VALUE FORCING: ALIGNING RETURN DISTRIBUTIONS AT ALL NOISE LEVELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning accurate value functions is central to reinforcement learning (RL), yet remains challenging because supervision is limited and indirect: ground-truth values are unavailable, and each transition in the data only provides an immediate reward that can be used to construct bootstrapped targets as supervision. This challenge is particularly pronounced when data is limited or reward signals are sparse, as is common in real-world applications such as robotics. We argue that the key to accurate value learning in this *limited* data regime is extracting richer learning signals from the available data. In this paper, we introduce *Value Forcing*, a new approach that uses flow matching to estimate return distributions that enables accurate value learning from limited data. Our key insight is that we can extract richer learning signals by enforcing Bellman consistency at every noise level. Empirically, Value Forcing improves the accuracy of estimating the return distribution, and prevents the distribution from collapsing to a single value, as occurs with prior methods. We instantiate Value Forcing with two practical value-learning algorithms: VF, an actor-critic method, and IVF, an implicit value-learning method. We evaluate both algorithms across 20 long-horizon tasks in the offline RL setting and find that they outperform prior value-learning methods, with particularly large gains when only limited data is available.

1 INTRODUCTION

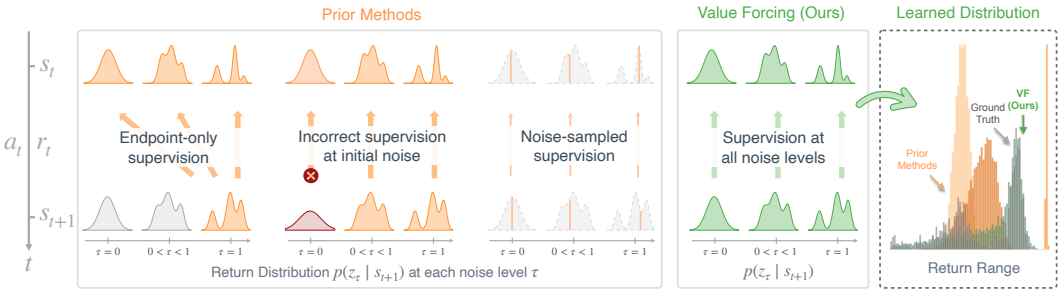


Figure 1: We introduce **Value Forcing**, a new approach to learning distributional value functions with flow matching that supervises the learned return distribution at every flow noise level through Bellman consistency. Unlike prior methods that provide supervision only at selected noise levels or use inaccurate targets at others, Value Forcing provides substantially richer learning signals and enables much more accurate estimation of the return distribution, preventing it from collapsing around a single value. This in turn leads to better policy performance, especially in the limited-data regime.

Value estimation lies at the heart of reinforcement learning (RL): value functions estimate the expected return of following a certain course of actions, guiding agents to select better actions that improve long-term returns. However, accurately estimating the values is hard, especially with *limited* amounts of data or if the reward feedback is sparse. Supervision for long-term value is limited and indirect: each observed transition provides only an immediate reward, but no ground-truth expected return. Temporal-difference (TD) methods construct prediction targets using their own value estimates, which could be inaccurate and bias learning. With limited data, there is little room to correct these errors. However, such settings are common in real-world applications such as robotics. This raises the question: *is there a way to extract richer learning signals from limited available transitions?*

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Many widely used RL methods learn scalar value functions that summarize the future outcome of a policy. This scheme has limited expressivity and may give a sparse supervision signal when used as the target for TD learning. Distributional RL instead models the full return distribution and can provide richer learning signals when the whole return distribution is used as the TD target (Wang et al., 2023; Rowland et al., 2023). For example, an action that produces returns of -1 and $+1$ with equal probability has the same expected return as another action that always produces 0 return, yet their distributions differ. Learning to distinguish these two actions and their outcomes can enable more effective use of the available data.

Recent work has begun to use modern generative models such as flow matching to represent the return distribution. These methods learn to map Gaussian noise to the return distribution through an iterative denoising process. Beyond capturing complex return distributions, flow matching also creates an opportunity to provide even richer supervision by learning at different noise levels (Farebrother et al., 2025; Chen et al., 2024). However, prior flow-based return estimators typically only provide supervision using the endpoint of the flow path (Chen et al., 2025; Zhong et al., 2025; Groom et al., 2026; Agrawalla et al., 2026), missing the opportunity to supervise all other noise levels. While others have explored applying Bellman consistency at all noise levels, they either suffer from enforcing an incorrect consistency at the initial flow step (Dong et al., 2025) or only apply the consistency along the flow paths starting from the same noise (Xu et al., 2026) (Fig. 1 *(Center)*). Empirically, we find that sparse supervision or incorrect dense supervision can lead to inaccurate modeling of the return distribution, and often results in the learned return distribution collapsing toward a point mass (see Fig. 2), thereby undermining the effectiveness of distributional value learning.

In this paper, we introduce Value Forcing, a new distributional value learning approach that enforces Bellman consistency at every noise level (see illustration in Fig. 1). At each noise level, Value Forcing uses the next-state marginal return distribution to supervise the current-state marginal return distribution. Unlike the standard Bellman operator, we introduce a new flow Bellman operator that includes a Gaussian correction. We show that Value Forcing has a unique fixed point at every noise level, preserving the initial Gaussian and recovering the distributional Bellman equation at the flow endpoint. Empirically, we find that Value Forcing can model the return distribution much more accurately compared to the prior methods and does not suffer from distribution collapse (Fig. 1).

We instantiate two practical value-learning algorithms with Value Forcing: VF, an actor-critic algorithm, and IVF, an implicit method via expectile-weighted flow matching. Theoretically, we show that IVF approaches the optimal value within actions presented in the dataset as the expectile increases. Empirically, we evaluate both VF and IVF across 20 long-horizon OGBench tasks and show that they improve return estimation and policy performance over prior flow-based value learning methods, with particularly strong gains when data is limited.

2 Related Work

Offline RL aims to learn a reward-maximizing policy from a fixed offline dataset. The core challenge in this setting is to learn a value function while preventing erroneous extrapolation outside the distribution of the dataset. Previous works have proposed strategies based on applying conservative penalties for out-of-distribution samples (Kumar et al., 2020; Nakamoto et al., 2024; Bai et al., 2022; Chae et al., 2026), constraining action selection in Bellman backups to the dataset support (Fujimoto et al., 2019; Kumar et al., 2019; Singh et al., 2022), or not querying samples outside the offline dataset at all (Kostrikov et al., 2022; Hansen-Estruch et al., 2023; Xu et al., 2023). While most previous works focus on training an accurate value function with the entire offline dataset, we focus on the setting with *extremely limited* offline data. Most similar to our focus is the work of Xu et al. (2023); Park et al. (2026), which also compares different value learning algorithms on subsets of the entire offline datasets. However, these methods learn a scalar return value, while we aim to learn the whole distribution of the return value using flow matching. This offers denser supervision during TD updates, and in practice leads to better performance in the limited dataset setting. We find that our method matches the state of the art on the full dataset, but outperforms it by a large margin when data is more limited.

Distributional and generative value functions. Distributional value functions aim to learn the entire distribution of the discounted cumulative return instead of a single scalar that estimates its expected value (Bellemare et al., 2017; 2023). Early methods model the value distribution as a discrete categorical distribution (Bellemare et al., 2017; Wang et al., 2026; Dabney et al., 2018b) or continuous

distributions with restrictive modeling assumptions (Ma et al., 2025; Dabney et al., 2018a). Recent methods have leveraged diffusion and flow matching to represent complex return distributions (Dong et al., 2025; Zhong et al., 2025; Chen et al., 2025; Espinosa-Dice et al., 2025) and their expected values (Agrawalla et al., 2026). Our method falls into the former category, where methods differ primarily in where the Bellman consistency is enforced in the denoising path. DFC (Chen et al., 2025), FlowCritic (Zhong et al., 2025), FlowIQN (Groom et al., 2026), and Dual-Flow RL (Li et al., 2026) construct clean TD targets at the flow endpoint before applying the flow-matching objectives. FlowIQN further introduces a quantile-aligned coupling between initial noises and final samples. Dong et al. (2025), Espinosa-Dice et al. (2025), and Xu et al. (2026) introduce stronger constraints along the flow path. Value Flows (Dong et al., 2025) and EVOR (Espinosa-Dice et al., 2025) derive flow-matching objectives by bootstrapping the velocity field directly, but naively apply the clean Bellman equation at arbitrary noise levels. Thus, the Bellman equation does not enter the intermediate noise levels. PCBF (Xu et al., 2026) couples two flow paths using the same initial noise and applies the Bellman operator at all noise levels along the coupled paths. This strategy additionally enforces Gaussian distributions at the initial noise level. However, the marginal flow paths lack Bellman consistency because of the coupling. In contrast, we define a flow Bellman operator whose fixed point aligns the return distributions at every noise level. Doing so results in more accurate return estimation, especially in the low-data regime.

3 Preliminaries

We consider an infinite-horizon Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \rho, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition distribution, $r : \mathcal{S} \times \mathcal{A} \to [r_{\min}, r_{\max}]$ is the *deterministic* bounded reward function, $\rho \in \Delta(\mathcal{S})$ is the initial state distribution, and $\gamma \in [0, 1)$ is the discount factor. Here, $\Delta(\cdot)$ denotes the set of all possible probability distributions. Throughout our discussions, we will use $t \in \mathbb{N}$ to denote the time step for RL and use $\tau \in [0, 1]$ to denote the noise level for flow-matching. We focus on offline RL, where the agent learns from a fixed dataset $D = \{(s, a, r, s')\}$ collected by a behavior policy. The goal is to learn a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the expected discounted return: $\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$.

Distributional RL. Instead of modeling the expected discounted return, distributional RL (Bellemare et al., 2017; Dabney et al., 2018a) models the full return distribution. For a policy π, the return random variable is defined as $Z^\pi(s, a) = \sum_{t=0}^{\infty} \gamma^t r(S_t, A_t)$, where (S_t, A_t) are state and action random variables at time t. We use $\mathcal{Z}$ to denote the set of return distributions with uniformly bounded moments (Bellemare et al., 2017) for all possible policies in this MDP. Taking the expectation of $Z^\pi(s, a)$ recovers the Q function: $Q^\pi(s, a) = \mathbb{E}_\pi \left[Z^\pi(s, a) \mid S_0 = s, A_0 = a\right]$. Prior work (Bellemare et al., 2017) has shown that $Z^\pi(s, a)$ satisfies the distributional Bellman equation

$$Z^\pi(s, a) \stackrel{d}{=} \mathcal{T}^\pi Z^\pi(s, a) \stackrel{d}{=} r(s, a) + \gamma Z^\pi(S', A'), \tag{1}$$

where (S', A') are random variables of the state and action at the next time step, $\stackrel{d}{=}$ indicates equality in distribution, and $\mathcal{T}^\pi$ is the distributional Bellman operator. This operator is a γ-contraction under the maximum Wasserstein metric with the unique fixed point $Z^\pi(s, a)$. Taking expectations of Eq. 1 recovers the standard Bellman equations for the Q function.

Flow matching. Flow matching (Lipman et al., 2022; Albergo & Vanden-Eijnden, 2023; Liu et al., 2022) is a generative modeling framework that learns a continuous velocity field to transport samples from a noise distribution to a target distribution. Let $u(x^\tau \mid \tau) : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d$ denote a velocity field that depends on noise levels. The corresponding flow path $\phi(x^\tau \mid \tau) : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d$ is defined as the solution to the following ordinary differential equation (ODE) (Lee, 2012)

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\phi(x^0 \mid \tau) = u\left(\phi(x^0 \mid \tau) \mid \tau\right), \qquad \phi(x^0 \mid 0) = x^0. \tag{2}$$

Starting from an initial noise distribution (e.g., the standard Gaussian $\mathcal{N}(0, I_d)$) at noise level $\tau = 0$, we learn the flow to transport samples to match the target distribution at noise level $\tau = 1$ by minimizing the conditional flow-matching objective (Lipman et al., 2022):

$$\mathbb{E}\left[\left\|v(x^\tau \mid \tau) - (x^1 - x^0)\right\|_2^2\right], \tag{3}$$

where $x^0 \sim \mathcal{N}(0, I_d)$ is a noise, x^1 is a sample from the target distribution, and $x^\tau = (1-\tau)x^0 + \tau x^1$ is the linear interpolation at noise level $\tau \sim \textsc{Unif}([0, 1])$. Prior work (Liu et al., 2022; Albergo et al., 2025) has shown that this loss is related to the noise-level differentiation of a stochastic process:

Lemma 1 (Informal; Sec. 2.3 and Sec. 3.3 of Liu et al. (2022)). *Let the stochastic process $X(\tau)$ that depends on the noise levels connect the initial random variable X^0 and the target random variable X^1, i.e., $X(0) = X^0$ and $X(1) = X^1$. Let $\mathcal{X}$ collect all random variables defining the stochastic process, i.e., $X(\tau)$ is a deterministic function of noise level τ given $\mathcal{X}$. Given a noisy sample x^τ at noise level τ, the velocity field defined as $u(x^\tau \mid \tau) \triangleq \mathbb{E}_{\mathcal{X}}\left[\frac{\mathrm{d}}{\mathrm{d}\tau}X(\tau) \,\middle|\, X(\tau) = x^\tau\right]$ generates (Lipman et al., 2022) the probability density $p(x^\tau \mid \tau)$ of the stochastic process. In addition, we can use a parametric velocity $v(x^\tau \mid \tau)$ to fit $u(x^\tau \mid \tau)$ by minimizing $\mathbb{E}_{\mathcal{X},\tau\sim\mathrm{UNIF}([0,1])}\left[\left\|v(x^\tau \mid \tau) - \frac{\mathrm{d}}{\mathrm{d}\tau}X(\tau)\right\|^2\right]$.*

Importantly, we can interpret the flow path as a stochastic process, where $u(x^\tau \mid \tau)$ and $p(x^\tau \mid \tau)$ are the marginal velocity and marginal probability path. We can recover the standard conditional flow matching loss (Eq. 3) by setting $\mathcal{X} = (X^0, X^1)$ and $X(\tau) = (1-\tau)X^0 + \tau X^1$. We will use this lemma to construct the flow matching objectives for fitting return distributions in Sec. 5.

4 Bellman Consistency at All Noise Levels

To extract richer signals from each offline transition, we enforce Bellman consistency throughout the return flow via a novel noise-dependent flow Bellman operator. We first derive this operator below, and show that it is consistent with the standard Bellman operator at the flow endpoints and has a unique fixed point at all noise levels. We later use this flow Bellman operator to derive practical flow-matching losses in Sec. 5.

4.1 Flow Bellman Equation

To extend the Bellman equation to the entire flow path, we first define the noisy return random variable at noise level τ as $Z^\pi(s,a,\tau) \in \mathcal{Z}$. At $\tau = 1$, $Z^\pi(s,a,1) \overset{d}{=} Z^\pi(s,a)$. Then we introduce a novel *flow Bellman operator* $\mathcal{T}_\tau^\pi : \mathcal{Z} \to \mathcal{Z}$, which also depends on the noise level τ and satisfies the ***flow Bellman equation***:

$$Z^\pi(s,a,\tau) \overset{d}{=} \mathcal{T}_\tau^\pi Z^\pi(s,a,\tau) \overset{d}{=} \tau r(s,a) + \gamma Z^\pi(S',A',\tau) + (1-\tau)\sqrt{1-\gamma^2}N. \tag{4}$$

Compared to the distributional Bellman equation in Eq. 1, this incorporates a standard Gaussian $N \sim \mathcal{N}(0,1)$ that is *independent* of the noise distribution. This Gaussian correction[1] is needed to account for the variance in the discounted return at the next RL time step. To understand this intuitively, we can examine the equation at the two ends of the flow path. At $\tau = 0$, the flow Bellman equation becomes $Z^\pi(s,a,0) \overset{d}{=} \gamma Z^\pi(S',A',0) + \sqrt{1-\gamma^2}N$. Note that both $Z^\pi(s,a,0)$ and $Z^\pi(S',A',0)$ are standard Gaussian because they are initial flow noises, so the Guassian correction N is essential for the equation to hold. Omitting the $\sqrt{1-\gamma^2}N$ term, as done in prior work (Dong et al., 2025), results in an incorrect equation. At $\tau = 1$, we recover the standard distributional Bellman equation in Eq. 1.

Unique fixed point at all noise levels. To provide a self-consistent target for recursive value learning, Bellman operators must provide a fixed point. In the following, we show that our flow Bellman operator in fact admits a unique fixed point at every noise level ($\tau \in [0,1]$). This allows us to impose Bellman consistency along the entire flow path, providing richer supervision signals. We start by showing that $\mathcal{T}_\tau^\pi$ is a contraction.

Remark 1 (Contraction). *For every noise level τ and a $p \in [1,\infty)$, $\mathcal{T}_\tau^\pi$ is a γ-contraction on two return random variables, Z_1 and Z_2, in $\mathcal{Z}$:*

$$W_p^{sup}(\mathcal{T}_\tau^\pi Z_1, \mathcal{T}_\tau^\pi Z_2) \le \gamma W_p^{sup}(Z_1, Z_2) \tag{5}$$

where W_p^{sup} is the supremum of the p-Wasserstein distance over state-action pairs.

We provide a formal statement and a proof in Appendix A.1. Since $\mathcal{T}_\tau^\pi$ is a γ-contraction, it follows from the contraction mapping theorem (Banach, 1922) that our flow Bellman operator admits the unique fixed point $Z^\pi(s,a,\tau)$. In other words, repeatedly applying $\mathcal{T}_\tau^\pi$ to an arbitrary member of $\mathcal{Z}$ will converge to this fixed point. With such a guarantee, we have shown that our proposed flow Bellman operator is well-defined and can provide a consistent target for value-learning. Next, we derive practical flow matching objectives with this operator.

[1] Using other distributions to sample the initial noise, e.g. $\mathrm{UNIF}([0,1])$, results in different correction terms.

5 Learning return distributions with the flow Bellman operator

In Sec. 4, we introduced a flow Bellman operator that can enforce Bellman consistency at every noise level of flow matching. However, it is not straightforward how one can use this operator to model state-value and state-action value distributions in practice. In this section, we derive the practical flow-matching losses.

