GENERATIVE VALUE CLASSIFIER

Anonymous authors
Paper under double-blind review

ABSTRACT

Q-learning is premised on learning a function to predict the value of a state-action pair, with the goal of using such a function to guide policy improvement. Most approaches learn this function from scratch using scalar or distributional return targets. This discriminative approach leaves another source of supervision for learning representations unused: the actions themselves. Rather than directly predict a return distribution $p(z \mid s, a)$, we ask how likely the state-action pair is under each possible return, $p(s, a \mid z)$. This perspective yields a critic based on Bayes' rule whose generative component is a return-conditioned policy, which can share the actor's architecture. Training this policy supervises the critic with every action rather than with returns alone, and a squared-error loss anchors the critic's Q-value to the bootstrapped target. We evaluate our approach on both offline and online RL tasks and show gains over leading discriminative methods, including distributional RL methods.

1 INTRODUCTION

Value functions play a key role in sample-efficient learning, enabling policy improvement to use past rollouts to better score newer rollouts. However, learning good value functions particularly in sparse-reward settings has remained a significant challenge (Arjona-Medina et al., 2019; Kumar et al., 2020). Standard Monte-Carlo (MC) approaches result in high variance and lack the ability to stitch information across trajectories whereas Temporal-Difference (TD) bootstrapping addresses this problem but at the cost of compounding errors that propagate across timesteps. Sparse-reward settings exacerbate these issues with longer-horizon trajectories and necessitate additional rollouts to provide accurate value estimation.

Prior work has sought to improve value learning under sparse supervision through richer prediction tasks. Distributional RL (Bellemare et al., 2017) learns the return distribution rather than a single scalar. Other approaches use contrastive learning to predict future-state occupancy by contrasting known future states with noisy negatives (Eysenbach et al., 2021; 2022; Mazoure et al., 2023). These methods reformulate value learning through additional representational capacity or inductive biases; we instead ask whether modeling actions can supply additional supervision.

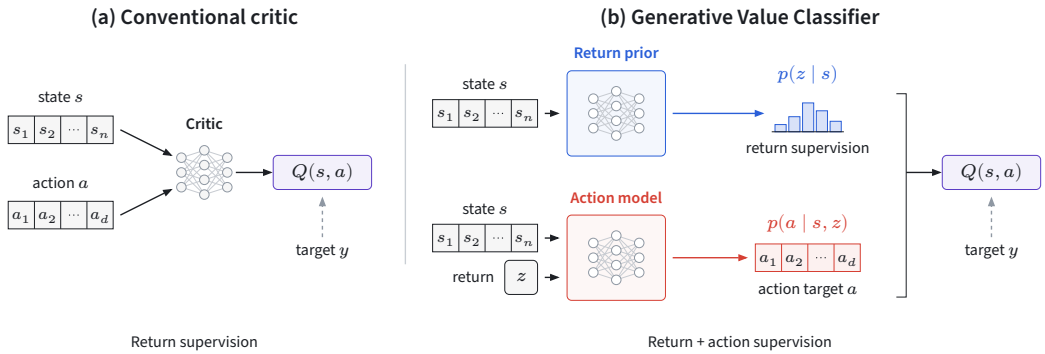


Figure 1: **Action data provides additional supervision.** A conventional critic fits $Q(s, a)$ to a scalar target y. GVC instead learns a return-conditioned action model $p(a \mid s, z)$ and a return prior $p(z \mid s)$, i.e., a distributional state-value function, and combines the two with Bayes' rule to obtain $Q(s, a)$.

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.

Our key insight is that traditional value learning fails to take advantage of a key source of supervision: the state-action distribution itself. Value functions are *conditioned* on state-action pairs, yet are only supervised by a scalar—or at best distributional—return. To use the actions as supervision, we learn $p(a \mid s, z)$ rather than directly predicting $p(z \mid s, a)$, where z denotes the return. Bayes' rule combines this likelihood with a return prior $p(z \mid s)$, the return distribution given the state alone. Normalizing their product over return bins gives a predicted return distribution whose mean is $Q(s, a)$. We also minimize the squared error between $Q(s, a)$ and the bootstrapped target return, which we call *value anchoring*. This uses a discriminative loss to fit the value while retaining the generative supervision and structure.

In summary, our main contributions are:

- We introduce a new formulation for Q-learning that leverages a return-conditioned generative likelihood to implicitly model the return distribution.
- We develop a hybrid generative–discriminative objective that combines return-conditioned flow matching with value anchoring. This aligns the return predictions with the standard Bellman backup targets while retaining the generative supervision.
- We evaluate GVC on challenging continuous control tasks both on online RL benchmarks on RoboMimic and offline and offline-to-online RL benchmarks on OGBench and demonstrate strong performance compared to prior methods. Our analysis demonstrates that removing this generative supervision correlates with lower performance on the same architecture.

2 Related Work

Discriminative Tasks Using Generative Models. Using generative models for discriminative tasks has a long history, with Ng & Jordan (2001) first comparing generative and discriminative classifiers and subsequent work learning a single model with both generative and discriminative objectives (Lasserre et al., 2006). Driven by the strong performance of conditional generative models, recent work has built upon this, demonstrating that pretrained image and text models act as strong zero-shot classifiers, such as with text-to-image models (Li et al., 2023; Clark & Jaini, 2023) or with large language models (Puri & Catanzaro, 2019; Kumar et al., 2024). Finally, another set of works, such as JEM (Grathwohl et al., 2020) and HDGE (Liu & Abbeel, 2020), reinterprets standard discriminative classifiers as energy-based models (EBMs) and adds maximum likelihood or contrastive losses. Our work brings this view to reinforcement learning, where the "class" is a discretized return and the generative model is an action generation policy, thus constructing an implicit Q-function.

Generative Policies as Value Functions. At the same time, both reward modeling and value learning have made significant use of generative policies. In the language domain, reward models are commonly finetuned generative language models used to provide scalar rewards (Ouyang et al., 2022), or to provide text-based responses to grade actions based on a set of criteria (Yuan et al., 2024; Zhang et al., 2025). Similar methods have been applied to robotics, leveraging pretrained *language* policies (Ma et al., 2025; Chen et al., 2026; Lee et al., 2026) for reward or value models, but these methods are unable to judge policy *outputs*, i.e. actions, unlike their language domain counterparts. Finally, a recent line of work on RL for Vision-Language-Action (VLA) models has adopted the pretrained *state* representation from policies as an initialization for value functions (Zhai et al., 2025; Liu et al., 2025). Similar to the language reward models above—though unlike these robotics methods—GVC scores the policy's actions directly. A robot policy (e.g., one that generates joint positions) has no direct way to output a value that is compatible with its pretraining, unlike a language model, which can simply state one in text. This motivates GVC: we instead condition a generative policy on the return and score each action by its likelihood under different returns.

Value Function Learning. A separate line of work maintains standard value-based RL but increases the expressivity of the value function itself. Distributional RL models the full return distribution rather than its mean (Bellemare et al., 2017), with quantile regression (Dabney et al., 2018b) and implicit quantile networks (Dabney et al., 2018a) increasing distributional flexibility. Recent work replaces these heads with stronger generative architectures for added expressivity (Dong et al., 2026b; Agrawalla et al., 2026), with Value Flows (Dong et al., 2026b) modeling the full return distribution

with flow matching. Another set of works instead uses a transformer backbone with self-attention over state-action tokens in place of an MLP: Q-Transformer (Chebotar et al., 2023) exploits this to causally predict Q-values across action dimensions and timesteps, while PAC (Springenberg et al., 2024) scales actor-critic training to large models; however, naively scaling such transformer critics induces attention entropy collapse, which TQL (Dong et al., 2026a) addresses by regularizing attention entropy toward a target value. This differs from sequence-modeling methods like Decision Transformer (Chen et al., 2021), which learn value-like representations implicitly via return-conditioned prediction rather than Bellman backups. These methods differ in how they represent the value and in the network that computes it, but all of them predict a return from a state–action pair. GVC works in the opposite direction: it models the action given the state and a return bin, and recovers $Q(s, a)$ with Bayes' rule. Decision Transformer (Chen et al., 2021) also models actions given a return, but it learns from MC returns-to-go and acts by conditioning on a desired return, whereas GVC learns from bootstrapped Bellman targets and uses its action model to score candidate actions.

3 PRELIMINARIES

Reinforcement learning. We consider a discounted Markov decision process (MDP) specified by the tuple $\{\mathcal{S}, \mathcal{A}, P, r, \rho_0, \gamma\}$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space, $P(s' \mid s, a)$ the state-transition distribution, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the one-step reward function, $\rho_0(s)$ the distribution over initial states, and $\gamma \in [0, 1)$ the discount factor. A policy $\pi(a \mid s)$ induces the discounted return $G_t = \sum_{k\geq 0} \gamma^k r(s_{t+k}, a_{t+k})$, and the goal of RL is to find a policy that maximizes $\mathbb{E}_\pi [G_0]$. The associated action-value function is $Q^\pi(s, a) = \mathbb{E}_\pi [G_t \mid s_t = s, a_t = a]$. Distributional RL models the conditional distribution of the return G_t, rather than its expectation $Q^\pi(s, a)$. We use a categorical approximation with support $\mathcal{Z} = \{z_1, \ldots, z_K\}$ and we refer to each support point z_i as a return bin. For $z \in \mathcal{Z}$, $p(z \mid s, a)$ denotes the probability mass at z.

