Diffusion Actor-Critic with Entropy Regulator

Yinuo Wang[1]   **Likun Wang**[1]   **Yuxuan Jiang**[1]   **Wenjun Zou**[1]   **Tong Liu**[1]
Xujie Song[1]   **Wenxuan Wang**[1]   **Liming Xiao**[2]   **Jiang Wu**[2]
Jingliang Duan[1,2*]   **Shengbo Eben Li**[1*]

[1]School of Vehicle and Mobility, Tsinghua University
[2]School of Mechanical Engineering, University of Science and Technology Beijing

Abstract

Reinforcement learning (RL) has proven highly effective in addressing complex decision-making and control tasks. However, in most traditional RL algorithms, the policy is typically parameterized as a diagonal Gaussian distribution with learned mean and variance, which constrains their capability to acquire complex policies. In response to this problem, we propose an online RL algorithm termed diffusion actor-critic with entropy regulator (DACER). This algorithm conceptualizes the reverse process of the diffusion model as a novel policy function and leverages the capability of the diffusion model to fit multimodal distributions, thereby enhancing the representational capacity of the policy. Since the distribution of the diffusion policy lacks an analytical expression, its entropy cannot be determined analytically. To mitigate this, we propose a method to estimate the entropy of the diffusion policy utilizing Gaussian mixture model. Building on the estimated entropy, we can learn a parameter α that modulates the degree of exploration and exploitation. Parameter α will be employed to adaptively regulate the variance of the added noise, which is applied to the action output by the diffusion model. Experimental trials on MuJoCo benchmarks and a multimodal task demonstrate that the DACER algorithm achieves state-of-the-art (SOTA) performance in most MuJoCo control tasks while exhibiting a stronger representational capacity of the diffusion policy.

1 Introduction

Recently, deep reinforcement learning (RL) has emerged as an effective method for solving optimal control problems in the physical world [14, 29, 22, 25]. In most existing RL algorithms, the policy is parameterized as a deterministic function or a diagonal Gaussian distribution with the learned mean and variance [31, 32, 16, 10]. However, the theoretically optimal policy may exhibit strong multimodality, which cannot be well modeled by deterministic or diagonal Gaussian policies [43, 21, 45]. Restricted policy representation capabilities can make algorithms prone to local optimal solutions, damaging policy performance. For instance, in situations where two distinct actions in the same state yield approximately the same Q-value, the Gaussian policy approximates the bimodal action by maximizing the Q-value. This results in the policy displaying mode-covering behavior, concentrating high density in the intermediate region between the two patterns, which is inherently a low-density region with a lower Q-value. Consequently, modeling the policy with a unimodal Gaussian distribution is likely to significantly impair policy learning.

Lately, the diffusion model has become widely known as a generative model for its powerful ability to fit multimodal distributions [18, 35, 8]. It learns the original data distribution through the idea of stepwise addition and removal of noise and has excellent performance in the fields of image [46, 28] and video generation [9, 3]. The policy network in RL can be seen as a state-conditional generative

---

*Corresponding author <duanjl15@163.com> <lishbo@tsinghua.edu.cn>.

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

model. Given the ability of diffusion models to fit complex distributions, there is increasing work on combining RL with diffusion models. Online RL learns policies by interacting with the environment [16, 32]. Offline RL, also known as batch RL, aims to effectively learn policies from previously collected data without interacting with the environment [1, 4]. In practical applications, many control problems have excellent simulators. At this time, using offline RL is not appropriate, as online RL with interaction capabilities performs better. Therefore, this paper focuses on how the diffusion model can be combined with online RL.

In this work, we propose diffusion actor-critic with entropy regulator (DACER), a generalized new approach to combine diffusion policy with online RL. Specifically, we base DACER on the denoising diffusion probabilistic model (DDPM) [18]. A recent work by He *et al.* [44] points out that the representational power of diffusion models stems mainly from the reverse diffusion processes, not from the forward diffusion processes. Inspired by this work, we reconceptualize the reverse process of the diffusion model as a novel policy approximator, leveraging its powerful representation capabilities to enhance the performance of RL algorithms. The optimization objective of this novel policy function is to maximize the expected Q-value. Maximizing entropy is important for policy exploration in RL, but the entropy of the diffusion policy is difficult to determine. Therefore, we choose to sample actions at fixed intervals and use a Gaussian mixture model (GMM) to fit the action distributions. Subsequently, We can calculate the approximate entropy of the policy in each state. The average of these entropies is then used as an approximation of the current diffusion policy entropy. Then, we use the estimated entropy to regulate the degree of exploration and exploitation of diffusion policy.

In summary, the key contributions of this paper are the following: 1) We propose to consider the reverse process of the diffusion model as a novel policy function. The objective function of the diffusion policy is to maximize the expected Q-value and thus achieve policy improvement. 2) We propose a method for estimating the entropy of diffusion policy. The estimated value is utilized to achieve an adaptive adjustment of the exploration level of the diffusion policy, thus improving the policy performance. 3) We evaluate the efficiency and generality of our method on the popular MuJoCo benchmarking. Compared with DDPG [33], TD3 [12], PPO [32], SAC [16], DSAC [11, 10], and TRPO [31], our approach achieves the SOTA performance. In addition, we demonstrate the superior representational capacity of our algorithm through a specific multi-goal task. 4) We provide the DACER code written in JAX to facilitate future researchers to follow our work [1].

Section 2 introduces and summarizes existing approaches to diffusion policy in offline RL and online RL, pointing out some of their problems. Section 3 provides an introduction to online RL and diffusion models. Our approach to combining diffusion policy with the mainstream actor-critic framework, as well as methods to enhance the performance of diffusion policy will be presented in section 4. The results of the experiments in the MuJoCo environment, the ablation experiments as well as the multimodality task will be presented in section 5. Section 6 provides the conclusions of this paper.

2   Related Work

Diffusion Policy in Offline RL   Offline RL leverages pre-collected datasets for policy development, circumventing direct environmental interaction. Current offline RL research utilizing diffusion models as policy networks primarily adhere to the behavioral cloning framework [7, 26]. Within this framework, two main objectives emerge: performance enhancement and training efficiency improvement. For the former, Cheng *et al.* [6] proposed a Diffusion Policy, casting the policy as a conditional denoising diffusion process within the action space to accommodate complex multimodal action distributions. Wang *et al.* [43] introduced Diffusion-QL, which integrates behavior cloning via diffusion model loss with Q-learning for policy improvement. Ajay *et al.* [2] created Decision Diffusion, incorporating classifier-free guidance into the diffusion model to integrate trajectory information, such as rewards and constraints. Addressing the latter, Kang *et al.* [20] developed efficient diffusion policies (EDP), an evolution of Diffusion-QL. EDP accelerates training by utilizing initial actions from state-action pairs in the buffer and applying a one-step sample for final action derivation. Chen *et al.* [5] proposed a consistency policy that enhances diffusion algorithm efficiency through one-step action generation from noise during training and inference. Although Diffusion Policy's powerful ability to fit multimodal policy distributions can achieve good performance in

---

[1]https://github.com/happy-yan/DACER-Diffusion-with-Online-RL

2

offline RL tasks, this method of policy improvement based on behavioral cloning cannot be directly transferred to online RL. In addition, the biggest challenge facing offline RL, the distribution shift problem, has not been completely solved. This paper focuses on online RL, moving away from the framework of behavioral cloning.

