Reducing Scalar Rewards to Binary Success: General Off-Policy Learning with Success Functions

Armaan A. Abraham
Stanford University
armaana@stanford.edu

Extended Abstract

Recent work finds that reframing value-based RL as a classification problem (learned with a cross-entropy loss) can unlock advantages over value functions trained by mean-squared error (MSE) to predict the sum of rewards. But existing methods either impose an arbitrary discretization of the value output, or apply only to goal-conditioned RL, a narrow subset of RL problems. The question we explore in this work is whether we can reframe off-policy value learning in a way that *inherently supports classification*.

We show that *any* discounted-reward problem can be recast as a modified, reward-free problem with a single success state and an equivalent optimal policy. Any value-learning problem can then be trained by classification with cross-entropy, in a way that is in principle exact and suffers no loss of precision. Crucially, unlike goal-conditioned RL, this framework supports arbitrary dense-reward problems.

Empirically, across three benchmark suites involving a soft actor-critic setup under varying value-learning rules, with both offline and online training, we find that the resulting success functions perform on par with, and in some cases slightly better than, value functions trained with mean squared error. Compared to discretization-based approaches, particularly distributional RL (C51), the success function approach tends to trail, most notably in our largest experiment.

Overall, we present this work both **(a)** as an additional axis for analyzing the important features of value-based RL, in particular separating the importance of classification from that of distribution modeling, which existing distributional methods conflate, and **(b)** as a framework with the potential to open future theoretical directions, indicated, for example, by our derived relationship between the actor update through critic maximization used in soft actor-critic and a Bayesian update on the actor.

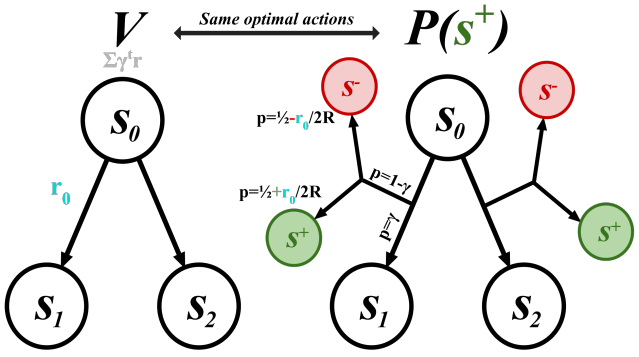

Figure 1. A discounted scalar-reward MDP (left) and its success MDP (right). The success MDP keeps the original states and transitions but, at every step, terminates with probability $1-\gamma$ into one of two absorbing states: the success state s^+ with probability $\frac{1}{2}+\frac{r}{2R}$, and the failure state s^- otherwise. The probability of eventually reaching s^+ is a positive affine transformation of the value function in the original MDP (Section 4.1), so the two problems share the same optimal actions.

Shortened Abstract

Recent work finds that reframing value-based RL as a classification problem (learned with a cross-entropy loss) can unlock advantages over value functions trained by mean-squared error (MSE) to predict the sum of rewards. But existing methods either impose an arbitrary discretization of the value output, or apply only to goal-conditioned RL, a narrow subset of RL problems.

We show that *any* discounted-reward problem can be recast as a modified, reward-free problem with a single success state and an equivalent optimal policy. Any value-learning problem can then be trained by classification with cross-entropy.

1. Introduction

Off-policy reinforcement learning holds the promise of turning past experience into future competence: by learning a value function, an agent can improve from data collected by older policies, other agents, or imperfect exploration, without requiring fresh on-policy rollouts at every update (Sutton & Barto, 1998; Watkins & Dayan, 1992). This promise is particularly compelling in domains where interaction is expensive (e.g., robotics), where we would like to extract as much learning signal as possible from each transition. More broadly, even when collecting fresh rollouts is relatively cheap, the off-policy nature of value-based methods opens the door to more scalable RL by allowing reuse of *all* experience ever collected, across the lifetime of an agent and across other agents.

Despite this promise, off-policy value-based RL has proven difficult to scale, and it often lags behind on-policy methods for reasons that remain unclear. Q-learning and its variants underlie many high-profile results in deep RL (Mnih et al., 2015; Silver et al., 2017; Kalashnikov et al., 2018; Chebotar et al., 2023), but scaling these mean-squared-error (MSE) regression targets to large networks has remained challenging (Farebrother et al., 2024). By contrast, deep learning breakthroughs in supervised settings have repeatedly leaned on classification with cross-entropy losses, from AlexNet (Krizhevsky et al., 2012) to Transformers trained for next-token prediction (Vaswani et al., 2017; Brown et al., 2020); even tasks with a natural regression flavor are often improved by reframing them as classification problems (Farebrother et al., 2024). This contrast motivates a basic question: can value learning be cast in a form that exposes it to the same advantages classification has provided in the rest of deep learning?

Existing work has explored the use of cross-entropy objectives for value learning along several axes. The most common approach is *distributional RL* (Bellemare et al., 2017; Hessel et al., 2017), which discretizes the output space of the value function and trains the network to produce a categorical distribution over bins that models the full return distribution. More recently, alternative classification-style objectives such as HL-Gauss (Imani & White, 2018; Farebrother et al., 2024) also use discrete bins over the return space, but fit a categorical model to a Gaussian distribution centered around the expected return; this approach has been shown to substantially improve scalability across single-task, multi-task, and high-capacity Transformer settings (Farebrother et al., 2024; Nauman et al., 2025), and recent work in robotic policy learning likewise relies on discretized cross-entropy targets at scale (Physical Intelligence et al., 2025; Chebotar et al., 2023). A separate line of work, goal-conditioned RL, instead models the probability of reaching a goal state, either via a binary cross-entropy loss directly (Eysenbach et al., 2021) or via an InfoNCE-style contrastive objective whose loss can be viewed as a form of cross-entropy over potential goals (Eysenbach et al., 2022; Wang et al., 2025). Across these settings, classification objectives consistently yield benefits over MSE, and recent analyses suggest several explanations: cross-entropy provides more stable gradients, mitigates sensitivity to noisy and non-stationary targets, and balances learning signals across tasks with heterogeneous reward scales, issues that are otherwise exacerbated by gradient interference (Yu et al., 2020; Liu et al., 2021) and reward-scale imbalance (Teh et al., 2017; Hessel et al., 2018) in multi-task RL (Farebrother et al., 2024; Nauman et al., 2025). However, each existing family of methods has limitations. Distributional and HL-Gauss-style approaches require choosing a number of bins and ultimately sacrifice precision through discretization. Goal-conditioned approaches do not require binning, but their reduction of the task to "reach a goal state" precludes the use of dense rewards, which provide a richer training signal and a learning curriculum. More abstractly, dense rewards allow us to specify either that a state is *intrinsically* partially desirable (e.g., a room that is partially clean), or that a state is *instrumentally* valuable as a prior on the policy (e.g., in a maze, states closer to the goal are typically more useful than ones further away); both of these are largely unsupported by goal-conditioned RL.

Motivated by prior work showing the empirical benefits of cross-entropy objectives in value learning, we derive a universal theoretical equivalence between any discounted, bounded-reward MDP and a modified, reward-free MDP. The modified MDP consists of the states and transitions of the original MDP augmented with two absorbing states, a success state and a failure state, where the probabilities of transitioning to these states are functions of the original rewards (Figure 1). We show that the value function in the original MDP is an affine function of the probability of reaching the success state in the modified MDP, with a positive proportionality constant. This allows us to take

any existing discounted-reward problem and translate it into our modified MDP, then learn an estimator of the probability of success, which we call a *success function*, that is in one-to-one correspondence with the value function in the original MDP. Because of this positive affine relationship, maximizing the success function in the modified MDP is equivalent to maximizing the value function in the original MDP for the purpose of policy extraction. We further derive a Bellman equation for the success function, which enables off-policy learning with a binary cross-entropy loss. In this way, we arrive at a principled approach to using a cross-entropy objective for any discounted-reward problem. This reformulation is fundamentally different from distributional and HL-Gauss approaches, both of which retain the original scalar-reward framing and impose a lower-precision discretization to model returns probabilistically; ours, by contrast, requires no loss of precision through binning. It also fundamentally differs from goal-conditioned approaches, because it inherently supports any dense-reward problem (e.g., velocity-based locomotion rewards, which lack any obvious formulation in terms of goal states).