In this paper, we study how to learn better value functions, asking whether the state-action distribution itself rather than only the scalar return can serve as supervision for Q, and how this affects downstream control performance. We focus on off-policy RL, where the Q-function estimates the discounted return of the policy given a state and action, and is learned from a replay buffer $\mathcal{D}$ of one-step transitions (s, a, r, s', m), with $m \in \{0, 1\}$ masking bootstrapping at terminal transitions. The critic is trained with TD learning:

$$\mathcal{L} = \mathbb{E}_{(s,a,r,s',m)\sim\mathcal{D}} \left[\left(r + \gamma\, m\, \bar{Q}(s', \tilde{a}') - Q(s, a) \right)^2 \right], \tag{1}$$

where $\bar{Q}$ is a frozen target copy of the critic Q and $\tilde{a}' \sim \pi(\cdot \mid s')$ is the next action selected by the RL policy.

Flow matching. A conditional flow transforms a simple base variable $x_0 \sim \mathcal{N}(0, I)$ into data $x_1 \sim p_{\text{data}}(\cdot \mid c)$. With flow time $\lambda \sim \mathcal{U}[0, 1]$, we use the straight path

$$x_\lambda = (1 - \lambda)x_0 + \lambda x_1, \qquad u_\lambda = x_1 - x_0, \tag{2}$$

and train a vector field $v_\theta(c, x_\lambda, \lambda)$ with

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}\Big[\|v_\theta(c, x_\lambda, \lambda) - u_\lambda\|_2^2 \Big]. \tag{3}$$

Sampling integrates $\dot{x}_\lambda = v_\theta(c, x_\lambda, \lambda)$ from Gaussian noise at $\lambda = 0$ to data at $\lambda = 1$.

Generative Flow Classifier. A class-conditional flow can also classify a fixed observation x. For each class c, we noise x along the same path and estimate its flow residual

$$E_\theta(x, c) = \frac{1}{|\mathcal{B}|} \sum_{(\epsilon,\lambda)\in\mathcal{B}} \frac{1}{D} \|v_\theta(c, (1 - \lambda)\epsilon + \lambda x, \lambda) - (x - \epsilon)\|_2^2 . \tag{4}$$

This term is a likelihood-like energy and not an exact normalizing-flow density. Reusing the same noise–time pairs $\mathcal{B}$ for every class gives the proxy, with inverse temperature $\beta > 0$:

$$\widetilde{p}_\theta(c \mid x) = \text{softmax}_c(-\beta E_\theta(x, c) + \log p(c)) . \tag{5}$$

Given a uniform class prior, the classifier selects the class with the smallest noising error. This is the flow-matching analogue of diffusion classifiers, which compare denoising errors across conditions (Li et al., 2023; Clark & Jaini, 2023).

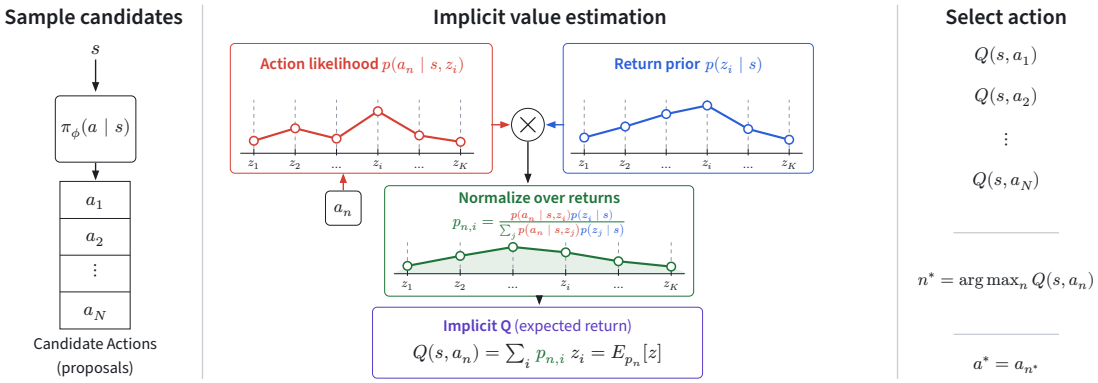


Figure 2: **Generative Value Classifier at inference time.** The actor proposes candidate actions (left). For each action, we compute the return-conditioned action likelihood, along with the return prior. Normalizing their product over return bins (center) provides a return distribution whose mean gives $Q(s, a)$, which we use to rank the action candidates (right).

4 Generative Value Classifier

Overview. GVC is an implicit generative critic: rather than predicting the return of a state–action pair directly, it models the likelihood of an action under a set of discretized returns and inverts this relationship with Bayes' rule. Conceptually, this changes the question from "What is the expected return given this state–action pair?" to "How likely is this action given this state and return?"

The critic consists of two models: $p_\theta(a \mid s, z)$ assigns a likelihood to a given action under each return bin, and $p_\psi(z \mid s)$ predicts the return distribution from the state alone. The latter is the *return prior*: a distributional state-value function. For each return bin, we multiply the action likelihood by its prior probability, normalize across bins, and take the mean to obtain $Q(s, a)$.

A separate flow-matching actor, trained by behavior cloning on dataset or replay actions, proposes candidate actions (Section 4.3). The critic scores these candidates, and we execute the one with the highest $Q(s, a)$. Algorithm 1 summarizes scoring, training, and action selection.

4.1 From discriminative values to a factorized generative critic

A conventional distributional critic directly models $p(z \mid s, a)$ over $\mathcal{Z}$. Our generative classifier starts from the equivalent joint model $p(s, a \mid z)p(z)$ and applies Bayes' rule:

$$p(z \mid s, a) = \frac{p(s, a \mid z)p(z)}{p(s, a)} = \frac{p(a \mid s, z)p(s \mid z)p(z)}{p(a \mid s)p(s)} \tag{6}$$

$$= \frac{p(a \mid s, z)}{p(a \mid s)} \cdot \frac{p(s \mid z)p(z)}{p(s)} = \frac{p(a \mid s, z)p(z \mid s)}{p(a \mid s)} \propto p(a \mid s, z)p(z \mid s), \tag{7}$$

where the first line expands $p(s, a \mid z)$ and $p(s, a)$ with the chain rule and the second applies Bayes' rule again, using $p(s \mid z)p(z)/p(s) = p(z \mid s)$. The denominator in (7) is constant w.r.t. z, so normalizing $p(a \mid s, z)\, p(z \mid s)$ over z recovers $p(z \mid s, a)$. This factorization avoids generating the state: it requires only the action model $p_\theta(a \mid s, z)$ and the return prior $p_\psi(z \mid s)$. We choose this factorization as it empirically outperforms the equivalent joint model.

Let p and Q denote the learned return posterior and its mean under (θ, ψ), with target copies $\bar{p}, \bar{Q}$:

$$p(z_i \mid s, a) = \frac{p_\theta(a \mid s, z_i)p_\psi(z_i \mid s)}{\sum_j p_\theta(a \mid s, z_j)p_\psi(z_j \mid s)}, \qquad Q(s, a) = \sum_i p(z_i \mid s, a)z_i. \tag{8}$$

To evaluate Eq. (8), for each bin z_i we take $p_\psi(z_i \mid s)$ from the prior, estimate $p_\theta(a \mid s, z_i)$ from the flow-matching error of a conditioned on (s, z_i), and then normalize over bins. We first introduce this flow-matching surrogate, then the objectives that train both models and align the resulting return.

Algorithm 1. Generative Value Classifier

function $\textsc{Score}_{\theta,\psi}(s, a)$
  $\epsilon \sim \mathcal{N}(0, I_d)$, $\lambda \sim \mathcal{U}[0, 1]$ ▷ *One draw for clarity; shared across bins*
  $p_i \leftarrow \mathrm{softmax}_i[\beta_{\text{ret}} \log p_\theta(a \mid s, z_i) + \beta_{\text{prior}} \log p_\psi(z_i \mid s)]$ ▷ *NLL surrogate from Eq. (9)*
  return $(p, \sum_i p_i z_i)$ ▷ $p(z \mid s, a)$ *and* $Q(s, a)$

function $\textsc{Train}((s, a, r, s', m) \sim \mathcal{D})$ ▷ $\mathcal{D}$*: offline dataset or replay buffer*
  Sample a' given s'. ▷ *e.g., Best-of-N sampling using policy* ϕ
  $(p, Q) \leftarrow \textsc{Score}_{\theta,\psi}(s, a)$; $(p', Q') \leftarrow \textsc{Score}_{\bar{\theta},\bar{\psi}}(s', a')$
  $\widehat{p} \leftarrow \mathrm{sg}[\Pi_{\mathcal{Z}}(r + \gamma m Z')]$; $y \leftarrow \mathrm{sg}(\sum_i \widehat{p}_i z_i)$ ▷ $\Pr(Z' = z_i) = p'_i$
  $\mathcal{L}_{\text{flow}} \leftarrow -\sum_i \widehat{p}_i \log p_\theta(a \mid s, z_i)$ ▷ *Critic: Conditional Flow loss*
  $\mathcal{L}_{\text{return}} \leftarrow -\sum_i \widehat{p}_i \log p_\psi(z_i \mid s)$ ▷ *Critic: Categorical Value loss*
  $\mathcal{L}_{\text{anchor}} \leftarrow (Q - y)^2$ ▷ *Critic: Value Anchoring loss*
  $\mathcal{L}_{\text{actor}} \leftarrow -\log p_\phi(a \mid s)$ ▷ *Actor: BC Flow loss; Eq. (14)*
  Minimize $\mathcal{L} = \mathcal{L}_{\text{actor}} + \lambda_F \mathcal{L}_{\text{flow}} + \lambda_R \mathcal{L}_{\text{return}} + \lambda_V \mathcal{L}_{\text{anchor}}$.
  Update target copies $(\bar{\theta}, \bar{\psi})$ with EMA.