We denote the state-value return random variable as $Z_V^\pi(s,\tau)$ and the state-action-value return random variable as $Z_Q^\pi(s,a,\tau)$. We can decompose the flow Bellman equation (Eq. 4) into two parts:

$$Z_V^\pi(s,\tau) \overset{d}{=} Z_Q^\pi(s,A,\tau), \quad Z_Q^\pi(s,a,\tau) \overset{d}{=} \tau r(s,a) + \gamma Z_V^\pi(S',\tau) + (1-\tau)\sqrt{1-\gamma^2}N, \tag{6}$$

where A is the action random variable. The first equation relates $Z_V^\pi(s,\tau)$ to $Z_Q^\pi(s,A,\tau)$ via actions sampled from the policy, and the second equation corresponds to the temporal-difference backup. To model this with flow matching, we need to use flow processes ϕ_V^π and ϕ_Q^π to transport independent Gaussian random variables N_V, N_Q into these laws:

$$\phi_V^\pi(N_V \mid s,\tau) \overset{d}{=} \phi_Q^\pi(N_Q \mid s,A,\tau), \tag{7}$$

$$\phi_Q^\pi(N_Q \mid s,a,\tau) \overset{d}{=} \tau r(s,a) + \gamma \phi_V^\pi(N_V \mid S',\tau) + (1-\tau)\sqrt{1-\gamma^2}N. \tag{8}$$

ϕ_V^π and ϕ_Q^π solve their corresponding ODEs with velocities u_V^π and u_Q^π, respectively.

Learning the V-flow ϕ_V^π. ϕ_V^π in Eq. 7 is a marginal distribution over the Gaussian random variable N_Q and the action random variable A. We can combine these into the set $\mathcal{X}_V = \{N_Q, A\}$ and use it to determine the stochastic process $X_V^\pi(s,\tau) \triangleq \phi_Q^\pi(N_Q \mid s,A,\tau)$ and its noise-level differentiation $\frac{\mathrm{d}}{\mathrm{d}\tau}X_V^\pi(s,\tau) = u_Q^\pi(\phi_Q^\pi(N_Q \mid s,A,\tau) \mid s,A,\tau)$. In $\frac{\mathrm{d}}{\mathrm{d}\tau}X_V^\pi(s,\tau)$, we plug in the ODE between ϕ_Q^π and u_Q^π as in Eq. 2. By Lemma 1, defining the stochastic process reveals the velocity of the state-value return, and we can take expectations over $\mathcal{X}_V$ and use a parametric state-value velocity $v_V(z^\tau \mid s,\tau)$ to fit $u_V^\pi(z^\tau \mid s,\tau)$ by minimizing

$$\mathbb{E}_{\substack{a\sim\pi(a|s),\ \tau\sim\textsc{Unif}([0,1]),\\ z^0\sim\mathcal{N}(0,1),\ z^\tau=\phi_Q^\pi(z^0|s,a,\tau)}} \left[\left(v_V(z^\tau \mid s,\tau) - u_Q^\pi(z^\tau \mid s,a,\tau)\right)^2\right]. \tag{9}$$

Learning the Q flow ϕ_Q^π. ϕ_Q^π in Eq. 8 is the marginal distribution over Gaussian random variables N_V, N and the next state S'. Similarly, we set $\mathcal{X}_Q = \{N_V, N, S'\}$ and define the stochastic process $X_Q^\pi(s,a,\tau)$ with noise-level differentiation as

$$\begin{aligned} X_Q^\pi(s,a,\tau) &\triangleq \tau r(s,a) + \gamma \phi_V^\pi(N_V \mid S',\tau) + (1-\tau)\sqrt{1-\gamma^2}N, \\ \frac{\mathrm{d}}{\mathrm{d}\tau}X_Q^\pi(s,a,\tau) &= r(s,a) + \gamma u_V^\pi(\phi_V^\pi(N_V \mid S',\tau) \mid S',\tau) - \sqrt{1-\gamma^2}N. \end{aligned}$$

Similarly, we can take expectations over $\mathcal{X}_Q$ and use a parametric state-action-value velocity $v_Q(z^\tau \mid s,a,\tau)$ to fit $u_Q^\pi(z^\tau \mid s,a,\tau)$ by minimizing

$$\begin{aligned} \mathbb{E}_{\substack{s'\sim p(s'|s,a),\ \tau\sim\textsc{Unif}([0,1]),\\ (z^0,\epsilon)\sim\mathcal{N}(0,1),\ z^\tau=\phi_V^\pi(z^0|s',\tau)}} \Bigg[\Bigg(v_Q\left(\tau r + \gamma z^\tau + (1-\tau)\sqrt{1-\gamma^2}\epsilon \,\middle|\, s,a,\tau\right) \\ - \left(r + \gamma u_V^\pi(z^\tau \mid s',\tau) - \sqrt{1-\gamma^2}\epsilon\right)\Bigg)^2\Bigg]. \end{aligned} \tag{10}$$

Practical objectives. Note that both losses we derived above (Eq. 9 and Eq. 10) require access to ground-truth velocities u_V^π and u_Q^π. Since the ground truth velocities are unknown, we learn parametric velocities v_V, v_Q using each other as targets. This procedure resembles Fitted Q-Evaluation (Riedmiller, 2005; Le et al., 2019) for learning the scalar V and Q functions. In addition, we in practice scale the reward function by $(1-\gamma)$ for numerical stability so that the discounted future return in an infinite horizon MDP is normalized to 1 (see Sec. 7.3 for empirical studies). This yields losses:

Practical objectives (Value Forcing)

$$\mathcal{L}(v_V) = \mathbb{E}_{\substack{a\sim\pi(a|s),\tau\sim\text{UNIF}([0,1]),\\ z^\tau\sim p_Q(z^\tau|s,a,\tau)}}\left[\left(v_V(z^\tau \mid s,\tau) - v_Q(z^\tau \mid s,a,\tau)\right)^2\right], \tag{11}$$

$$\mathcal{L}(v_Q) = \mathbb{E}_{\substack{s'\sim p(s'|s,a),\ \tau\sim\text{UNIF}([0,1]),\\ \epsilon\sim\mathcal{N}(0,1),\ z^\tau\sim p_V(z^\tau|s',\tau)}}\Big[\Big(v_Q\left(\tau(1-\gamma)r + \gamma z^\tau + (1-\tau)\sqrt{1-\gamma^2}\epsilon \,\Big|\, s,a,\tau\right) - \left((1-\gamma)r + \gamma v_V(z^\tau \mid s',\tau) - \sqrt{1-\gamma^2}\epsilon\right)\Big)^2\Big], \tag{12}$$

where $z^\tau \sim p_Q(z^\tau \mid s,a,\tau)$ and $z^\tau \sim p_V(z^\tau \mid s',\tau)$ denote the sampling of the noisy returns.

We alternate between minimizing $\mathcal{L}(v_V)$ and $\mathcal{L}(v_Q)$ iteratively, holding the other component fixed during each update. Minimizing these losses performs policy evaluation to find the return flows under policy π at each noise level. In the next section, we will discuss two strategies to improve the policy towards the optimal policy by explicit or implicit policy improvement.

6 Value Forcing Algorithms

We consider two ways to turn the richer distributional learning signals of Value Forcing into value-learning algorithms. First, we can explicitly evaluate a policy by learning its return distributions and then performing policy improvement. We refer to this approach as the *explicit* method, and introduce a practical algorithm named VF (Sec. 6). Second, motivated by prior work on implicit value learning (Kostrikov et al., 2022; Sikchi et al., 2024), we learn flow-based return distributions through an expectile-weighted flow-matching objective without an explicit actor. We refer to this type of approach as the *implicit* method, and introduce an implicit variant of our method called IVF (Implicit VF) (Sec. 6).

Explicit method. As in the standard actor-critic framework, the objective is to learn a policy that maximizes the Q estimate. In the context of return flows, Q-values are estimated as: $Q(s,a) \triangleq \mathbb{E}_{z^0\sim\mathcal{N}(0,1)}\left[\int_0^1 v_Q(z^\tau \mid s,a,\tau)d\tau\right]$. In practice, we compute this using M noise samples. We instantiate Q maximization as rejection sampling (Chen et al., 2023) on K actions proposed by a flow behavioral cloning (BC) policy $\pi_\beta(a \mid s)$:

$$\pi(a \mid s) \overset{d}{=} \underset{a_1,\cdots,a_K:\, a_i\sim\pi_\beta(a|s)}{\arg\max} Q(s,a_i). \tag{13}$$

We learn the BC policy using the standard flow-matching loss (Eq. 3) and denote the loss as $\mathcal{L}(\pi_\beta)$. The policy π is then used to sample actions for the objective in Eq. 11, thereby coupling policy evaluation with action selection under the updated critic.

Implicit method. We describe below another instantiation of our method, IVF, that avoid explicit actors and only queries actions within the dataset during training. We instantiate this idea using a weighted regression loss. Following prior work (e.g., Sec. 10.3.1 of Fahrmeir et al. (2022)), we define the expectile weight with expectile $\mu \in [0.5, 1)$ as

$$w^\mu(s,a) \triangleq \mu\mathbb{1}\left(Q(s,a) \geq V(s)\right) + (1-\mu)\mathbb{1}(Q(s,a) < V(s)), \tag{14}$$

where $V(s) \triangleq \mathbb{E}_{z^0\sim\mathcal{N}(0,1)}\left[\int_0^1 v_V(z^\tau \mid s,\tau)d\tau\right]$. The resulting expectile-weighted regression objective is

$$\mathcal{L}^\mu(v_V) = \mathbb{E}_{\substack{a\sim D,\ \tau\sim\text{UNIF}([0,1]),\\ z^\tau\sim p_Q(z^\tau|s,a,\tau)}}\left[w^\mu(s,a)\left(v_V(z^\tau \mid s,\tau) - v_Q(z^\tau \mid s,a,\tau)\right)^2\right]. \tag{15}$$

This loss extends the scalar expectile regression loss (Kostrikov et al., 2022) to flow-based critics. We show in Appendix A.2 that *(1)* minimizing the expectile-weighted loss induces a V estimate that is the expectile of the Q estimate, and *(2)*, when the expectile $\mu \to 1$, the value estimate converges to the optimal value within the dataset support.

Implementation Details To summarize, VF and IVF consist of a state-value velocity $v_V(z^\tau \mid s,\tau)$, a state-action-value velocity $v_Q(z^\tau \mid s,a,\tau)$, and a flow BC policy $\pi_\beta(a \mid s)$. We use neural networks

Algorithm 1 Value Forcing (VF) and Implicit Value Forcing (IVF)

Input: Dataset D, discount γ, expectile μ, number of action candidates K, number of MC return samples M.

Initialize velocities $v_V^\theta, v_Q^\psi, v_\beta^\xi$ and initialize target velocity $\bar{\psi} \leftarrow \psi$.
for each training iteration **do**
  $(s, a, r, s') \sim D$, $\tau \sim \text{Unif}([0, 1])$.
  $a_\pi \sim$ rejection sampling (Eq. 13), $z^\tau \sim p_Q^{\bar{\psi}}(z^\tau \mid s, a_\pi, \tau)$.
  $\Delta\theta \leftarrow \nabla_\theta \mathcal{L}(v_V^\theta)$ (Eq. 11). ◁ Learn velocity for VF
  Compute $w^\mu(s, a)$ using Eq. 14, $z^\tau \sim p_Q^{\bar{\psi}}(z^\tau \mid s, a, \tau)$.
  $\Delta\theta \leftarrow \nabla_\theta \mathcal{L}^\mu(v_V^\theta)$ (Eq. 15). ◁ Learn velocity for IVF
  $z^\tau \sim p_V^\theta(z^\tau \mid s', \tau)$, $\epsilon \sim \mathcal{N}(0, 1)$.
  $\Delta\psi \leftarrow \nabla_\psi \mathcal{L}(v_Q^\psi)$ (Eq. 12). ◁ Learn state-action-value velocity
  $\Delta\xi \leftarrow \nabla_\xi \mathcal{L}(v_\beta^\xi)$ (Eq. 3). ◁ Learn flow BC policy
  $\bar{\psi} \leftarrow$ the exponential moving average of ψ. ◁ Update the target parameters
end for

to parameterize the state-value velocity v_V^θ, state-action-value velocity v_Q^ψ, and flow BC velocity v_β^ξ. The scalar estimates $V(s)$ and $Q(s, a)$ are obtained by averaging M return samples generated by their return flows. At evaluation time, both algorithms select actions using rejection sampling over K candidates from π_β. VF also uses this action selection procedure to learn the state-value velocity, while IVF uses actions in the dataset to perform expectile-weighted regression. Alg. 1 summarizes the training procedures of VF and IVF.

7 Experiments

We conduct experiments to study how well Value Forcing can learn the return distribution on a range of challenging long-horizon tasks. Concretely, we ask the research questions: **(1)** Does Value Forcing improve return estimation accuracy? **(2)** How do VF and IVF compare to prior offline RL methods across different dataset sizes? **(3)** What are the important design decisions in VF and IVF?

Experimental setup. We consider 4 different domains with 5 tasks each from OGBench (Park et al., 2024a) in the offline RL setting: `cube-triple`, `cube-quadruple`, `puzzle-4x4`, and `humanoidmaze-giant`. We adopt the action chunking setting (Li et al., 2025), where the policy learns to predict a sequence of actions and execute the full action chunk before predicting another. In this higher-dimensional action space, value learning is also harder compared to using atomic actions, making this setting a challenging testbed for value learning. Following Zhou et al. (2026), we use the dataset with 100M transitions as the full offline dataset, but also experiment with subsets of it, ranging from 10% to 100%, to study the effectiveness of different value learning algorithms on decreasing dataset sizes. See details on dataset construction and evaluation protocols in Appendix D.1.

Comparisons. We compare VF and IVF with prior value learning methods, grouped into explicit or implicit methods. For *explicit methods*, compare against prior flow-based distributional critics **FlowCritic** (Zhong et al., 2025), **Value Flows** (Dong et al., 2025), **PCBF** (Xu et al., 2026), prior categorical distributional critics **IQN** (Dabney et al., 2018a) and **C51** (Bellemare et al., 2017), prior flow-based scalar critic **Floq** (Agrawalla et al., 2026), and scalar critic **QC**. For *implicit methods*, there are no prior flow-based distributional critics, so we compare against categorical distributional critics **DIVL** (Wang et al., 2026), our novel adaptations of C51 and IQN termed **Implicit C51** and **Implicit IQN**, and scalar critic **BFN**. See Appendix D.2 for details.

7.1 Value Forcing improves return estimation accuracy

We investigate whether Value Forcing improves return distribution prediction by conducting *policy evaluation* experiments comparing VF with Value Flows, FlowCritic, and PCBF at a single state randomly sampled from each of the four OGBench domains. Our goal is to compare the return estimation accuracy starting from a single state in each domain and then following a fixed policy. Specifically, from each state, we execute a fixed starting action chunk and then roll out a data-collection policy to collect 1000 trajectories, whose discounted returns form the ground-truth return distribution. We train every method on these trajectories, using the recorded actions in the bootstrap

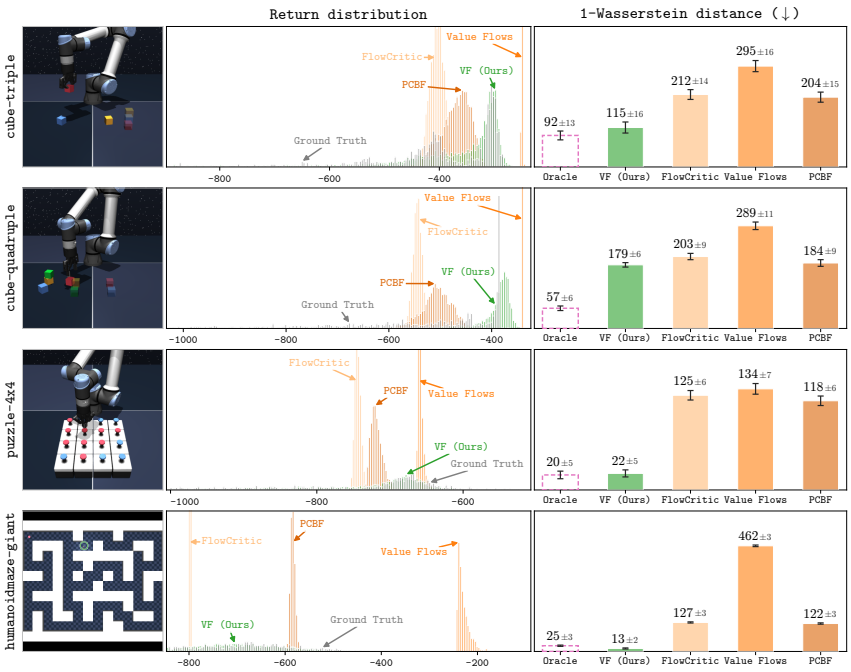


Figure 2: **Value Forcing learns more accurate return distributions than prior flow-based return estimators.** *(Left)* Value Forcing captures the ground-truth return distributions more accurately, while prior methods are biased and collapse their predictions into narrow peaks. *(Right)* Value Forcing achieves lower mean 1-Wasserstein distances than all three prior methods in all tasks.

targets, effectively performing SARSA (Sutton et al., 1998). Thus, all methods evaluate the same data-collection policy. Following Dong et al. (2025), we visualize the predicted return distributions and measure their 1-Wasserstein distances to the ground-truth returns. As a reference, an `Oracle` return estimator fits the realized returns of the same trajectories using the standard flow-matching loss (Eq. 3). This provides a reference for errors other than bootstrapping: function approximation, optimization, finite samples, etc. See Appendix D.4 for the full experimental details.

Fig. 2 shows that VF achieves lower mean 1-Wasserstein distances than Value Flows, FlowCritic, and PCBF on all four tasks. Observe that VF captures more of the ground-truth spread, while Value Flows and FlowCritic concentrate their predictions into narrow peaks. PCBF produces broader predictions on the `cube` tasks, but still misses much of the ground-truth spread. Relative to PCBF, VF reduces the mean 1-Wasserstein distance by 81% on `puzzle-4x4` and 89% on `humanoidmaze-giant`. On `cube-triple` and `puzzle-4x4`, VF's 1-Wasserstein distances are also close to the `Oracle`. In particular, VF corrects Value Flows' inconsistent supervision at $\tau = 0$, offering an explanation for the improvement over its narrow, shifted predictions. In Appendix Fig. 6, we visualize the return flow paths at different noise levels, showing that VF spreads its return paths at early noise levels (around $\tau = 0.25$), while baselines collapse (around $\tau = 0.9$). The gains over FlowCritic and PCBF further suggest the benefit of enforcing Bellman consistency of noisy return distributions at every noise level.

