EXPRESSIVE VALUE LEARNING FOR SCALABLE OFFLINE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) is a powerful paradigm for learning to make sequences of decisions. However, RL has yet to be fully leveraged in robotics, principally due to its lack of scalability. *Offline* RL offers a promising avenue by training agents on large, diverse datasets, avoiding the costly real-world interactions of *online* RL. Scaling offline RL to increasingly complex datasets requires expressive generative models such as diffusion and flow matching. However, existing methods typically depend on either backpropagation through time (BPTT), which is computationally prohibitive, or policy distillation, which introduces compounding errors and limits scalability to larger base policies. In this paper, we consider the question of how to develop a scalable offline RL approach without relying on distillation or backpropagation through time. We introduce *Expressive Value Learning for Offline Reinforcement Learning* (EVOR): a scalable offline RL approach that integrates *both* expressive policies *and* expressive value functions. EVOR learns an optimal, regularized Q-function via flow matching during training. At inference-time, EVOR performs inference-time policy extraction via rejection sampling against the expressive value function, enabling efficient optimization, regularization, and compute-scalable search *without retraining*. Empirically, we show that EVOR outperforms baselines on a diverse set of offline RL tasks, demonstrating the benefit of integrating expressive value learning into offline RL.

1 INTRODUCTION

Reinforcement learning (RL) is a powerful paradigm for learning to make sequences of decisions, having been widely applied to applications such as the fine-tuning of pretrained large language models (LLMs). However, the success of RL in the language domain has yet to be matched in robotics. In contrast to the language setting, robot interactions occur in the real world, which can be costly, time-consuming, and may pose safety concerns. These constraints naturally motivate the *offline* RL setting, where agents attempt to learn from a diverse, often sub-optimal dataset *without* further interaction with the environment.

In considering how to make offline RL more scalable, there are three primary axes: (1) *scaling data*, (2) *scaling models*, and (3) *scaling compute*. In order to scale data, offline RL algorithms must be capable of learning from larger, more diverse datasets that are often sub-optimal and often multi-modal (e.g. the datasets may be generated by multiple data-generating policies of varying quality). Naturally, the need to model complex data distributions necessitates the use of more powerful models. One promising avenue for scaling offline RL is leveraging powerful, expressive generative models like diffusion (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021) and flow matching (Lipman et al., 2024; Esser et al., 2024).

Existing approaches to offline RL with generative models predominantly use the generative model as the policy, improving the policy's ability to model complex distributions over the standard, Gaussian-based policies used in continuous action spaces (Hansen-Estruch et al., 2023; Chen et al., 2023; Ding & Jin, 2023; Wang et al., 2022; Espinosa-Dice et al., 2025; Park et al., 2025b; Zhang et al., 2025). At a high-level, diffusion and flow-based RL policies sample actions via an iterative noise sampling procedure, which requires backpropagating through time in the iterative noise sampling procedure. Backpropagation through time is computationally expensive, memory-intensive, and can degrade the general knowledge of the underlying base policy (e.g. a vison-lanuage model (VLM) in the vision-language-action (VLA) setting) (Ding & Jin, 2023; Zhou et al., 2025b;c).

As an alternative to backpropagation through time, distillation-based methods compress the multi-step policy (e.g. a standard diffusion or flow model) into a one-step model, which can be more efficiently optimized through standard policy gradient techniques (Ding & Jin, 2023; Chen et al., 2023; Park et al., 2025b). However, distillation-based methods have a fundamental limitation: while expressive models can be used for the base policy (e.g. to model the offline data distribution), the policy that is actually being optimized and rolled out is a less expressive, one-step model. While a one-step model may be sufficient for easier simulation-based tasks, they are difficult to scale to larger base policies (e.g. VLAs) or more complex and real-world tasks, partially due to the compounding errors between the teacher network (i.e. the base policy) and the student network (i.e. the distilled policy).

Ultimately, we seek an offline RL approach that is scalable (e.g. to large base policies), and we tackle this question in our paper:

Can we develop a scalable offline RL approach
without relying on policy distillation or backpropagation through time?

A natural alternative to policy gradients is rejection sampling: sample multiple action candidates from the base policy and choose the one with the highest value according to a learned value function (e.g. Q-function). However, existing rejection sampling methods suffer from two key limitations: (1) the learned value function is not regularized, and (2) the learned value function is limited to Gaussian-based models. Standard approaches learn the value function from the offline dataset, resulting in $Q^{\pi_{\text{ref}}}$, the Q-function under the data-generating policy π_{ref}, which is not the optimal solution to the standard KL-regularized offline RL objective. Additionally, the Q-functions used in continuous state-action spaces are standard, Gaussian-based value functions. Like Gaussian-based policies, these models are less expressive in modeling complex distributions than diffusion and flow-based methods.

Finally, we consider the third axis of scale—compute—and, in particular, how to take advantage of additional inference-time compute. Existing approaches to inference-time scaling generally leverage dynamics or world models for additional planning at inference-time, such as model predictive path integral control (Williams et al., 2017), model-based offline planning (Hafner et al., 2019; Argenson & Dulac-Arnold, 2020), planning with world models (Hafner et al., 2023), and Monte Carlo tree search (Chen et al., 2024). While effective, these methods either do not leverage expressive models, instead relying on Gaussian-based approaches, or they require learning and maintaining an auxiliary model of the environment, which can introduce additional sources of approximation error and scaling challenges.

These limitations point to a key gap in the scalability of existing offline RL approaches: although expressive generative models have been integrated into policies, the same level of expressivity has yet to be brought to value functions, which remain restricted to Gaussian-based models. In this paper, we bridge this gap through *Expressive Value Learning for Offline Reinforcement Learning* (EVOR): an approach for learning an optimal solution to the KL-regularized offline RL objective with *both* expressive policies *and* expressive value functions. EVOR achieves the following desiderata for scalable offline RL:

1. **EVOR avoids policy distillation and backpropagation through time during policy optimization.** EVOR does not learn require learning a new policy and instead optimizes the base policy through *inference-time policy extraction*. Unlike standard rejection sampling approaches, EVOR uses an optimal, regularized Q-function.

2. **EVOR learns an expressive, optimal Q-function via flow matching.** In contrast to standard value learning methods that employ Gaussian-based models, EVOR uses expressive flow models for value learning. Moreover, the Q-function learned is an optimal, regularized solution to the regularized offline RL objective.

3. **EVOR enables inference-time scaling and regularization.** EVOR provides a natural mechanism for inference-time scaling: performing additional search, guided by the expressive value function, *without retraining*.

2 Background

Markov Decision Process. We consider a finite-horizon Markov decision process (MDP) $(\mathcal{X}, \mathcal{A}, P, r, H)$, where $\mathcal{X}$ is the state space, $\mathcal{A}$ is the action space, P is the transition function, $r : \mathcal{X} \times \mathcal{A} \to [0, 1]$ is the reward function, and H is the MDP's horizon (Puterman, 2014). An offline dataset $\mathcal{D} = \{(x_h, a_h, r_h, x_{h+1})\}$ is collected under some unknown reference policy π_{ref}, which could be multi-modal and sub-optimal. In the offline RL setting, we do not assume access to environment interactions.

Offline Reinforcement Learning. The offline RL objective is generally expressed as combination of a policy optimization term and a regularization term, such that

$$\underset{\pi\in\Pi}{\operatorname{argmax}} \quad \underbrace{J_{\mathcal{D}}(\pi)}_{\texttt{Policy Optimization}} \quad - \underbrace{\eta \text{Reg}(\pi, \pi_{\text{ref}})}_{\texttt{Regularization}} \tag{1}$$

where $J_{\mathcal{D}}(\pi)$ is the expected return over offline dataset $\mathcal{D}$, π_{ref} is the unknown data generating policy, and $\text{Reg}(\pi, \pi_{\text{ref}})$ is a regularization term (Espinosa-Dice et al., 2025). The regularization term generally takes the form of a divergence measure between π and π_{ref}, with KL divergence commonly used. The offline RL objective can be expressed as the soft value of a policy subject to KL regularization:

$$V^{\pi,\eta} = \mathbb{E}_{\pi}\left[\sum_{h=1}^{H} r(x_h, a_h) - \eta \text{KL}\left(\pi(x_h) \| \pi_{\text{ref}}(x_h)\right)\right], \tag{2}$$

where the expectation is over a random trajectory $(x_1, a_1, \ldots, x_H, y_H)$ sampled according to π and the KL divergence is $\text{KL}(p\|q) = \mathbb{E}_{z\sim p}\left[\log\left(p(z)/q(z)\right)\right]$ (Zhou et al., 2025a). The objective is to learn the optimal, regularized policy $\pi^\star = \operatorname{argmax}_{\pi\in\Pi} V^{\pi,\eta}$. Ziebart et al. (2008) showed that

$$V_{H+1}^{\star,\eta}(x) = 0, \tag{3}$$

$$Q_h^{\star,\eta}(x,a) = r(x,a) + \mathbb{E}_{x'\sim P_h(x,a)}\left[V_{h+1}^{\star}(x')\right], \tag{4}$$

$$\pi^{\star,\eta}(a|x) \propto \pi_{\text{ref}}(a|x)\exp\left(\eta^{-1} Q_h^{\star,\eta}(x,a)\right), \tag{5}$$

$$V_h^{\star,\eta}(x) = \eta \ln \mathbb{E}_{a\sim\pi_{\text{ref}}}(x)\left[\exp\left(\eta^{-1} Q_h^{\star,\eta}(x,a)\right)\right]. \tag{6}$$

For convenience, we drop the η superscript when clear from context.

