trained diffusion policy on the original tasks.

We conduct empirical evaluations of **FDPP** across various

[1]Mechanical Systems Control Lab, UC Berkeley, Berkeley, CA, USA {yuxinc, tomizuka}@berkeley.edu
[2]Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, USA {jha, romeres}@merl.com
[†]Work done during MERL internship.

robotic tasks and preferences, highlighting these contributions in Sec. V.

II. Related Work

A. Diffusion Policy

Diffusion policy, initially introduced by Chi et al. [3], represents a significant advancement in imitation learning by leveraging generative models, specifically diffusion models [8], [9], [10], to replicate complex behaviors from demonstrations. This approach takes in the most recent observations, either the low-dimensional state representations or high-dimensional images, and outputs a sequence of future actions over a prediction horizon. It has achieved state-of-the-art results in various robotic tasks [11], [12]. Building on this, Ze et al. [7] developed the 3D diffusion policy, which incorporates 3D visual representations (such as 3D point clouds) to enhance generalizability and effectiveness compared to the original diffusion policy. However, despite their success, both diffusion policies and their variants share limitations typical of traditional imitation learning, such as the need for large amounts of task-specific demonstrations and sensitivity to environmental changes between training and deployment. Compared to this line of works, we integrate RL fine-tuning to expand the applicability of diffusion policies in practical scenarios, where robustness and adaptability are essential.

B. Preference-based Reward Learning

In the past, demonstrations have been the preferred method for reward learning. A popular paradigm is inverse reinforcement learning (IRL) [13], where a reward function is extracted to capture why specific behaviors may be desirable from the demonstration. Recently, there is a growing trend toward using preference-based learning [14], [15], [16], [17], [18], [19], [20], [21]. In the proposed approach, humans are asked to compare two (or more) trajectories (or states) and provide labels, allowing the model to infer a mapping from these ranked trajectories to a scalar reward. In general, human preferences and rankings of robot trajectories are easier for people to provide than kinesthetic teaching or detailed feedback [21], [22]. We leverage these approaches to derive a reward function aligned with human preferences.

C. RL-based Fine-tuning of Diffusion Models

There are multiple strategies for fine-tuning diffusion models. Fan and Lee [23] are the first to introduce the idea of fine-tuning pre-trained diffusion models by combining policy gradients with generative adversarial networks (GANs) [24]. In their approach, policy gradient updates are guided by reward signals from the discriminator of GAN to refine the diffusion model. They demonstrate that this fine-tuning method enables the model to generate realistic samples with fewer diffusion steps, particularly when using denoising diffusion probabilistic models (DDPMs) [8] sampling in simpler domains. More recently, Black et al. [25] and Fan et al. [26] have proposed fine-tuning text-to-image diffusion models using RL. Both studies treat the fine-tuning process as a multi-step decision-making problem and show that RL-based fine-tuning can surpass supervised fine-tuning methods that rely on reward-weighted loss [27]. Fan et al. further provide a detailed analysis of KL-regularization for both supervised and RL-based fine-tuning, supported by theoretical justifications. In contrast to previous work, which primarily focuses on text-to-image diffusion models, our approach extends the application to diffusion policies in the context of robotic tasks.

Ren et al. introduce DPPO in a concurrent study [28] focusing on fine-tuning a diffusion policy using the policy gradient method to enhance training stability and policy robustness. The reward for fine-tuning remains tied to the original task objective. In contrast, our method fine-tunes the policy using a new reward derived from human preferences, which may differ from the original task objective. This necessitates the use of KL regularization to prevent over-fitting and preserve the original performance of diffusion policy. Furthermore, we demonstrate how to use a preference-based reward model with contrastive labels to obtain this reward function from human preference, effectively aligning the policy with user expectations.

III. Preliminaries

In this section, we present a concise overview of the RL problem formulation (Sec. III-A) and the diffusion policy framework (Sec. III-B).

A. Markov Decision Process and Reinforcement Learning

A Markov Decision Process (MDP) defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, p, \rho_0)$, where $\mathcal{S} \in \mathbb{R}^S$ represents the state space, $\mathcal{A} \in \mathbb{R}^A$ is the action space, $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function, $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, \infty)$ defines the probability density of the next state $\mathbf{s}_{t+1} \in \mathcal{S}$, given the current state $\mathbf{s}_t \in \mathcal{S}$ and action $\mathbf{a}_t \in \mathcal{A}$. The initial state distribution is denoted by ρ_0. At each time step t, the robot agent observes the state $\mathbf{s}_t$, selects an action $\mathbf{a}_t$, receives a reward $r(\mathbf{s}_t, \mathbf{a}_t)$, and transitions to the next state $\mathbf{s}_{t+1}$ following the transition probability $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$.

