β-DQN: Improving Deep Q-Learning By Evolving the Behavior

Hongming Zhang
University of Alberta and Amii
Edmonton, Canada
hongmin2@ualberta.ca

Fengshuo Bai
Shanghai Jiao Tong University
Shanghai, China
fengshuobai@sjtu.edu.cn

Chenjun Xiao
CUHK-Shenzhen
Shenzhen, China
chenjunx@cuhk.edu.cn

Chao Gao
Edmonton Research Center, Huawei
Edmonton, Canada
chao.gao4@huawei.com

Bo Xu
CASIA
Beijing, China
boxu@ia.ac.cn

Martin Müller
University of Alberta and Amii
Edmonton, Canada
mmueller@ualberta.ca

ABSTRACT

While many sophisticated exploration methods have been proposed, their lack of generality and high computational cost often lead researchers to favor simpler methods like ϵ-greedy. Motivated by this, we introduce β-DQN, a simple and efficient exploration method that augments the standard DQN with a behavior function β. This function estimates the probability that each action has been taken at each state. By leveraging β, we generate a population of diverse policies that balance exploration between state-action coverage and overestimation bias correction. An adaptive meta-controller is designed to select an effective policy for each episode, enabling flexible and explainable exploration. β-DQN is straightforward to implement and adds minimal computational overhead to the standard DQN. Experiments on both simple and challenging exploration domains show that β-DQN outperforms existing baseline methods across a wide range of tasks, providing an effective solution for improving exploration in deep reinforcement learning.

KEYWORDS

Deep Reinforcement Learning; Exploration

ACM Reference Format:
Hongming Zhang, Fengshuo Bai, Chenjun Xiao, Chao Gao, Bo Xu, and Martin Müller. 2025. β-DQN: Improving Deep Q-Learning By Evolving the Behavior. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 22 pages.

1 INTRODUCTION

Exploration is considered as a major challenge in deep reinforcement learning (DRL) [53, 61]. The agent needs to trade off between exploiting current knowledge for known rewards and exploring the environment for potential better rewards. While many complex methods have been proposed for efficient exploration, the most commonly used ones are still simple methods such as ϵ-greedy and entropy regularization [5, 40, 47, 66]. We identify the possible reasons. First, more advanced methods require meticulous hyper-parameter tuning and much computational cost [2, 3, 20]. Second, these methods adopt specialized inductive biases, which may achieve high performance in specific hard exploration environments but tend to underperform compared to simpler methods across a broad range of domains, highlighting their lack of generality [4, 11, 55].

We improve exploration while considering the following aspects: (1) **Simplicity**. We aim to achieve clear improvement while keeping the method simple. This ensure the method is straightforward to implement and minimizes the burden of hyper-parameters tuning. (2) **Mild Increase in Computational Cost**. While prioritizing sample efficiency in RL tasks, we aim to strike a balance that avoids substantial increase in training time. Our goal is to develop a method that is both effective and efficient. (3) **Generality Across Tasks**. The method should maintain generality, rather than being tailored to specific hard exploration environments.

Motivated by these considerations, we propose β-DQN, a simple and efficient exploration method that augments the standard DQN with a behavior function β. The function β represents the behavior policy that collects data in the replay memory, estimating the probability that each action has been taken at each state. Combined with the Q function in DQN, we use β for three purposes: (1) **Exploration for state-action coverage**. Taking actions with low probabilities based on β encourages the agent to explore the state-action space for better coverage; (2) **Exploration for overestimation bias correction**. Exploring overestimated actions to get feedback and correct the overestimation bias in the Q function; (3) **Pure exploitation**. Using β to mask the Q function at unseen actions derives a greedy policy that represents pure exploitation. Interpolating among them allows us to construct a population of temporally-extended policies that interleave exploration and exploitation at intra-episodic level with clear purposes [44]. We then design a meta-controller to adaptively select an effective policy for each episode, providing flexibility without an accompanying hyperparameter-tuning burden.

Our method have the following advantages: (1) We only additionally learn a behavior function, which is straightforward to implement and computationally efficient compared to previous methods [2, 33]. (2) When constructing diverse polices, we do not inject inductive biases specialized for one specific task, making the method general and applicable across a wide range of domains. (3) Our method interleaves exploitation and exploration at the intra-episodic level, carries out temporal-extended exploration, and is state-dependent, which is considered the most effective approach [15, 41, 44]. We report promising results on dense-reward

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

MinAtar [62] and sparse-reward MiniGrid [13], demonstrating that our method significantly enhances performance and exhibits broad applicability in both easy and hard exploration domains.

2 RELATED WORK

Reinforcement learning (RL) is known for learning through trial and error [17, 53]. If a state-action pair has not been encountered, it cannot be learned [44], making exploration a central challenge in RL. The most commonly used exploration strategies are simple dithering methods like ϵ-greedy and entropy regularization [27, 40, 47, 63, 67]. While these methods are general, they are often inefficient because they are state-independent and lack temporal persistence [44, 61]. Therefore, inducing a consistent, state-dependent exploration policy over multiple time steps has been a key pursuit in the field [15, 18, 31, 41, 44, 48, 49].

Temporally-Extended Exploration. Bootstrapped DQN [41] induces temporally-extended exploration by building K bootstrapped estimates of the Q-value function in parallel and sampling a single Q function for the duration of one episode. The computational cost increases linearly with the number of heads K. Temporally-extended ϵ-Greedy (ϵz-greedy) [15] simply repeats the sampled random action for a random duration, but its exploration remains state-independent. Our method, based on the behavior function β, induces temporally-extended exploration by selecting actions according to state-dependent probabilities.

Besides adding temporal persistence to the exploration policy explicitly, another line of work involves adding exploration bonuses to the environment reward [7, 26, 30, 37, 38, 42, 51, 56, 68]. These bonuses are designed to encourage the agent to visit states that are novel. The count-based exploration bonus encourages the agent to visit states with low visit counts [7, 42, 56]. Prediction error-based methods, such as the Intrinsic Curiosity Module (ICM) [43] and Random Network Distillation (RND) [3, 11], operate on the intuition that the prediction error will be low on states that are previously visited and high on newly visited states. These methods are designed to tackle difficult exploration problems and usually perform well in hard exploration environments. However, they often underperform compared to simple methods like ϵ-greedy in easy exploration environments [55], highlighting their lack of generality. In contrast, our method is designed to be general and applicable across a wide range of tasks.

Population-based Exploration. Recent promising works tried to handle the exploration problem using population-based methods, which collect samples with diverse behaviors derived from a population of different exploratory policies [1, 2, 19, 20, 32, 33, 52]. These methods have demonstrated powerful performance, outperforming the standard human benchmark on all 57 Atari games [8]. They maintain a group of diverse actors with independent parameters, build distributed systems, and interact with the environment over billions of frames. While these methods achieve significant performance gains, their computational cost is substantial and often unaffordable in practice. This widens the gap between researchers with ample access to computational resources and those without [12]. β-DQN introduces a population of diverse policies with minimal computational overhead, making it much more accessible. Our goal is to absorb the strengths of existing population-based methods and design an effective approach with a low computational cost.

3 BACKGROUND

Markov Decision Process (MDP). Reinforcement learning (RL) [53] is a paradigm of agent learning via interaction. It can be modeled as a Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, R, P, \rho_0, \gamma)$. $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the environment transition dynamics, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, $\rho_0 : \mathcal{S} \to \mathbb{R}$ is the initial state distribution and $\gamma \in (0, 1)$ is the discount factor. The goal of the agent is to learn an optimal policy that maximizes the expected discounted cumulative rewards $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$.