7.2 VF and IVF outperform prior offline RL methods on smaller datasets

We next compare VF and IVF against prior value-learning algorithms under varying dataset sizes. Specifically, we train each method using the full dataset as well as subsets containing 50%, 25%, and 10% of the available transitions. We subsample the dataset by using a softmax over trajectories, scored by the average reward over all time steps (Appendix D.1). Fig. 3 summarizes the success rates over all 4 OGBench domains, grouping the methods into explicit and implicit methods. Among explicit methods, VF achieves the highest mean success rates with 10% and 25% of the data and remains competitive at larger dataset sizes. Most prior flow-based distributional RL methods underperform the scalar baseline QC, particularly with 10% of the data. In contrast, VF maintains relatively stable performance as the dataset size decreases. Among implicit methods, IVF performs comparably to the strongest baselines on the full dataset and achieves higher mean success rates than all prior methods at every reduced dataset size. The advantage is particularly pronounced with 25% and 10% of the data, where IVF achieves approximately 33% and 54% higher success rates, respectively, than the

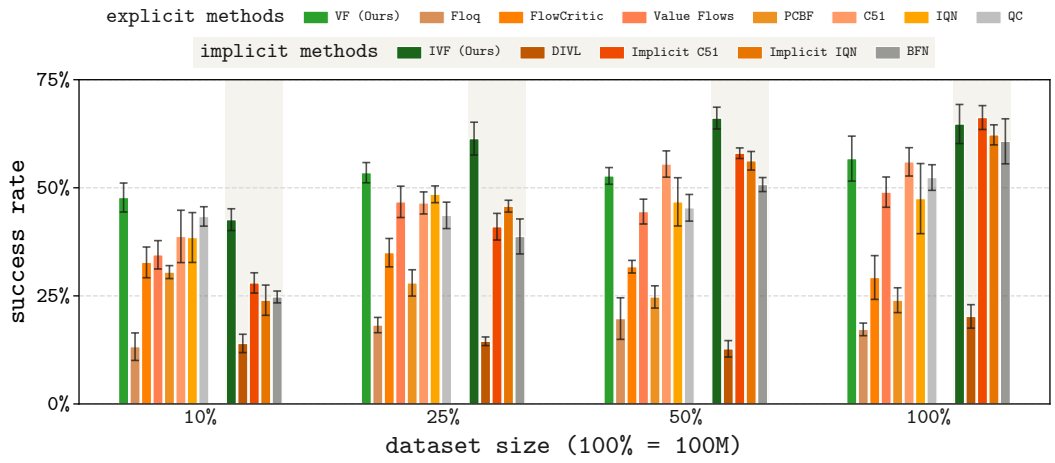


Figure 3: **Value Forcing improves offline RL performance with limited data.** VF and IVF significantly outperform their respective explicit and implicit baselines at 10% and 25% data, while remaining competitive at larger dataset sizes. Shaded groups indicate implicit methods. See Appendix C for the complete results.

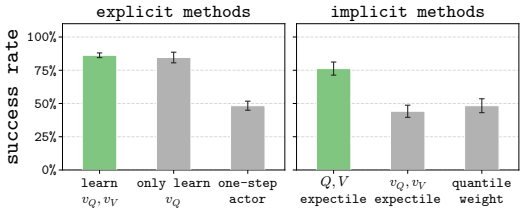


Figure 4: Alternative Value Forcing instantiations for explicit methods (Left) and implicit methods (Right).

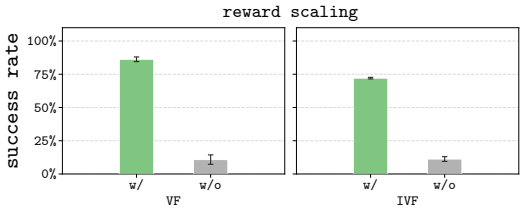


Figure 5: **Reward scaling improves VF and IVF.** Both VF and IVF achieve substantially higher success rates with reward scaling (`w/`) than without it (`w/o`).

strongest implicit methods at each dataset size. These results suggest that aligning return distributions at all noise levels is particularly beneficial when offline data are limited.

7.3 Key Design Decisions in Value Forcing

Finally, we study key design decision for VF and IVF, ablate them on tasks 2 and 4 from `cube-triple` and `puzzle-4x4`. See Appendix D.5 for details on our ablation designs below.

For VF, we ablate two alternative designs: removing the separate state-value velocity in Eq. 12 and instead using state-action-value velocity to bootstrap itself (v_Q `only`), and replacing rejection sampling with a learned one-step actor that maximizes the Q estimate (Park et al., 2025b) (`one-step actor`). Fig. 4 *(Left)* shows that only learning v_Q yields similar success rates to VF, whereas using a one-step actor reduces success rates by 44%. These results suggest that VF accommodates alternative state-action-value velocity updates, while using rejection sampling is important for VF.

For IVF, we examine alternatives to the expectile weighting in the value velocity loss (Eq. 15). We compare weighting based on scalar advantages (Q, V `expectile`) against applying expectile regression directly to velocity residuals (v_Q, v_V `expectile`) or weighting based on return quantiles (`quantile weight`). Fig. 4 *(Right)* shows that weighting based on scalar advantages achieves much higher success rates than its two counterparts. These results support using scalar advantages to determine the expectile weights when learning return velocities.

Further, Fig. 5 examines the effect of reward scaling. Removing scaling reduces mean success rates from approximately 86% to 11% for VF and from 72% to 11% for IVF, highlighting the importance of reward scaling in the practical implementation of both methods.

8 Conclusion

In this paper, we propose Value Forcing, a new approach to learning distributional value functions with flow matching, with Bellman consistency at every noise level. This provides substantially richer supervision during training and enables much more accurate estimation of the return distribution and better policy performance, especially in the limited-data regime.

LIMITATIONS

Our theoretical analysis of IVF assumes exact optimization and convergence to a self-consistent fixed point. These results do not establish convergence of the practical algorithms with neural network approximation. Empirically, our evaluation is limited to simulated offline RL tasks. Extending Value Forcing to online learning and real-world applications remains future work.

AI USE STATEMENT

In this work, we used generative AI tools for polishing the writing and implementing the methods. We have not used generative AI tools for designing or providing feedback on research methodology or experiments, proposing or refining hypotheses, assisting with translation. Supporting qualitative and thematic data analysis and interpreting the results, generating synthetic data sets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, cleaning and reformatting datasets do not apply to this work. The authors have manually reviewed the paper to prevent plagiarism, and read the LLM-generated code and verified and tested its correctness. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We include implementation details needed for reproduction, such as hyperparameters (including the baselines), datasets, and evaluation protocols in Appendix D. Clear explanations and the proofs of the theoretical results are provided in Appendix A.

REFERENCES

Bhavya Agrawalla, Michal Nauman, Khush Agrawal, and Aviral Kumar. floq: Training critics via flow-matching for scaling compute in value-based rl. In *International Conference on Learning Representations*, volume 2026, pp. 54569–54595, 2026.

Michael Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research*, 26(209):1–80, 2025.

Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *International Conference on Learning Representations (ICLR)*, 2023.

Chenjia Bai, Lingxiao Wang, Zhuoran Yang, Zhihong Deng, Animesh Garg, Peng Liu, and Zhaoran Wang. Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning. *arXiv preprint arXiv:2202.11566*, 2022.

Stefan Banach. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta mathematicae*, 3(1):133–181, 1922.

Vansh Bansal, Saptarshi Roy, Alessandro Rinaldo, and Purnamrita Sarkar. On the convergence and straightness of rectified flow. In *The 29th International Conference on Artificial Intelligence and Statistics*, 2026. URL https://openreview.net/forum?id=qKKxzTucgy.

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.

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

Qi Cai, Zhuoran Yang, Jason D. Lee, and Zhaoran Wang. Neural temporal-difference and q-learning provably converge to global optima, 2020. URL https://arxiv.org/abs/1905.10027.

Jongseong Chae, Jongeui Park, Yongjae Shin, Gyeongmin Kim, Seungyul Han, and Youngchul Sung. Flow actor-critic for offline reinforcement learning. *arXiv preprint arXiv:2602.18015*, 2026.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2024.

Deshu Chen, Yuchen Liu, Zhijian Zhou, Chao Qu, and Yuan Qi. Unleashing flow policies with distributional critics. *arXiv preprint arXiv:2509.23087*, 2025.

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=42zs3qa2kpy.

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

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, 2018b.

Perry Dong, Chongyi Zheng, Chelsea Finn, Dorsa Sadigh, and Benjamin Eysenbach. Value flows. *arXiv preprint arXiv:2510.07650*, 2025.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6, 2005.

Nicolas Espinosa-Dice, Kiante Brantley, and Wen Sun. Expressive value learning for scalable offline reinforcement learning. *arXiv preprint arXiv:2510.08218*, 2025.

Ludwig Fahrmeir, Thomas Kneib, Stefan Lang, and Brian D Marx. Regression models. In *Regression: Models, methods and applications*, pp. 23–84. Springer, 2022.

Jesse Farebrother, Matteo Pirotta, Andrea Tirinzoni, Remi Munos, Alessandro Lazaric, and Ahmed Touati. Temporal difference flows. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=j6H7c3aQyb.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

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.

Mudit Gaur, Vaneet Aggarwal, and Mridul Agarwal. On the global convergence of fitted q-iteration with two-layer neural network parametrization. In *International Conference on Machine Learning*, pp. 11013–11049. PMLR, 2023.

Michael Groom, Victor-Alexandru Darvariu, Lars Kunze, James Wilson, and Nick Hawes. Quantile-coupled flow matching for distributional reinforcement learning. *arXiv preprint arXiv:2605.08515*, 2026.

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*, pp. 1861–1870. PMLR, 2018.

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.

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

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Ehsan Imani and Martha White. Improving regression performance with distributional losses. In *International conference on machine learning*, pp. 2157–2166. PMLR, 2018.

JAX Developers. The autodiff cookbook. https://docs.jax.dev/en/latest/notebooks/autodiff_cookbook.html, 2025. URL https://docs.jax.dev/en/latest/notebooks/autodiff_cookbook.html.

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

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=68n2s9ZJWF8.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in neural information processing systems*, 32, 2019.

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.

Hoang Le, Cameron Voloshin, and Yisong Yue. Batch policy learning under constraints. In *International Conference on Machine Learning*, pp. 3703–3712. PMLR, 2019.

John M Lee. *Introduction to Smooth Manifolds*. Springer, 2012.

Qijun Li, Zheng Fu, Qi Song, Yifei He, Weitao Zhou, Kun Jiang, and Diange Yang. Dual-flow reinforcement learning with state-aware exploration. *arXiv preprint arXiv:2606.29820*, 2026.

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.

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.

Xiaoteng Ma, Junyao Chen, Li Xia, Jun Yang, Qianchuan Zhao, and Zhengyuan Zhou. Dsac: Distributional soft actor-critic for risk-sensitive reinforcement learning. *Journal of Artificial Intelligence Research*, 83, 2025.

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, 2024.

Jongchan Park, Mingyu Park, and Donghwan Lee. Pretraining a shared q-network for data-efficient offline reinforcement learning. *Advances in Neural Information Processing Systems*, 38:130527–130555, 2026.

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

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

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.

Martin Riedmiller. Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method. In *European conference on machine learning*, pp. 317–328. Springer, 2005.

Mark Rowland, Yunhao Tang, Clare Lyle, Rémi Munos, Marc G Bellemare, and Will Dabney. The statistical benefits of quantile temporal-difference learning for value estimation. In *International Conference on Machine Learning*, pp. 29210–29231. PMLR, 2023.

Harshit Sikchi, Qinqing Zheng, Amy Zhang, and Scott Niekum. Dual rl: Unification and new methods for reinforcement and imitation learning. In *International Conference on Learning Representations*, volume 2024, pp. 9305–9352, 2024.

Anikait Singh, Aviral Kumar, Quan Vuong, Yevgen Chebotar, and Sergey Levine. Offline rl with realistic datasets: Heteroskedasticity and support constraints. *arXiv preprint arXiv:2211.01052*, 2022.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Cédric Villani. *Optimal Transport: Old and New*, volume 338 of *Grundlehren der mathematischen Wissenschaften*. Springer, 2009. doi: 10.1007/978-3-540-71050-9.

Kaiwen Wang, Kevin Zhou, Runzhe Wu, Nathan Kallus, and Wen Sun. The benefits of being distributional: Small-loss bounds for reinforcement learning. *Advances in neural information processing systems*, 36:2275–2312, 2023.

Yi Wang, Xinchen Li, Pengwei Xie, Pu Yang, Buqing Nie, Yunuo Cai, Qinglin Zhang, Chendi Qu, Jeffrey Wu, Jianheng Song, et al. Learning while deploying: Fleet-scale reinforcement learning for generalist robot policies. *arXiv preprint arXiv:2605.00416*, 2026.

Boyang Xu, Qing Zou, Siqin Yang, and Hao Yan. Path-coupled bellman flows for distributional reinforcement learning. *arXiv preprint arXiv:2605.08253*, 2026.

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.

Shan Zhong, Shutong Ding, He Diao, Xiangyu Wang, Kah Chan Teh, and Bei Peng. Flow-critic: Bridging value estimation with flow matching in reinforcement learning. *arXiv preprint arXiv:2510.22686*, 2025.

Zhiyuan Zhou, Andy Peng, Charles Xu, Qiyang Li, Tobias Springenberg, Kevin Frans, and Sergey Levine. Test-time gradient guidance of flow policies in reinforcement learning. *arXiv preprint arXiv:2606.11087*, 2026.

A Proofs and Derivations

A.1 Contraction of the flow Bellman operator

Proposition 1. *Given a policy π, a discount factor $\gamma \in [0,1)$, a noise level $\tau \in [0,1]$, and $p \in [1,\infty)$, for two arbitrary members of the return random variables $Z_1 \in \mathcal{Z}$ and $Z_2 \in \mathcal{Z}$, we define the supremum of the p-Wasserstein distance over all state-action pairs as*

$$W_p^{sup}(Z_1, Z_2) \triangleq \sup_{s,a} W_p\left(Z_1(s,a,\tau), Z_2(s,a,\tau)\right),$$

where $W_p(Z_1, Z_2)$ is the p-Wasserstein distance between the corresponding probability laws of random variables Z_1 and Z_2 (Villani, 2009). Then, the flow Bellman operator $\mathcal{T}_\tau^\pi$ satisfies

$$W_p^{sup}\left(\mathcal{T}_\tau^\pi Z_1, \mathcal{T}_\tau^\pi Z_2\right) \leq \gamma W_p^{sup}(Z_1, Z_2).$$

Therefore, the flow Bellman operator $\mathcal{T}_\tau^\pi$ is a γ-contraction under the metric W_p^{sup}.

Proof. Given a state-action pair (s,a), let $S' \sim p(S' \mid s,a)$ be the random variable of the next state and $A' \sim \pi(A' \mid S')$ be the random variable of the next action. Our goal is to find a coupling between random variables $Z_1(S',A',\tau)$ and $Z_2(S',A',\tau)$ and a coupling between random variables $\mathcal{T}_\tau^\pi Z_1(s,a,\tau)$ and $\mathcal{T}_\tau^\pi Z_2(s,a,\tau)$ to relate the p-Wasserstein distance $W_p^{\sup}(\mathcal{T}_\tau^\pi Z_1, \mathcal{T}_\tau^\pi Z_2)$ with the p-Wasserstein distance $W_p^{\sup}(Z_1, Z_2)$.

Let $F_{Z_1}^{-1}(z \mid S',A',\tau)$ and $F_{Z_2}^{-1}(z \mid S',A',\tau)$ denote the quantile function (inverse cumulative distribution function) of the random variable $Z_1(S',A',\tau)$ and $Z_2(S',A',\tau)$, respectively. Let

$(C_1(S', A', \tau), C_2(S', A', \tau)) \in \mathcal{Z} \times \mathcal{Z}$ be a pair of return random variables. We define (C_1, C_2) by transforming a uniform random variable $U \sim \text{UNIF}([0, 1])$ using the quantile functions $F^{-1}_{Z_1}$ and $F^{-1}_{Z_2}$ at the next state S' and the next action A':

$$(C_1(S', A', \tau),\ C_2(S', A', \tau)) \triangleq \left(F^{-1}_{Z_1}(U \mid S', A', \tau),\ F^{-1}_{Z_2}(U \mid S', A', \tau)\right),\ U \sim \text{UNIF}([0, 1]),$$

where U is independent of (S', A'). Importantly, the random variables C_1 and C_2 are generally dependent on each other because they depend on the same set of random variables (S', A', U). By definition, we can show that $(C_1(S', A', \tau), C_2(S', A', \tau))$ is a coupling between $Z_1(S', A', \tau)$ and $Z_2(S', A', \tau)$ by computing their CDFs: for any $y \in \mathbb{R}$,

$$\begin{aligned} F_{C_1}(y \mid S', A', \tau) &\overset{(a)}{=} \mathbb{P}\left(F^{-1}_{Z_1}(U \mid S', A', \tau) \le y\right) \\ &\overset{(b)}{=} \mathbb{P}\left(U \le F_{Z_1}(y \mid S', A', \tau)\right) \\ &\overset{(c)}{=} F_{Z_1}(y \mid S', A', \tau), \end{aligned}$$

where *(a)* uses the definition of the conditional CDF, *(b)* applies the CDF F_{Z_1} on both sides of the inequality, and *(c)* uses the CDF of the uniform random variable. Therefore, the random variable C_1 has the same law as Z_1: $C_1(S', A', \tau) \overset{d}{=} Z_1(S', A', \tau)$. Similarly, we can show that $C_2(S', A', \tau) \overset{d}{=} Z_2(S', A', \tau)$. Together, these two equalities in distribution show that the joint distribution $(C_1(S', A', \tau), C_2(S', A', \tau))$ is a coupling between $Z_1(S', A', \tau)$ and $Z_2(S', A', \tau)$. This coupling recovers the one-dimensional quantile formula for the p-Wasserstein distance:

$$\begin{aligned} W_p(Z_1(S', A', \tau), Z_2(S', A', \tau)) &= \left(\int_0^1 |F^{-1}_{Z_1}(u \mid S', A', \tau) - F^{-1}_{Z_2}(u \mid S', A', \tau)|^p du\right)^{1/p} \\ &= \mathbb{E}_U\left[|F^{-1}_{Z_1}(U \mid S', A', \tau) - F^{-1}_{Z_2}(U \mid S', A', \tau)|^p\right]^{1/p} \\ &\overset{(a)}{=} \mathbb{E}_U\left[|C_1(S', A', \tau) - C_2(S', A', \tau)|^p\right]^{1/p}, \end{aligned} \tag{16}$$

where *(a)* plugs in the definition of random variables C_1 and C_2.