With a given policy $\pi_\theta(\mathbf{a}|\mathbf{s})$, parameterized by θ, and the initial state $\mathbf{s}_0 \sim \rho_0$, the robot agent generates a trajectory, which is a sequence of state-action pairs, $\xi = \{(\mathbf{s}_0, \mathbf{a}_0), (\mathbf{s}_1, \mathbf{a}_1), \ldots, (\mathbf{s}_T, \mathbf{a}_T)\}$. The objective of reinforcement learning (RL) is to maximize the expected cumulative reward over trajectories sampled from the policy $\xi \sim p(\xi|\pi_\theta)$:

$$\mathcal{J}_{\mathrm{RL}}(\pi_\theta) = \mathbb{E}_\xi \left[\sum_{t=0}^{T} r(\mathbf{s}_t, \mathbf{a}_t) \right]. \tag{1}$$

There are various methods to train the policy in RL, with one popular approach being *policy gradient* algorithms [29], such as REINFORCE [30]. These methods update the policy parameters θ in the direction of the objective gradient:

$$\nabla_\theta \mathcal{J}_{\mathrm{RL}}(\pi_\theta) = \mathbb{E}_\xi \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) \, Q^{\pi_\theta}(\mathbf{s}_t, \mathbf{a}_t) \right], \tag{2}$$

where Q^{π_θ} is the state-action value function (also known as the Q-function) estimator [31].

B. Diffusion Model and Diffusion Policy

Denoising diffusion probabilistic models (DDPM) [8], [9] are used to model the distribution of a dataset of samples, $\mathbf{x}^0$, conditioned on some context $\mathbf{c}$, represented as $\mathbf{x}^0 \sim p\left(\mathbf{x}^0|\mathbf{c}\right)$, where $\mathbf{x}^0 \in \mathbb{R}^n$. This conditional distribution is learned by modeling the reverse denoising process of a Markovian *forward process* $q\left(\mathbf{x}^k|\mathbf{x}^{k-1}\right)$, which progressively adds Gaussian noise to the data samples over time.

The *reverse process* $p\left(\mathbf{x}^{k-1}|\mathbf{x}^k, \mathbf{c}\right)$ is designed to recover the original, noise-free sample $\mathbf{x}^0$ from an initial Gaussian noise $\mathbf{x}^K \sim \mathcal{N}(0, \mathbf{I})$ through K iterations of denoising. This process generates a series of intermediate samples with progressively less noise, denoted as $\{\mathbf{x}^K, \mathbf{x}^{K-1}, \ldots, \mathbf{x}^0\}$. Specifically, the reverse process is defined as

$$p\left(\mathbf{x}^{k-1}|\mathbf{x}^k, \mathbf{c}\right) = \mathcal{N}\left(\mathbf{x}^{k-1}; \mu_\theta\left(\mathbf{x}^k, \mathbf{c}, k\right), \sigma_k^2 \mathbf{I}\right), \quad (3)$$

where μ_θ is a neural network parameterized by θ that predicts the added noise at each iteration, and σ_k represents the step-dependent variance governed by a variance schedule.

The noise predictor μ_θ is trained with the following objective:

$$\mathcal{L}_{\text{DM}}(\theta) = \mathbb{E}_{(\mathbf{x}^0, \mathbf{c}, \mathbf{x}^k, k)} \|\bar{\mu}\left(\mathbf{x}^0, k\right) - \mu_\theta\left(\mathbf{x}^k, \mathbf{c}, k\right)\|^2, \quad (4)$$

where $\bar{\mu}$ is the posterior mean of the forward process.

The diffusion policy (DP) models visuomotor robot policies using DDPMs, incorporating two key modifications: 1) The predicted data sample represents an *action sequence* $\mathbf{A}_t$ of length T_a, defined as the action execution horizon; 2) The latest T_s steps of *state sequence* $\mathbf{S}_t$ at the time step t is used as the conditional context for the denoising process.

Given $\mathbf{S}_t$, the conditional distribution of $\mathbf{A}_t$ is recovered through K steps of reverse process, using a modified version of Eq. 3:

$$p\left(\mathbf{A}_t^{k-1}|\mathbf{A}_t^k, \mathbf{S}_t\right) = \mathcal{N}\left(\mathbf{A}_t^{k-1}; \mu_\theta\left(\mathbf{A}_t^k, \mathbf{S}_t, k\right), \sigma_k^2 \mathbf{I}\right). \quad (5)$$

The noise predictor μ_θ is trained with a modified $\mathcal{L}_{\text{DM}}$, defined as:

$$\mathcal{L}_{\text{DP}}(\theta) = \mathbb{E}_{\left(\mathbf{A}_t, \mathbf{S}, \mathbf{A}_t^k, k\right)} \|\bar{\mu}\left(\mathbf{A}_t, k\right) - \mu_\theta\left(\mathbf{A}_t^k, \mathbf{S}, k\right)\|^2, \quad (6)$$

where $\mathbf{A}_t^0$ is shorthand for $\mathbf{A}_t$, representing the final action sequence for execution.

IV. Fine-tuning Diffusion Policy with Human Preference

In this section, we describe our approach for online RL-based fine-tuning of diffusion policy to align with human preference. First, a reward function representing human preference is obtained through preference-based reward learning (Sec. IV-A). Then, the reward model is used to fine-tune the diffusion policy (Sec. IV-B) using RL. We also incorporate KL regularization to stabilize fine-tuning, preventing overfitting to human preferences while preserving the model's ability to solve the original task (Sec. IV-C).