This paper makes the following contributions. **Theoretically**, we derive the affine equivalence between value functions in the original MDP and success functions in the success-augmented MDP, and we derive a Bellman equation for the success function that enables off-policy learning via binary cross-entropy. We then show how, with a particular choice of the reduction, the success function can always be made to range over the full $[0, 1]$ interval of the sigmoid output, allowing the network to use its full output range. We further illustrate how this conceptual reframing of reward maximization as success-probability maximization opens up new theoretical directions: as one example, we show that a Bayesian update on the actor in actor-critic settings corresponds to an update that closely resembles the deep deterministic policy gradient (Lillicrap et al., 2015) update, in that it optimizes the actor by taking the gradient of the success function with respect to the actor's output. **Empirically**, we focus on dense-reward problems, since the goal-conditioned RL literature already provides strong inherent support for sparse-reward and goal-based settings. The main question is whether success functions, when applied to standard dense-reward RL benchmarks, offer benefits over value functions trained with MSE, and a secondary question is how they compare with discretization-based classification approaches such as distributional RL and HL-Gauss. We also evaluate at a larger scale, given the known advantages of cross-entropy in this regime; we train on large offline datasets and in multi-task settings, both with and without online experience collection.

2. Related work

Cross-entropy objectives in value learning. Prior work has shown empirical benefits to discretizing the output space of the value function and using cross-entropy objectives to learn either the full categorical distribution of returns, as in C51 (Bellemare et al., 2017) and follow-ups that integrate this idea into broader deep RL recipes (Hessel et al., 2017; Nauman et al., 2025) and large-scale robotic policy learning (Physical Intelligence et al., 2025), or, alternatively, a categorical distribution that does not model the full return distribution but rather a Gaussian centered on the bootstrapped scalar target, as in HL-Gauss (Imani & White, 2018; Farebrother et al., 2024). Both families share the discretization of the value output space and the use of a cross-entropy training objective (Rowland et al., 2018); both ultimately sacrifice precision through binning and require choosing a number of bins.

Control as probabilistic inference. A separate line of work casts policy optimization as inference in a graphical model. Toussaint et al. (2006) construct, given a model of the environment, a mixture of finite-time MDPs in which each component emits a single binary reward at its horizon, and show that the likelihood of observing reward in this mixture is an affine transformation of the value function in the original MDP. Their algorithmic instantiation applies an expectation-maximization (EM) algorithm whose E-step consists of forward-backward sweeps through the dynamics under the current policy; the construction does not yield a Bellman equation that allows off-policy learning, since the inference iterates over multiple steps forward and backward in time under the model. The framework surveyed by Levine (2018) instead keeps the rewards of the existing MDP and adds an "optimality" latent variable on top of it, rather than making a structural change to the MDP itself; variational inference under this graphical model recovers maximum-entropy RL, i.e., a policy maximizing expected return plus an entropy term rather than the standard discounted return, with natural connections to several practical methods, including soft actor-critic (Haarnoja et al., 2018), maximum a posteriori policy optimisation (Abdolmaleki et al., 2018), and reward-weighted regression (Peters & Schaal, 2007). This line of work provides rich theory, but it stays within the scalar-reward framework and requires a chosen temperature on the entropy term.

Goal-conditioned reinforcement learning. Goal-conditioned RL addresses a problem class conceptually distinct from reward-based RL: a known set of success states is provided and, instead of maximizing a discounted sum of rewards, the objective is to maximize the probability of reaching one of these states (Eysenbach et al., 2020; 2021; Wang et al., 2025). At their core, these methods rely on providing concrete future or goal states (e.g., specific block

configurations for a robotic manipulation task) that the policy should pursue. One advantage of this framing is that any state encountered, even ones not ultimately desired as goals, can be used from experience by learning a goal-parameterized value function and passing such states to it; this allows for a natural curriculum and supports learning even from a single goal demonstration (Andrychowicz et al., 2017; Liu et al., 2024). The example-based control framing of Eysenbach et al. (2021) extends this idea further by allowing demonstration states whose desirability is mediated by an unknown joint probability with a latent success event, enabling a notion of intrinsically varying state desirability; Rudner et al. (2021) similarly admit outcome examples within a variational inference framing. Several works in this line have derived Bellman-like recursions in terms of probabilities (Eysenbach et al., 2020; 2021) or log-probabilities (Rudner et al., 2021), and related formulations have studied successor-state densities for goal-dependent values (Blier et al., 2021). In particular, because the joint probability between demonstration states and the latent success event is unknown in Eysenbach et al. (2021), their recursive classification objective targets an indirect classifier ratio that must be Bayesian-transformed to recover the success probability, whereas in our reduction the Bellman equation acts directly on the predicted success probability. All of these approaches deal with concrete goal or success states in one way or another, and, unlike our approach, do not provide a way to apply this style of probabilistic value learning to dense scalar-reward problems (e.g., a velocity-based locomotion reward) that lack any obvious specification in terms of goal states. While our approach also supports the provision of concrete goal states and the use of cross-entropy losses to learn a policy reaching them, it embraces scalar rewards by treating the success state as an unobserved augmentation of the original MDP rather than a concrete state of the environment. In this sense, success functions can be viewed as a unification of goal-conditioned and scalar-reward reinforcement learning. In sum, none of these lines of work brings cross-entropy value learning to general dense-reward problems with off-policy Bellman updates.

3. Preliminaries

For the purpose of our analysis, we describe the standard reinforcement learning setup. Consider a Markov decision process (MDP) with state space $\mathcal{S}$, action space $\mathcal{A}$, transition kernel $P(s' \mid s, a)$, bounded reward function $r(s, a) \in [-R, R]$ for some $R > 0$, discount $\gamma \in [0, 1)$, and a set of terminal states $\mathcal{S}_T \subseteq \mathcal{S}$. A policy $\pi(\cdot \mid s)$ induces trajectories $(s_0, a_0, r_0, s_1, a_1, \ldots)$ with $a_t \sim \pi(\cdot \mid s_t)$, $r_t = r(s_t, a_t)$, and $s_{t+1} \sim P(\cdot \mid s_t, a_t)$. Let $\tau = \min\{t \geq 0 : s_t \in \mathcal{S}_T\}$ denote the first time a terminal state is reached, with $\tau = \infty$ if no such state is ever reached. The objective is to find a policy maximizing the expected discounted return

$$V^\pi(s_0) \;=\; \mathbb{E}_\pi\left[\sum_{t=0}^{\tau-1} \gamma^t\, r(s_t, a_t)\right]. \tag{1}$$

Boundedness of r together with $\gamma < 1$ ensure that V^π is well-defined even when $\tau = \infty$; the analysis to follow applies equally to non-terminating settings (formally, $\mathcal{S}_T = \emptyset$ and $\tau = \infty$).

Bellman equation. We adopt the standard convention that terminal states are absorbing with zero reward in the underlying MDP, i.e., $r(s, a) = 0$ and $P(s \mid s, a) = 1$ for all $s \in \mathcal{S}_T$ and $a \in \mathcal{A}$ (Sutton & Barto, 1998); under this convention $V^\pi(s) = 0$ for any $s \in \mathcal{S}_T$. The value function V^π for a policy π satisfies the Bellman equation

$$V^\pi(s) \;=\; \mathbb{E}_{a\sim\pi(\cdot|s)}\left[r(s, a) + \gamma\, \mathbb{E}_{s'\sim P(\cdot|s,a)}[V^\pi(s')]\right], \tag{2}$$

which underlies bootstrapped, off-policy value learning: a sampled transition (s, a, r, s') yields the regression target $r + \gamma \bar{V}(s')$, where $\bar{V}$ is a (typically slow-moving) estimate of V^π.

Discounting-termination equivalence. A standard observation from prior work (Sutton & Barto, 1998, Sec. 5.8) is that maximizing the discounted return in (1) is equivalent to maximizing the expected undiscounted return in an augmented MDP in which, at each step, the agent transitions to an absorbing termination state with probability $1 - \gamma$ and otherwise follows the original transition kernel P. The discount factor in the original MDP thus plays the role of a per-step continuation probability in the augmented MDP. This equivalence is the structural starting point for the construction we develop next.