Deep Q-Network (DQN). Q-learning is a classic algorithm to learn the optimal policy. It learns the Q function with Bellman optimality equation [9], $Q^*(s, a) = \mathbb{E}[r + \gamma \max_{a'} Q^*(s', a')]$. An optimal policy is then derived by taking an action with maximum Q value at each state. DQN [40] scales up Q-learning by using deep neural networks and experience replay [36]. It stores transitions in a replay memory and samples batches of data uniformly to estimate an action-value function Q_θ with temporal-difference (TD) learning:

$$Q_\theta(s, a) \leftarrow r(s, a) + \gamma \max_{a'} Q_\theta(s', a'). \tag{1}$$

A target network with parameters θ^- copies the parameters from θ only every C steps to stabilize the computation of learning target $y = r(s, a) + \gamma \max_{a'} Q_{\theta^-}(s', a')$.

4 METHOD

Drawing from insights introduced in Section 2, promising exploration strategies should be state-dependent, temporally-extended, and consist of a set of diverse policies. Keeping simplicity and generality in mind, we design an exploration method for DQN that performs well across a wide range of domains and is computationally affordable for the research community. We additionally learn a behavior function β and construct a set of policies that balance exploration between state coverage and bias correction. A meta-controller is then designed to adaptively select a policy for each episode. In Section 4.1, we introduce how to learn the behavior function β and augment it with the Q function for three purposes: exploration for state-action coverage, exploration for overestimation bias correction, and pure exploitation. In Section 4.2, we derive a set of policies by interpolating exploration and exploitation at the intra-episodic level. Based on this policy set, we design an adaptive meta-controller in Section 4.3 to choose an effective policy for interacting with the environment in each episode. Figure 1 provides an overview of our method.

4.1 Behavior Function β

Learning the behavior function β is straightforward. We sample a batch of data B from the replay memory and train a network using supervised learning with cross entropy loss:

$$\mathcal{L}_\beta = -\frac{1}{|B|} \sum_{(s,a)\in B} \log \beta(s, a). \tag{2}$$

β represents an average of the policies that collect data in the replay memory. It estimates the probability of each action that has been

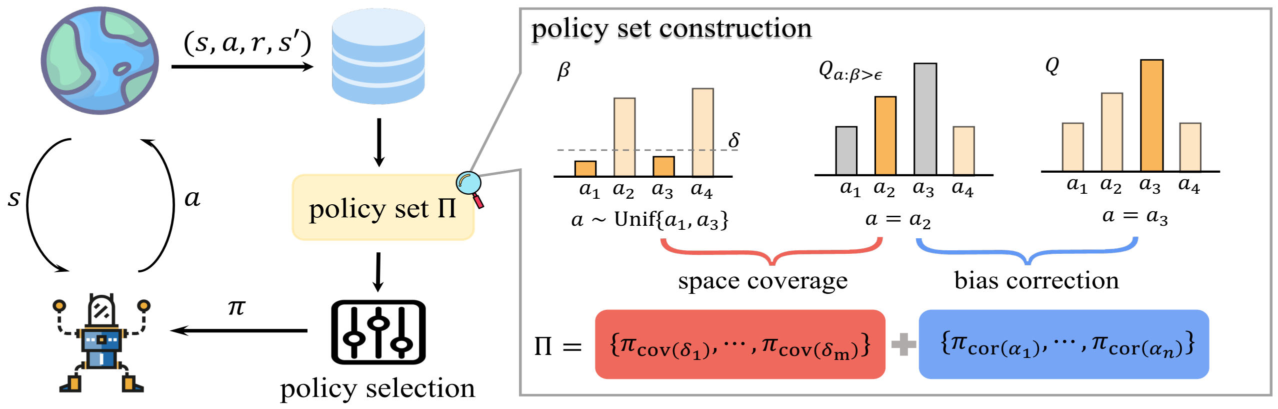


Figure 1: Method overview. We learn a behavior function β from the replay memory and augment it with the Q function for three purposes: state-action coverage (left), pure exploitation (middle), and overestimation correction (right). By interpolating between these strategies, we construct a policy set that balances exploration and exploitation at the intra-episodic level. A meta-controller adaptively selects a policy for each episode.

taken at each state. We use the same data batch to learn β and Q, thus incurring no additional computational cost for sampling.

Exploration for State-Action Coverage. β differentiates between actions that are frequently taken and those that are rarely taken. We sample actions with probabilities lower than a threshold δ to explore the state-action space for better coverage:

$$a \sim \mathrm{Unif}\{a : \beta(a|s) \le \delta\} \tag{3}$$

Here, $\mathrm{Unif}(\cdot)$ denotes selecting an action randomly from a given set, and $\delta \in [0, 1]$ is a parameter. This policy is purely exploratory, focusing on better state-action coverage without considering the rewards obtained.

Exploration for Overestimation Bias Correction. Valued-based methods such as DQN estimate an action-value function Q with temporal-difference (TD) learning with Equation (1). The maximum operator in the Bellman update may lead to overestimation of action values [22, 57]. The greedy policy based on Q,

$$a = \arg\max_a Q(s, a), \tag{4}$$

is an optimistic policy that may take erroneously overestimated actions. This can be one kind of exploration that induces corrective feedback to mitigate biased estimation in Q function [34, 46].

An alternative way to benefit from β is constraining the learning to in-sample state-action pairs [6, 60, 64, 65]:

$$Q(s, a) \leftarrow r + \gamma \max_{a':\beta(a'|s')>\epsilon} Q(s', a'). \tag{5}$$

The max operator only bootstraps from actions well-supported in the replay memory, determined by $\beta(s, a) > \epsilon$, where ϵ is a small number. Because the data coverage in the replay memory is limited to a tiny subset of the entire environment space, when combing with deep neural networks, Equation (5) learns an in-sample estimation at existing state-action pairs and generalizes to missing data. While it may still overestimate out-of-distribution state-action pairs, leading to unrealistic values and inducing exploration to correct the bias, an additional benefit is that the learning process becomes more stable and exhibits better convergence properties.

Proposition 4.1. *In the tabular case with finite state action space $\mathcal{S} \times \mathcal{A}$, the temporal difference learning masked by β given in Equation* (5) *uniquely converges to the optimal in-sample value $\widehat{Q}^*$ on explored state-action pairs. When $\beta(a|s) > \epsilon$ for all $a \in \mathcal{A}$, $\widehat{Q}^*$ equals to Q^*, which recovers the original temporal difference learning without action mask in Equation* (1).

We prove the convergence of Equation (5) by showing that the update rule is a γ-contraction mapping. The contraction property ensures that the update rule converges to a unique fixed point. The proof is provided in Appendix A.

This indicates that if the state-action space is fully covered, the two update rules are equivalent. When there are missing transitions, Equation (5) converges on the explored state-action pairs. In contrast, Equation (1) does not guarantee convergence even for the explored state-action pairs, as shown in previous work [23].

Pure Exploitation. Although the Q function may overestimate, the behavior function β can differentiate between frequently and rarely taken actions. By combining Q with β, we can mask actions with low probabilities in β before taking the greedy action of Q:

$$a = \arg\max_{a:\beta(a|s)>\epsilon} Q(s, a), \tag{6}$$

where ϵ is a small number. This policy is purely exploitative, aiming to maximize rewards based on the current experiences stored in the replay memory.

