Broader Impact

In this paper, we propose an efficient yet powerful policy class for offline reinforcement learning. We show that this method is superior to most existing methods on simulated robotic tasks. However, some war robots or weapon robots might employ our EDP to learn strategic agents considering the generalization ability of EDP. Depending on the specific application scenarios, it might be harmful to domestic privacy and safety.

A Reinforcement Learning Algorithms

In this section, we introduce the details of the RL algorithms experimented.

A.1 TD3+BC

TD3 [8] is a popular off-policy RL algorithm for continuous control. TD3 improves DDPG [18] by addressing the value overestimation issue. Specifically, TD3 adopts a double Q-learning paradigm that computes the TD-target $\hat{Q}(\boldsymbol{s},\boldsymbol{a})$ as

$$\hat{Q}(\boldsymbol{s},\boldsymbol{a}) = r(\boldsymbol{s},\boldsymbol{a}) + \gamma \min(Q_1(\boldsymbol{s}',\boldsymbol{a}'), Q_2(\boldsymbol{s}',\boldsymbol{a}')), \quad \boldsymbol{a}' = \pi(\boldsymbol{s}'), \tag{14}$$

where $\pi(\boldsymbol{s}')$ is a deterministic policy, Q_1 and Q_2 are two independent value networks. Specifically, TD3 takes **only** Q_1 for policy improvement

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[Q_1(\boldsymbol{s},\hat{\boldsymbol{a}})\right], \quad \hat{a} = \pi(\boldsymbol{s}). \tag{15}$$

Built on top of TD3, TD3+BC simply adds an additional behavior cloning term for its policy improvement

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{s,a\sim\mathcal{D}}\left[Q(\boldsymbol{s},\hat{\boldsymbol{a}}) - \alpha(\hat{a}-a)^2\right], \quad \hat{\boldsymbol{a}} = \pi(\boldsymbol{s}), \tag{16}$$

where α is a hyper-parameter that balances these two terms. However, as the scale of Q-values are different from the behavior cloning loss, TD3+BC normalizes it with $\frac{1}{N}\sum\limits_{i=1}^{N}|Q(\boldsymbol{s_i},\hat{\boldsymbol{a}}_i)|$ for numerical stability.

In the context of EDP, we observe that 1) the behavior cloning term can be naturally achieved as a diffusion loss as defined in Eqn. 5; 2) the sampled action $\hat{\boldsymbol{a}}$ is replaced by action approximation as in Eqn. 9 for efficient policy improvement. Different from the original TD3+BC that uses only Q_1 for policy improvement, we sample from Q_1 and Q_2 with equal probability for each policy improvement step.

A.2 Critic Regularized Regression

CRR follows Advantage Weighted Regression (AWR) [27], which is a simple yet effective off-policy RL method, for policy improvement. Specifically, AWR considers a constrained policy improvement step

$$\arg\max_{\pi} \int_{\boldsymbol{s}} d_\mu(\boldsymbol{s}) \int_{\boldsymbol{a}} \pi(\boldsymbol{a}\mid\boldsymbol{s}) A(\boldsymbol{s},\boldsymbol{a}) d\boldsymbol{a} d\boldsymbol{s}, \quad \text{s.t.} \quad \int_{s} d_\mu(\boldsymbol{s}) D_{\mathrm{KL}}\left[\pi(\cdot\mid\boldsymbol{s}) \,\|\, \mu(\cdot\mid\boldsymbol{s})\right] \le \epsilon, \tag{17}$$

where $A(\boldsymbol{s},\boldsymbol{a})$ is the advantage function, μ is a behavior policy that is used to generate trajectories during off-policy RL, $d_\mu(\boldsymbol{s})$ is the state distribution induced by μ, D_{KL} is the KL divergence, and ϵ is a threshold parameter. This constrained optimization problem can be solved in closed form, which gives an optimal policy of

$$\pi^*(\boldsymbol{a}\mid\boldsymbol{s}) = \frac{1}{Z(\boldsymbol{s})}\mu(\boldsymbol{a}\mid\boldsymbol{s}) \exp\left(\frac{1}{\beta}A(\boldsymbol{s},\boldsymbol{a})\right), \tag{18}$$

with $Z(\boldsymbol{s})$ being the partition function and β is a hyper-parameter. For policy improvement, AWR simply distills the one-step improved optimal policy to the learning policy π by minimizing the