4. Method

We first construct the success MDP and prove the affine equivalence between the probability of success and the value function (Section 4.1), then derive the Bellman recursion that makes the success probability learnable off-policy with a binary cross-entropy loss (Section 4.2). Section 4.3 extends the construction to asymmetric reward bounds, which we use in practice, and Section 4.4 derives the Bayesian actor update used by the logarithm variant of our agent. Section 4.5 briefly reviews the Beta distribution, and Section 4.6 uses it to construct a distributional variant of the success critic.

4.1. The success MDP

Definition 4.1. Given the MDP of Section 3, define the **success MDP** by adding two absorbing states s^+ (success) and s^- (failure). From any non-terminal state $s \notin \mathcal{S}_T$, under action a:

- With probability γ: transition to $s' \sim P(\cdot \mid s, a)$.
- With probability $1-\gamma$: transition to

$$\begin{cases} s^+ & \text{with probability } \dfrac{1}{2} + \dfrac{r(s,a)}{2R}, \\ s^- & \text{with probability } \dfrac{1}{2} - \dfrac{r(s,a)}{2R}. \end{cases}$$

Upon reaching any terminal state $s \in \mathcal{S}_T$, transition immediately to s^+ or s^- each with probability $\frac{1}{2}$.

The reward bound R ensures that $\frac{1}{2} + \frac{r(s,a)}{2R}$ is a valid probability. We write $P^\pi(s^+ \mid s_0)$ for the probability that a trajectory of the success MDP starting at s_0 under π is eventually absorbed at s^+, and we call $P^\pi(s^+ \mid \cdot)$ the *success function*.

Theorem 4.2. *For any policy π,*

$$P^\pi(s^+ \mid s_0) \;=\; \frac{1-\gamma}{2R}\, V^\pi(s_0) \;+\; \frac{1}{2}.$$

Proof. Fix a policy π and a trajectory $(s_0, a_0, s_1, a_1, \ldots)$ that first reaches $\mathcal{S}_T$ at step τ. The agent enters s^+ either by discount-termination at some step $t < \tau$ or by the final 50/50 transition at step τ. Conditioning on the trajectory:

$$P^\pi(s^+ \mid s_0) = \mathbb{E}_\pi\Bigg[\sum_{t=0}^{\tau-1} \gamma^t (1-\gamma)\left(\frac{1}{2} + \frac{r(s_t,a_t)}{2R}\right) + \gamma^\tau \cdot \frac{1}{2}\Bigg]. \tag{3}$$

Expand the sum in (3) into two parts:

$$\sum_{t=0}^{\tau-1} \gamma^t (1-\gamma)\left(\frac{1}{2} + \frac{r(s_t,a_t)}{2R}\right) = \frac{1}{2}(1-\gamma)\sum_{t=0}^{\tau-1}\gamma^t \;+\; \frac{1-\gamma}{2R}\sum_{t=0}^{\tau-1}\gamma^t\, r(s_t,a_t). \tag{4}$$

The first term in (4) evaluates via the geometric series:

$$\frac{1}{2}(1-\gamma)\sum_{t=0}^{\tau-1}\gamma^t = \frac{1}{2}(1-\gamma^\tau).$$

Substituting back into (3):

$$P^\pi(s^+ \mid s_0) = \mathbb{E}_\pi\Bigg[\frac{1}{2}(1-\gamma^\tau) + \frac{1}{2}\gamma^\tau + \frac{1-\gamma}{2R}\sum_{t=0}^{\tau-1}\gamma^t\, r(s_t,a_t)\Bigg].$$

The first and second terms combine:

$$\frac{1}{2}(1-\gamma^\tau) + \frac{1}{2}\gamma^\tau = \frac{1}{2},$$

which holds for every trajectory regardless of τ. Therefore

$$P^\pi(s^+ \mid s_0) = \frac{1}{2} + \frac{1-\gamma}{2R}\,\mathbb{E}_\pi\left[\sum_{t=0}^{\tau-1}\gamma^t\, r(s_t,a_t)\right]$$

and the expectation is exactly $V^\pi(s_0)$. □

Since $\frac{1-\gamma}{2R} > 0$, maximizing $P^\pi(s^+ \mid s_0)$ is equivalent to maximizing $V^\pi(s_0)$: the success MDP and the original MDP share the same optimal policies.

4.2. A Bellman recursion for the success function

We construct a recursion for the success function at a given state by considering two branches. With probability $1-\gamma$, the process takes the *termination path*: the episode resolves immediately, ending in success with probability $\frac{1}{2} + \frac{r}{2R}$ and in failure otherwise, so the reward determines the split between success and failure. With probability γ, the process takes the *continuation path*: the episode continues to the next state, and the probability of eventually succeeding is the success function at that state, for which we substitute our own estimate (the bootstrap). Concretely,

$$P^\pi(s^+ \mid s_t) = \mathbb{E}\Bigg[\underbrace{(1-\gamma)}_{\substack{\text{termination}\\ \text{path}}}\underbrace{\left(\frac{1}{2} + \frac{r(s_t,a_t)}{2R}\right)}_{\substack{\text{success vs. failure,}\\ \text{given by the reward}}} + \underbrace{\gamma}_{\substack{\text{continuation}\\ \text{path}}}\underbrace{P^\pi(s^+ \mid s_{t+1})}_{\substack{\text{bootstrapped}\\ \text{estimate}}}\Bigg], \tag{5}$$

with the expectation over $a_t \sim \pi(\cdot \mid s_t)$ and $s_{t+1} \sim P(\cdot \mid s_t, a_t)$. Replacing the expectation with a sampled transition (s, a, r, s') and reading the next-state success probability from a target network $\bar{P}$ gives the target

$$y \;=\; (1-\gamma)\big(\tfrac{1}{2} + \tfrac{r}{2R}\big) + \gamma\, \bar{P}(s^+ \mid s'), \tag{6}$$

which lies in $[0,1]$ and can therefore be used directly as the target in a binary cross-entropy loss on the critic's sigmoid output.

4.3. Asymmetric reward bounds

The symmetric construction of Definition 4.1 uses a single bound R on the reward magnitude. When separate bounds are known, i.e., $r(s,a) \in [-R_-, R_+]$ with $R_-, R_+ \geq 0$ and $R_- + R_+ > 0$, the construction can be modified to use them directly.

Definition 4.3. Modify Definition 4.1 as follows. From any non-terminal state $s \notin \mathcal{S}_T$, under action a: with probability γ, transition to $s' \sim P(\cdot \mid s, a)$; with probability $1-\gamma$, transition to s^+ with probability $\frac{R_- + r(s,a)}{R_- + R_+}$, else to s^-. Upon reaching any terminal state $s \in \mathcal{S}_T$, transition immediately to s^+ with probability $\frac{R_-}{R_- + R_+}$ and to s^- otherwise.

Theorem 4.4. *For any policy π,*

$$P^\pi(s^+ \mid s_0) \;=\; \frac{R_-}{R_- + R_+} \;+\; \frac{1-\gamma}{R_- + R_+}\, V^\pi(s_0).$$

The proof follows the same decomposition as Theorem 4.2 and is given in Appendix C.1. The slope is again positive, so the affine equivalence is preserved, and the Bellman recursion and binary cross-entropy target of Section 4.2 carry over with $\frac{1}{2} + \frac{r}{2R}$ replaced by $\frac{R_- + r}{R_- + R_+}$. The termination-success probability $\frac{R_- + r(s,a)}{R_- + R_+}$ now spans $[0,1]$ as r ranges over $[-R_-, R_+]$, and $P^\pi(s^+ \mid s_0)$ ranges over the full $[0,1]$ interval as $V^\pi(s_0)$ ranges over its attainable extremes: the success function can use the full output range of a sigmoid critic, rather than being confined to a sub-interval by a loose symmetric bound. We use this construction in practice, with R_- and R_+ set from the most negative and most positive rewards of the environment. The symmetric case $R_- = R_+ = R$ recovers Definition 4.1.

