# Exploit Reward Shifting in Value-Based Deep RL: Optimistic Curiosity Exploration and Conservative Exploitation with Linear Reward Shaping

**Hao Sun**[1*] **Lei Han**[2] **Rui Yang**[3] **Xiaoteng Ma**[4] **Jian Guo**[5] **Bolei Zhou**[6]

## Abstract

In this work, we study the simple yet universally applicable case of reward shaping in value-based Deep Reinforcement Learning (DRL). We show that reward shifting in the form of a linear transformation is equivalent to changing the initialization of the $Q$-function in function approximation. Based on such an equivalence, we bring the key insight that a positive reward shifting leads to conservative exploitation, while a negative reward shifting leads to curiosity-driven exploration. Accordingly, conservative exploitation improves offline RL value estimation, and optimistic value estimation improves exploration for online RL. We validate our insight on a range of RL tasks and show its improvement over baselines: (1) In offline RL, the conservative exploitation leads to improved performance based on off-the-shelf algorithms; (2) In online continuous control, multiple value functions with different shifting constants can be used to tackle the exploration-exploitation dilemma for better sample efficiency; (3) In discrete control tasks, a negative reward shifting yields an improvement over the curiosity-based exploration method.

## 1 Introduction

While reward shaping is a well-established practice in reinforcement learning and has a long-standing history [1, 2], specifying a certain reward to incentivize the learning agent requires domain knowledge and a thorough understanding of the task [3–6]. Even with careful design and tuning, learning with a shaped reward that intends to accelerate learning may on the contrary hinder the learning performance by incurring the sub-optimal behaviors of the agent [7, 8]. Although Ng et al. [9] theoretically points out that optimal policy remains unchanged under certain form of reward transformation, and in the later work of Wiewiora et al. [10] a framework is proposed to guide policies with prior knowledge under tabular setting, how reward shifting accommodates recent Deep Reinforcement Learning (DRL) algorithms remains much less explored.

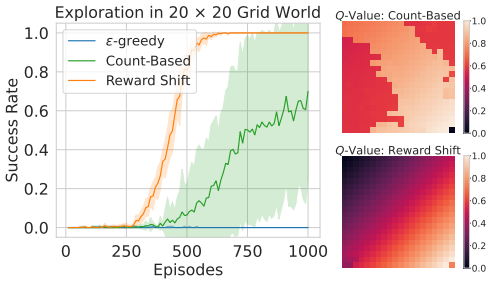


Figure 1: Our work is inspired by the observation that reward shifting remarkably helps exploration and outperforms count-based exploration in $Q$-learning (left). Reward shifting does not change the primal optimal $Q$-value landscape, and is able to learn a near-optimal $Q$-value (right). While count-based exploration suffers from the curse of dimensionality, reward shifting can be seamlessly applied to high-dim tasks including continuous control.

In this work, we study a special linear transformation, which is the simplest form of reward shaping, in value-based DRL [11–14]. We start with understanding how such a specific kind of reward shaping

*hs789@cam.ac.uk. [1]University of Cambridge; [2] Tencent RoboticsX; [3]Hong Kong University of Science and Technology; [4] Tsinghua University; [5] IDEA; [6] University of California, Los Angeles

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

works in value-based DRL function approximations and show that reward shifting is equivalent to different $Q$-value initialization, extending previous discovery of [10] to the function approximation setting. Figure 1 showcases reward shifting benefits exploration in a maze game and outperforms count-based and $\epsilon$-greedy exploration and learns near optimal value function.

Based on such an equivalence, we introduce the key insight of this work:

***A positive reward shifting leads to conservative exploitation, while a negative reward shifting leads to curiosity-driven exploration.***

We demonstrate the application of such an insight in **three Deep-RL scenarios: (S1) for offline RL**, we show that conservative exploitation induced by reward shifting improves learning performance of off-the-shelf algorithms; **(S2) for online RL setting**, we show multiple value functions with different reward shifting constants can be used to trade-off between exploration and exploitation, thus improve learning efficiency; **(S3) for curiosity-driven exploration,** we introduce a simple yet crucial adaptation on a prevailing curiosity-based exploration algorithm, the Random Network Distillation [15], making it compatible with value-based DRL algorithms. Experiments on a diverse set of tasks including continuous and discrete action space control show our method brings substantial improvement over baselines.

**Our contributions** can be summarized as follows:

1. Analytically, we introduce the key insight that reward shifting is equivalent to diversified $Q$-value network initialization in value-based DRL, which can be applied to curiosity-driven exploration and conservative exploitation;
2. Practically, we instantiate the key insight to three different scenarios, namely the offline conservative exploitation, sample-efficient continuous control, and curiosity-driven exploration, to contrast the generality of reward shifting;
3. Empirically, we demonstrate the effectiveness of the proposed method integrated with multiple off-the-shelf baselines on both continuous and discrete control tasks.

## 2 Preliminaries and Related Work

### 2.1 Online RL

We follow a standard MDP formulation in the online RL setting, i.e., $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \rho_0, \gamma, T\}$, where $\mathcal{S} \subset \mathbb{R}^d$ denotes the $d$-dim state space, $\mathcal{A}$ is the action space (note for discrete action space $|\mathcal{A}| < \infty$ and for continuous control $|\mathcal{A}| = \infty$), we consider a deterministic transition dynamics $\mathcal{T} : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$ and deterministic reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$. $\rho_0 = p(s_0) \in \Delta(\mathcal{S})$ denotes the initial state distribution. $\gamma$ is the discount factor and $T$ is the episodic decision horizon. Online RL considers the problem of learning a policy $\pi \in \Pi : \mathcal{S} \mapsto \Delta(\mathcal{A})$ (or $\pi \in \Pi : \mathcal{S} \mapsto \mathcal{A}$ with a deterministic policy class), such that the expected cumulative reward $\mathbb{E}_{a_t \sim \pi, s_{t+1} \sim \mathcal{T}, s_0 \sim \rho_0} \sum_{t=0}^{T} \gamma^t r_t(s_t, a_t)$ in the Markovian decision process is maximized. In online RL setting, an agent learns through trials and errors [11], through either on-policy [16–18] or off-policy paradigm [12–14, 19–21]. In this work, we focus on the off-policy value-based methods which are in general more sample efficient. Specifically, our discussions assume the policy learning is based on a learned $Q$-value function that approximates the cumulative reward an agent can gain in the following part of an episode. Formally, $Q(s_t, a_t) = \mathbb{E}_{\pi, \mathcal{T}} \sum_{\tau=t}^{T} \gamma^t r(s_\tau, a_\tau)$, and can be estimated by propagating the Bellman operator $\mathbb{B}Q(s, a) = r(s, a) + \gamma \mathbb{E} Q(s', a')$. For value-based methods, the (soft-)optimal policy is then produced by

$$\pi^*_\alpha(a|s) = \frac{\exp \frac{1}{\alpha} Q^*(s, a)}{\sum_{a'} \exp \frac{1}{\alpha} Q^*(s, a')}, \tag{1}$$

where $Q^*$ is the optimal $Q$-value function. We can also set the temperature parameter close to 0 to have the deterministic policy class. Simplifying the notion we have $\pi^*(s) = \arg\max_a Q^*(s, a)$. Algorithms like DPG [22] can be used to address the intractable analytical argmax issue that arises in continuous action space. We choose to develop our work on top of prevailing baseline algorithms of DQN [13], BCQ [23], CQL [24] and TD3 [14] as a minimal example to isolate the source of gains. It should be straightforward to extend the method on top of other learning algorithms.

Table 1: Reward shifting is flexible to be plugged into both online and offline RL algorithms to guarantee conservative exploitation or pursue optimistic exploration. It covers both discrete and continuous control tasks, with only a little additional computational expense. Moreover, the optimal policy learned with shifted reward is not biased.

| Covered Topics | Related Work | Plug-in | Online | Offline | Discrete | Continuous | Unbiased | Examples |
|---|---|---|---|---|---|---|---|---|
| Exploration | Curiosity | ✓ | ✓ | · | ✓ | · | · | Burda et al. [15] |
| | Ensemble | · | ✓ | · | ✓ | · | ✓ | Osband et al. [25] |
| | Initialization | · | ✓ | · | ✓ | · | ✓ | Rashid et al. [26] |
| | Optimism | · | ✓ | · | · | ✓ | · | Ciosek et al. [27] |
| Exploitation | Conservatism | · | · | ✓ | ✓ | ✓ | · | Bharadhwaj et al. [24] |
| | Policy Constraints | ✓ | · | ✓ | ✓ | ✓ | ✓ | Fujimoto et al. [23] |
| | Reward Shifting | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | Ours |

### 2.2 Exploration and the Curiosity-Driven Methods

One of the most important issues in online RL is the exploration-exploitation dilemma [11] that the agent must simultaneously learn to exploit its accumulated knowledge on the task and explore new states and actions. Previous works address the exploration problem from various perspectives: for discrete action space tasks, count-based methods like [28–30] are proposed to motivate the revisiting of under-explored state-action pairs. Specifically, Choshen et al. [31] extended the idea into general settings by constructing an additional MDP for Exploration-value estimation, as a generalized counter for count-based exploration. To boost exploration in continuous state tasks, curiosity-driven methods are investigated by Houthooft et al. [32], Pathak et al. [33], Burda et al. [34, 15], where variety of intrinsic rewards are designed as supplementary to the primal task reward for better exploration. Self-imitate approaches like Oh et al. [35], Ecoffet et al. [36], Sun et al. [37] repeat success trajectories but require extra assumptions on the environment.