A. Preference-based Reward Learning

The reward function estimator $\widehat{r}_\psi$ can be seen as encapsulating human judgments about various robot behaviors. This follows a standard framework where a reward function is trained to align with human preference labels [14], [32], [33]. In this setup, a segment is defined as a sequence of states $\sigma = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_H\}$, where $H \geq 1$. In our case, we consider $H = 1$, meaning each segment consists of a single state. For a pair of segments $\left(\sigma^0, \sigma^1\right)$, a human annotator provides a feedback label $y \in \{-1, 0, 1\}$, indicating which segment is preferred, where 0 means the segment σ^0 is preferred, 1 means the segment σ^1 is preferred, and -1 means both segments are equally preferable.

Using the Bradley-Terry model [34], which assumes the probability of preferring one segment over another is exponentially dependent on the sum of an underlying reward function over the segment, the preference probability for a pair of segments, given the parameterized reward estimator $\widehat{r}_\psi$, can be expressed as

$$p_\psi[\sigma^1 \succ \sigma^0] = \frac{\exp\left(\sum_{h=1}^H \widehat{r}_\psi\left(\sigma^1\right)\right)}{\sum_{i\in\{0,1\}} \exp\left(\sum_{h=1}^H \widehat{r}_\psi\left(\sigma^i\right)\right)}, \quad (7)$$

where $\sigma^i \succ \sigma^j$ denotes segment σ^i being preferred over σ^j. In our case, Eq. 7 simplifies to

$$p_\psi[\mathbf{s}^1 \succ \mathbf{s}^0] = \frac{\exp\left(\widehat{r}_\psi\left(\mathbf{s}^1\right)\right)}{\sum_{i\in\{0,1\}} \exp\left(\widehat{r}_\psi\left(\mathbf{s}^i\right)\right)}. \quad (8)$$

Given a dataset of preference labels $D = \{\left(\sigma_i^0, \sigma_i^1, y_i\right)\}$, the reward function $\widehat{r}_\psi$ can be optimized by minimizing the following loss:

$$\begin{aligned}\mathcal{L}_{\text{RWD}}(\psi) = -\,\mathbb{E}_{(\sigma^0, \sigma^1, y)}\Big[&\mathbb{1}\{y = \left(\sigma^0 \succ \sigma^1\right)\} \log p_\psi[\sigma^0 \succ \sigma^1] \\ &+ \mathbb{1}\{y = \left(\sigma^1 \succ \sigma^0\right)\} \log p_\psi[\sigma^1 \succ \sigma^0]\Big],\end{aligned} \quad (9)$$

where $\mathbb{1}\{\cdot\}$ equals to 1 if the statement inside is true, and equals to 0 otherwise.

In the setting of diffusion policy, at any time step t, we generate an action sequence $\mathbf{A}_t = \{\mathbf{a}_t, \mathbf{a}_{t+1}, \ldots, \mathbf{a}_{t+T_a-1}\}$ using the diffusion policy conditioned on the state sequence $\mathbf{S}_t = \{\mathbf{s}_{t-T_s+1}, \mathbf{s}_{t-T_s+2}, \ldots, \mathbf{s}_t\}$. Consequently, we can express the reward as a function of the state-action sequence pair $(\mathbf{S}_t, \mathbf{A}_t)$ by

$$r_\psi\left(\mathbf{S}_t, \mathbf{A}_t\right) = \sum_{j=1}^{T_a} \widehat{r}_\psi\left(\mathbf{s}_{t+j}\right), \quad (10)$$

where each future state $\mathbf{s}_{t+j}$ is obtained by rolling out the action sequence $\mathbf{A}_t$ starting from the current state $\mathbf{s}_t$. Specifically, the state $\mathbf{s}_{t+j}$ is sampled according to the transition probability $\mathbf{s}_{t+j} \sim p\left(\mathbf{s}_{t+j} \mid \mathbf{s}_{t+j-1}, \mathbf{a}_{t+j-1}\right)$.

B. RL-based Fine-tuning

Assume a pre-trained diffusion policy $p_\theta\left(\mathbf{A}_t^{0:K}|\mathbf{S}_t\right)$ is given. We can fine-tune this diffusion policy with the aforementioned reward function $r_\psi\left(\mathbf{S}_t, \mathbf{A}_t\right)$ by maximizing the

Fig. 2: **Environments for Evaluation.** To evaluate the effectiveness of FDPP, We choose two long-horizon manipulation tasks including (left) a 2D pushing task PUSH-T [35], [3] and (right) a 3D pick-and-place task STACKING from MIMICGEN [36].

denoising diffusion RL objective:

$$\mathcal{J}_{\text{DDRL}}(\theta) = \mathbb{E}_{(\mathbf{S}_t, \mathbf{A}_t)}\left[r_\psi\left(\mathbf{S}_t, \mathbf{A}_t\right)\right], \tag{11}$$

where $\mathbf{S}_t$ is obtained from roll-outs starting with an initial state sequence with padding $\mathbf{S}_0 = \{\mathbf{s}_0, \ldots, \mathbf{s}_0\}$, where $\mathbf{s}_0 \sim \rho_0$ follows the initial state distribution. The action sequence $\mathbf{A}_t$ is sampled through the pre-trained diffusion policy $\mathbf{A}_t \sim p_\theta\left(\mathbf{A}_t^{0:K}|\mathbf{S}_t\right)$. Note that we only keep the final action sequence $\mathbf{A}_t = \mathbf{A}_t^0$.

