Value Diffusion Reinforcement Learning

Xiaoliang Hu[1] **Fuyun Wang**[1] **Tong Zhang**[1*] **Zhen Cui**[2*]
[1] School of Computer Science and Engineering, Nanjing University of Science and Technology
[2] School of Artificial Intelligence, Beijing Normal University
{peter_hu_xl, fyw271828}@njust.edu.cn

Abstract

Model-free reinforcement learning (RL) combined with diffusion models has achieved significant progress in addressing complex continuous control tasks. However, a persistent challenge in RL remains the accurate estimation of Q-values, which critically governs the efficacy of policy optimization. Although recent advances employ parametric distributions to model value distributions for enhanced estimation accuracy, current methodologies predominantly rely on unimodal Gaussian assumptions or quantile representations. These constraints introduce distributional bias between the learned and true value distributions, particularly in some tasks with a nonstationary policy, ultimately degrading performance. To address these limitations, we propose value diffusion reinforcement learning (VDRL), a novel model-free online RL method that utilizes the generative capacity of diffusion models to represent multimodal value distributions. The core innovation of VDRL lies in the use of the variational loss of diffusion-based value distribution, which is theoretically proven to be a tight lower bound for the optimization objective under the KL-divergence measurement. Furthermore, we introduce double value diffusion learning with sample selection to enhance training stability and further improve value estimation accuracy. Extensive experiments conducted on the MuJoCo benchmark demonstrate that VDRL significantly outperforms some SOTA model-free online RL baselines, showcasing its effectiveness and robustness.

1 Introduction

Recent advancements in model-free reinforcement learning (RL) integrated with diffusion models [1, 2, 3, 4, 5, 6, 7] have demonstrated remarkable success in solving practical decision-making tasks, such as robotic locomotion control [8], robotic navigation [9], and robot arm manipulation [10, 11, 12]. By harnessing the expressive power of diffusion models [13, 14, 15], these works focus on enhancing the exploration efficiency of policy by effectively capturing the expressiveness and multimodality of policy beyond traditional action distribution assumption (e.g., unimodal Gaussian) [16]. However, a core challenge in RL remains the accurate estimation of state-action values (i.e., Q-values), which is fundamental to both policy evaluation and improvement [17]. Despite their enhanced exploration capabilities, RL methods combined with diffusion policies often suffer from value estimation errors, particularly overestimation [18], leading to suboptimal solutions [19].

To mitigate value overestimation, some techniques like double Q-network [20] and approximate upper-bound [21] have been incorporated into prominent RL algorithms, such as twin delayed deep deterministic policy gradient (TD3) [22] and soft actor-critic (SAC) [23], achieving significant performance improvements. Nevertheless, these methods may introduce a considerable underestimation bias and increase the computation cost for each iteration [24]. Therefore, inaccuracies in value estimation remain a limiting factor for further advancements in policy learning [25].

[*]Corresponding author <zhen.cui@bnu.edu.cn, tong.zhang@njust.edu.cn>.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

On the other hand, distributional RL [26], rooted in the idea of modeling the value distribution rather than estimating only the expected return (i.e., Q-value), has emerged as a novel and promising avenue for improving the accuracy of value estimation. Building on this concept, some early works [27, 28, 29, 30, 31] leverage quantile functions to represent the value distribution. However, these approaches are limited to discrete or low-dimensional action spaces. To address this challenge, the recent representative works such as DSAC [24] and DSACT [32] learn a continuous Gaussian value distribution and provide solid theoretical insights into how value distribution learning reduces overestimation. Despite the reduction in value overestimation, the parametric value distributions of these methods are constrained by the assumption (i.e., unimodal Gaussian), which may introduce bias between the approximated and true value distributions in tasks with non-stationary policies, where the true value distribution often exhibits multimodal characteristics. Consequently, this bias leads to imprecise value estimation, yielding a degraded performance. The above limitation motivates a key question: *Can we develop a universal value distribution representation of RL that simultaneously achieves multimodal expressiveness and flexibility for complex continuous control tasks?*

Inspired by the considerable success of diffusion models applied in RL [4, 5, 6, 7], we propose Value Diffusion Reinforcement Learning (VDRL), a novel model-free online RL method. To the best of our knowledge, this is the first work to integrate diffusion models into distributional RL, enabling precise value estimation. The core idea of VDRL is conceptually elegant yet highly effective: we harness diffusion models to generate more expressive and multimodal distributions for representing value distributions. Our key theoretical insight reveals that the optimization objective of value distribution learning for distributional RL, under the KL-divergence measure, can be reformulated as the variational bound objective of the diffusion model. To mitigate variance introduced by iterative diffusion sampling, we introduce double value diffusion learning with adaptive sample selection, ensuring stable training while preserving distributional fidelity. In summary, the key contributions of this work are as follows:

i) To improve the accuracy of value estimation, we propose the value diffusion reinforcement learning (VDRL) method. VDRL leverages the capability of the diffusion model to fit more expressive and multimodal value distributions. More importantly, we theoretically prove that the variational loss of diffusion-based value distribution is a tight bound for the optimization objective of value distribution learning under the KL-divergence measurement.

ii) To address the challenge of unstable learning introduced by iterative diffusion sampling, we propose the double value diffusion learning with sample selection for value distributions. This mechanism can further reduce the value overestimation bias and might introduce a minor value underestimation, which enhances learning stability and promotes better performance.

iii) We perform extensive experiments on eight popular MuJoCo [33] benchmark tasks, comparing the performance of VDRL to eleven model-free online RL baselines, including traditional RL, distributional RL, and diffusion-based policy RL methods. The experimental results demonstrate that VDRL achieves stable and consistently superior performance across most evaluated tasks, highlighting its robustness and effectiveness.

2 Preliminaries

2.1 Problem Formulation

The standard RL task could be regarded as a Markov decision process (MDP) [17]. Formally, the task can be formulated as a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma \rangle$, where each element is defined as follows:

- $\mathcal{S}$: a finite set of states. Given a current state $s \in \mathcal{S}$, we often denote the next state as $s' \in \mathcal{S}$.
- $\mathcal{A}$: a set of actions. Note that the action may be discrete or continuous.
- $\mathcal{P}(s'|s, a)$: the state transition function, which usually denotes the probability from the current state s to the next state s' when the agent selects the action a.
- $R(s, a)$: the reward function. After the agent performs an action a under the environment state s, the agent will receive an immediate global reward $r = R(s, a)$. Notably, in this work, the reward is explicitly treated as a random variable.
- $\gamma \in (0, 1]$: the discount factor used in the computation of cumulative return.

Further, the behavior of the agent is defined by a stochastic policy $\pi(a|s)$, which maps a given state to a probability distribution over actions. The state-action value function $Q^\pi(s,a)$ associated with a given policy π denotes the expectation of cumulative state-action return when the agent taking the action $a \in \mathcal{A}$ under the state $s \in \mathcal{S}$, i.e., $Q^\pi(s,a) := \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R(s_k,a_k)|s_0=s,a_0=a\right]$.

2.2 Distributional Reinforcement Learning

Distributional Reinforcement Learning [26, 34, 29] learns the full probability distribution of the cumulative state-action return $\mathcal{Z}(\cdot|s,a)$ instead of the expected return, i.e., a single scalar Q-value $Q(s,a)$, which behaves better than traditional RL methods in terms of learning stability and performance. In specific, the cumulative state-action return $Z^\pi(s,a)$ associated with a given policy π is the sum of discounted rewards along the trajectory of interactions with the environment, i.e., $Z^\pi(s,a) = \sum_{k=0}^{\infty}\gamma^k R(s_k,a_k)$. Note that the cumulative state-action return is a random variable, which describes the intrinsic randomness of the agent's interactions with its environment, i.e., the randomness of state transition $P(s'|s,a)$, reward $R(s,a)$, and next state-action return $Z(s',a')$ [26]. The distribution of the cumulative state-action return $\mathcal{Z}^\pi(\cdot|s,a)$ [2] is a mapping from state-action pairs to distribution over the random return, i.e., $Z^\pi(s,a) \sim \mathcal{Z}^\pi(\cdot|s,a)$. Significant to distributional RL is the use of the distributional Bellman operator to describe the value distribution, defined by

$$\mathcal{T}^\pi Z(s,a) \overset{D}{:=} R(s,a) + \gamma P^\pi Z(s',a'), \tag{1}$$

where the equation $X \overset{D}{:=} Y$ represents that the random variable X is distributed according to the same law with Y, $s' \sim P(\cdot|s,a)$, $a' \sim \pi(\cdot|s')$, and $\mathcal{T}^\pi Z(s,a)$ is a new random variable, whose distribution is defined as $\mathcal{T}^\pi \mathcal{Z}(\cdot|s,a)$.