DORA [31] constructed an additional MDP to estimate the Exploration-value as a generalised counter for count-based exploration, yet those count-based methods are orthogonal to reward shifting: in intrinsic reward methods, an agent must **first experience** a new $(s, a)$ pair before receiving a high intrinsic reward — this is extremely hard with an arg-max style policy. On the other hand, with optimistic initialization, the rarely-visited $(s, a)$ pairs will naturally have higher $Q$-values **before experiencing** it — as the frequently-visited pairs have updated their values with a negatively shifted reward. From such a perspective, reward shifting not only works by itself motivates exploratory behaviors, but can also be seamlessly plugged into intrinsic reward methods to **encourage the first visitation** of new states.

The works of DIAYN and DADS [38, 39] show that various skills can be developed even without the primal extrinsic reward. For continuous control tasks, OAC [27] improves the SAC [20] with informative action space noise based on the optimism in face of uncertainty (OFU) [40–43]. GAC [44] addresses the exploration issue with a richer functional class for the policy.

In the work of Rashid et al. [26], the problematic pessimistic initialization is addressed for better exploration, yet the work focuses on specific settings of tabular and discrete control exploration. In the work of Osband et al. [45, 25], ensemble models with diverse initialization and randomized priors are used to resemble the insight of bootstrap sampling and facilitate better value estimation, yet those methods are only applicable to discrete control tasks. Noted that although the reward shifting can be regarded as a special case of these random priors, it can be distinguished by not changing the optimal $Q$-value, and its flexibility to be plugged into both continuous and discrete control algorithms.

Random Network Distillation (RND) [15] defines the intrinsic reward as the output difference between a fixed neural network $\phi_1$ and a trainable network $\phi_2$ given state-actions as the inputs. e.g.,

$$r_{\text{int}}(s, a) = |\phi_2(s, a) - \phi_1(s, a)|, \tag{2}$$

where both networks are activated by a sigmoid function. After optimizing the learnable $\phi_2$ to approximate $\phi_1$ with seen $(s, a)$ pairs [2], the value of $r_{\text{int}}(s, a)$ will decay to 0 for such state-action pairs that are frequently visited but remain high for thoase are seldom visited.

In this work, we show that exploratory behavior can be achieved simply by shifting the reward function with a constant. Thus our method is orthogonal to those previous approaches in the sense that our intrinsic exploration behavior is motivated by high function approximation error in the

[2] Or computed with only states, i.e., $r_{\text{int}}(s) = |\phi_2(s) - \phi_1(s)|$.

under-explored state-action pairs. We demonstrate such insight by showing that RND in its original design is not suitable for value-based methods in developing exploratory behaviors, but integrating RND with a shifted reward function remarkably improves the learning performance.

### 2.3 Offline RL

The offline RL, also known as batch-RL, focuses on the problems where interaction with the environment is impossible, and the policy can only be optimized based on the logged dataset. In those tasks, a fixed buffer $\mathcal{B} = \{s_i, a_i, r_i, s'_i\}_{i=[N]}$, collected from some unknown behavior policy $\pi_\beta$, is provided. In general, such a dataset can either be generated by rolling out an expert that generates high-quality solutions to the task [23, 46, 47] or a non-expert that executes sub-optimal behaviors [47–51] or be a mixture of both [24]. As the agent in the offline RL setting can not correct its potentially biased knowledge through interactions, the most important issue is to address the extrapolation error [23] induced by distributional mismatch [49]. To address such an issue, a series of algorithms optimize the policy learning under the constraint of distributional similarity [48, 49, 52, 53].

Bharadhwaj et al. [24] propose CQL to solve the offline RL tasks with a conservative value estimation. Specifically, CQL learns the $Q$-value estimation by jointly maximizing the $Q$-values of actions sampled from the behavior offline dataset and minimizing the $Q$-values of actions sampled with pre-defined prior distributions (e.g., uniform distribution over the action space). As we will show in this work, an alternative approach to have a lower bound for the optimal $Q$-value function is to use an appropriately shifted reward function. This idea leads to the direct application of our proposed framework in the offline setting. In general, reward shift can be plugged in many distribution-matching offline-RL algorithms [23, 48, 49, 52] to further improve the performance with conservative $Q$-value estimation.

Table 1 contextualizes reward shifting with respect to related works we discussed above.

## 3 A Motivating Example

We start with two intuitive remarks and a "counter-intuitive" motivating example in this section before introducing our method.

**Remark 1.** Given an MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \rho_0, \gamma\}$, where $|\mathcal{A}| < \infty$, scaling the reward function with linear transformation, i.e., $\mathcal{R}_{k,b} = k \cdot \mathcal{R} + b, \forall k > 0, b \in \mathbb{R}$, such that $r'_t = kr_t + b \in \mathcal{R}_{k,b}$, does not change the optimal policy induced by the corresponding value function $Q^*_{k,b}(s,a) := \sum_t \gamma^t r'_t$:

$$\pi^*(s) = \arg\max_{a\in\mathcal{A}} Q^*_{k,b}(s,a) = \arg\max_{a\in\mathcal{A}} kQ^*(s,a) + \frac{b}{1-\gamma} = \arg\max_{a\in\mathcal{A}} Q^*(s,a), \tag{3}$$

**Remark 2.** When $|\mathcal{A}| = \infty$, scaling the reward function with linear transformation does not change the optimal policy induced by deterministic policy gradient [22], given proper learning rate $\eta' = \eta/k$:

$$\nabla_\theta J(\mu_\theta) = \mathbb{E}_{s_t}[\nabla_a Q^*(s_t,a_t)|_{a_t=\mu_\theta(s_t)} \nabla_\theta \mu_\theta(s_t)] = \mathbb{E}_{s_t}[\nabla_a Q^*_{k,b}(s_t,a_t)|_{a_t=\mu_\theta(s_t)} \nabla_\theta \mu_\theta(s_t)]/k, \tag{4}$$

Remark 1 and Remark 2 declare the fact that constant reward shifting does not affect the optimal policy induced by the optimal $Q$-value function calculated by the shifted reward. However, Figure 1 presents "counter-intuitive" results in a demonstrative exploration task of Grid World. In this task, an agent located at the upper left corner of a map needs to explore without reward and reach the goal point located at the lower right corner. A non-trivial reward of $+1$ will be assigned only when the goal is reached. We report learning curves with regard to the episodic success rate in reaching the goal point and the learned $Q$-values. In this toy example, we find a negative reward shifting remarkably boosts the learning efficiency of $Q$-learning, and surpasses the conventional count-based method for exploration. Moreover, Remark 1 is empirically verified with such a toy example: reward shifting does not change the optimal $Q$-value as well as its induced policy, but on the contrary accelerates the discovery of (near-)optimal policy.

In the following of this work, we study the effect of varying constant bias $b$ and fix the scaling factor $k = 1$ to avoid trivial discussions on the choices of learning rate — there should be no surprise that choosing an appropriate learning rate is empirically important. We focus on revealing the importance of selecting the universal bias term $b$ in the reward function through the lens of initialization priors in function approximation and show such a bias is generally helpful for both online and offline

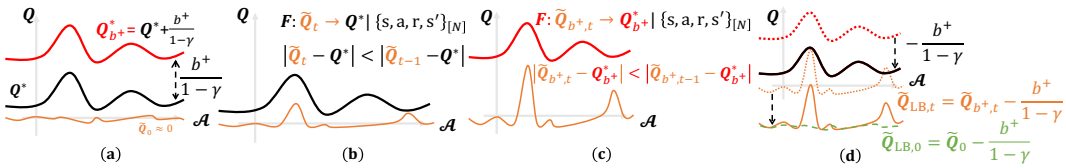


Figure 2: Illustrative figure for conservative exploitation, with a positive constant bias added to the reward function. We use **Black** lines to denote the original value function, and use **Red** lines to denote the shifted value function with the constant reward shift. **Orange** lines are used to denote the function approximation during different learning stages and **Green** line shows the equivalent **pessimistic initialization**. **(a)** shifting the reward function with a positive bias term $b^+$ leads to an uniformly increased $Q$-value function, namely $Q^*_{b^+} = Q^* + \frac{b^+}{1-\gamma}$, during learning, a neural network estimator $\tilde{Q}$ initialized with $\tilde{Q}_0 \approx 0$ is optimized to approximate the $Q$-value functions (e.g., through TD or MC estimation). **(b)** for any value-based RL algorithm, the value optimization step can be regarded as a function $F$ that minimizes the difference between the estimated $Q$-value function $\tilde{Q}_t$ and the optimal one $Q^*$, given the interaction experience with the environment (e.g., a replay buffer $\mathcal{B}$ for off-policy methods). Note $\tilde{Q}_t$ approximating $Q^*$ better than $\tilde{Q}_{t-1}$ holds in expectation as long as the RL algorithm is designed to approximate the optimal value function. **(c)** similarly, the optimizer given the same interactive experience (e.g., replay buffer $\mathcal{B}$) will learn to minimize the difference between $Q$-value function $\tilde{Q}_{b^+,t}$ and the optimal one $Q^*_{b^+}$, after re-labeling the rewards in the buffer by $r' = r + b^+$. **(d)** according to Remark 2, the optimization conducted in (c) is equivalent to (b) with the neural network $Q$-value estimator initialized as $\tilde{Q}_{\text{LB},0} \approx \tilde{Q}_0 - \frac{b^+}{1-\gamma}$, rather than $\tilde{Q}_0 \approx 0$. i.e., by shifting the reward with proper positive value $b^+$, we are able to initialize the $Q$-value network that lower-bounds the optimal $Q$-value.

settings. In the online settings, it improves learning efficiency, and in the offline settings, it promotes conservative exploitation.