We next define a joint distribution that couples the random variables $\mathcal{T}^\pi_\tau Z_1(s, a, \tau)$ and $\mathcal{T}^\pi_\tau Z_2(s, a, \tau)$. Specifically, let $(T_1(s, a, \tau),\ T_2(s, a, \tau)) \in \mathcal{Z} \times \mathcal{Z}$ be another pair of return random variables. We define (T_1, T_2) as transforming the random variables C_1 and C_2 using the flow Bellman equation (Eq. 4) with a Gaussian correction $N \sim \mathcal{N}(0, 1)$ that is independent of (S', A', U):

$$\begin{aligned} (T_1(s, a, \tau),\ T_2(s, a, \tau)) \triangleq \Big(&\tau r(s, a) + \gamma C_1(S', A', \tau) + (1 - \tau)\sqrt{1 - \gamma^2}N, \\ &\tau r(s, a) + \gamma C_2(S', A', \tau) + (1 - \tau)\sqrt{1 - \gamma^2}N\Big), \\ S' \sim p(S' \mid s, a),\ A' \sim \pi(A' \mid S'),\ &U \sim \text{UNIF}([0, 1]),\ N \sim \mathcal{N}(0, 1). \end{aligned}$$

Here, the random variables T_1 and T_2 are also generally dependent on each other because they share the same set of random variables (S', A', U, N). Since $(C_1(S', A', \tau), C_2(S', A', \tau))$ is a coupling between $Z_1(S', A', \tau)$ and $Z_2(S', A', \tau)$, we have

$$\begin{aligned} T_1(s, a, \tau) &\overset{d}{=} \tau r(s, a) + \gamma C_1(S', A', \tau) + (1 - \tau)\sqrt{1 - \gamma^2}N \\ &\overset{d}{=} \tau r(s, a) + \gamma Z_1(S', A', \tau) + (1 - \tau)\sqrt{1 - \gamma^2}N \\ &\overset{d}{=} \mathcal{T}^\pi_\tau Z_1(s, a, \tau), \end{aligned}$$

which shows that T_1 has the same law as $\mathcal{T}^\pi_\tau Z_1$. Similarly, we can show that $T_2(s, a, \tau) \overset{d}{=} \mathcal{T}^\pi_\tau Z_2(s, a, \tau)$. Therefore, these two equalities in distribution suggest that the joint distribution $(T_1(s, a, \tau),\ T_2(s, a, \tau))$ is a coupling between random variables $\mathcal{T}^\pi_\tau Z_1(s, a, \tau)$ and $\mathcal{T}^\pi_\tau Z_2(s, a, \tau)$.

Finally, we can show the desired inequality for the supremum of the p-Wasserstein distance over all state-action pairs. We consider the p-Wasserstein distance between $\mathcal{T}_\tau^\pi Z_1(s,a,\tau)$ and $\mathcal{T}_\tau^\pi(s,a,\tau)$:

$$\begin{aligned} W_p\left(\mathcal{T}_\tau^\pi Z_1(s,a,\tau), \mathcal{T}_\tau^\pi Z_2(s,a,\tau)\right) &\overset{(a)}{\leq} \mathbb{E}_{S',\,A',\,U,\,N}\left[|T_1(s,a,\tau) - T_2(s,a,\tau)|^p\right]^{1/p} \\ &\overset{(b)}{=} \gamma \mathbb{E}_{S',\,A',\,U}\left[|C_1(S',A',\tau) - C_2(S',A',\tau)|^p\right]^{1/p} \\ &\overset{(c)}{=} \gamma \mathbb{E}_{S',A'}\left[\left(W_p(Z_1(S',A',\tau), Z_2(S',A',\tau))\right)^p\right]^{1/p} \\ &\overset{(d)}{\leq} \gamma \sup_{s',a'} W_p(Z_1(s',a',\tau), Z_2(s',a',\tau)) \\ &\overset{(e)}{=} \gamma W_p^{\sup}(Z_1, Z_2), \end{aligned}$$

where *(a)* uses the definition of the p-Wasserstein distance (the infimum of the p-norm over all couplings), *(b)* plugs in the definitions of T_1 and T_2, *(c)* applies the law of total expectation and Eq. 16, *(d)* uses the supremum over the state-action pairs to bound the expectation over the next state and the next action, and *(e)* applies the definition of the supremum of the p-Wasserstein distance. Because this inequality holds for every state-action pair (s,a), we can take the supremum over all state-action pairs to conclude that

$$W_p^{\sup}\left(\mathcal{T}_\tau^\pi Z_1, \mathcal{T}_\tau^\pi Z_2\right) \leq \gamma W_p^{\sup}(Z_1, Z_2).$$

□

A.2 Theoretical analyses of IVF

We conduct theoretical analyses to examine the behavior of the expectile-weighted regression objective in Eq. 15. Our analyses answer two questions:

Q1. Does the state-value velocity that minimizes the expectile-weighted loss induce a scalar value that is the expectile of the scalar critic?

Q2. Does increasing the expectile recover the optimal value within the support of the dataset?

We proceed by first discussing the setup of optimization and assumptions about the fixed point of the expectile-weighted regression objective. We then show that minimizing the expectile-weighted loss $\mathcal{L}^\mu$ fits the return velocities of an implicit policy (Hansen-Estruch et al., 2023; Sikchi et al., 2024). Moreover, at the fixed point, the scale value and the scale critic induced by the return velocities satisfy the expectile Bellman equations as in Kostrikov et al. (2022). Finally, we show that as the expectile μ approaches 1, the induced scale value converges to the optimal value function within the support of the dataset, as in Theorem 3 of Kostrikov et al. (2022).

Optimaization setup. We consider an optimization procedure similar to Fitted Q iteration (Ernst et al., 2005): at each iteration k, we learn the state-value velocity v_V^k and the state-action-value velocity v_Q^k using the state-value velocity v_V^{k-1} and the state-action-value velocity v_Q^{k-1} from the previous iteration. Here, $k = 1, 2, \cdots$ indexes the iterations, with $k = 0$ denoting random initialization. We consider learning v_V^k *until convergence* and then increase the number of iterations. In practice, we perform one gradient update at each iteration, which preserves a similar global convergence guarantee (Gaur et al., 2023; Cai et al., 2020).

Under this setup, we can write the expectile-weighted objective at iteration k as

$$\mathcal{L}_k^\mu(v_V) = \mathbb{E}_{\substack{(s,\,a)\sim D,\,\tau\sim\text{Unif}([0,1]),\\ z^\tau\sim p_Q^{k-1}(z^\tau|s,a,\tau)}}\left[w_{k-1}^\mu(s,a)(v_V(z^\tau \mid s,\tau) - v_Q^{k-1}(z^\tau \mid s,a,\tau))^2\right],$$

where v_Q^{k-1} is the state-action-value velocity from the previous iteration $k-1$, which induces the probability density path p_Q^{k-1}, and the expectile weight w_{k-1}^μ is defined using the return velocities

from the iteration $k-1$ as

$$Q^{k-1}(s,a) \triangleq \mathbb{E}_{z^0\sim\mathcal{N}(0,1)}\left[\int_0^1 v_Q^{k-1}(z^\tau \mid s,a,\tau)d\tau\right], \tag{17}$$

$$V^{k-1}(k) \triangleq \mathbb{E}_{z^0\sim\mathcal{N}(0,1)}\left[\int_0^1 v_V^{k-1}(z^\tau \mid s,\tau)d\tau\right], \tag{18}$$

$$w_{k-1}^{\mu}(s,a) \triangleq \mu\mathbb{1}\left(Q^{k-1}(s,a) \geq V^{k-1}(s)\right) + (1-\mu)\mathbb{1}(Q^{k-1}(s,a) < V^{k-1}(s)). \tag{19}$$

Similarly, the objective of the state-action-value velocity at iteration k is

$$\mathcal{L}_k(v_Q) = \mathbb{E}_{\substack{(s,a)\sim D,\ s'\sim p(s'|s,a),\\ \tau\sim\text{UNIF}([0,1]),\\ \epsilon\sim\mathcal{N}(0,1),\ z^\tau\sim p_V^{k-1}(z^\tau|s',\tau)}}\left[\left(v_Q\left(\tau(1-\gamma)r + \gamma z^\tau + (1-\tau)\sqrt{1-\gamma^2}\epsilon \,\middle|\, s,a,\tau\right)\right.\right.$$
$$\left.\left. - \left((1-\gamma)r + \gamma v_V^{k-1}(z^\tau \mid s',\tau) - \sqrt{1-\gamma^2}\epsilon\right)\right)^2\right], \tag{20}$$

where v_V^{k-1} is the state-value velocity from the previous iteration $k-1$, which induces the probability density path p_V^{k-1}. At each iteration k, we find the state-value velocity v_V^k by minimizing the expectile-weighted loss $\mathcal{L}_k^\mu(v_V)$ until convergence and find the state-action-value velocity v_Q^k by minimizing the state-action-value velocity loss $\mathcal{L}_k(v_Q)$ until convergence.

Assumption. We assume that the iterative optimization procedure above will converge to a fixed point when $k \to \infty$. At such a fixed point, the state-value velocity v_V^∞ minimizes the expectile-weighted loss with the expectile weight defined by itself (self-consistent):

$$v_V^\infty = \arg\min_{v_V} \mathcal{L}_\infty^\mu(v_V),$$
$$\mathcal{L}_\infty^\mu(v_V^\infty) = \mathbb{E}_{\substack{(s,a)\sim D,\ \tau\sim\text{UNIF}([0,1]),\\ z^\tau\sim p_Q^\infty(z^\tau|s,a,\tau)}}\left[w_\infty^\mu(s,a)(v_V^\infty(z^\tau \mid s,\tau) - v_Q^\infty(z^\tau \mid s,a,\tau))^2\right] = 0, \tag{21}$$

where $w_\infty^\mu(s,a) = \mu\mathbb{1}\left(Q^\infty(s,a) \geq V^\infty(s)\right) + (1-\mu)\mathbb{1}(Q^\infty(s,a) < V^\infty(s))$. Similarly, the state-action-value velocity v_Q^∞ minimizes the state-action-value velocity loss:

$$v_Q^\infty = \arg\min_{v_Q} \mathcal{L}_\infty^\mu(v_Q),$$
$$\mathcal{L}_\infty^\mu(v_Q^\infty) = \mathbb{E}_{\substack{(s,a,r)\sim D,\ s'\sim p(s'|s,a),\\ \tau\sim\text{UNIF}([0,1]),\\ \epsilon\sim\mathcal{N}(0,1),\ z^\tau\sim p_V^\infty(z^\tau|s',\tau)}}\left[\left(v_Q\left(\tau(1-\gamma)r + \gamma z^\tau + (1-\tau)\sqrt{1-\gamma^2}\epsilon \,\middle|\, s,a,\tau\right)\right.\right.$$
$$\left.\left. - \left((1-\gamma)r + \gamma v_V^\infty(z^\tau \mid s',\tau) - \sqrt{1-\gamma^2}\epsilon\right)\right)^2\right] = 0, \tag{22}$$

Under these setup and assumption, we can answer the aforementioned questions.

A1: Yes, the fixed point of optimizing the expectile-weighted loss induces an expectile value.

We first find the minimizer v_V^k of the bootstrapped flow-matching loss $\mathcal{L}_k^\mu(v_V)$ and the minimizer v_Q^k of the state-action-value velocity loss $\mathcal{L}_k(v_Q)$ by writing the losses in the forms as in Lemma 1. The main idea is to define stochastic processes similar to those in Sec. 5 and Sec. 5 and employ Lemma 1. Finding the fixed point of the bootstrapped flow-matching loss and the state-action-value velocity loss helps us recover the flow Bellman operator in Eq. 6 at convergence. In particular, this flow Bellman equation corresponds to an implicit policy, and we then use the flow Bellman equation to show that the induced scalar value and scalar critic satisfy the expectile Bellman equations as in (Kostrikov et al., 2022).

We start by defining an *implicit* policy (Hansen-Estruch et al., 2023) at each iteration:

$$\pi_{k-1}^\mu(a \mid s) \triangleq \frac{\pi_\beta(a \mid s)w_{k-1}^\mu(s,a)}{\mathbb{E}_{\pi_\beta(a|s)}[w_{k-1}^\mu(s,a)]}, \tag{23}$$

where $\pi_\beta(a \mid s)$ is the behavioral policy that collects the actions in the dataset. Since $w^\mu_{k-1} \in [\mu, 1-\mu]$ and $\mu \in [0.5, 1)$, the implicit policy remains well-defined (non-negative density). Using this implicit policy, we can rewrite the bootstrapped flow-matching loss $\mathcal{L}^\mu_k(v_V)$ as

$$\mathcal{L}^\mu_k(v_V) \overset{\text{const.}}{=} \mathbb{E}_{\substack{s\sim D,\, a\sim\pi^\mu_{k-1}(a|s) \\ \tau\sim\text{Unif}([0,1]), \\ z^\tau\sim p^{k-1}_Q(z^\tau|s,a,\tau)}} \left[\Big(v_V(z^\tau \mid s,\tau) - v^{k-1}_Q(z^\tau \mid s,a,\tau) \Big)^2 \right],$$

where const. denotes the denominator in the implicit policy (Eq. 23) that does not affect the gradient of $\mathcal{L}^\mu_k(v_V)$ with respect to v_V. Similar to the discussion in Sec. 5, we can write the sampling of noisy return $z^\tau \sim p^{k-1}_Q(z^\tau \mid s,a,\tau)$ as transporting a Gaussian noise $z^0 \sim \mathcal{N}(0,1)$ using the flow $\phi^{k-1}_Q(z^\tau \mid s,a,\tau)$ induces by the state-action-value velocity $v^{k-1}_Q(z^\tau \mid s,a,\tau)$:

$$z^\tau \sim p^{k-1}_Q(z^\tau \mid s,a,\tau) \iff z^\tau \sim \mathcal{N}(0,1),\ z^\tau = \phi^{k-1}_Q(z^0 \mid s,a,\tau),$$

and define the state-value return process corresponds to the expectile-weighted objective $\mathcal{L}^\mu_k(v_V)$ as

$$X^k_V(s,\tau) \triangleq \phi^{k-1}_Q(N_Q \mid s,A,\tau), \quad \frac{\mathrm{d}}{\mathrm{d}\tau} X^k_V(s,\tau) \overset{(a)}{=} v^{k-1}_Q(\phi^{k-1}_Q(N_Q \mid s,A,\tau) \mid s,A,\tau),$$

where N_Q is the Gaussian random variable, A is the action random variable as in Sec. 5, and *(a)* is true because v^{k-1}_Q and ϕ^{k-1}_Q satisfy their own flow ODE (Eq. 2). We can also collect the set of random variables for this stochastic process as $\mathcal{X}^k_V = \{N_Q, A\}$. Applying Lemma 1 gives that the minimizer v^k_V of the expectile-weighted objective $\mathcal{L}^\mu_k(v_V)$ fits to the noise-level differentiation $\mathrm{d}X^k_V(s,\tau)/\mathrm{d}\tau$:

$$\begin{aligned} v^k_V(z^\tau \mid s,\tau) &= \arg\min_{v_V} \mathcal{L}^\mu_k(v_V) \\ &= \mathbb{E}_{\mathcal{X}^k_V}\left[\frac{\mathrm{d}}{\mathrm{d}\tau} X^k_V(s,\tau) \middle| X^k_V(s,\tau) = z^\tau \right] \\ &= \mathbb{E}_{a\sim\pi^\mu_{k-1}(a|s),\, z^\tau\sim p^{k-1}_Q(z^\tau|s,a,\tau)}\left[v^{k-1}_Q(z^\tau \mid s,a,\tau) \right]. \end{aligned}$$

In addition, Lemma 1 suggests that the state-value velocity $v^k_V(z^\tau \mid s,\tau)$ generates the probability density path of the stochstic process $X^k_V(s,\tau)$, i.e., $X^k_V(s,\tau)$ has the same law as the flow ϕ^k_V that transports a Gaussian random variable N_V independent of N_Q:

$$\phi^k_V(N_V \mid s,\tau) \overset{d}{=} X^k_V(s,\tau) = \phi^{k-1}_Q(N_Q \mid s,A,\tau).$$

We note that this identity is similar to the state-value return flow in Eq. 7.