As introduced in [25] and [26], we can represent the denoising process of DDPMs as a multi-step MDP, where the log-likelihood can be obtained via Monte-Carlo sampling. Specifically, we define the diffusion policy MDP $\mathcal{M}_{\text{DP}}$ as

$$\tilde{\mathbf{s}}_\tau \triangleq \left(\mathbf{S}_t, k, \mathbf{A}_t^k\right), \tag{12}$$

$$\tilde{\mathbf{a}}_\tau \triangleq \mathbf{A}_t^{k-1}, \tag{13}$$

$$\pi\left(\tilde{\mathbf{a}}_\tau|\tilde{\mathbf{s}}_\tau\right) \triangleq p_\theta\left(\mathbf{A}_t^{k-1}|\mathbf{A}_t^k, \mathbf{S}_t\right), \tag{14}$$

$$\rho_0\left(\tilde{\mathbf{s}}_0\right) \triangleq \left(p\left(\mathbf{S}_t\right), \delta_K, \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)\right), \tag{15}$$

$$p\left(\tilde{\mathbf{s}}_{\tau+1}|\tilde{\mathbf{s}}_\tau, \tilde{\mathbf{a}}_\tau\right) \triangleq \left(\delta_{\mathbf{S}_t}, \delta_{k-1}, \delta_{\mathbf{A}_t^{k-1}}\right), \tag{16}$$

$$r\left(\tilde{\mathbf{s}}_\tau, \tilde{\mathbf{a}}_\tau\right) \triangleq \begin{cases} r_\psi\left(\mathbf{A}_t, \mathbf{S}_t\right) & \text{if } \tau = 0 \\ 0 & \text{otherwise} \end{cases}. \tag{17}$$

Here, δ_y represents the Dirac delta distribution, which has non-zero density only at y. The trajectories in the diffusion policy MDP $\mathcal{M}_{\text{DP}}$ consist of K time steps, after which the state transition probability p leads to a termination state. It is important to note that τ refers to the time step in $\mathcal{M}_{\text{DP}}$, k refers to the denoising step in DDPM, and t refers to the time step in the original environment where the diffusion policy is applied.

Since the cumulative reward of each trajectory $\tilde{\xi}$ in $\mathcal{M}_{\text{DP}}$ is equal to the final step reward $r_\psi\left(\mathbf{A}_t, \mathbf{S}_t\right)$, maximizing $\mathcal{J}_{\text{DDRL}}(\theta)$ in Eq. 11 is equivalent to maximizing $\mathcal{J}_{\text{RL}}(\pi)$ in Eq. 1. Therefore, we can take the gradients with respect to the pre-trained diffusion policy parameters following Eq. 2:

$$\begin{aligned}\nabla_\theta \mathcal{J}_{\text{DDRL}} &= \mathbb{E}_{\tilde{\xi}}\left[\sum_{\tau=0}^{K} \nabla_\theta \log \pi_\theta\left(\tilde{\mathbf{a}}_\tau|\tilde{\mathbf{s}}_\tau\right) Q^{\pi_\theta}\left(\tilde{\mathbf{s}}_\tau, \tilde{\mathbf{a}}_\tau\right)\right], \\ &= \mathbb{E}\left[r_\psi\left(\mathbf{A}_t, \mathbf{S}_t\right) \sum_{k=1}^{K} \nabla_\theta \log p_\theta\left(\mathbf{A}_t^{k-1}|\mathbf{A}_t^k, \mathbf{S}_t\right)\right],\end{aligned} \tag{18}$$

where the expectation is taken over denoising trajectories generated by the current parameters θ.

C. KL Regularization

Fine-tuning a pre-trained diffusion policy solely using the preference-based reward model derived from human feedback risks over-fitting to the reward and forgetting the original task objective learned by the initial policy [26]. A common approach to mitigate this issue is to incorporate KL regularization [19], [26], [37]. Specifically, we compute the KL divergence between the fine-tuned and pre-trained models for the final action sequence as a regularization term, *i.e.*, $\mathcal{C} = \mathcal{D}_{\text{KL}}\left[p_\theta\left(\mathbf{A}_t|\mathbf{S}_t\right) \| p_{\text{pre}}\left(\mathbf{A}_t|\mathbf{S}_t\right)\right]$. Since obtaining a closed-form expression for $p_\theta\left(\mathbf{A}_t|\mathbf{S}_t\right)$ is challenging, we instead introduce an upper bound for this KL term into the objective function, following Lemma 4.2 in [26]:

$$\mathbb{E}_{\mathbf{S}_t}\left[\mathcal{C}\right] \leq \mathbb{E}_{\mathbf{S}_t}\left[\sum_{k=1}^{K} \mathbb{E}_{\mathbf{A}_t^k}\left(\overline{\mathcal{C}}\right)\right], \tag{19}$$