Diffusion Policy in Online RL Online RL, characterized by real-time environment interaction, contrasts with offline RL's dependence on pre-existing datasets. To date, only two studies have delved into integrating online RL with diffusion models. Yang *et al.* [45] pioneered this approach by proposing action gradient method. This approach achieves policy improvement by updating the action in the replay buffer through the $\nabla_a Q$, followed by mimicry learning of the action post-update using a diffusion model. However, action gradient increased the additional training time. Furthermore, it is difficult to fully learn both the action gradient and imitation learning steps simultaneously, which also resulted in suboptimal performance of this method in MuJoCo tasks. Psenka *et al.* [30] proposed Q-score matching (QSM), a new methodology for off-policy reinforcement learning that leverages the score-based structure of diffusion model policies to align with the $\nabla_a Q$. This approach aims to overcome the limitations of simple behavior cloning in actor-critic settings by integrating the policy's score with the Q-function's action gradient. However, QSM needs to accurately learn $\nabla_a Q$ in most of the action space to achieve optimal guidance. This is difficult to accomplish, resulting in suboptimal performance of QSM.

Our method is motivated to propose a diffusion policy that can be combined with most existing actor-critic frameworks. We first consider the reverse diffusion process of the diffusion model as a policy function with strong representational power. Then, we use the entropy estimation method to balance the exploration and utilization of diffusion policy and improve the performance of the policy.

Comparison with Diffusion-QL Diffusion-QL [43] made a successful attempt by replacing the diagonal Gaussian policy with a diffusion model. It also guides the updating of the policy by adding the normalized Q-value in the policy loss term. The main differences between our work and Diffusion-QL are as follows: 1) Diffusion-QL is still essentially an architecture for imitation learning, and policy updates are mainly motivated by the imitation learning loss term. 2) Our work adaptively regulates the standard deviation of random noise in the sampling process $\boldsymbol{a} = \boldsymbol{a} + \lambda\alpha \cdot \mathcal{N}(0, \boldsymbol{I})$, where α is a learned parameter, λ is a hyperparameter. This method effectively balances exploration and exploitation and subsequently enhances the performance of the diffusion policy. Ablation experiments provide evidence supporting these findings.

3 Preliminaries

3.1 Online Reinforcement Learning

In the conventional framework of RL, interactions between the agent and its environment occur in sequential discrete time steps. Typically, the environment is modeled as a Markov decision process (MDP) with continuous states and actions [38]. The environment provides feedback through a bounded reward function denoted by $r(s_t, a_t)$. The likelihood of transitioning to a new state based on the agent's action is expressed by the probability $p(s_{t+1}|s_t, a_t)$. State-action pairs for the current and next steps are indicated as (s, a) and (s', a'). The decision-making of an agent at any state s_t is guided by a stochastic policy $\pi(a_t|s_t)$, which determines the probability distribution over feasible actions at that state.

In the realm of online RL, agents engage in real-time learning and decision-making through direct interactions with their environments. Such interactions are captured within a tuple (s_t, a_t, r_t, s_{t+1}), representing the transition during each interaction. It is common practice to store these transitions in an experience replay buffer, symbolized as $\mathcal{B}$. Throughout the training phase, random samples drawn from $\mathcal{B}$ produce batches of data that contribute a more consistent training process. The fundamental aim of traditional online RL strategies is to craft a policy that optimizes the expected total reward:

$$J_\pi = \mathbb{E}_{(s_{i \geq t}, a_{i \geq t}) \sim \pi} \left[\sum_{i=t}^{\infty} \gamma^{i-t} r(s_i, a_i) \right],\tag{1}$$

3

where $\gamma \in (0, 1)$ represents the discount factor. The Q-value for a state-action pair (s, a) is given by

$$Q(s, a) = \mathbb{E}_\pi \Big[\sum_{i=0}^{\infty} \gamma^i r(s_i, a_i) | s_0 = s, a_0 = a \Big]. \tag{2}$$

RL typically employs an actor-critic framework [25, 24], which includes both a policy function, symbolized by π, and a corresponding Q-value function, noted as Q^π. The process of policy iteration is often used to achieve the optimal policy π^*, cycling through phases of policy evaluation and enhancement. In the policy evaluation phase, the Q-value Q^π is recalibrated according to the self-consistency requirements dictated by the Bellman equation:

$$Q^\pi(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim p, a' \sim \pi}[Q^\pi(s', a')]. \tag{3}$$

In the policy improvement phase, an enhanced policy π_{new} is sought by optimizing current Q-value $Q^{\pi_{\text{old}}}$:

$$\pi_{\text{new}} = \arg \max_\pi \mathbb{E}_{s \sim d_\pi, a \sim \pi}[Q^{\pi_{\text{old}}}(s, a)]. \tag{4}$$

In practical applications, neural networks are often used to parameterize both the policy and value functions, represented by π_θ and Q_ϕ, respectively. These functions are refined through the application of gradient descent methods aimed at reducing the loss functions for both the critic, $\mathcal{L}_q(\theta) = \mathbb{E}_{(s,a,s') \sim \mathcal{B}} \Big[\Big(r(s, a) + \gamma \mathbb{E}_{s' \sim p, a' \sim \pi}[Q^\pi(s', a')] - Q_\phi^\pi(s, a) \Big)^2 \Big]$, and the actor, $\mathcal{L}_\pi(\phi) = -\mathbb{E}_{s \sim d_\pi, a \sim \pi}[Q^{\pi_{\text{old}}}(s, a)]$. These loss functions are structured based on the principles outlined in (3) and (4).

3.2 Diffusion Models

Diffusion models [34, 18, 36, 37] are highly effective generative tools. They convert data from its original distribution to a Gaussian noise distribution by gradually adding noise and then reconstruct the data by gradually removing this noise through a reverse process. This process is typically described as a continuous Markov chain: the forward process incrementally increases the noise level, while the reverse process involves a conditional generative model trained to predict the optimal reverse transitions at each denoising step. Consequently, the model reverses the diffusion sequence to generate data samples starting from pure noise.

Let us define $p_\theta(\boldsymbol{x}_0) := \int p_\theta(\boldsymbol{x}_{0:T}) \mathrm{d}\boldsymbol{x}_{1:T}$, where $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ denote latent variables sharing the same dimensionality as the data variable $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$, where $q(\boldsymbol{x}_0)$ means original data distribution. In a forward diffusion chain, the noise is incrementally introduced to the data $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$ across T steps, adhering to a predetermined variance sequence denoted by β_t, described as

$$q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) = \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}), \quad q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{1 - \beta_t}\boldsymbol{x}_{t-1}, \beta_t). \tag{5}$$

When $T \to \infty$, $\boldsymbol{x}_T$ distributes as an isotropic Gaussian distribution [21]. The reverse diffusion process of the diffusion model can be represented as

$$p_\theta(\boldsymbol{x}_{0:T}) = p(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t), \quad p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t)), \tag{6}$$

where $p(\boldsymbol{x}_T) = \mathcal{N}(\boldsymbol{x}_T; \boldsymbol{0}, \boldsymbol{I})$ under the condition that $\prod_{t=1}^{T}(1 - \beta_t) \approx 0$.