4.4. A Bayesian actor update

Let $P^\pi(s^+ \mid s, a)$ denote the action-conditioned success function, i.e., the probability of reaching s^+ after taking action a in state s and following π thereafter. The actor-critic gradient through a critic, e.g., the DDPG actor gradient $\nabla_\theta \mathbb{E}_s[Q^\pi(s, \mu_\theta(s))]$ (Lillicrap et al., 2015; Silver et al., 2014), corresponds in the success MDP to a gradient through $P^\pi(s^+ \mid s, a)$. The KL-regularized version of this objective with $\log P^\pi(s^+ \mid s, a)$ as the reward admits the Bayesian posterior in closed form, with no free temperature. Fix a reference policy π with $P^\pi(s^+ \mid s, a) > 0$ for all (s, a), and define the per-state objective over distributions $\mu \in \Delta(\mathcal{A})$:

$$J_s(\mu) \;=\; \mathbb{E}_{a\sim\mu}\big[\log P^\pi(s^+ \mid s, a)\big] \;-\; \mathrm{KL}(\mu \,\|\, \pi(\cdot \mid s))\,. \tag{7}$$

Theorem 4.5. *J_s has the unique maximizer*

$$\mu^*(a) \;=\; \frac{\pi(a \mid s)\, P^\pi(s^+ \mid s, a)}{P^\pi(s^+ \mid s)} \;=\; P^\pi(a \mid s, s^+).$$

The proof is given in Appendix C.2. The maximizer is the Bayesian posterior obtained by conditioning the joint distribution on the success event, and the temperature on the KL term is fixed at 1 by the requirement that $\log P^\pi(s^+ \mid s, a)$ be a log-probability rather than a free scalar. A DDPG-style actor update that ascends the *logarithm* of a success-probability critic therefore performs a Bayesian update of the policy conditioned on success; this motivates the logarithm variant of the actor objective in our experiments.

4.5. The Beta distribution

Our experiments include a distributional variant of the success function that maintains a full distribution over the success probability rather than a point estimate. The natural family for this is the Beta distribution: a distribution over $[0,1]$ with density $p(x) \propto x^{\alpha-1}(1-x)^{\beta-1}$ for parameters $\alpha, \beta > 0$, and the conjugate prior of the Bernoulli. We parameterize it by its mean $\mu = \alpha/(\alpha+\beta)$ and concentration $\nu = \alpha + \beta$ (so $\alpha = \mu\nu$ and $\beta = (1-\mu)\nu$); its variance is $\mu(1-\mu)/(\nu+1)$, so larger concentration corresponds to a sharper distribution about the mean.

4.6. A Beta-distributional success critic

Beyond the point-estimate success function, the success-MDP view admits a distributional variant that models the success probability itself with a Beta distribution: the critic outputs two channels parameterizing $\mathrm{Beta}(\alpha_\theta(s,a), \beta_\theta(s,a))$, trained by projecting the distribution implied by each transition at the next state onto a Beta distribution and minimizing a KL divergence to it. Concretely, for a transition (s, a, r, s'), the target mean follows the Bellman recursion (5), $m^* = p_+ + \gamma\,\mu'$, where $p_+ = (1-\gamma)\,\frac{R_- + r}{R_- + R_+}$ is the termination-success probability under the asymmetric bounds of Section 4.3 and μ' is the target network's Beta mean at s'; the target variance follows the law of total variance across the three branches of the recursion (termination into success, termination into failure, continuation), combining the between-branch variation induced by the reward with the bootstrapped variance of the Beta at the next state:

$$v^* = p_+(1-m^*)^2 + p_-(m^*)^2 + \gamma\big((\mu' - m^*)^2 + v'\big),$$

where $p_- = (1-\gamma) - p_+$ is the termination-failure probability and v' is the target Beta variance at s'. The pair (m^*, v^*) is converted to mean-concentration form via $\nu^* = m^*(1-m^*)/v^* - 1$, with ν^* floored at a hyperparameter ν_{floor} to keep the projected $(\alpha^*, \beta^*) = (m^*\nu^*, (1-m^*)\nu^*)$ strictly positive when the matched variance approaches or exceeds $m^*(1-m^*)$, and the critic minimizes the forward KL $\mathrm{KL}\big(\mathrm{Beta}(\alpha^*, \beta^*) \,\|\, \mathrm{Beta}(\alpha_\theta, \beta_\theta)\big)$. The actor ascends the Beta mean. Collapsing the Beta to a point Bernoulli recovers binary cross-entropy on the target (6).

5. Experiments

We design our experiments to test whether success functions, applied to standard dense-reward benchmarks under an actor-critic setup shared across all the value-learning rules, offer benefits over value functions trained with MSE, and how they compare with discretization-based classification approaches. Concretely, we ask:

1. Do success functions improve over MSE-trained value functions (TD)?
2. How do they compare with cross-entropy methods that discretize the return support (Distributional, HL-Gauss)?

3. Does the Bayesian (logarithm) actor update of Section 4.4 change performance relative to ascending the success probability directly?
4. Does modeling a full Beta posterior over the success probability, rather than a point estimate, help?

5.1. Main comparison

Value-learning rules. The actor-critic setup is shared across the value-learning rules: within each benchmark, we hold every other hyperparameter constant and vary only the value-learning rule, using soft actor-critic style updates with a DDPG-style actor that samples from a tanh-squashed Gaussian and maximizes a value read from the critic (Lillicrap et al., 2015; Silver et al., 2014; Haarnoja et al., 2018; Ball et al., 2023). We compare four rules: **TD**, a scalar critic trained with MSE; **HL-Gauss** (Farebrother et al., 2024) and **Distributional** (categorical) (Bellemare et al., 2017), both cross-entropy objectives over a discretized return support (51 bins on D4RL, 51 and 204 on ExORL, 204 on MT80); and our **success function** (Ps^+), a scalar critic that outputs a success probability through a sigmoid and is trained with binary cross-entropy on the target (6), using the asymmetric-bounds construction of Section 4.3. For the Ps^+ actor we maximize one of two quantities read from the critic: its probability $\sigma(z)$, or its logarithm $\log \sigma(z)$ (the variant Ps^+ (log)), where the logarithm performs the Bayesian policy update of Section 4.4. Training details common to all agents (entropy regularization, precision, network structure) are given in Appendix A.2; the precise settings of this shared setup (network sizes, batch sizes, n-step returns) vary across benchmarks and are listed in Appendix A.3.

D4RL. Five tasks spanning dexterous manipulation, maze navigation, and locomotion from the D4RL suites (Fu et al., 2020), accessed through Minari (Younis et al., 2024): Adroit hammer and pen (24-DoF hand manipulation), PointMaze (force-actuated navigation with a dense distance-based reward), and MuJoCo HalfCheetah and Hopper (velocity-reward locomotion), each paired with its offline dataset (`expert`; `medium-dense` for PointMaze). Training uses RLPD-style continual mixing (Ball et al., 2023): there is no separate offline phase; every gradient step is accompanied by an environment step, and each batch is split $50/50$ between the offline dataset and the online replay buffer. Runs train for 2M online steps.

ExORL. Three DeepMind Control locomotion tasks (Tassa et al., 2018) (cheetah-run, quadruped-run, walker-run) from the ExORL benchmark (Yarats et al., 2022), using the exploratory `proto` datasets, which were collected by an unsupervised exploration agent and relabeled with task rewards. Training is two-stage offline→online: 1M gradient steps of offline pretraining on the ExORL dataset, then 1M steps of online fine-tuning (2M total).

MT80. TD-MPC2's 80-task multitask suite (Hansen et al., 2023): 50 Meta-World manipulation tasks (Yu et al., 2019) and 30 DeepMind Control tasks (Tassa et al., 2018), trained purely offline on TD-MPC2's released MT80 dataset (≈545M transitions). We cloned the TD-MPC2 repository and implemented our agents within it; task identity is supplied through TD-MPC2's learnable task embedding (96-dim, ℓ_2-norm ≤ 1, concatenated to network inputs). As external reference points we reuse the published MT80 scores of TD-MPC2 (70.6) and of the categorical BRC+BC agent (64.5), taken from their respective papers, which use the same scoring methodology (Hansen et al., 2023; Nauman et al., 2025).

Metrics and seeds. For D4RL and ExORL we run four seeds per task (3 for β-KL in Figure 4), score every run at exactly 2M training steps, normalize each task's final return by the best run on that task (per-task best $= 100$; the figures display the same quantity scaled to $[0, 1]$), and aggregate with a task-stratified bootstrap (95% CIs). For MT80 we run a single seed per agent (the multitask runs are expensive) and report TD-MPC2's normalized score (Hansen et al., 2023), with every agent scored at 3.8M training steps. Full evaluation details are in Appendix A.1, and per-benchmark hyperparameters in Appendix A.3.