Distributional Soft Actor-Critic. In most previous distributional RL methods, the value distribution is commonly modeled by a discrete probability density function [26, 30, 29] or a quantile function [27, 28, 35] due to its highly expressive advantage [36]. However, these functions restrict direct access to the mean value, necessitating its approximation via a weighted sum of finite quantiles. To overcome this issue, the recent work [24] proposed the distributional soft actor-critic (DSAC) method by directly learning a continuous parameterized probability density function (Gaussian) for the random return $Z(s,a)$, i.e., $\mathcal{Z}_\theta(\cdot|s,a) = \mathcal{N}(Q_\theta(s,a), \sigma^2_\theta(s,a))$. The parameter θ can be updated by optimizing the Kullback-Leibler (KL) divergence between the target value distribution $\mathcal{T}^\pi\mathcal{Z}(\cdot|s,a)$ and value distribution $\mathcal{Z}^\pi(\cdot|s,a)$ via gradient descent, i.e.,

$$J^{\text{DSAC}}_{\mathcal{Z}}(\theta) = \mathbb{E}_{(s,a)\sim\mathcal{B}}\left[D_{\text{KL}}(\mathcal{T}^{\pi_{\bar{\phi}}}\mathcal{Z}_{\bar{\theta}}(\cdot|s,a) \parallel \mathcal{Z}_\theta(\cdot|s,a))\right], \tag{2}$$

where $\mathcal{B}$ is a replay buffer of previously sampled experience, $\bar{\theta}$ and $\bar{\phi}$ represent the parameters of the target value distribution and policy functions.

2.3 Diffusion Model

Diffusion models [13, 14] are highly powerful tools that learn to generate data by gradually denoising a normally distributed variable, which have achieved superior expressiveness in representing complex probability distributions, particularly in image and video generation [37, 38, 39, 40, 41]. A representative algorithm is the well-known denoising diffusion probabilistic models (DDPM) [15], which synthesizes data by learning to reverse a predefined noising process, i.e., forward process, through iterative denoising steps modeled by a neural network, i.e., reverse process. Given the real data x_0 samples from the date distribution $x_0 \sim q(x_0)$, DDPM uses a parameterized latent variable model, i.e., $p_\theta(x_0) = \int p_\theta(x_{0:T})dx_{1:T}$, to model how the isotropic Gaussian noise $x_T \sim \mathcal{N}(\mathbf{0},\mathbf{I})$ is denoised into real data x_0, where x_t denotes the step of diffusion process and T represents the total number of diffusion steps. The forward process is a Markov chain that gradually adds Gaussian noise to the data according to a variance schedule $\beta_1, \cdots, \beta_T$, expressed by

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}), \quad q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I}). \tag{3}$$

The reverse process reconstructs data distribution starting at $p(x_T) = \mathcal{N}(x_T;\mathbf{0},\mathbf{I})$, represented as

$$p_\theta(x_{0:T}) = p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t), \quad p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t,t), \Sigma_\theta(x_t,t)), \tag{4}$$

[2] For notational simplicity, this distribution is called as value distribution in this work.

where $\mu_\theta(x_t, t)$ denotes the mean function approximator, and $\Sigma_\theta(x_t, t)$ is set $\sigma_t^2\mathbf{I}$ to untrained time dependent constants. Inspired by VAE [42], the variational lower bound (VLB) is utilized to optimize the negative log likelihood as the training objective of DDPM:

$$J_{\text{VLB}}(\theta) = \mathbb{E}_{x_0\sim p_0, x_{1:T}\sim q}\left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right]. \tag{5}$$

To this end, the diffusion model is trained by minimizing the simplified loss function:

$$\mathcal{L}_{\text{DDPM}}(\theta) = \mathbb{E}_{t\sim[1,T],x_0,\epsilon}\left[\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\|^2\right], \tag{6}$$

where $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$, $\alpha_t = 1 - \beta_t$ and ϵ is sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

3 Method

In this section, we propose a novel model-free online RL method called value diffusion reinforcement learning (VDRL) to improve the accuracy of value estimation. The key innovation of VDRL is the use of the variational loss of diffusion-based value distribution, which can be theoretically proven as a tight lower bound for the optimization objective of value distribution learning under the KL-divergence measurement. Furthermore, we introduce double value diffusion learning with sample selection to stabilize training and further enhance estimation accuracy.

3.1 Variational Objective for Value Diffusion Learning

According to the distributional Bellman operator in Eq. (1), the optimization objective of value distribution learning in distributional RL is to minimize the distance between the target value distribution, i.e.,$\mathcal{Z}_{\text{tar}}(\cdot|s,a)$, and the current value distribution, i.e.,$\mathcal{Z}_{\text{cur}}(\cdot|s,a)$, formally,

$$\min_{\mathcal{Z}_{\text{cur}}}\left\{\mathop{\mathbb{E}}_{(s,a)\sim\mathcal{B}}\left[d(\mathcal{Z}_{\text{tar}}(\cdot|s,a), \mathcal{Z}_{\text{cur}}(\cdot|s,a))\right]\right\}, \tag{7}$$

where $\mathcal{Z}_{\text{tar}}(\cdot|s,a) = \mathcal{T}^\pi\mathcal{Z}_{\text{old}}(\cdot|s,a)$ and d is the distance metric for measuring two probability distributions, such as Kullback-Leibler (KL) divergence, Wasserstein distance.

To optimize the objective in Eq. (7), in practice, we need to choose a tractable approximating distribution for modeling the value distribution. For instance, the value distribution is modeled as a unimodal Gaussian distribution in DSAC [24] and DSACT [32]. However, as highlighted in [43], the unimodal Gaussian approximation fails to capture the inherent expressiveness and multimodality of value distributions, which may introduce bias between the approximated and true distributions in tasks with non-stationary policies, leading to imprecise value estimation and then yielding degraded performance. More importantly, we find that when the value distribution is modeled by a diffusion model, the optimization objective under the KL-divergence measurement can be reformulated and simplified into a variational bound objective, which is formalized as follows:

Theorem 1. ***(Lower Bound of Learning Objective for Value Distribution)*** *If the value distribution is modeled by a diffusion model, the variational bound objective*

$$\mathbb{E}_{\substack{(s,a,r,s')\sim\mathcal{B},a'\sim\pi_{\bar{\phi}},\\ Z(s',a')\sim\mathcal{Z}_{\bar{\theta}}(\cdot|s',a')}}\left[\mathbb{E}_{\mathcal{T}^\pi Z_{1:T}\sim q(\mathcal{T}^\pi Z_{1:T}|s,a,\mathcal{T}^\pi Z_0)}\left[\log\frac{P_\theta(\mathcal{T}^\pi Z_{0:T}|s,a)}{q(\mathcal{T}^\pi Z_{1:T}|s,a,\mathcal{T}^\pi Z_0)}\right]\right], \tag{8}$$

is the tight lower bound of the optimization objective of value distribution learning for distributional RL under the KL-divergence measurement

$$\mathbb{E}_{(s,a)\sim\mathcal{B}}\left[D_{KL}(\mathcal{T}^\pi\mathcal{Z}_{\bar{\theta}}(\cdot|s,a) \parallel \mathcal{Z}_\theta(\cdot|s,a))\right], \tag{9}$$

where $\mathcal{Z}_\theta(\cdot|s,a)$ denotes the value distribution to be optimized and $\mathcal{Z}_{\bar{\theta}}(\cdot|s,a)$ is the parameterized target value distribution, and the equality holds when the policy converges.

Proof. The proof is deferred to Appendix A. □

Remark 1. *Although the variational lower bound objective in our method (VDRL), as shown in Eq. (8), appears similar to QVPO [6], they differ fundamentally in both derivation and formulation:*

- ***Derivation****: Our variational lower bound is derived from the optimization objective of the value distribution, whereas QVPO derives its lower bound from the policy objective in RL.*
- ***Formulation****: The variational lower bound objective of QVPO incorporates Q-value under the specific condition, i.e.,* $Q(s, a) \geq 0$*, while our method maintains a straightforward form without introducing additional constraints.*

According to Theorem 1, we can derive the variational loss function of value diffusion learning as Eq. (10), which can be applied to value distribution optimization for distributional RL.

$$\mathcal{L}(\theta) = \mathbb{E}_{\substack{(s,a,r,s')\sim\mathcal{B}, a'\sim\pi_{\bar{\phi}}, \\ Z(s',a')\sim\mathcal{Z}_{\bar{\theta}}(\cdot|s',a'), \boldsymbol{\epsilon}, t}} \left[\left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \left(\sqrt{\bar{\alpha}_t} \mathcal{T}^\pi Z + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, s, a, t \right) \right\|^2 \right], \tag{10}$$

Where $\mathcal{T}^\pi Z(s,a) = R(s,a) + \gamma Z(s',a')|_{Z(s',a')\sim\mathcal{Z}_{\bar{\theta}}(\cdot|s',a')}$.

3.2 Double Value Diffusion Learning with Sample Selection

While the value diffusion learning captures the inherent expressiveness and multimodality of the true value distribution, it also introduces a large variance, which may lead to unstable learning processes in some tasks. To address this problem, we propose double value diffusion learning with sample selection for value distribution $\mathcal{Z}(\cdot|s,a)$. Specifically, we parameterize two value distribution functions $\mathcal{Z}_{\theta_i}(\cdot|s,a)$, then choose sample with lower random return $Z(s',a')$ to calculate the target random return $\mathcal{T}^\pi Z(s,a)$, formally,

$$\mathcal{T}^\pi Z(s,a) = R(s,a) + \gamma \min_{i=1,2} Z(s',a')|_{Z(s',a')\sim\mathcal{Z}_{\bar{\theta}_i}(\cdot|s',a')}. \tag{11}$$

Furthermore, as shown in Table 2, this mechanism for value distribution can further reduce the value overestimation, and might induce a relatively minor value underestimation. In fact, as discussed in TD3 [22], the slight value underestimation is preferable to value overestimation, as unlike overestimation, underestimated value do not propagate explicitly through policy update during training.