4 Method

In this section, we detail the design of our diffusion actor-critic with entropy regulator (DACER). First, we consider the reverse diffusion process of the diffusion model as a new policy approximator, serving as the policy function in RL. Second, We directly optimize the diffusion policy using gradient descent, whose objective function is to maximize expected Q-values. This feature allows it to be integrated with mainstream RL algorithms that do not require entropy. However, the diffusion policy

4

learned this way produces overly deterministic actions with poor performance. When attempting to integrate the maximization entropy RL framework, we find the entropy of the diffusion policy is difficult to analytically determine. Therefore, we use GMM to approximate the entropy of the diffusion policy, and then learn a parameter α based on it to adjust the exploration level of diffusion policy.

4.1 Diffusion Policy Representation

We use the reverse process of a conditional diffusion model as a parametric policy:

$$\pi_\theta(\boldsymbol{a}|\boldsymbol{s}) = p_\theta(\boldsymbol{a}_{0:T}|\boldsymbol{s}) = p(\boldsymbol{a}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{a}_{t-1}|\boldsymbol{a}_t, \boldsymbol{s}), \tag{7}$$

where $p(\boldsymbol{a}_T) = \mathcal{N}(0, \boldsymbol{I})$, the end sample of the reverse chain, $\boldsymbol{a}_0$, is the action used for RL evaluation. Generally, $p_\theta(\boldsymbol{a}_{t-1}|\boldsymbol{a}_t, \boldsymbol{s})$ could be modeled as a Gaussian distribution $\mathcal{N}(\boldsymbol{a}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{a}_t, \boldsymbol{s}, t), \boldsymbol{\Sigma}_\theta(\boldsymbol{a}_t, \boldsymbol{s}, t))$. We choose to parameterize $\pi_\theta(\boldsymbol{a}|\boldsymbol{s})$ like DDPM [18], which sets $\boldsymbol{\Sigma}_\theta(\boldsymbol{a}_t, \boldsymbol{s}, t) = \beta_t I$ to fixed time-dependent constants, and constructs the mean $\boldsymbol{\mu}_\theta$ from a noise prediction model as

$$\boldsymbol{\mu}_\theta(\boldsymbol{a}_t, \boldsymbol{s}, t) = \frac{1}{\sqrt{\alpha_t}} \left(\boldsymbol{a}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\boldsymbol{a}_t, \boldsymbol{s}, t) \right), \tag{8}$$

where $\alpha_t = 1 - \beta_t, \bar{\alpha}_t = \prod_{k=1}^{t} \alpha_k$, and ϵ_θ is a parametric model.

To obtain an action from DDPM, we need to draw samples from T different Gaussian distributions sequentially. The sampling process can be reformulated as

$$\boldsymbol{a}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\boldsymbol{a}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\boldsymbol{a}_t, \boldsymbol{s}, t) \right) + \sqrt{\beta_t} \boldsymbol{\epsilon}, \tag{9}$$

with the reparametrization trick, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$, t is the reverse timestep from T to 0, $\boldsymbol{a}_T \sim \mathcal{N}(0, \boldsymbol{I})$.

4.2 Diffusion Policy Learning

In integrating diffusion policy with offline RL, policy improvement relies on minimizing the behavior-cloning term. However, in online RL, without a dataset to imitate, we discarded the behavior-cloning term and the imitation learning framework. In this study, the policy-learning objective is to maximize the expected Q-values of the actions generated by the diffusion network given the state:

$$\max_\theta \mathbb{E}_{\boldsymbol{s} \sim \mathcal{B}, \boldsymbol{a}_0 \sim \pi_\theta(\cdot|\boldsymbol{s})} [Q_\phi(\boldsymbol{s}, \boldsymbol{a}_0)]. \tag{10}$$

Unlike the traditional reverse diffusion process, our study requires recording the gradient of the whole process. The gradient of the Q-value function with respect to the action is backpropagated through the entire diffusion chain.

Policy improvement is introduced above; next, we introduce policy evaluation. The Q-value function is learned through a conventional approach, which involves minimizing the Bellman operator [13, 25, 38] with the double Q-learning trick [40]. We built two Q-networks $Q_{\phi_1}(\boldsymbol{s}, \boldsymbol{a}), Q_{\phi_2}(\boldsymbol{s}, \boldsymbol{a})$, and target network $Q_{\phi_1'}(\boldsymbol{s}, \boldsymbol{a}), Q_{\phi_2'}(\boldsymbol{s}, \boldsymbol{a})$. Then we give the objective function of policy evaluation, which is shown as

$$\min_{\phi_i} \mathbb{E}_{(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') \sim \mathcal{B}} \left[\left(\left(r(\boldsymbol{s}, \boldsymbol{a}) + \gamma \min_{i=1,2} Q_{\phi_i'}(\boldsymbol{s}', \boldsymbol{a}') \right) - Q_{\phi_i}(\boldsymbol{s}, \boldsymbol{a}) \right)^2 \right], \tag{11}$$

where $\boldsymbol{a}'$ is obtained by inputting the $\boldsymbol{s}'$ into the diffusion policy, $\mathcal{B}$ means replay buffer. Building on this, we employ the tricks in DSAC [11, 10] to mitigate the problem of Q-value overestimation.

The diffusion policy we construct can be directly combined with mainstream RL algorithms that do not require policy entropy. However, training with the above diffusion policy learning method suffers from overly deterministic policy actions, resulting in poor performance of the final diffusion policy. In the next section, we will propose entropy estimation to solve this problem and obtain diffusion policy with SOTA performance.

5

4.3 Diffusion Policy with Entropy

The diffusion policy's distribution lacks an analytic expression, so we cannot directly determine its entropy. However, in the same state, we can use multiple samples to obtain a series of actions. By fitting these action points, we can estimate the action distribution corresponding to the state.

In this paper, we use Gaussian mixture model (GMM) to fit the policy distribution. The GMM forms a complex probability density function by combining multiple Gaussian distributions, which can be represented as

$$\hat{f}(\boldsymbol{a}) = \sum_{k=1}^{K} w_k \cdot \mathcal{N}(\boldsymbol{a}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{12}$$

where K is the number of Gaussian distributions, and w_k is the mixing weight of the k-th component, satisfying $\sum_{k=1}^{K} w_k = 1, w_k \geq 0$. $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ are the mean and covariance matrices of the k-th Gaussian distribution, respectively.