Reward-To-Go. We define the reward-to-go under the unknown data-generating policy π_{ref}, starting at state x and taking action a, as

$$Z(x,a) := \sum_{h=0}^{H} r(x_h, a_h), \quad x_0 = x,\ a_0 = a,\ x_{h+1} \sim P_h(\cdot \mid x_h, a_h),\ a_{h+1} \sim \pi_{\text{ref}}(\cdot \mid x_{h+1}), \tag{7}$$

We define $R(\cdot \mid x, a)$ as the law of the random variable $Z(x,a)$, so $R(\cdot \mid x,a) \overset{D}{=} Z(x,a)$. In other words, $R(\cdot \mid x, a)$ is the distribution of rewards-to-go under π_{ref}, starting at state x and taking action sequence a. We can thus define

$$\pi^{Z,\eta}(a|x) \propto \pi_{\text{ref}}(a|x)\mathbb{E}_{z\sim Z(x,a)}\left[\exp\left(z/\eta\right)\right]. \tag{8}$$

We can also define $R^{\pi}(\cdot \mid x, a)$ as the distribution of rewards-to-go under a policy π.

Flow Matching. We define flow matching (Lipman et al., 2022; Liu et al., 2022; Lipman et al., 2024) as follows. Let $p(x) \in \Delta(\mathbb{R}^d)$ be a data distribution. Given a vector field v_t, we construct its corresponding flow, $\phi : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$, by the ordinary differential equation (ODE)

$$\frac{d}{dt}\phi_t(x) = v_t(\phi_t(x)) \tag{9}$$

$$\phi_0(x) = x \tag{10}$$

We employ Lipman et al. (2024)'s flow matching, which is based on linear paths and uniform time sampling, such that the objective is

$$\min_{\theta} \mathbb{E}_{\substack{x^0\sim\mathcal{N}(0,I_d)\\ x^1\sim p(x)\\ t\sim U[0,1]}}\left[\|v_\theta(t, x^t) - (x^1 - x^0)\|_2^2\right] \tag{11}$$

where $x^t = (1-t)x^0 + tx^1$ is the linear interpolation between x^0 and x^1.

3 Expressive Value Learning for Offline Reinforcement Learning

In this section, we present *Expressive Value Learning for Offline Reinforcement Learning* (EVOR).

Algorithm 1: EVOR Training via Flow-Based TD Learning

Data: Offline dataset $\mathcal{D}$;
while not converged **do**
  Sample $(x, a^1, x', r) \sim \mathcal{D}$ `# Parallelize batch`
  ▷ **Base Policy Update via Flow Matching**
  $a^0 \sim \mathcal{N}(0, I_d),\ t \sim \text{Unif}(0,1)$ `# Sample noise and time`
  $a^t \leftarrow (1-t)a^0 + ta^1$ `# Noise action`
  $\phi \leftarrow \nabla_\phi \|v_\phi(a^t, t \mid x) - (a^1 - a^0)\|^2$ `# Update actor`
  ▷ **Reward Model Update via Flow-Based TD Learning**
  $z^0 \sim \mathcal{N}(0, I_d),\ z^1 \sim R(\cdot \mid x, a),\ t \sim \text{Unif}(0,1)$ `# Sample noise, reward-to-go, time`
  $z^t \leftarrow (1-t)z^0 + tz^1$ `# Noise reward-to-go`
  $a' \sim \pi_{\text{base}}(\cdot \mid x')$ `# Sample action from base policy`
  $\text{target}(x, a, z^t, t) \leftarrow r(x, a) + \bar{s}_\theta(z^t, t \mid x', a')$ `# Flow-matching target`
  $\theta \leftarrow \nabla_\theta \|s_\theta(z^t, t \mid x, a) - \text{target}(x, a, z^t, t)\|^2$ `# Update critic`

3.1 Optimal, Expressive Value Learning via Regression

First, we tackle the question:

How do we learn an optimal, expressive value function for the KL-regularized offline RL objective?

In offline reinforcement learning, we aim to learn or fine-tune a policy from a dataset collected under some unknown data-generating policy π_{ref}. Depending on the setting, we either have access to a base policy π_{base} (e.g. a pre-trained generalist model) or we must learn the policy from scratch. Both settings are compatible with our approach, and we first show how a base policy can be learned in the setting where a starting base policy is not given.

Base Policy Learning. In the setting where a starting base policy is not known, we train a policy π_{base} that predicts actions via behavioral cloning (Pomerleau, 1988) on the offline dataset's state-action pairs. By formulating the objective as supervised learning, rather than a more complicated RL procedure, we can employ any generative model to learn the base policy, and we choose flow matching (Lipman et al., 2022; Liu et al., 2022) here. By leveraging an expressive model like flow matching, we can model multi-modal offline data.

We present the flow matching objective below, where a^0 represents a fully noised action (i.e. noise sampled from a Gaussian) and a^1 represents a real action (i.e. action sampled from the offline data $\mathcal{D}$). Through Equation 12 below, we will learn a base policy $\pi_{\text{base}} \approx \pi_{\text{ref}}$, subject to finite sample and optimization errors. The flow matching loss is given by:

$$\mathcal{L}_{\text{BC}}(\phi) = \mathbb{E}_{\substack{(x,a^1)\sim\mathcal{D},\, a^0\sim\mathcal{N} \\ t\sim\text{Unif}(0,1)}}\left[\Big\| \underbrace{v_\phi(a^t, t \mid x)}_{\texttt{Velocity Prediction}} - \underbrace{(a^1 - a^0)}_{\texttt{Velocity Target}} \Big\|^2\right] \tag{12}$$

Value Learning. Next, we consider how to learn an expressive value function. Our key insight is, rather than use standard methods for value function learning, we instead train a reward model on the distribution of rewards-to-go of π_{ref}, a distribution we have samples from in the offline dataset. Intuitively, we can think of flow matching as a method of transporting samples from a starting distribution (e.g. samples from Gaussian noise) to a target distribution (i.e. the data distribution). In this case, we simply set the target distribution to π_{ref}'s distribution of rewards-to-go, $R(\cdot \mid x, a)$. We can then cast the problem as flow matching:

$$\mathcal{L}_{\text{RM}}(\theta) = \mathbb{E}_{\substack{(x,a)\sim\mathcal{D},\, r^1\sim R(\cdot|x,a), \\ r^0\sim\mathcal{N},\, t\sim\text{Unif}(0,1)}}\left[\Big\| \underbrace{v_\theta(r^t, t \mid x, a)}_{\texttt{Velocity Prediction}} - \underbrace{(r^1 - r^0)}_{\texttt{Velocity Target}} \Big\|^2\right] \tag{13}$$

where r^1 denotes a sample from the data distribution (i.e. the dataset's rewards-to-go), r^0 denotes a sample from the base noise distribution, and r^t is a linear interpolation between the two.

Algorithm 2: EVOR Inference via $Q_\theta^\star$ Reweighting

Input: State x; number of action candidates N_π; number of reward-to-go samples N; temperatures τ_R, τ_Q

Output: Action a

$\{a^{(i)}\}_{i=1}^{N_\pi} \sim \pi_{\text{base}}(\cdot \mid x)$ `Sample` N_π `candidate actions`

$\{r^{(i,j)}\}_{j=1}^{N} \sim R_\theta(\cdot \mid x, a^{(i)})$ `Sample` N `reward-to-go samples`

$Q_\theta^\star(x, a^{(i)}) \leftarrow \tau_R \operatorname{LogSumExp}_{r \in \{r^{(i,j)}\}_{j=1}^{N}} (r/\tau_R)$ `Sample average` $Q_\theta^\star$

$a^\star \sim \operatorname{softmax}_{a \in \{a^{(i)}\}_{i=1}^{N_\pi}} (Q_\theta^\star(x,a)/\tau_Q)$ `Softmax over action candidates`

return $a^\star$

Next, we consider how to learn a value function using the reward model. Leveraging results from Ziebart et al. (2008) and Zhou et al. (2025a), we show that the optimal, regularized Q-function can be learned using the learned reward model.

Theorem 1 (Optimal Regularized Value Functions (Zhou et al., 2025a)). *Under deterministic transitions, the optimal value and Q-functions are given by*

$$V_h^{\star,\pi}(x_h) = \eta \ln \mathbb{E}_{\pi_{\text{ref}}} \left[\exp\left(\eta^{-1} \sum_{t \geq h}^{H} r(x_t, a_t) \right) \middle| x_h \right], \tag{14}$$

$$Q_h^{\star,\pi}(x_h, a_h) = \eta \ln \mathbb{E}_{\pi_{\text{ref}}} \left[\exp\left(\eta^{-1} \sum_{t \geq h}^{H} r(x_t, a_t) \right) \middle| x_h, a_h \right]. \tag{15}$$

Using Theorem 1, we can express the optimal, regularized Q-function as a function of π_{ref}'s reward-to-go distribution, such that

$$Q_h^\star(x_h, a_h) = \eta \ln \mathbb{E}_{z \sim R_h(\cdot \mid x_h, a_h)} \exp(\eta^{-1} z) \tag{16}$$

In practice, we can approximate the expectation via sample averaging. The assumption of deterministic dynamics is strong, and often does not hold in real-world robotics, but it is frequently imposed in offline RL algorithms (Edwards et al., 2020; Ma et al., 2022; Schweighofer et al., 2022; Park et al., 2023; Ghosh et al., 2023; Wang et al., 2023; Karabag & Topcu, 2023; Park et al., 2024a;c). However, in the next section, we tackle how to learn an expressive value function under stochastic dynamics.