3.3 Value Diffusion Reinforcement Learning

According to the above introduction, we present a practical model-free online off-policy RL algorithm called value diffusion reinforcement learning (VDRL), which includes two value distribution functions $\mathcal{Z}(\cdot|s,a)$ and a policy function $\pi(a|s)$. To that end, we use function approximators for both the value distribution and the policy, and alternate between optimizing both networks with stochastic gradient descent. Specifically, we consider two parameterized value distribution functions $\mathcal{Z}_{\theta_1}(\cdot|s,a), \mathcal{Z}_{\theta_2}(\cdot|s,a)$ and a parameterized policy function $\pi_\phi(a|s)$. The parameters of these networks are θ_1, θ_2, and ϕ, respectively. Furthermore, we employ the idea of a target network, where parameters of the target network can be an exponentially moving average of the main network weights, which has been shown to stabilize training in DQN [44]. According to the theorem 1 in section 3.1, the value distribution functions can be trained to minimize the variational objective,

$$J_{\mathcal{Z}}(\theta_i) = \mathbb{E}_{\substack{(s,a,r,s')\sim\mathcal{B}, a'\sim\pi_{\bar{\phi}}, \\ Z(s',a')\sim\mathcal{Z}_{\bar{\theta}_i}(\cdot|s',a')}} \left[\mathbb{E}_{\mathcal{T}^\pi Z_{1:T}\sim q(\mathcal{T}^\pi Z_{1:T}|s,a,\mathcal{T}^\pi Z_0)} \left[\log \frac{P_{\theta_i}(\mathcal{T}^\pi Z_{0:T}|s,a)}{q(\mathcal{T}^\pi Z_{1:T}|s,a,\mathcal{T}^\pi Z_0)} \right] \right], \tag{12}$$

Then, the policy function parameters can be learned by directly maximizing the expected cumulative state-action return $Z(s,a)$, formally,

$$J_\pi(\phi) = \mathbb{E}_{s\sim\mathcal{B}, a\sim\pi_\phi} \left[\min_{i=1,2} \mathbb{E}_{Z(s',a')\sim\mathcal{Z}_{\bar{\theta}_i}(s',a')} \left[Z(s,a) \right] \right]. \tag{13}$$

In implementation, we employ Monte Carlo estimation to approximate the expectation of each value distributions, i.e., $\mathbb{E}_{Z(s',a')\sim\mathcal{Z}_{\bar{\theta}_i}(s',a')}[Z(s,a)]$. For the sake of comprehensiveness, we provide a complete training process of the proposed VDRL method, as shown in Algorithm 1.

4 Related Work

In this section, we will look over existing works that are the most relevant to our proposed method. We divide our surveys into two main parts. Firstly, we provide an overview of distributional reinforcement

Algorithm 1 Value Diffusion Reinforcement Learning

Input: Diffusion-based value distribution networks $\{\mathcal{Z}_{\theta_i}(Z|s,a)\}_{i=1}^2$, target diffusion-based value distribution networks $\{\mathcal{Z}_{\overline{\theta}_i}(Z|s,a)\}_{i=1}^2$, policy network $\pi_\phi(a|s)$, target policy network $\pi_{\overline{\phi}}(a|s)$, the learning rate of value distribution and policy lr_θ, lr_ϕ, the momentum coefficient τ

Initializing: all networks parameters $\overline{\theta}_1 \leftarrow \theta_1$, $\overline{\theta}_2 \leftarrow \theta_2$, $\overline{\phi} \leftarrow \phi$;
for each iteration **do**
  Initialize the environment;
  for each sampling step **do**
    Select action $a \sim \pi_\phi(s,a)$;
    Execute action a to obtain the reward r and the next state s';
    Store $(s,a,r,s',done)$ in replay buffer $\mathcal{B}$;
  end for
  for each optimization step **do**
    Sample a random minibatch of N samples from replay buffer:$(s,a,r,s',done)$;
    Update the parameters of diffusion-based value distribution networks $\{\theta_i\}_{i=1}^2$: $\theta_i \leftarrow \theta_i - \beta_\theta \nabla_{\theta_i} J_{\mathcal{Z}}(\theta_i)$;
    Update the parameters of policy network ϕ: $\phi \leftarrow \phi + \beta_\phi \nabla_\phi J_\pi(\phi)$;
    Update the parameters of target diffusion-based value distribution networks $\overline{\theta}_i = \tau\theta_i + (1-\tau)\overline{\theta}_i$ for $i = \{1,2\}$;
    Update the parameters of target policy network $\overline{\phi} = \tau\phi + (1-\tau)\phi$;
  end for
end for

learning. Secondly, we discuss the diffusion-based RL methods that use diffusion models as polices for model-free online RL setups.

Distributional Reinforcement Learning. Recent interest in distributional methods [45, 46, 43] for RL has grown with the introduction of deep RL approaches for learning the value distribution. This paradigm was first formalized in the seminal work [26], which introduced the Categorical DQN (C51) algorithm that approximates the value distribution using a fixed categorical distribution over discrete support points and applies a distributional Bellman operator. Building on this foundation, subsequent works [27, 28, 29, 30, 31] leveraged quantile functions to represent the value distribution, enabling significant advancements. However, these approaches are inherently limited to discrete and low-dimensional action spaces. In order to deal with continuous control settings, methods such as [34, 35, 47, 48] have been proposed in recent years. Nevertheless, these approaches approximate the mean value through a weighted average of finite quantiles, which may reduce the accuracy of the mean value estimation, leading to suboptimal performance in some tasks. The distributional soft actor-critic (DSAC) [24] addressed the above limitation by learning a continuous Gaussian value distribution with the Q-value directly parameterized by the critic network, improving value estimation accuracy. Despite its effectiveness, DSAC suffers from training instability and sensitivity. To alleviate these issues, the term DSAC with three refinements (DSACT) [32] introduces three key refinements to refine Q-value estimation. However, both DSAC and DSACT constrain the value distribution and stochastic policy space to unimodal Gaussian distributions, which hinders exploration efficiency and learning performance, particularly in tasks with multiple goals or complex dynamics, where the true distribution often exhibits multimodality. In contrast, *the proposed VDRL method employs diffusion models to capture more expressive and multimodal representations of the value distribution, which further enhances value estimation and learning performance in tasks requiring sophisticated modeling of uncertainty and dynamics.*

Diffusion Models for Model-free Online Reinforcement Learning. Recently, numerous works [1, 49, 50, 51, 52, 53, 54, 2, 3] have employed diffusion models as the policy class in model-free offline RL to tackle decision-making tasks, due to their superior expressive richness in representing complex distributions. In contrast, diffusion policies have been less widely explored in model-free online RL, owing to the noisier and policy-varying nature of value estimations [16]. To date, only several studies [4, 5, 6, 7] have investigated the application of diffusion models for online RL. The pioneering work, DIPO [4], provided a solid convergence guarantee for diffusion policies and facilitated policy improvement by updating the action via the action gradient of the Q-value. However, fully learning of action gradient not only presents a significant challenge but also introduces additional computational costs. Unlike DIPO, Q-score matching (QSM) [5] proposed a novel policy update mechanism by aligning the score of the learned diffusion model, i.e., $\nabla_a \log(\pi(a|s))$ with the action gradient of the Q-value. Unfortunately, it neglects inaccuracies in the value function gradient and biases introduced during the alignment process when updating the policy. To mitigate these limitations, Q-weighted variational policy optimization (QVPO) [6] introduced the Q-weighted variational lower bound (VLO) loss, and then designed a tailored entropy regularization term for diffusion polices. Concurrently, Wang *et al*. [7] proposed the diffusion actor-critic with entropy regulator (DACER) method, and employed Gaussian mixture models for entropy estimation. Despite these great advances, fundamental challenges persist: approximation inaccuracies and computational inefficiencies continue to constrain the effectiveness of diffusion models in model-free online RL. Different from all existing model-free online RL methods with diffusion policy, *from the value perspective, the proposed VDRL method utilizes diffusion models to represent value distributions, which enhances the accuracy of value estimation, yielding significantly improved performance.*

5 Experiments

In this section, we empirically validate the effectiveness of the proposed algorithm on different decision-making tasks and with various hyperparameters. With these experimental results, we aim to answer the following questions: 1) How does VDRL compare to traditionally canonical and existing diffusion-based model-free online RL methods? 2) What contributes to the performance improvement of VDRL on the continuous locomotion tasks? Note that all experiments are conducted on a 2.90GHz Intel Core i7-10700 CPU, 64G RAM, and NVIDIA GeForce RTX 3090 GPU.