# 4 Shifted Priors for $Q$-value Estimation

## 4.1 Reward Shift Equals to Different Initialization

We start by formally introducing notions and the key idea of this work: reward shifting equals different initialization. We use Figure 2 to illustrate how reward shifting affects function approximation, hence changing the learning dynamics for value-based algorithms. The original optimal $Q$-value function is denoted as $Q^*$, and plotted in the figures as **Black** curves. We then denote the shifted optimal $Q$-value function as $Q^*_{b^+}$, which is the $Q$-value function with the shifted reward $r' = r + b^+$. We use **Red** curves in figures to denote those shifted $Q$-value functions. In this section we provide analysis with a positive bias $b^+ > 0$ to illustrate how positively shifted value function affects function approximation, and coordinates the normally applied near-zero initialized function approximators (e.g., neural networks. **Orange** curves in the figure) to inspire conservative behaviors. The discussion of negative biases is elaborated in the next section.

To summarize, shifting the reward function with a positive constant is equivalent to initializing the value network with a smaller value — as the $Q$-value of unseen state-action pairs during training are much lower than their shifted optimal values, those actions will not be selected in argmax-style policy updates — leading to conservative learning behaviors that benefit policy learning in offline settings.

## 4.2 (S1) Offline RL: Conservative Exploitation

According to the key insight we presented in Section 4.1, conservative $Q$-value estimation can emerge with a positively shifted reward. And such a value estimation empirically lower-bounds the optimal $Q$-value function. As shown in Figure 2(a), a positive constant $b^+$ added to the reward function will lead to a universal positively shifted optimal $Q$-value function, and the gap between the primal $Q$-value function and the new one is $\frac{b^+}{1-\gamma}$. Optimizing the $Q$-value approximator with logged data will minimize the difference between the predicted value and the optimal value with observed data. For the unobserved data, the pessimistically initialized $Q$-value approximator guarantees the prediction is lower than the positively shifted optimal ones, thus conservative exploitation can be achieved with argmax-style value propagation and action execution on such a value function.

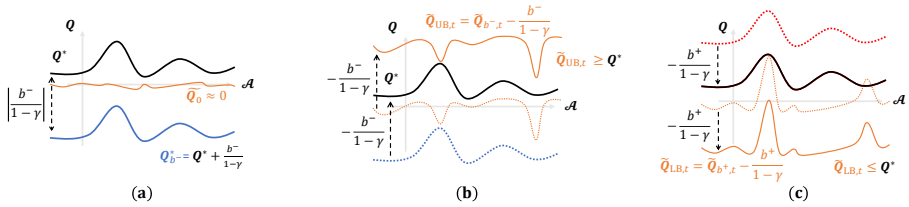


Figure 3: Illustrative figure for curiosity-driven exploration with a negative shifted reward. **(a)** While adding a negative constant value $b^-$ on the reward function leads to negatively shifted optimal $Q$-value function $Q^*_{b^-}$. **(b)** Minimizing the difference between a $Q$-value approximator and the optimal $Q$-value will enable calculating an upper-bound estimation for $Q^*$ which can be used for **optimistic exploration**. **(c)** A positive constant shift added to the reward function can be used for **conservative policy update**.

### 4.3 (Curiosity-Driven) Optimistic Exploration

On the other hand, if we shift the reward function to the negative side, it is equivalent to an optimistic initialization. Figure 3 (a-b) illustrate how adding a negative bias leads to curiosity-driven exploration. With sufficiently small $b^-$ (so that $|b^-|$ larger than the maximal value of any $(s, a)$-pair), such an upper bound of $Q^*$ can be used to conduct curiosity-driven exploration. Intuitively, initializing a value network that always predicts a value larger than the true optimal value will lead to curiosity-driven exploration behavior, as any visited state will be assigned a relatively smaller value and the argmax-style policy will tend to choose under-explored actions.

Based on the analysis above that (1) a positive constant shift added to the reward function can be used for conservative policy updates and (2) a negative constant shift added to the reward function can be used for curiosity-driven exploration, we are ready to access both the upper bound, i.e., the optimistic estimation with $b^-$, and the lower bound, i.e., the conservative estimation with $b^+$ of the optimal value function. Formally, we use

$$\tilde{Q}_{\mathrm{LB},t}(s,a) = \tilde{Q}_{b^+,t}(s,a) - \frac{b^+}{1-\gamma} \tag{5}$$

to denote the empirical lower bound of value estimation (cf. Figure 2(d), **Orange** line), and

$$\tilde{Q}_{\mathrm{UB},t}(s,a) = \tilde{Q}_{b^-,t}(s,a) - \frac{b^-}{1-\gamma} \tag{6}$$

to denote the empirical upper bound of value estimation (cf. Figure 3(b), **Orange** line). In both notions, $t$ denotes the optimization step. When $t = 0$, with a near-zero initialization $\tilde{Q}_0 \approx 0$, $\tilde{Q}_{\mathrm{LB},0}(s,a) = -\frac{b^+}{1-\gamma}$ is able to lower bound the unknown optimal $Q$-value given sufficiently large $b^+$. (cf. Figure 2(d), **Green** line). Similarly, $Q$-value is upper-bounded by $\tilde{Q}_{\mathrm{UB},0}(s,a)$ with a sufficiently small $b^-$.

Following those notions, we introduce our sample-efficient algorithms for both continuous control and discrete action space respectively. We propose a practical algorithm for sample-efficient continuous control in Section 4.3.1, and focus on a special class of curiosity-driven exploration method, the RND [15], in Section 4.3.2.

#### 4.3.1 (S2) Online RL: Trading-off Exploration and Exploitation with Reward Shift

According to the principle of optimism in the face of uncertainty (OFU), an exploration bonus that manifests the uncertainty of the $Q$-value function can be introduced into the value estimation: integrating optimistic exploration with conservative exploitation,

$$\begin{aligned}\hat{Q}(s,a) &= \tilde{Q}_{\mathrm{LB},t}(s,a) + \beta[\tilde{Q}_{\mathrm{UB},t}(s,a) - \tilde{Q}_{\mathrm{LB},t}(s,a)] \\ &= (1-\beta)(\tilde{Q}_{b^+,t}(s,a) - \frac{b^+}{1-\gamma}) + \beta(\tilde{Q}_{b^-,t}(s,a) - \frac{b^-}{1-\gamma}) \\ &= (1-\beta)\tilde{Q}_{b^+,t}(s,a) + \beta\tilde{Q}_{b^-,t}(s,a) - \frac{(1-\beta)b^+ + \beta b^-}{1-\gamma},\end{aligned} \tag{7}$$

where the second term with coefficient $\beta$ denotes exploration bonus that is composed of uncertainty.

For those under-explored state-action pairs, i.e., extremely out-of-distribution samples for our neural network, both $\tilde{Q}_{b^+,t}(s,a)$ and $\tilde{Q}_{b^-,t}(s,a)$ will give near-zero predictions as a consequence of initialization (detailed implications are provided in Appendix C). Hence, the explorative bonus becomes $-\frac{(1-\beta)b^+ + \beta b^-}{1-\gamma}$, which is equivalent to applying another constant reward shift with value of $c_r = (1-\beta)b^+ + \beta b^-$, formally, we have

**Proposition 4.1.** *Assuming we have access to an unbiased estimator for the optimal value function $Q^*$, e.g., with Monte-Carlo estimation $\hat{Q}^* = \mathbb{E}\sum_t \gamma^t r$, and the optimization is based on minimizing the MSE between the unbiased estimator and the function approximator, i.e., $\epsilon_t^2 = (\tilde{Q}_t - \hat{Q}^*)^2$, $\tilde{Q}_t = \tilde{Q}_{t-1} - 2\eta(\tilde{Q}_{t-1} - \hat{Q}^*)$, then the linear combination in Equation (7) is equivalent to a linear combination of the constants with value of $c_r = (1-\beta)b^+ + \beta b^-$.*

The proof can be found in the Appendix B. According to Proposition 4.1, a grid search for trading-off between the three hyper-parameters, i.e., the exploration bias $b^-$, the exploitation bias $b^+$ and the coefficient $\beta$ in Equation (7) is trivial as they only lead to a linear combination as $c_r = (1-\beta)b^+ + \beta b^-$, indicating that

**Corollary 4.2.** *Changing the reward shifting constant $b$ is sufficient to trade off between exploration and exploitation.*

The corollary says, in principle, a meta-learner can be trained to monitor the learning process and select a proper reward shifting constant automatically [54, 55] to balance exploration and exploitation. For the ease of exposition, in this work we choose to focus on the simplest yet effective uniform sampling strategy from multiple shift constants, which has been shown as a strong baseline of those meta-learner approaches [56, 57], and leave more complicated meta-learner-based shifting constant adjustment for future investigation.