4.2 Constructing Policy Set

Previous work [44] has shown that intra-episodic exploration, i.e., changing the mode of exploitation and exploration in one episode, is the most promising diagram. With the three policies in Equations (3), (4) and (6) with clear purposes, we interpolate exploration and exploitation at the intra-episodic level to construct a set of diverse policies that balance exploration between state coverage and overestimation bias correction.

Polices for State-Action Coverage. One leaving question in Equation (3) is which action to take when all actions have probabilities higher than δ. We address this by interpolating the pure exploitation policy from Equation (6) into the exploration policy in Equation (3) to create a policy that enhances state-action coverage:

$$\pi_{\mathrm{cov}(\delta)} := \begin{cases} \arg\max_{a:\beta(a|s)>\epsilon} Q(s,a), & \text{if } \beta(s,a) > \delta \ \forall\ a \in \mathcal{A} \\ \mathrm{Unif}\{a : \beta(a|s) \le \delta\}, & \text{otherwise} \end{cases} \tag{7}$$

The intuition behind $\pi_{\mathrm{cov}(\delta)}$ is straightforward: if all actions at a state have been tried several times, we follow the pure exploitation mode to choose actions and reach the boundary of the explored area. Otherwise, we sample an action uniformly from the rarely taken actions, as determined by δ.

Proposition 4.2. *In the tabular case with finite state action space $\mathcal{S} \times \mathcal{A}$ and finite horizon H, taking actions following policy $\pi_{cov(\delta)}$ guarantees infinite state-action visitations for all state action pairs. However, the expected cumulative regret is linear.*

We prove this proposition by first demonstrating its validity on bandit problems and then extending the proof to MDPs. The detailed proof is provided in Appendix A.

Selecting actions according to $\pi_{\mathrm{cov}(\delta)}$ ensures that all state-action pairs are visited infinitely often in the long run, guaranteeing the convergence of value iteration in the tabular case [50, 58]. By setting different values of δ, we obtain a range of policies with varying degrees of exploration. Specifically, $\pi_{\mathrm{cov}(0)}$ is the pure exploitation policy as defined in Equation (6), while $\pi_{\mathrm{cov}(\delta)}$ becomes a random policy when $\delta \ge 1/|\mathcal{A}|$.

Proposition 4.2 also indicates that the total regret is linear if we only follow $\pi_{\mathrm{cov}(\delta)}$ to choose actions. An excessive amount of steps can be wasted traveling through already-explored but unpromising states, reducing overall efficiency. Since the goal of learning is to obtain an accurate action value function, we design exploration to try overestimated actions and correct the estimation bias.

Polices for Overestimation Bias Correction. Action value-based algorithms are known to overestimate action values [22, 57]. Accurate value estimation is critical for extracting a good policy in DRL. We interpolate between the overestimated policy in Equation (4) and the pure exploitation policy in Equation (6) to create a policy that explores overestimated actions for corrective feedback:

$$\pi_{cor(\alpha)} = \arg\max_a (\alpha Q(s,a) + (1-\alpha)\hat{Q}(s,a)). \tag{8}$$

Here, $\hat{Q}$ is the value function where we suppress the overestimated action values identified by β,

$$\hat{Q}(s,a) = \begin{cases} Q(s,a), & \text{if } \beta(s,a) > \epsilon \\ \min_{a\in\mathcal{A}} Q(s,a), & \text{otherwise} \end{cases} \tag{9}$$

The intuition is to follow the current best actions at some states while exploring overestimated actions at others. The parameter $\alpha \in [0,1]$ allows us to set different values to obtain policies with varying degrees of exploration for bias correction. Specifically, $\pi_{\mathrm{cor}(0)}$ recovers the pure exploitation policy in Equation (6), and $\pi_{\mathrm{cor}(1)}$ recovers the overestimated policy in Equation (4).

Constructing Policy Set. By setting different values for δ and α, we generate a policy set Π that ranges from exploration for better state-action coverage to overestimation bias correction:

$$\Pi = \left\{\pi_{\mathrm{cov}(\delta_1)}, \cdots, \pi_{\mathrm{cov}(\delta_m)}, \pi_{\mathrm{cor}(\alpha_1)}, \cdots, \pi_{\mathrm{cor}(\alpha_n)}\right\}. \tag{10}$$

Algorithm 1 β-DQN

```
1:  Initialize replay memory $\mathcal{D}$ with fixed size
2:  Initialize functions $\beta, Q$ and construct policy set $\Pi$ following Equations (7), (8) and (10)
3:  for episode $k = 0$ to $K$ do
4:      Select a policy $\pi$ according to Equation (12)
5:      Initialize the environment $s_0 \leftarrow Env$
6:      for environments step $t = 0$ to $T$ do
7:          Select an action $a_t \sim \pi(\cdot|s_t)$
8:          Execute $a_t$ in $Env$ and get $r_t, s_{t+1}$
9:          Store transition $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{D}$
10:         Update $\beta$ and $Q$ following Equations (2) and (5)
11:     end for
12: end for
```

This policy set does not inject specialized inductive biases, making it a general method across a wide range of tasks. Additionally, the computational cost does not increase when adding more policies with different δ and α values.

4.3 Meta-Controller for Policy Selection

After constructing a set of policies, we need to select an effective policy to interact with the environment for each episode. Similar to previous work [2, 19, 20, 33], we consider the policy selection problem as a non-stationary multi-armed bandit (MAB) [24, 35], where each policy in the set is an arm. We design a meta-controller to select policies adaptively.

Assume there are N policies in the policy set $\Pi = \{\pi_0, \cdots, \pi_{N-1}\}$. For each episode $k \in \mathbb{N}$, the meta-controller selects a policy $A_k = \pi_i$ and receives an episodic return $R_k(A_k)$. Our objective is to obtain a policy π that maximizes the return within a given interaction budget K.

Due to the non-stationarity of the policies, we consider the recent $L < K$ results. Let $N_k(\pi_i, L)$ be the number of times policy π_i has been selected after k episodes, and $\mu_k(\pi_i, L)$ be the mean return π_i obtained by π_i. We design a bonus b to encourage exploration. An action is considered exploratory if it differs from the pure exploitation action taken by Equation (6). The exploration bonus for policy π_i is computed as:

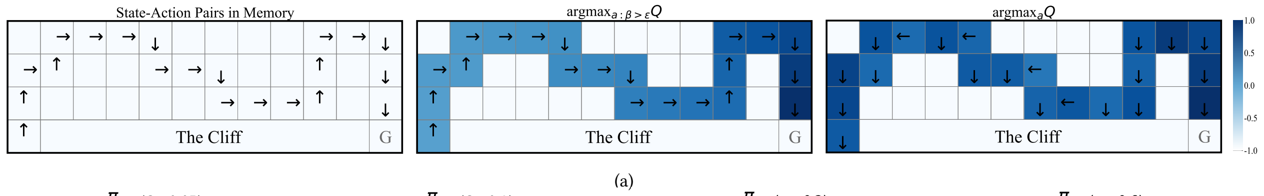
$$b_k(\pi_i, L) = \frac{1}{N_k(\pi_i, L)} \sum_{m=\max(0,k-L)}^{k-1} B_m(\pi_i)\mathbb{I}(A_m = \pi_i), \tag{11}$$

where $B_m(\pi_i)$ computes the ratio of exploration actions taken by policy π_i at episode m and $\mathbb{I}(\cdot)$ is the indicator function.