function $\textsc{Act}(s)$
  Integrate f_ϕ given N seeds, $\epsilon \sim \mathcal{N}(0, I_d)$, to obtain $\mathcal{A}_N$.
  return $\arg\max_{a \in \mathcal{A}_N} Q(s, a)$, with Q from $\textsc{Score}_{\theta,\psi}(s, a)$. ▷ *Execute the best action*

4.2 Flow likelihood

We use the conditional flow-matching error as a surrogate for the negative log-likelihood (Kingma & Gao, 2023; McAllister et al., 2026), writing its negative as $\log p_\theta(a \mid s, z_i)$ below. The vector field $g_\theta(s, x, \lambda, z_i)$ is evaluated for every return bin in a single forward pass: a shared trunk encodes (s, x, λ), and each bin has its corresponding learned embedding added to this before a shared head.

$$-\log p_\theta(a \mid s, z_i) = \mathbb{E}_{\epsilon,\lambda}\left[\frac{1}{d}\left\|g_\theta(s, (1-\lambda)\epsilon + \lambda a, \lambda, z_i) - (a - \epsilon)\right\|_2^2\right]. \tag{9}$$

Substituting this surrogate into Eq. (8), with an exponent on each factor, gives

$$p(z_i \mid s, a) \propto p_\theta(a \mid s, z_i)^{\beta_{\text{ret}}}\, p_\psi(z_i \mid s)^{\beta_{\text{prior}}}, \tag{10}$$

where β_{ret} and β_{prior} are hyperparameters that weight the two factors; both would be 1 for exact likelihoods. Because the flow error is only a likelihood-like energy, we tune them instead and use a value anchor (Section 4.3) to fit the mean $Q(s, a)$ to the TD target, since the mean is what ranks actions. Algorithm 1 shows a single noise–time draw for clarity; when scoring, we average over multiple stratified flow timesteps with antithetic noise, shared across return bins and action candidates to reduce variance (Li et al., 2023; Chen et al., 2024a).

4.3 Objectives

Critic objectives. For each transition, we choose a next action a', for example via best-of-N sampling. Let $\widehat{p}$ be the distribution of $r + \gamma Z'$ projected onto $\mathcal{Z}$, where $Z' \sim \bar{p}(\cdot \mid s', a')$. The projection clips shifted atoms to $[z_1, z_K]$ (Appendix C.1).

We use this distribution as a soft target, weighting each bin by $\widehat{p}_i$ to supervise both the return-conditioned generative loss and the return prior:

$$\mathcal{L}_{\text{flow}} = \mathbb{E}_{\mathcal{D}}\left[-\sum_i \widehat{p}_i \log p_\theta(a \mid s, z_i)\right], \tag{11}$$

$$\mathcal{L}_{\text{return}} = \mathbb{E}_{\mathcal{D}}\left[-\sum_i \widehat{p}_i \log p_\psi(z_i \mid s)\right], \tag{12}$$

where $\mathcal{L}_{\text{flow}}$ is just the flow-matching loss of Eq. (9) on the transition's action a (a fresh noise–time sample per transition), with each return bin weighted by its TD-target probability $\widehat{p}_i$, and $\mathcal{L}_{\text{return}}$ fits

the prior to the same target. The flow error is only a surrogate, with unknown scale and bin-dependent offsets, so these losses alone do not ensure that $Q(s, a)$ matches the target mean $y = \sum_i \widehat{p}_i z_i$. To penalize the value error directly, we add a squared-error value anchor:

$$\mathcal{L}_{\text{anchor}} = \mathbb{E}_{\mathcal{D}}\left[\left(Q(s, a) - \text{sg}(y)\right)^2\right]. \tag{13}$$

The value anchor constrains the return which is used to rank actions.

Actor Objective. GVC is independent of the policy extraction mechanism and can work with both implicit and explicit extraction methods. For all experiments, except the ablation in Appendix A.3, we use best-of-N sampling for implicit extraction alongside a flow-matching behavior policy trained on replay actions, both to act and to select a' for the TD target; Appendix C.1 lists N for each setting.

The actor is a behavior-cloning flow with vector field f_ϕ. Its loss is $-\mathbb{E}_{\mathcal{D}} \log p_\phi(a \mid s)$, using the flow-matching surrogate

$$-\log p_\phi(a \mid s) = \mathbb{E}_{\epsilon,\lambda}\left[\frac{1}{d}\left\|f_\phi(s, (1-\lambda)\epsilon + \lambda a, \lambda) - (a - \epsilon)\right\|_2^2\right]. \tag{14}$$

Complete Objective. The complete update minimizes

$$\mathcal{L} = \mathcal{L}_{\text{actor}} + \lambda_F \mathcal{L}_{\text{flow}} + \lambda_R \mathcal{L}_{\text{return}} + \lambda_V \mathcal{L}_{\text{anchor}}. \tag{15}$$

At evaluation, we sample N distinct actions from the actor, score them using Eq. (10), execute the action with the largest $Q(s, a)$, and replan at the next environment step.

5 Experiments

The goal of our experiments is to answer the following key questions:

- **(Q1)** Does replacing a discriminative critic with GVC improve online RL when the actor and training recipe are held fixed?
- **(Q2)** Does the gain hold in offline RL and during offline-to-online finetuning, including against distributional critics?
- **(Q3)** Which parts of the GVC objective are responsible for the gains?

Environments. We evaluate GVC in two settings. For online RL, we use four robotic manipulation tasks—Lift, Can, Square, and Tool Hang—from RoboMimic (Mandlekar et al., 2021), where a 7-DoF robot arm must perform diverse manipulation behaviors under sparse task-completion rewards. Lift requires the robot to grasp and lift up an object; Can requires grasping a cylinder and placing it at a specified location; Square involves precisely inserting a square nut onto a peg; Tool Hang is a long-horizon, multi-stage assembly task in which the robot must insert a hook into a base to make a frame and then hang a wrench from it. For offline and offline-to-online RL, we evaluate on OGBench (Park et al., 2025a), which provides offline datasets paired with online finetuning for long-horizon, sparse-reward control. Offline, we use the `cube`, `puzzle`, and sparse-reward `scene` domains listed in Table 1; for offline-to-online finetuning we use the six tasks from Value Flows (Dong et al., 2026b).

Baselines. We organize the baselines based on what they change relative to GVC. BFN (Li et al., 2025) keeps the flow-matching actor and the best-of-N selection of GVC and swaps the critic for a regression head, so the two differ only in the critic. IDQL (Hansen-Estruch et al., 2023) and its flow-policy variant IFQL (Park et al., 2025b) also select actions by best-of-N from a behavior-cloned generative policy but fit their critic by expectile regression, without querying the policy. C51 (Bellemare et al., 2017), IQN (Dabney et al., 2018a), CODAC (Ma et al., 2021), and Value Flows (Dong et al., 2026b) keep a discriminative head but model the full return distribution, with categorical, quantile, conservative quantile, and flow-matching parameterizations respectively; they test whether distributional targets alone explain our gains. FQL (Park et al., 2025b), QSM (Psenka et al., 2024), and QAM (Li & Levine, 2026) instead keep a standard critic and train an explicit flow or diffusion policy against it, by distillation with a BC penalty, score matching, or adjoint

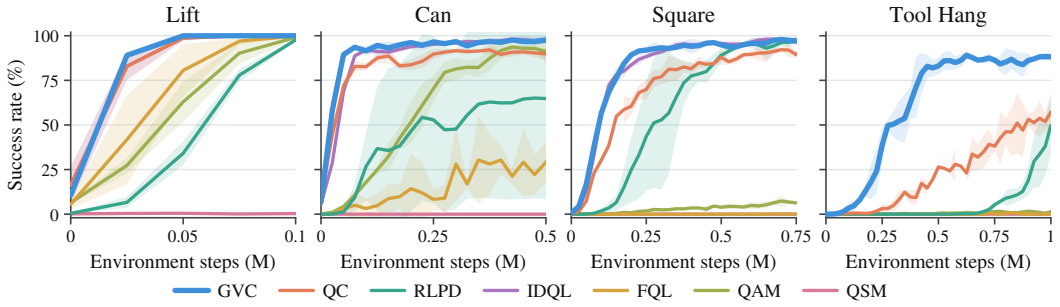


Figure 3: **Online RL on RoboMimic.** Success rate versus environment steps (in millions) on Lift, Can, Square, and Tool Hang, from left to right. We compare GVC with QC, RLPD, IDQL, FQL, QAM, and QSM. Shaded regions show one standard deviation across seeds.

matching, while QC (Li et al., 2025) runs TD learning over action chunks to shorten the effective horizon. Finally, we include RLPD (Ball et al., 2023), IQL (Kostrikov et al., 2022), and, offline only, BC, ReBRAC (Tarasov et al., 2023), and FBRAC (Wu et al., 2019; Park et al., 2025b) as standard references; offline baseline numbers are taken from Value Flows or reproduced directly where unavailable (Appendix C.3). RLPD and FQL are also run offline-to-online, initialized from the offline dataset and finetuned online.

Implementation details. To fairly compare GVC to prior methods, we follow the same training recipe as QC (Li et al., 2025) for the RoboMimic online experiments and Value Flows (Dong et al., 2026b) for the OGBench offline and offline-to-online experiments. We utilize the same BFN actor architecture, only modifying the critic architecture to implement our method. Appendix C.1 lists the GVC hyperparameters for each setting, Appendix C.2 the training and evaluation protocol, and Appendix C.3 the baseline tuning.