Similarly, we can derive the minimizer v^k_Q of the state-action-value velocity objective $\mathcal{L}_k(v_Q)$. Because the implicit policy π^μ_{k-1} does not appear in the loss $\mathcal{L}_k(v_Q)$, we can make an analogy to the discussion in Sec. 5 directly. Specifically, we write the sampling of noisy return $z^\tau \sim p^{k-1}_V(z^\tau \mid s',\tau)$ in Eq. 20 as transporting a Gaussian noise $z^0 \sim \mathcal{N}(0,1)$ using the flow $\phi^{k-1}_V(z^\tau \mid s',\tau)$ induced by the state-value velocity $v^{k-1}_V(z^\tau \mid s',\tau)$:

$$z^\tau \sim p^{k-1}_V(z^\tau \mid s',\tau) \iff z^\tau \sim \mathcal{N}(0,1),\ z^\tau = \phi^{k-1}_V(z^0 \mid s',\tau),$$

and define the state-action-value return process corresponds to the state-action-value velocity $\mathcal{L}_k(v_Q)$ as

$$\begin{aligned} X^k_Q(s,a,\tau) &\triangleq \tau r(s,a) + \gamma\phi^{k-1}_V(N_V \mid S',\tau) + (1-\tau)\sqrt{1-\gamma^2}N, \\ \frac{\mathrm{d}}{\mathrm{d}\tau} X^k_Q(s,a,\tau) &\overset{(a)}{=} r(s,a) + \gamma v^{k-1}_V(\phi^{k-1}_V(N_V \mid S',\tau) \mid S',\tau) - \sqrt{1-\gamma^2}N, \end{aligned}$$

where N_V and N are independent Gaussian random variables as in Sec. 5, S' is the random variable of the next state as in Eq. 3, and *(a)* is correct because v^{k-1}_V and ϕ^{k-1}_V satisfy their own flow ODE

(Eq. 2). We also collect the set of random variables for this stochastic process as $\mathcal{X}_Q^k = \{N_V, N, S'\}$ and apply Lemma 1 to find the minimizer v_Q^k of the state-action-value velocity objective $\mathcal{L}_k(v_Q)$:

$$\begin{aligned} v_Q^k(z^\tau \mid s,a,\tau) &= \arg\min_{v_Q} \mathcal{L}_k(v_Q) \\ &= \mathbb{E}_{\mathcal{X}_Q^k}\left[\frac{\mathrm{d}}{\mathrm{d}\tau} X_Q^k(s,a,\tau)\middle| X_Q^k(s,a,\tau) = z^\tau\right] \\ &= \mathbb{E}_{\substack{s'\sim p(s'|s,a),\epsilon\sim\mathcal{N}(0,1) \\ z^\tau \sim p_V^{k-1}(z^\tau|s,a,\tau)}}\Big[(1-\gamma)r - \sqrt{1-\gamma^2}\epsilon \\ &\qquad +\gamma v_V^{k-1}\left(\tau(1-\gamma) + \gamma z^\tau + (1-\tau)\sqrt{1-\gamma^2}\epsilon \,\middle|\, s',\tau\right)\Big]. \end{aligned}$$

In addition, Lemma 1 suggests that the state-action-value velocity $v_Q^k(z^\tau \mid s,a,\tau)$ generates the probability density path of the stochastic process $X_Q^k(s,a,\tau)$, i.e., $X_Q^k(s,a,\tau)$ has the same law as the flow ϕ_V^k that transports a Gaussian random variable N_Q independent of N_V and N:

$$\phi_Q^k(N_Q \mid s,a,\tau) \overset{d}{=} \tau r(s,a) + \gamma\phi_V^{k-1}(N_V \mid S',\tau) + (1-\tau)\sqrt{1-\gamma^2}N.$$

We also note that this identity is similar to the state-action-value return flow in Eq. 8.

Therefore, under the assumption in Eq. 21 and Eq. 22 and when $k \to \infty$, the implicit policy can be written as

$$\pi_\infty^\mu(a \mid s) = \frac{\pi_\beta(a \mid s) w_\infty^\mu(s,a)}{\mathbb{E}_{\pi_\beta(a|s)}\left[w_\infty^\mu(s,a)\right]},$$

and we have

$$\begin{aligned} \phi_V^\infty(N_V \mid s,\tau) &\overset{d}{=} \phi_Q^\infty(N_Q \mid s,A,\tau), \\ \phi_Q^\infty(N_Q \mid s,a,\tau) &\overset{d}{=} \tau r(s,a) + \gamma\phi_V^\infty(N_V \mid S',\tau) + (1-\tau)\sqrt{1-\gamma^2}N, \end{aligned}$$

which is exactly the flow Bellman equations (Eq. 6) of the implicit policy π_∞^μ with a unique fixed point:

$$\phi_V^\infty(N_V \mid s,\tau) \overset{d}{=} Z_V^{\pi_\infty^\mu}(s,\tau), \quad \phi_Q^\infty(N_Q \mid s,a,\tau) \overset{d}{=} Z_Q^{\pi_\infty^\mu}(s,a,\tau).$$

As discussed in Sec. 4.1, at the noise level $\tau = 1$, we recover the standard distributional Bellman equations:

$$\phi_V^\infty(N_V \mid s,1) \overset{d}{=} \phi_Q^\infty(N_Q \mid s,A,1), \quad \phi_Q^\infty(N_Q \mid s,a) \overset{d}{=} r(s,a) + \gamma\phi_V^\infty(N_V \mid S',1).$$

Taking expectations over the random variables (N_Q, N, A, S') on both sides of these identities, we have

$$
\begin{aligned}
V^\infty(s) &\overset{(a)}{=} \mathbb{E}_{z^0\sim\mathcal{N}(0,1)}\left[\int_0^1 v_V^\infty(z^\tau \mid s,a,\tau)d\tau\right] \\
&= \mathbb{E}_{z^0\sim\mathcal{N}(0,1)}\left[\phi_V^\infty(z^0 \mid s,1)\right] \\
&\overset{(b)}{=} \mathbb{E}_{\substack{z^0\sim\mathcal{N}(0,1)\\ a\sim\pi_\infty^\mu(a|s)}}\left[\phi_Q^\infty(z^0 \mid s,a,1)\right] \\
&= \mathbb{E}_{\substack{z^0\sim\mathcal{N}(0,1)\\ a\sim\pi_\infty^\mu(a|s)}}\left[\int_0^1 v_Q^\infty(z^\tau \mid s,a,\tau)d\tau\right] \\
&\overset{(c)}{=} \mathbb{E}_{a\sim\pi_\infty^\mu(a|s)}\left[Q^\infty(s,a)\right], \\
Q^\infty(s,a) &\overset{(d)}{=} \mathbb{E}_{z^0\sim\mathcal{N}(0,1)}\left[\int_0^1 v_Q^\infty(z^\tau \mid s,a,\tau)\tau\right] \\
&= \mathbb{E}_{z^0\sim\mathcal{N}(0,1)}\left[\phi_Q^\infty(z^0 \mid s,a,1)\right] \\
&\overset{(e)}{=} (1-\gamma)r(s,a) + \gamma\mathbb{E}_{\substack{z^0\sim\mathcal{N}(0,1),\\ s'\sim p(s'|s,a)}}\left[\phi_V^\infty(z^0 \mid s',1)\right] \\
&= (1-\gamma)r(s,a) + \gamma\mathbb{E}_{\substack{z^0\sim\mathcal{N}(0,1),\\ s'\sim p(s'|s,a)}}\left[\int_0^1 v_V^\infty(z^0 \mid s',1)d\tau\right] \\
&\overset{(f)}{=} (1-\gamma)r(s,a) + \gamma\mathbb{E}_{s'\sim p(s'|s,a)}\left[V^\infty(s')\right],
\end{aligned}
$$

where *(a)* and *(d)* use the definition of V^∞ (Eq. 18), *(b)* and *(e)* apply the identities between ϕ_V^∞ and ϕ_Q^∞, and *(c)* and *(f)* use the definition of Q^∞ (Eq. 17). Since the implicit policy π_∞^ν itself is defined by the scalar value V^∞ and the scalar critic Q^∞. Applying Theorem 4.1 and Eq. 9 from Hansen-Estruch et al. (2023), we conclude that V^∞ and Q^∞ converges to the expectile Bellman equations. Because different expectiles μ induce different V^∞ and Q^∞, we use V_μ^∞ and Q_μ^∞ to denote V^∞ and Q^∞ for a fixed μ (V_τ and Q_τ in Kostrikov et al. (2022)).

Our answers to the first question help us to address the second question easily.

A2: Yes, increasing the expectile recovers the optimal value within the support of the dataset.

We reuse theoretical results from (Kostrikov et al., 2022) to show the answer. For a fixed expectile τ, when the number of iterations $k \to \infty$, we have shown that the scalar value V_μ^∞ and the scalar critic Q_μ^∞ induced by their velocities satisfy the expectile Bellman equations: for any (s,a),

$$
V_\mu^\infty(s) = \mathbb{E}_{a\sim\pi_\infty^\mu(a|s)}\left[Q_\mu^\infty(s,a)\right], \quad Q_\mu^\infty(s,a) = (1-\gamma)r(s,a) + \gamma\mathbb{E}_{s'\sim p(s'|s,a)}\left[V_\mu^\infty(s')\right].
$$

Applying Theorem 3 from Kostrikov et al. (2022) yields that, when the expectile μ reaches 1,

$$
\begin{aligned}
\lim_{\mu\to 1} V_\mu^\infty(s) &= \max_{a:\,\pi_\beta(a|s)>0} Q_\beta^\star(s,a) = V_\beta^\star(s), \\
\lim_{\mu\to 1} Q_\mu^\infty(s,a) &= (1-\gamma)r(s,a) + \gamma\mathbb{E}_{s'\sim p(s'|s,a)}\left[V_\beta^\star(s,a)\right] = Q_\beta^\star(s,a),
\end{aligned}
$$

where $V_\beta^\star$ and $Q_\beta^\star$ are optimal value and critic within the support of the dataset: $a \in \mathcal{A}$ s.t. $\pi_\beta(a \mid s) > 0$. Therefore, we conclude that, when increasing the expectile μ to 1, the scalar value V_μ^∞ and the scalar critic Q_μ^∞ recover the optimal value $V_\beta^\star$ and the optimal critic $Q_\beta^\star$ within the support of the dataset.

B Additional Experiments

B.1 Visualizing return flows across noise levels

We next examine how the learned flows transport Gaussian noise into return predictions. Following the policy evaluation setup in Sec. 7.1, we compare VF, FlowCritic, and Value Flows on the same

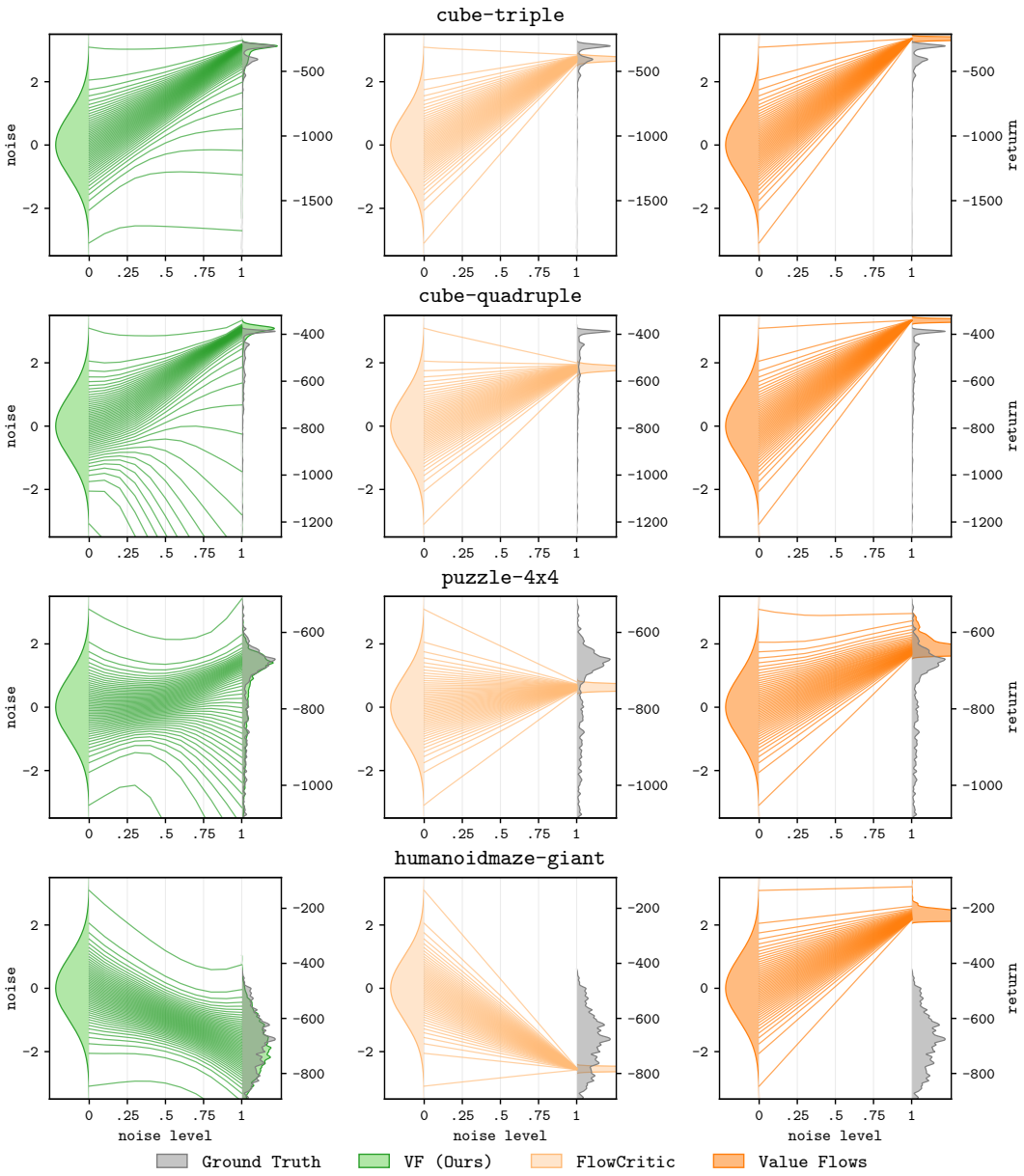
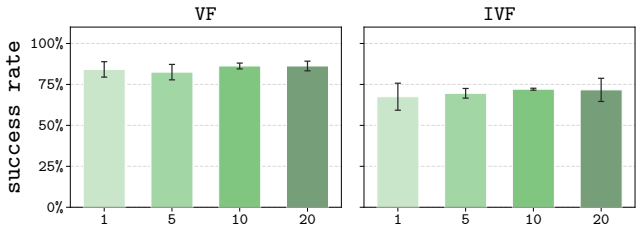


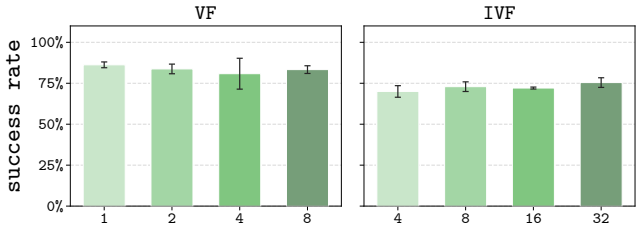
Figure 6: **Value Forcing learns return flows that better capture the spread of ground-truth returns.** Following the policy evaluation experiments in Sec. 2, we compare the return flow paths of VF *(Left)*, FlowCritic *(Center)*, and Value Flows *(Right)* from Gaussian noise ($\tau = 0$) to predicted returns ($\tau = 1$). VF spreads its return paths at early noise levels (around $\tau = 0.25$), while baselines concentrate their return paths close to the flow endpoint (around $\tau = 0.9$).

four states from each domain. Each state comes with a fixed first action chunk. For each method, we sample initial noise $z^0 \sim \mathcal{N}(0, 1)$ and numerically integrate the learned return velocity from $\tau = 0$ to $\tau = 1$ with 10 Euler steps, recording the intermediate values z^τ. Each curve follows one initial noise sample throughout integration. We draw the return flow paths of 51 return samples starting from the 0th, 2nd, 4th, …, 100th quantiles of the Gaussian distribution. We also plot the initial noise density and the predicted return density, together with the ground-truth return density at the endpoint. We show return predictions that span from the 5th percentile of the ground-truth returns to the largest return, extended to cover every Value Flows prediction.

Fig. 6 shows the evolutions of return flow paths. Although all methods start from Gaussian noise, VF preserves flow paths that lead to different regions of the return distribution. FlowCritic and



(a) Number of return flow steps during ODE integration.



(b) Number of return samples M for estimating V and Q

Figure 7: **Return flow hyperparameter ablations.** We conduct ablation experiments to study the sensitivity of hyperparameters related to return flows on task 2 and task 4 of `cube-triple` and `puzzle-4x4`. For each hyperparameter, we show the success rates after aggregating over these 4 tasks.

Value Flows instead concentrate many flow paths toward a common return value as the noise level approaches 1. The broader predictions of VF better capture the variation in ground-truth returns, particularly on `puzzle-4x4` and `humanoidmaze-giant`. In addition, different methods also differ in where their trajectories terminate. On `cube-quadruple` and `humanoidmaze-giant`, FlowCritic underestimates the return distribution, while Value Flows overestimates it. VF captures both the location and spread of the ground-truth return distributions. Together with the 1-Wasserstein distance comparisons in Fig. 2, these visualizations suggest that noisy return alignment helps spread the return samples while improving the accuracy of the final predictions.

B.2 Additional ablations on return flow hyperparameters

In this section, we study how the two important hyperparameters in return flow affect policy performance for both VF and IVF. Because of compute constraints, we ablate these parameters on `cube-triple` and `puzzle-4x4` with task 2 and task 4, which are representative tasks in that domain.

First, we ablate the number of flow steps used for the return flows v_V and v_Q. Fig. 7a shows that different number of flow steps has no significant impact on policy performance for both Value Forcing algorithms. Perhaps surprisingly, 1 flow step can yield similar performance.

Second, we ablate the number of return samples M we use to estimate V and Q (see Sec. 6). Fig. 7b shows that different numbers M also do not have a significant impact.

C Complete numerical results

For completeness, Tables 2-5 provide the full results for every task and dataset size. The results are averaged over 4 seeds, and we report the mean and the standard deviations for each task. We highlight the results within 95% of the best performance using .

D EXPERIMENTAL DETAILS

D.1 DATASET CONSTRUCTION AND EVALUATION PROTOCOL

Following Zhou et al. (2026), we use OGBench datasets with 100M transitions (Park et al., 2024b; 2025a) as the full offline dataset for `cube-triple`, `cube-quadruple`, `puzzle-4x4` and `humanoidmaze-giant`. These datasets are collected by scripted policies: `humanoidmaze-giant` has trajectories with 4000 time steps, and the other datasets have trajectories with 1000 time steps.

To subsample the full offline datasets with fractions within $\{10\%, 25\%, 50\%\}$, we draw a specific fraction of all trajectories in a shard whenever we swap shards. Specifically, for each trajectory in a shard, we compute the mean of rewards over time steps as the score and use a softmax distribution over these scores to sample trajectories without replacement. This subsampling procedure is invoked during the entire training process, suggesting that the subset selection depends on the random seed.

For evaluation, we follow a similar protocol as in Zhou et al. (2026). For each task in the OGBench benchmarks, the agent aims to reach a fixed goal. The episode terminates if the agent reaches the goal or exceeds the maximum number of time steps: 1000 for `cube-triple` and `cube-quadruple`, 500 for `puzzle-4x4`, and 4000 for `humanoidmaze-giant`. In Sec. 7.2 and Sec. 7.3, we train all methods for 500K gradient steps and evaluate them every 100K gradient steps for 10 episodes per task. We report the success rate at the final evaluation in Sec. 7.2, showing the mean and one standard deviation across 4 random seeds. In our ablation experiments (Sec. 7.3 and Appendix B.2), we report the success rate averaged over the last three evaluations: 300K, 400K, and 500K gradient steps. For figures that aggregate results over different tasks or domains, we first average the success rate at the same random seed and then compute the mean and the standard deviation over different random seeds.