3.2 Scalable Value Learning via Flow-Based TD Learning

Next, we tackle the question:

How do we learn an expressive value function under stochastic dynamics?

We present *flow-based temporal difference (TD) learning*, a flow matching-based approach to TD learning. Using TD learning will enable us to handle the non-deterministic dynamics setting, while still leveraging the expressive modeling power of flow matching. In this section, we present the flow-based TD objective and high-level intuition behind it, and we more formally explain its derivation in Appendix B. The full training procedure is shown in Algorithm 1.

Distributional Bellman. TD learning uses the Bellman equation to learn a value function by constructing a bootstrap target (i.e. the right-hand side (RHS) of the Bellman equation) (Bellman, 1966; Sutton & Barto, 1998), such that

$$Q(x,a) = r(x,a) + \mathbb{E}_{P,\pi} Q(X', A'). \tag{17}$$

The Bellman equation also holds under distributions (Jaquette, 1973; Sobel, 1982; White, 1988; Bellemare et al., 2017), such that

$$\underbrace{Z(x,a)}_{\texttt{LHS of Distributional Bellman}} \stackrel{D}{=} \overbrace{r(x,a) + Z(X',A')}^{\texttt{RHS of Distributional Bellman}} \tag{18}$$

where $Z(X',A')$ denotes the random return.

Flow-Based TD Objective. At a high-level, flow matching learns how to transport a known prior distribution into a target data distribution. To construct a flow-based TD objective, we set the RHS of the distributional Bellman equation as the target distribution, and match the velocities between the LHS and RHS distributions.

We will learn a conditional flow model $s_\theta(\cdot \mid x,a,t)$ that transports base noise $z^0_{x,a} \sim \mathcal{N}(0, I_d)$ to a terminal variable $z^1_{x,a} \sim R_\theta(\cdot \mid x,a)$, such that the distribution $R_\theta(\cdot \mid x,a) \approx R(\cdot \mid x,a)$. The bootstrap target is given by

$$\text{target}(x,a,z^t,t) := r(x,a) + \mathbb{E}_{a' \sim \pi_{\text{base}}(\cdot|x')} s_{\bar{\theta}}(z^t \mid x', a', t), \tag{19}$$

and the loss is given by

$$\mathcal{L}_{\text{FlowTD}}(\theta) = \underbrace{\mathbb{E}_{(x,a,r,x')\sim\mathcal{D}}}_{\texttt{Dataset's State-Action-Reward}} \overbrace{\mathbb{E}_{z^1 \sim R_{\bar{\theta}}(\cdot|x,a)}}^{\texttt{Sample Reward-To-Go}} \underbrace{\mathbb{E}_{t\sim \text{Unif}(0,1)}}_{\texttt{Sample Time}} \Big\| \overbrace{s_\theta(z^t \mid x,a,t)}^{\texttt{Velocity Prediction of LHS}} - \underbrace{\text{target}(x,a,z^t,t)}_{\texttt{Velocity Target of RHS}} \Big\|_2^2. \tag{20}$$

We sample a state-action-reward-next-state tuple $(x,a,r,x') \sim \mathcal{D}$ from the offline data, a time $t \sim \text{Unif}(0,1)$, and the next action from the base policy $a' \sim \pi_{\text{base}}(\cdot \mid x')$. We construct an interpolant $z^t = (1-t)z^0 + tz^1$, which serves to noise the ground-truth sample, by sampling a reward-to-go $z^1 \sim R(\cdot \mid x,a)$ and a noise sample $z^0 \sim \mathcal{N}(0, I_d)$. The reward-to-go sample z^1 can be sampled from the dataset or a target version of the learned reward model $R_{\bar{\theta}}(\cdot \mid x,a)$. To sample a reward-to-go from the distribution $R_\theta(\cdot \mid x,a)$, we employ the standard forward Euler method with the learned flow model $s_\theta(\cdot \mid x,a,t)$.

3.3 Inference-Time Policy Extraction, Regularization, and Scaling

EVOR's training procedure focuses on learning an expressive value function, and it trains the base policy via flow matching on the offline dataset, leading to the natural question:

How does EVOR optimize the base policy beyond the offline dataset without distillation or backpropagation through time?

Inference-Time Policy Extraction. Instead of learning a new policy during training, EVOR performs *inference-time policy extraction* using the learned distributional reward model. A common approach to inference-time policy extraction is to perform rejection sampling with the learned Q-function as a "verifier": given a state x, sample actions independently from the base policy $a_1, a_2, \ldots, a_N \sim \pi_{\text{base}}(\cdot \mid x)$, and select the action with the largest Q value, such that

$$\operatorname*{argmax}_{a \in \{a_1, a_2, \ldots, a_N\}} Q(x,a) \tag{21}$$

However, using the Q-function trained on the offline dataset $\mathcal{D}$ will result in an unregularized Q-function, specifically $Q^{\pi_{\text{base}}}$. In the offline RL setting, π_{base} is often sub-optimal, so optimizing $Q^{\pi_{\text{base}}}$ may lead to distribution shift at test-time and poor performance (Zhou et al., 2025a). Instead, we utilize our expression for the *optimal* Q-function from Subsection 3.1,

$$Q^\star(x,a) = \eta \ln \mathbb{E}_{r \sim R(\cdot|x,a)} \exp(r/\eta), \tag{22}$$

where R is the conditional distribution of rewards-to-go under π_{ref}. In practice, we approximate the expectation via sample averaging, and we can construct a softmax over the $Q^\star$ values, as shown in Algorithm 2.

Inference-Time Regularization and Scaling. EVOR's formulation provides a natural mechanism for inference-time regularization and scaling. Since actions are sampled from the base policy, running EVOR with varying temperatures τ_R and τ_Q controls the strength of regularization and policy optimization. Increasing N_π corresponds to performing additional test-time search, while decreasing N_π will allow for faster inference under lower compute budgets. Crucially, these parameters can be varied at test-time *without retraining*, allowing for both inference-time scaling and regularization.

Table 1: **`EVOR`'s Overall Performance.** `EVOR` outperforms the baselines on all 5 environments, for a total of 25 unique tasks in the OGBench task suite (Park et al., 2024a). Results are averaged over 3 seeds per task, with standard deviations reported. The full results are reported in Appendix C.

Task Category	QC-1	QC-5	EVOR
OGBench antmaze-large-navigate-singletask (5 tasks)	11 ±4	7 ±2	**49** ±4
OGBench antmaze-large-stitch-singletask (5 tasks)	5 ±3	4 ±4	**18** ±1
OGBench cube-double-play-100M-singletask (5 tasks)	54 ±6	50 ±12	**82** ±2
OGBench pointmaze-medium-naviate-singletask (5 tasks)	**96** ±0	**99** ±1	**99** ±0
OGBench scene-play-singletask (5 tasks)	45 ±2	**85** ±4	**86** ±7

4 Experimental Results

In this section, we investigate the performance of `EVOR`, and in particular, we focus on the following question:

What is the benefit of expressive value learning?

4.1 Experimental Setup

Environments and Tasks. We follow the experimental setup of prior works that leverage the OGBench task suite (Park et al., 2024a; 2025b; Espinosa-Dice et al., 2025; Li et al., 2025), specifically evaluating `EVOR` on locomotion and manipulation robotics tasks. We describe the full implementation details in Appendix E.

Baselines. Rather than compare to all of the existing offline RL algorithms benchmarked on OGBench, we instead aim to isolate the effect of expressive value learning in order to demonstrate its benefit specifically. Thus, we compare to Q-chunking (`QC`, Li et al. (2025)), a recent offline RL algorithm that is closest to `EVOR`. Like `EVOR`, `QC` learns a base policy via flow matching and extracts an optimized policy via rejection sampling. The key difference between `QC` and `EVOR` is in how the value function is learned, which is the exact difference we aim to isolate. `QC` can employ action chunking in both its policy and value function, and we compare `EVOR` to both `QC` with (`QC-5`) action chunking and without it (`QC-1`). We select the action chunk length (5) based on Li et al. (2025)'s recommendation.

Evaluation. To construct a fair comparison, we use the same network size, number of gradients, and discount factor for all algorithms, similar to Park et al. (2025b); Espinosa-Dice et al. (2025). Moreover, we use the official `QC` implementation and its parameters. We bold values at 95% of the best performance in tables.

4.2 Experimental Results

Q: What is `EVOR`'s overall performance?

Across 5 environments and 25 unique tasks, `EVOR` achieves the best performance compared to the baselines.

We present the environment aggregation results in Table 1, and we present the full results in Appendix C

Q: Does using expressive models for value learning improve performance?

Yes, `EVOR`'s expressive value learning method outperforms standard value learning methods.

From the results in Table 1, we observe that `EVOR` outperforms or matches standard value function learning methods (`QC`), even compared to a method that employs action chunking (`QC-5`), suggesting that expressive value learning can improve performance over standard value function learning.

Q: How can `EVOR` take advantage of greater inference-time compute?

As shown in Figure 1, when given access to greater inference-time compute, `EVOR` can increase the number of action candidates N_π, resulting in better performance (up to a saturation point).

We present the full results for inference-time scaling in Appendix D.