5.1 Experimental Setup

Evaluation Environment. In this work, we use eight different level tasks of the popular MuJoCo [33] benchmark [3] to evaluate the performance of all methods, including Ant-v3, HalfCheetah-v3, Hopper-v3, Humanoid-v3, Inverted2Pendulum-v2, Pusher-v2, Swimmer-v3, and Walker2d-v3. The more details of these tasks and the MuJoCo benchmark are deferred to Appendix B.

Compared Baselines. We evaluate the performance of VDRL against eleven model-free online RL algorithms, categorized into three groups. The first group comprises **traditional model-free RL** baselines, including two on-policy methods (PPO [55] and SPO [56]) and three off-policy algorithms (DDPG [57], TD3 [22], and SAC [23]). The second group focuses on **distributional RL** approaches, represented by DSAC [24] and DSACT [32]. Finally, the third group consists of **advanced diffusion-based policy RL** methods, including DIPO [4], QSM [5], QVPO [6], and DACER [7]. The more details of implementation and hyperparameters for all baselines are deferred to Appendix D.

Methodology for Reported Metrics. For the proposed method and all baselines, all experiments are conducted over 800,000 training interaction steps with four runs and different random seeds (0, 50, 100, 200), with each performing one evaluation rollout every 10,000 interaction steps. In particular, each evaluation result is the average of ten episodes. The mean return, standard deviation, and average highest return are logged as the performance of methods.

5.2 Details of Implementation

The value distribution is modeled using the reverse process of a conditional diffusion model, parameterized similarly to DDPM [15], as detailed in Section 2.3. The effects of reverse diffusion steps and noise schedules are analyzed in Section 5.4. For the policy, a Gaussian distribution with a mean and diagonal covariance is adopted. The policy network architecture consists of two hidden layers, each with 256 units and GeLU activation functions. Parameter updates are performed using the Adam optimizer. A detailed description of all training hyperparameters is deferred to Appendix C.

5.3 Comparative Evaluation

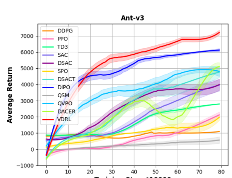
The best performance of the proposed method (VDRL) and all baselines across eight MuJoCo benchmark tasks is presented in Table 1. VDRL achieves results comparable to baseline methods

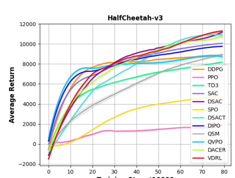
[3] The version of MuJoCo utilized in all experiments of this work is mujoco210, which is the same version used as DSAC [24], QVPO [6], DACER [7], and DSACT [32].

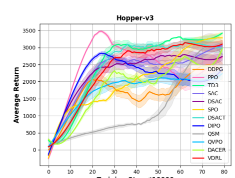
Table 1: Performance on eight tasks of OpenAI Gym MuJoCo benchmark. The results show the best mean returns and standard deviations over 800,000 training interaction steps and four random seeds.

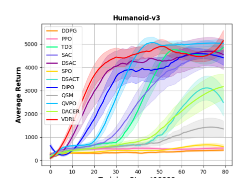
	METHOD	ANT-V3	HALFCHEETAH-V3	HOPPER-V3	HUMANOID-V3
Traditional Model-Free RL	PPO [55]	2129.33 ± 98.68	1757.57 ± 25.86	3291.76 ± 18.28	612.15 ± 117.34
	SPO [56]	2019.02 ± 46.91	4725.37 ± 102.64	3480.20 ± 92.66	749.70 ± 131.01
	DDPG [57]	1124.35 ± 51.49	8993.40 ± 140.39	3326.17 ± 12.22	597.52 ± 104.79
	TD3 [22]	2836.68 ± 41.87	8306.89 ± 82.31	3497.59 ± 4.39	5000.91 ± 13.38
	SAC [23]	4270.41 ± 130.00	10214.06 ± 98.44	3380.51 ± 8.14	5314.84 ± 43.05
Distributional RL	DSAC [24]	4264.95 ± 215.50	11610.33 ± 135.05	3436.95 ± 13.14	5009.60 ± 271.94
	DSACT [32]	5313.04 ± 122.46	11670.82 ± 131.81	3654.39 ± 7.06	5087.93 ± 165.00
Diffusion-based Policy RL	DIPO [4]	6224.93 ± 82.27	10052.71 ± 49.81	3454.60 ± 37.90	5266.87 ± 5.93
	QSM [5]	839.53 ± 510.14	9204.16 ± 85.61	3513.23 ± 22.65	1950.42 ± 1270.96
	QVPO [6]	5222.62 ± 78.02	9067.63 ± 72.47	3369.51 ± 439.56	5252.72 ± 13.75
	DACER [7]	5506.40 ± 99.67	11350.96 ± 119.48	3488.54 ± 2.79	4447.02 ± 750.42
	VDRL (Ours)	$\mathbf{7236.67 \pm 89.44}$	$\mathbf{11815.20 \pm 142.04}$	$\mathbf{3679.41 \pm 7.47}$	$\mathbf{5497.15 \pm 48.58}$
	METHOD	PUSHER-V2	SWIMMER-V3	WALKER2D-V3	INVERTED2PENDULUM-V2
Traditional Model-Free RL	PPO [55]	-22.92 ± 3.47	137.59 ± 1.35	3168.82 ± 104.73	9359.67 ± 0.21
	SPO [56]	-22.29 ± 4.99	136.06 ± 1.41	3849.82 ± 245.47	9359.53 ± 0.35
	DDPG [57]	-35.53 ± 4.94	42.64 ± 1.70	3478.72 ± 1185.37	9359.74 ± 0.22
	TD3 [22]	-23.06 ± 2.33	75.78 ± 0.86	3834.99 ± 64.33	92.13 ± 17.69
	SAC [23]	$\mathbf{-20.28 \pm 1.62}$	77.49 ± 3.07	4486.88 ± 42.63	9359.83 ± 0.15
Distributional RL	DSAC [24]	-31.43 ± 3.93	140.76 ± 9.20	4819.56 ± 171.21	9359.83 ± 0.08
	DSACT [32]	-25.65 ± 2.63	134.46 ± 1.96	5134.79 ± 22.68	9359.75 ± 0.11
Diffusion-based Policy RL	DIPO [4]	-39.60 ± 5.50	55.10 ± 2.01	4508.09 ± 10.41	9359.85 ± 0.16
	QSM [5]	-46.69 ± 3.41	105.04 ± 2.77	3897.01 ± 721.01	2475.43 ± 89.95
	QVPO [6]	-38.52 ± 5.74	124.14 ± 1.44	4198.86 ± 46.44	9354.38 ± 0.86
	DACER [7]	-24.91 ± 3.23	$\mathbf{151.55 \pm 1.04}$	4053.14 ± 62.19	9359.79 ± 0.14
	VDRL (Ours)	$\mathbf{-26.92 \pm 3.95}$	$\mathbf{144.40 \pm 2.27}$	$\mathbf{5223.77 \pm 10.30}$	$\mathbf{9359.89 \pm 0.16}$

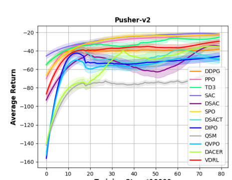
on simpler tasks such as Pusher-v2, Swimmer-v3, and InvertedPendulum-v2, while significantly outperforming them on more complex locomotion tasks, including Ant-v3, Hopper-v3, Humanoid-v3, and Walker2d-v3. The Figure 1 illustrates the learning curves of all algorithms, further demonstrating that VDRL delivers stable and consistently high performance across all evaluated tasks, highlighting its superior robustness. In contrast, baseline methods struggle with poor performance on one or more tasks, emphasizing the limitations of existing approaches.

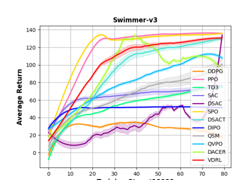

(a) Ant-v3

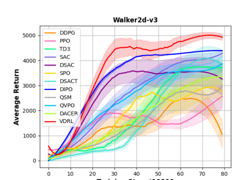

(b) HalfCheetah-v3


(c) Hopper-v3


(d) Humanoid-v3


(e) Pusher-v2


(f) Swimmer-v3


(g) Walker2d-v3

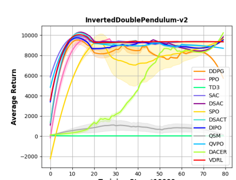

(h) Inverted2Pendulum-v2

Figure 1: Learning curves of the proposed method and all baselines on 8 continuous tasks of MuJoCo benchmark over 10 evaluation episodes every 10k iterations, where the solid lines correspond to the mean and the shaded regions to standard deviations over 4 different random seeds.

Table 2 shows the average value estimation bias, i.e., the difference between the estimated Q-value and the true Q-value, for VDRL and some baselines, including traditional model-free online RL methods and distributional RL approaches. The true Q-value is approximated by the average actual discounted cumulative return across states over ten episodes, evaluated every 10,000 iterations. Notably, the value estimation bias for the InvertedPendulum-v2 task is excluded, as effective policies are learned before the value function converges. The experimental results yield several key insights. First, compared to conventional RL algorithms, both VDRL and distributional RL baselines demonstrate reduced value estimation bias in most tasks, underscoring the effectiveness of value distribution learning in mitigating estimation inaccuracies. Second, VDRL consistently achieves lower bias than DASC and DSACT, highlighting the advantages of leveraging diffusion models to capture multimodal and complex value distributions for improved estimation accuracy. Last, even when compared to on-policy RL methods such as PPO and SPO, VDRL exhibits a significant advantage in value estimation accuracy across most tasks, further validating its effectiveness.