D.2 METHODS FOR COMPARISON

We compare VF and IVF against 10 baselines in two categories spanning explicit methods and implicit methods (Sec. 7): 9 prior RL methods and 2 implicit variants of prior distributional RL methods that we construct for comparisons. Below, we describe the formulation of each method.

(1) Explicit methods.

Floq (Agrawalla et al., 2026). Floq fits a scalar Q function using flow matching. Let $v_Q(z^\tau \mid s, a, \tau)$ denote its state-action-value velocity and $\phi_Q(z^0 \mid s, a, \tau)$ denote the corresponding flow. Given a transition (s, a, r, s'), Floq constructs a TD target by averaging over M bootstrapped samples:

$$y = r + \frac{\gamma}{M} \sum_{i=1}^{M} \bar{\phi}_Q(z_i^0 \mid s', a_i', 1),$$

where z_i^0 is sampled from the initial noise distribution $p_0(z_i^0) \overset{d}{=} \text{Unif}(\kappa[r_{\min}, r_{\max}]/(1-\gamma))$, where κ is the noise coverage, a_i' is the next action sampled from a one-step policy that transports Gaussian noise: $a_i' = \pi_{\text{one-step}}(s', a_i^0)$, with $a_i^0 \sim \mathcal{N}(0, I_{|\mathcal{A}|})$, and $\bar{\phi}_Q$ is the flow induced by the target state-action-value velocity $\bar{v}_Q$. Here, $\pi_{\text{one-step}}$ denotes a one-step flow policy as in Park et al. (2025b). Each TD target independently draws the initial return noise $z_i^0 \sim p_0(z_i^0)$, and the policy noise a_i^0. To train the state-action-value velocity, Floq samples a noise level $\tau \sim \text{UNIF}([0,1])$, construct the linear interpolant $z^\tau = (1-\tau)z^0 + \tau y$, and fits

$$\arg\min_{v_Q} \mathbb{E}_{\substack{(s,a,r,s') \sim D,\ \tau \sim \text{UNIF}([0,1]),\\ z^0 \sim \mathcal{N}(0,1),\ \{a_i^0 \sim \mathcal{N}(0, I_{|\mathcal{A}|})\}_{i=1}^M,\\ \{z_i^0 \sim \text{UNIF}(\kappa[r_{\min}, r_{\max}]/(1-\gamma))\}_{i=1}^M}} \left[(v_Q(z^\tau \mid s, a, \tau)) - (y - z^0))^2 \right].$$

This loss fits a velocity to transport initial noise toward the averaged TD target. Floq therefore uses the flow to estimate the scalar Q rather than directly fitting the return distribution. The mean flow endpoint is then distilled into a scalar critic, $Q(s,a) \approx \mathbb{E}_{z^0 \sim p_0}[\phi_Q(z^0 \mid s, a, 1)]$. As in Park et al. (2025b), Floq learns a one-step flow policy to maximize the critic estimate while staying close to a flow BC policy:

$$\arg\min_{\pi_{\text{one-step}}} \mathbb{E}_{s \sim D,\ a^0 \sim \mathcal{N}(0,I)} \left[-Q(s, \pi_{\text{one-step}}(s, a^0)) + \alpha \left\| \pi_{\text{one-step}}(s, a^0) - \pi_\beta(s, a^0) \right\|_2^2 \right],$$

where $\pi_\beta(s, a^0)$ denotes the action obtained by integrating the BC flow from the same initial noise a^0, and α controls the regularization strength.

FlowCritic (Zhong et al., 2025). FlowCritic models the full return distribution with flow matching. This method constructs TD targets using the return at the flow endpoint and then fits them using the standard flow-matching objective (Eq. 3). Specifically, let $v_Q(z^\tau \mid s, a, \tau)$ denote its state-action-value velocity and $p_Q(z^\tau \mid s, a, \tau)$ denotes the corresponding probability density path. Given a transition (s, a, r, s'), we select the next action a' from the policy $\pi(a' \mid s')$. FlowCritic constructs the TD target a $y = r + \gamma z^1$, where $z^1 \sim \bar{p}_Q(z^1 \mid s', a', 1)$ is sampled from the probability density induced by the target state-action-value velocity $\bar{v}_Q$ at the next time step, and a' is the next action sampled from the policy $\pi(a' \mid s')$ that is based on flow rejection sampling as in Eq. 13. To train the critic velocity, FlowCritic samples a noise level $\tau \sim \text{UNIF}([0, 1])$, constructs the linear interpolant $z^\tau = (1 - \tau)z^0 + \tau y$, and fits

$$\arg\min_{v_Q} \mathbb{E}_{\substack{(s,a,r,s')\sim D,\ \tau\sim\text{UNIF}([0,1]),\\ z^0\sim\mathcal{N}(0,1),\ z^1\sim\bar{p}_Q(z^1|s,a,\tau)}} \left[(v_Q(z^\tau \mid s, a, \tau) - (y - z^0))^2\right]. \tag{24}$$

Importantly, because the state-action-value velocity loss only uses the *clean* return sample at the next time step, it does not enforce Bellman consistency at intermediate noise levels. For flow rejection sampling, FlowCritic generates M MC return samples from the learned state-action-value velocity and discards the m largest ones to construct the critic estimate:

$$z_i^1 = z_i^0 + \int_0^1 v_Q(z_i^\tau \mid s, a, \tau)d\tau,\ z_i^0 \sim \mathcal{N}(0, 1) \overset{\text{ranking}}{\Longrightarrow} z_1^1 \le \cdots \le z_M^1, \quad Q(s, a) = \frac{1}{M - m}\sum_{i=1}^{M-m} z_i^1.$$

Value Flows (Dong et al., 2025). Value Flows models the return distribution using two complementary flow-matching objectives: DCFM loss and BCFM loss. The DCFM loss aims to incorporate the standard distributional Bellman equation (Eq. 1) into every noise level. However, as discussed in Sec. 4.1, naively applying the standard distributional Bellman equation at all noise levels results in incorrect supervision at the noise level $\tau = 0$. Specifically, let $v_Q(z^\tau \mid s, a, \tau)$ denote the state-action-value velocity and $p_Q(z^\tau \mid s, a, \tau)$ denotes the corresponding probability density path. Given a transition $(s, a, r, s') \sim D$, we sample the next action $a' \sim \pi(a' \mid s')$ using flow rejection sampling (Eq. 13). The DCFM loss bootstraps directly from the target state-action-value velocity at the next state: we generate a noisy return $z^\tau \sim \bar{p}_Q(z^\tau \mid s', a', \tau)$ and minimizes

$$\mathcal{L}_{\text{DCFM}}(v_Q) = \mathbb{E}_{\substack{(s,a,r,s')\sim D,\ a'\sim\pi(a'|s'),\\ \tau\sim\text{UNIF}([0,1]),\\ z^\tau\sim\bar{p}_Q(z^\tau|s',a',\tau)}} \left[\left(v_Q(r + \gamma z^\tau \mid s, a, \tau) - \bar{v}_Q(z^\tau \mid s', a', \tau)\right)^2\right],$$

where $\bar{v}_Q$ is the target state-action-value velocity. To stabilize learning, Value Flows also includes a BCFM loss $\mathcal{L}_{\text{BCFM}}(v_Q)$, which is the same as the objective used by FlowCritic in Eq. 24. Following the original implementation, we also include a confidence weight $w(s, a, z^0)$ with temperature λ_w to reweight both $\mathcal{L}_{\text{DCFM}}$ and $\mathcal{L}_{\text{BCFM}}$. The final loss function is $\mathcal{L}_{\text{DCFM}} + \lambda\mathcal{L}_{\text{BCFM}}$ with λ balancing the strengths of both losses. For flow rejection sampling, Value Flows selects action candidates using a critic estimate derived from one-step integration of return samples: $Q(s, a) \approx z^0 + v_Q(z^0 \mid s, a, 0), z^0 \sim \mathcal{N}(0, 1)$.

PCBF (Xu et al., 2026). PCBF models return distributions using flow matching with two return flow paths sharing the same initial noise. It also includes a control-variate target. Specifically, given a transition $(s, a, r, s') \sim D$, we sample the next action $a' \sim \pi(a' \mid s')$ using flow rejection sampling (Eq. 13). We draw $z^0 \sim \mathcal{N}(0, 1)$ and generate a return at the next time step: $z' = \bar{\phi}(z^1 \mid s', a', 1)$, where $\bar{\phi}_Q$ is the return flow path corresponding to the target velocity $\bar{v}_Q$. Using the same noise sample, PCBF constructs the two interpolation paths:

$$\tilde{z}^\tau = (1 - \tau)z^0 + \tau z',$$
$$z^\tau = (1 - \tau)z^0 + \tau(r + \gamma z') = \tau r + \gamma\tilde{z}^\tau + (1 - \tau)(1 - \gamma)z^0,$$

where the second interpolation preserves the initial noise z^0 at $\tau = 0$, while the other endpoint satisfies $z^1 = r + \gamma z'$. To reduce the variance of the sample-based velocity target, PCBF introduces the control-variate correction

$$u_\lambda = r + \gamma z' - z^0 + \lambda\left[\bar{v}_Q(\tilde{z}^\tau \mid s', a', \tau) - (z' - z^0)\right],$$

with λ as a balancing coefficient, and minimizes

$$\arg\min_{v_Q} \mathbb{E}_{\substack{(s,a,r,s')\sim D,\ a'\sim\pi(a'|s'),\\ z^0\sim\mathcal{N}(0,1),\ \tau\sim\text{UNIF}([0,1])}} \left[\left(v_Q(z^\tau \mid s,a,\tau) - u_\lambda \right)^2 \right],$$

Setting $\lambda = 0$ recovers the flow-matching objective in FlowCritic, while $\lambda > 0$ controls a bias-variance trade-off.

C51 (Bellemare et al., 2017). C51 represents the return distribution as a categorical distribution over fixed atoms on the return support. This method minimizes the cross-entropy between this distribution and the projection of the distributional Bellman target onto the atoms. Following Dong et al. (2025), we pair C51 with a flow BC policy and rejection sampling (Eq. 13).

IQN (Dabney et al., 2018a). IQN is a distributional RL method that learns the full return distribution by modeling the quantile function at randomly sampled quantile fractions. This method learns to fit the quantile function using a Huber loss with a threshold κ. Following Dong et al. (2025), we pair IQN with a flow BC policy and rejection sampling (Eq. 13).

QC (Li et al., 2025). Q-chunking (QC) learns a critic over action chunks with standard TD learning and selects actions by rejection sampling using a flow BC policy (Eq. 13).

VF (Ours). VF follows the pseudo code in Alg. 1. We learn the state-action-value velocity and the state-value velocity using the regression objectives derived from the corresponding stochastic processes (Eq. 11 and Eq. 12). We learn a flow BC policy and select actions using rejection sampling as in Eq. 13. We also use this rejection sampling to select the actions in the state-value velocity loss (Eq. 11). During training, VF also selects action candidates using a critic estimate derived from one-step integration of return samples: $Q(s,a) \approx z^0 + v_Q(z^0 \mid s,a,0), z^0 \sim \mathcal{N}(0,1)$.

(2) Implicit methods.

DIVL (Wang et al., 2026). DIVL replaces scalar expectile regression with a categorical model of the state-action values associated with dataset actions. Let $p_{V,i}(s)$ denote the probability assigned to atom c_i with in a set of atoms $\{c_i\}_{i=1}^N$, and let $\bar{Q}_k(s,a)$ denote the target Q estimate. The categorical model minimizes

$$= -\mathbb{E}_{(s,a)\sim D}\left[\sum_{i=1}^{N} m_i\big(\bar{Q}(s,a)\big) \log p_{V,j}(s) \right],$$

where $m_i(q)$ is the two-hot projection of q onto the neighboring atoms. Thus, the learned model captures variation in state-action values across dataset actions at each state. DIVL extracts a bootstrap target from this model using an adaptive quantile:

$$\bar{\mathcal{H}}(s) = -\frac{1}{\log N} \sum_{i=1}^{N} p_{V,i}(s) \log p_{V,i}(s), \quad \varrho(s) = \text{clip}\left(\varrho_{\text{base}} - \eta\, \bar{\mathcal{H}}(s),\ 0,\ 1\right),$$

$$V_\varrho(s) = \min\left\{ c_i : \sum_{j=1}^{i} p_{V,j}(s) \geq \varrho(s) \right\}.$$

A higher entropy $\bar{H}(s)$ lowers the selected quantile, making the bootstrap targer less optimistic. For a transition $(s,a,r,s') \sim D$, the Q estimate minimizes

$$\mathbb{E}_{(s,a,r,s')\sim D}\left[\left(Q(s,a) - (r + \gamma V_\varrho(s')) \right)^2 \right].$$

We replace the original adjoint-matching policy extraction with flow rejection sampling (Eq. 13), ranking action candidates by $Q(s,a)$.

Implicit C51. Implicit C51 is an implicit variant of C51 (Bellemare et al., 2017) that we construct for comparison. Instead of using the expectile weight to fit the state-value velocity as in Eq. 15, we learn use the expectile weight to fit the state-value categorical distribution. Let $p_{Q_k,i}(s,a)$ and $p_{V,i}(s)$ denote the probabilities assigned to atom c_i by the state-action-value model and the state-value model, respectively. Their scalar means are

$$Q(s,a) = \sum_{i=1}^{N} c_i p_{Q,i}(s,a), \qquad V(s) = \sum_{i=1}^{N} c_i p_{V,i}(s). \tag{25}$$

For each transition (s,a,r,s'), we minimize the expectile weighted cross-entropy

$$-\mathbb{E}_{(s,a)\sim D}\left[w^{\mu}(s,a)\sum_{i=1}^{N}\bar{p}_{Q,i}(s,a)\log p_{V,i}(s)\right].$$

with $w^{\mu}(s,a)$ as in Eq. 14 but defined by the Q and V in Eq. 25, to learn the state-value distribution. For the state-action-value distribution, we use the state-value distribution at the next state s' to construct the TD target.

Implicit IQN. Implicit IQN is an implicit variant of IQN that we construct for comparison. Following Kostrikov et al. (2022), implicit IQN adds a state-value quantile $q_V(s,\omega)$ that uses the IQN network architecture without the action input. Let $q_Q(s,a,\omega)$ denote the state-action-value quantile at quantile fraction $\omega \sim \textsc{Unif}([0,1])$. The scalar Q estimate and V estimate can be written as

$$Q(s,a) = \mathbb{E}_{\omega}[q_Q(s,a,\omega)], \quad V(s) = \mathbb{E}_{\omega}[q_V(s,\omega)]. \tag{26}$$

Similar to Implicit C51, we learn the state-value quantile by minimizing the expectile weighted loss

$$\mathbb{E}_{(s,a)\sim D,\ \omega\sim\textsc{Unif}([0,1])}\left[w^{\mu}(s,a)\left(q_V(s,\omega)-\bar{q}_Q(s,a,\omega)\right)^2\right],$$

with $w^{\mu}(s,a)$ as in Eq. 14 but defined by the Q and V in Eq. 26. For the state-action-value quantile, we use the state-value quantile at the next state s' to construct the TD target for the quantile Huber loss with threshold κ:

$$\rho^{\kappa}_{\omega}(u) = |\omega - \mathbb{1}\{u<0\}|\frac{\mathcal{H}_{\kappa}(u)}{\kappa}, \qquad \mathcal{H}_{\kappa}(u) = \begin{cases} \frac{1}{2}u^2, & |u|\le\kappa, \\ \kappa\left(|u|-\frac{1}{2}\kappa\right), & |u|>\kappa. \end{cases}$$

BFN (Hansen-Estruch et al., 2023; Li et al., 2025). BFN combines the scalar Q and V estimate from IQL (Kostrikov et al., 2022) with best-of-N rejection sampling from a flow BC policy. To make it consistent with other methods, we use K to denote the number of action candidates instead of N.

IVF (Ours). IVF follows the pseudo code in Alg. 1. Similar to VF, we learn the state-action-value velocity and the state-value velocity using the regression objectives derived from the corresponding stochastic processes (Eq. 15 and Eq. 12). We learn a flow BC policy and select actions using rejection sampling as in Eq. 13.

D.3 Implementations and hyperparameters

We implement VF and IVF and all baselines in a `JAX` (JAX Developers, 2025) codebase, adapting implementations from Floq (Agrawalla et al., 2026), Value Flows (Dong et al., 2025), PCBF (Xu et al., 2026), FQL (Park et al., 2025b), and QGF (Zhou et al., 2026). We describe the implementation details and hyperparameters for each method below.

Hyperparameter tuning. Following Zhou et al. (2026), we tune method-specific hyperparameters per domain rather than per task. We train on tasks 2 and 4 of each domain with the full dataset, and average the success rates at 500K gradient steps over tasks and seeds. We then use the selected values for all 5 tasks and all data budgets. We show common hyperparameters in Table 6 and domain-specific hyperparameters in Table 7.

(1) Explicit methods.

Floq. We use α to denote the BC regularization coefficient. Following their original implementation, the state-action-value velocity preprocesses the noisy critic using an HL-Gauss histogram (Imani & White, 2018) and preprocesses the noise level using cosine embedding. We use a uniform distribution $\text{UNIF}(\kappa \cdot [r_{\text{min}}, r_{\text{max}}]/(1-\gamma))$ to sample the initial noise. We use an MLP with 512 units and different depths (`vel-depth`) to parameterize the state-action-value velocity. For κ and `vel-depth`, we reuse values from the most similar domains in the original implementations. We sweep α within $\alpha \in \{300, 1000, 3000, 10000\}$ on each domain separately.

FlowCritic. To speed up training, flow rejection sampling uses $M_{\text{train}} = 4$ with 1 Euler steps (Tweedie's formula (Dong et al., 2025; Zhou et al., 2026)). During evaluation, we use $M_{\text{eval}} = 10$ with 10 Euler steps. For both phases, we discard the largest return sample ($m = 1$). We do not tune hyperparameters for FlowCritic.