where $\overline{\mathcal{C}} = \mathcal{D}_{\text{KL}}\left[p_\theta\left(\mathbf{A}_t^{k-1}|\mathbf{A}_t^k, \mathbf{S}_t\right) \middle\| p_{\text{pre}}\left(\mathbf{A}_t^{k-1}|\mathbf{A}_t^k, \mathbf{S}_t\right)\right]$. Therefore, we include the upper bound into Eq. 11 to get the new KL regularized objective function:

$$\mathcal{J}_{\text{DDRL}}(\theta) = \mathbb{E}_{\mathbf{S}_t}\left[\mathbb{E}_{\mathbf{A}_t}\left(r_\psi\left(\mathbf{S}_t, \mathbf{A}_t\right)\right) + \alpha \sum_{k=1}^{K} \mathbb{E}_{\mathbf{A}_t^k}\left(\overline{\mathcal{C}}\right)\right], \tag{20}$$

where $\alpha \geq 0$ is the weight of the KL term. Similarly, the new gradient is

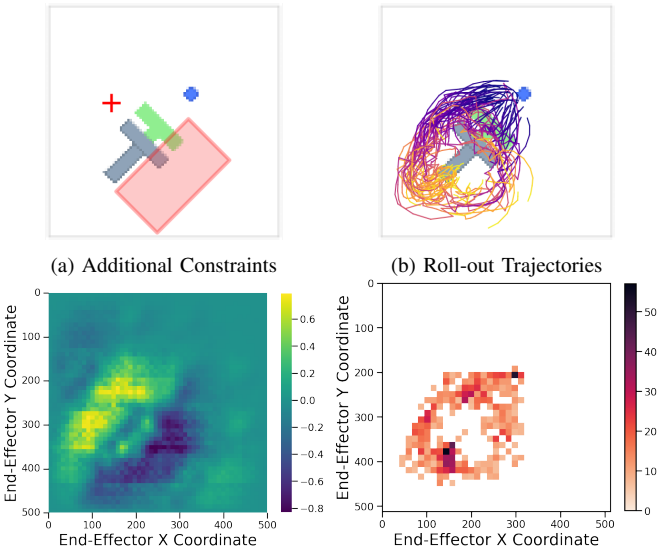
$$\begin{aligned}\nabla_\theta \mathcal{J}_{\text{DDRL}} = \mathbb{E}\Bigg[&r_\psi\left(\mathbf{A}_t, \mathbf{S}_t\right) \sum_{k=1}^{K} \nabla_\theta \log p_\theta\left(\mathbf{A}_t^{k-1}|\mathbf{A}_t^k, \mathbf{S}_t\right) \\ &+ \alpha \sum_{k=1}^{K} \nabla_\theta \mathcal{D}_{\text{KL}}\left[p_\theta\left(\mathbf{A}_t^{k-1}|\mathbf{A}_t^k, \mathbf{S}_t\right) \middle\| p_{\text{pre}}\left(\mathbf{A}_t^{k-1}|\mathbf{A}_t^k, \mathbf{S}_t\right)\right]\Bigg].\end{aligned} \tag{21}$$

D. Implementation Details

We apply Proximal Policy Optimization (PPO) [38] for the policy gradient update in the RL-based fine-tuning, which is more robust than the vanilla policy gradient methods. Diffusion policies are typically trained with stochastic sampler (*e.g.*, DDPMs) with large sampling steps K. During fine-tuning, we use the Denoising Diffusion Implicit Model (DDIM) [39] to reduce the number of sampling steps. One can change the deterministic level of DDIM through η, which controls the amount of noise injected into the sampling process, with 0 being fully deterministic and 1 being equivalent to the DDPM sampler. In practice, we set $\eta = 1$ with $K^{\text{DDIM}} = 50$ to improve the fine-tuning efficiency.

V. EXPERIMENTAL EVALUATION

The purpose of our experiments is to evaluate the effectiveness of **FDPP** for fine-tuning diffusion policies to align with a variety of user-specified objectives. We focus on the following questions:



(a) Additional Constraints (b) Roll-out Trajectories

(c) Reward Distribution (d) Sample Frequency

Fig. 3: **Preference-based Reward Model.** Incorporating the additional constraints on the end-effector's location (Top-Left), where the red box represents the undesirable region, we perform roll-outs of the diffusion policy to gather trajectory samples (Top-Right). A reward model is then trained as described in Sec. IV-A, assigning varying values to different end-effector locations (Bottom-Left). Locations outside the sample distribution result in reward values remaining near zero (Bottom-Right).

TABLE I: **Preference Alignment. FDPP** successfully adjusts the pre-trained diffusion policy to match human preferences for the desired feature. We present both the average feature over the entire trajectory and the feature at the terminal state.