KL-divergence

$$\begin{aligned}&\arg\min_{\pi} \mathbb{E}_{\boldsymbol{s}\sim d_{\mu}(\boldsymbol{s})}\left[D_{\mathrm{KL}}\left[\pi^*(\cdot \mid \boldsymbol{s}) \parallel \pi(\cdot \mid \boldsymbol{s})\right]\right]\\ =&\arg\max_{\pi} \mathbb{E}_{\boldsymbol{s}\sim d_{\mu}(\boldsymbol{s}),\boldsymbol{a}\sim\mu(\cdot|\boldsymbol{s})}\left[\log \pi(\boldsymbol{a} \mid \boldsymbol{s}) \exp(\frac{1}{\beta}A(\boldsymbol{s},\boldsymbol{a}))\right]\end{aligned} \tag{19}$$

CRR performs policy improvement in the same way as AWR, which simply replaces the sampling distribution with a fixed dataset

$$\arg\max_{\pi} \mathbb{E}_{\boldsymbol{s},\boldsymbol{a}\sim\mathcal{D}}\left[\log \pi(\boldsymbol{a} \mid \boldsymbol{s}) \exp(\frac{1}{\beta}A(\boldsymbol{s},\boldsymbol{a}))\right]. \tag{20}$$

As a result, this naturally imposes an implicit constraint on its policy improvement step.

However, as computing the log-likelihood $\log \pi(a \mid s)$ is intractable in diffusion models, in practical implementations, we use Eqn. 13 instead. In addition, we compute the advantage by

$$A(\boldsymbol{s},\boldsymbol{a}) = \min(Q_1(\boldsymbol{s},\boldsymbol{a}) - Q_2(\boldsymbol{s},\boldsymbol{a})) - \frac{1}{N}\sum_{i=1}^{N}\min(Q_1(\boldsymbol{s},\hat{\boldsymbol{a}}_i), Q_2(\boldsymbol{s},\hat{\boldsymbol{a}}_i)), \tag{21}$$

where $\hat{a}_i \sim \mathcal{N}(\hat{\boldsymbol{a}}^0, \boldsymbol{\sigma})$ is a sampled action with the mean of approximated action and an additional standard deviation. In our experiment, we found using a fixed standard deviation, an identity matrix of $\boldsymbol{\sigma} = \mathbb{I}$, $\beta = 1$, and sample size $N = 10$ generally produce a good performance.

A.3 Implicit Q Learning

Similar to CRR, IQL also adopts the AWR-style policy improvement that naturally imposes a constraint to encourage the learning policy to stay close to the behavior policy that generates the dataset. Different from CRR query novel and potentially out-of-distribution (OOD) actions when computing advantages, IQL aims to completely stay in-distribution with only dataset actions, while maintaining the ability to perform effective multi-step dynamic programming during policy evaluation. IQL achieves this by introducing an additional value function $V(\boldsymbol{s})$ and performing expectile regression. Specifically, the policy evaluation in IQL is implemented as

$$\begin{aligned}&\min_{V} \mathbb{E}_{\boldsymbol{s},\boldsymbol{a}\sim\mathcal{D}}\left[L_2^{\tau}(Q(\boldsymbol{s},\boldsymbol{a})) - V(\boldsymbol{s})\right]\\ &\min_{Q} \mathbb{E}_{\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}'\sim\mathcal{D}}\left[(r(\boldsymbol{s},\boldsymbol{a}) + \gamma V(\boldsymbol{s}') - Q(\boldsymbol{s},\boldsymbol{a}))^2\right],\end{aligned} \tag{22}$$

where L_2^{τ} is the expectile regression loss defined as $L_2^{\tau}(x) = |\tau - \mathbb{1}(x < 0)|x^2$ with hyper-paramter $\tau \in (0, 1)$. The intuition behind IQL is that with a larger τ, we will be able to better approximate the $\max$ operator. As a result, IQL approximates the Bellman's optimality equation without querying OOD actions.