For each state, we use a diffusion policy to sample N actions, $\boldsymbol{a}^1, \boldsymbol{a}^2, \ldots, \boldsymbol{a}^N \in \mathcal{A}$. The Expectation-Maximization algorithm is then used to estimate the parameters of the GMM. In the expectation step, the posterior probability that each data point $\boldsymbol{a}^i$ belongs to each component k is computed, denoted as

$$\gamma(\boldsymbol{z}_k^i) = \frac{w_k \cdot \mathcal{N}(\boldsymbol{a}^i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} w_j \cdot \mathcal{N}(\boldsymbol{a}^i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}, \tag{13}$$

where $\gamma(\boldsymbol{z}_k^i)$ denotes that under the current parameter estimates, the observed data $\boldsymbol{a}^i$ come from the k-th component of the probability. In the maximization step, the results of the Eq. (13) calculations are used to update the parameters and mixing weights for each component:

$$w_k = \frac{1}{N} \sum_{i=1}^{N} \gamma(\boldsymbol{z}_k^i), \boldsymbol{\mu}_k = \frac{\sum_{i=1}^{N} \gamma(\boldsymbol{z}_k^i) \cdot \boldsymbol{a}^i}{\sum_{i=1}^{N} \gamma(\boldsymbol{z}_k^i)}, \boldsymbol{\Sigma}_k = \frac{\sum_{i=1}^{N} \gamma(\boldsymbol{z}_k^i)(\boldsymbol{a}^i - \boldsymbol{\mu}_k)(\boldsymbol{a}^i - \boldsymbol{\mu}_k)^{\mathrm{T}}}{\sum_{i=1}^{N} \gamma(\boldsymbol{z}_k^i)}. \tag{14}$$

Iterative optimization continues until parameter convergence. Based on our experimental experience in the MuJoCo environments, a general setting of $K = 3$ provides a better fit to the action distribution.

According to Eq. (12), we can estimate the entropy of the action distribution corresponding to the state by [19]

$$\mathcal{H}_{\boldsymbol{s}} \approx -\sum_{k=1}^{K} w_k \log w_k + \sum_{k=1}^{K} w_k \cdot \frac{1}{2} \log\left((2\pi e)^d |\boldsymbol{\Sigma}_k|\right), \tag{15}$$

where d is the dimension of action. Then, the mean of the entropy of the actions associated with the chosen batch of states is used as the estimated entropy $\hat{\mathcal{H}}$ of the diffusion policy.

Similar to maximizing entropy RL, we learn a parameter α based on the estimated entropy. We update this parameter using

$$\alpha \leftarrow \alpha - \beta_\alpha [\hat{\mathcal{H}} - \overline{\mathcal{H}}], \tag{16}$$

where $\overline{\mathcal{H}}$ is target entropy. Finally, we use $\boldsymbol{a} = \boldsymbol{a} + \lambda \alpha \cdot \mathcal{N}(0, \boldsymbol{I})$ to adjust the diffusion policy entropy during training, where λ is a hyperparameter and $\boldsymbol{a}$ is the output of diffusion policy. Additionally, no noise is added during the evaluation phase. We summarize our implementation in Algorithm 1.

5 Experiments

We evaluate the performance of our method in some control tasks of RL within MuJoCo [39]. The benchmark tasks utilized in this study are depicted in Fig. 5, including Humanoid-v3, Ant-v3, HalfCheetah-v3, Walker2d-v3, InvertedDoublePendulum-v3, Hopper-v3, Pusher-v2, and Swimmer-v3. Moreover, we conducted experiments in a multi-goal task to demonstrate the excellent representational and exploratory capabilities of our diffusion policy. We also provide ablation studies on the critical components for better understanding. All baseline algorithms are available in GOPS [42], an open-source RL solver developed with PyTorch.

Baselines. Our algorithm is compared and evaluated against the six well-known model-free algorithms. These include DDPG [33], TD3 [12], PPO [32], SAC [16], DSAC [11, 10], and TRPO [31]. These baselines have been extensively tested and applied in a series of demanding domains.

6

Algorithm 1 Diffusion Actor-Critic with Entropy Regulator for Online RL

Input: λ, θ, ϕ_1, ϕ_2, ϕ_1', ϕ_2', α, β_q, β_α, β_π, and ρ
for each iteration **do**
    for each sampling step **do**
        Sample $\boldsymbol{a} \sim \pi_\theta(\cdot|\boldsymbol{s})$ by Eq. (7)
        Add noise $\boldsymbol{a} = \boldsymbol{a} + \lambda\alpha \cdot \mathcal{N}(0, \boldsymbol{I})$
        Get reward $\boldsymbol{r}$ and new state $\boldsymbol{s}'$
        Store a batch of samples $(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{r}, \boldsymbol{s}')$ in replay buffer $\mathcal{B}$
    end for
    for each update step **do**
        Sample data from $\mathcal{B}$
        Update critic networks using $\phi_i \leftarrow \phi_i - \beta_q\nabla_{\phi_i}\mathcal{L}_q(\phi_i)$ for $i = \{1, 2\}$
        Update diffusion policy network using $\theta \leftarrow \theta - \beta_\pi\nabla_\theta\mathcal{L}_\pi(\theta)$
        if step $\bmod\ 10000 == 0$ **then**
            Estimate the entropy of diffusion policy $\hat{\mathcal{H}} = \mathbb{E}_{\boldsymbol{s}\sim\mathcal{B}}[\mathcal{H}_{\boldsymbol{s}}]$
            Update α using Eq. (16)
        Update target networks using $\phi_i' = \rho\phi_i' + (1 - \rho)\phi_i$ for $i = \{1, 2\}$
    end for
end for

Experimental details. To ensure a fair comparison, we incorporated the diffusion policy as a policy approximation function within GOPS and implemented DACER with JAX, which improves training speed by 4-5 times compared to PyTorch while maintaining consistent performance. All algorithms and tasks use the same three-layer MLP neural network with GeLU [17] or Mish [27] activation functions, the latter used only for the noise prediction network in the diffusion policy. Initially, we encode timestep t into 16 dimensions using sinusoidal embedding [41], then merge this encoded result with the state $\boldsymbol{s}$ and action $\boldsymbol{a}_t$ during the current denoising step, and input it into the prediction noise network to generate the output. The impact of the reverse diffusion step size, T, on the experimental results will be examined in the ablation experiments. T is set to 20 for all experiments eventually. The Adam [23] optimization method is employed for all parameter updates. In this paper, the total training step size for all experiments is set at 1.5 million, with the results of all experiments averaged over five random seeds. The CPU used for the experiment is the AMD Ryzen Threadripper 3960X 24-Core Processor, and the GPU is NVIDIA GeForce RTX 3090Ti. Taking Humanoid-v3 as an example, the time taken to train 1.5 million in the JAX framework is 6 hours. More detailed hyperparameters are provided in Appendix A.2 due to space limits.

Evaluation Protocol. In this paper, we use the same assessment metrics as DSAC. For each seed, the metric is derived by averaging the highest return values observed during the final 10% of iteration steps in each run, with evaluations conducted every 15,000 iterations. Each assessment result is the average of ten episodes. The results from the five seeds are then aggregated to calculate the mean and standard deviation. Additionally, the training curves in Fig. 1 provide insights into the stability of the training process.