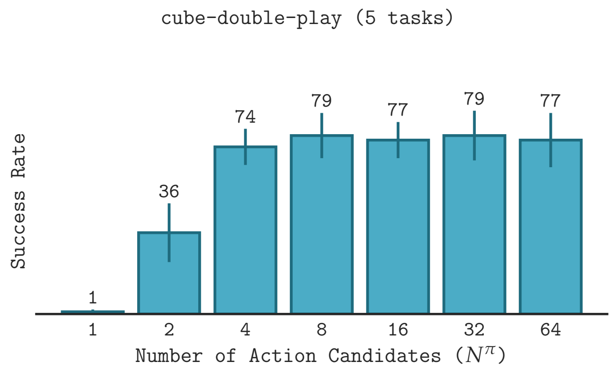

Figure 1: **EVOR's Inference-Time Scaling.** EVOR can perform inference-time scaling by increasing the number of action candidates N_π, performing greater search at inference time with the expressive value function. Leveraging greater inference-time compute results in better performance, up to a saturation point. Results are averaged over 3 seeds per task, with standard deviations reported.

Q: How can EVOR perform inference-time regularization?

As shown in Figure 2, by increasing varying the temperature parameters τ_R and τ_Q, EVOR can vary the level of regularization to the base policy compared to the level of policy optimization.

As τ_Q decreases, the action selection becomes more greedy, while as τ_Q increases, the action selection becomes more regularized to the base policy π_{base} (i.e. the performance of EVOR with $N_\pi = 1$).

Q: What training parameters must EVOR tune per environment?

EVOR uses the same training and evaluation parameters for all environments.

A key benefit of EVOR is that it reduces the number of training parameters that must be tuned per environment. In particular, EVOR uses the same training parameters for all environments in Table 1, despite the environments spanning distinct locomotion and manipulation tasks. In contrast, policy gradient-based offline RL algorithms generally tune parameters per environment (Park et al., 2025b; Espinosa-Dice et al., 2025). We present an ablation study of evaluation parameters in Appendix D.

Q: Does rejection sampling-based policy extraction outperform reparameterized policy gradients?

We do not consider that claim in this paper.

The purpose of this paper is to investigate scalable methods for expressive value learning in offline RL. In our empirical results, we aim to isolate the effect of expressive value learning over standard value function learning by utilizing the same policy extraction method (rejection sampling).

5 Related Work

We present an extended related work in Appendix A.

Offline Reinforcement Learning with Generative Models. Standard offline RL approaches rely on Gaussian-based models in continuous state-action spaces. However, recent work has focused on representing policies via powerful sequence or generative models Chen et al. (2021); Janner et al. (2021; 2022); Wang et al. (2022); Ren et al. (2024a); Wu et al. (2024); Black et al. (2024); Park et al. (2025b); Espinosa-Dice et al. (2025), taking advantage of more powerful generative models like diffusion (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021) and flow matching (Lipman et al., 2022; Liu et al., 2022; Lipman et al., 2024). These generative models are known to be more expressive than Gaussian-based models, enabling them to capture more complex, multi-modal distributions. Modeling complex distributions is particularly relevant to the offline RL setting, where the offline dataset may be composed of multiple data-generating policies of varying qualities. However, diffusion and flow models rely on an iterative sampling

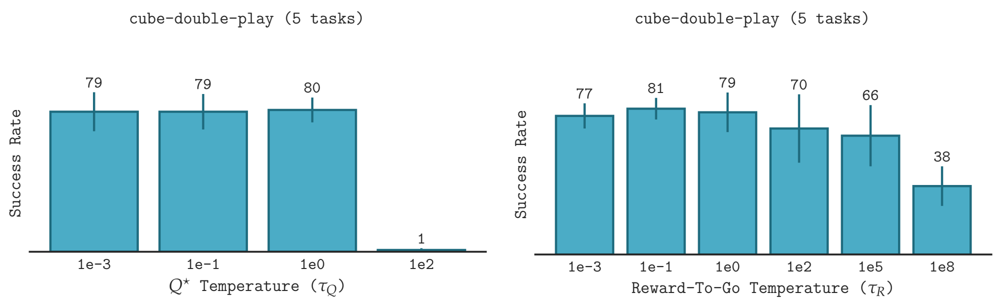

Figure 2: **Ablation Over EVOR's Evaluation Parameters.** EVOR uses the same training parameters for all environments in this paper. However, we investigate the effect of varying the temperature parameters τ_R and τ_Q at inference-time on the performance of EVOR. As τ_Q decreases, the action selection becomes more greedy, while as τ_Q increases, the action selection becomes more regularized. Set to a high value, EVOR becomes equivalent to the base policy (i.e. the performance with $N_\pi = 1$). Results are averaged over 3 seeds per task, with standard deviations reported.

process that can be computationally expensive (Ding & Jin, 2023). To address this problem, some methods utilize a two-stage procedure to first train an expressive generative model on the offline dataset, and then distill it into a one-step model that is then used for policy optimization (Ding & Jin, 2023; Chen et al., 2023; Meng et al., 2023; Park et al., 2025b). Espinosa-Dice et al. (2025) propose an approach for avoiding both distillation and extensive backpropagation through time by leveraging shortcut models for flexible inference, but rely on a standard, Gaussian-based value function. Additionally, generative models have been used for plan generation in offline RL (Zheng et al., 2023) and energy-guided flow and diffusion models, incorporating reward feedback in the flow and diffusion training (Zhang et al., 2025). Farebrother et al. (2025) propose integrating flow matching with Bellman-style updates for successor representation learning.

Inference-Time Scaling in Offline Reinforcement Learning. Inference-time scaling in reinforcement learning often takes the form of leveraging dynamics or world models for additional planning at inference-time. Approaches include model predictive control (Richalet et al., 1978; Hansen et al., 2022), model predictive path integral control (Williams et al., 2015; 2017; Gandhi et al., 2021), model-based offline planning (Hafner et al., 2019; Argenson & Dulac-Arnold, 2020), sequence modeling (Janner et al., 2021; 2022; Kong et al., 2024), planning with world models (Hafner et al., 2023), and Monte Carlo tree search (Chen et al., 2024). Additional approaches include applying rejection sampling to the learned value function at inference-time (Chen et al., 2022; Fujimoto et al., 2019; Ghasemipour et al., 2021; Hansen-Estruch et al., 2023; Park et al., 2024b) or using the gradient of the learned value function to adjust actions at inference-time (Park et al., 2024b). Generative models like flow matching and diffusion models naturally support a form of sequential scaling by increasing the number of steps in the iterative sampling process (Ho et al., 2020; Song et al., 2020; Liu et al., 2022; Lipman et al., 2022). Espinosa-Dice et al. (2025) takes advantage of flexibility in the number of denoising steps used when sampling actions from the policy. However, existing approaches do not leverage generative models for value learning like EVOR. By leveraging more expressive models for value learning, EVOR can better take advantage of larger, more complex offline datasets.

6 Discussion

In summary, EVOR is an approach to scalable offline reinforcement learning that integrates *both* expressive policies *and* expressive value learning. EVOR learns an optimal solution to the KL-regularized offline RL objective, which is used for inference-time policy extraction without model distillation or backpropagation through time, making EVOR scalable (e.g. to larger base policies). Furthermore, EVOR can perform inference-time scaling by performing greater search, using the expressive value function for guidance. Additionally, EVOR can adjust the level of regularization to the base policy without retraining. Future work may investigate how EVOR can be combined with policy gradient-based policy extraction schemes. In this paper, we aim to avoid distillation and backpropagation through time, leading us to rejection sampling against an expressive value function. However, as noted by Park et al. (2024b), reparameterized policy gradients are an effective policy extraction technique. It is possible that EVOR's expressive value learning can further improve policy gradient-based techniques.

Reproducibility Statement

We have taken several steps to ensure the reproducibility of our work. All code necessary to reproduce our experiments, along with instructions for installation and execution, is included in the supplementary materials as an anonymized repository. Detailed descriptions of the experimental setup and parameters are provided in Appendix E. The environments and datasets used in our experiments are publicly available. Together, these resources enable independent verification of our findings. We employ LLMs to aid and polish writing based on drafts that we wrote.

References

Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in neural information processing systems*, 34:7436–7447, 2021.

Arthur Argenson and Gabriel Dulac-Arnold. Model-based offline planning. *arXiv preprint arXiv:2008.05556*, 2020.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pp. 1577–1594. PMLR, 2023.

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

Richard Bellman. Dynamic programming. *science*, 153(3731):34–37, 1966.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. *arXiv preprint arXiv:2209.14548*, 2022.

Huayu Chen, Cheng Lu, Zhengyi Wang, Hang Su, and Jun Zhu. Score regularized policy optimization through diffusion behavior. *arXiv preprint arXiv:2310.07297*, 2023.

Jiayu Chen, Wentse Chen, and Jeff Schneider. Bayes adaptive monte carlo tree search for offline model-based reinforcement learning. *arXiv preprint arXiv:2410.11234*, 2024.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Zihan Ding and Chi Jin. Consistency models as a rich and efficient policy class for reinforcement learning. *arXiv preprint arXiv:2309.16984*, 2023.

Ashley Edwards, Himanshu Sahni, Rosanne Liu, Jane Hung, Ankit Jain, Rui Wang, Adrien Ecoffet, Thomas Miconi, Charles Isbell, and Jason Yosinski. Estimating q (s, s') with deep deterministic dynamics gradients. In *International Conference on Machine Learning*, pp. 2825–2835. PMLR, 2020.

Nicolas Espinosa-Dice, Yiyi Zhang, Yiding Chen, Bradley Guo, Owen Oertell, Gokul Swamy, Kiante Brantley, and Wen Sun. Scaling offline rl via efficient and expressive shortcut models. *arXiv preprint arXiv:2505.22866*, 2025.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taïga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, et al. Stop regressing: Training value functions via classification for scalable deep rl. *arXiv preprint arXiv:2403.03950*, 2024.