In practical implementations, we also adopt double Q learning for IQL, where we replace the $Q(\boldsymbol{s},\boldsymbol{a})$ in Eqn. 22 with $\min(Q_1(\boldsymbol{s},\boldsymbol{a}), Q_2(\boldsymbol{s},\boldsymbol{a}))$, and then use the updated value network to train both Q value networks. For IQL, we follow the policy improvement steps of CRR, as described in Eqn. 20 and Eqn. 21. The key difference is that instead of sampling actions to compute $\frac{1}{N}\sum_{i=1}^{N}\min(Q_1(\boldsymbol{s},\hat{\boldsymbol{a}}_i), Q_2(\boldsymbol{s},\hat{\boldsymbol{a}}_i))$, IQL replaces it directly with the learned $V(\boldsymbol{s})$. As for hyper-parameters, we use a temperature of $\beta = 1$, a fixed standard deviation $\boldsymbol{\sigma} = \mathbb{I}$, and for expectile ratio τ, we use $\tau = 0.9$ for antmaze-v0 environments and $\tau = 0.7$ for other tasks.

B Reinforcement Guided Diffusion Policy Details

The overall algorithm for our Reinforcement Guided Diffusion Policy Learning is given in Alg. 1.

The detailed algorithm for energy-based action selection is given in Alg. 2.

C Environmental Details

C.1 Hyper-Parameters

We detail our hyperparameters in Tab. **??**.

```
Algorithm 1: Reinforcement Guided Diffusion Policy
Input: $I$, $\lambda$, $\eta$, $\mathcal{D}$, $\boldsymbol{\epsilon}_\theta(\boldsymbol{a}^k, k; \boldsymbol{s})$, $Q_\phi(\boldsymbol{s}, \boldsymbol{a})$
Output: $\theta$, $\phi$
for $i = 0, \ldots, I$ do
    // Sample a batch of data
    $\{(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')\} \sim \mathcal{D}$
    // Sample next actions with DPM-Solver
    $\boldsymbol{a}' \sim \pi_\theta(\cdot|\boldsymbol{s})$
    $\phi \leftarrow \phi - \eta \nabla_\phi L_{\text{TD}}(\phi)$
    // Action approximation
    $\hat{\boldsymbol{a}}^0 = \frac{1}{\sqrt{\bar{\alpha}^k}} \boldsymbol{a}^k - \frac{\sqrt{1-\bar{\alpha}^k}}{\sqrt{\bar{\alpha}^k}} \boldsymbol{\epsilon}_\theta(\boldsymbol{a}^k, k; \boldsymbol{s})$
    $\theta \leftarrow \theta - \eta \nabla_\theta (L_{\text{diff}}(\theta) + \lambda L_\pi(\theta))$
```

```
Algorithm 2: Energy-based Action Selection
Input: number of actions: $N$, $\pi_\theta(\boldsymbol{a}|\boldsymbol{s})$, $Q_\phi(\boldsymbol{s}, \boldsymbol{a})$
Output: $a$
$\boldsymbol{a}_i \sim \pi_\theta(\boldsymbol{a}|\boldsymbol{s}) \quad i = 1, \ldots, N$
$\boldsymbol{a}$ = categorical_sample($\{\boldsymbol{a}_i\}$, $\{e^{Q_\phi(\boldsymbol{s}, \boldsymbol{a}_i)}\}$)
```

C.2 More Results

We first expand Tab. 1 by providing detailed numbers for each of the tasks used in Tab. 4.

We report the performance of EDP trained with TD3, CRR, and IQL in Tab. 5, where we directly compare the scores of different evaluation metrics, *i.e.*, OMS and RAT. We can observe that there are huge gaps between OMS and RAT for all domains and all algorithms. However, IQL and CRR are relatively more stable than $TD3$. For example, on the antmaze domain, TD3 achieves a best score of 80.5, while the average score is just 29.8. In comparison, the best and average scores of IQL are 89.2 and 73.4, respectively.

C.3 More Results on EAS

We compare normal TD3+BC, TD3+BC with EAS for evaluation and TD3 + EDP in Tab. **??**.

C.4 More Experiments on Controlled Sampling

As in Sec. 4.5, we discussed reducing variance with EAS. We now detail another two methods experimented as below.

Policy scaling Instead of sampling from the policy directly, we can sample from a sharper policy $\pi_\theta^\tau(\boldsymbol{a}|\boldsymbol{s})$, where $\tau > 1$. The scaled policy π_θ^τ shares the same distribution modes as π_θ but with reduced variance. Since diffusion policies are modeling the scores $\log \pi_\theta(\boldsymbol{a}_t|\boldsymbol{s})$, policy scaling can be easily achieved by scaling the output of noise-prediction network by the factor τ. We conduct experiments on the gym-locomotion tasks in D4RL by varying τ from 0.5 to 2.0, as shown in Fig. 6, the results show the best performance is achieved when $\tau = 1.0$. It means sampling from a scaled policy does not work.