5.1 Online RL

Figure 3 shows learning curves on the four RoboMimic tasks. Lift, Can, and Square are eventually solved by GVC, QC, and IDQL, although GVC demonstrates small sample efficiency gains over all curves. Tool Hang is where the critic methods diverge, with GVC reaching roughly 85% success within 0.5M steps whereas QC, whose recipe GVC shares apart from the critic, and RLPD only reach about 50% at 1M steps. Meanwhile, IDQL, which uses the same best-of-N selection but a regression critic, does not learn the task.

5.2 Offline RL

Table 1 reports per-task success on 35 tasks across seven OGBench domains. On the two largest puzzles, `puzzle-4x5` and `puzzle-4x6`, GVC reaches 91% and 58% on the first task, compared to BFN—the closest comparison, differing from GVC only in the critic—which achieves 83% and 22%, respectively. The two tie on `cube-double` at 66% and are both near ceiling on the sparse `puzzle-3x3` tasks; GVC leads by 12 points on the sparse `scene` tasks (98% versus 86%), mostly on `task5` where BFN reaches 45%, and by 4 on `cube-triple` (10% versus 6%). IFQL, which also selects by best-of-N but trains its critic by expectile regression, is further behind: GVC is ahead on six of the seven domains. GVC also improves over the leading distributional baseline, Value Flows, on `puzzle-4x4`, `puzzle-4x5`, and `puzzle-4x6` by an average of 9 points, while trailing slightly on `cube-double` and `cube-triple` by an average of 3 points.

Table 1: **Offline evaluation on OGBench.** Success rate (%) per task. We bold values within 95% of the best performance for each task.

		Gaussian Policies			Flow Policies								
Environment	Task	BC	IQL	ReBRAC	FBRAC	IFQL	FQL	C51	IQN	CODAC	Value Flows	BFN	GVC
scene-sparse	task1	19 ±6	0 ±0	9 ±8	**100** ±1	**96** ±3	65 ±16	19 ±6	**99** ±2	**100** ±1	**100** ±1	**100** ±1	**99** ±1
	task2	1 ±1	0 ±0	66 ±19	86 ±10	83 ±5	52 ±12	2 ±2	**98** ±2	**98** ±3	**99** ±1	**100** ±1	**96** ±3
	task3	1 ±1	0 ±0	0 ±0	**94** ±8	56 ±8	60 ±2	2 ±2	**93** ±6	72 ±38	89 ±5	**94** ±6	**96** ±2
	task4	2 ±2	0 ±0	0 ±0	77 ±34	72 ±12	69 ±16	4 ±4	82 ±15	76 ±30	92 ±10	90 ±4	**99** ±1
	task5	0 ±0	0 ±0	0 ±0	75 ±19	52 ±8	19 ±5	1 ±2	57 ±30	66 ±20	91 ±7	45 ±14	**98** ±2
	avg.	5	0	15	86	72	53	6	86	82	**94**	86	**98**
puzzle3x3-sparse	task1	5 ±2	1 ±1	**98** ±2	75 ±19	**99** ±1	**100** ±0	6 ±3	37 ±29	5 ±4	78 ±15	**99** ±1	**100** ±0
	task2	1 ±1	0 ±0	66 ±15	35 ±37	**99** ±2	**100** ±0	3 ±3	13 ±18	1 ±1	5 ±7	**97** ±5	**99** ±1
	task3	1 ±1	0 ±0	25 ±16	21 ±22	**98** ±2	**97** ±4	0 ±1	41 ±48	1 ±1	10 ±5	**95** ±5	**98** ±1
	task4	1 ±1	0 ±0	55 ±13	41 ±34	**98** ±2	**98** ±2	0 ±1	2 ±2	1 ±2	8 ±6	**98** ±3	**94** ±4
	task5	1 ±0	0 ±0	68 ±18	43 ±40	**97** ±4	**100** ±0	1 ±1	1 ±2	0 ±1	4 ±4	**96** ±2	**99** ±1
	avg.	2	0	62	43	**98**	**99**	2	19	2	21	**97**	**98**
puzzle4x4	task1	1 ±1	12 ±2	26 ±4	32 ±9	49 ±9	34 ±8	0 ±0	41 ±2	37 ±32	36 ±3	**62** ±10	**64** ±4
	task2	0 ±0	7 ±4	12 ±4	5 ±3	4 ±4	16 ±5	0 ±0	12 ±4	10 ±10	**27** ±5	20 ±7	23 ±8
	task3	0 ±0	9 ±3	15 ±3	20 ±10	50 ±14	18 ±5	0 ±0	45 ±7	33 ±29	30 ±4	41 ±8	**61** ±8
	task4	0 ±0	5 ±2	10 ±3	5 ±1	21 ±11	11 ±3	0 ±0	23 ±2	12 ±10	28 ±5	21 ±6	**38** ±6
	task5	0 ±0	4 ±1	7 ±3	2 ±2	2 ±2	7 ±3	0 ±0	16 ±6	10 ±8	13 ±2	14 ±6	**24** ±7
	avg.	0	7	14	13	25	17	0	27	20	27	32	**42**
puzzle4x5	task1	0 ±0	14 ±7	0 ±1	32 ±30	86 ±8	53 ±19	0 ±0	**90** ±5	51 ±10	67 ±4	83 ±4	**91** ±4
	task2	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±1	**1** ±2	0 ±1	1 ±1	0 ±0
	task3	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	**0** ±1	**0** ±1	0 ±0	0 ±0
	task4	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0**	**0** ±0
	task5	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0
	avg.	0	3	0	6	17	11	0	**18**	10	13	17	**18**
puzzle4x6	task1	0 ±0	2 ±3	8 ±7	16 ±20	0 ±0	22 ±5	0 ±0	30 ±14	10 ±11	22 ±25	22 ±8	**58** ±15
	task2	0 ±0	0 ±1	0 ±1	0 ±1	0 ±0	0 ±1	0 ±0	0 ±1	**2** ±2	0 ±1	1 ±1	1 ±1
	task3	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0
	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
	task5	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0** ±0	**0**	**0** ±0
	avg.	0	0	2	3	0	4	0	6	2	4	5	**12**
cube-double	task1	8 ±3	27 ±5	45 ±6	47 ±11	35 ±9	61 ±9	9 ±0	70 ±14	80 ±11	**97** ±1	73 ±4	88 ±21
	task2	0 ±0	1 ±1	7 ±3	22 ±12	9 ±5	36 ±6	0 ±0	24 ±9	63 ±4	**76** ±7	**78** ±4	72 ±12
	task3	0 ±0	0 ±0	4 ±1	4 ±2	8 ±5	22 ±5	0 ±0	25 ±6	66 ±9	73 ±4	**77** ±7	**74** ±26
	task4	0 ±0	0 ±0	1 ±1	0 ±1	1 ±1	5 ±2	0 ±0	10 ±1	13 ±2	**30** ±5	23 ±6	19 ±6
	task5	0 ±0	4 ±3	4 ±2	2 ±2	17 ±6	19 ±10	0 ±0	**81** ±8	**82** ±4	69 ±5	**79** ±6	75 ±19
	avg.	2	6	12	15	14	29	2	42	61	**69**	**66**	**66**
cube-triple	task1	1 ±1	4 ±4	1 ±2	0 ±0	2 ±2	20 ±6	1 ±0	29 ±2	9 ±5	**59** ±12	28 ±13	49 ±30
	task2	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	**1** ±2	0 ±0	0 ±0	**1** ±0	0 ±0	1 ±1	1 ±1
	task3	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	1 ±0	0 ±0	**7** ±3	2 ±0	2 ±3
	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
	task5	0 ±0	1 ±1	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	**2** ±1	0 ±0	0 ±0
	avg.	0	1	0	0	0	4	0	6	2	**14**	6	10

5.3 Offline-to-online RL

Figure 4 and Table 2 show offline-to-online RL performance of GVC on six OGBench tasks. GVC has the highest average success at the end of both phases and finishes within 95% of the best method on all six tasks. Its average success rises from 67.1% to 98.8%, about 18 points above Value Flows and 26 above FQL in Table 2. BFN finishes second at 88.7% but scores 45.4% after offline training, below Value Flows. It edges ahead of GVC on `antmaze-large` but reaches only 90% on the two cube tasks and 89% on `humanoidmaze-medium`, and two of its eight seeds collapse on `puzzle-4x4`. RLPD, which learns only online, reaches 99–100% on three tasks but stays near zero on the other three.

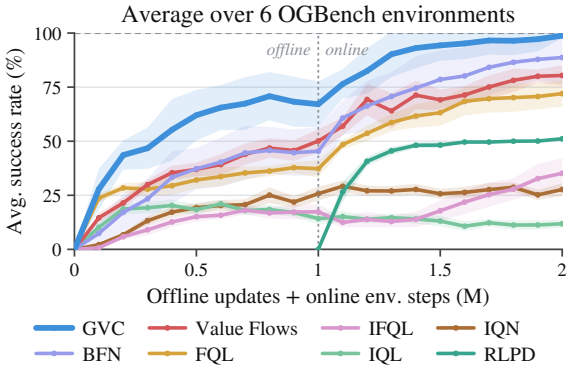


Figure 4: **Offline-to-online RL on OGBench.** Average success rate (%) across six OGBench environments. The first 1e6 steps are offline, followed by online finetuning for 1e6 steps. We compare GVC with BFN, Value Flows, FQL, IFQL, IQL, IQN, and RLPD.

Table 2: **Offline-to-online evaluations on OGBench tasks.** Success rate (%) on 6 OGBench tasks after offline training → after online finetuning. Mean ± standard deviation over 8 seeds, with values within 95% of the best final performance for each task in bold.