Jesse Farebrother, Matteo Pirotta, Andrea Tirinzoni, Rémi Munos, Alessandro Lazaric, and Ahmed Touati. Temporal difference flows. *arXiv preprint arXiv:2503.09817*, 2025.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

Manan S Gandhi, Bogdan Vlahov, Jason Gibson, Grady Williams, and Evangelos A Theodorou. Robust model predictive path integral control: Analysis and performance guarantees. *IEEE Robotics and Automation Letters*, 6(2): 1423–1430, 2021.

Divyansh Garg, Joey Hejna, Matthieu Geist, and Stefano Ermon. Extreme q-learning: Maxent rl without entropy. *arXiv preprint arXiv:2301.02328*, 2023.

Seyed Kamyar Seyed Ghasemipour, Dale Schuurmans, and Shixiang Shane Gu. Emaq: Expected-max q-learning operator for simple yet effective offline and online rl. In *International Conference on Machine Learning*, pp. 3682–3691. PMLR, 2021.

Dibya Ghosh, Chethan Anand Bhateja, and Sergey Levine. Reinforcement learning from passive data via latent intentions. In *International Conference on Machine Learning*, pp. 11321–11339. PMLR, 2023.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. *arXiv preprint arXiv:2203.04955*, 2022.

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.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

Stratton C Jaquette. Markov decision processes with a new optimality criterion: Discrete time. *The Annals of Statistics*, 1(3):496–505, 1973.

Mustafa O Karabag and Ufuk Topcu. On the sample complexity of vanilla model-based offline reinforcement learning with dependent samples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 8195–8202, 2023.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33:21810–21823, 2020.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Deqian Kong, Dehong Xu, Minglu Zhao, Bo Pang, Jianwen Xie, Andrew Lizarraga, Yuhao Huang, Sirui Xie, and Ying Nian Wu. Latent plan transformer for trajectory abstraction: Planning as latent space inference. *Advances in Neural Information Processing Systems*, 37:123379–123401, 2024.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

Jongmin Lee, Wonseok Jeon, Byungjun Lee, Joelle Pineau, and Kee-Eung Kim. Optidice: Offline policy optimization via stationary distribution correction estimation. In *International Conference on Machine Learning*, pp. 6120–6130. PMLR, 2021.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement learning with action chunking. *arXiv preprint arXiv:2507.07969*, 2025.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky TQ Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022.

Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14297–14306, 2023.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Mitsuhiko Nakamoto, Simon Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *Advances in Neural Information Processing Systems*, 36:62244–62269, 2023.

Michal Nauman, Marek Cygan, Carmelo Sferrazza, Aviral Kumar, and Pieter Abbeel. Bigger, regularized, categorical: High-capacity value functions are efficient multi-task learners. *arXiv preprint arXiv:2505.23150*, 2025.

Alexander Nikulin, Vladislav Kurenkov, Denis Tarasov, and Sergey Kolesnikov. Anti-exploration by random network distillation. In *International Conference on Machine Learning*, pp. 26228–26244. PMLR, 2023.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. Hiql: Offline goal-conditioned rl with latent states as actions. *Advances in Neural Information Processing Systems*, 36:34866–34891, 2023.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. *arXiv preprint arXiv:2410.20092*, 2024a.

Seohong Park, Kevin Frans, Sergey Levine, and Aviral Kumar. Is value learning really the main bottleneck in offline rl? *arXiv preprint arXiv:2406.09329*, 2024b.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with hilbert representations. *arXiv preprint arXiv:2402.15567*, 2024c.

Seohong Park, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon reduction makes rl scalable. *arXiv preprint arXiv:2506.04168*, 2025a.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. *arXiv preprint arXiv:2502.02538*, 2025b.

Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Allen Z Ren, Justin Lidard, Lars L Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. *arXiv preprint arXiv:2409.00588*, 2024a.

Juntao Ren, Gokul Swamy, Zhiwei Steven Wu, J Andrew Bagnell, and Sanjiban Choudhury. Hybrid inverse reinforcement learning. *arXiv preprint arXiv:2402.08848*, 2024b.

Jacques Richalet, André Rault, JL Testud, and J Papon. Model predictive heuristic control. *Automatica (journal of IFAC)*, 14(5):413–428, 1978.

Oleh Rybkin, Michal Nauman, Preston Fu, Charlie Snell, Pieter Abbeel, Sergey Levine, and Aviral Kumar. Value-based deep rl scales predictably. *arXiv preprint arXiv:2502.04327*, 2025.

Kajetan Schweighofer, Marius-constantin Dinu, Andreas Radler, Markus Hofmarcher, Vihang Prakash Patil, Angela Bitto-Nemling, Hamid Eghbal-Zadeh, and Sepp Hochreiter. A dataset perspective on offline reinforcement learning. In *Conference on Lifelong Learning Agents*, pp. 470–517. PMLR, 2022.

Harshit Sikchi, Qinqing Zheng, Amy Zhang, and Scott Niekum. Dual rl: Unification and new methods for reinforcement and imitation learning. *arXiv preprint arXiv:2302.08560*, 2023.

Matthew J Sobel. The variance of discounted markov decision processes. *Journal of Applied Probability*, 19(4): 794–802, 1982.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=PxTIG12RRHS`.

Yuda Song, Yifei Zhou, Ayush Sekhari, J Andrew Bagnell, Akshay Krishnamurthy, and Wen Sun. Hybrid rl: Using both offline and online data can make rl efficient. *arXiv preprint arXiv:2210.06718*, 2022.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 1998.

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:11592–11620, 2023.

Tongzhou Wang, Antonio Torralba, Phillip Isola, and Amy Zhang. Optimal goal-reaching reinforcement learning via quasimetric learning. In *International Conference on Machine Learning*, pp. 36411–36430. PMLR, 2023.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

Douglas J White. Mean, variance, and probabilistic criteria in finite markov decision processes: A review. *Journal of Optimization Theory and Applications*, 56(1):1–29, 1988.

Grady Williams, Andrew Aldrich, and Evangelos Theodorou. Model predictive path integral control using covariance variable importance sampling. *arXiv preprint arXiv:1509.01149*, 2015.

Grady Williams, Andrew Aldrich, and Evangelos A Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, 2017.

Runzhe Wu, Yiding Chen, Gokul Swamy, Kianté Brantley, and Wen Sun. Diffusing states and matching scores: A new framework for imitation learning. *arXiv preprint arXiv:2410.13855*, 2024.

Haoran Xu, Li Jiang, Jianxiong Li, Zhuoran Yang, Zhaoran Wang, Victor Wai Kin Chan, and Xianyuan Zhan. Offline rl with no ood actions: In-sample learning via implicit value regularization. *arXiv preprint arXiv:2303.15810*, 2023.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33: 14129–14142, 2020.

Zishun Yu and Xinhua Zhang. Actor-critic alignment for offline-to-online reinforcement learning. In *International Conference on Machine Learning*, pp. 40452–40474. PMLR, 2023.

Shiyuan Zhang, Weitong Zhang, and Quanquan Gu. Energy-weighted flow matching for offline reinforcement learning. *arXiv preprint arXiv:2503.04975*, 2025.

Qinqing Zheng, Matt Le, Neta Shaul, Yaron Lipman, Aditya Grover, and Ricky TQ Chen. Guided flows for generative modeling and decision making. *arXiv preprint arXiv:2311.13443*, 2023.

Jin Peng Zhou, Kaiwen Wang, Jonathan D Chang, Zhaolin Gao, Nathan Kallus, Kilian Q Weinberger, Kianté Brantley, and Wen Sun. q: Provably optimal distributional rl for llm post-training. *CoRR*, 2025a.

Zhongyi Zhou, Yichen Zhu, Junjie Wen, Chaomin Shen, and Yi Xu. Vision-language-action model with open-world embodied reasoning from pretrained knowledge. *arXiv preprint arXiv:2505.21906*, 2025b.

Zhongyi Zhou, Yichen Zhu, Minjie Zhu, Junjie Wen, Ning Liu, Zhiyuan Xu, Weibin Meng, Ran Cheng, Yaxin Peng, Chaomin Shen, et al. Chatvla: Unified multimodal understanding and robot control with vision-language-action model. *arXiv preprint arXiv:2502.14420*, 2025c.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

A Extended Related Work

Offline Reinforcement Learning. Offline RL tackles the problem of learning a policy from a fixed dataset without additional environment interactions (Levine et al., 2020). In addition to the standard reward maximization goal of online RL, the key problem of offline RL is avoiding distribution shift between train-time (i.e. the offline dataset) and test-time (i.e. the learned policy's rollout). Numerous strategies have been proposed for the offline RL setting. A common approach is to employ behavior regularization, which forces the learned policy to stay close to the dataset via behavioral cloning or divergence penalties (Nair et al., 2020; Fujimoto & Gu, 2021; Tarasov et al., 2023). Other approaches include in-distribution maximization (Kostrikov et al., 2021; Xu et al., 2023; Garg et al., 2023), dual formulations of RL (Lee et al., 2021; Sikchi et al., 2023), out-of-distribution detection (Yu et al., 2020; Kidambi et al., 2020; An et al., 2021; Nikulin et al., 2023), and conservative value estimation (Kumar et al., 2020). Farebrother et al. (2024); Nauman et al. (2025) propose training value functions via classification-based objectives, instead of the standard regression-based objectives. Rybkin et al. (2025) propose scaling laws for value-based reinforcement learning. Policies trained via offline RL can subsequently be used for sample efficient online RL in a procedure known as offline-to-online RL (Lee et al., 2021; Song et al., 2022; Nakamoto et al., 2023; Ball et al., 2023; Yu & Zhang, 2023; Ren et al., 2024b; Park et al., 2025b; Li et al., 2025).