Table 2: Average value estimation bias over four runs for the proposed VDRL method, conventional model-free RL, and distributional RL baselines on eight tasks of MuJoCo benchmark. In particular, the bias is calculated as the difference between the estimated Q-value Q_{est} and the true Q-value Q_{true}, i.e., $Q_{\text{est}} - Q_{\text{true}}$, where the true Q-value is accessed based on the discounted accumulation of rewards. Furthermore, the name of each task are simplified because of the space limitation.

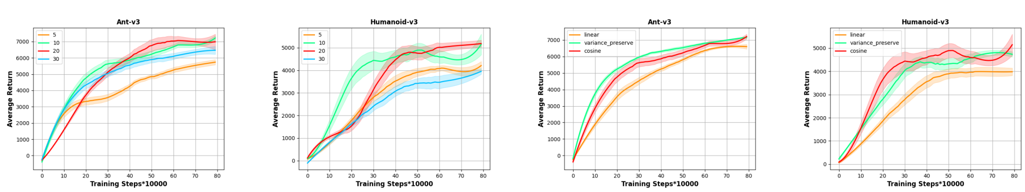
	METHOD	ANT3	HAL3	HOP3	HUM3	PUS2	SWI3	WAL3
Traditional on-policy RL	PPO [55]	9.45	320.36	271.29	16.74	-18.05	0.94	2.51
	SPO [56]	11.52	182.19	275.74	15.26	-21.35	0.88	3.60
Conventional off-policy RL	DDPG [57]	94.63	37.27	476.89	49.08	7.93	12.30	126.04
	TD3 [22]	-373.20	-372.05	-929.33	-225.06	-19.76	-3.07	-58.56
	SAC [23]	-27.94	-6.28	251.92	-92.48	-13.51	-1.06	-3.14
Distributional RL	DSAC [24]	37.94	83.47	2264.98	65.72	-5.15	1.84	69.93
	DSACT [32]	-25.17	36.76	-181.54	-59.24	$\mathbf{-4.73}$	-1.13	-9.80
	VDRL (Ours)	$\mathbf{-7.36}$	$\mathbf{-4.13}$	$\mathbf{-113.70}$	$\mathbf{-12.69}$	$\mathbf{-6.85}$	$\mathbf{0.42}$	$\mathbf{-0.95}$

5.4 Sensitivity Analysis

We perform sensitivity analysis on different diffusion steps and diffusion noise schedule, using the Ant-v3 and Humanoid-v3 tasks as examples.

Diffusion Steps. We evaluate the performance of VDRL under varying diffusion steps ($T = 5, 10, 20, 30$), as shown in Figure 2(a)-2(b). The results indicate that the performance does not improve monotonically with an increasing number of diffusion steps. Adopting larger diffusion steps degrades the performance of VDRL, potentially due to gradient vanishing or exploding issues. Additionally, larger diffusion steps will lead to increased training and evaluation costs. To balance the performance and computational efficiency, we choose 10 diffusion steps for all tasks.

Diffusion Noise Schedule. Figure 2(c)-2(d) illustrates the performance of VDRL with distinct diffusion noise schedules. It can be observed that the cosine and variance preserve schedule obtain comparable results, and both outperform the linear schedule. Thus, we select the cosine schedule for all experiments.



(a) Diffusion Steps (b) Diffusion Steps (c) Noise Schedule (d) Noise Schedule

Figure 2: Ablation learning curves for different diffusion steps and diffusion noise schedule on Ant-v3 and Humanoid-v3 tasks, where the solid lines correspond to the mean and the shaded regions to standard deviations over 4 different random seeds.

6 Conclusion and Discussion

In this work, we presented a novel model-free online RL method called value diffusion reinforcement learning (VDRL) to overcome the representational limitations of learned value distributions. By harnessing the inverse denoising process of diffusion models, VDRL captures expressive, multimodal value distributions, thereby substantially improving value estimation. The theoretical cornerstone of VDRL lies in its diffusion-based variational loss, which we rigorously establish as a tight variational lower bound for optimizing value distributions under the KL-divergence measurement. Furthermore, to improve the training stability and further enhance value estimation, we introduce double value diffusion learning with sample selection. Comprehensive experiments on eight continuous control tasks from the MuJoCo benchmark demonstrate the effectiveness and robustness of VDRL.

Limitations. While value diffusion learning presents a promising advance, its application to online RL faces several unresolved challenges. For instance, the reliance on iterative value diffusion sampling incurs significantly increased computational costs than traditional model-free RL and distributional RL methods, especially in high-dimensional action spaces. In addition, this work is evaluated primarily on MuJoCo benchmarks, leaving its scalability to real-world robotics applications with partial observability and sensor noise as an open question.

Future Work. In our future research, we will pursue two key avenues to advance this paradigm. First, we will develop lightweight diffusion models, potentially through knowledge distillation, to reduce the training and inference latency. Second, we will explore hybrid methods that integrate value diffusion with model-based planning, aiming to tackle the challenges of complex, long-horizon tasks and thereby unlock further performance gains in online RL.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (Grants No. 62476133) and the Fundamental Research Funds for the Central Universities (Grant No. 11300-312200502507).

References

[1] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pages 9902–9915. PMLR, 2022.

[2] Bogdan Mazoure, Walter Talbott, Miguel Angel Bautista, Devon Hjelm, Alexander Toshev, and Josh Susskind. Value function estimation using conditional diffusion models for control. *arXiv preprint arXiv:2306.07290*, 2023.

[3] Zhengbang Zhu, Minghuan Liu, Liyuan Mao, Bingyi Kang, Minkai Xu, Yong Yu, Stefano Ermon, and Weinan Zhang. Madiff: Offline multi-agent learning with diffusion models. In *Advances in Neural Information Processing Systems*, volume 37, pages 4177–4206, 2024.

[4] 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.

[5] Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. In *International Conference on Machine Learning*, pages 41163–41182. PMLR, 2024.

[6] Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. In *Advances in Neural Information Processing Systems*, volume 37, pages 54183–54204, 2024.

[7] Yinuo Wang, Likun Wang, Yuxuan Jiang, Wenjun Zou, Tong Liu, Xujie Song, Wenxuan Wang, Liming Xiao, Jiang Wu, Jingliang Duan, et al. Diffusion actor-critic with entropy regulator. In *Advances in Neural Information Processing Systems*, volume 37, pages 54183–54204, 2024.

[8] Xiaoyu Huang, Yufeng Chi, Ruofeng Wang, Zhongyu Li, Xue Bin Peng, Sophia Shao, Borivoje Nikolic, and Koushil Sreenath. Diffuseloco: Real-time legged locomotion control with diffusion from offline datasets. *arXiv preprint arXiv:2404.19264*, 2024.

[9] Gengyu Zhang, Hao Tang, and Yan Yan. Versatile navigation under partial observability via value-guided diffusion policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17943–17951, 2024.

[10] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[11] Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *arXiv preprint arXiv:2402.10885*, 2024.

[12] Paul Maria Scheikl, Nicolas Schreiber, Christoph Haas, Niklas Freymuth, Gerhard Neumann, Rudolf Lioutikov, and Franziska Mathis-Ullrich. Movement primitive diffusion: Learning gentle robotic manipulation of deformable objects. *IEEE Robotics and Automation Letters*, 2024.

[13] Jascha Sohl-Dickstein, Eric A Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, volume 37, pages 2256–2265. PMLR, 2015.

[14] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, volume 32, pages 11918–11930, 2019.

[15] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Proceedings of the 34rd International Conference on Neural Information Processing Systems*, 33:6840–6851, 2020.

[16] Zhengbang Zhu, Hanye Zhao, Haoran He, Yichao Zhong, Shenyu Zhang, Haoquan Guo, Tingting Chen, and Weinan Zhang. Diffusion models for reinforcement learning: A survey, 2024.

[17] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[18] Hado van Hasselt. Double q-learning. In *Proceedings of the 24th International Conference on Neural Information Processing Systems-Volume 2*, pages 2613–2621, 2010.

[19] Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 connectionist models summer school*, pages 255–263, 1993.

[20] 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*, pages 2094–2100, 2016.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] Xiangkun He, Wenhui Huang, and Chen Lv. Toward trustworthy decision-making for autonomous vehicles: A robust reinforcement learning approach with safety guarantees. *Engineering*, 33:77–89, 2024.

[26] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pages 449–458. PMLR, 2017.

[27] Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning. In *International conference on machine learning*, pages 1096–1105. PMLR, 2018.

[28] Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI conference on artificial intelligence-Volume 32*, pages 2892–2901, 2018.