Task	IQN	IFQL	FQL	RLPD	IQL	Value Flows	BFN	GVC
antmaze-large	55 ±5 → 91 ±1	11 ±3 → 3 ±2	**87** ±7 → **100** ±1	0 ±0 → **99** ±2	41 ±5 → 45 ±5	**72** ±4 → **96** ±1	**66** ±19 → **99** ±1	**79** ±4 → **97** ±2
humanoidmaze-medium	23 ±3 → 14 ±2	56 ±12 → 82 ±7	12 ±7 → 22 ±12	0 ±0 → 8 ±3	21 ±5 → 16 ±3	29 ±6 → 86 ±3	32 ±8 → 89 ±4	**96** ±2 → **100** ±0
cube-double	29 ±4 → 42 ±7	12 ±3 → 41 ±2	40 ±11 → 92 ±3	0 ±0 → 0 ±0	0 ±0 → 0 ±0	65 ±7 → 79 ±6	79 ±7 → 90 ±7	**71** ±19 → **97** ±2
cube-triple	25 ±5 → 14 ±6	2 ±1 → 7 ±5	4 ±1 → 83 ±12	0 ±0 → 0 ±0	3 ±1 → 0 ±0	29 ±8 → 70 ±7	21 ±10 → 90 ±17	**40** ±19 → **100** ±0
puzzle-4x4	22 ±2 → 6 ±1	23 ±2 → 19 ±12	8 ±3 → 38 ±52	**0** ±0 → **100** ±0	5 ±1 → 0 ±0	14 ±3 → 51 ±12	24 ±6 → 68 ±39	**30** ±5 → **99** ±3
scene	0 ±0 → 0 ±0	0 ±0 → 60 ±14	**82** ±11 → **100** ±1	**0** ±0 → **100** ±0	15 ±4 → 10 ±3	**92** ±23 → **100** ±0	**51** ±16 → **97** ±4	**87** ±15 → **100** ±0

5.4 Ablations

Our ablations look at the generative loss, which distinguishes GVC from a discriminative critic, as well as the impact of timestep sampling to approximate the action likelihood. First, Figure 5a varies the coefficient λ_F on $\mathcal{L}_{\text{flow}}$ from its default of 1 down to 0. On Tool Hang, we find that the ordering is monotone in λ_F, and success drops from about 85% at $\lambda_F = 1$ to 80% at 0.1, 65% at 0.01, and 45% either at 0.001 or with the loss removed. Next, Figure 5b looks at which part of the loss matters. On Tool Hang, we find that weighting all return bins uniformly—so that the flow loss trains the action model but no longer links actions to returns—recovers about 70%, between the full loss at 85% and no flow loss at about 50%: roughly half the benefit comes from the generative supervision as such, and the rest from conditioning it on the return. Finally, Figure 5c varies the number of noise timesteps, T, used to score actions, as well as how the noise is sampled across action candidates. With noise shared across candidates, $T = 1$ reaches about 80% on Tool Hang, and using $T \geq 2$ adds a few points; without sharing, $T = 1$ falls to about 60%. Appendix A shows results for the same set of ablations on the easier Square task, as well as ablations over the value-anchor weight λ_V, and a comparison of best-of-N selection with a one-step actor distilled from critic gradients.

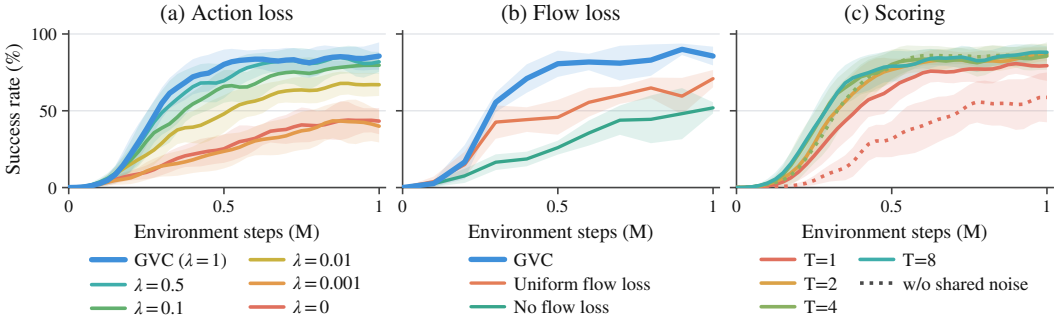


Figure 5: **Ablations on Tool Hang.** Success rate (%) vs. environment steps. (a) Action loss: varying flow loss coefficients, λ_F. (b) Flow loss: GVC where the flow loss is weighted by the target return distribution $\widehat{p}_i$ (blue), uniform weighting across return bins (coral), and the flow loss removed (teal). (c) Scoring: different T values, with and without shared noise. Shaded regions show one standard deviation across seeds.

6 Discussion

We have presented GVC, a method for value learning that leverages a return-conditioned generative likelihood to implicitly model the return distribution. Our work builds off of recent work on generative classifiers and extends it to a domain where sample efficiency is particularly important.

Limitations. GVC increases the theoretical compute costs of the critic, requiring T forward passes to compute the likelihood, over K bins, possibly over ensembled copies. In practice, we reduce this to only one forward pass per bin and find that the compute costs are comparable to those of other performant distributional RL methods (Appendix B), but this remains a consideration. Finally, in the form presented, GVC is factorized into two model components and requires discretizing the return space, both of which increase the implementation complexity.

REPRODUCIBILITY STATEMENT

Our experiments use public datasets and benchmarks—RoboMimic (Mandlekar et al., 2021) for online RL and OGBench (Park et al., 2025a) for offline and offline-to-online RL. We describe reward relabeling and task selection in Appendix C. The GVC method is fully specified in Section 4, with the training objective in Eq. (15) and the complete algorithm in Algorithm 1. Appendix C.2 states the number of seeds, the evaluation schedule, and how the reported metrics are computed. We will release our code and evaluation scripts upon publication.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods, in particular to write and review the code for our method and experiments. We also used generative AI tools to help create the method figures and to review the paper's writing and grammar. We have not used generative AI tools for the rest of the required disclosure tasks. We have reviewed all AI-assisted work.

REFERENCES

Bhavya Agrawalla, Michal Nauman, and Aviral Kumar. What does flow-matching bring to TD-learning? In *International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=e7EKA7Y9Im.

Jose A. Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. RUDDER: Return decomposition for delayed rewards. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 1577–1594. PMLR, 2023. URL https://proceedings.mlr.press/v202/ball23a.html.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 449–458. PMLR, 2017. URL https://proceedings.mlr.press/v70/bellemare17a.html.

Yevgen Chebotar, Quan Vuong, Karol Hausman, Fei Xia, Yao Lu, Alex Irpan, Aviral Kumar, Tianhe Yu, Alexander Herzog, Karl Pertsch, Keerthana Gopalakrishnan, Julian Ibarz, Ofir Nachum, Sumedh Anand Sontakke, Grecia Salazar, Huong T. Tran, Jodilyn Peralta, Clayton Tan, Deeksha Manjunath, Jaspiar Singh, Brianna Zitkovich, Tomas Jackson, Kanishka Rao, Chelsea Finn, and Sergey Levine. Q-Transformer: Scalable offline reinforcement learning via autoregressive Q-functions. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 3909–3928. PMLR, 2023. URL https://proceedings.mlr.press/v229/chebotar23a.html.

Huanran Chen, Yinpeng Dong, Shitong Shao, Zhongkai Hao, Xiao Yang, Hang Su, and Jun Zhu. Diffusion models are certifiably robust classifiers. In *Advances in Neural Information Processing Systems*, volume 37, pp. 50062–50097. Curran Associates, Inc., 2024a. doi: 10.52202/079017-1584. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/59a3444d39b97ba01a17994f938e1ccc-Paper-Conference.pdf.

Huanran Chen, Yinpeng Dong, Zhengyi Wang, Xiao Yang, Chengqi Duan, Hang Su, and Jun Zhu. Robust classification via a single diffusion model. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 6643–6665. PMLR, 2024b. URL https://proceedings.mlr.press/v235/chen24k.html.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems*, volume 34, pp. 15084–15097. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/7f489f642a0ddb10272b5c31057f0663-Abstract.html.

Shirui Chen, Cole Harrison, Ying-Chun Lee, Angela Jin Yang, Zhongzheng Ren, Lillian J. Ratliff, Jiafei Duan, Dieter Fox, and Ranjay Krishna. TOPReward: Token Probabilities as Hidden Zero-Shot Rewards for Robotics, February 2026. URL http://arxiv.org/abs/2602.19313. arXiv:2602.19313 [cs.RO].

Kevin Clark and Priyank Jaini. Text-to-Image Diffusion Models are Zero Shot Classifiers. In *Advances in Neural Information Processing Systems*, volume 36, pp. 58921–58937. Curran Associates, Inc., December 2023. doi: 10.52202/075280-2571. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/b87bdcf963cad3d0b265fcb78ae7d11e-Abstract-Conference.html.

Will Dabney, Georg Ostrovski, David Silver, and Remi Munos. Implicit quantile networks for distributional reinforcement learning. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine*

Learning Research, pp. 1096–1105. PMLR, 10–15 Jul 2018a. URL https://proceedings.mlr.press/v80/dabney18a.html.

Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018b. doi: 10.1609/aaai.v32i1.11791. URL https://doi.org/10.1609/aaai.v32i1.11791.