5.1 Comparative Evaluation

Each algorithm was subjected to five distinct tests, utilizing a variety of consistent random seeds to ensure robustness in the results. Fig. 1 and Table 1 display the learning curves and performance strategies, respectively. Our comprehensive findings reveal that across all evaluated tasks, the DACER algorithm consistently matched or surpassed the performance of all competing benchmark algorithms. Specifically, in the Humanoid-v3 scenario, our algorithm demonstrated enhancements of 124.7%, 111.1%, 73.1%, 27.3%, 9.8%, and 1131.9% over DDPG, TD3, PPO, SAC, DSAC, and TRPO, respectively.

5.2 Policy Representation Experiment

In this section, we conduct an experiment to confirm the representation capability of the diffusion policy. We use an environment called "Multi-goal" [15], as shown in Fig. 2, where the x-axis and

7

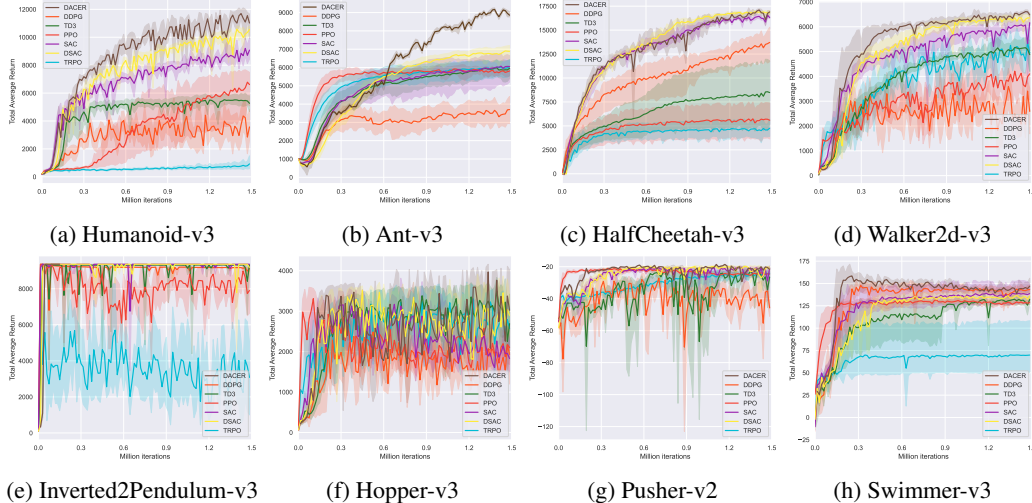



Figure 1: **Training curves on benchmarks.** The solid lines represent the mean, while the shaded regions indicate the 95% confidence interval over five runs. The iteration of PPO and TRPO is measured by the number of network updates.

Table 1

Average final return. Computed as the mean of the highest return values observed in the final 10% of iteration steps per run, with an evaluation interval of 15,000 iterations. The maximum value for each task is bolded. $\pm$ corresponds to standard deviation over five runs.

Task	DACER	DSAC	SAC	TD3	DDPG	TRPO	PPO
Humanoid-v3	**11888 ± 244**	10829 ± 243	9335 ± 695	5631 ± 435	5291 ± 662	965 ± 555	6869 ± 1563
Ant-v3	**9108 ± 103**	7086 ± 261	6427 ± 804	6184 ± 486	4549 ± 788	6203 ± 578	6156 ± 185
Halfcheetah-v3	**17177 ± 176**	17025 ± 157	16573 ± 224	8632 ± 4041	13970 ± 2083	4785 ± 967	5789 ± 2200
Walker2d-v3	**6701 ± 62**	6424 ± 147	6200 ± 263	5237 ± 335	4095 ± 68	5502 ± 593	4831 ± 637
Inverteddoublependulum-v3	**9360 ± 0**	**9360 ± 0**	**9360 ± 0**	9347 ± 15	9183 ± 9	6259 ± 2065	9356 ± 2
Hopper-v3	**4104 ± 49**	3660±533	2483 ± 943	3569 ± 455	2644 ± 659	3474 ± 400	2647 ± 482
Pusher-v2	**-19 ± 1**	**-19 ± 1**	-20 ± 0	-21 ± 1	-30 ± 6	-23 ± 2	-23 ± 1
Swimmer-v3	**152 ± 7**	138±6	140 ± 14	134 ± 5	146 ± 4	70 ± 38	130 ± 2

y-axis represent 2D states. In this setup, the agent is represented as a 2D point mass situated on a $7 * 7$ plane. The objective for the agent is to navigate towards one of four symmetrically positioned points: $(0, 5)$, $(0, -5)$, $(5, 0)$, and $(-5, 0)$. Since the goal positions are symmetrically distributed at the four points, a policy with strong representational capacity should enable the Q-function to learn the four symmetric peaks across the entire state space. This result reflects the policy's capacity for exploration in understanding the environment.

We compare the performance of DACER with DSAC, TD3, and PPO, as shown in Fig. 2. The results show that DACER's actions are likely to point to the nearest peak in different states. DACER's value function curve shows four symmetrical peaks, aligning with the previous analysis. Compared to DSAC, our method learns a better policy representation, mainly due to using a diffusion policy instead of an MLP. In contrast, TD3 and PPO generate more random actions with poorer policy representation, lacking the symmetrical peaks in their value function curves. Overall, our method demonstrates superior representational capability.

To demonstrate the powerful multimodality of DACER, we select five points requiring multimodal policies: (0.5, 0.5), (0.5, -0.5), (-0.5, -0.5), (-0.5, 0.5), and (0, 0). For each point, we sampled 100 trajectories. The trajectories are plotted in Fig. 3. The results show that compared with DSAC, DACER exhibits strong multimodality. This also explains why only the Q-function of DACER can learn the nearly perfectly symmetrical four peaks.

5.3 Ablation Study

In this section, we analyze why DACER outperforms all other baseline algorithms on MuJoCo tasks. We conduct ablation experiments to investigate the impact of the following three aspects on the

8

(a) DACER (b) DSAC (c) TD3 (d) PPO

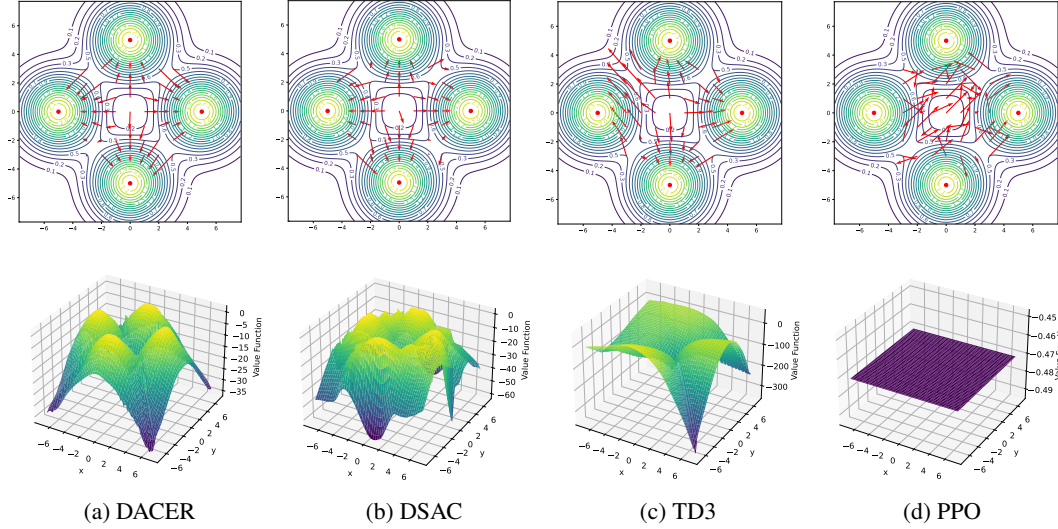



Figure 2: **Policy representation comparison of different policies on a multimodal environment.**
The first row exhibits the policy distribution. The length of the red arrowheads denotes the size of the
action vector, and the direction of the red arrowheads denotes the direction of actions. The second
row shows the value function of each state point.