5.2. Beta-distributional critic evaluation

We evaluate the Beta-distributional success critic of Section 4.6 on D4RL. The agent introduces several new hyperparameters (the parameterization of the two output channels, concentration biases and floors, and the target projection floor ν_{floor}). In preliminary experiments, results were particularly sensitive to ν_{floor}; we therefore evaluate the agent at two values of this floor, $\nu_{\text{floor}} \in \{4, 8\}$, with all other hyperparameters matched to the D4RL setup (Appendix A.4).

5.3. Results

At the D4RL and ExORL scale, the value-learning rules perform similarly; differences are larger on MT80. Figure 2 summarizes final performance, Figure 3 shows training curves, and per-task scores for all agents are tabulated in Appendix B (Tables 4–6). On D4RL, the spread in aggregate normalized return between the lowest and highest scoring rules (TD and Distributional) is about 9 points (72.2 to 80.8), with overlapping 95% confidence intervals; the spread on ExORL is similar (85.0 to 94.1). On the larger-scale MT80 suite the spread grows to about 13 normalized-score points (75.8 to 88.7). All of our agents exceed both external baselines on MT80 despite those baselines using much larger networks (TD-MPC2) and categorical critic losses (both); we use a separate entropy temperature per task on this suite (Appendix A.2), and the importance of getting the entropy handling right may explain why even our TD baseline (75.8)

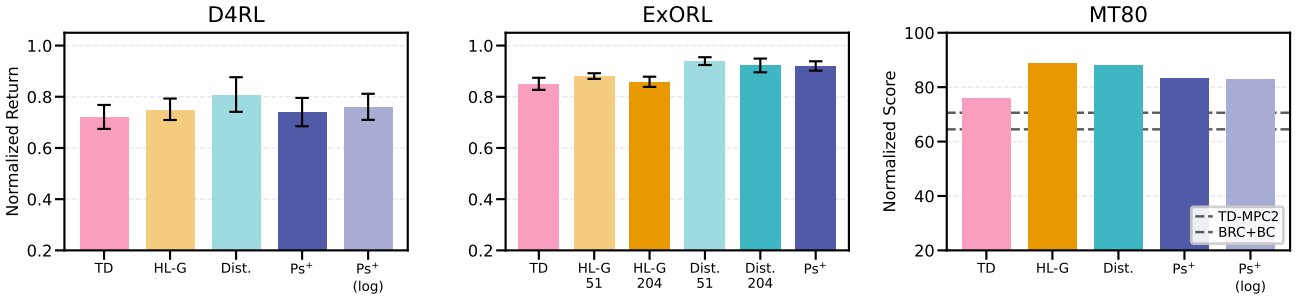

Figure 2. Final performance of the value-learning rules under the same policy extraction mechanism: TD, HL-Gauss, Distributional, and the success function (Ps^+; Ps^+ (log) is its logarithm variant). **Left:** D4RL (five tasks). **Middle:** ExORL (three DeepMind Control tasks; 51 and 204 bins shown for HL-Gauss and Distributional). **Right:** MT80 (80-task suite); dashed lines mark the external baselines TD-MPC2 (Hansen et al., 2023) and BRC+BC (Bigger, Regularized, Categorical) (Nauman et al., 2025). Four seeds per task for D4RL/ExORL; one seed for MT80.

exceeds both external baselines.

Success functions do not provide a consistent benefit over TD learning, and where they do, the benefit is small. On D4RL the two are essentially indistinguishable (74.1 for Ps^+ and 76.2 for Ps^+ (log), against 72.2 for TD, with overlapping confidence intervals). On ExORL and MT80, success functions outperform TD (92.1 vs. 85.0, and 83.4 vs. 75.8), but the margins are modest relative to the spread across rules.

Distributional RL provides the most consistent performance, and HL-Gauss performs best on MT80. The Distributional rule is the best rule on D4RL (80.8) and ExORL (94.1 at 51 bins, 92.3 at 204), and is within a point of the best rule on MT80 (88.0 vs. 88.7). HL-Gauss is less consistent: it is the best rule on MT80 (88.7), but trails Distributional on D4RL (74.7 vs. 80.8) and trails both Distributional and the success function on ExORL (88.1 and 85.7 at 51 and 204 bins, vs. 94.1 and 92.1).

The Bayesian actor update is not consistently better or worse than ascending the success probability directly. The logarithm variant improves over the sigmoid variant on D4RL (76.2 vs. 74.1) but scores slightly lower on MT80 (82.8 vs. 83.4). We did not run the logarithm variant on ExORL.

The Beta-distributional critic performs poorly at the hyperparameters we tested. Figure 4 shows training curves on D4RL for the β-KL agent at $\nu_{\text{floor}} \in \{4, 8\}$ alongside the success function. At both floor values, the projected target concentration sits at the floor for essentially all transitions throughout training: the KL update is consistently targeting a very broad distribution, which is potentially not a sharp learning signal. In aggregate the agent reaches a normalized return of 2.8 ($\nu_{\text{floor}} = 4$) and 16.7 ($\nu_{\text{floor}} = 8$), against 74.1 for the success function under the same setup, and it underperforms the success function on every task at both floor values (Table 4). At the same time, the results are sensitive to the floor: raising it from 4 to 8 improves both tasks on which the agent makes progress (PointMaze, 24.7 to 53.0; HalfCheetah, -2.7 to 37.1), which suggests that further hyperparameter tuning could improve performance.

5.4. Why do the discretization-based rules perform well?

We consider three hypotheses for the benefit of the discretization-based cross-entropy rules.

Hypothesis 1: binning. The benefit comes from binning the output space and modeling a distribution over the bins in some form. This is supported by MT80, where the two binned rules score highest, but ExORL is contrary to it, since HL-Gauss trails the bin-free success function there. The two-hot results of Farebrother et al. (2024) are also contrary: their two-hot agent represents the expected return exactly by interpolating between the two nearest bins, yet performs worse than HL-Gauss and categorical distributional RL.

Hypothesis 2: reduced precision. The loss of precision that comes with binning is itself responsible for the benefit. This is likewise only partially consistent with our results, since HL-Gauss does not perform as well at the smaller scales; the two-hot results of Farebrother et al. (2024) support it, however, since two-hot uses bins without losing fidelity in the representation and performs worse than the lossy alternatives.

Hypothesis 3: distribution modeling. Modeling the full distribution of returns is the benefit, potentially because it enriches the learning signal or reduces overfitting. This is the hypothesis most consistent with our results: Distributional is the best rule, or within a very small gap of the best, in every suite (best on D4RL and ExORL; 0.7 points behind HL-Gauss on MT80). Our Beta-distributional critic, which also models a distribution yet performed poorly, is the exception; given the limited tuning and the relatively undeveloped state of that agent, we do not take those results