Perry Dong, Kuo-Han Hung, Alexander Swerdlow, Dorsa Sadigh, and Chelsea Finn. TQL: Scaling Q-functions with transformers by preventing attention collapse. In *International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=lwHSE1xYHH.

Perry Dong, Chongyi Zheng, Chelsea Finn, Dorsa Sadigh, and Benjamin Eysenbach. Value flows. In *International Conference on Learning Representations*, 2026b. URL https://arxiv.org/abs/2510.07650.

Gabriel Dulac-Arnold, Nir Levine, Daniel J. Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021. doi: 10.1007/s10994-021-05961-4. URL https://doi.org/10.1007/s10994-021-05961-4.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. C-Learning: Learning to achieve goals via recursive classification. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=tc5qisoB-C.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Ruslan Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 35603–35620. Curran Associates, Inc., 2022. doi: 10.52202/068431-2580. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/e7663e974c4ee7a2b475a4775201ce1f-Abstract-Conference.html.

Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Hkxzx0NtDB.

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, 2023. URL https://arxiv.org/abs/2304.10573.

Diederik Kingma and Ruiqi Gao. Understanding diffusion objectives as the ELBO with simple data augmentation. In *Advances in Neural Information Processing Systems*, volume 36, pp. 65484–65516. Curran Associates, Inc., 2023. doi: 10.52202/075280-2858. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/ce79fbf9baef726645bc2337abb0ade2-Paper-Conference.pdf.

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, Abhishek Gupta, and Sergey Levine. DisCor: Corrective feedback in reinforcement learning via distribution correction. In *Advances in Neural Information Processing Systems*, volume 33, pp. 18560–18572. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/d7f426ccbc6db7e235c57958c21c5dfa-Abstract.html.

Sachin Kumar, Chan Young Park, and Yulia Tsvetkov. Gen-Z: Generative zero-shot text classification with contextualized label descriptions. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=rkplYfqUr0.

Julia A. Lasserre, Christopher M. Bishop, and Thomas P. Minka. Principled hybrids of generative and discriminative models. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pp. 87–94, 2006. doi: 10.1109/CVPR.

2006.227. URL https://www.microsoft.com/en-us/research/publication/principled-hybrids-of-generative-and-discriminative-models/.

Tony Lee, Andrew Wagenmaker, Karl Pertsch, Percy Liang, Sergey Levine, and Chelsea Finn. RoboReward: General-purpose vision-language reward models for robotics, 2026. URL https://arxiv.org/abs/2601.00675.

Alexander C. Li, Mihir Prabhudesai, Shivam Duggal, Ellis Brown, and Deepak Pathak. Your diffusion model is secretly a zero-shot classifier. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2206–2217. IEEE, 2023. doi: 10.1109/ICCV51070.2023.00210. URL https://doi.org/10.1109/ICCV51070.2023.00210.

Qiyang Li and Sergey Levine. Q-learning with adjoint matching. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=vd4eNAdtO6.

Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement learning with action chunking. In *Advances in Neural Information Processing Systems*, volume 38, pp. 55518–55553, 2025. doi: 10.52202/085713-1856. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/50348e8f9aef984abe0ea1ec2a326f78-Abstract-Conference.html. Results from https://github.com/ColinQiyangLi/qc/blob/48283b4f662bd1c127ec9fa80647b49759f10653/plot_data/ogbench-individual.pkl.

Hao Liu and Pieter Abbeel. Hybrid discriminative-generative training via contrastive learning, 2020. URL https://arxiv.org/abs/2007.09070.

Jijia Liu, Feng Gao, Bingwen Wei, Xinlei Chen, Qingmin Liao, Yi Wu, Chao Yu, and Yu Wang. What can RL bring to VLA generalization? an empirical study. In *Advances in Neural Information Processing Systems*, volume 38, pp. 97121–97151, 2025. doi: 10.52202/085713-3248. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/8c0fabe372177d2aded596be2d3b4544-Abstract-Conference.html.

Yecheng Ma, Dinesh Jayaraman, and Osbert Bastani. Conservative offline distributional reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34, pp. 19235–19247. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/a05d886123a54de3ca4b0985b718fb9b-Abstract.html.

Yecheng Jason Ma, Joey Hejna, Chuyuan Fu, Dhruv Shah, Jacky Liang, Zhuo Xu, Sean Kirmani, Peng Xu, Danny Driess, Ted Xiao, Osbert Bastani, Dinesh Jayaraman, Wenhao Yu, Tingnan Zhang, Dorsa Sadigh, and Fei Xia. Vision language models are in-context value learners. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=friHAl5ofG.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp. 1678–1690. PMLR, 2021. URL https://proceedings.mlr.press/v164/mandlekar22a.html.

Bogdan Mazoure, Benjamin Eysenbach, Ofir Nachum, and Jonathan Tompson. Contrastive value learning: Implicit models for simple offline RL. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 1257–1267. PMLR, 2023. URL https://proceedings.mlr.press/v229/mazoure23a.html.

David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=eoEmoKoQpJ.

Andrew Y. Ng and Michael I. Jordan. On Discriminative vs. Generative Classifiers: A comparison of logistic regression and naive Bayes. In *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001. URL https://papers.nips.cc/paper_files/paper/2001/hash/7b7a53e239400a13bd6be6c91c4f6c4e-Abstract.html.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744. Curran Associates, Inc., 2022. doi: 10.52202/068431-2011. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=M992mjgKzI.

Seohong Park, Qiyang Li, and Sergey Levine. Flow Q-learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48104–48127. PMLR, 2025b. URL https://proceedings.mlr.press/v267/park25f.html.

Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via Q-score matching. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 41163–41182. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/psenka24a.html.

Raul Puri and Bryan Catanzaro. Zero-shot text classification with generative language models, 2019. URL https://arxiv.org/abs/1912.10165.

Jost Tobias Springenberg, Abbas Abdolmaleki, Jingwei Zhang, Oliver Groth, Michael Bloesch, Thomas Lampe, Philemon Brakel, Sarah Maria Elisabeth Bechtle, Steven Kapturowski, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Offline actor-critic reinforcement learning scales to large models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 46323–46350. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/springenberg24a.html.

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 11592–11620. Curran Associates, Inc., 2023. doi: 10.52202/075280-0511. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/26cce1e512793f2072fd27c391e04652-Abstract-Conference.html.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning, 2019. URL https://arxiv.org/abs/1911.11361.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. Self-rewarding language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 57905–57923. PMLR, 2024. URL https://proceedings.mlr.press/v235/yuan24d.html.

Shaopeng Zhai, Qi Zhang, Tianyi Zhang, Fuxian Huang, Haoran Zhang, Ming Zhou, Shengzhe Zhang, Litao Liu, Sixu Lin, and Jiangmiao Pang. A Vision-Language-Action-Critic model for robotic real-world reinforcement learning, 2025. URL https://arxiv.org/abs/2509.15937.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Ccwp4tFEtE.

A Additional Ablations

Section 5 reports our ablations on Tool Hang. This appendix adds the same ablations on Square (Appendix A.1), a sweep over the weight λ_V of the value anchor (Appendix A.2), and a comparison of best-of-N selection with a one-step actor trained by gradient ascent on Q (Appendix A.3). Unless stated otherwise, every run uses the settings of Table 4.

A.1 Ablations on Square

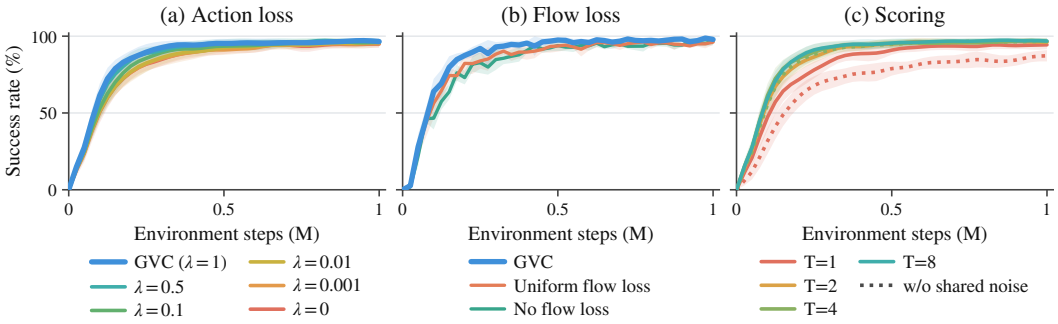


Figure 6: **Ablations on Square.** Success rate (%) versus environment steps. (a) Action loss: different flow-loss coefficients λ_F, compared with the default. (b) Flow loss: `GVC` with the default flow loss weighted by the target return distribution $\widehat{p}_i$ (blue), uniform weighting across return bins (coral), and the flow loss removed (teal). (c) Scoring: different T values, with and without shared noise. Shaded regions show one standard deviation across seeds.

Figure 6 repeats the three ablations of Figure 5 on Square, with the same settings and 8 seeds per variant. Square is the easier task and separates the variants far less. Every flow-loss coefficient in (a) and every flow-loss variant in (b), including removing the loss, ends near 95%; the default learns slightly faster early on, but the curves otherwise nearly coincide. Scoring (c) tells the same story as on Tool Hang. With shared noise, $T = 1$ learns a little more slowly than $T \geq 2$ but catches up; without sharing, $T = 1$ learns more slowly still and ends several points lower.