	Average		Terminal State	
	Pre-trained	*Fine-tuned*	*Pre-trained*	*Fine-tuned*
PUSH-T [%]	0	**100**	–	–
STACK-DIST [cm]	6.28	**3.17**	3.91	**0.94**
STACK-ALIGN [°]	36.73	**26.16**	32.97	**21.20**

1) Can **FDPP** align the pre-trained diffusion policy with human preference? (Sec. V-B)
2) Does fine-tuning affect the performance of the original policy? (Sec. V-C)
3) How does KL regularization help preserve original task objective during fine-tuning? (Sec. V-D)

A. Setup

We choose two long-horizon manipulation tasks to evaluate **FDPP** as shown in Fig. 2. For each environment, we train a CNN-based diffusion policy following [3] with pre-collected human demonstrations as the pre-trained policy.

1) PUSH-T*:* This environment is adapted from IBC [35], where the task is to push a T-shaped block (gray) to a designated target location (green) using a circular end-effector (blue). The action space consists of the 2D planar position of the end-effector, while the observation space is 96×96 RGB image captured from a top-down view of the workspace.

We introduce an additional constraint for fine-tuning, ensuring that the end-effector does not enter an undesirable area located in the bottom-right of the workspace, depicted as the red box in Fig. 3(a).

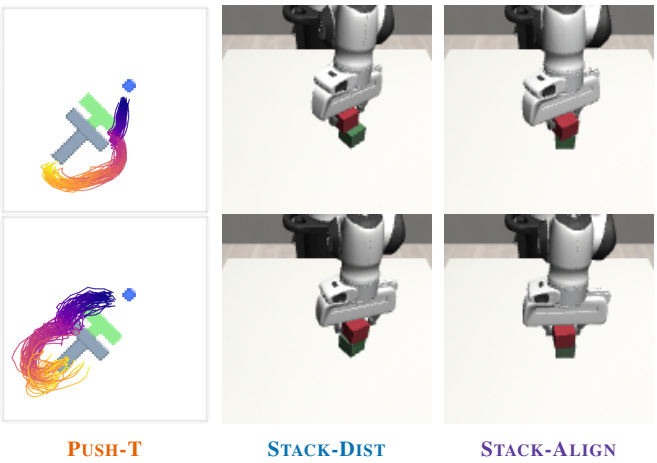


Fig. 4: **Change of Behaviors. FDPP** can effectively shift the behavior distribution of the pre-trained diffusion policy (Top) to align with additional constraints or human preferences (Bottom).

TABLE II: **Effect of Fine-tuning on Policy Performance.** The impact of **FDPP** on the performance of the fine-tuned policy varies depending on the specific preferences being incorporated. The average roll-out length of STACK is reported as box-lifted-length / total-trajectory-length.

	Success Rate		Average Roll-out Length	
	Pre-trained	*Fine-tuned*	*Pre-trained*	*Fine-tuned*
PUSH-T	**98%**	90%	**58.20**	120.20
STACK-DIST	88%	**92%**	**36.20/121.80**	37.32/131.28
STACK-ALIGN	88%	**96%**	36.20/121.80	**45.52/118.16**

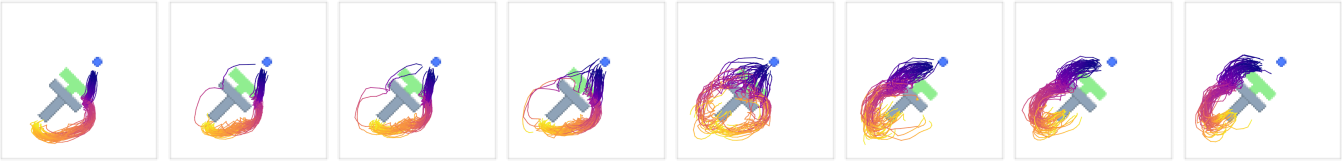
2) STACK*:* This environment is adapted from MIMICGEN [36], which is a large-scale robotic manipulation benchmark for imitation learning and offline RL. We choose the block stacking task where the robot is required to pick up the red block and stack it onto the green block. The action space consists of the target joint angles of the 7-DoF Panda robot, while the observation space is the low dimensional state representation of the environment.

We introduce two additional preferences: 1) STACK-DIST, which prefers minimizing the horizontal displacement between the centers of the red and green blocks; and 2) STACK-ALIGN, which encourages reducing the misalignment angle between the red and green blocks.

For each environment, we obtain a diffusion policy following [3] as the pre-trained policy with the same training dataset and hyper-parameters.

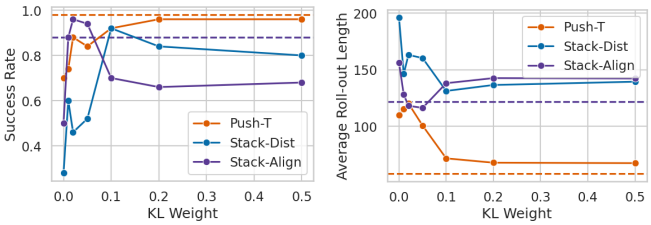
B. Preference Alignment

For fine-tuning, we first train the preference-based reward model as described in Sec. IV-A. Figure 3 presents the resulting reward model for PUSH-T. The training samples are generated by rolling out the pre-trained policy in the simulation, as depicted in Fig. 3(b) (showing the first 40 steps of each trajectory for the end-effector). A human annotator provides preference labels on randomly sampled state pairs (see Fig. 3(d) for sample frequency at the end-effector location). Each reward model is trained with 1024 state pairs. The final reward model (Fig. 3(c)) effectively penalizes the area enclosed by the red box (undesirable



Pre-trained | $\alpha = 0.5$ | $\alpha = 0.2$ | $\alpha = 0.1$ | $\alpha = 0.05$ | $\alpha = 0.02$ | $\alpha = 0.01$ | $\alpha = 0.0$

Fig. 5: **Effect of KL Regularization on Policy Behavior.** A large KL regularization weight results in minimal deviation from the pre-trained policy. Reducing the KL weight allows the fine-tuned policy to better align with the reward model.