[29] Borislav Mavrin, Hengshuai Yao, Linglong Kong, Kaiwen Wu, and Yaoliang Yu. Distributional reinforcement learning for efficient exploration. In *International conference on machine learning*, pages 4424–4434. PMLR, 2019.

[30] Mark Rowland, Robert Dadashi, Saurabh Kumar, Rémi Munos, Marc G Bellemare, and Will Dabney. Statistics and samples in distributional reinforcement learning. In *International Conference on Machine Learning*, pages 5528–5536. PMLR, 2019.

[31] Daniel Brown, Scott Niekum, and Marek Petrik. Bayesian robust optimization for imitation learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 2479–2491, 2020.

[32] Jingliang Duan, Wenxuan Wang, Liming Xiao, Jiaxin Gao, Shengbo Eben Li, Chang Liu, Ya-Qin Zhang, Bo Cheng, and Keqiang Li. Distributional soft actor-critic with three refinements. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(5):3935–3946, 2025.

[33] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[34] Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, TB Dhruva, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. In *International Conference on Learning Representations*, 2018.

[35] Chen Tessler, Guy Tennenholtz, and Shie Mannor. Distributional policy optimization: an alternative approach for continuous control. In *Advances in Neural Information Processing Systems*, volume 32, pages 1352–1362, 2019.

[36] Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pages 1747–1756. PMLR, 2016.

[37] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems*, pages 8780–8794, 2021.

[38] 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.

[39] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[40] Yuanzhi Wang, Yong Li, and Zhen Cui. Incomplete multimodality-diffused emotion recognition. *Advances in Neural Information Processing Systems*, 36:17117–17128, 2023.

[41] Yuanzhi Wang, Yong Li, Mengyi Liu, Xiaoya Zhang, Xin Liu, Zhen Cui, and Antoni B Chan. Re-attentional controllable video diffusion editing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 8123–8131, 2025.

[42] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[43] Liming Xiao, Yao Lyu, Fawang Zhang, Liangfa Chen, Guangyuan Yu, Shengbo Eben Li, Fei Ma, and Jingliang Duan. Multi-style distributional soft actor-critic: Learning a unified policy for diverse control behaviors. *IEEE Transactions on Intelligent Vehicles*, pages 1–12, 2024.

[44] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[45] Will Dabney, Zeb Kurth-Nelson, Naoshige Uchida, Clara Kwon Starkweather, Demis Hassabis, Rémi Munos, and Matthew Botvinick. A distributional code for value in dopamine-based reinforcement learning. *Nature*, 577(7792):671–675, 2020.

[46] Marc G Bellemare, Will Dabney, and Mark Rowland. *Distributional reinforcement learning*. MIT Press, 2023.

[47] Yangang Ren, Jingliang Duan, Shengbo Eben Li, Yang Guan, and Qi Sun. Improving generalization of reinforcement learning with minimax distributional soft actor-critic. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2020.

[48] Yash Chandak, Scott Niekum, Bruno da Silva, Erik Learned-Miller, Emma Brunskill, and Philip S Thomas. Universal off-policy evaluation. In *Advances in Neural Information Processing Systems*, volume 34, pages 27475–27490, 2021.

[49] Wenhao Li, Xiangfeng Wang, Bo Jin, and Hongyuan Zha. Hierarchical diffusion for offline decision making. In *International Conference on Machine Learning*, volume 202, pages 20035–20064. PMLR, 2023.

[50] Fei Ni, Jianye Hao, Yao Mu, Yifu Yuan, Yan Zheng, Bin Wang, and Zhixuan Liang. Metadiffuser: Diffusion model as conditional planner for offline meta-rl. In *International Conference on Machine Learning*, pages 26087–26105. PMLR, 2023.

[51] Zhixuan Liang, Yao Mu, Mingyu Ding, Fei Ni, Masayoshi Tomizuka, and Ping Luo. Adaptdiffuser: Diffusion models as adaptive self-evolving planners. In *International Conference on Machine Learning*, pages 20725–20745. PMLR, 2023.

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

[53] Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

[54] Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In *International Conference on Learning Representations*, 2023.

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

[56] Zhengpeng Xie, Qiang Zhang, Fan Yang, Marco Hutter, and Renjing Xu. Simple policy optimization. *arXiv preprint arXiv:2401.16025*, 2025.

[57] 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.

[58] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. `https://github.com/openai/baselines`, 2017.

A Proof

Theorem 1. ***(Lower Bound of Learning Objective for Value Distribution)*** *If the value distribution is modeled by a diffusion model, the variational bound objective*

$$\mathbb{E}_{\substack{(s,a,r,s')\sim\mathcal{B},a'\sim\pi_{\bar{\phi}},\\ Z(s',a')\sim\mathcal{Z}_{\bar{\theta}}(\cdot|s',a')}}\left[\mathbb{E}_{\mathcal{T}^\pi Z_{1:T}\sim q(\mathcal{T}^\pi Z_{1:T}|s,a,\mathcal{T}^\pi Z_0)}\left[\log\frac{P_\theta(\mathcal{T}^\pi Z_{0:T}|s,a)}{q(\mathcal{T}^\pi Z_{1:T}|s,a,\mathcal{T}^\pi Z_0)}\right]\right], \tag{14}$$

is the tight lower bound of the optimization objective of value distribution learning for distributional RL under the KL-divergence measurement

$$\mathbb{E}_{(s,a)\sim\mathcal{B}}\left[D_{KL}(\mathcal{T}^\pi\mathcal{Z}_{\bar{\theta}}(\cdot|s,a)\parallel\mathcal{Z}_\theta(\cdot|s,a))\right], \tag{15}$$

where $\mathcal{Z}_\theta(\cdot|s,a)$ denotes the value distribution to be optimized and $\mathcal{Z}_{\bar{\theta}}(\cdot|s,a)$ is the parameterized target value distribution, and the equality holds when the policy converges.

Proof. The optimization objective of value distribution learning under the KL-divergence measurement is

$$\min_\theta\left\{\mathop{\mathbb{E}}_{(s,a)\sim\mathcal{B}}\left[D_{\mathrm{KL}}(\mathcal{T}^\pi\mathcal{Z}_{\bar{\theta}}(\cdot|s,a)\parallel\mathcal{Z}_\theta(\cdot|s,a))\right]\right\}, \tag{16}$$

where $\mathcal{B}$ is the replay buffer of historical samples, $\mathcal{Z}_\theta(\cdot|s,a))$ denotes the value distribution to be optimized, and $\mathcal{Z}_{\bar{\theta}}(\cdot|s,a))$ is the parameterized target value distribution. According to the definition of KL-divergence, we can show that

$$\begin{aligned}
J_{\mathcal{Z}}(\theta) &= \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{B}}\left[D_{\mathrm{KL}}(\mathcal{T}^\pi\mathcal{Z}_{\bar{\theta}}(\cdot|s,a)\parallel\mathcal{Z}_\theta(\cdot|s,a))\right]\\
&= \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{B}}\left[\int_{\mathcal{T}^\pi Z(s,a)} P_{\bar{\theta}}(\mathcal{T}^\pi Z(s,a))\log\frac{P_{\bar{\theta}}(\mathcal{T}^\pi Z(s,a))}{P_\theta(\mathcal{T}^\pi Z(s,a))}\right]\\
&= -\mathop{\mathbb{E}}_{(s,a)\sim\mathcal{B}}\left[\int_{\mathcal{T}^\pi Z(s,a)} P_{\bar{\theta}}(\mathcal{T}^\pi Z(s,a))\log P_\theta(\mathcal{T}^\pi Z(s,a))\right]+c\\
&= -\mathop{\mathbb{E}}_{(s,a)\sim\mathcal{B}}\left[\mathbb{E}_{\mathcal{T}^\pi Z(s,a)\sim\mathcal{T}^\pi\mathcal{Z}_{\bar{\theta}}(\cdot|s,a)}\left[\log P_\theta(\mathcal{T}^\pi Z(s,a))\right]\right]+c\\
&= -\mathop{\mathbb{E}}_{\substack{(s,a,r,s')\sim\mathcal{B},a'\sim\pi_{\bar{\phi}},\\ Z(s',a')\sim\mathcal{Z}_{\bar{\theta}}(\cdot|s',a')}}\left[\log P_\theta(\mathcal{T}^\pi Z(s,a))\right],
\end{aligned} \tag{17}$$

where the constant c denotes the term independent of θ, $\pi_{\bar{\phi}}$ denotes the parameterized target policy, and $\mathcal{T}^\pi Z(s,a)=R(s,a)+\gamma Z(s',a')$. For simplicity of exposition, we let Y denotes $\mathcal{T}^\pi Z(s,a)$. Thus, the optimization objective of value distribution learning can be formalized as follows:

$$\max_\theta\left\{\mathop{\mathbb{E}}_{\substack{(s,a,r,s')\sim\mathcal{B},a'\sim\pi_{\bar{\phi}},\\ Z(s',a')\sim\mathcal{Z}_{\bar{\theta}}(\cdot|s',a')}}\left[\log P_\theta(Y)\right]\right\}. \tag{18}$$