Offline Reinforcement Learning with Generative Models. Standard offline RL approaches rely on Gaussian-based models in continuous state-action spaces. However, recent work has focused on representing policies via powerful sequence or generative models Chen et al. (2021); Janner et al. (2021; 2022); Wang et al. (2022); Ren et al. (2024a); Wu et al. (2024); Black et al. (2024); Park et al. (2025b); Espinosa-Dice et al. (2025), taking advantage of more powerful generative models like diffusion (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021) and flow matching (Lipman et al., 2022; Liu et al., 2022; Lipman et al., 2024). These generative models are known to be more expressive than Gaussian-based models, enabling them to capture more complex, multi-modal distributions. Modeling complex distributions is particularly relevant to the offline RL setting, where the offline dataset may be composed of multiple data-generating policies of varying qualities. However, diffusion and flow models rely on an iterative sampling process that can be computationally expensive (Ding & Jin, 2023). To address this problem, some methods utilize a two-stage procedure to first train an expressive generative model on the offline dataset, and then distill it into a one-step model that is then used for policy optimization (Ding & Jin, 2023; Chen et al., 2023; Meng et al., 2023; Park et al., 2025b). Espinosa-Dice et al. (2025) propose an approach for avoiding both distillation and extensive backpropagation through time by leveraging shortcut models for flexible inference, but rely on a standard, Gaussian-based value function. Additionally, generative models have been used for plan generation in offline RL (Zheng et al., 2023) and energy-guided flow and diffusion models, incorporating reward feedback in the flow and diffusion training (Zhang et al., 2025). Farebrother et al. (2025) propose integrating flow matching with Bellman-style updates for successor representation learning.

Inference-Time Scaling in Offline Reinforcement Learning. Inference-time scaling in reinforcement learning often takes the form of leveraging dynamics or world models for additional planning at inference-time. Approaches include model predictive control (Richalet et al., 1978; Hansen et al., 2022), model predictive path integral control (Williams et al., 2015; 2017; Gandhi et al., 2021), model-based offline planning (Hafner et al., 2019; Argenson & Dulac-Arnold, 2020), sequence modeling (Janner et al., 2021; 2022; Kong et al., 2024), planning with world models (Hafner et al., 2023), and Monte Carlo tree search (Chen et al., 2024). Additional approaches include applying rejection sampling to the learned value function at inference-time (Chen et al., 2022; Fujimoto et al., 2019; Ghasemipour et al., 2021; Hansen-Estruch et al., 2023; Park et al., 2024b) or using the gradient of the learned value function to adjust actions at inference-time (Park et al., 2024b). Generative models like flow matching and diffusion models naturally support a form of sequential scaling by increasing the number of steps in the iterative sampling process (Ho et al., 2020; Song et al., 2020; Liu et al., 2022; Lipman et al., 2022). Espinosa-Dice et al. (2025) takes advantage of flexibility in the number of denoising steps used when sampling actions from the policy. However, existing approaches do not leverage generative models for value learning like `EVOR`. By leveraging more expressive models for value learning, `EVOR` can better take advantage of larger, more complex offline datasets.

B Flow-Based Temporal Difference Learning

We restate the flow-based TD objective and describe its derivation.

Distributional Bellman. TD learning uses the Bellman equation to learn a value function by constructing a bootstrap target (i.e. the right-hand side (RHS) of the Bellman equation) (Bellman, 1966; Sutton & Barto, 1998), such that

$$Q(x,a) = r(x,a) + \mathbb{E}_{P,\pi} Q(X', A'). \tag{23}$$

The Bellman equation also -holds under distributions (Jaquette, 1973; Sobel, 1982; White, 1988; Bellemare et al., 2017), such that

$$\underbrace{Z(x,a)}_{\texttt{LHS of Distributional Bellman}} \overset{D}{=} \overbrace{r(x,a) + Z(X', A')}^{\texttt{RHS of Distributional Bellman}} \tag{24}$$

where $Z(X', A')$ denotes the random return.

Goal. At a high-level, flow matching learns how to transport a known prior distribution into a target data distribution. To construct a flow-based TD objective, we set the RHS of the distributional Bellman equation as the target distribution, and match the velocities between the LHS and RHS distributions. We will learn a conditional flow model $s_\theta(\cdot \mid x, a, t)$ that transports base noise $Y_{x,a}(0) \sim \mathcal{N}(0, I_d)$ to a terminal variable $Y_{x,a}(1) \sim R_\theta(\cdot \mid x, a)$, such that the distribution $R_\theta(\cdot \mid x, a) \approx R(\cdot \mid x, a)$.

Conditional Flow Model. We learn a conditional velocity field $s_\theta(y \mid x, a, t)$ that defines the ODE

$$\frac{d}{dt} Y_{x,a}(t) = s_\theta(Y_{x,a}(t) \mid x, a, t), \qquad Y_{x,a}(0) \sim p_0. \tag{25}$$

Solving (i.e. "running") this ODE from $t = 0$ to $t = 1$ is done by integration, giving the terminal random variable

$$Y_{x,a}(1) = Y_{x,a}(0) + \int_0^1 s_\theta(Y_{x,a}(\tau) \mid x, a, \tau) d\tau. \tag{26}$$

Let $R_\theta(\cdot \mid x, a)$ denote the induced terminal distribution. Our goal is to learn $R_\theta(\cdot \mid x, a) \approx R(\cdot \mid x, a)$.

Distributional Bellman. By the definition of discounted reward-to-go,

$$Z(x,a) \overset{D}{=} r(x,a) + Z(X', A'), \tag{27}$$

where $X' \sim P(\cdot \mid x, a)$, $A' \sim \pi_{\text{base}}(\cdot \mid X')$, and $Z(X', A') \sim R(\cdot \mid X', A')$. Equivalently, we can say

$$R(\cdot \mid x, a) = \mathcal{L}\left(r(x,a) + Z'\right), \quad Z' \sim R(\cdot \mid X', A'), \tag{28}$$

where $\mathcal{L}$ is the law of the random variable. Taking expectation of Equation 27 yields

$$\mathbb{E}_{Z \sim R(\cdot|x,a)}[Z] = r(x,a) + \mathbb{E}_{X' \sim P(\cdot|x,a)} \mathbb{E}_{A' \sim \pi_{\text{base}}(\cdot|X')} \mathbb{E}_{Z' \sim R(\cdot|X',A')}[Z']. \tag{29}$$

Flow Integral and Expectation. Going back to the ODE solution, we have

$$Y_{x,a}(1) = Y_{x,a}(0) + \int_0^1 s_\theta(Y_{x,a}(\tau) \mid x, a, \tau) d\tau. \tag{30}$$

Taking expectation first and then applying Fubini's theorem, we have

$$\mathbb{E}\left[Y_{x,a}(1) \mid x, a\right] = \mathbb{E}\left[Y_{x,a}(0)\right] + \mathbb{E}\left[\int_0^1 \left[s_\theta(Y_{x,a}(\tau) \mid x, a, \tau)\right] d\tau \mid x, a\right] \tag{31}$$

$$= \mathbb{E}\left[Y_{x,a}(0)\right] + \int_0^1 \mathbb{E}\left[s_\theta(Y_{x,a}(\tau) \mid x, a, \tau) \mid x, a\right] d\tau. \tag{32}$$

By definition of p_0 being zero mean, $\mathbb{E}[Y_{x,a}(0)] = 0$, leaving us with:

$$\mathbb{E}\left[Y_{x,a}(1) \mid x, a\right] = \int_0^1 \mathbb{E}\left[s_\theta(Y_{x,a}(\tau) \mid x, a, \tau) \mid x, a\right] d\tau. \tag{33}$$

If we perform flow matching well, such that $R_\theta(\cdot \mid x, a) \approx R(\cdot \mid x, a)$ (subject to finite sample and optimization errors), then

$$\int_0^1 \mathbb{E}\left[s_\theta(Y_{x,a}(\tau) \mid x, a, \tau) \mid x, a\right] d\tau = r(x, a) + \mathbb{E}_{X',A'}\left[\int_0^1 \mathbb{E}\left[s_\theta(Y_{X',A'}(\tau) \mid X', A', \tau) \mid X', A'\right] d\tau\right]. \tag{34}$$

The equation above is a necessary, integral-level condition. We additionally consider a stronger condition that is pointwise in t, such that for all $t \in [0, 1]$,

$$\mathbb{E}\left[s_\theta(Y_{x,a}(t) \mid x, a, t) \mid x, a\right] = r(x, a) + \mathbb{E}_{X',A'}\mathbb{E}\left[s_\theta(Y_{X',A'}(t) \mid X', A', t) \mid X', A'\right], \quad \forall t \in [0, 1]. \tag{35}$$

Flow-Based TD Objective. Putting this all together, we have the flow-based TD loss

$$\mathcal{L}_{\text{FlowTD}}(\theta) = \underbrace{\mathbb{E}_{(x,a,r,x')\sim\mathcal{D}}}_{\texttt{Dataset's State-Action-Reward}} \overbrace{\mathbb{E}_{z^1\sim R_{\bar{\theta}}(\cdot\mid x,a)}}^{\texttt{Sample Reward-To-Go}} \underbrace{\mathbb{E}_{t\sim\text{Unif}(0,1)}}_{\texttt{Sample Time}} \Big\| \overbrace{s_\theta(z^t \mid x, a, t)}^{\texttt{Velocity Prediction of LHS}} - \underbrace{\text{target}(x, a, z^t, t)}_{\texttt{Velocity Target of RHS}} \Big\|_2^2], \tag{36}$$

where

$$\text{target}(x, a, z^t, t) := r(x, a) + \mathbb{E}_{a'\sim\pi_{\text{base}}(\cdot\mid x')} s_{\bar{\theta}}(z^t \mid x', a', t). \tag{37}$$

We sample a state-action-reward-next-state tuple $(x, a, r, x') \sim \mathcal{D}$ from the offline data, a time $t \sim \text{Unif}(0, 1)$, and the next action from the base policy $a' \sim \pi_{\text{base}}(\cdot \mid x')$. We construct an interpolant $z^t = (1 - t)z^0 + tz^1$, which serves to noise the ground-truth sample, by sampling a reward-to-go $z^1 \sim R(\cdot \mid x, a)$ and a noise sample $z^0 \sim \mathcal{N}(0, I_d)$. The reward-to-go sample z^1 can be sampled from the dataset or a target version of the learned reward model $R_{\bar{\theta}}(\cdot \mid x, a)$. To sample a reward-to-go from the distribution $R_\theta(\cdot \mid x, a)$, we employ the standard forward Euler method with the learned flow model $s_\theta(\cdot \mid x, a, t)$.