To select a policy for episode k, we consider the return and exploration bonus of each policy within the sliding window L:

$$A_k = \begin{cases} \pi_i, & \text{if } N_k(\pi_i, L) = 0, \\ \arg\max_{\pi_i} (\mu_k(\pi_i, L) + b_k(\pi_i, L)), & \text{otherwise.} \end{cases} \tag{12}$$

In this formula, if a policy has not been selected in the last L episodes, we will prioritize selecting that policy. Otherwise, the policy that explores more frequently and also gets higher returns is preferred. Algorithm 1 summarizes our method.



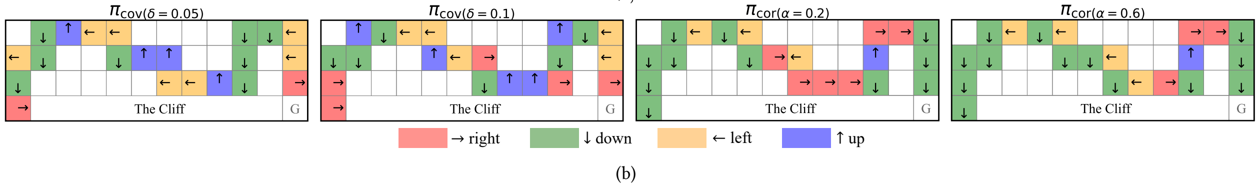

$\pi_{\text{cov}(\delta=0.05)}$
$\pi_{\text{cov}(\delta=0.1)}$
$\pi_{\text{cor}(\alpha=0.2)}$
$\pi_{\text{cor}(\alpha=0.6)}$
The Cliff
G
→ right
↓ down
← left
↑ up
(b)

Figure 2: (a) The first image shows the state-action pairs in memory. It implies that taking actions with low probabilities according to β will try missing actions. The second and third images show the masked/unmasked Q values and the corresponding actions. (b) Policies with different δ and α, demonstrating the effectiveness of our method in constructing a diverse policy set.

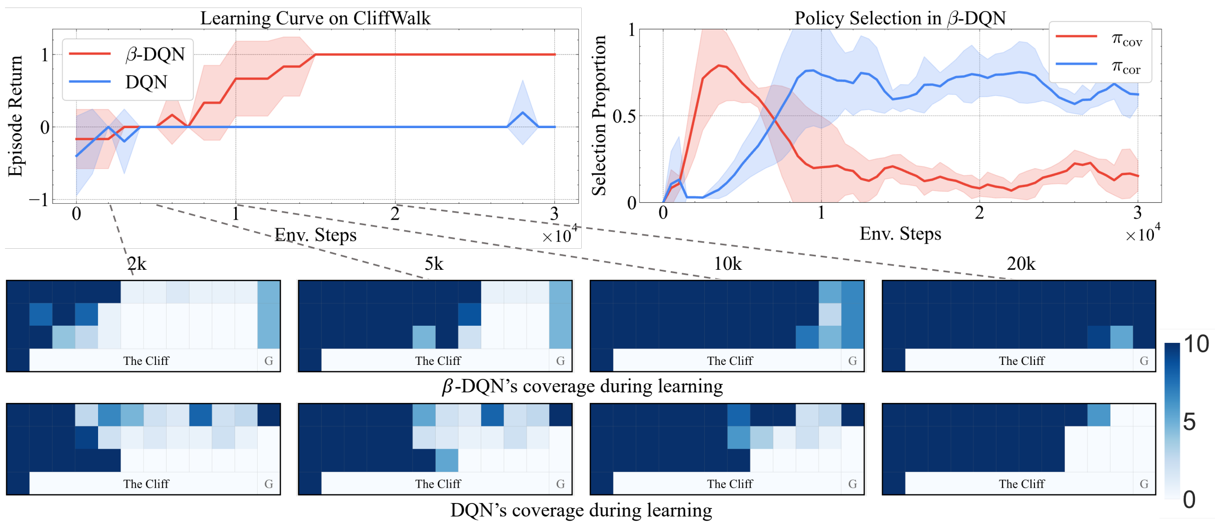


Figure 3: Details during the learning. (1) The top left learning curves show that β-DQN successfully reaches the goal state, while DQN only avoids the cliff. (2) The heatmaps illustrate that β-DQN explores the entire state space efficiently. (3) The top right curves show that β-DQN initially uses π_{cov} for state space exploration, then switches to π_{cor} for correcting biased estimations.

5 EXPERIMENTS

In this section, we aim to answer the following questions:

- Does our method lead to diverse exploration, thereby enhancing the learning process and overall performance?
- Can β-DQN improve performance in both dense and sparse reward environments while maintaining a low computational cost?
- How do the exploration policies π_{cov} and π_{cor} contribute to the learning process in different environments?
- Is there a difference when learning Q with constraints imposed by β compared to without such constraints?

5.1 A Toy Example

In this section, we present a toy example using the CliffWalk environment [53], to illustrate the policy diversity and the learning efficacy of our method. The CliffWalk environment comprises 48 states and 4 actions, as depicted in Figure 2. Starting from the bottom left, the goal is to navigate to state G located at the bottom right. Reaching G yields a reward of 1, while falling into the cliff incurs a penalty of -1; all other moves have a reward of 0.

For a clear illustration of policy diversity, we design a scenario with only one suboptimal trajectory in the replay memory (the left image in Figure 2(a)). The function β, learned from this trajectory, assigns a probability of 1 to the only existing action at each state, while assigning zero probabilities to other actions. The second and third images in Figure 2(a) show the action values masked/unmasked by β, and the corresponding actions taken at each state. This gives us a clear understanding of what actions are taken by different policies introduced in Section 4.1.

In Figure 2(b), by assigning different values to δ and α, we generate a group of diverse policies. Each policy takes different actions, leading to novel states or those with biased estimates. Colors are used to differentiate actions and illustrate policy diversity. This indicates that our method can create a diverse set of policies by simply interpolating between exploration and exploitation policies.

Table 1: Overall performance on MiniGrid (Success Rate in $[0, 1]$) and MinAtar (Episode Return). Bold numbers indicate the method that achieves the best performance. Our method outperforms others in most games with a mild computational cost.

Environment		DQN	Bootstrapped DQN	ϵz-greedy	RND	LESSON	β-DQN (Ours)
MiniGrid	DoorKey	0.44	0.11	0.0	**0.99**	0.86	0.98
	Unlock	0.22	0.17	0.0	0.95	0.64	**0.99**
	SimpleCrossing-Easy	**1.0**	**1.0**	0.95	0.95	0.97	0.99
	SimpleCrossing-Hard	**1.0**	0.81	0.05	0.93	0.6	**1.0**
	LavaCrossing-Easy	0.29	0.66	0.26	0.68	0.75	**0.84**
	LavaCrossing-Hard	0.0	0.01	0.0	**0.39**	0.06	0.16
	Average	0.49	0.46	0.21	0.82	0.65	**0.83**
MinAtar	Asterix	22.78	22.54	18.79	13.4	18.43	**39.09**
	Breakout	16.69	21.88	19.06	14.1	17.71	**29.04**
	Freeway	60.78	59.94	59.68	49.26	54.38	**62.56**
	Seaquest	14.66	14.31	16.98	5.61	9.41	**33.23**
	SpaceInvaders	67.28	69.91	68.7	31.58	55.94	**98.28**
	Average	36.44	36.55	36.64	22.79	31.17	**52.44**
Computational Cost		100%	195.34 %	**94.32 %**	152.57 %	371.07 %	138.78 %
Performance/Computational Cost		1	0.50	0.76	0.75	0.29	**1.13**

Figure 3 further details the online learning process. In the top left, the learning curves show β-DQN outperforms standard DQN by reaching the goal state and obtaining the reward, while DQN primarily learns to avoid the cliff without reaching the goal. This distinction is further illustrated in state coverage shown in the images below. Unlike DQN, which avoids areas near the cliff, β-DQN consistently explores the entire state space, including the challenging regions near the cliff and the goal state. Regarding policy selection, the top right image indicates that β-DQN initially favors space coverage strategies (π_{cov}). Once good coverage is achieved, the exploration shifts towards bias correction strategies (π_{cor}) in this sparse reward environment. This strategic shift highlights β-DQN's adaptability in optimizing exploration to enhance overall learning performance in challenging settings. For a broader perspective, an additional example is provided in Appendix D Figure 12.