If $\mathcal{Z}_\theta(\cdot|s,a)$ is a diffusion-based value distribution, we have

$$\begin{aligned}
\log P_\theta(Y) =&\mathbb{E}_{Y_{1:T}\sim q(Y_{1:T}|s,a,Y_0)}\left[\log\frac{P_\theta(Y_{1:T}|s,a,Y_0)}{P_\theta(Y_{1:T}|s,a,Y_0)}P_\theta(Y_0)\right]\\
=&\mathbb{E}_{Y_{1:T}\sim q(Y_{1:T}|s,a,Y_0)}\left[\log\frac{P_\theta(Y_{0:T}|s,a,Y_0)P_\theta(Y_0)}{P_\theta(Y_{1:T}|s,a,Y_0)}\right]\\
=&\mathbb{E}_{Y_{1:T}\sim q(Y_{1:T}|s,a,Y_0)}\left[\log\frac{P_\theta(Y_{0:T}|s,a)}{q(Y_{1:T}|s,a,Y_0)}\frac{q(Y_{1:T}|s,a,Y_0)}{P_\theta(Y_{1:T}|s,a,Y_0)}\right]\\
=&\mathbb{E}_{Y_{1:T}\sim q(Y_{1:T}|s,a,Y_0)}\left[\log\frac{P_\theta(Y_{0:T}|s,a)}{q(Y_{1:T}|s,a,Y_0)}\right]+D_{\mathrm{KL}}(q(Y_{1:T}|s,a,Y_0)\parallel P_\theta(Y_{1:T}|s,a,Y_0)).
\end{aligned} \tag{19}$$

According to the inequality of Jensen, the term $D_{\text{KL}}(q(Y_{1:T}|s,a,Y_0) \parallel P_\theta(Y_{1:T}|s,a,Y_0))$ in Eq. (19) can be shown that

$$\begin{aligned}
D_{\text{KL}}(q(Y_{1:T}|s,a,Y_0) \parallel P_\theta(Y_{1:T}|s,a,Y_0)) =& \mathbb{E}_{Y_{1:T}\sim q(Y_{1:T}|s,a,Y_0)}\left[\log\frac{q(Y_{1:T}|s,a,Y_0)}{P_\theta(Y_{1:T}|s,a,Y_0)}\right] \\
=& \mathbb{E}_{Y_{1:T}\sim q(Y_{1:T}|s,a,Y_0)}\left[-\log\frac{P_\theta(Y_{1:T}|s,a,Y_0)}{q(Y_{1:T}|s,a,Y_0)}\right] \\
\geq& -\log\left(\mathbb{E}_{Y_{1:T}\sim q(Y_{1:T}|s,a,Y_0)}\left[\frac{P_\theta(Y_{1:T}|s,a,Y_0)}{q(Y_{1:T}|s,a,Y_0)}\right]\right).
\end{aligned} \tag{20}$$

When the learning policy converges, the term $\frac{P_\theta(Y_{1:T}|s,a,Y_0)}{q(Y_{1:T}|s,a,Y_0)}$ is equal to 1, and we can show that

$$\begin{aligned}
J_{\mathcal{Z}}(\theta) =& \mathop{\mathbb{E}}_{(s,a)\sim\mathcal{B}}\left[D_{\text{KL}}(\mathcal{T}^\pi \mathcal{Z}_{\bar{\theta}}(\cdot|s,a) \parallel \mathcal{Z}_\theta(\cdot|s,a))\right] \\
=& -\mathop{\mathbb{E}}_{\substack{(s,a,r,s')\sim\mathcal{B},a'\sim\pi_{\bar{\phi}},\\ Z(s',a')\sim\mathcal{Z}_{\bar{\theta}}(\cdot|s',a')}}\left[\log P_\theta(Y)\right] \\
\geq& -\mathop{\mathbb{E}}_{\substack{(s,a,r,s')\sim\mathcal{B},a'\sim\pi_{\bar{\phi}},\\ Z(s',a')\sim\mathcal{Z}_{\bar{\theta}}(\cdot|s',a')}}\left[\mathbb{E}_{Y_{1:T}\sim q(Y_{1:T}|s,a,Y_0)}\left[\log\frac{P_\theta(Y_{0:T}|s,a)}{q(Y_{1:T}|s,a,Y_0)}\right]\right]
\end{aligned} \tag{21}$$

Thus, the variational bound objective

$$\mathbb{E}_{\substack{(s,a,r,s')\sim\mathcal{B},a'\sim\pi_{\bar{\phi}},\\ Z(s',a')\sim\mathcal{Z}_{\bar{\theta}}(\cdot|s',a')}}\left[\mathbb{E}_{\mathcal{T}^\pi Z_{1:T}\sim q(\mathcal{T}^\pi Z_{1:T}|s,a,\mathcal{T}^\pi Z_0)}\left[\log\frac{P_\theta(\mathcal{T}^\pi Z_{0:T}|s,a)}{q(\mathcal{T}^\pi Z_{1:T}|s,a,\mathcal{T}^\pi Z_0)}\right]\right], \tag{22}$$

is the tight lower bound of the learning objective of value distribution for distributional RL under the KL-divergence measurement

$$\mathbb{E}_{(s,a)\sim\mathcal{B}}\left[D_{\text{KL}}(\mathcal{T}^\pi \mathcal{Z}_{\bar{\theta}}(\cdot|s,a) \parallel \mathcal{Z}_\theta(\cdot|s,a))\right]. \tag{23}$$

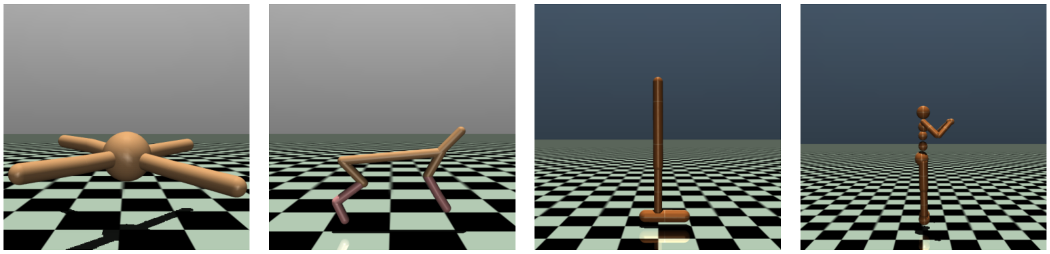
This completes the proof of Theorem 1. □

B Environmental Details

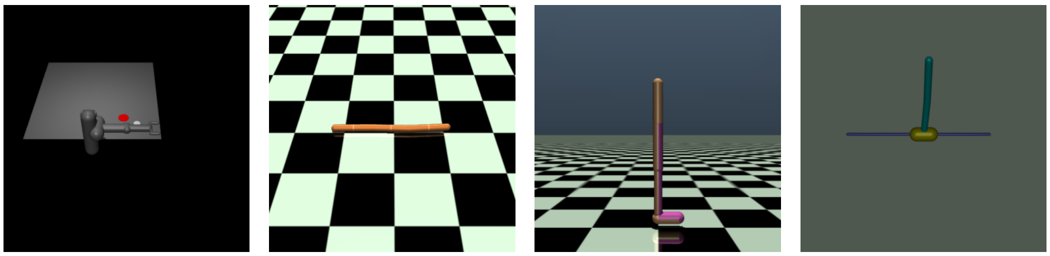
MuJoCo (Multi-Joint dynamics with Contact) [33] is a high-performance physics engine widely used in robotics and biomechanics simulations, renowned for its efficiency and accuracy in modeling complex dynamics. It provides a versatile platform for developing and benchmarking reinforcement learning (RL) algorithms, enabling the simulation of articulated bodies with intricate interactions, including friction, contact, and soft constraints. These features make it particularly suitable for tasks involving multi-joint movement and physical interactions.

In this work, we use eight MuJoCo benchmark tasks, categorized into three classes based on their dynamics and objectives: **locomotion**, **control**, and **navigation**. Specifically, the locomotion tasks focus on the ability of agents to move efficiently across a terrain, including Ant-v3, HalfCheetah-v3, Hopper-v3, Humanoid-v3, and Walker2d-v3. The control tasks, such as Inverted2Pendulum-v2 and Pusher-v2, require agents to manipulate objects or maintain certain positions. The navigation task, exemplified by Swimmer-v3, evaluates the agent's ability to move through a fluid medium. Figure 3 shows the visualization of all tasks used in this study. Furthermore, these tasks are detailed as follows:

- **Ant-v3**: $(s \times a) \in \mathbb{R}^{111} \times \mathbb{R}^8$, a four-legged robot learns to navigate and cover distance by coordinating its legs, with key challenges in balancing and adapting to varying terrains.
- **HalfCheetah-v3**: $(s \times a) \in \mathbb{R}^{17} \times \mathbb{R}^6$, a bipedal robot emulates the running motion of a cheetah, with the objective to maximize forward speed while maintaining stability.
- **Hopper-v3**: $(s \times a) \in \mathbb{R}^{11} \times \mathbb{R}^3$, a single-legged robot learns to hop forward efficiently, focusing on balance and joint control to maximize distance.
- **Humanoid-v3**: $(s \times a) \in \mathbb{R}^{376} \times \mathbb{R}^{17}$, a humanoid robot must learn to walk and run, tackling the complex challenge of maintaining balance during motion.