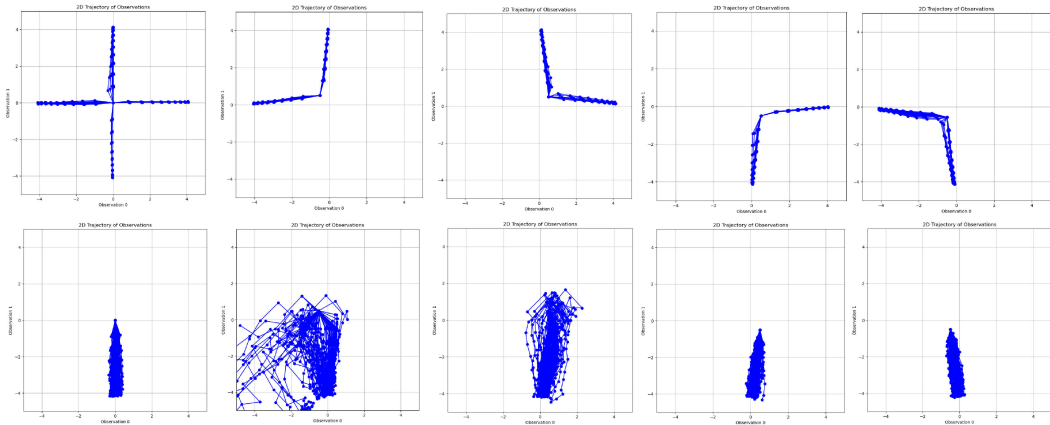



Figure 3: **Multi-goal multimodal experiments.** We selected 5 points that require multimodal
policies: (0, 0), (-0.5, 0.5), (0.5, 0.5), (0.5, -0.5), (-0.5, -0.5), and sampled 100 trajectories for each
point. The top row shows the experimental results of DACER, another shows the experimental results
of DSAC.

performance of the diffusion policy: 1) whether adding Gaussian noise to the final output action of
the diffusion policy; 2) whether the standard deviation of the added Gaussian noise can be adaptively
adjusted by estimated entropy; 3) different reverse diffusion step size T.

Only Q-learning. In section 4.2, we propose a method using the reverse diffusion process as
a policy approximator, which can be combined with the non-maximizing entropy RL algorithm.
However, the diffusion policy trained without entropy exhibits poor exploratory properties, leading to
suboptimal performance. Using Walker2d-v3 as an example, we compared the training curves of this
method with the DACER algorithm, as shown in Fig. 4(a).

Fixed and linear decay noise factor. In order to verify that using the estimated entropy to adaptively
adjust the noise factor plays an important role in the final performance, we conducted the following
two experiments in the Walker2d-v3 task: 1) Fixed noise factor to 0.1; 2) The noise factor starts

9

from 0.27 and linearly decreases to 0.1 during the training process. These two values were chosen because the starting and ending noise factor for adaptive tuning in this setting is about in this range. As shown in Fig. 4(b), our method of adaptively adjusting the noise factor based on the estimated entropy achieves the best performance.

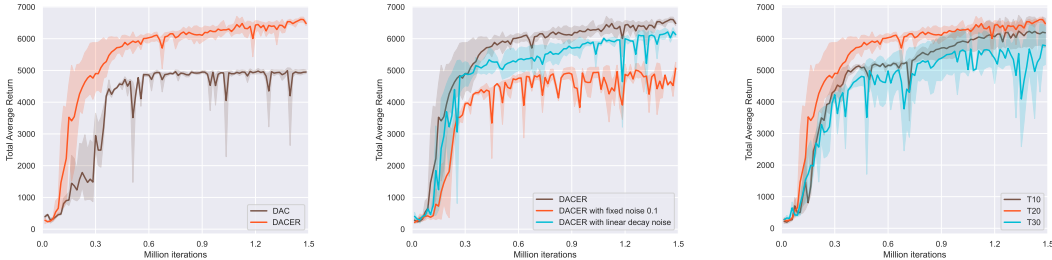
Diffusion steps. We further examined the performance of the diffusion policy as the number of diffusion timesteps T varied. We used the Walker2d-v3 task to plot training curves for $T = 10, 20$, and 30, as shown in Fig. 4(c). Experimental results indicate that a larger number of diffusion steps does not necessarily lead to better performance. Excessive diffusion steps can cause gradient explosion, significantly reducing the performance of diffusion policy. After balancing performance and computational efficiency, we selected 20 diffusion steps for all experiments.

| (a) Ablation for the entropy regulator mechanism. | (b) Ablation for the noise factor modulation mechanism. | (c) Ablation for the different diffusion steps. |

Figure 4: **Ablation experiment curves.** (a) DAC stands for not using the entropy regulator. DACER's performance on Walker2d-v3 is far better than DAC. (b) Adaptive tuning of the noise factor based on the estimated entropy achieved the best performance compared to fixing the noise factor or using the adaptive tuning method with initial, end values followed by a linear decay method. (c) The best performance was achieved with diffusion steps equal to 20, in addition to the instability of the training process when equal to 30.

6 Conclusion

In this study, we propose the diffusion actor-critic with entropy regulator (DACER) algorithm, a novel RL method designed to overcome the limitations of traditional RL methods that use diagonal Gaussian distributions for policy parameterization. By utilizing the inverse process of the diffusion model, DACER effectively handles multimodal distributions, enabling the creation of more complex policies and improving policy performance. A significant challenge arises from the lack of analytical expressions to determine the entropy of a diffusion strategy. To address this, we employ GMM to estimate entropy, thereby facilitating the learning of a key parameter, α, which adjusts the exploration-exploitation balance by regulating the noise variance in the action output. Empirical tests on the MuJoCo benchmark and a multimodal task show the superior performance of DACER.

7 Acknowledgements

This study is supported by National Key R&D Program of China with 2022YFB2502901, and Tsinghua University Initiative Scientific Research Program.

10

References

[1] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pages 104–114. PMLR, 2020.

[2] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *The Eleventh International Conference on Learning Representations*, 2023.