5.2 Overall Performance

Environments. We evaluate our method on MiniGrid [14] and MinAtar [62] based on the OpenAI Gym interface [10]. MiniGrid presents numerous tasks in a grid world, characterized by sparse rewards that pose significant challenges in achieving high success rates. MinAtar is an image-based miniaturized version of Atari games [8], preserves the core mechanics of the original games while significantly enhancing processing speed, which facilitates faster model training. In MiniGrid, maps are randomly generated in each episode, and in MinAtar, object placements vary across different time steps, necessitating robust policy generalization. For the evaluation metrics, MiniGrid measures the success rate between 0 and 1, while MinAtar employs episode return.

Baselines and Implementation Details. We compare β-DQN with DQN [40], Bootstrapped DQN [41], ϵz-greedy [15], RND [11], and LESSON [33]. These algorithms are derivatives of DQN, differing primarily in their exploration strategies. RND targets environments with sparse rewards, while the others are general for all kinds of domains. A policy gradient method [47] is included as a reference in Appendix D Figure 13. For fair comparison, we use the same network architecture for all algorithms as used in MinAtar [62]. Learning rates for the baselines are searched over $\{3e^{-4}, 1e^{-4}, 3e^{-5}\}$, reporting the highest performance achieved. Our approach introduces additional hyper-parameters, yet employs a consistent parameters set across all environments. We instantiate the policy set as $\Pi = \{\pi_{\text{cov}(0.05)}, \pi_{\text{cov}(0.1)}, \pi_{\text{cor}(0)}, \pi_{\text{cor}(0.1)}, \pi_{\text{cor}(0.2)}, \cdots, \pi_{\text{cor}(1)}\}$. For the parameter L, we search over {100, 500, 1000, 2000} on SimpleCrossing-Easy and Asterix. We find no significant differences between 500, 1000 and 2000, but observe lower performance for $L = 100$. Based on these results, we fix $L = 1000$ for all environments. For the parameter ϵ, we use a fixed value of $\epsilon = 0.05$, with the sensitivity analysis provided in Appendix D Figure 18. Other common parameters are outlined in Appendix B Table 2. Each algorithm is assessed using 10 different random seeds, with each run consisting of 5 million steps. Performance evaluation occurs every 100k steps over 30 episodes.

Performance. The final performance is presented in Table 1, displaying the mean success rate on MiniGrid and mean return on MinAtar. Our method consistently outperforms others across a diverse range of environments, effectively addressing both dense and sparse reward scenarios. Bootstrapped DQN shows modest improvement on MinAtar and MiniGrid, indicating its generality but limited improvement on performance. However, ϵz-greedy fails to deliver significant improvements on MinAtar and suffers a substantial decrease on MiniGrid, as repetitive unguided actions often result in the agent colliding with obstacles, thereby squandering numerous trials. This inefficiency underscores the limitations of state-independent exploration, even when augmented by temporal persistence. RND excels in sparse reward settings but is the least effective on MinAtar, highlighting its lack of versatility. LESSON, while somewhat improving MiniGrid, performs poorly on MinAtar and incurs a significantly higher computational cost (371%) compared to DQN. In contrast, our method increases computational demand by only 38%. ϵz-greedy run slightly faster than DQN by selecting random actions for random durations, reducing demands on the Q network's inference.

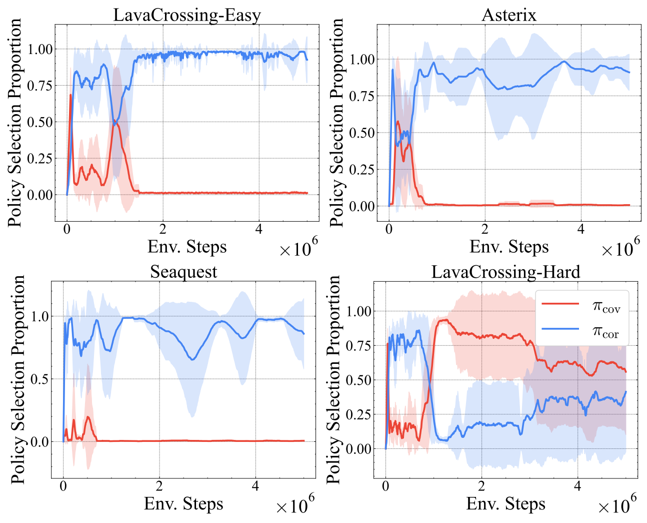


Figure 4: Policy selection varies across different tasks. In simple (LavaCrossing-Easy) or dense reward (Asterix) tasks, exploration primarily corrects estimation biases. In harder tasks (LavaCrossing-Hard), two types of exploration alternate, leading to a more complex policy selection strategy.

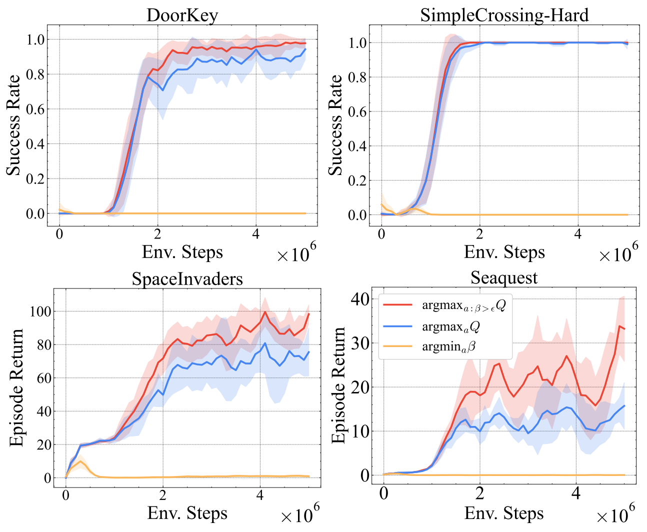


Figure 5: The performance of the three basic polices. $\arg\min_a \beta$ learns nothing since it does not consider rewards. $\arg\max_{a:\beta>\epsilon} Q$ chooses in-sample greedy actions and performs the best. $\arg\max_a Q$ takes greedy actions among the entire action space and may take overestimated actions.

The last row of Table 1 highlights the performance/computational-cost ratio, which considers both performance and computational cost. The ratio is calculated as the performance divided by the computational cost relative to DQN. Our method achieves the highest value of 1.13, indicating superior performance relative to computational cost, while other methods fall short. In summary, β-DQN proves to be general, effective, and computationally efficient. Detailed learning curves, including mean values and confidence intervals, are provided in Figure 13 of Appendix D.