Table 3: Hyperparameters used by EDP.

Task	Learning rate	Gradient norm clipping	Loss weight	Epochs	Batch size
Locomotion	0.0003	5	1.0	2000	256
Antmaze	0.0003	5	1.0	1000	256
Adroit	0.00003	5	0.1	1000	256
Kitchen	0.0003	5	0.005	1000	256

Table 4: The performance of Diffusion-QL with efficient diffusion policy. The results for Diffusion-QL are directly quoted from [35]. EDP is our method. DQL (JAX) is a variant that uses the exact same configurations as Diffusion-QL. All results are reported based on the OMS metric.

Dataset	Diffusion-QL	DQL (JAX)	EDP
halfcheetah-medium-v2	51.5	52.3	52.8
hopper-medium-v2	96.6	95.3	98.6
walker2d-medium-v2	87.3	86.9	89.6
halfcheetah-medium-replay-v2	48.3	50.3	50.4
hopper-medium-replay-v2	102.0	101.8	102.7
walker2d-medium-replay-v2	98.0	96.3	97.7
halfcheetah-medium-expert-v2	97.2	97.3	97.1
hopper-medium-expert-v2	112.3	113.1	112.0
walker2d-medium-expert-v2	111.2	111.5	112.0
average	89.4	89.4	90.3
antmaze-umaze-v0	96.0	93.4	93.4
antmaze-umaze-diverse-v0	84.0	74.0	66.0
antmaze-medium-play-v0	79.8	96.0	88.0
antmaze-medium-diverse-v0	82.0	82.0	96.0
antmaze-large-play-v0	49.0	66.0	60.0
antmaze-large-diverse-v0	61.7	60.0	64.0
average	75.4	78.6	77.9
pen-human-v1	75.7	74.0	98.3
pen-cloned-v1	60.8	59.1	79.9
average	68.3	66.5	89.1
kitchen-complete-v0	84.5	100.0	97.0
kitchen-partial-v0	63.7	73.0	71.5
kitchen-mixed-v0	66.6	76.0	73.0
average	71.6	83.0	80.5

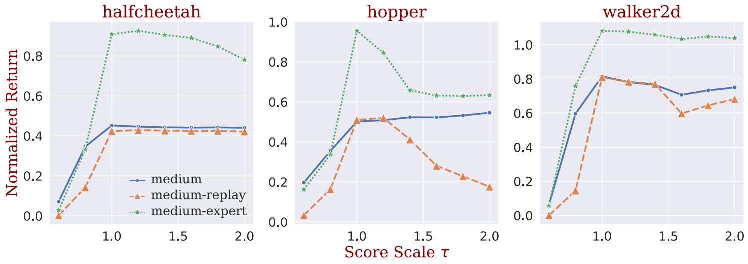

Figure 6: Performance of EDP + TD3 on gym-locomotion tasks with varying τ.

Deterministic Sampling This method is based on the observation that the sampling process of the DPM-Solver is deterministic except for the first step. The first step is to sample from an isotropic Gaussian distribution. We modify it to use the mean, thus avoiding stochasticity. Consider a initial noise $\boldsymbol{a}^K \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, we rescale $\boldsymbol{a}^K$ by a noise scale factor. We show how this factor affects the final performance by varying it from 0.0 to 1.0. As illustrated in Fig. 7, the best performance is achieved at zero noise in most cases, and a normal $\boldsymbol{a}^K$ performs worst. This means reducing the variance of the initial noise is able to improve the performance of a diffusion policy. However, the best performance achieved in this way still falls behind EAS.

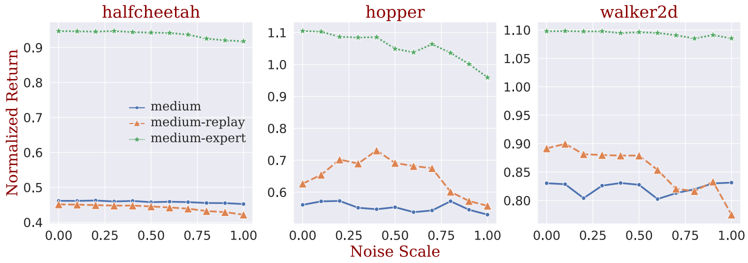

Figure 7: Performance of EDP + TD3 on gym-locomotion tasks with varying initial noise scale.