Specifically, we use multiple Q-networks to learn with transition tuples $(s, a, r, s')$ sampled from the identical buffer that collects the policy's historcal interactions with the environment. In propagating Q-values through the temporal difference loss, the primal recorded reward value $r$ is replaced with shifted rewards with *different* constant biases to update their individual Q-networks. We then uniformly sample one of those learned Q-networks for the optimization of policy networks (e.g., with DPG [22]). It is worth noting that our approach only requires post-hoc revision of the primal reward function, rather than interacting with the environment multiple times to collect samples for each value network. We dub the proposed method Random Reward Shift (RRS), and provide the pseudo-code in Algorithm 1 of Appendix D.2.

#### 4.3.2 (S3) Compatible Curiosity: Tailored Curiosity-Driven Exploration for Value-Based RL

In previous works, the curiosity-driven exploration methods are always work with policy-based methods. In this section, we cast the key insight introduced above to RND [15] — one of the leading algorithms for exploration — to answer why is its vanilla design not suitable for value-based algorithms like Q-learning, and propose a variant of RND that is tailored for DQN.

The vanilla RND use two randomly initialized networks $\phi_1$ and $\phi_2$ to generate the intrinsic reward $r_{\text{int}} = |\phi_1(s,a) - \phi_2(s,a)| \geq 0$ for exploration. During learning, $\phi_2$ is a fixed network and the parameters of $\phi_1$ is optimized to approximate the output of $\phi_2$ for frequently-visited states. We follow Burda et al. [15] to bound the intrinsic reward below 1 and use $r_{\text{int},t}$ to denote the intrinsic reward after $t$ step of optimization. Specifically, $r_{\text{int},0}(s,a) = 1, \forall(s,a)$ — an universally positive bonus is added to the primal reward function at beginning.

According to our analysis in previous sections, such a positive reward shift is equivalent to pessimistic initialization and will lead to conservative behaviors in $Q$-value estimation. Therefore, the exploration behaviors at the beginning of learning will be hindered, rather than boosted. To overcome such a conservative behavior induced by the pessimistic initialization, we proposed to use $r_{\text{int}}^-(s,a) = |\phi_1(s,a) - \phi_2(s,a)|^2 - I, \forall s, a$, where $I$ is a positive constant that assures $r_{\text{int}}^- \leq 0$ is negatively initialized for optimistic exploratory behaviors.

**The Chicken-and-Egg Problem** To further understand the difference between the curiosity-driven method in ways of intrinsic reward and our reward shifting-based optimistic initialization, let us consider when do those curiosities work in each case: for intrinsic reward methods, an agent must **first experience** a new $(s, a)$ pair before receiving a high intrinsic reward — this is extremely hard

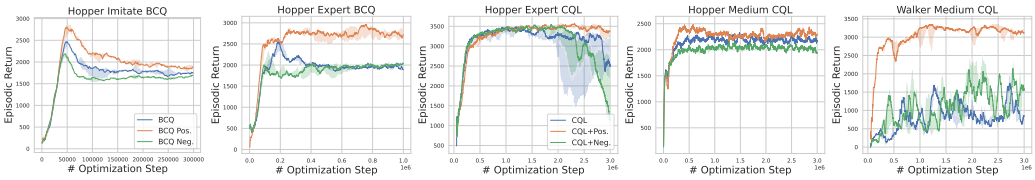


Figure 4: Results on offline RL settings. We verify our key insight that a positive reward shift equals to conservative exploitation thus helps offline value estimation, while a negative reward shift leads to worse performance. Results are from 10 runs with shaded areas indicating the $25\%$-$75\%$ quantiles.

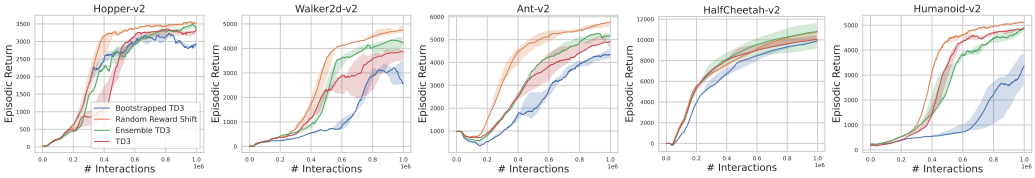


Figure 5: Results on continuous control tasks, Random Reward Shift (RRS) outperforms its value-based baselines in most environments. Results are from 10 runs with shaded areas indicating the $25\%$-$75\%$ quantiles.

with an arg-max style policy. On the other hand, with optimistic initialization, the rarely-visited $(s, a)$ pairs will naturally have higher $Q$-values **before experiencing** it — as the frequently-visited pairs have updated their values with a negatively shifted reward. From such a perspective, reward shifting not only works by itself motivates exploratory behaviors, but can also be seamlessly plugged into intrinsic reward methods to **encourage the first visitation** of new states.

# 5 Experiments

## 5.1 (S1): Offline RL with Conservative $Q$-value Estimation

**Experiment Setup** We start our experiments by demonstrating the effectiveness of reward shifting in offline RL benchmarks. As discussed in Section 4.2, a positive reward shift is equivalent to pessimistic initialization that benefits conservative exploitation. In general, our proposed method can be plugged in to any off-the-shelf offline RL algorithm, we choose to verify the effectiveness and generality of such a conservative $Q$-value estimation based on BCQ [23] and CQL [24], i.e., both distribution-matching approach and conservative value estimation approaches in offline RL. To verify our insight, we experiment with both positive reward shift (**Pos.**) and negative reward shift (**Neg.**), added on either BCQ or CQL.

**Results** Figure 4 shows our experiment results. We experiment with both the dataset generated in [23] (Hopper Imitate) and the dataset used in [47] (others), and find in our experiments that learning with the CQL dataset is much more stable. The first two panels show results with BCQ as the backbone algorithm. We observe a clear performance drop during the training of BCQ as the policy overfits the batched dataset. Differently, positive reward shift can alleviate such a problem and outperform BCQ in terms of both best-achieved performance and performance after convergence. The following three panels in Figure 4 use CQL as the backbone. Implementation details and more results can be found in Appendix D.1.

**Take-Away Message** In all experiments, shifting the reward with a positive constant improves learning performance, while a negative reward shift impedes to efficient learning — as expected.

## 5.2 (S2): Online RL with Randomized Priors

**Experiment Setup** We then conduct experiments in the MuJoCo locomotion benchmarks to demonstrate reward shifting improves learning efficiency in the online RL settings. As our implementation is based on TD3, we use TD3-based variants as our baselines: The **TD3** is trained with default settings according to Fujimoto et al. [14]. We also include **Ensemble TD3** and **Bootstrapped TD3** as baselines due to they are similar to our work in using multiple $Q$-networks in value estimation.

We follow Osband et al. [45] but extend it to the continuous control settings. In continuous control settings, the argmax operator is approximated by the policy network, and multiple policy networks are needed to cooperate with the multiple bootstrapped $Q$-value networks. Otherwise, multiple $Q$-value networks are not independent of each other thus breaking the condition of bootstrapped value estimation. The Ensemble TD3 presents the baseline performance when multiple $Q$-networks are used for value estimation in TD3, and works as an ablation of our method where all shift priors are set to be $0$.

As has been illustrated in Sec. 4.3.1, learning with different reward shifting values is equivalent to learning with optimistic or conservative initialization. In our instantiating of RRS, we use $3$ $Q$-networks with different priors. We empirically find $\pm 0.5, 0$ work universally good for all environments. Though, future investigation on hyper-parameter may help to further improve the performance.

**Results** Results are shown in Figure 5. RRS outperforms the vanilla TD3 in all five environments and outperforms all baseline methods in most tasks. In the experiment of Bootstrapped TD3 and Ensemble TD3, we also use $3$ $Q$-networks for a fair comparison. Note that there is a trade-off between computational complexity and sample efficiency, i.e., using more $Q$-networks may further improve the performance at the cost of more computational expenses, as reported in [45]. More implementation details, pseudo-code of RRS, and ablation studies can be found in Appendix D.2.

**Take-Away Message** Shifting the reward function can trade-off between exploration and exploitation. The ensemble performance of multiple value networks with random reward shifting drastically improve learning efficiency in continuous control.

### 5.3 (S3): Optimistic Random Network Distillation

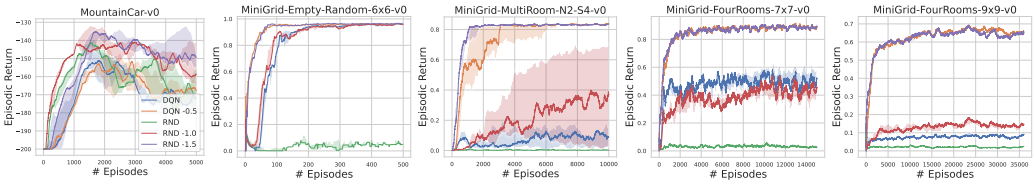


Figure 6: Value-based RND with shifted prior: Plugging the vanilla RND into DQN is not well-motivated according to our analysis in Section 4.3.2. The insight of equivalence between negative reward shifting and curiosity-driven exploration motivates us to negatively shift the intrinsic reward of RND, which drastically improves DQN-based RND. Results are from $10$ runs with shaded areas indicating the $25\%$-$75\%$ quantiles.

**Experiment Setting** To demonstrate the key insight and effectiveness of reward shifting in optimistic exploration, we benchmark with five discrete exploration tasks, including the classic MountainCar control and four MiniGrid navigation tasks [58], environment details can be found at Appendix D.3.