5.3 Analysis of Our Method

Our method construct two types of exploration polices in our policy set based on the behavior function β: π_{cov} for state-action coverage and π_{cor} for overestimation bias correction. A meta-controller is then used to select an effective policy for each episode. We explore several interesting questions: (1) What type of policy in the policy set is preferred by the meta-controller during learning? (2) Which policy performs better, $\arg\max_a Q$ or $\arg\max_{a:\beta(a|s)>\epsilon} Q(s,a)$?

Policy Selection During Learning. For the first question, we illustrate the selection proportions of the two types of polices within the siding window L during the learning process in Figure 4. We group $\{\pi_{\text{cov}(0.05)}, \pi_{\text{cov}(0.1)}\}$ together as π_{cov} for state-action coverage, and $\{\pi_{\text{cor}(0.1)}, \pi_{\text{cor}(0.2)}, \cdots, \pi_{\text{cor}(1)}\}$ together as π_{cor} for overestimation bias correction. We leave $\pi_{\text{cor}(0)}$ in Appendix D.3 Figure 14, as it is a pure exploitation policy.

In simple environments such as LavaCrossing-Easy and dense reward environments such as Asterix, exploring for bias correction plays a more significant role. This suggests that exploring some overestimated actions is sufficient to achieve good performance, without the need to focus extensively on discovering hard-to-reach rewards. In contrast, in hard exploration environments such as LavaCrossing-Hard, the two types of policies interleave, resulting in a more intricate selection pattern. This indicates that relying on a single type of policy may not be enough to achieve good performance in hard exploration environments. Novel states require effort to explore, and overestimated state-actions also need to be corrected.

In summary, our method dynamically selects the exploration policy based on the environment. In simple environments, it is usually more beneficial to find low-hanging-fruit rewards rather than spending much effort exploring novel areas. In hard exploration environments, state-novelty exploration plays a more important role in finding new states with high rewards. This policy selection mechanism parallels the principles of depth-first search (DFS) and breadth-first search (BFS). When encountering positive rewards, our approach adopts a depth-first exploration, delving deeper into the discovered areas for further exploration. Conversely, in the absence of immediate rewards, we shift towards a breadth-first strategy, exploring widely in search of promising areas.

Polices Performance in the Policy Set. For the second question, we show the performance of the three basic polices defined in Equations (3), (4) and (6), which form the basis of our policy set.

As shown in Figure 5, the policy $\arg\min_a \beta$ always selects actions with the lowest probability, disregarding performance and thus learning nothing. The policy $\arg\max_{a:\beta(a|s)>\epsilon} Q(s,a)$ chooses greedy actions that are well-supported in the replay memory and performs the best. The policy $\arg\max_a Q$ takes greedy actions across the entire action space, which may take overestimated actions, resulting in performance that is not as good as the policy $\arg\max_{a:\beta(a|s)>\epsilon} Q(s,a)$. This result aligns with our expectations, highlighting the distinct purposes of the three basic policies.

In some environments such as SimpleCrossing-Hard, $\arg\max_a Q$ performs similar to $\arg\max_{a:\beta(a|s)>\epsilon} Q(s,a)$. This indicates that the space has been fully explored and there is little estimation bias in

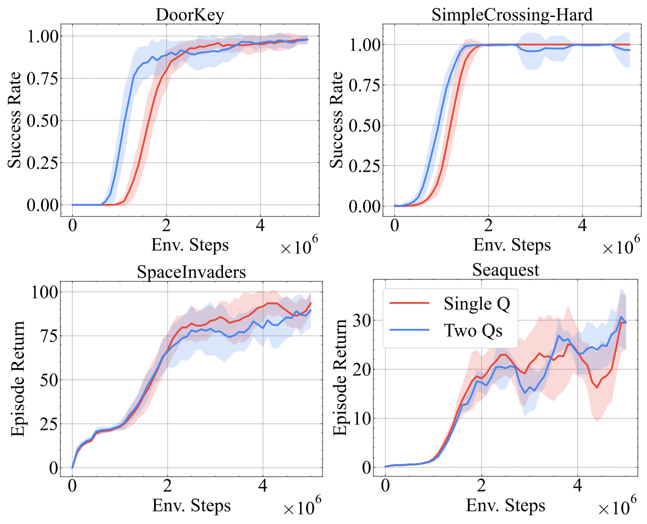


Figure 6: No significant difference is observed between two separate Q functions and a single Q function.

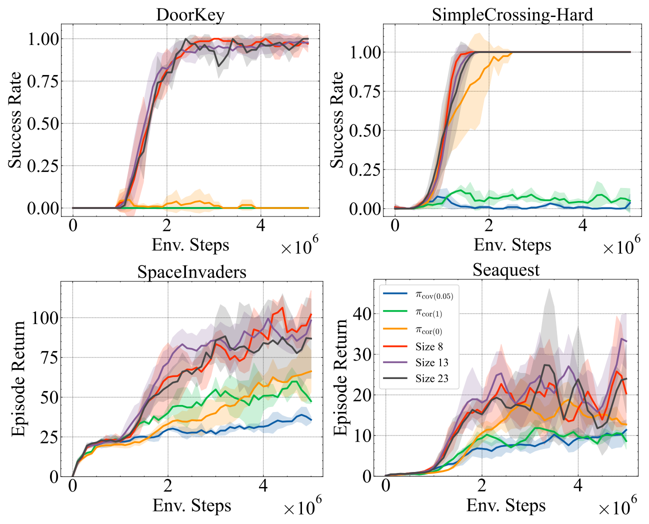


Figure 7: The influence of the policy set size. Performance improves as the policy set size increases, with significant gains observed when all three basic functions are included.

the Q function. In contrast, in environments such as SpaceInvaders, there is a significant performance gap between $\arg\max_a Q$ and $\arg\max_{a:\beta(a|s)>\epsilon} Q(s,a)$. This suggests that many underexplored, overestimated actions still need correction.

5.4 Ablation Studies

In this section, we study two questions: (1) Is there a difference when learning the Q function with and without the constraint of β? (2) Since we can set different values for δ and α to construct the policy set, what is the influence of the policy set size?

Learning Two Separate Q Functions. In our method, we learn a single Q function using Equation (5). Taking the argmax of Q yields an optimistic policy that explores for overestimation bias correction. Masking Q with β before taking the argmax provides a pure exploitation policy. The intuition is that while Equation (5) offers a conservative estimate based on in-distribution data, it may still overestimate at unseen state-action pairs. The practical benefit is that learning one Q function is more computationally efficient.

An alternative approach is to learn two separate Q functions: one for conservative estimation and the other for optimistic estimation. The conservative Q is learned with the constraint of β and then masked to obtain the pure exploitation policy. The optimistic Q is learned without the constraint, and the argmax is taken to derive the optimistic policy and try overestimated actions.

We compare the performance of these two approaches in Figure 6. We find no significant performance difference across environments, indicating that learning a single Q function is sufficient to achieve both conservative and optimistic estimations while being more computationally efficient.

Size of Policy Set. One benefit of our method is that we can construct policy sets of varying sizes without increasing computational cost. By adding different δ and α, we can create larger policy set. We construct policy sets of different sizes and compare their performance, as shown in Figure 7.