A.2 Value-anchor coefficient

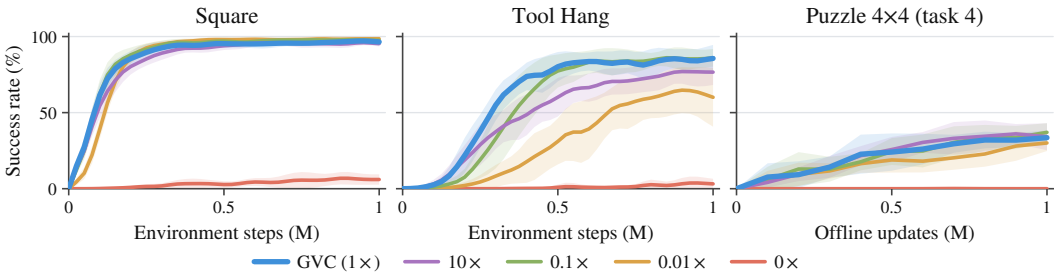


Figure 7: **Value-anchor ablation on Square, Tool Hang, and `puzzle-4x4`.** Success rate versus environment steps, or offline updates on `puzzle-4x4` (`task4`), with the anchor coefficient λ_V scaled to 10, 0.1, 0.01, and 0 times its default (1×, Table 4), 8 seeds per setting.

Value anchoring (Eq. (13)) fits $Q(s, a)$—the mean of the return distribution—to the TD target. Figure 7 varies its weight λ_V on Square, Tool Hang, and offline `puzzle-4x4`, with everything else as in Table 4. Without the anchor, success stays below 10% on all three tasks. With it, Square is insensitive to the weight, and any non-zero value matches the default. Tool Hang is more sensitive but tolerates a wide range: 0.1× ends near the default 1× (88% vs. 86%), 10× learns more slowly and ends at 77%, and 0.01× is slower still (57%). On `puzzle-4x4` the weight matters little once it is non-zero, and all four settings end between 30% and 38% after 10^6 updates.

A.3 BEST-OF-N SELECTION VERSUS A DISTILLED ACTOR

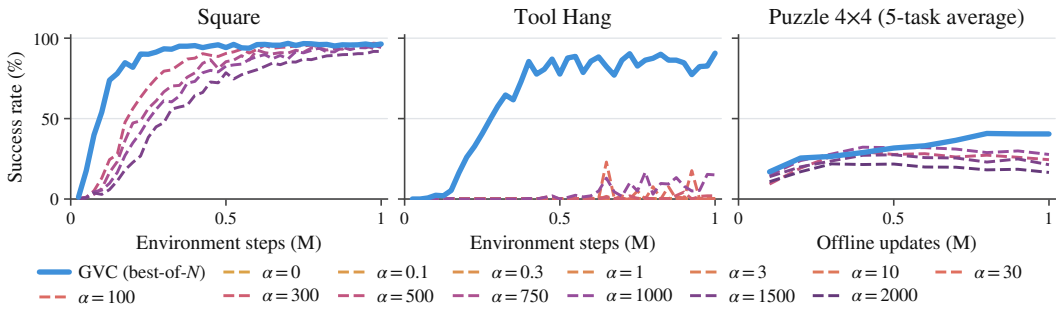


Figure 8: **Best-of-N selection versus a distilled one-step actor.** Success rate of `GVC` with its default best-of-N extraction (solid) and with a one-step actor (shortcut model) trained by direct Q-gradients under `GVC`, for a range of distillation coefficients α (dashed). Left and center: online RL on Square and Tool Hang. Right: offline RL on `puzzle-4x4`, averaged over its five tasks. $\alpha = 0$ removes the behavior anchor from the one-step actor.

`GVC` scores actions, so it can also drive an explicit actor that maximizes Q by gradient ascent. Here we test the extraction recipe of FQL (Park et al., 2025b): a one-step policy $\mu_\eta(s, \epsilon)$ that maps noise to an action is trained with

$$\mathcal{L}_{\text{distill}}(\eta) = \mathbb{E}_{s\sim\mathcal{D},\, \epsilon\sim\mathcal{N}(0,I)}\Big[\alpha \left\|\mu_\eta(s,\epsilon) - \text{Flow}_\phi(s,\epsilon)\right\|_2^2 - Q(s, \mu_\eta(s,\epsilon))\Big], \tag{16}$$

where $\text{Flow}_\phi(s,\epsilon)$ is the action obtained by integrating the BC flow f_ϕ from ϵ with 10 Euler steps, Q is the `GVC` critic, and its gradient with respect to the action is taken through the flow surrogate of Eq. (9). The BC flow actor is trained as before, and the one-step actor is used instead of best-of-N selection. This follows FQL (Park et al., 2025b) where the Q term is not normalized and other hyperparameters follow Table 4. We sweep α over $\{0, 0.1, 0.3, 1, 3, 10, 30, 100, 300, 1000\}$ on Tool Hang, $\{500, 750, 1000, 1500\}$ on Square, and $\{500, 1000, 1500, 2000\}$ on `puzzle-4x4`.

Figure 8 shows the results. On Square, the distilled actor eventually matches the performance of best-of-N by 10^6 steps (96% for the best α) but learns more slowly, reaching 78–90% at 5×10^5 steps vs. 96% for best-of-N. On Tool Hang, experiments do not find any coefficient that works: best-of-N reaches 91%, while the distilled actors stay near zero. On `puzzle-4x4`, the distilled actor(s) peak at $\sim 30\%$ around 4×10^5 steps and then decline, while best-of-N improves throughout to 40%.

Best-of-N sampling only considers actions that have been sampled from the behavior flow, so it does not leave the support of the data. The distilled actor follows $\nabla_a Q$ instead, increasing the risk of instability. We find that only coefficients in the hundreds and above make progress, and on Tool Hang none does. The main results therefore use best-of-N throughout. Extraction methods that use the action gradient while staying in support, such as adjoint matching (Li & Levine, 2026), may narrow this gap in the future.

B COMPUTE COST

`GVC` spends more compute per scored action than a discriminative critic. Instead of predicting a value in a single forward pass, it evaluates the return-conditioned likelihood of the action under every return bin. In principle, the action model could be any generative model with an exact likelihood in a single forward pass, such as a discrete autoregressive or Gaussian model. Diffusion and flow-matching policies are the strongest architectures on the tasks we evaluate, however, so we use the same architecture for the critic and approximate the likelihood over noise timesteps as described in Section 4.2.

We also reuse the policy architecture for the critic in every experiment, whereas most strong value-learning algorithms use a critic far smaller than the policy (Dong et al., 2026a). We see this as a strength, since most prior methods cannot benefit from naively scaling the critic and in fact suffer from it (Dong et al., 2026a). In practice, moreover, value learning is usually bottlenecked by environment interaction rather than by critic compute (Dulac-Arnold et al., 2021).

We keep the cost down by predicting the action likelihood for all return bins in a single forward pass through a lightweight head, following Chen et al. (2024b). Most of the network capacity therefore goes into learning $p_\theta(a \mid s)$, and the head performs the final conversion to $p_\theta(a \mid s, z)$. Measured in forward evaluations (NFEs), `GVC` is then comparable to any diffusion or flow-based critic, such as Value Flows, that must integrate over timesteps.

Concretely, Figure 9 reports the wall-clock time per gradient update for `GVC` and the offline baselines on `puzzle-4x4`, each run with its reference implementation and published hyperparameters on a single NVIDIA H100. `GVC` takes 15.6 ms per update, comparable to Value Flows (13.5 ms) and IQN (13.1 ms), the two distributional critics that also evaluate many return bins or quantiles per transition, and 5–9× the cost of the scalar regression critics of IQL, ReBRAC, FBRAC, IFQL, and FQL (1.8–3.4 ms).

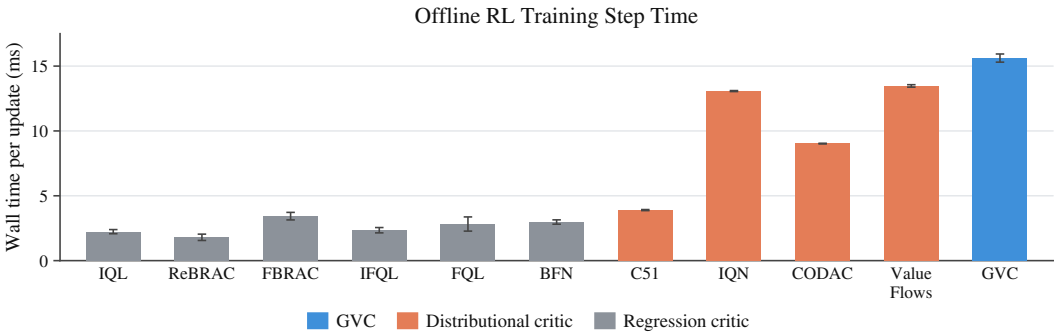


Figure 9: **Offline RL training step time.** Wall-clock time per gradient update on `puzzle-4x4` on a single H100, mean and standard deviation over 3 seeds.

C Implementation Details

C.1 GVC hyperparameters

Table 3 lists the hyperparameters shared by all experiments and Table 4 lists those that differ by domain. The paragraphs below expand on both.

Table 3: **Common GVC hyperparameters.** Values shared by all experiments. Domain-specific values are given in Table 4.