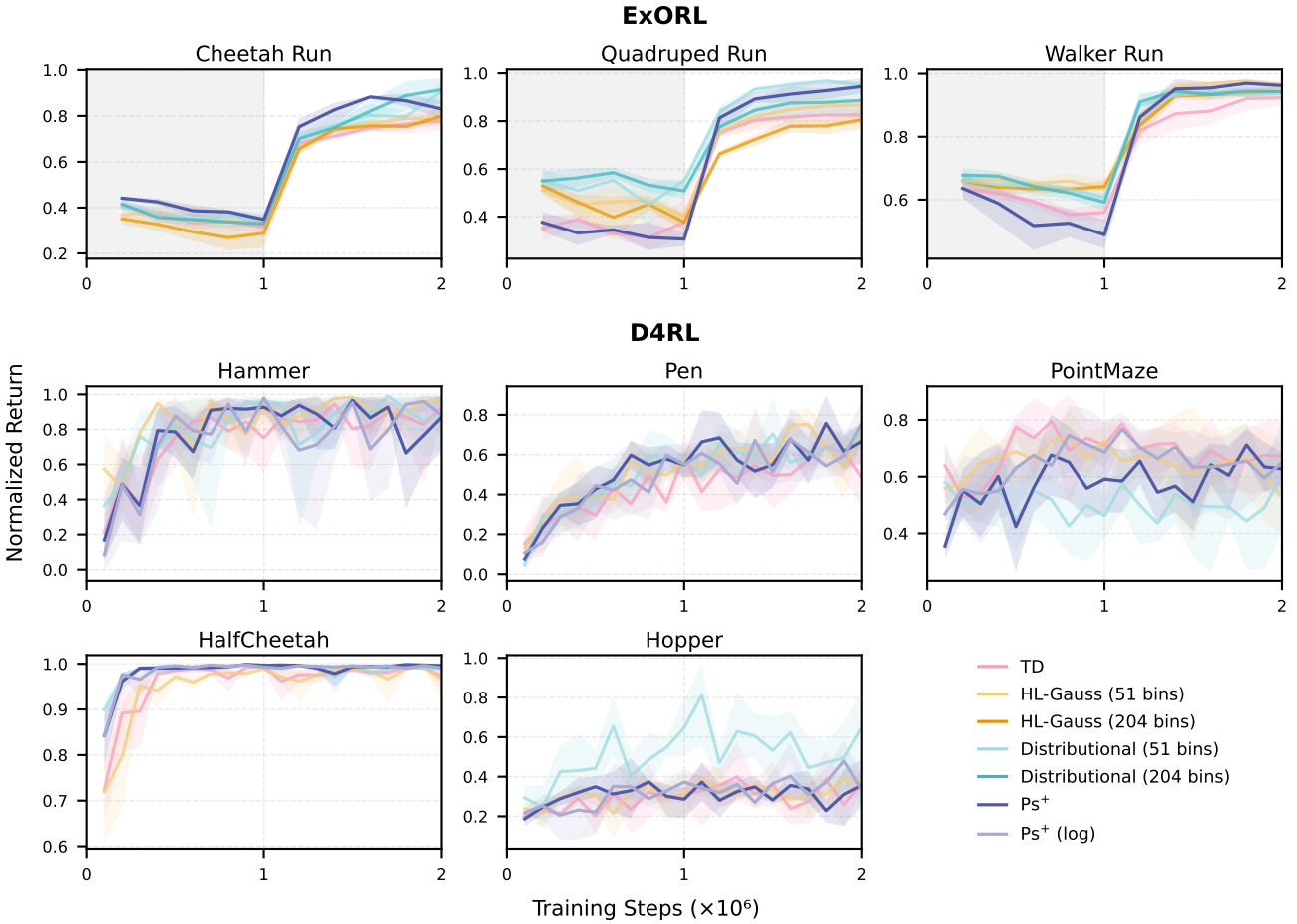

Figure 3. Training curves for the value-learning rules on ExORL (top) and D4RL (bottom). Returns are normalized per task (Appendix A.1). The grey region marks offline training and the white region online training: ExORL runs train offline for the first 1M steps and online for the second 1M, while D4RL runs are online throughout, with RLPD-style data mixing.

as reliable evidence about distribution modeling as a whole.

6. Conclusion

We derived an exact equivalence between any discounted, bounded-reward MDP and a reward-free success MDP, in which the value function corresponds, through a positive affine map, to the probability of reaching a single absorbing success state. The resulting success functions support off-policy learning with a binary cross-entropy loss on any dense-reward problem, with no discretization of the value output. Their main limitation is empirical: success functions do not provide a consistent or substantial performance benefit, performing on par with, and in some cases slightly better than, MSE-trained value functions, while tending to trail distributional RL, particularly in our largest-scale experiment.

Beyond raw performance, we see two uses for the construction. First, as an experimental probe: success functions realize a classification-style value loss with no loss of precision from discretization, which helps separate which features of classification-based value learning are responsible for its benefits (the cross-entropy objective, the binning, or modeling a full distribution). Under this reading, our results suggest that the cross-entropy objective alone is not the operative factor, and are most consistent with distribution modeling being part of the benefit. Second, as a source of theory: analogously to how control as inference obtains a body of analysis from a probabilistic view of RL while remaining in the scalar-reward framework, the success MDP makes the value function itself a probability, and quantities like the Bayesian actor update of Section 4.4, which recovers a DDPG-style update on the logarithm of the critic, follow directly. Further theoretical development of this view, along with the practical tools it may yield, is a natural direction for future work.

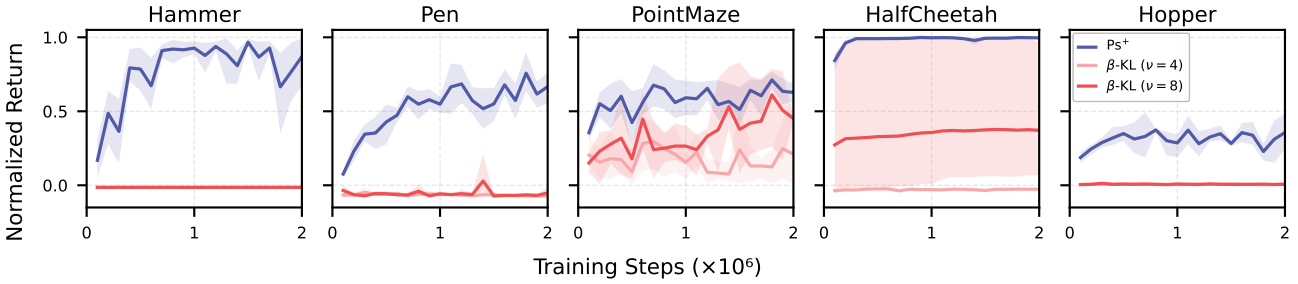

Figure 4. The Beta-distributional success critic (β-KL) on D4RL at two values of the target concentration floor, $\nu_{\text{floor}} \in \{4, 8\}$, alongside the success function (Ps^+). Returns are normalized by the same per-task denominators as Figure 3.

References

Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N., and Riedmiller, M. Maximum a posteriori policy optimisation, 2018. URL https://arxiv.org/abs/1806.06920.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. Hindsight experience replay, 2017. URL https://arxiv.org/abs/1707.01495.

Ball, P. J., Smith, L., Kostrikov, I., and Levine, S. Efficient online reinforcement learning with offline data, 2023. URL https://arxiv.org/abs/2302.02948.

Bellemare, M. G., Dabney, W., and Munos, R. A distributional perspective on reinforcement learning, 2017. URL https://arxiv.org/abs/1707.06887.

Blier, L., Tallec, C., and Ollivier, Y. Learning successor states and goal-dependent values: A mathematical viewpoint, 2021. URL https://arxiv.org/abs/2101.07123.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Chebotar, Y., Vuong, Q., Irpan, A., Hausman, K., Xia, F., Lu, Y., Kumar, A., Yu, T., Herzog, A., Pertsch, K., Gopalakrishnan, K., Ibarz, J., Nachum, O., Sontakke, S., Salazar, G., Tran, H. T., Peralta, J., Tan, C., Manjunath, D., Singht, J., Zitkovich, B., Jackson, T., Rao, K., Finn, C., and Levine, S. Q-Transformer: Scalable offline reinforcement learning via autoregressive Q-functions, 2023. URL https://arxiv.org/abs/2309.10150.

Eysenbach, B., Salakhutdinov, R., and Levine, S. C-Learning: Learning to achieve goals via recursive classification, 2020. URL https://arxiv.org/abs/2011.08909.

Eysenbach, B., Levine, S., and Salakhutdinov, R. Replacing rewards with examples: Example-based policy search via recursive classification, 2021. URL https://arxiv.org/abs/2103.12656.

Eysenbach, B., Zhang, T., Salakhutdinov, R., and Levine, S. Contrastive learning as goal-conditioned reinforcement learning, 2022. URL https://arxiv.org/abs/2206.07568.

Farebrother, J., Orbay, J., Vuong, Q., Taïga, A. A., Chebotar, Y., Xiao, T., Irpan, A., Levine, S., Castro, P. S., Faust, A., Kumar, A., and Agarwal, R. Stop regressing: Training value functions via classification for scalable deep RL, 2024. URL https://arxiv.org/abs/2403.03950.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4RL: Datasets for deep data-driven reinforcement learning, 2020. URL https://arxiv.org/abs/2004.07219.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. URL https://arxiv.org/abs/1801.01290.

Hansen, N., Su, H., and Wang, X. TD-MPC2: Scalable, robust world models for continuous control, 2023. URL https://arxiv.org/abs/2310.16828.

Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning, 2017. URL https://arxiv.org/abs/1710.02298.