(a) Ant-v3 (b) HalfCheetah-v3 (c) Hopper-v3 (d) Humanoid-v3



(e) Pusher-v2 (f) Swimmer-v3 (g) Walker2d-v3 (h) Inverted2Pendulum-v2

Figure 3: The screenshots of MuJoCo tasks utilized in this work.

- **Walker2d-v3**: $(s \times a) \in \mathbb{R}^{17} \times \mathbb{R}^{6}$, a two-legged robot learns to walk forward, emphasizing coordination and adaptability in dynamic environments.
- **Inverted2Pendulum-v2**: $(s \times a) \in \mathbb{R}^{6} \times \mathbb{R}^{1}$, a classic control task where the agent balances an inverted pendulum on a pivot, requiring precise application of forces to maintain stability.
- **Pusher-v2**: $(s \times a) \in \mathbb{R}^{23} \times \mathbb{R}^{7}$, the agent controls a robot arm to push a block to a target location, testing precision in force application and obstacle navigation.
- **Swimmer-v3**: $(s \times a) \in \mathbb{R}^{8} \times \mathbb{R}^{2}$, a two-dimensional swimmer propels itself forward by coordinating body segments, emphasizing control over body dynamics in a fluid-like environment.

C Hyperparameters

The setting of all hyperparameters for the training is presented in Table 3.

Table 3: Hyperparameters for training.

name	Value	Description
Optim	Adam	the optimizer of method
n_iteration	8×10^5	Maximum iteration steps until the end of training
buffer_size	1×10^6	capacity of replay buffer
batch_size	256	number of samples from each update
evaluate_cycle	10000	how often to evaluate the model
No. of hidden layers	2	the number of hidden layers
No. of hidden nodes	256	the number of hidden nodes
Activation_π	GeLU	the activation of policy
Activation_$\mathcal{Z}_i$	Mish	the activation of value distributions
lr_ϕ	3×10^{-4}	learning rate for policy
lr_{θ_i}	3×10^{-4}	learning rate for value distributions
τ	0.005	the target momentum coefficient
γ	0.99	discount factor
T	10	diffusion step

D Details of the baseline methods

In this section, we discuss details of the baseline methods with which we compare our method, including the implementation and hyperparameters of each algorithm.

DDPG [57] We use the high-quality implementations of reinforcement learning algorithms, i.e., OpenAI Baselines [58]

`https://github.com/openai/baselines`

with configuration `baselines/ddpg/ddpg.py` to evaluate the DDPG baseline on the MuJoCo benchmark.

PPO [55]. We use the code from the authors' repository

`https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail`

for their evaluation in continuous control tasks, i.e., the MuJoCo benchmark. We employ their configuration `/a2c_ppo_acktr/arguments.py`.

TD3 [22]. We use the official code from

`https://github.com/sfujim/TD3`

with hyper-parameters from `main.py` for the MuJoCo experiments.

SAC [23]. We utilize the code from

`https://github.com/toshikwa/soft-actor-critic.pytorch`

with their configuration `code/main.py` for the MuJoCo continuous control tasks in OpenAI gym, which is the PyTorch implementation of Soft Actor-Critic.

DSAC [24]. We use the code from

`https://github.com/Jingliang-Duan/DSAC-v1`

with their configuration `example_train/main.py` for the evaluation of MuJoCo continuous control tasks, which is the PyTorch implementation of Distributional Soft Actor-Critic.

DSACT [23]. We utilize the code from

`https://github.com/Jingliang-Duan/DSAC-v2`

with their configuration `example_train/dsacv2_mlp_mujoco_offserial.py` for the MuJoCo benchmark, which is the PyTorch implementation of Distributional Soft Actor-Critic with Three Refinements.

SPO [56]. We use the official code from

`https://github.com/MyRepositories-hub/Simple-Policy-Optimization`

with hyper-parameters from `mujoco/main.py`.

DIPO [4]. We utilize the code from the authors' repository

`https://github.com/BellmanTimeHut/DIPO`

with hyper-parameters from `main.py`.

QSM [5]. We provide a PyTorch implementation of the QSM baseline based on the authors' repository, which is built on top of a re-implementation of the JAXRL framework

`https://github.com/Alescontrela/score_matching_rl`

with hyper-parameters from `examples/states/configs/score_matching_config.py`.

QVPO [6]. We use the official code from `https://github.com/wadx2019/qvpo`

with hyper-parameters from `main.py`.

DACER [7]. For a fair comparison, we provide an implementation of the DACER method with PyTorch, which is based on the official code (JAX) from

`https://github.com/happy-yan/DACER-Diffusion-with-Online-RL`

with hyper-parameters from `scripts/train_mujoco.py`.

E Training and Inference Time

The training and inference time of VDRL and all baselines on three tasks of MuJoCo, i.e., Ant-v3, Humanoid-v3, and Walker2d-v3, are shown in the Table 4-5. In particular, Notably, the official code of the QSM and DACER methods is based on the JAX framework, but the implementation of other baselines and our method is based on PyTorch. In practice, the algorithm realized with JAX is 6-10 times faster than that realized with PyTorch [6]. To ensure a fair comparison, we utilize the implementation of the QSM and DACER methods with PyTorch, which is based on the official code (JAX). Furthermore, the number of diffusion steps for all diffusion-based online RL baselines (DIPO, QSM, QVPO, and DACER) is set to 20.

Table 4: The training time (h) comparison on three tasks of MuJoCo Benchmarks.

	VDRL	**DIPO**	**QSM**	**QVPO**	**DACER**	*DSAC*	*DSACT*	PPO	SAC
Ant-v3	7.9	11.1	10.2	9.1	9.8	5.3	5.8	0.4	3.8
Humanoid-v3	8.2	11.5	10.7	9.4	10.1	5.5	6.1	0.5	3.9
Walker2d-v3	7.8	11.0	9.9	8.9	9.6	5.2	5.7	0.4	3.8

Table 5: The inference time (ms) comparison on three tasks of MuJoCo Benchmarks.

	VDRL	**DIPO**	**QSM**	**QVPO**	**DACER**	*DSAC*	*DSACT*	PPO	SAC
Ant-v3	3.6	5.8	6.4	6.2	5.4	1.9	2.1	0.2	0.4
Humanoid-v3	3.7	6.0	6.5	6.3	5.5	2.0	2.2	0.2	0.5
Walker2d-v3	3.6	5.7	6.3	6.1	5.3	1.9	2.1	0.2	0.4

From the results of the above tables, we can observe:

i) The training and inference time of VDRL is faster than the diffusion-based online RL methods (i.e., DIPO, QSM, QVPO, and DACER), because the lower diffusion steps of VDRL (T=10) compared with diffusion-based RL baselines (T=20) reduce the computational complexity.

ii) All diffusion-based RL methods are slower than distributional (i.e., DSAC and DSACT) or traditional (i.e., SAC and PPO) ones, due to the iterative denoising process with some steps. Although VDRL is slower than the traditional or distributional RL methods in inference, the inference times (3.7ms) are still tolerable in a real-time application. As discussed in QVPO [6], most existing real robots only require a 50-Hz control policy, i.e., output action per 20 ms. Moreover, the training and inference time of VDRL can be further reduced by using the JAX framework if it is necessary, like the official code of QSM and DACER methods. Hence, the inference time is not a bottleneck to applying our method to real-time applications.

Overall, the performance improvements of VDRL are relatively worth it with appropriate longer training and inference time.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

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

   Answer: [Yes]

   Justification: The contributions and scope of this paper are clearly delineated in the abstract and introduction sections.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

Justification: The limitations of this work are discussed in Section 6.

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: [Yes]

   Justification: The complete proof is deferred to Appendix A.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The proposed algorithm of the training process is presented in Section 3.3. The intricate implementation details of and training hyperparameters are deferred to Section 5.2 and Appendix C.

   Guidelines:

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

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

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

   Answer: [No]

   Justification: The proposed algorithm of the training process is presented in Section 3.3. The intricate implementation details of and training hyperparameters are deferred to Section 5.2 and Appendix C.

   Guidelines:

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

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: The intricate implementation details of and training hyperparameters are deferred to Section 5.2 and Appendix C. The more details of MuJoCo are introduced in Appendix B.

   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 experimental results are based on four training runs with different random seeds in the experiments.

   Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

Justification: The information on computer resources needed to reproduce the experiments is provided in Section 5. Specifically, the all experiments are conducted on a 2.90GHz Intel Core i7-10700 CPU, 64G RAM, and NVIDIA GeForce RTX 3090 GPU.

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 computing 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: In every respect, the research conducted in the paper conforms with the NeurIPS Code of Ethics.

   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: [NA]

    Justification: This paper is foundational research and not tied to particular applications.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [NA]

    Justification: This paper poses no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: This work cites the original paper that produced the code package. The details are deferred to Appendix D.

    Guidelines:

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

13. **New assets**

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

    Answer: [NA]

    Justification: There are no new assets in this paper.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.

- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

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

    Answer: [NA]

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

    Guidelines:

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

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

    Answer: [NA]

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

    Guidelines:

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

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.