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

[4] David Brandfonbrener, Will Whitney, Rajesh Ranganath, and Joan Bruna. Offline rl without off-policy evaluation. *Advances in Neural Information Processing Systems*, 34:4933–4946, 2021.

[5] Yuhui Chen, Haoran Li, and Dongbin Zhao. Boosting continuous control with consistency policy. *arXiv preprint arXiv:2310.06343*, 2023.

[6] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[7] Felipe Codevilla, Eder Santana, Antonio M López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9329–9338, 2019.

[8] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[9] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

[10] Jingliang Duan, Yang Guan, Shengbo Eben Li, Yangang Ren, Qi Sun, and Bo Cheng. Distributional soft actor-critic: Off-policy reinforcement learning for addressing value estimation errors. *IEEE Transactions on Neural Networks and Learning Systems*, 33(11):6584–6598, 2021.

[11] Jingliang Duan, Wenxuan Wang, Liming Xiao, Jiaxin Gao, and Shengbo Eben Li. Dsac-t: Distributional soft actor-critic with three refinements. *arXiv preprint arXiv:2310.05858*, 2023.

[12] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

[13] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062. PMLR, 2019.

[14] Yang Guan, Yangang Ren, Qi Sun, Shengbo Eben Li, Haitong Ma, Jingliang Duan, Yifan Dai, and Bo Cheng. Integrated decision and control: Toward interpretable and computationally efficient driving intelligence. *IEEE Transactions on Cybernetics*, 53(2):859–873, 2022.

[15] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning*, pages 1352–1361. PMLR, 2017.

[16] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

[17] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

11

[18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[19] Marco F Huber, Tim Bailey, Hugh Durrant-Whyte, and Uwe D Hanebeck. On entropy approximation for gaussian mixture random vectors. In *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 181–188. IEEE, 2008.

[20] Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. *Conference and Workshop on Neural Information Processing Systems*, 2023.

[21] Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[22] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, 2023.

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[24] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in Neural Information Processing Systems*, 12, 1999.

[25] S Eben Li. *Reinforcement Learning for Sequential Decision and Optimal Control*. Springer Verlag, Singapore, 2023.

[26] Abdoulaye O Ly and Moulay Akhloufi. Learning to drive by imitation: An overview of deep behavior cloning methods. *IEEE Transactions on Intelligent Vehicles*, 6(2):195–209, 2020.

[27] Diganta Misra. Mish: A self regularized non-monotonic activation function. *arXiv preprint arXiv:1908.08681*, 2019.

[28] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023.

[29] Baiyu Peng, Qi Sun, Shengbo Eben Li, Dongsuk Kum, Yuming Yin, Junqing Wei, and Tianyu Gu. End-to-end autonomous driving through dueling double deep q-network. *Automotive Innovation*, 4:328–337, 2021.

[30] Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. *arXiv preprint arXiv:2312.11752*, 2023.

[31] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897. PMLR, 2015.

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

[33] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, pages 387–395. PMLR, 2014.

[34] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[35] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. *Advances in Neural Information Processing Systems*, 34:1415–1428, 2021.

[36] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

12

[37] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[38] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

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

[40] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[42] Wenxuan Wang, Yuhang Zhang, Jiaxin Gao, Yuxuan Jiang, Yujie Yang, Zhilong Zheng, Wenjun Zou, Jie Li, Congsheng Zhang, Wenhan Cao, et al. Gops: A general optimal control problem solver for autonomous driving and industrial control applications. *Communications in Transportation Research*, 3:100096, 2023.

[43] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *The Eleventh International Conference on Learning Representations*, 2023.

[44] Saining Xie Xinlei Chen, Zhuang Liu and Kaiming He. Deconstructing denoising diffusion models for self-supervised learning. *arXiv preprint arXiv:2401.14404*, 2024.

[45] Long Yang, Zhixiong Huang, Fenghao Lei, Yucun Zhong, Yiming Yang, Cong Fang, Shiting Wen, Binbin Zhou, and Zhouchen Lin. Policy representation via diffusion probability model for reinforcement learning. *arXiv preprint arXiv:2305.13122*, 2023.

[46] Ruihan Yang, Prakhar Srivastava, and Stephan Mandt. Diffusion probabilistic modeling for video generation. *Entropy*, 25(10):1469, 2023.

13

A Environmental Details

A.1 Experimental Environment Introduction

The benchmark tasks utilized in this study are depicted in Fig. 5, including Humanoid-v3, Ant-v3, HalfCheetah-v3, Walker2d-v3, InvertedDoublePendulum-v3, Hopper-v3, Pusher-v2, and Swimmer-v3.

(a)   (b)   (c)   (d)

(e)   (f)   (g)   (h)

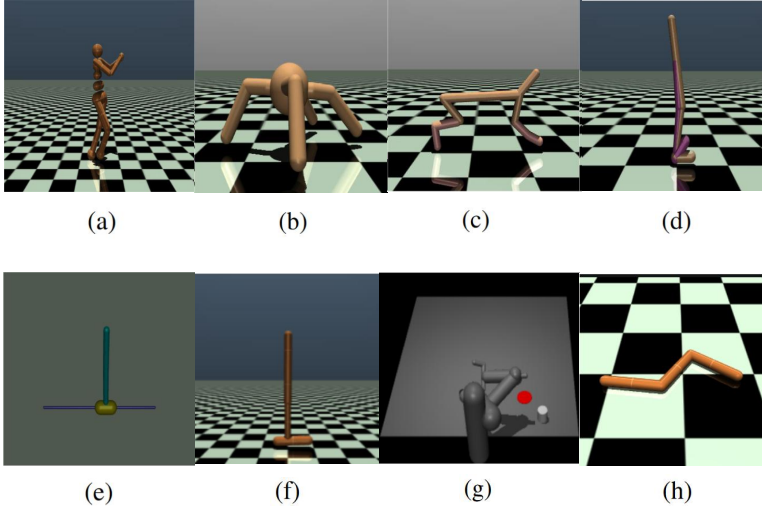



Figure 5: **Simulation tasks.** (a) Humanoid-v3:$(s \times a) \in \mathbb{R}^{376} \times \mathbb{R}^{17}$. (b) Ant-v3: $(s \times a) \in \mathbb{R}^{111} \times \mathbb{R}^{8}$. (c) HalfCheetah-v3 : $(s \times a) \in \mathbb{R}^{17} \times \mathbb{R}^{6}$. (d) Walker2d-v3: $(s \times a) \in \mathbb{R}^{17} \times \mathbb{R}^{6}$. (e) InvertedDoublePendulum-v3: $(s \times a) \in \mathbb{R}^{6} \times \mathbb{R}^{1}$. (f) Hopper-v3: $(s \times a) \in \mathbb{R}^{11} \times \mathbb{R}^{3}$. (g) Pusher-v2: $(s \times a) \in \mathbb{R}^{23} \times \mathbb{R}^{7}$. (h) Swimmer-v3: $(s \times a) \in \mathbb{R}^{8} \times \mathbb{R}^{2}$.