**Results** Figure 6 presents the results of $5$ different methods for comparison: besides the as-is ①**DQN** and ②**RND** baselines, we use ③**DQN -0.5** to indicate DQN with a $-0.5$ reward shift, and ④**RND -1.0** ⑤ **RND -1.5** to indicate RND with $-1.0, -1.5$ reward shift, respectively. In the following, we use $\succ$ between methods to indicate the former outperforms the latter. Comparing the results:

**1. reward shifting is equivalent to optimistic initialization, thus boosting exploration**
③ $\succ$ ①: a negative reward shift is equivalent to optimistic initialization and helps exploration.

**2. RND is effective for value-based exploration — as long as a negative intrinsic reward is used**
① $\succ$ ②: vanilla RND with positive intrinsic reward is always worse than DQN: adding a positive intrinsic reward like RND is harmful for exploration as it is equivalent to a pessimistic initialization.
④ $\succ$①; ⑤ $\succ$ ③: RND is effective for exploration (i.e., improve over DQN) as long as the intrinsic reward bonus is always negative.

**3. reward shifting for optimistic initialization is orthogonal to other exploration methods**
⑤ $\succ$ ④: increasing the magnitude of the negatively shifted reward can further improve the exploration performance — reward shifting can either work in isolation or get combined with other exploration algorithms as they are working orthogonally. [3]

**Take-Away Message** Reward shifting is equivalent to optimistic initialization, it can help with exploration in value-based methods. Importantly, such an intrinsic motivation is orthogonal to previous count-based methods, such that it can either work in isolation or be combined with conventional curiosity-driven exploration methods.

## 6 Conclusion

In this work, we studied how reward shifting affects policy learning in value-based deep reinforcement learning algorithms. Although constant reward shift should not change the optimal policy induced by the optimal value function, in practice such a constant shift *does affect the function approximation*, and leads to different learning behaviors. Our detailed analysis manifests the fact that a constant reward shift is equivalent to using different initialization in the value function approximation. The proposed idea is then verified through a variety of application settings. Specifically, we show that a negative reward shift leads to curiosity-driven exploration, while a positive reward shift helps conservative exploitation. Importantly, our analysis reveals that changing reward shifting constant itself is sufficient in trading-off between exploration and exploitation. We empirically verify the effectiveness of the proposed method in a variety of experiments, including better exploitation in offline RL, sample-efficient learning in continuous control benchmarks, and enhanced curiosity-driven exploration in value-based discrete control.

While our experiments demonstrate the performance gain is quite robust to the shifting constant, we would like to point out that the theoretical guidance for such a shifting constant is missing. Potential solutions may lie in analysis from the perspective of optimization for black-box models, yet it is out of the scope of the current empirical study and left for future research.

## 7 Acknowledgement

We thank all anonymous reviewers, ACs, PCs for their efforts and time in the reviewing process and in improving our paper. This work is done with the warm supports from the MMLab members. We acknowledge the insightful discussions with Ziping Xu, the Hot Spring Harbor group and the van der Schaar Lab members in improving the presentations of this paper. We thank Takuya Kanazawa in pointing out the concurrent work of Dubey et al. [59] that also discusses the effects of shifting terms in reward function.

[3] Code is available at GitHub

## References


[1] Jette Randløv and Preben Alstrøm. Learning to drive a bicycle using reinforcement learning and shaping. In *ICML*, volume 98, pages 463–471. Citeseer, 1998.

[2] Adam Daniel Laud. *Theory and application of reward shaping in reinforcement learning*. University of Illinois at Urbana-Champaign, 2004.

[3] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[4] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[5] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[6] Mahmoud Elbarbari, Kyriakos Efthymiadis, Bram Vanderborght, and Ann Nowé. Ltlf-based reward shaping for reinforcement learning. In *Adaptive and Learning Agents Workshop 2021*, 2021.

[7] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pages 482–495. PMLR, 2017.

[8] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

[9] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287, 1999.

[10] Eric Wiewiora, Garrison W Cottrell, and Charles Elkan. Principled methods for advising reinforcement learning agents. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 792–799, 2003.

[11] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. 1998.

[12] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[14] Scott Fujimoto, Herke Van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[15] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[16] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[18] Karl W Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. Phasic policy gradient. In *International Conference on Machine Learning*, pages 2020–2027. PMLR, 2021.

[19] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016.

[20] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[21] Hao Sun, Ziping Xu, Yuhang Song, Meng Fang, Jiechao Xiong, Bo Dai, and Bolei Zhou. Zeroth-order supervised policy improvement. *arXiv preprint arXiv:2006.06600*, 2020.

[22] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014.

[23] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. *arXiv preprint arXiv:1812.02900*, 2018.

[24] Homanga Bharadhwaj, Aviral Kumar, Nicholas Rhinehart, Sergey Levine, Florian Shkurti, and Animesh Garg. Conservative safety critics for exploration. *arXiv preprint arXiv:2010.14497*, 2020.

[25] Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 8617–8629, 2018.

[26] Tabish Rashid, Bei Peng, Wendelin Boehmer, and Shimon Whiteson. Optimistic exploration even with a pessimistic initialisation. *arXiv preprint arXiv:2002.12174*, 2020.

[27] Kamil Ciosek, Quan Vuong, Robert Loftin, and Katja Hofmann. Better exploration with optimistic actor critic. In *Advances in Neural Information Processing Systems*, pages 1785–1796, 2019.

[28] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29:1471–1479, 2016.

[29] Georg Ostrovski, Marc G Bellemare, Aäron Oord, and Rémi Munos. Count-based exploration with neural density models. In *International conference on machine learning*, pages 2721–2730. PMLR, 2017.

[30] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *31st Conference on Neural Information Processing Systems (NIPS)*, volume 30, pages 1–18, 2017.

[31] Leshem Choshen, Lior Fox, and Yonatan Loewenstein. Dora the explorer: Directed outreaching reinforcement action-selection. *arXiv preprint arXiv:1804.04012*, 2018.

[32] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Variational information maximizing exploration. 2016.

[33] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17, 2017.

[34] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.

[35] Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. *arXiv preprint arXiv:1806.05635*, 2018.

[36] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

[37] Hao Sun, Zhizhong Li, Xiaotong Liu, Bolei Zhou, and Dahua Lin. Policy continuation with hindsight inverse dynamics. In *Advances in Neural Information Processing Systems*, pages 10265–10275, 2019.

[38] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

[39] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

[40] Ronen I Brafman and Moshe Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct):213–231, 2002.

[41] Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(Apr):1563–1600, 2010.

[42] Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 263–272. JMLR. org, 2017.

[43] Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. Is q-learning provably efficient? In *Advances in Neural Information Processing Systems*, pages 4863–4873, 2018.

[44] Chen Tessler, Guy Tennenholtz, and Shie Mannor. Distributional policy optimization: An alternative approach for continuous control. *arXiv preprint arXiv:1905.09855*, 2019.

[45] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems*, pages 4026–4034, 2016.

[46] Chi Zhang, Sanmukh Rao Kuppannagari, and Viktor Prasanna. Brac+: Going deeper with behavior regularized offline reinforcement learning. 2020.

[47] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[48] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[49] Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *arXiv preprint arXiv:1906.00949*, 2019.

[50] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pages 104–114. PMLR, 2020.

[51] Daniel Jarrett, Alihan Hüyük, and Mihaela Van Der Schaar. Inverse decision modeling: Learning interpretable representations of behavior. In *International Conference on Machine Learning*, pages 4755–4771. PMLR, 2021.

[52] Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

[53] Rui Yang, Chenjia Bai, Xiaoteng Ma, Zhaoran Wang, Chongjie Zhang, and Lei Han. Rorl: Robust offline reinforcement learning via conservative smoothing. *arXiv preprint arXiv:2206.02829*, 2022.

[54] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[55] Rémy Portelas, Cédric Colas, Katja Hofmann, and Pierre-Yves Oudeyer. Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments. In *Conference on Robot Learning*, pages 835–853. PMLR, 2020.

[56] Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *international conference on machine learning*, pages 1311–1320. PMLR, 2017.

[57] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher–student curriculum learning. *IEEE transactions on neural networks and learning systems*, 31(9):3732–3740, 2019.

[58] Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid`, 2018.

[59] Rachit Dubey, Thomas L Griffiths, and Peter Dayan. The pursuit of happiness: A reinforcement learning perspective on habituation and comparisons. 2022.

[60] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pages 449–458. PMLR, 2017.

[61] Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[62] Clare Lyle, Marc G Bellemare, and Pablo Samuel Castro. A comparative analysis of expected and distributional reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4504–4511, 2019.

[63] Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva Tb, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*, 2018.

[64] Will Dabney, Zeb Kurth-Nelson, Naoshige Uchida, Clara Kwon Starkweather, Demis Hassabis, Rémi Munos, and Matthew Botvinick. A distributional code for value in dopamine-based reinforcement learning. *Nature*, 577(7792):671–675, 2020.

[65] Núria Armengol Urpí, Sebastian Curi, and Andreas Krause. Risk-averse offline reinforcement learning. *arXiv preprint arXiv:2102.05371*, 2021.

[66] Grégoire Delétang, Jordi Grau-Moya, Markus Kunesch, Tim Genewein, Rob Brekelmans, Shane Legg, and Pedro A Ortega. Model-free risk-sensitive reinforcement learning. *arXiv preprint arXiv:2111.02907*, 2021.

[67] Richard Y Chen, Szymon Sidor, Pieter Abbeel, and John Schulman. Ucb exploration via q-ensembles. *arXiv preprint arXiv:1706.01502*, 2017.