Value Flows. Value Flows learns the return distribution with a two-member state-action-value velocity similar to the twin Q trick in Fujimoto et al. (2018). Following the original implementation, we clip returns to the range $[r_{\text{min}}, r_{\text{max}}]/(1-\gamma)$ after every integration step of the return flow. We also clip the BC actions after every Euler step. We tune $\lambda \in \{0.3, 1, 3, 10\}$ and $\lambda_w \in \{0.3, 1\}$ jointly.

PCBF. PCBF also learns the return distribution with a two-member state-action-value velocity similar to the twin Q trick in Fujimoto et al. (2018). We also use the same return flow clipping and action flow clipping as in Value Flows. We sweep $\lambda \in \{0.3, 0.4, 0.6\}$ on the two `cube` domains and $\lambda \in \{0.1, 0.2, 0.4\}$ on `puzzle-4x4`.

C51. We use the twin Q trick from Fujimoto et al. (2018) in C51. We use a fixed number of 51 atoms spread evenly on $[r_{\text{min}}, r_{\text{max}}]/(1-\gamma)$. We do not tune C51.

IQN. IQN uses a network that takes in a quantile fraction within $(0, 1)$ as input, and embeds it using 64-dimensional cosine features. We also use the twin Q trick from Fujimoto et al. (2018) in IQN. For estimating the $Q(s, a)$, we average the quantile predictions from 8 random fractions, drawn independently from a uniform distribution $\text{UNIF}([0, 1])$. The (s, a) input and the cosine features of the quantile are each projected linearly to $d_{\text{emb}} = 32$ dimensions. The two projections are multiplied elementwise and then passed to an MLP. We tune the threshold κ within $\{30, 300, 3000\}$.

QC. We use the best-of-N variant of QC. In this variant, the best of the K action candidates at the next state is used as the next action in the TD target: the critic evaluates the rejection-sampling policy. We use the twin Q trick from Fujimoto et al. (2018) in QC. We use $K = 16$ action candidates on every domain, as in the other methods. We do not tune QC.

VF (Ours). The state-action-value velocity and the state-value velocity are each ensembles of two members as in the twin Q trick from (Fujimoto et al., 2018). We use $K = 16$ action candidates during both training and evaluation. During training, we use $M = 1$ one-step integration of return samples to approximate the critic. During evaluation, VF ranks $K = 16$ action candidates by a full 10-step intergration over $M = 16$ return samples whose initial noise is shared across candidates to reduce variance. We do not tune hyperparameters for VF.

(2) Implicit methods.

DIVL. We tune $\varrho_{\text{base}} \in \{0.7, 0.8, 0.9\}$, while retaining $\eta = 0.3$. We use 2001 atoms on $[r_{\text{min}}/(1-\gamma), r_{\text{max}}/(1-\gamma)]$, increasing the number of atoms from the original 201 to obtain finer quantile resolution.

Implicit C51. We use the same atoms, support, state-value model, and state-action-value model as C51. We adopt IVF's domain-specific expectile μ (Table 7) without separately tuning it for Implicit C51.

Implicit IQN. We reuse κ from IQN and use $\mu = 0.9$ on all domains. We use 8 sampled quantile fractions, 64 cosine features, embedding dimension $d_{\text{emb}} = 32$.

BFN. We use the twin Q trick from (Fujimoto et al., 2018) in BFN. We use expectile $\mu = 0.9$ for learning the Q and V implicitly. We do not tune BFN.

IVF (Ours). We use one network for both the state-value velocity and state-action-value velocity as adapted from the classifier-free guidance literature (Ho & Salimans, 2022). This network contains two emebers as in the twin Q trick from (Fujimoto et al., 2018). We tune $\mu \in \{0.7, 0.8, 0.9\}$ and fixed $M = 16$ on all domains.

D.4 Value Forcing improves return estimation accuracy

For each domain, we restore a single state from the OGBench 100M dataset, execute a fixed starting action chunk, and then follow a fixed data-collection policy until success or the time limit. The data-collection policy is the scripted policy with action noise or a noisy expert policy from Park et al. (2024a). From each representative state, we collect a training set and a held-out evaluation set, each with 1000 trajectories. We use reward functions of `cube-triple-task5`, `cube-quadruple-task2`, `puzzle-4x4-task4` and `humanoidmaze-giant-task2` to compute the return distributions.

We train VF, Value Flows, FlowCritic and `Oracle` on each training set for 100K gradient steps using each method's default hyperparameters on each domain. To evaluate the data-collection policy, all methods except `Oracle` bootstrap with the action chunk in the dataset (SARSA). For `Oracle`, it instead fits the discounted returns from different trajectories with the standard flow-matching loss (Eq. 3). We draw 5000 return samples from each estimator at the single state and action chunk pair to visualize the predicted return histograms and compute the 1-Wasserstein distance to the 1000 evaluation returns. Our histograms use 200 bins between the 10th percentile and the maximum of the ground-truth returns, widened to include all Value Flows samples.

D.5 Key design decisions in Value Forcing

Below, we describe the alternative instantiations of RL algorithms evaluated in Fig. 4. Since we use a target state-action-value velocity $v_Q^{\bar{\psi}}$ to update the state-value velocity in Alg. 1, we will use $\bar{v}_Q$ to simplify its notation and use $\bar{p}_Q$ to denote its induced noisy return distribution.

VF without the state-value velocity. Similar to prior actor-critic algorithms (Fujimoto et al., 2018; Haarnoja et al., 2018) that only learn a scalar critic estimate, one variant of VF can only learn a state-action-value velocity by bootstrapping the exponential moving average of itself (`no state-value velocity`).

Specifically, we can remove the state-value velocity in VF and use the target state-action-value velocity at the next action $a' \sim \pi(a' \mid s')$ to sample the noisy return $z^\tau \sim \bar{p}_Q(z^\tau \mid s', a', \tau)$. We also replace the state-value velocity v_V in the regression target with the target state-action-value velocity $\bar{v}_Q$, resulting in the following objective for `no value velocity`:

$$\arg\min_{v_Q} \mathbb{E}_{\substack{s'\sim p(s'|s,a),\, a'\sim\pi(a'|s') \\ \tau\sim\text{Unif}([0,1]), \\ \epsilon\sim\mathcal{N}(0,1),\, z^\tau\sim\bar{p}_Q(z^\tau|s',a',\tau)}} \Big[\Big(v_Q\left(\tau(1-\gamma)r + \gamma z^\tau + (1-\tau)\sqrt{1-\gamma^2}\epsilon \,\middle|\, s, a, \tau\right) - \left((1-\gamma)r + \gamma \bar{v}_Q(z^\tau \mid s', a', \tau) - \sqrt{1-\gamma^2}\epsilon\right)\Big)^2\Big],$$

where the policy π is the rejection sampling policy as in Eq. 13. This loss directly instantiates the flow Bellman backup without using the state-value velocity in the critic loss.

VF with a one-step actor. Following Park et al. (2025b), the `one-step actor` variant of VF replaces rejection sampling with a one-step policy $\pi_{\text{one-step}}(s, a^0)$, where $a^0 \sim \mathcal{N}(0, I_{|\mathcal{A}|})$. Let $\pi_\beta(s, a^0)$ denote the action generated by the BC flow policy from integrating the same initial noise a^0. We train the BC flow policy using the standard conditional flow matching loss $\mathcal{L}(\pi_\beta)$ as in Eq. 13. Then, the one-step policy learns to maximize the scalar Q estimate induced by our state-action-value velocity while distilling from the BC flow policy:

$$\arg\min_{\pi_{\text{one-step}}} \mathbb{E}_{s\sim\mathcal{D},\, a^0\sim\mathcal{N}(0,I_{|\mathcal{A}|})}\left[-Q(s, \pi_{\text{one-step}}(s, a^0)) + \alpha\left\|\pi_{\text{one-step}}(s, a^0) - \pi_\beta(s, a^0)\right\|_2^2\right], \quad (27)$$

where $\alpha \geq 0$ controls the strength of the distillation regularizer. Again, we compute the Q estimate by averaging over M MC return samples and propagate gradients for the one-step actor through the return flow integration. For the ablation experiments, we tune $\alpha \in \{1, 3, 10\}$ for each domain by averaging the success rates across task 2 and task 4 and choose $\alpha = 3$ for both `cube-triple` and `puzzle-4x4`. In addition, we use the one-step actor $\pi_{\text{one-step}}$ to sample actions in the state-value velocity loss (Eq. 11) during training and select actions during evaluation.

IVF with naive expectile regression. The most naive way to extend the expectile regression in Kostrikov et al. (2022) to return estimation is to directly apply the expectile loss to the value residual (as opposed to Eq. 15). We call this variant of IVF `expectile regression`. Specifically, we can define the naive expectile weight as

$$w^{\mu}_{\text{naive}}(s, a, z^\tau, \tau) \triangleq \mu \mathbb{1}(\bar{v}_Q(z^\tau \mid s, a, \tau) \geq v_V(z^\tau \mid s, \tau)) + (1 - \mu)\mathbb{1}(\bar{v}_Q(z^\tau \mid s, a, \tau) < v_V(z^\tau \mid s, \tau)),$$

and then use it to reweight the state-value velocity loss:

$$\begin{aligned}
&\arg\min_{v_V} \mathbb{E}_{a\sim D,\, \tau\sim\text{Unif}([0,1]),\, z^\tau\sim p_Q(z^\tau|s,a,\tau)} \left[w^{\mu}_{\text{naive}}(s, a)(\bar{v}_Q(z^\tau \mid s, a, \tau) - v_V(z^\tau \mid s, \tau))^2\right] \\
&= \arg\min_{v_V} \mathbb{E}_{a\sim D,\, \tau\sim\text{Unif}([0,1]),\, z^\tau\sim p_Q(z^\tau|s,a,\tau)} \left[\ell^2_\mu\left(\bar{v}_Q(z^\tau \mid s, a, \tau) - v_V(z^\tau \mid s, \tau)\right)\right],
\end{aligned}$$

where $\ell^2_\mu(\cdot)$ is the standard expectile regression loss with expectile μ. However, when μ approaches 1, applying Lemma 1 of Kostrikov et al. (2022), the minimizer of the expectile regression loss above $v_V^{\mu,\,\text{naive}}$ satisfies

$$\lim_{\mu\to 1} v_V^{\mu,\,\text{naive}}(z^\tau \mid s, \tau) = \max_{a:\, \pi_\beta(a|s)>0} \bar{v}_Q(z^\tau \mid s, a, \tau).$$

Integrating both sides gives an *upper envelope* over the state-action-value velocity instead of the maximum Q estimate within the dataset support:

$$\begin{aligned}
\lim_{\mu\to 1} \mathbb{E}_{z^0\sim\mathcal{N}(0,1)}\left[\int_0^1 v_V^{\mu,\,\text{naive}}(z^\tau \mid s, \tau)d\tau\right] &= \mathbb{E}_{z^0\sim\mathcal{N}(0,1)}\left[\int_0^1 \max_{a:\, \pi_\beta(a|s)>0} \bar{v}_Q(z^\tau \mid s, a, \tau)d\tau\right] \\
&\geq \max_{a:\, \pi_\beta(a|s)>0} \mathbb{E}_{z^0\sim\mathcal{N}(0,1)}\left[\int_0^1 \bar{v}_Q(z^\tau \mid s, a, \tau)d\tau\right] \\
&= \max_{a:\, \pi_\beta(a|s)>0} \bar{Q}(s, a).
\end{aligned}$$

Thus, naively applying expectile regression to the return velocity residual does not recover the value and critic estimates as in Kostrikov et al. (2022).

IVF with quantile weights. Prior work has found that a 1-dimensional flow avoids crossing along the flow paths (Lemma 3.9 of Liu et al. (2022)). This property preserves quantiles (Lemma 7 of (Bansal et al., 2026)): the q-quantile of the noise distribution maps exactly to the q-quantile of the target distribution. Using this property, we can derive a variant of IVF called `quantile weight` that uses the q-quantile of the return flows to construct the weights in Eq. 15. Specfically, for a quantile $q \in (0, 1)$, we can define the quantile weight with $\mu \in [0.5, 1)$ as

$$w^{\mu}_q(s, a) \triangleq \mu \mathbb{1}(\bar{Q}_q(s, a) \geq V_q(s)) + (1 - \mu)\mathbb{1}(\bar{Q}_q(s, a) < V_q(s)),$$

where $\bar{Q}_q(s, a)$ and $V_q(s)$ are q-quantiles of the state-action-value return flow and the state-value return flow obtained by integrating the q-quantile of the standard Gaussian:

$$\bar{Q}_q(s, a) = \int \bar{v}_Q(z^\tau \mid s, a, \tau)d\tau, \quad V_q(s) = \int v_V(z^\tau \mid s, \tau)d\tau, \quad z^0 = \text{quantile}_{\mathcal{N}(0,1)}(q).$$

Note that we do not need to compute quantiles $\bar{Q}_q$ and V_q using MC return samples because of the quantile-preserving property. For the ablation experiments, we tune $q \in \{0.7, 0.8, 0.9\}$ for each domain by averaging the success rates across task 2 and task 4 and choose $q = 0.9$ for both `cube-triple` and `puzzle-4x4`.

Table 1: **Comparison of Bellman constraints across flow-based value learning methods.** Among the methods compared, VF and IVF model the full return distribution, while enforcing Bellman consistency at every noise level. This provides consistent supervision throughout the flow path, improving return estimation (Sec. 7.1) and offline RL performance (Sec. 7.2), particularly when data are limited.

Method	Floq Agrawalla et al. (2026)	FlowCritic Zhong et al. (2025)	FlowIQN Groom et al. (2026)	Value Flows Dong et al. (2025)	EVOR Espinosa-Dice et al. (2025)	PCBF Xu et al. (2026)	VF/IVF (Ours)
Full return distribution?	✗	✓	✓	✓	✓	✓	✓
Regression target	scalar endpoint	endpoint samples	quantile-coupled samples	return velocity	return velocity	noise-coupled velocity	return velocity
Gaussians at initial noise levels?	✗	✗	✗	✗[†]	✗	✓	✓
Bellman constraints at intermediate noise levels?	✗	✗	✗	✓	✗[‡]	✗[§]	✓
Bellman constraints at final noise levels?	✓	✓	✓	✓	✓	✓	✓

[†]Value Flows imposes a Bellman equation on intermediate marginals, but it does not recover Gaussians at the initial noise levels. [‡]EVOR uses return velocities to construct TD targets directly. [§]PCBF couples two flow paths using shared initial noises rather than enforcing each intermediate marginal distribution to satisfy a Bellman equation.

Table 2: Complete results for explicit and implicit methods with 10% of the dataset. We highlight nonzero mean performance at or above 95% of the best in each row within each method group in green.

	Explicit methods								*Implicit methods*				
Environment	Floq	FlowCritic	Value Flows	PCBF	C51	IQN	QC	VF (Ours)	Implicit DIVL	Implicit C51	Implicit IQN	BFN	IVF (Ours)
cube-triple-task1	0 ±0	75 ±21	55 ±21	75 ±7	75 ±35	100 ±0	95 ±10	100 ±0	0 ±0	95 ±7	50 ±14	95 ±7	100 ±0
cube-triple-task2	0 ±0	60 ±57	75 ±7	45 ±21	80 ±0	65 ±7	48 ±13	90 ±8	0 ±0	10 ±0	30 ±28	40 ±14	78 ±33
cube-triple-task3	0 ±0	90 ±14	60 ±28	40 ±0	35 ±50	90 ±14	63 ±5	83 ±15	0 ±0	35 ±7	25 ±7	35 ±21	58 ±21
cube-triple-task4	0 ±0	25 ±21	5 ±7	20 ±0	5 ±7	40 ±57	25 ±13	33 ±26	0 ±0	0 ±0	10 ±14	5 ±7	20 ±8
cube-triple-task5	0 ±0	75 ±7	50 ±71	35 ±7	60 ±28	30 ±28	65 ±17	85 ±17	0 ±0	25 ±21	30 ±0	10 ±0	70 ±32
Domain average	0 ±0	65 ±13	49 ±13	43 ±1	51 ±24	65 ±1	59 ±5	78 ±10	0 ±0	33 ±4	29 ±13	37 ±1	65 ±4
cube-quadruple-task1	0 ±0	10 ±14	5 ±7	65 ±7	0 ±0	25 ±35	45 ±31	28 ±21	0 ±0	5 ±7	0 ±0	15 ±21	18 ±21
cube-quadruple-task2	0 ±0	0 ±0	0 ±0	20 ±14	15 ±7	5 ±7	15 ±13	48 ±22	0 ±0	10 ±0	5 ±7	0 ±0	8 ±5
cube-quadruple-task3	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	8 ±15	10 ±12	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0
cube-quadruple-task4	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0
cube-quadruple-task5	0 ±0	0 ±0	0 ±0	0 ±0	5 ±7	0 ±0	5 ±10	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0
Domain average	0 ±0	2 ±3	1 ±1	17 ±1	4 ±3	6 ±8	15 ±8	17 ±8	0 ±0	3 ±1	1 ±1	3 ±4	5 ±3
puzzle-4x4-task1	85 ±21	95 ±7	100 ±0	100 ±0	100 ±0	95 ±7	100 ±0[†]	100 ±0	75 ±7	70 ±28	65 ±7	50 ±14	88 ±10
puzzle-4x4-task2	25 ±35	25 ±7	90 ±14	25 ±21	100 ±0	95 ±7	100 ±0	100 ±0	30 ±14	25 ±7	10 ±14	20 ±0	38 ±19
puzzle-4x4-task3	90 ±14	95 ±7	100 ±0	80 ±14	100 ±0	85 ±21	100 ±0	100 ±0	70 ±14	75 ±21	85 ±7	55 ±7	80 ±22
puzzle-4x4-task4	30 ±28	70 ±0	65 ±7	55 ±7	100 ±0	65 ±50	100 ±0	100 ±0	55 ±7	40 ±28	30 ±14	50 ±14	53 ±25
puzzle-4x4-task5	35 ±7	35 ±21	85 ±7	50 ±14	100 ±0	55 ±35	100 ±0	80 ±18	50 ±0	5 ±7	30 ±14	10 ±14	45 ±13
Domain average	53 ±13	64 ±6	88 ±3	62 ±6	100 ±0	79 ±21	100 ±0[†]	96 ±4	56 ±8	43 ±7	44 ±0	37 ±1	61 ±7
humanoidmaze-giant-task1	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	5 ±7	0 ±0	13 ±10
humanoidmaze-giant-task2	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	15 ±7	0 ±0	0 ±0	0 ±0	55 ±7	30 ±14	25 ±7	53 ±17
humanoidmaze-giant-task3	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	15 ±7	5 ±7	5 ±6
humanoidmaze-giant-task4	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	20 ±14	15 ±7	5 ±7	43 ±22
humanoidmaze-giant-task5	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	5 ±7	0 ±0	0 ±0	0 ±0	90 ±0	45 ±7	75 ±7	88 ±10
Domain average	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	4 ±3	0 ±0	0 ±0	0 ±0	33 ±4	22 ±6	22 ±3	40 ±5
Overall average	13 ±3	33 ±4	35 ±3	31 ±2	39 ±6	39 ±6	43 ±2	48 ±3	14 ±2	28 ±2	24 ±4	25 ±1	43 ±3

Table 3: Complete results for explicit and implicit methods with 25% of the dataset. We highlight nonzero mean performance at or above 95% of the best in each row within each method group in green.