Hyperparameter	**Value**
Optimizer	Adam
Learning rate	3×10^{-4}
Batch size	256
Target network update rate	0.005
Actor f_ϕ	MLP, 4 hidden layers of width 512, no layer normalization
Action model g_θ trunk	MLP, 3 hidden layers of width 512, layer normalization, GELU
Return-bin head	per-bin 512-d embedding, one hidden layer of width 512, linear output
Return prior p_ψ	MLP on s, 3 hidden layers of width 512, layer normalization, GELU
Flow-time embedding (critic)	sinusoidal, 32-d
Flow steps (Euler)	10
Flow time for $\mathcal{L}_{\text{flow}}$ and $\mathcal{L}_{\text{actor}}$	$\lambda \sim \mathcal{U}[0, 1]$, one sample per transition
Scoring flow times T	4, stratified on $[0.1, 0.9]$, antithetic noise pair per time
Loss and prior weights	$\lambda_F = \lambda_R = 1$, $\beta_{\text{prior}} = 0.1$
Per-domain weights	β_{ret}, λ_V (Table 4)
Update-to-data ratio (online)	1
Offline gradient steps (OGBench)	10^6
Online environment steps	10^6
Evaluation episodes	100

Table 4: **Domain-specific GVC hyperparameters.** K is the number of return bins; N is the number of candidates for best-of-N selection; "Critics" gives the ensemble size and aggregation.

Domain	γ	K	β_{ret}	λ_V	N	Critics
RoboMimic, online ($h = 5$)						
`lift`, `can`, `square` (MH)	0.99	5	40	0.15	8	2, min
`tool-hang` (PH)	0.99	5	40	0.05	8	2, min
OGBench, offline ($h = 1$)						
`cube-double-play`	0.995	21	40	0.2	8	2, mean
`cube-triple-play`	0.995	21	10	0.2	8	2, mean
`puzzle-3x3-play-sparse`	0.99	51	40	0.2	8	2, mean
`puzzle-4x4-play`	0.99	11	40	0.2	8	2, mean
`puzzle-4x5-play`	0.99	21	40	0.5	8	2, mean
`puzzle-4x6-play`	0.99	21	40	0.5	8	2, mean
`scene-play-sparse`	0.995	11	40	0.05	4	1
OGBench, offline-to-online ($h = 1$)						
`antmaze-large-navigate`	0.99	21	40	0.2	4	2, mean
`humanoidmaze-medium-navigate`	0.995	21	40	0.2	8	2, mean
`cube-double-play`	0.995	21	40	0.2	8	2, mean
`cube-triple-play`	0.995	21	40	0.2	4	2, mean
`puzzle-4x4-play`	0.99	51	40	0.2	8	2, mean
`scene-play`	0.995	51	40	0.05	8	2, mean

Networks. The actor f_ϕ is the flow policy of FQL and QC, an MLP on (s, x_λ, λ). The action model g_θ takes the same inputs, with λ passed through a sinusoidal embedding. The action model conditions on the return bin only after the trunk, and we add a learned embedding of each bin to the trunk features, and apply a small head, shared across bins. One forward pass therefore covers every bin, and these embeddings are the only bin-specific parameters. The return prior $p_\psi(z \mid s)$ is a separate MLP on s with the same trunk architecture and K output logits. For ensembling (as denoted in Table 4), we train independent copies of the action model and prior. With "mean" we average the two return distributions bin by bin. With "min" we keep whichever one has the lower mean. This combined distribution is what ranks actions and what $\bar{p}$ denotes in Algorithm 1. Target networks are exponential moving averages of the online ones.

Scoring. We estimate Eq. (9) at $T = 4$ flow times, stratified over $[0.1, 0.9]$, with an antithetic noise pair $(\epsilon, -\epsilon)$ at each time. Scoring a candidate therefore costs eight forward passes of g_θ, each of which returns all K bins. The same noise and times are reused across bins and across the N candidates at a state. We use this estimator everywhere the critic is used for scoring. The training loss $\mathcal{L}_{\text{flow}}$ instead draws one fresh (ϵ, λ) per transition, with $\lambda \sim \mathcal{U}[0, 1]$. Figure 5c varies T while keeping the antithetic pairs; its variants with non-shared noise draw independent noise and times for each candidate.

Return bins and loss weights. We discretize returns into K uniformly spaced bins over $[z_1, z_K]$. This is set from the analytical minimum and maximum return given the environment reward and discount. The projection $\Pi_{\mathcal{Z}}$ in Algorithm 1 is defined in Bellemare et al. (2017) and the shifted atoms $r + \gamma m z_j$ are first clipped to $[z_1, z_K]$. The mass is then split between the two closest bins by linear interpolation.

For Eq. (10), we weight these two terms separately, with β_{ret} on $\log p_\theta(a \mid s, z_i)$ and β_{prior} on $\log p_\psi(z_i \mid s)$ (Table 3). Unless otherwise noted, we set $\lambda_F = \lambda_R = 1$ and pick λ_V for each domain (Table 4). Figure 5a varies λ_F, and Figure 7 varies λ_V.

Actor and extraction. In all settings, the actor is the behavior-cloning flow of Eq. (14), trained on the actions in the batch (dataset actions offline and replay actions online). We extract a policy by best-of-N sampling: sample N actions from the flow with 10 Euler steps, score them with the critic, and execute the one with the highest $Q(s, a)$.

We re-sample at every environment step, or after each chunk for RoboMimic. We choose a' for the TD target the same way and use the online critic to rank the candidates and the target critic to score the selected one. We use direct Q-gradients only in the ablation in Appendix A.3.

C.2 Training and evaluation protocol

RoboMimic. We follow the QC recipe. There is no offline phase: updates start after $5{,}000$ environment steps with an update-to-data (UTD) ratio of 1, and each batch is a 50/50 mix of demonstration transitions (multi-human for Lift, Can, and Square; proficient-human for Tool Hang) and replay transitions. At inference, the policy predicts chunks of $h = 5$ actions and executes them open-loop. As in QC, the critic scores whole chunks. A transition is $(s_t, a_{t:t+h}, \sum_{k<h} \gamma^k r_{t+k}, s_{t+h})$ and the target uses γ^h in place of γ. We use a sparse reward of -1 at every step until success and 0 at success. Episodes end at success, so a chunk that ends the episode has $m = 0$, and time-outs still bootstrap. We use 3 seeds and evaluate for 100 episodes every $25{,}000$ steps. The ablations in Section 5 and Appendix A.1 use the Square and Tool Hang settings of Table 4 with 8 seeds.

OGBench. We follow the Value Flows recipe. Offline results use 10^6 gradient steps on the offline dataset with $h = 1$, whereas offline-to-online results follow the same recipe but add an additional 10^6 environment steps at an update-to-data (UTD) ratio of 1, with updates resuming after $1{,}000$ online steps. The sparse `scene` and `puzzle-3x3` variants follow QC. We relabel the dataset rewards to -1, except at steps where the task is complete, which get 0. For offline-to-online we use `task1` of `antmaze-large-navigate`, `humanoidmaze-medium-navigate`, and `cube-triple-play`, `task2` of `cube-double-play` and `scene-play`, and `task4` of `puzzle-4x4-play`, all with the default rewards. The replay buffer starts out as the offline dataset. Online transitions are added to it as they are collected, and batches are always sampled uniformly from the entire buffer. We use 8 seeds and evaluate for 100 episodes every 10^5 steps; Table 1 reports the final checkpoint.

Tables. Entries in Tables 1 and 2 are means $\pm$ sample standard deviations over 8 seeds, in percentage points.

Learning curves. In Figure 3 we smooth each seed's curve with a Gaussian kernel (bandwidth 2% of the observations) and then take the mean and sample standard deviation across seeds. In Figure 4 the `GVC` and BFN curves are unsmoothed averages of the six per-task means and their bands are averages of the six per-task standard deviations, with 8 seeds per task. The other baseline curves and bands are taken from Value Flows (Dong et al., 2026b), except RLPD, which we ran ourselves. Figures 5, 6 and 7 use 8 seeds per variant, and shading shows one standard deviation.

C.3 Baseline tuning

Reference results. For offline OGBench results on `puzzle-4x4-play`, `cube-double-play`, and `cube-triple-play`, we report the baseline numbers from Value Flows (Dong et al., 2026b). We run the remaining baselines ourselves and tune them as described below. For offline-to-online OGBench results, we report the numbers from Value Flows (Dong et al., 2026b) for all baselines except BFN.

Tuning procedure. We follow the tuning protocol of Value Flows (Dong et al., 2026b): each hyperparameter is swept on the domain's default task over the values in Table 5. The BC policy is Gaussian with four hidden layers of width 512, and CODAC's conservative penalty is fixed at 0.1.

BFN. BFN shares the actor, discount, critic ensemble, and training schedule of `GVC` (Table 4) and replaces the generative critic with the regression critic of QC (Li et al., 2025).

Table 5: Hyperparameter search spaces for tuning our baselines, following Value Flows (Dong et al., 2026b). BC denotes behavioral cloning; BCFM denotes bootstrapped conditional flow matching.

Method	Hyperparameter	Search values
IQL	AWR inverse temperature	$\{0.3, 1, 3, 10\}$
ReBRAC	Actor BC coefficient	$\{0.003, 0.01, 0.03, 0.1, 0.3, 1\}$
	Critic BC coefficient	$\{0, 0.001, 0.01, 0.1\}$
IFQL	Action candidates	$\{32, 64, 128\}$
FBRAC	BC coefficient	$\{1, 3, 10, 30, 100, 300\}$
FQL	BC distillation coefficient	$\{3, 10, 30, 100, 300, 1000\}$
C51	Distributional atoms	$\{51, 101\}$
IQN, CODAC	Quantile-loss κ	$\{0.7, 0.8, 0.9, 0.95\}$
CODAC	BC coefficient	$\{100, 300, 1000, 3000, 10000, 30000\}$
Value Flows	BCFM coefficient λ	$\{0.3, 0.5, 1, 3, 10\}$
	Confidence temperature τ	$\{0.01, 0.03, 0.1, 0.3, 1, 3\}$
	Online distillation coefficient	$\{10, 30, 100, 300\}$