(a) Success Rate vs. α (b) Average Roll-out Length vs. α

Fig. 6: **Effect of KL Regularization on Policy Performance.** A small KL weight leads to a significant decline in policy performance. Increasing the KL weight helps the fine-tuned policy's performance become more similar to that of the pre-trained policy. Dashed line represents the pre-trained policy performance.

region) and encourages the end-effector to move to the other side. Locations outside the sample distribution result in reward values that remain near zero. The reward model training for **STACK** follows a similar approach. However, instead of labeling pairs of end-effector locations, we label pairs based on horizontal displacement for **STACK-DIST** and pairs based on orientation angle differences for **STACK-ALIGN**.

Figure 4 shows the qualitative results of the fine-tuning process. Utilizing the preference-based reward, we can either completely alter the behavior distribution of the pre-trained policy in **PUSH-T**, achieve a smaller block displacement in **STACK-DIST**, or reduce block misalignment in **STACK-ALIGN**.

Table I quantitatively measures the alignment between the fine-tuned policy and human preference. For **PUSH-T**, it shows the percentage of trajectories entering the undesirable area. For **STACK-DIST**/**STACK-ALIGN**, it lists the average and final distances/orientation misalignment between blocks. We conclude that **FDPP** successfully adjusts the pre-trained diffusion policy to match human preferences for the desired feature.

C. Fine-tuned Policy Performance

Table II presents the success rate and average roll-out length for both pre-trained and fine-tuned policies across each environment. In **PUSH-T**, success is defined as achieving 90% coverage of the gray T-shaped block on the green target. In **STACK**, success is determined by the red block being successfully stacked on the green block. Each environment has a maximum of 200 steps.

We observe that the fine-tuned policy's performance varies based on the specific preferences incorporated. In **PUSH-T**, performance slightly decreases, with a longer average roll-out length, as the end-effector must take an alternate route to avoid the undesirable area. However, in **STACK-DIST** and **STACK-ALIGN**, fine-tuning enhances the success rate and reduces the average roll-out length. This improvement occurs because the human preferences are well-aligned with the task objectives, allowing fine-tuning with these rewards to boost policy performance.

D. KL Regularization

To prevent over-fitting, we introduce the KL regularizer (see Sec. IV-C). Figure 5 shows the effect of KL regularization on policy fine-tuning in **PUSH-T**. A large KL regularization weight results in minimal deviation from the pre-trained policy, while reducing the KL weight allows the fine-tuned policy to better align with the reward model. However, as illustrated in Fig. 6, a small KL weight can cause a significant decrease in policy performance because reinforcement learning tends to over-fit to the reward function, forgetting the original task objective of the pre-trained policy. This highlights the importance of KL divergence, as the original objective is not included in the reward function used for fine-tuning. In **STACK-DIST** and **STACK-ALIGN**, the impact of the KL weight on policy performance is more complex. Therefore, choosing an appropriate KL weight is essential to balance preference alignment and policy performance.

VI. CONCLUSION

We presented FDPP, a framework for adapting pre-trained diffusion policies to new human preferences through preference-guided reward learning and reinforcement learning fine-tuning. FDPP addresses a key limitation of existing imitation-based methods—the inability to generalize to evolving or personalized task specifications—by introducing a human-in-the-loop fine-tuning pipeline. Our empirical results demonstrate that FDPP effectively shifts the behavioral distribution of the policy to better match user intent, while KL regularization plays a critical role in maintaining the original task performance.

Looking ahead, we plan to extend FDPP in three directions: (1) leveraging vision-language models to automatically infer preferences from natural language descriptions, (2) validating FDPP on longer-horizon, real-world robotic tasks, and (3) developing automated strategies for tuning reward and regularization weights to further streamline the fine-tuning process.

REFERENCES

[1] C. Wang, L. Fan, J. Sun, R. Zhang, L. Fei-Fei, D. Xu, Y. Zhu, and A. Anandkumar, "Mimicplay: Long-horizon imitation learning by watching human play," in *7th Annual Conference on Robot Learning*.

[2] Y. Ze, G. Yan, Y.-H. Wu, A. Macaluso, Y. Ge, J. Ye, N. Hansen, L. E. Li, and X. Wang, "Gnfactor: Multi-task real robot learning with generalizable neural feature fields," in *Conference on Robot Learning*. PMLR, 2023, pp. 284–301.