[68] Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *International Conference on Machine Learning*, pages 6131–6141. PMLR, 2021.

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

[70] Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized ensembled double q-learning: Learning fast without a model. *arXiv preprint arXiv:2101.05982*, 2021.

[71] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

[72] Hao Sun, Boris van Breugel, Jonathan Crabbe, Nabeel Seedat, and Mihaela van der Schaar. Daux: a density-based approach for uncertainty explanations. *arXiv preprint arXiv:2207.05161*, 2022.

[73] William R Clements, Bastien Van Delft, Benoît-Marie Robaglia, Reda Bahi Slaoui, and Sébastien Toth. Estimating risk and uncertainty in deep reinforcement learning. *arXiv preprint arXiv:1905.09638*, 2019.

[74] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. *Advances in neural information processing systems*, 29, 2016.

[75] Rui Yang, Yiming Lu, Wenzhe Li, Hao Sun, Meng Fang, Yali Du, Xiu Li, Lei Han, and Chongjie Zhang. Rethinking goal-conditioned supervised learning and its connection to offline rl. *arXiv preprint arXiv:2202.04478*, 2022.

[76] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

## Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We discussed in the Conclusion that this work is mainly limited in empirical studies.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] Our research studies the general problem of reward shifting in reinforcement learning, aiming at improve learning efficiency.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] Please refer to Appendix B
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Implementation details can be found at Appendix D. Code is open-sourced at https://github.com/holarissun/RewardShifting
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Implementation details can be found at Appendix D.1, D.2, and D.3
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] In our experiments, the results are averaged over 10 runs, and shaded areas indicate the $25\%$ - $75\%$ quantile values.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Please refer to Appendix D.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] [23, 24]
   (b) Did you mention the license of the assets? [Yes] MIT license.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] We follow BCQ and CQL to use the dataset they used in the papers.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] Not applicable.

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] Not applicable.

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] Not applicable.

# A Extended Related Work

We extend our related work section on the following related topics as suggested by reviewers:

### A.1 Discussion on Ensembles and Distributional RL

In Distributional RL literature [60–64], the distribution of the $Q$-value, rather than the mean scaler, is estimated. Distributional-RL focuses on stochastic reward mechanisms and smooth the temporal difference learning in a distributional level, and can be applied to risk-sensitive scenarios [65] where the worst-case performance can be controlled [66]. In those scenarios, the systematic uncertainty is the crucial issue to address, whereas in our work, we focus on deterministic transition dynamics and use OFU to tackle the epistemic uncertainty in the section of RRS.

Several previous works discussed ensemble methods for exploration, both for discrete control [67, 68] and for continuous control [69, 70]. However, in our work we show that more exploratory behavior can emerge with the help of reward shifting under only single $Q$-value network — as a proof of concept that negative reward shift is equivalent to optimistic initialization.

### A.2 Model-Based Exploration and Uncertainty Estimation

Besides the model-free value-based RL methods we focused in our paper, there exist literature like R-Max [40] that work with model-based methods for better exploration. Moreover, combining multiple neural networks for uncertainty estimation is well-established in supervised learning [71], and similari idea has been explored in the context of RL for discrete control [45, 72]. In this work, we showcase that a diversified set of reward shifting constants can work as priors for such an ensemble.

Given the ensemble networks without ground-truth, it is in general hard to disentangle the aleatoric uncertainty from epistemic uncertainty. Our method works on deterministic reward and transition dynamics to circumvent the discussion of uncertainty stratification. In the deterministic settings, the source of uncertainty can be solely attributed to the epistemic uncertainty and hence help informative exploration. When it comes to the stochastic environments, the entanglement of aleatoric uncertainty and epistemic uncertainty will make the problem much more difficult [73] as intrinsic motivation methods may get trapped by pursuing the actions that result in high aleatoric uncertainty in certain circumstances (e.g., the Noisy-TV) [34].

# B Proof of Proposition 4.1

*Proof.* the estimated $Q$-value $\hat{Q}(s,a)$ is composed by the two estimators with function approximation error, defined as $\epsilon_{b^+}(s,a) = \tilde{Q}_{b^+,t}(s,a) - \frac{b^+}{1-\gamma} - Q^*(s,a)$, and $\epsilon_{b^-}(s,a) = \tilde{Q}_{b^-,t}(s,a) - \frac{b^-}{1-\gamma} - Q^*(s,a)$.

$$
\begin{aligned}
&(1-\beta)\epsilon_{A,t} + \beta\epsilon_{B,t} \\
&= 2\eta(1-\beta)\hat{Q}^* + (1-\beta)(1-2\eta)\tilde{Q}_{A,t} + 2\eta\beta\hat{Q}^* + \beta(1-2\eta)\tilde{Q}_{B,t} \\
&= 2\eta\hat{Q}^* + (1-2\eta)[(1-\beta)\tilde{Q}_{A,t} + \beta\tilde{Q}_{B,t}] \\
&= 2\eta\hat{Q}^* + (1-2\eta)[(1-\beta)(1-2\eta)^t\tilde{Q}_{A,0} + \frac{4(1-\beta)\eta^2}{1-(1-2\eta)^t}\hat{Q}^* + \beta(1-2\eta)^t\tilde{Q}_{B,0} + \frac{4\beta\eta^2}{1-(1-2\eta)^t}\hat{Q}^*] \\
&= 2\eta\hat{Q}^* + (1-2\eta)[(1-2\eta)^t((1-\beta)\tilde{Q}_{A,0} + \beta\tilde{Q}_{B,0}) + \frac{4\eta^2}{1-(1-2\eta)^t}\hat{Q}^*] \\
&= \epsilon_{C,t}
\end{aligned}
\tag{8}
$$

where $C = (1-\beta)A + \beta B$ and the last line requires $\tilde{Q}_{A,0} = \tilde{Q}_{B,0} = \tilde{Q}_{C,0}$ are identical initialization.

With this notion, Equation (7) can be re-written as

$$\begin{aligned}\hat{Q}(s,a) &= Q^*(s,a) + (1-\beta)\epsilon_{b^+}(s,a) + \beta\epsilon_{b^-}(s,a) \\ &= Q^*(s,a) + \epsilon_{(1-\beta)b^+ + \beta b^-}(s,a) \\ &= Q^*(s,a) + \epsilon_{c_r}(s,a)\end{aligned} \tag{9}$$

where the second line relies on the linear assumption of the approximation error $(1-\beta)\epsilon_{b^+}(s,a) + \beta\epsilon_{b^-}(s,a)$. We further have $(1-\beta)\tilde{Q}_{b^+} + \beta\tilde{Q}_{b^-} = \tilde{Q}_{(1-\beta)b^+ + \beta b^-}$ and $\hat{Q}(s,a) = \tilde{Q}_{c_r}(s,a)$, telling us that trading-off between the constant $b^-$ used for exploration and the constant $b^+$ used for exploitation with the coefficient $\beta$ is equivalent to use another constant with value of $c_r = (1-\beta)b^+ + \beta b^-$.

□

# C Implications of Assumption in Section 4.3.1

In our main text, the estimated values for extremely o.o.d. samples are assumed to be near zeros. We provide detailed implications and explanations in this section.

On the one hand, it's clear that such an assumption holds for the tabular settings, that un-visited state-action pairs have the value in tabular initialization.

On the other hand, we acknowledge it as a mild assumption that there always exists o.o.d. samples that have the $Q$-values near zero for function approximation settings. Interpolation between those o.o.d. samples and other state-action pairs will clearly lead to an "in-between" value estimation, which in practice can be achieved with properly regularized neural networks.

The key insight we want to emphasize in Section 4.3.1 is that for frequently visited state-action pairs, the value discrepancy with different initialization are small, while for seldomly-visited state-action pairs, the discrepancy are relatively large, enabling the usage of such discrepancy as exploration bonus.

# D Implementation Details and Ablation Studies

**Hardware and Training Time** We experiment on a server with 8 TITAN X GPUs and 32 Intel(R) E5-2640 CPUs. In general, shifting the reward does not introduce further computation burden except in the continuous control tasks, our method of Random Reward Shift (RRS) requires two additional $Q$-value networks. In our PyTorch-based implementation, those additional networks can be easily implemented and optimized in a parallel manner, and the extra computational burden is equivalent to using a $\sqrt{3}$ times wider neural network during optimization. It is worth noting that RRS is computationally much cheaper than the Bootstrapped TD3, where additional policy networks are also needed.

**Network Structure** Our implementation of TD3, BCQ and CQL are based on code released by the authors, without changing hyper-parameters. We implement DQN based on a 3-layer fully connected neural network with $64$ hidden units for the $Q$-value function, using ReLU and linear activation respectively. We use the Adam optimizer with learning rate of $0.001$, and use an epsilon-greedy approach as naive exploration strategy. In our RND, we use two 4-layer fully connected neural networks with $512$ units and ReLU activation in each hidden layer, and a softmax activation for the output layer. Adam optimizer is used for the optimization of the RND networks with learning rate $0.0001$.

Our code is provided in the supplementary materials, and will be made public available.