Hessel, M., Soyer, H., Espeholt, L., Czarnecki, W., Schmitt, S., and van Hasselt, H. Multi-task deep reinforcement

learning with PopArt, 2018. URL https://arxiv.org/abs/1809.04474.

Imani, E. and White, M. Improving regression performance with distributional losses, 2018. URL https://arxiv.org/abs/1806.04613.

Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., and Levine, S. QT-Opt: Scalable deep reinforcement learning for vision-based robotic manipulation, 2018. URL https://arxiv.org/abs/1806.10293.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

Levine, S. Reinforcement learning and control as probabilistic inference: Tutorial and review, 2018. URL https://arxiv.org/abs/1805.00909.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning, 2015. URL https://arxiv.org/abs/1509.02971.

Liu, B., Liu, X., Jin, X., Stone, P., and Liu, Q. Conflict-averse gradient descent for multi-task learning, 2021. URL https://arxiv.org/abs/2110.14048.

Liu, G., Tang, M., and Eysenbach, B. A single goal is all you need: Skills and exploration emerge from contrastive RL without rewards, demonstrations, or subgoals, 2024. URL https://arxiv.org/abs/2408.05804.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 0028-0836, 1476-4687. doi: 10.1038/nature14236. URL https://www.nature.com/articles/nature14236.

Nauman, M., Cygan, M., Sferrazza, C., Kumar, A., and Abbeel, P. Bigger, regularized, categorical: High-capacity value functions are efficient multi-task learners, 2025. URL https://arxiv.org/abs/2505.23150.

Peters, J. and Schaal, S. Reinforcement learning by reward-weighted regression for operational space control. In *International Conference on Machine Learning*, 2007. doi: 10.1145/1273496.1273590.

Physical Intelligence, Amin, A., Aniceto, R., Balakrishna, A., Black, K., Conley, K., Connors, G., Darpinian, J., Dhabalia, K., DiCarlo, J., Driess, D., Equi, M., Esmail, A., Fang, Y., Finn, C., Glossop, C., Godden, T., Goryachev, I., Groom, L., Hancock, H., Hausman, K., Hussein, G., Ichter, B., Jakubczak, S., Jen, R., Jones, T., Katz, B., Ke, L., Kuchi, C., Lamb, M., LeBlanc, D., Levine, S., Li-Bell, A., Lu, Y., Mano, V., Mothukuri, M., Nair, S., Pertsch, K., Ren, A. Z., Sharma, C., Shi, L. X., Smith, L., Springenberg, J. T., Stachowicz, K., Stoeckle, W., Swerdlow, A., Tanner, J., Torne, M., Vuong, Q., Walling, A., Wang, H., Williams, B., Yoo, S., Yu, L., Zhilinsky, U., and Zhou, Z. $\pi^*_{0.6}$: a VLA that learns from experience, 2025. URL https://arxiv.org/abs/2511.14759.

Rowland, M., Bellemare, M. G., Dabney, W., Munos, R., and Teh, Y. W. An analysis of categorical distributional reinforcement learning, 2018. URL https://arxiv.org/abs/1802.08163.

Rudner, T. G. J., Pong, V. H., McAllister, R., Gal, Y., and Levine, S. Outcome-driven reinforcement learning via variational inference, 2021. URL https://arxiv.org/abs/2104.10190.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. A. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, 2014.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017. URL https://arxiv.org/abs/1712.01815.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT Press, Cambridge, MA, 1998.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., de Las Casas, D., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T., and Riedmiller, M. DeepMind control suite, 2018. URL https://arxiv.org/abs/1801.00690.

Teh, Y. W., Bapst, V., Czarnecki, W. M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N., and Pascanu, R. Distral: Robust multitask reinforcement learning, 2017. URL https://arxiv.org/abs/1707.04175.

Toussaint, M., Harmeling, S., and Storkey, A. Probabilistic inference for solving (PO)MDPs, 2006. URL https://homepages.inf.ed.ac.uk/amos/publications/ToussaintStorkey2006ProbabilisticInferenceSolvingMDPs.pdf. Edinburgh Informatics Research Report 0934.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need, 2017. URL https://arxiv.org/abs/1706.03762.

Wang, K., Javali, I., Bortkiewicz, M., Trzciński, T., and Eysenbach, B. 1000 layer networks for self-supervised RL: Scaling depth can enable new goal-reaching capabilities, 2025. URL https://arxiv.org/abs/2503.14858.

Watkins, C. J. C. H. and Dayan, P. Q-learning. *Machine Learning*, 8(3-4):279–292, May 1992. ISSN 0885-6125, 1573-0565. doi: 10.1007/BF00992698. URL http://link.springer.com/10.1007/BF00992698.

Yarats, D., Brandfonbrener, D., Liu, H., Laskin, M., Abbeel, P., Lazaric, A., and Pinto, L. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning, 2022. URL https://arxiv.org/abs/2201.13425.

Younis, O. G., Perez-Vicente, R., Balis, J. U., Dudley, W., Davey, A., and Terry, J. K. Minari. Zenodo, 2024. URL https://doi.org/10.5281/zenodo.13767625. Version 0.5.0.

Yu, T., Quillen, D., He, Z., Julian, R. C., Hausman, K., Finn, C., and Levine, S. Meta-World: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, 2019.

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C. Gradient surgery for multi-task learning, 2020. URL https://arxiv.org/abs/2001.06782.

A. Experiment Details

A.1. Evaluation protocol

For D4RL and ExORL we run four seeds per task (the β-KL variant of Section 5.2 uses three) and score every run at exactly 2M training steps. Because raw returns span very different scales across tasks, each task's returns are normalized by the best run observed on that task across all agents (per-task max $= 100$; the figures display the same quantity scaled to $[0, 1]$) before aggregating. The point estimate and 95% confidence interval come from a task-stratified bootstrap: seeds are resampled with replacement within each task, averaged per task, and tasks are weighted equally. Throughout the paper, the error bars and shaded bands in the figures and the bracketed intervals in the tables are these 95% bootstrap confidence intervals; the training-curve figures (Figures 3 and 4) use the same per-task normalization, with the best value logged on each task as the denominator. For MT80 we run a single seed per agent and report TD-MPC2's normalized score (Meta-World success rates and DeepMind Control returns rescaled to $[0, 100]$, averaged over the 80 tasks), with every agent scored at 3.8M training steps.

A.2. Training details

The Gaussian actor is entropy-regularized as in SAC/RLPD (Haarnoja et al., 2018; Ball et al., 2023): a Lagrangian dual temperature drives it toward a target entropy of -0.5 per action dimension, and the entropy term is *not* bootstrapped into the TD target. The value the actor maximizes is read from the critic at a reparameterized Gaussian sample; for the success function this value is the sigmoid probability $\sigma(z)$ or its logarithm $\log \sigma(z)$ (the log variant), the latter giving the Bayesian update of Section 4.4. On MT80 we use a separate entropy temperature per task (one of 80), so the entropy-regularization strength adapts per task and every task is held to the same per-dimension actor-entropy constraint despite differing action dimensions; in separate experiments we found this important for performance. The network forward pass runs in bf16, but all surrounding calculations are done in fp32.

A.3. Hyperparameters

All agents share: Adam (learning rate 3×10^{-4}), Polyak target updates ($\tau = 0.005$), discount $\gamma = 0.99$, a tanh-squashed Gaussian actor, a dual entropy temperature (target entropy -0.5 per action dimension, initial temperature 1.0), and MLP actor/critic with LayerNorm and residual connections. HL-Gauss and Distributional learn a categorical return distribution over the indicated number of bins. Tables 1 to 3 list the per-benchmark settings.

Table 1. D4RL (Minari) hyperparameters.