A.2 Training Details on MuJoCo tasks

Mujoco [39] is a simulation engine primarily designed for research in RL and robotics. It provides a versatile, physics-based platform for developing and testing various RL algorithms. Core features of Mujoco include a highly efficient physics engine, realistic modeling of dynamic systems, and support for complex articulated robots. Currently, it is one of the most recognized benchmark environments for RL and continuous control.

The hyperparameters of all baseline algorithms are shown in Table 2. Moreover, the hyperparameters of the DACER in the MuJoCo task are shown in Table 3.

B Limitation and Future Work

In this study, we propose using GMM to estimate the entropy of the diffusion policy and, based on this estimate, learn a parameter α to balance exploration and exploitation. However, the process of estimating entropy requires a large number of samples and takes a long time (about 40 ms). Therefore, we estimate the entropy of diffusion policy every 10,000 iterations to reduce the impact on training time. But, this approach prevents perfect integration with maximizing entropy RL. In future work, we will avoid using batch-size data to estimate entropy and find a balance between estimation accuracy and computational efficiency so as to better combine our method with maximizing entropy RL.

C Positive and Negative Social Impact

In this paper, we propose DACER, an online RL algorithm that uses the reverse diffusion process as a policy approximator. Diffusion policy has powerful multimodal representation capabilities, making it widely applicable in complex environments such as automated manufacturing, autonomous

14

TABLE 2
DETAILED HYPERPARAMETERS.

Hyperparameters	Value
Shared	
Replay buffer capacity	1000000
Buffer warm-up size	30000
Batch size	256
Initial alpha α	0.272
Action bound	$[-1, 1]$
Hidden layers in critic network	[256, 256, 256]
Hidden layers in actor network	[256, 256, 256]
Activation in critic network	GeLU
Activation in actor network	GeLU
Optimizer	Adam ($\beta_1 = 0.9, \beta_2 = 0.999$)
Actor learning rate	1e−4
Critic learning rate	1e−4
Discount factor (γ)	0.99
Policy update interval	2
Target smoothing coefficient (ρ)	0.005
Reward scale	0.2
Maximum-entropy framework	
Learning rate of α	3e−4
Expected entropy ($\overline{\mathcal{H}}$)	$\overline{\mathcal{H}} = -\dim(\mathcal{A})$
Deterministic policy	
Exploration noise	$\epsilon \sim \mathcal{N}(0, 0.1^2)$
Off-policy	
Replay buffer size	1×10^6
Sample batch size	20
On-policy	
Sample batch size	2000
Replay batch size	2000

driving, and industrial control. However, DACER could also enhance the exploratory capabilities and operational efficiency of military AI, potentially posing threats to citizen privacy and security.

15

TABLE 3
ALGORITHM HYPERPARAMETER

Parameter	Setting
Replay buffer capacity	1000000
Buffer warm-up size	30000
Buffer warm-up size (Humanoid, HalfCheetah)	200000
Batch size	256
Discount γ	0.99
Initial alpha α	0.27
Target network soft-update rate ρ	0.005
Network update times per iteration	1
Action bound	$[-1, 1]$
Reward scale	0.2
Hidden layers in noise prediction network	[256, 256, 256]
Hidden layers in noise prediction network (Humanoid)	[512, 512, 512]
Hidden layers in critic network	[256, 256, 256]
Activations in critic network	GeLU
Activations in actor network	Mish
Policy act distribution	TanhGauss
Policy min log std	-20
Policy max log std	0.5
Policy delay update	2
Number of Gaussian distributions for mixing	3
Number of action samples in entropy estimation	200
Alpha delay update	10000
Noise scale λ	0.1
Noise scale λ (Humanoid, HalfCheetah)	0.15
Optimizer	Adam
Actor learning rate	$1 \cdot 10^{-4}$
Critic learning rate	$1 \cdot 10^{-4}$
Alpha learning rate	$3 \cdot 10^{-2}$
Target entropy	$-0.9 \cdot \dim(\mathcal{A})$

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: In both the abstract and the introduction we give a formulation of the contribution points. These include considering the reverse diffusion process as a novel policy approximator, proposing a Gaussian mixture model method to estimate the entropy of the diffusion policy and learning a parameter α for the regulation of the policy exploration level, and open-sourcing the code. Experimental results in eight MuJoCo environments as well as one multimodal environment demonstrate the good performance of our method and the strong characterization capability of the diffusion policy.

   Guidelines:

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

16

- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

Answer: [Yes]

Justification: In Section B of the appendix, we acknowledge the limitations of our methodology, estimating entropy requires a considerable amount of time, making it difficult to perfectly integrate this method with maximum entropy RL. However, in future work, we will improve the estimation efficiency by reducing the number of states used for estimation.

Guidelines:

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

3. **Theory Assumptions and Proofs**

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

Answer: [NA]

Justification:

Guidelines:

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

4. **Experimental Result Reproducibility**

17

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

Answer: [Yes]

Justification: First, in Section 4.3, we provide pseudo-code for the algorithm 1 of diffusion actor-critic with entropy regulator (DACER). Second, in Section 5, we provide details of the experimental environment setup. Finally, among the two tables in the Appendix A.2, we give details of the hyperparameter configuration.

Guidelines:

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

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

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

Answer: [Yes]

Justification: All the reproductions of the experimental results of the baseline algorithms can be obtained by running the general optimal control problem solver (GOPS). And GOPS can be searched on github. We provide the PyTorch code for DACER as a function approximator in GOPS. Besides, we have implemented the DACER algorithm in JAX.

Regarding the DACER algorithm, we give an implementation of all the core parts, but the complete training code we will open source after the review.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.

18

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

6. **Experimental Setting/Details**

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

   Answer: [Yes]

   Justification: In Appendix A.2, we give the design of all hyperparameters, including optimizer selection, learning rate, neural network configuration, and so on.

   Guidelines:

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

7. **Experiment Statistical Significance**

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

   Answer: [Yes]

   Justification: All results of this experiment came from in 5 random seeds. In Section 5, we mention the specific way of evaluation and the fact that all results are presented as mean ± standard deviation. Regarding the training curves, the solid lines represent the mean, while the shaded regions indicate the 95% confidence interval over five runs.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.

19

- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

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

Answer: [Yes]

Justification: In Section 5, we provide the CPU and GPU models used for training, which are Xeon(R) Platinum 8352V and NVIDIA GeForce RTX 4090, respectively. We take the example of training 1.5 million Humanoid-v3, and the time required to train it in the JAX framework is about 7 hours.

Guidelines:

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

9. **Code Of Ethics**

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

Answer: [Yes]

Justification: We made sure the code was anonymous.

Guidelines:

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

10. **Broader Impacts**

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

Answer: [Yes]

Justification: In Appendix C, we discuss the potential positive and negative social impacts of our work.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate

20

to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

Answer: [NA]

Justification:

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: We used the GOPS solver for training and cited the corresponding paper. This open-source code library is licensed under the Apache-2.0 license. Copyright © 2022 Intelligent Driving Laboratory (iDLab). All rights reserved.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

21

13. **New Assets**

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

    Answer: [NA]

    Justification:

    Guidelines:

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

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

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

    Answer: [NA]

    Justification:

    Guidelines:

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

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

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

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

22