C FULL RESULTS

Table 2: **EVOR's Overall Performance By Task.** We present the full results on each OGBench task. `(*)` indicates the default task in each environment. The results are averaged over 3 seeds with standard deviations reported.

Task	QC-1	QC-5	EVOR
antmaze-large-navigate-singletask-task1-v0 (*)	4 ±3	2 ±1	**14** ±3
antmaze-large-navigate-singletask-task2-v0	0 ±0	0 ±0	**61** ±4
antmaze-large-navigate-singletask-task3-v0	**53** ±19	27 ±12	33 ±5
antmaze-large-navigate-singletask-task4-v0	2 ±1	0 ±0	**66** ±11
antmaze-large-navigate-singletask-task5-v0	0 ±0	4 ±3	**69** ±6
antmaze-large-stitch-singletask-task1-v0 (*)	**3** ±0	2 ±1	0 ±0
antmaze-large-stitch-singletask-task2-v0	**0** ±0	0 ±0	**0** ±1
antmaze-large-stitch-singletask-task3-v0	4 ±1	12 ±14	**67** ±5
antmaze-large-stitch-singletask-task4-v0	0 ±0	0 ±0	**4** ±2
antmaze-large-stitch-singletask-task5-v0	**20** ±15	12 ±5	5 ±4
cube-double-play-singletask-task1-v0 (*)	**90** ±5	73 ±5	**95** ±2
cube-double-play-singletask-task2-v0	54 ±4	46 ±20	**96** ±2
cube-double-play-singletask-task3-v0	48 ±12	49 ±21	**96** ±2
cube-double-play-singletask-task4-v0	22 ±5	30 ±5	**36** ±8
cube-double-play-singletask-task5-v0	59 ±8	52 ±25	**87** ±6
pointmaze-medium-navigate-singletask-task1-v0 (*)	**97** ±3	**99** ±1	**100** ±1
pointmaze-medium-navigate-singletask-task2-v0	89 ±7	**100** ±1	**99** ±2
pointmaze-medium-navigate-singletask-task3-v0	**100** ±0	**99** ±2	**98** ±2
pointmaze-medium-navigate-singletask-task4-v0	94 ±7	**100** ±0	**100** ±0
pointmaze-medium-navigate-singletask-task5-v0	**100** ±0	**100** ±0	**100** ±0
scene-play-singletask-task1-v0	94 ±2	**100** ±0	**100** ±0
scene-play-singletask-task2-v0 (*)	87 ±2	**99** ±1	**98** ±1
scene-play-singletask-task3-v0	44 ±7	**93** ±3	**94** ±2
scene-play-singletask-task4-v0	1 ±1	**90** ±4	76 ±28
scene-play-singletask-task5-v0	0 ±0	41 ±14	**60** ±17

Q: What is EVOR's task-level performance?

Across 5 environments and 25 unique tasks, EVOR achieves the best performance compared to the baselines. EVOR's expressive value learning method outperforms standard value learning methods. From the results in Table 2, we observe that EVOR outperforms or matches standard value function learning methods (QC), even compared to a method that employs action chunking (QC-5), suggesting that expressive value learning can improve performance over standard value function learning.

D ABLATION STUDIES

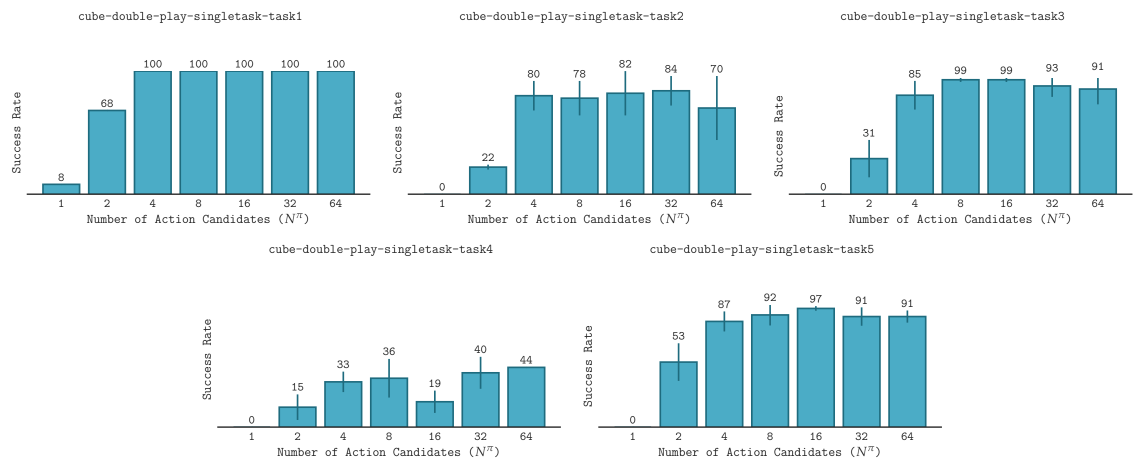

Figure 3: **Ablation Over Number of Action Candidates N_π.** Results are averaged over 3 seeds per task, with standard deviations reported.

Q: [Task-Level] How can EVOR take advantage of greater inference-time compute?

As shown in Figure 3, when given access to greater inference-time compute, EVOR can increase the number of action candidates N_π, resulting in better performance (up to a saturation point).

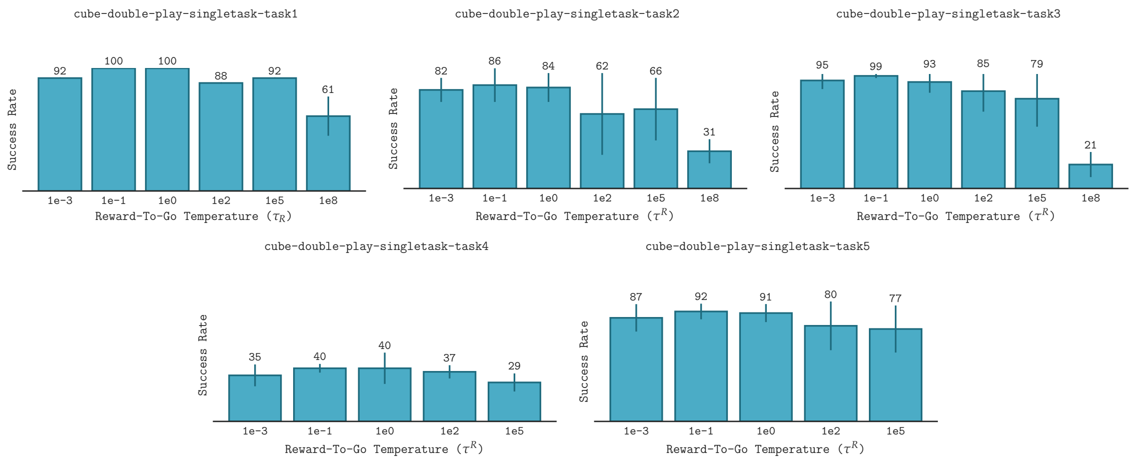

Figure 4: **Ablation Over Reward-To-Go Temperature Parameter τ_R.** Results are averaged over 3 seeds per task, with standard deviations reported.

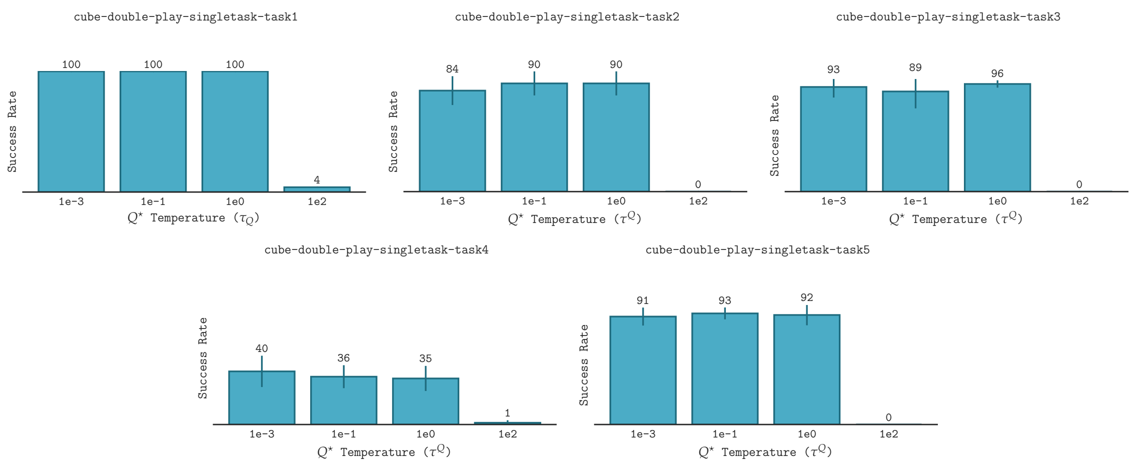

Figure 5: **Ablation Over $Q^\star$ Temperature Parameter τ_Q.** Results are averaged over 3 seeds per task, with standard deviations reported.