The policies $\pi_{\text{cov}(0.05)}, \pi_{\text{cor}(0)}, \pi_{\text{cor}(1)}$ indicate that there is only one policy. And others show the size of the policy set. **Size 8** denotes the policy set $\Pi = \{\pi_{\text{cov}(0.05)}, \pi_{\text{cov}(0.1)}, \pi_{\text{cor}(0)}, \pi_{\text{cor}(0.2)}, \pi_{\text{cor}(0.4)}, \cdots, \pi_{\text{cor}(1)}\}$. **Size 13** denotes the policy set $\Pi = \{\pi_{\text{cov}(0.05)}, \pi_{\text{cov}(0.1)}, \pi_{\text{cor}(0)}, \pi_{\text{cor}(0.1)}, \pi_{\text{cor}(0.2)}, \cdots, \pi_{\text{cor}(1)}\}$, which we used in our main results. **Size 23** denote the policy set $\Pi = \{\pi_{\text{cov}(0.05)}, \pi_{\text{cov}(0.1)}, \pi_{\text{cor}(0)}, \pi_{\text{cor}(0.05)}, \pi_{\text{cor}(0.1)}, \cdots, \pi_{\text{cor}(1)}\}$.

We find that $\pi_{\text{cov}(0.05)}$ does not learn effectively in most of environments, indicating that solely focusing on state-action coverage does no benefit learning. This may be because novel states do not always correlate with improved rewards [7, 49]. Although both $\pi_{\text{cor}(0)}$ and $\pi_{\text{cor}(1)}$ learn something, they perform worse than a larger policy set. This suggests that a single policy is insufficient for achieving good performance due to the lack of diverse exploration. In contrast, combining the three basic polices results in significant performance gains with larger policy set sizes, emphasizeing the importance of diverse exploration. When we increase the policy size to 13 and 23, there is no significant difference. This may indicate the diversity is similar in this two policy sets.

6 CONCLUSION

In this paper, we enhance exploration by constructing a group of diverse polices through the additional learning of a behavior function β from the replay memory using supervised learning. With β, we create a set of exploration policies that range from exploration for state-action coverage to overestimation bias correction. An adaptive meta-controller is then designed to select the most effective policy for interacting with the environment in each episode. Our method is simple, general, and adds minimal computational overhead to DQN. Experiments conducted on MinAtar and MiniGrid demonstrate that our method is effective and broadly applicable in both easy and hard exploration tasks. Future work could extend our method to environments with continuous action spaces.

REFERENCES

[1] Oron Anschel, Nir Baram, and Nahum Shimkin. 2017. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In *International conference on machine learning*. PMLR, 176–185.

[2] Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. 2020. Agent57: Outperforming the atari human benchmark. In *International conference on machine learning*. PMLR, 507–517.

[3] Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martin Arjovsky, Alexander Pritzel, Andrew Bolt, and Charles Blundell. 2020. Never Give Up: Learning Directed Exploration Strategies. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Sye57xStvB

[4] Fengshuo Bai, Runze Liu, Yali Du, Ying Wen, and Yaodong Yang. 2024. RAT: Adversarial Attacks on Deep Reinforcement Agents for Targeted Behaviors. *arXiv preprint arXiv:2412.10713* (2024).

[5] Fengshuo Bai, Hongming Zhang, Tianyang Tao, Zhiheng Wu, Yanna Wang, and Bo Xu. 2023. PiCor: Multi-Task Deep Reinforcement Learning with Policy Correction. *Proceedings of the AAAI Conference on Artificial Intelligence* 37, 6 (Jun. 2023), 6728–6736.

[6] Fengshuo Bai, Rui Zhao, Hongming Zhang, Sijia Cui, Ying Wen, Yaodong Yang, Bo Xu, and Lei Han. 2024. Efficient Preference-based Reinforcement Learning via Aligned Experience Estimation. *arXiv preprint arXiv:2405.18688* (2024).

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

[8] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research* 47 (2013), 253–279.

[9] Richard E. Bellman. 2021. *Dynamic Programming*. Princeton University Press, Princeton. https://doi.org/10.1515/9781400835386

[10] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. *arXiv preprint arXiv:1606.01540* (2016).

[11] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. 2019. Exploration by random network distillation. In *International Conference on Learning Representations*. https://openreview.net/forum?id=H1lJJnR5Ym

[12] Johan Samir Obando Ceron and Pablo Samuel Castro. 2021. Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. In *International Conference on Machine Learning*. PMLR, 1373–1383.

[13] Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo de Lazcano, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. 2023. Minigrid & Miniworld: Modular & Customizable Reinforcement Learning Environments for Goal-Oriented Tasks. *CoRR* abs/2306.13831 (2023).

[14] Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. 2018. Minimalistic Gridworld Environment for OpenAI Gym. https://github.com/maximecb/gym-minigrid.

[15] Will Dabney, Georg Ostrovski, and Andre Barreto. 2021. Temporally-Extended ε-Greedy Exploration. In *International Conference on Learning Representations*. https://openreview.net/forum?id=ONBPHFZ7zG4

[16] Zihan Ding, Tianyang Yu, Hongming Zhang, Yanhua Huang, Guo Li, Quancheng Guo, Luo Mai, and Hao Dong. 2021. Efficient reinforcement learning development with rlzoo. In *Proceedings of the 29th ACM International Conference on Multimedia*. 3759–3762.

[17] Hao Dong, Zihan Ding, and Shanghang Zhang. 2020. *Deep Reinforcement Learning: Fundamentals, Research and Applications*. Springer Nature.

[18] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. 2021. First return, then explore. *Nature* 590, 7847 (2021), 580–586.

[19] Jiajun Fan and Changnan Xiao. 2022. Generalized Data Distribution Iteration. In *International Conference on Machine Learning*. PMLR, 6103–6184.

[20] Jiajun Fan, Yuzheng Zhuang, Yuecheng Liu, Jianye HAO, Bin Wang, Jiangcheng Zhu, Hao Wang, and Shu-Tao Xia. 2023. Learnable Behavior Control: Breaking Atari Human World Records via Sample-Efficient Behavior Selection. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=FeWvD0L_a4

[21] William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. 2020. Revisiting fundamentals of experience replay. In *International Conference on Machine Learning*. PMLR, 3061–3071.

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

[23] Scott Fujimoto, David Meger, Doina Precup, Ofir Nachum, and Shixiang Shane Gu. 2022. Why Should I Trust You, Bellman? The Bellman Error is a Poor Replacement for Value Error. In *International Conference on Machine Learning*. PMLR, 6918–6943.

[24] Aurélien Garivier and Eric Moulines. 2008. On upper-confidence bound policies for non-stationary bandit problems. *arXiv preprint arXiv:0805.3415* (2008).

[25] Andrzej Granas and James Dugundji. 2003. *Fixed point theory*. Vol. 14. Springer.

[26] Zhaohan Guo, Shantanu Thakoor, Miruna Pîslar, Bernardo Avila Pires, Florent Altché, Corentin Tallec, Alaa Saade, Daniele Calandriello, Jean-Bastien Grill, Yunhao Tang, et al. 2022. Byol-explore: Exploration by bootstrapped prediction. *Advances in neural information processing systems* 35 (2022), 31855–31870.

[27] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR, 1861–1870.

[28] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. 2022. CleanRL: High-quality Single-file Implementations of Deep Reinforcement Learning Algorithms. *Journal of Machine Learning Research* 23, 274 (2022), 1–18. http://jmlr.org/papers/v23/21-1342.html

[29] Tommi Jaakkola, Michael Jordan, and Satinder Singh. 1993. Convergence of stochastic iterative dynamic programming algorithms. *Advances in neural information processing systems* 6 (1993).