## D.1 Offline RL

In our experiments, we use a fixed dataset with 10k offline trainsition tuples for offline RL learning. Our implementation of BCQ and CQL are both based on the code provided by the authors. The only change we made to verify our insight is to shift the reward by a constant. In most environments, we

**Algorithm 1** Sample-Efficient Continuous Control with Random Reward Shift

**Require**
Size of mini-batch $N$, smoothing factor $\tau > 0$, $K$ reward shift values $r'_k = r + b_k, k = 1, \ldots, K$.
Random initialized policy network $\pi_\theta$, target policy network $\pi_{\theta'}$, $\theta' \leftarrow \theta$.
$K$ random initialized $Q$ networks, and corresponding target networks, parameterized by $w_k, w'_k$, $w'_k \leftarrow w_k$ for $k = 1, \ldots, K$. (e.g., a ModuleList in PyTorch).
**for** iteration $= 1, 2, \ldots$ **do**
  Uniformly sample one of the $K$ $Q$-functions, $Q_{w_k}$, for policy update
  **for** t $= 1, 2, \ldots$ **do**
    # Interaction
    Run policy $\pi_\theta$, and collect transition tuples $(s_t, a_t, s'_t, r_t)$.
    Sample a mini-batch of transition tuples $\{(s, a, s', r)_i\}_{i=1}^N$.
    # Update $Q_w$ (in parallel)
    Calculate the $k$-th target $Q$ value $y_{k,i} = r_i + b_k + Q_{w'_k}(s'_i, \pi_{\theta'}(s'_i))$
    Update $w_k$ with loss $\sum_{i=1}^N (y_{k,i} - Q_{w_k}(s_i, a_i))^2$.
    # Update $\pi_\theta$
    Update policy $\pi_\theta$ with $Q_{w_k}$
  **end for**
  # Update target networks
  $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$.
  $w'_k \leftarrow \tau w_k + (1-\tau) w'_k, k = 1, \ldots, K$.
**end for**

find $r' = r + 8$ provides good enough performance. While in Hopper Medium CQL we find using a smaller positive reward shift $r' = r + 1$ works better than $r' = r + 8$, and for Walker Medium CQL, using a larger reward shift of $r' = r + 50$ further improves the result with $r' = r + 8$.

Figure 7 shows different performance under different choices of the reward shift constant. We denote a positive reward shift $r' = r + 8$ as **Pos.1**, denote $r' = r + 20$ as **Pos.2** and denote $r' = r + 50$ as **Pos.3** for all experiments excetp in the Hopper Medium CQL we use **Pos.1** to denote $r' = r + 1$.

In the experiments based on BCQ (first two figures). We can observe a uniformly performance improvement with all choices of reward shift constants. As the algorithm of CQL has already taken the conservative value estimation into consideration, in the experiments based on CQL, the performance is more closely related to the constant we use. Specifically, in Hopper Expert, while using any of the positive reward shift constants improve the learning stability, $r' = r + 8$ performs better on preserving the learning efficiency during early learning stage. For Hopper Medium, we find using larger positive constants hinder the performance. For Walker Medium, using a larger constant in reward shift performs much better than using a smaller one.

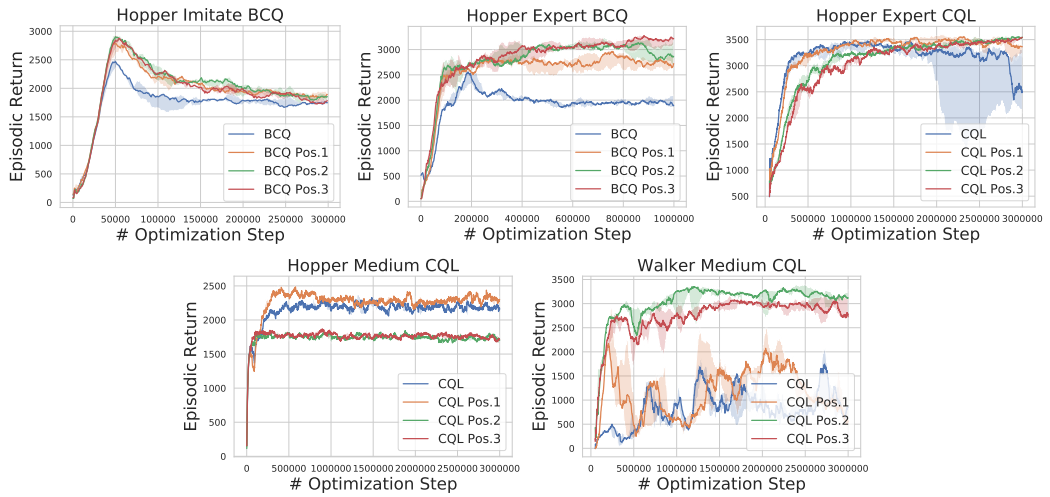


Figure 7: Performance with different reward shift constants.

### D.2 Continuous Control

**Details of RRS** Although we find in the motivating example that a $-5$ reward shift is able to remarkably improve the asymptotic performance of TD3, in this work we aim at proposing an uniformly suitable method based on the insight behind the motivating example. Therefore we propose to use $\pm 0.5, 0$ as the reward shifting constants. We find in experiment that the sampling frequency does not affect the performance. And in the experiments we follow BDQN [45] to use a fixed value network throughout a whole trajectory. i.e., one of the $K$ $Q$-networks is sampled uniformly after each episode with length of 1000 timesteps. Intuitively, searching for more suitable reward randomization designs may further improve the performance, yet that is beyond the coverage of this work.

**Ablation Studies** We experiment with different number of $Q$-value networks as well as different choices of the random reward shifting ranges. Results are presented in Figure 8. We denote RRS with 7 reward shifting constants ( and therefore also 7 $Q$-networks) as **RRS-7**, and denote RRS with 3 reward shifting constants ( and therefore also 3 $Q$-networks) as **RRS-3**. The constants following **RRS-3/RRS-7** are the ranges of those random constants. Specifically, we use $[-0.5, 0, 0.5]$ for the **RRS-3 0.5** settings, $[-1.0, 0, 1.0]$ for the **RRS-3 1.0** settings, $[-0.5, -0.33, -0.17, 0, 0.17, 0.33, 0.5]$ for the **RRS-7 0.5** settings and $[-1.0, -0.67, -0.33, 0, 0.33, 0.67, 1.0]$ for the **RRS-7 1.0** settings. According to the experimental results, RRS is not sensitive to hyper-parameters, showing the robustness of the proposed method. We believe further search for those hyper-parameters can further improve the learning efficiency, yet this is off the main scope of this work and therefore left for the future research.

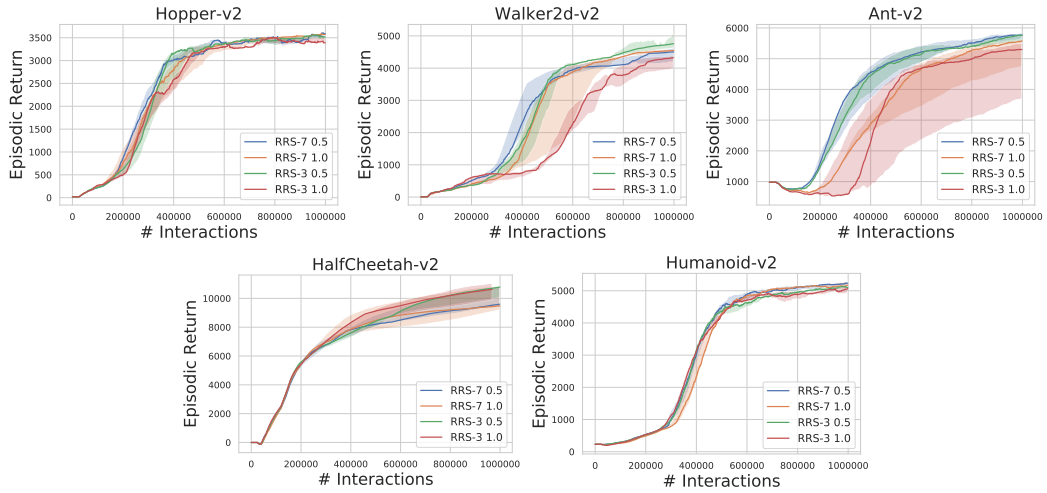


Figure 8: Performance with different reward shift constants and different number of $Q$-networks.

### D.3 Random Network Distillation

**Environments** In this work, we experiment with five discrete (sparse reward) exploration tasks , namely the MountainCar-v0, and four navigation tasks of MiniGrid suite [58], namely the task of Empty-Random, MultiRoom, and FourRooms, to verify our insight on improving RND for value-based curiosity-driven exploration. Figure 9 shows example of different tasks.

**Hyper-Parameter Settings** We use a 2-layer NN with 64 hidden units for $Q$-networks in DQN and set RND networks to be 3-layer NN with 512 hidden units. $\epsilon$-greedy exploration is applied to DQN with $\epsilon$ decays from 0.9 to 0.05 in the first 1/5 episodes. Size of replaybuffer is set to be 100000.

**Ablation Studies** We experiment with different reward shifting constants in the discrete control settings. We use a relatively large range in choosing constants, i.e., $\{-0.05, -0.15, -1.0, -1.5, -2.0, -2.5, -5.0, -10.0\}$. Results are presented in Figure 10. In all experiments, using a moderate reward shifting constant like $\{-1.0, -1.5, -2.0, -2.5\}$ remarkably improves the learning efficiency. On the other hand, a too aggressive reward shifting will lead to too much curiosity exploration and hinder the learning efficiency in the limited number of interactions.