Tasks	5 (hammer-expert, pen-expert, pointmaze-medium-dense, halfcheetah-expert, hopper-expert)
Protocol	RLPD-style continual mixing ($50/50$ offline/online)
Training steps	2M (online)
Batch size	256
Critics (ensemble)	2 (mean aggregation)
n-step returns	$n = 1$
Actor MLP	512-d $\times$ 4 layers
Critic MLP	512-d $\times$ 32 layers
HL-Gauss / Distributional bins	51
Gradient clip norm	none
Seeds	4 (3 for β-KL)

A.4. Beta-distributional critic details

The Beta critic outputs two raw channels per ensemble member, one per Beta parameter. Each channel is mapped through a softplus and then clamped from below at 10^{-3}, keeping the online parameters $(\alpha_\theta, \beta_\theta)$ strictly positive; this clamp on the online network is distinct from the floor ν_{floor} on the projected target concentration (Section 4.6). Two critics are used, with the per-critic Beta moments at the next state aggregated by their mean. The actor is the same entropy-regularized Gaussian actor as in the main comparison, reading the Beta mean $\mu = \alpha/(\alpha + \beta)$ from the critic. Targets are 1-step, matching the D4RL setup ($n = 1$), and all remaining hyperparameters match Table 1. The target projection floor was swept over

Table 2. ExORL hyperparameters.

Tasks	3 (cheetah-run, quadruped-run, walker-run; `proto` datasets)
Protocol	two-stage offline→online
Training steps	1M offline + 1M online (2M total)
Batch size	1024
Critics (ensemble)	1
n-step returns	$n = 4$
Actor MLP	1024-d $\times$ 8 layers
Critic MLP	1024-d $\times$ 8 layers
HL-Gauss / Distributional bins	51 and 204
Gradient clip norm	100
Seeds	4

Table 3. MT80 hyperparameters.

Tasks	80 (50 Meta-World + 30 DeepMind Control)
Dataset	TD-MPC2 MT80, ≈545M transitions (offline only)
Codebase	forked TD-MPC2; learnable task embedding (96-d, $\ell_2 \leq 1$)
Training steps	3.8M
Batch size	4096
Critics (ensemble)	1
n-step returns	$n = 4$
Actor MLP	2048-d $\times$ 16 layers
Critic MLP	2048-d $\times$ 16 layers
HL-Gauss / Distributional bins	204
Gradient clip norm	100
Seeds	1

$\nu_{\text{floor}} \in \{4, 8\}$.

B. Full Results

Tables 4, 5, and 6 report per-task scores for all agents on D4RL, ExORL, and MT80.

Table 4. **D4RL/Minari results.** Return at 2M training steps, normalized per task so the best run on each task scores 100, averaged over 4 seeds (three for the β-KL variants). The Total row is the equal-weight mean over the 5 tasks, matching the aggregate bar figure (Figure 2). **Bold** marks agents within 95% of the row maximum. The Ps^+ β-KL columns are the Beta-KL variant of the success-function agent with concentration floor ν. Entries can be negative because some tasks have rewards with negative components, so a poorly performing run can accumulate a negative raw return, whose sign the normalization preserves.

	TD	HL-Gauss (51 bins)	Distributional (51 bins)	Ps^+	Ps^+ (log)	Ps^+ β-KL (ν=4)	Ps^+ β-KL (ν=8)
`hammer`	**92.9** [84,99]	**97.5** [95,99]	89.2 [69,100]	87.3 [67,100]	88.6 [74,99]	-1.4 [-1.5,-1.3]	-1.4 [-1.4,-1.4]
`pen`	52.4 [36,70]	77.0 [74,79]	75.9 [70,81]	71.9 [60,82]	**81.5** [70,94]	-7.7 [-8,-7]	-5.6 [-8,-1]
`pointmaze`	**79.0** [64,94]	62.3 [47,85]	71.1 [48,87]	73.8 [67,81]	**76.4** [61,93]	24.7 [1,71]	53.0 [50,55]
`halfcheetah`	**97.3** [94,99]	**96.6** [91,100]	**99.2** [99,100]	**99.6** [99,100]	**99.1** [98,100]	-2.7 [-3,-2]	37.1 [7,97]
`hopper`	39.3 [35,45]	40.1 [34,46]	**68.5** [51,89]	37.7 [26,51]	35.3 [27,43]	0.9 [0,2]	0.7 [0,1]
Total	72.2 [67,77]	74.7 [71,79]	**80.8** [74,88]	74.1 [68,80]	76.2 [71,81]	2.8 [-2,12]	16.7 [10,29]

Table 5. **ExORL results.** Return at the end of training (2M steps: 1M offline + 1M online; proto dataset), normalized per task so the best run on each task scores 100, averaged over 4 seeds. The Total row is the equal-weight mean over the 3 tasks, matching the aggregate bar figure (Figure 2). **Bold** marks agents within 95% of the row maximum.

	TD	HL-Gauss (51 bins)	HL-Gauss (204 bins)	Distributional (51 bins)	Distributional (204 bins)	Ps^+
`cheetah-run`	77.9 [74,83]	78.7 [77,80]	79.9 [76,85]	**90.8** [86,95]	**91.5** [86,97]	83.0 [79,87]
`quadruped-run`	82.5 [79,88]	86.7 [84,89]	80.6 [77,85]	**95.3** [95,96]	88.7 [83,95]	**94.5** [91,98]
`walker-run`	**94.7** [92,98]	**98.9** [98,99]	**96.7** [95,99]	**96.2** [96,97]	**96.7** [96,98]	**98.7** [97,100]
Total	85.0 [83,87]	88.1 [87,89]	85.7 [84,88]	**94.1** [92,95]	**92.3** [90,95]	**92.1** [90,94]

Table 6. **MT80 results.** TD-MPC2 normalized score (0–100) over the 80-task multitask benchmark, evaluated at 3.8M training steps. The runs are single-seed, so each cell is one run and no confidence intervals are shown. **Bold** marks agents within 95% of the row maximum. †External baselines from prior work.

	TD	HL-Gauss (204 bins)	Distributional (204 bins)	Ps^+	Ps^+ (log)	TD-MPC2†	BRC+BC†
Normalized score	75.8	**88.7**	**88.0**	83.4	82.8	70.6	64.5

C. Proofs

C.1. Proof of Theorem 4.4 (asymmetric reward bounds)

Proof. Write $c = \frac{R_-}{R_- + R_+}$ and $b = \frac{1}{R_- + R_+}$, so that the discount-termination transition enters s^+ with probability $c + b\, r(s,a)$ and terminal states enter s^+ with probability c. Following the decomposition in the proof of Theorem 4.2,

$$\begin{aligned} P^\pi(s^+ \mid s_0) &= \mathbb{E}_\pi\left[\sum_{t=0}^{\tau-1} \gamma^t (1-\gamma)\big(c + b\, r(s_t,a_t)\big) + \gamma^\tau c\right] \\ &= c(1-\gamma)\,\mathbb{E}_\pi\left[\sum_{t=0}^{\tau-1} \gamma^t\right] + b(1-\gamma)\,\mathbb{E}_\pi\left[\sum_{t=0}^{\tau-1} \gamma^t\, r(s_t,a_t)\right] + c\,\mathbb{E}_\pi[\gamma^\tau] \\ &= c\,\mathbb{E}_\pi[1-\gamma^\tau] + b(1-\gamma)\,V^\pi(s_0) + c\,\mathbb{E}_\pi[\gamma^\tau] \\ &= c + b(1-\gamma)\,V^\pi(s_0). \end{aligned}$$

□

C.2. Proof of Theorem 4.5 (Bayesian actor update)

Proof. $\mathbb{E}_\mu[\log P^\pi(s^+ \mid s, \cdot)]$ is linear in μ and $-\mathrm{KL}(\mu \,\|\, \pi(\cdot \mid s))$ is strictly concave in μ on $\Delta(\mathcal{A})$, so J_s is strictly concave. Form the Lagrangian for the constraint $\sum_a \mu(a) = 1$:

$$\mathcal{L}(\mu, \lambda) \;=\; \sum_a \mu(a)\left[\log P^\pi(s^+ \mid s,a) - \log \frac{\mu(a)}{\pi(a \mid s)}\right] \;-\; \lambda\left(\sum_a \mu(a) - 1\right).$$

The first-order condition is

$$\frac{\partial \mathcal{L}}{\partial \mu(a)} \;=\; \log P^\pi(s^+ \mid s,a) \;-\; \log \frac{\mu(a)}{\pi(a \mid s)} \;-\; 1 \;-\; \lambda \;=\; 0,$$

giving $\mu(a) \propto \pi(a \mid s)\, P^\pi(s^+ \mid s,a)$. The normalizer is $\sum_a \pi(a \mid s)\, P^\pi(s^+ \mid s,a) = P^\pi(s^+ \mid s)$. □