[30] Daniel Jarrett, Corentin Tallec, Florent Altché, Thomas Mesnard, Rémi Munos, and Michal Valko. 2022. Curiosity in Hindsight: Intrinsic Exploration in Stochastic Environments. *arXiv preprint arXiv:2211.10515* (2022).

[31] Tianying Ji, Yu Luo, Fuchun Sun, Xianyuan Zhan, Jianwei Zhang, and Huazhe Xu. 2023. Seizing serendipity: Exploiting the value of past success in off-policy actor-critic. *arXiv preprint arXiv:2306.02865* (2023).

[32] Steven Kapturowski, Víctor Campos, Ray Jiang, Nemanja Rakicevic, Hado van Hasselt, Charles Blundell, and Adria Puigdomenech Badia. 2023. Human-level Atari 200x faster. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=JtC6yOHRoJJ

[33] Woojun Kim, Jeonghye Kim, and Youngchul Sung. 2023. LESSON: Learning to Integrate Exploration Strategies for Reinforcement Learning via an Option Framework. In *International Conference on Machine Learning*. PMLR.

[34] Aviral Kumar, Abhishek Gupta, and Sergey Levine. 2020. Discor: Corrective feedback in reinforcement learning via distribution correction. *Advances in Neural Information Processing Systems* 33 (2020), 18560–18572.

[35] Tor Lattimore and Csaba Szepesvári. 2020. *Bandit algorithms*. Cambridge University Press.

[36] Long-Ji Lin. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning* 8, 3 (1992), 293–321.

[37] Runze Liu, Fengshuo Bai, Yali Du, and Yaodong Yang. 2022. Meta-Reward-Net: Implicitly Differentiable Reward Learning for Preference-based Reinforcement Learning. In *Advances in Neural Information Processing Systems*, Vol. 35. 22270–22284.

[38] Sam Lobel, Akhil Bagaria, and George Konidaris. 2023. Flipping coins to estimate pseudocounts for exploration in reinforcement learning. In *International Conference on Machine Learning*. PMLR, 22594–22613.

[39] Francisco S Melo. 2001. Convergence of Q-learning: A simple proof. *Institute Of Systems and Robotics, Tech. Rep* (2001), 1–4.

[40] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[41] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. 2016. Deep exploration via bootstrapped DQN. *Advances in neural information processing systems* 29 (2016).

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

[43] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. 2017. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*. PMLR, 2778–2787.

[44] Miruna Pislar, David Szepesvari, Georg Ostrovski, Diana L Borsa, and Tom Schaul. 2022. When should agents explore?. In *International Conference on Learning Representations*. https://openreview.net/forum?id=dEwfxt14bca

[45] Herbert Robbins and Sutton Monro. 1951. A stochastic approximation method. *The annals of mathematical statistics* (1951), 400–407.

[46] Tom Schaul, Andre Barreto, John Quan, and Georg Ostrovski. 2022. The Phenomenon of Policy Churn. In *Advances in Neural Information Processing Systems*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.). https://openreview.net/forum?id=qTCiw1frE_l

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

[48] Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. 2020. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*. PMLR, 8583–8592.

[49] Riley Simmons-Edler, Ben Eisner, Daniel Yang, Anthony Bisulco, Eric Mitchell, Sebastian Seung, and Daniel Lee. 2021. Reward prediction error as an exploration

objective in deep RL. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*. 2816–2823.

[50] Satinder Singh, Tommi Jaakkola, Michael L Littman, and Csaba Szepesvári. 2000. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine learning* 38 (2000), 287–308.

[51] Hao Sun, Lei Han, Rui Yang, Xiaoteng Ma, Jian Guo, and Bolei Zhou. 2022. Optimistic curiosity exploration and conservative exploitation with linear reward shaping. *arXiv preprint arXiv:2209.07288* (2022).

[52] Hao Sun, Zhenghao Peng, Bo Dai, Jian Guo, Dahua Lin, and Bolei Zhou. 2020. Novel policy seeking with constrained optimization. *arXiv preprint arXiv:2005.10696* (2020).

[53] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[54] Csaba Szepesvári. 2010. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning* 4, 1 (2010), 1–103.

[55] Adrien Ali Taiga, William Fedus, Marlos C. Machado, Aaron Courville, and Marc G. Bellemare. 2020. On Bonus Based Exploration Methods In The Arcade Learning Environment. In *International Conference on Learning Representations*. https://openreview.net/forum?id=BJewlyStDr

[56] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. 2017. # exploration: A study of count-based exploration for deep reinforcement learning. *Advances in neural information processing systems* 30 (2017).

[57] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double Q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30.

[58] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8 (1992), 279–292.

[59] Jiayi Weng, Huayu Chen, Dong Yan, Kaichao You, Alexis Duburcq, Minghao Zhang, Yi Su, Hang Su, and Jun Zhu. 2022. Tianshou: A Highly Modularized Deep Reinforcement Learning Library. *Journal of Machine Learning Research* 23, 267 (2022), 1–6. http://jmlr.org/papers/v23/21-1127.html

[60] Chenjun Xiao, Han Wang, Yangchen Pan, Adam White, and Martha White. 2023. The In-Sample Softmax for Offline Reinforcement Learning. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=u-RuvyDYqCM

[61] Tianpei Yang, Hongyao Tang, Chenjia Bai, Jinyi Liu, Jianye Hao, Zhaopeng Meng, Peng Liu, and Zhen Wang. 2021. Exploration in deep reinforcement learning: a comprehensive survey. *arXiv preprint arXiv:2109.06668* (2021).

[62] Kenny Young and Tian Tian. 2019. MinAtar: An Atari-Inspired Testbed for Thorough and Reproducible Reinforcement Learning Experiments. *arXiv preprint arXiv:1903.03176* (2019).

[63] Hongming Zhang, Tongzheng Ren, Chenjun Xiao, Dale Schuurmans, and Bo Dai. 2024. Provable Representation with Efficient Planning for Partially Observable Reinforcement Learning. In *Forty-first International Conference on Machine Learning*. https://openreview.net/forum?id=O6tenHWTUU

[64] Hongming Zhang, Chenjun Xiao, Chao Gao, Han Wang, bo xu, and Martin Müller. 2024. Exploiting the Replay Memory Before Exploring the Environment: Enhancing Reinforcement Learning Through Empirical MDP Iteration. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=lsd27JUJ8v

[65] Hongming Zhang, Chenjun Xiao, Han Wang, Jun Jin, Bo Xu, and Martin Müller. 2023. Replay Memory as An Empirical MDP: Combining Conservative Estimation with Experience Replay. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=SjzFVSJUt8S

[66] Hongming Zhang and Tianyang Yu. 2020. AlphaZero. *Deep Reinforcement Learning: Fundamentals, Research and Applications* (2020), 391–415.

[67] Hongming Zhang and Tianyang Yu. 2020. Taxonomy of reinforcement learning algorithms. *Deep Reinforcement Learning: Fundamentals, Research and Applications* (2020), 125–133.

[68] Tianjun Zhang, Huazhe Xu, Xiaolong Wang, Yi Wu, Kurt Keutzer, Joseph E Gonzalez, and Yuandong Tian. 2020. Bebold: Exploration beyond the boundary of explored regions. *arXiv preprint arXiv:2012.08621* (2020).