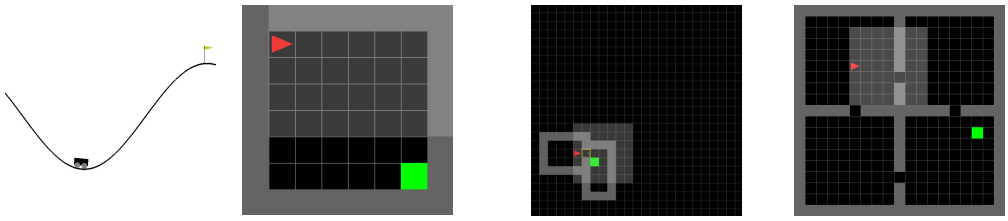

Figure 9: Examples of environments used in Section 5.3. The first figure shows the MountainCar-v0 environment where a car needs to accumulate potential energy to reach the flag, to receive a positive reward. The second figure shows the maze of the Empty-Random task with size of 6, the third one shows the MultiRoom of level S2-N4, where there are 2 rooms with size 4, the last figure shows example of FourRoom task with size 17. In our experiments, as we use the vanilla DQN as the baseline, which is not suitable for partial observable tasks, we use a smaller maze of size 7 and 9 to avoid further dependency on memories. In all tasks of the MiniGrid domain, the triangular red agent need to navigate to the green goal square, and the observable region is only a 7x7 square the agent is facing to (i.e., the regions with shallower color in the last three figures).

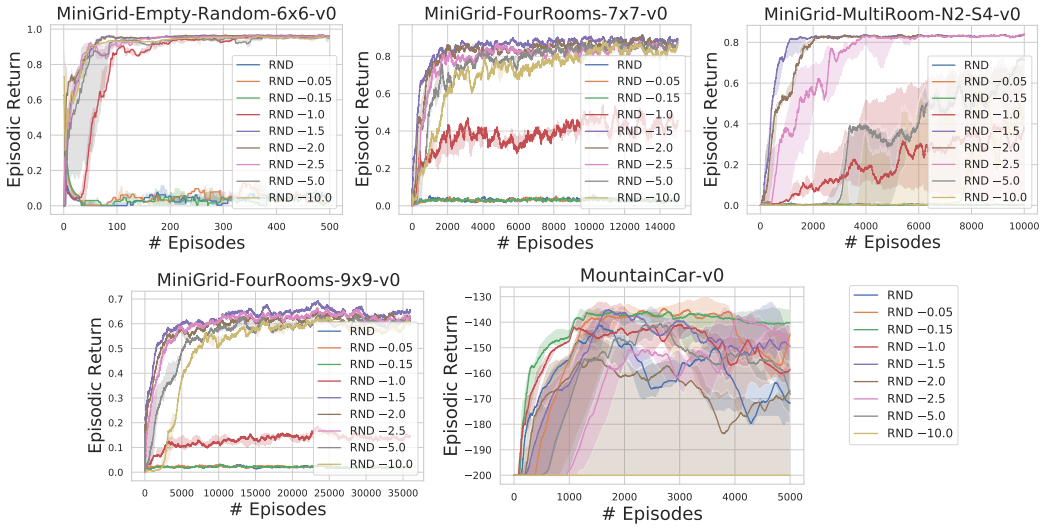


Figure 10: Performance with different reward shift constants in RND.

# E Additional Experiments

## E.1 Necessity of Explorative Behaviors in Maze Tasks

We demonstrate the benefits of reward shifting for deep-exploration tasks and the on-par performance of reward shifting for easy tasks that does not require deep-exploration. We experiment on the maze environment and change the size of maze to vary from 2 to 20, denoting as S2, S5, S10, S15, S20, separately. Results averaged over 8 runs are reported in Figure 11. We find in easy tasks (S2, S5, S10), both count-based exploration and reward shifting perform similarly to the $\epsilon$-greedy exploration, while on challenging tasks (S15, S20), more explorative behavior encouraged by reward shifting and count-based exploration are important for efficient learning.

## E.2 Performance in Challenging Continuous Control Exploration Tasks

Figure 12 show experiments on (1) the HalfCheetah-SparseReward, where a reward of +1 is provided only when the forward movement of the halfcheetah is larger than 5 unit with regard to the timeframe; We note that this environment is different from the SparseHalfCheetah environment that first introduced in VIME [74], as we use different time frames. (We acknowledge and thank the anonymous reviewer FqMf for pointing out this difference.) and (2) the Humanoid, which is a high dimensional continuous control task in the MuJoCo locomotion suite; to verify the effectiveness of reward shifting in exploration. In the HalfCheetah-SparseReward environment, the maximum score is 1000 for each

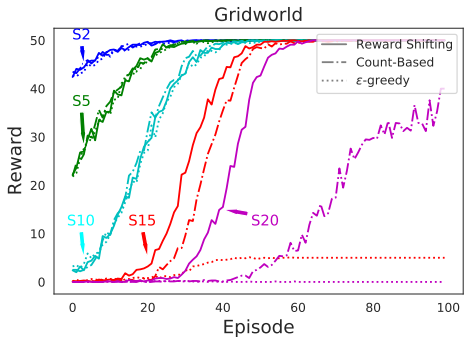


Figure 11: In deep-exploration tasks, reward shifting benefits exploration by optimistic initialization, while in easier tasks, reward shifting does not hinder exploitation, convergence efficiency, and the asymptotic performance.

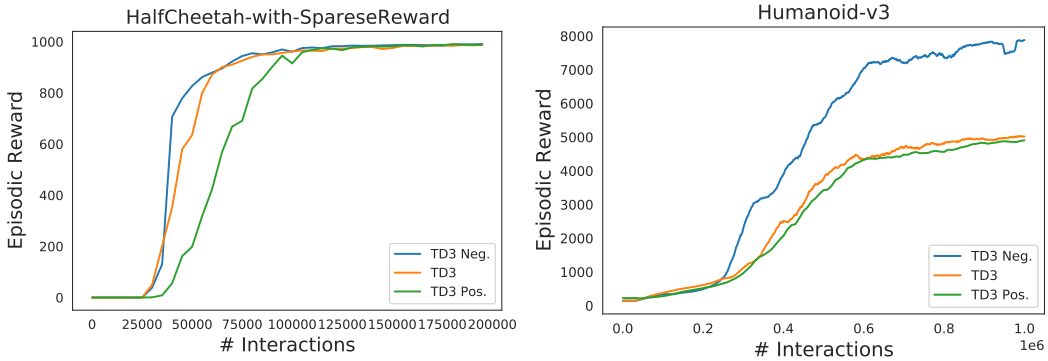


Figure 12: Experiments on two challenging continuous control tasks. Experiments are repeated with 8 runs.

episode. We use $-0.5$ as the negative reward shift for exploration and use $0.5$ for comparison. In Humanoid, the per-step reward is approximately 5 in previous well-performing agents [14, 20], and we hereby use negative shift $-5$ and use positive shift 5, for comparison.

In HalfCheetah-SparseReward, we find a negative reward shift lead to more explorative behavior and improves the learning efficiency while a positive reward shift hinders the learning efficiency. In Humanoid, we find using a reward shift can drastically improve the asymptotic performance by $+60\%$, while learning with a positive reward shift retard the learning and converge to a lower performance.

### E.3 Performance in Goal-Conditioned Continuous Control (Robotics) Suite

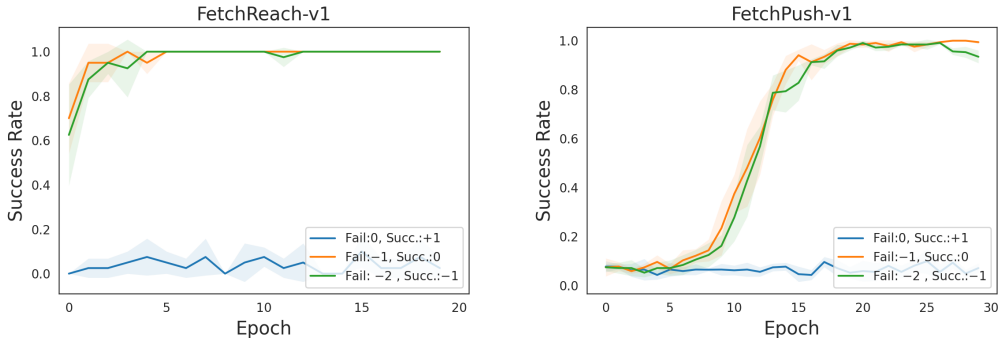


Figure 13: Experiments on two GCRL robotics tasks. Experiments are repeated with 4 runs.

To further address the reviewer's concern on the applicability of reward shifting on challenging continuous control exploration tasks, we benchmark reward shifting on the FetchRobotics suite [8] that is usurally considered to be challenging exploration task in GCRL literature [75, 76]. Figure 13 shows the results we get on FetchReach-v1 and FetchPush-v1 environment. We use HER [76] as the backbone algorithms and vary the reward for failure and success in achieving the goals. In their default setting, reaching the goal will receive a reward of $0$, otherwise, the agent will receive $-1$ as punishment. In experiments, we find using a positive reward $+1$ in reaching the goals while using a trivial $0$ reward otherwise will drastically hinder the learning efficiency of HER. Similar empirical discovery has been reported in Sun et al. [37] in the PPO-based learners. This set of experiments verifies our key insight one more time that explorative behaviors emerge with a negatively shifted reward function, and a positive reward shift leads to conservative behavior.

To sum up, our key insight reveals the mechanism of how such an empirically verified heuristic design in GCRL works: a negative reward $-1$ (also interpreted as cost) works in the same way as reward shifting to improve exploration.