	Explicit methods								*Implicit methods*				
Environment	Floq	FlowCritic	Value Flows	PCBF	C51	IQN	QC	VF (Ours)	DIVL	Implicit C51	Implicit IQN	BFN	IVF (Ours)
cube-triple-task1	15 ±21	90 ±14	95 ±7	55 ±64	80 ±28	85 ±21	98 ±5	98 ±5	0 ±0	100 ±0	85 ±7	95 ±7	98 ±5
cube-triple-task2	0 ±0	60 ±14	95 ±7	35 ±7	65 ±35	80 ±14	53 ±22	93 ±5	0 ±0	80 ±14	90 ±14	60 ±14	80 ±34
cube-triple-task3	0 ±0	70 ±28	95 ±7	35 ±7	85 ±7	80 ±28	73 ±10	80 ±12	0 ±0	65 ±7	75 ±7	65 ±7	90 ±8
cube-triple-task4	0 ±0	45 ±21	20 ±0	10 ±0	55 ±7	60 ±14	58 ±17	80 ±24	0 ±0	20 ±14	20 ±14	0 ±0	63 ±22
cube-triple-task5	0 ±0	35 ±7	85 ±21	10 ±14	45 ±50	65 ±35	58 ±35	98 ±5	0 ±0	55 ±21	75 ±7	75 ±7	100 ±0
Domain average	3 ±4	60 ±3	78 ±6	29 ±7	66 ±6	74 ±3	68 ±11	90 ±3	0 ±0	64 ±3	69 ±4	59 ±7	86 ±7
cube-quadruple-task1	0 ±0	70 ±14	50 ±28	75 ±21	10 ±14	35 ±21	15 ±17	55 ±26	0 ±0	50 ±0	20 ±0	40 ±57	73 ±29
cube-quadruple-task2	0 ±0	45 ±7	20 ±28	35 ±21	55 ±7	40 ±42	3 ±5	43 ±33	0 ±0	5 ±7	35 ±7	5 ±7	18 ±5
cube-quadruple-task3	0 ±0	10 ±14	0 ±0	0 ±0	5 ±7	15 ±21	8 ±10	23 ±13	0 ±0	0 ±0	0 ±0	0 ±0	3 ±5
cube-quadruple-task4	0 ±0	0 ±0	0 ±0	0 ±0	10 ±0	0 ±0	0 ±0	5 ±6	0 ±0	5 ±7	0 ±0	0 ±0	0 ±0
cube-quadruple-task5	0 ±0	0 ±0	0 ±0	0 ±0	20 ±28	5 ±7	13 ±19	0 ±0	0 ±0	0 ±0	10 ±0	0 ±0	5 ±6
Domain average	0 ±0	25 ±1	14 ±11	22 ±0	20 ±8	19 ±1	8 ±6	25 ±9	0 ±0	12 ±0	13 ±1	9 ±13	20 ±5
puzzle-4x4-task1	90 ±14	95 ±7	100 ±0	100 ±0	100 ±0	100 ±0	100 ±0	100 ±0	75 ±21	70 ±14	95 ±7	80 ±14	100 ±0
puzzle-4x4-task2	55 ±7	50 ±28	95 ±7	45 ±21	100 ±0	90 ±14	98 ±5	98 ±5	30 ±0	10 ±14	40 ±14	15 ±7	80 ±22
puzzle-4x4-task3	85 ±7	75 ±7	100 ±0	85 ±7	100 ±0	85 ±21	100 ±0	100 ±0	90 ±14	80 ±14	85 ±7	85 ±7	100 ±0
puzzle-4x4-task4	35 ±7	25 ±7	90 ±14	50 ±14	100 ±0	95 ±7	100 ±0	100 ±0	40 ±28	40 ±0	55 ±7	50 ±14	90 ±8
puzzle-4x4-task5	85 ±21	30 ±28	90 ±14	25 ±7	100 ±0	95 ±7	100 ±0	100 ±0	25 ±7	0 ±0	30 ±14	30 ±14	83 ±17
Domain average	70 ±6	55 ±13	95 ±7	61 ±10	100 ±0	93 ±4	100 ±1	100 ±1	52 ±3	40 ±8	61 ±1	52 ±6	91 ±8
humanoidmaze-giant-task1	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	18 ±22
humanoidmaze-giant-task2	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	25 ±7	0 ±0	0 ±0	25 ±7	85 ±21	60 ±0	60 ±14	58 ±17
humanoidmaze-giant-task3	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	15 ±7	10 ±0	5 ±7	13 ±10
humanoidmaze-giant-task4	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	50 ±14	45 ±7	15 ±21	63 ±10
humanoidmaze-giant-task5	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	15 ±21	0 ±0	0 ±0	5 ±7	90 ±0	85 ±7	95 ±7	98 ±5
Domain average	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	8 ±6	0 ±0	0 ±0	6 ±3	48 ±8	40 ±3	35 ±4	50 ±10
Overall average	18 ±2	35 ±3	47 ±4	28 ±3	47 ±3	49 ±2	44 ±3	54 ±2	15 ±1	41 ±3	46 ±1	39 ±4	61 ±4

Table 4: Complete results for explicit and implicit methods with 50% of the dataset. We highlight nonzero mean performance at or above 95% of the best in each row within each method group in green.

	Explicit methods								*Implicit methods*				
Environment	Floq	FlowCritic	Value Flows	PCBF	C51	IQN	QC	VF (Ours)	DIVL	Implicit C51	Implicit IQN	BFN	IVF (Ours)
cube-triple-task1	30 ±28	75 ±35	90 ±14	55 ±7	90 ±14	100 ±0	98 ±5	100 ±0	0 ±0	100 ±0	100 ±0	100 ±0	100 ±0
cube-triple-task2	0 ±0	70 ±14	55 ±7	40 ±0	80 ±0	95 ±7	50 ±26	93 ±5	0 ±0	85 ±21	80 ±14	85 ±21	93 ±10
cube-triple-task3	0 ±0	55 ±7	95 ±7	40 ±14	90 ±14	65 ±35	78 ±10	83 ±21	0 ±0	75 ±7	95 ±7	95 ±7	90 ±12
cube-triple-task4	0 ±0	35 ±7	70 ±28	15 ±21	30 ±14	80 ±14	70 ±14	70 ±8	0 ±0	60 ±0	60 ±0	55 ±7	68 ±22
cube-triple-task5	0 ±0	75 ±7	0 ±0	10 ±0	95 ±7	0 ±0	60 ±45	95 ±6	0 ±0	100 ±0	75 ±7	85 ±21	100 ±0
Domain average	6 ±6	62 ±6	62 ±11	32 ±8	77 ±1	68 ±8	71 ±12	88 ±5	0 ±0	84 ±3	82 ±6	84 ±3	90 ±5
cube-quadruple-task1	0 ±0	5 ±7	40 ±42	25 ±7	90 ±14	35 ±21	23 ±13	70 ±22	0 ±0	45 ±21	0 ±0	75 ±21	90 ±12
cube-quadruple-task2	0 ±0	5 ±7	35 ±35	5 ±7	10 ±0	60 ±14	8 ±10	30 ±22	0 ±0	15 ±7	0 ±0	15 ±7	35 ±24
cube-quadruple-task3	0 ±0	5 ±7	10 ±14	0 ±0	40 ±28	5 ±7	20 ±18	15 ±13	0 ±0	5 ±7	35 ±35	5 ±7	8 ±15
cube-quadruple-task4	0 ±0	0 ±0	0 ±0	0 ±0	20 ±0	0 ±0	3 ±5	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	3 ±5
cube-quadruple-task5	0 ±0	0 ±0	0 ±0	0 ±0	25 ±35	0 ±0	0 ±0	0 ±0	0 ±0	15 ±7	25 ±7	35 ±7	8 ±10
Domain average	0 ±0	3 ±1	17 ±1	6 ±0	37 ±4	20 ±6	11 ±4	23 ±6	0 ±0	16 ±3	12 ±6	26 ±6	29 ±4
puzzle-4x4-task1	90 ±14	100 ±0	100 ±0	95 ±7	100 ±0	100 ±0	100 ±0	100 ±0	70 ±28	100 ±0	100 ±0	95 ±7	100 ±0
puzzle-4x4-task2	50 ±14	20 ±0	100 ±0	45 ±7	100 ±0	65 ±50	100 ±0	100 ±0	15 ±21	85 ±21	95 ±7	15 ±7	100 ±0
puzzle-4x4-task3	95 ±7	80 ±0	100 ±0	80 ±0	100 ±0	95 ±7	100 ±0	100 ±0	75 ±7	100 ±0	100 ±0	95 ±7	100 ±0
puzzle-4x4-task4	45 ±64	55 ±7	100 ±0	40 ±14	100 ±0	95 ±7	100 ±0	100 ±0	35 ±21	80 ±28	100 ±0	45 ±7	95 ±6
puzzle-4x4-task5	85 ±7	55 ±7	95 ±7	40 ±42	100 ±0	65 ±50	100 ±0	100 ±0	20 ±14	85 ±7	70 ±0	20 ±14	93 ±15
Domain average	73 ±18	62 ±0	99 ±1	60 ±6	100 ±0	84 ±20	100 ±0	100 ±0	43 ±7	90 ±0	93 ±1	54 ±0	98 ±4
humanoidmaze-giant-task1	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	15 ±7	10 ±14	5 ±7	20 ±8
humanoidmaze-giant-task2	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	25 ±35	0 ±0	0 ±0	40 ±14	55 ±21	35 ±7	50 ±0	68 ±15
humanoidmaze-giant-task3	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	5 ±7	25 ±21	15 ±7	10 ±8
humanoidmaze-giant-task4	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	55 ±21	45 ±7	25 ±21	63 ±13
humanoidmaze-giant-task5	0 ±0	0 ±0	0 ±0	5 ±7	40 ±57	50 ±28	0 ±0	0 ±0	0 ±0	80 ±28	75 ±7	100 ±0	83 ±5
Domain average	0 ±0	0 ±0	0 ±0	1 ±1	8 ±11	15 ±1	0 ±0	0 ±0	8 ±3	42 ±3	38 ±3	39 ±1	49 ±7
Overall average	20 ±5	32 ±1	45 ±3	25 ±3	56 ±3	47 ±6	45 ±3	53 ±2	13 ±2	58 ±1	56 ±2	51 ±2	66 ±3

Table 5: Complete results for explicit and implicit methods with 100% of the dataset. We highlight nonzero mean performance at or above 95% of the best in each row within each method group in green.

	Explicit methods								*Implicit methods*				
Environment	Floq	FlowCritic	Value Flows	PCBF	C51	IQN	QC	VF (Ours)	DIVL	Implicit C51	Implicit IQN	BFN	IVF (Ours)
cube-triple-task1	10 ±0	100 ±0	95 ±7	25 ±35	100 ±0	100 ±0	98 ±5	98 ±5	5 ±7	95 ±7	100 ±0	95 ±7	100 ±0
cube-triple-task2	5 ±7	25 ±7	100 ±0	45 ±21	55 ±35	80 ±0	80 ±27	98 ±5	0 ±0	100 ±0	100 ±0	85 ±21	90 ±14
cube-triple-task3	0 ±0	85 ±7	70 ±28	40 ±14	95 ±7	75 ±35	80 ±8	70 ±18	0 ±0	95 ±7	100 ±0	90 ±14	88 ±5
cube-triple-task4	0 ±0	25 ±21	60 ±28	15 ±7	75 ±7	85 ±7	58 ±17	70 ±18	0 ±0	70 ±14	95 ±7	65 ±50	58 ±22
cube-triple-task5	0 ±0	55 ±64	0 ±0	0 ±0	95 ±7	15 ±7	90 ±8	98 ±5	0 ±0	95 ±7	70 ±0	95 ±7	98 ±5
Domain average	3 ±1	58 ±20	65 ±13	25 ±7	84 ±3	71 ±10	81 ±8	87 ±3	1 ±1	91 ±4	93 ±1	86 ±17	87 ±5
cube-quadruple-task1	0 ±0	0 ±0	85 ±7	20 ±28	50 ±57	60 ±57	73 ±17	83 ±22	0 ±0	50 ±28	0 ±0	85 ±21	90 ±8
cube-quadruple-task2	0 ±0	0 ±0	45 ±35	5 ±7	20 ±14	45 ±64	10 ±14	53 ±33	0 ±0	10 ±0	15 ±21	60 ±42	50 ±24
cube-quadruple-task3	0 ±0	0 ±0	40 ±0	5 ±7	30 ±42	0 ±0	45 ±17	40 ±22	0 ±0	10 ±14	25 ±21	20 ±28	25 ±19
cube-quadruple-task4	0 ±0	0 ±0	0 ±0	0 ±0	30 ±14	5 ±7	0 ±0	18 ±22	0 ±0	0 ±0	5 ±7	0 ±0	8 ±10
cube-quadruple-task5	0 ±0	0 ±0	0 ±0	0 ±0	15 ±21	0 ±0	5 ±10	8 ±15	0 ±0	10 ±14	0 ±0	5 ±7	8 ±5
Domain average	0 ±0	0 ±0	34 ±6	6 ±8	29 ±1	22 ±25	27 ±8	40 ±20	0 ±0	16 ±6	9 ±1	34 ±11	36 ±12
puzzle-4x4-task1	90 ±14	100 ±0	100 ±0	80 ±0	100 ±0	80 ±28	100 ±0	100 ±0	80 ±14	100 ±0	100 ±0	100 ±0	100 ±0
puzzle-4x4-task2	35 ±7	10 ±14	95 ±7	40 ±14	100 ±0	70 ±42	100 ±0†	98 ±5	35 ±21	100 ±0	100 ±0	65 ±21	85 ±13
puzzle-4x4-task3	90 ±0	90 ±14	95 ±7	65 ±7	100 ±0	100 ±0	100 ±0	100 ±0	85 ±7	100 ±0	100 ±0	90 ±14	100 ±0
puzzle-4x4-task4	40 ±28	50 ±0	95 ±7	45 ±7	100 ±0	90 ±14	100 ±0	100 ±0	55 ±7	100 ±0	100 ±0	55 ±7	75 ±13
puzzle-4x4-task5	75 ±7	45 ±7	100 ±0	75 ±7	100 ±0	85 ±21	100 ±0	100 ±0	40 ±28	100 ±0	75 ±35	65 ±21	50 ±39
Domain average	66 ±6	59 ±4	97 ±1	61 ±1	100 ±0	85 ±4	100 ±0†	100 ±1	59 ±4	100 ±0	95 ±7	75 ±4	82 ±11
humanoidmaze-giant-task1	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	5 ±7	0 ±0	0 ±0	0 ±0	25 ±21	30 ±0	10 ±0	28 ±26
humanoidmaze-giant-task2	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	10 ±14	0 ±0†	0 ±0	40 ±14	70 ±0	70 ±14	65 ±7	75 ±6
humanoidmaze-giant-task3	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	45 ±7	20 ±14	25 ±7	13 ±10
humanoidmaze-giant-task4	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	25 ±21	60 ±14	50 ±0	50 ±14	70 ±14
humanoidmaze-giant-task5	0 ±0	0 ±0	0 ±0	20 ±14	55 ±64	45 ±64	8 ±15	5 ±10	40 ±57	90 ±14	90 ±0	90 ±14	88 ±5
Domain average	0 ±0	0 ±0	0 ±0	4 ±3	11 ±13	12 ±17	2 ±3†	1 ±2	21 ±10	58 ±8	52 ±6	48 ±0	55 ±6
Overall average	17 ±1	29 ±5	49 ±4	24 ±3	56 ±3	48 ±8	52 ±3	57 ±5	20 ±3	66 ±3	62 ±2	61 ±5	65 ±5

Table 6: Common hyperparameters for VF, IVF and all baselines.

Hyperparameter	Value
Optimizer	Adam (Kingma & Ba, 2014)
Learning rate	3×10^{-4}
Gradient steps	500K
Minibatch size	1024
MLP hidden dimensions	$[1024, 1024, 1024, 1024]$
Nonlinearity	GELU (Hendrycks & Gimpel, 2016)
Layer normalization	Yes
Noise-level embedding	16-dimensional sinusoidal
Noise-level distribution	$\text{UNIF}([0, 1])$
Target network update rate	0.005
Discount factor γ	0.999
Action chunk length H	5
BC flow intergration steps	10 (VF = 4 during training)
Action candidates K	16
Return-flow intergration steps	10

Table 7: Domain-specific hyperparameters.

	Explicit methods						
	Floq			Value Flows		PCBF	IQN
Domain	α	`vel-depth`	κ	λ	λ_w	λ	κ
`cube-triple`	300	2	0.1	0.3	0.3	0.6	300
`cube-quadruple`	300	2	0.1	0.3	0.3	0.6	300
`puzzle-4x4`	3000	4	0.25	3	1	0.4	300
`humanoidmaze-giant`	300	4	0.1	3	0.3	0.2	30

		Implicit methods			
	All methods	DIVL	Implicit C51	Implicit IQN	IVF
Domain	$[r_{\min}, r_{\max}]$	ϱ_{base}	μ	κ	μ
`cube-triple`	$[-3, 0]$	0.9	0.9	300	0.9
`cube-quadruple`	$[-4, 0]$	0.9	0.8	300	0.8
`puzzle-4x4`	$[-16, 0]$	0.9	0.9	300	0.9
`humanoidmaze-giant`	$[-1, 0]$	0.9	0.8	30	0.8