Q: [Task-Level] How can EVOR perform inference-time regularization?

As shown in Figure 4 and Figure 5, by increasing varying the temperature parameters τ_R and τ_Q, EVOR can vary the level of regularization to the base policy compared to the level of policy optimization. As τ_Q decreases, the action selection becomes more greedy, while as τ_Q increases, the action selection becomes more regularized. Set to a high value, EVOR becomes equivalent to the base policy (i.e. the performance with $N_\pi = 1$).

E EXPERIMENTAL AND IMPLEMENTATION DETAILS

In this section, we describe the setup, implementation details, and baselines used in the paper.

E.1 EXPERIMENTAL SETUP

We follow OGBench's official evaluation scheme (Park et al., 2024a), with the reward-maximizing offline setup of Park et al. (2025b); Espinosa-Dice et al. (2025). We restate the experimental setup here. Following Park et al. (2025b); Espinosa-Dice et al. (2025), we use OGBench's `singletask` variants for all experiments, corresponding to reward-based tasks that are suitable for our reward-maximizing offline RL setting.

Environments and Tasks. `EVOR` is evaluated on manipulation and locomotion robotics tasks in version `1.1.0` of OGBench (Park et al., 2024a), including

1. `antmaze-large-navigate-singletask-v0`
2. `antmaze-large-stitch-singletask-v0`
3. `cube-double-play-singletask-v0`
4. `pointmaze-medium-navigate-singletask-v0`
5. `scene-play-singletask-v0`

We use the 5 unique tasks (e.g. `antmaze-large-navigate-singletask-task{1,2,3,4,5}-v0`) for each environment listed above, where each task provides a unique evaluation goal. Each environment's dataset is labeled with a semi-sparse reward (Park et al., 2024a; 2025b). For the `cube-double-play-singletask-v0` environment, we use the 100M size dataset provided by Park et al. (2025a).

The selected environments consist of locomotion and manipulation control problems. The `antmaze` tasks consist of navigating a quadrupedal agent (8 degrees of freedom) through complex mazes. The `cube` and `scene` environments manipulated objects with a robotic arm. The goal of `scene` tasks is to sequence multiple subtasks. The environments are state-based. We test both `navigate` and `stitch` datasets for locomotion and `play` for manipulation. These datasets are built from suboptimal, goal-agnostic trajectories, which poses a challenge for goal-directed policy learning (Park et al., 2024a). Following Park et al. (2025b); Espinosa-Dice et al. (2025), we evaluate agents using binary task success rates (i.e., goal completion percentage), which is consistent with OGBench's evaluation setup (Park et al., 2024a).

Evaluation. We follow OGBench's official evaluation scheme (Park et al., 2024a). Algorithms are trained for 1,000,000 gradient steps and evaluated on 50 episodes every 100,000 gradient steps. The average success rates of the final three evaluations (i.e. the evaluation results at 800,000, 900,000, and 1,000,000 gradient steps) are reported. Tables average over 3 seeds per task and report standard deviations, bolding values within 95% of the best performance.

E.2 `EVOR` IMPLEMENTATION DETAILS

Flow Matching. `EVOR` is implemented on top of Li et al. (2025)'s open-source implementation of `QC`, which is adapted from Park et al. (2024a)'s open-source codebase. We implement flow matching using the same, standard velocity field as `QC`.

Network Architecture and Optimizer. Following Park et al. (2025b); Espinosa-Dice et al. (2025); Li et al. (2025), we use a multi-layer perceptron with 4 hidden layers of size 512 for both the value and policy networks. We apply layer normalization (Ba et al., 2016) to value networks and use the Adam optimizer (Kingma, 2014). All of these parameters are shared between `EVOR` and the baselines.

Hyperparameters. We use the same hyperparameters for both `EVOR` and `QC`. Unlike many offline RL algorithms (Park et al., 2025b; Espinosa-Dice et al., 2025), `EVOR` does not change training parameters between environments in this paper. `EVOR` uses $N = 1$ during training (instead of the $N > 1$ used during evaluation) for better efficiency.

Algorithm 3: π_{base} Action Sampling via Forward Euler Method

Input: State x, number of inference steps M
Output: Action a

```
a ~ N(0, I)                                   Sample starting action noise
t ← 0
for m ∈ {0, ..., M} do
    a ← a + (1/M) v_φ(a, t, | x)              Follow ODE
    t ← t + 1/M
return a
```

$a \sim \mathcal{N}(0, I)$ — Sample starting action noise
$t \leftarrow 0$
for $m \in \{0, \ldots, M\}$ **do**
  $a \leftarrow a + \frac{1}{M} v_\phi(a, t, \mid x)$ — Follow ODE
  $t \leftarrow t + \frac{1}{M}$
return a

Inference Procedure. EVOR's inference procedure is shown in Algorithm 2. Actions are sampled from the base policy π_{base} via the forward Euler method, shown in Algorithm 3.

Recall that the *optimal* Q-function is given by:

$$Q^\star(x, a) = \eta \ln \mathbb{E}_{r \sim R(\cdot|x,a)} \exp(r/\eta), \tag{38}$$

where R is the conditional distribution of rewards-to-go under π_{ref}. We learn an estimate of R via the flow-based TD objective, such that $R_\theta(\cdot \mid x, a) \approx R(\cdot \mid x, a)$. We approximate the expectation via sample averaging, as shown in Algorithm 2, such that

$$\text{LogSumExp}(z^{(j)}) = \tau^\star \log \frac{1}{N} \sum_{j=1}^{N} \exp\left(\frac{z^{(j)}}{\tau^\star}\right) \tag{39}$$

We then construct a weighted softmax via the $Q^\star$ approximation, such that

$$\text{softmax}(Q^\star(x, a^{(j)})) = \frac{\exp(Q^\star(x, a^{(j)})/\tau)}{\sum_{j=1}^{N_\pi} \exp(Q^\star(x, a^{(j)})/\tau)} \tag{40}$$

Table 3: Shared Hyperparameters Between QC Baselines and EVOR.

Parameter	Value
Learning rate	3e-4
Optimizer	Adam (Kingma, 2014)
Gradient Steps	1e6
Minibatch Size	256
MLP Dimensions	[512, 512, 512, 512]
Target Network Smoothing Coefficient	5e-3
Discount Factor γ	0.99
Discretization Steps	10
Time Sampling Distribution	Unif([0,1])
Number of Action Candidates N_π	32

Table 4: Hyperparameters for EVOR.

Hyperparameter	Value
Beta β	1e-3
$Q^\star$ Beta $\beta^\star$	1
Number of RTG Samples N^{RTG}	1 (Train), 50 (Eval)

Table 5: Hyperparameters for QC.

Hyperparameter	Value
Action Chunk Length	1 (QC-1), 5 (QC-5)
Critic Ensemble Size	2

E.3 BASELINES

Rather than compare to all of the existing offline RL algorithms benchmarked on OGBench, we instead aim to isolate the effect of expressive value learning in order to demonstrate its benefit specifically. Thus, we compare to Q-chunking (`QC`, Li et al. (2025)), a recent offline RL algorithm that is closest to `EVOR`. Like `EVOR`, `QC` learns a base policy via flow matching and extracts an optimized policy via rejection sampling. The key difference between `QC` and `EVOR` is in how the value function is learned, which is the exact difference we aim to isolate. `QC` can employ action chunking in both its policy and value function, and we compare `EVOR` to both `QC` with (`QC-5`) action chunking and without it (`QC-1`). We select the action chunk length (5) based on Li et al. (2025)'s recommendation.

Given an action chunk length of k, represented as $\boldsymbol{a}_{t:t+k} = (a_t, a_{t+1}, \ldots, a_{t+k})$, the Q-function is updated via

$$Q(x_t, \boldsymbol{a}_{t:t+k}) \leftarrow \sum_{t'=1}^{t+k-1} [r_{t'}] + Q(x_{t+k}, \boldsymbol{a}_{t+k:t+2k}) \tag{41}$$

and actions are sampled via

$$\boldsymbol{a} \leftarrow \operatorname*{argmax}_{\boldsymbol{a} \in \{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_N\}} Q(x, \boldsymbol{a}), \tag{42}$$

where $a_1, a_2, \ldots, a_N \sim \pi_{\text{base}}(\cdot \mid x)$. This yields the following loss function for learning the Q-function:

$$L(\theta) = \mathbb{E}_{\substack{x_t, \boldsymbol{a}_t \sim \mathcal{D} \\ \{a_{t+k}^i\}_{i=1}^N \sim \pi_{\text{base}}(\cdot \mid x_{t+k})}} \left[\left(Q_\theta(x_t, \boldsymbol{a}_t) - \sum_{t'=1}^{k} r_{t+t'} - Q_{\bar{\theta}}(x_{t+k}, \boldsymbol{a}_{t+k}) \right)^2 \right], \tag{43}$$

where $\boldsymbol{a}_{t+k} = \operatorname{argmax}_{a \in \{\boldsymbol{a}_{t+k}^i\}} Q(s, \boldsymbol{a})$.