Table 5: Average normalized score on the D4RL benchmark of EDP trained with different algorithms. “Best” represents the online model selection metric, while “Average” is our runing average at training metric.

	EDP + TD3		EDP + CRR		EDP + IQL	
	Average	Best	Average	Best	Average	Best
halfcheetah-medium-v2	52.1	52.8	49.2	50.2	48.1	48.7
hopper-medium-v2	81.9	98.6	78.7	95.0	63.1	97.3
walker2d-medium-v2	86.9	89.6	82.5	85.8	85.4	88.7
halfcheetah-medium-replay-v2	49.4	50.4	43.5	47.8	43.8	45.5
hopper-medium-replay-v2	101.0	102.7	99.0	101.7	99.1	100.9
walker2d-medium-replay-v2	94.9	97.7	63.3	89.8	84.0	93.4
halfcheetah-medium-expert-v2	95.5	97.1	85.6	93.5	86.7	80.9
hopper-medium-expert-v2	97.4	112.0	92.9	109.4	99.6	95.7
walker2d-medium-expert-v2	110.2	112.0	110.1	112.3	109.0	111.5
average	85.5	90.3	78.3	87.3	79.9	84.7
kitchen-complete-v0	61.5	93.4	73.9	95.8	75.5	95.0
kitchen-partial-v0	52.8	66.0	40.0	56.7	46.3	72.5
kitchen-mixed-v0	60.8	88.0	46.1	59.2	56.5	70.0
average	58.4	96.0	53.3	70.6	59.4	79.2
pen-human-v0	48.2	60.0	70.2	127.8	72.7	130.3
pen-cloned-v0	15.9	64.0	54.0	106.0	70.0	138.2
average	32.1	77.9	62.1	116.9	71.3	134.3
antmaze-umaze-v0	96.6	98.3	95.9	98.0	94.2	98.0
antmaze-umaze-diverse-v0	69.5	79.9	15.9	80.0	79.0	90.0
antmaze-medium-play-v0	0.0	89.1	33.5	82.0	81.8	89.0
antmaze-medium-diverse-v0	6.4	97.0	32.7	72.0	82.3	88.0
antmaze-large-play-v0	1.6	71.5	26.0	57.0	42.3	52.0
antmaze-large-diverse-v0	4.4	73.0	58.5	71.0	60.6	68.0
average	29.8	80.5	43.8	76.7	73.4	89.2

Table 6: Energy-based Action Selection + Normal TD3

Dataset	TD3+Diff	TD3+EAS	TD3
halfcheetah-medium-v2	52.1	48.7	47.8
hopper-medium-v2	81.9	50.8	54.0
walker2d-medium-v2	86.9	77.7	53.4
halfcheetah-medium-replay-v2	49.4	44.4	44.1
hopper-medium-replay-v2	101.0	59.8	59.1
walker2d-medium-replay-v2	94.9	74.8	71.6
halfcheetah-medium-expert-v2	95.5	85.9	92.3
hopper-medium-expert-v2	97.4	71.9	95.1
walker2d-medium-expert-v2	110.2	108.4	110.0
gym-locomotion-v2 (avg)	85.5	69.2	69.7

C.5 Computational Cost Comparison between EDP and Feed-Forward Policy networks

We benchmark the training speed of TD3+BC with EDP on walker2d-medium-expert-v2, by training each agent for 10,000 iterations of policy updates. The training speed for TD3+BC is 689 iterations-per-second (IPS), while EDP is 412 IPS. In other words, it takes around 3 hours to train an agent with the feed-forward network, while EDP needs around 5 hours. We also conducted experiments by double the network used in TD3+BC; unfortunately, there was no performance gain on the locomotion tasks (75.3 to 75.6). Moreover, both feed-forward policy and diffusion policy utilize a 3-layer MLP (hidden size 256) as the backbone network. Therefore, the network capacity should be comparable.

D Negative Societal Impacts

In this paper, we propose an efficient yet powerful policy class for offline reinforcement learning. We show that this method is superior to most existing methods on simulated robotic tasks. However, some war robots or weapon robots might employ our EDP to learn strategic agents considering the generalization ability of EDP. Depending on the specific application scenarios, it might be harmful to domestic privacy and safety.