[3] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, p. 02783649241273668, 2023.

[4] Y. Qin, Y.-H. Wu, S. Liu, H. Jiang, R. Yang, Y. Fu, and X. Wang, "Dexmv: Imitation learning for dexterous manipulation from human videos," in *European Conference on Computer Vision*. Springer, 2022, pp. 570–587.

[5] S. P. Arunachalam, S. Silwal, B. Evans, and L. Pinto, "Dexterous imitation made easy: A learning-based framework for efficient dexterous manipulation," in *2023 ieee international conference on robotics and automation (icra)*. IEEE, 2023, pp. 5954–5961.

[6] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," *arXiv preprint arXiv:2004.00784*, 2020.

[7] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, "3d diffusion policy," *arXiv preprint arXiv:2403.03954*, 2024.

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

[9] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*. PMLR, 2015, pp. 2256–2265.

[10] 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.

[11] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," *arXiv preprint arXiv:2205.09991*, 2022.

[12] Z. Wu, S. Ye, M. Natarajan, and M. C. Gombolay, "Diffusion-reinforcement learning hierarchical motion planning in adversarial multi-agent games," *arXiv preprint arXiv:2403.10794*, 2024.

[13] A. Y. Ng, S. Russell *et al.*, "Algorithms for inverse reinforcement learning." in *Icml*, vol. 1, no. 2, 2000, p. 2.

[14] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," *Advances in neural information processing systems*, vol. 30, 2017.

[15] D. Sadigh, A. Dragan, S. Sastry, and S. Seshia, *Active preference-based learning of reward functions*, 2017.

[16] E. Biyik and D. Sadigh, "Batch active preference-based learning of reward functions," in *Conference on robot learning*. PMLR, 2018, pp. 519–528.

[17] C. Wirth, R. Akrour, G. Neumann, and J. Fürnkranz, "A survey of preference-based reinforcement learning methods," *Journal of Machine Learning Research*, vol. 18, no. 136, pp. 1–46, 2017.

[18] 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.

[19] N. Stiennon, L. Ouyang, J. Wu, D. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, and P. F. Christiano, "Learning to summarize with human feedback," *Advances in Neural Information Processing Systems*, vol. 33, pp. 3008–3021, 2020.

[20] D. Zhang, M. Carroll, A. Bobu, and A. Dragan, "Time-efficient reward learning via visually assisted cluster ranking," *arXiv preprint arXiv:2212.00169*, 2022.

[21] D. Shin, A. D. Dragan, and D. S. Brown, "Benchmarks and algorithms for offline preference-based reward learning," *arXiv preprint arXiv:2301.01392*, 2023.

[22] R. Tian, C. Xu, M. Tomizuka, J. Malik, and A. Bajcsy, "What matters to you? towards visual representation alignment for robot learning," *arXiv preprint arXiv:2310.07932*, 2023.

[23] Y. Fan and K. Lee, "Optimizing ddpm sampling with shortcut fine-tuning," *arXiv preprint arXiv:2301.13362*, 2023.

[24] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[25] K. Black, M. Janner, Y. Du, I. Kostrikov, and S. Levine, "Training diffusion models with reinforcement learning," *arXiv preprint arXiv:2305.13301*, 2023.

[26] Y. Fan, O. Watkins, Y. Du, H. Liu, M. Ryu, C. Boutilier, P. Abbeel, M. Ghavamzadeh, K. Lee, and K. Lee, "Reinforcement learning for fine-tuning text-to-image diffusion models," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[27] K. Lee, H. Liu, M. Ryu, O. Watkins, Y. Du, C. Boutilier, P. Abbeel, M. Ghavamzadeh, and S. S. Gu, "Aligning text-to-image models using human feedback," *arXiv preprint arXiv:2302.12192*, 2023.

[28] A. Z. Ren, J. Lidard, L. L. Ankile, A. Simeonov, P. Agrawal, A. Majumdar, B. Burchfiel, H. Dai, and M. Simchowitz, "Diffusion policy policy optimization," *arXiv preprint arXiv:2409.00588*, 2024.

[29] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*. Pmlr, 2014, pp. 387–395.

[30] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, pp. 229–256, 1992.

[31] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[32] K. Lee, L. Smith, and P. Abbeel, "Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training," *arXiv preprint arXiv:2106.05091*, 2021.

[33] Y. Wang, Z. Sun, J. Zhang, Z. Xian, E. Biyik, D. Held, and Z. Erickson, "Rl-vlm-f: Reinforcement learning from vision language foundation model feedback," *arXiv preprint arXiv:2402.03681*, 2024.

[34] R. A. Bradley and M. E. Terry, "Rank analysis of incomplete block designs: I. the method of paired comparisons," *Biometrika*, vol. 39, no. 3/4, pp. 324–345, 1952.

[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] A. Mandlekar, S. Nasiriany, B. Wen, I. Akinola, Y. Narang, L. Fan, Y. Zhu, and D. Fox, "Mimicgen: A data generation system for scalable robot learning using human demonstrations," *arXiv preprint arXiv:2310.17596*, 2023.

[37] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022.

[38] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[39] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.