OPTIMISTIC EXPLORATION WITH BACKWARD BOOTSTRAPPED BONUS FOR DEEP REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Optimism in the face of uncertainty and *posterior sampling* are two principled approaches for provably efficient exploration in tabular and linear settings. However, these principles are challenging in developing practical exploration algorithms for Deep Reinforcement Learning (DRL). To address this problem, we propose an Optimistic Exploration algorithm with Backward Bootstrapped Bonus (OEB3) for DRL by following these two principles. OEB3 is built on bootstrapped deep Q-learning, a non-parametric posterior sampling method to perform temporally-extended exploration. We propose an UCB-bonus indicated the uncertainty of Q-functions to construct an optimistic Q-value, which encourages the agent to explore the meaningful states and actions to reduce this uncertainty. We use the episodic backward update to perform sample-efficient training. OEB3 outperforms several state-of-the-art exploration approaches in Mnist maze and 49 Atari games.

1 INTRODUCTION

In Reinforcement learning (RL) (Sutton & Barto, 2018) formalized by the Markov decision process (MDP), the agent aims to maximize the long-term reward by interacting with the unknown environment. The agent takes actions according to the knowledge of experiences, which leads to a fundamental problem of the exploration-exploitation dilemma. The agent may choose the best decision given current information or instead acquires more information by exploring the poorly understood states and actions. Exploring the environment may sacrifice immediate rewards but potentially improves future performance. The exploration strategy is crucial for the RL agent to find the optimal policy.

The theoretical RL offers a variety of provably efficient exploration methods in tabular and linear MDPs with the basic value iteration algorithm: least-squares value iteration (LSVI). *Optimism in the face of uncertainty* (Auer & Ortner, 2007; Jin et al., 2018) and *posterior sampling* (Osband et al., 2013; Osband & Van Roy, 2016; Osband et al., 2019) are two principled exploration approaches. In tabular cases, the optimism-based methods incorporate upper confidence bound (UCB) into the value functions as bonuses and attain the optimal worst-case regret (Azar et al., 2017; Jaksch et al., 2010; Dann & Brunskill, 2015). Randomized value function based on posterior sampling chooses actions according to the randomly sampled statistically plausible value functions and is known to achieve near-optimal worst-case and Bayesian regrets (Osband & Van Roy, 2017; Russo, 2019). Recently, the theoretical analyses in tabular cases were extended to linear MDP where the transition and the reward function are assumed to be linear. In linear cases, optimistic LSVI (Jin et al., 2020b) is the first provably efficient algorithm to attain a near-optimal worst-case regret by using a designed bonus. Randomized LSVI (Zanette et al., 2019) is also shown to attain a near-optimal worst-case regret.

Although the analyses in tabular and linear cases provide valuable principles for the efficient exploration, these principles are still challenging in developing a practical exploration algorithm for Deep Reinforcement Learning (DRL). For example, in linear cases, the specialized bonus in optimistic LSVI (Jin et al., 2020b) and nontrivial noise in randomized LSVI (Zanette et al., 2019) are tailored to linear models (Abbasi-Yadkori et al., 2011) rather than practically powerful function approximations (e.g., neural network). DRL (Mnih et al., 2015; Fujimoto et al., 2018; Schulman et al., 2017) has achieved human-level performance in large-scale tasks like Atari games (Machado et al., 2018; Kaiser

et al., 2020) and robotic tasks (Andrychowicz et al., 2020; 2017; Fang et al., 2019), it is valuable to extend the theoretical works in exploration to DRL.

To address this problem, we propose an Optimistic Exploration algorithm with Backward Bootstrapped Bonus (OEB3) for DRL. OEB3 is an instantiation of optimistic LSVI (Jin et al., 2020b) in DRL by using a general-purpose UCB-bonus to provide an optimistic Q-value and a randomized value function to perform temporally-extended exploration. Specifically, we use bootstrapped DQN (Osband et al., 2016), a non-parametric posterior sampling method with randomized value function as the basis of OEB3. Bootstrapped DQN constructs a posterior distribution of Q-value by training several bootstrapped Q-functions. We propose an UCB-bonus that represents the disagreement of these Q-functions to measure the epistemic uncertainty of the unknown optimal value function. Q-value added by UCB-bonus becomes an optimistic Q^+ function that is higher than Q for novel state-action pairs and remains close to Q for frequently visited ones. The optimistic Q^+ function encourages the agent to explore the states and actions with high UCB-bonuses, which signify novel areas or meaningful events in completing a task. When these states and actions are explored more often, the bootstrapped value functions gradually concentrate around the true value, and the corresponding UCB-bonuses are reduced. OEB3 uses an extension of the episodic backward update (Lee et al., 2019) to perform sample-efficient training and guarantee the theoretical consistency with optimistic LSVI (Jin et al., 2020b). We evaluate OEB3 on Mnist maze and 49 Atari games. The results show that our method outperforms several state-of-the-art exploration approaches.

2 Background

OEB3 is built on bootstrapped DQN (Osband et al., 2016) to construct the UCB-bonus and perform temporally-consistent exploration. We consider OEB3 as an instantiation of optimistic LSVI (Jin et al., 2020b) in DRL.

2.1 Bootstrapped DQN

Considering an episodic MDP represented as $(\mathcal{S}, \mathcal{A}, T, \mathbb{P}, r)$, where $T \in \mathbb{Z}_+$ is the episode length, $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, r is the reward function, and $\mathbb{P}$ is the unknown dynamics. In each timestep, the agent obtains the current state s_t, takes an action a_t, interacts with the environment, receives a reward r_t, and updates to the next state s_{t+1}. The action-value function $Q^\pi(s_t, a_t) := \mathbb{E}_\pi\big[\sum_{i=t}^{T-1} \gamma^{i-t} r_i\big]$ represents the expected cumulative reward starting from state s_t, taking action a_t, and thereafter following policy $\pi(a_t|s_t)$ until the end of the episode. $\gamma \in [0, 1)$ is the discount factor. The optimal value function $Q^* = \max_\pi Q^\pi$, and the optimal action $a^* = \arg\max_{a\in\mathcal{A}} Q^*(s, a)$.

Deep Q-Network (DQN) uses a deep neural network with parameters θ to approximate the Q-function. The loss function is $L(\theta) = \mathbb{E}[(y_t - Q(s_t, a_t; \theta))^2 \big| (s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}]$, where θ^- is the target network, and $y_t = r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-)$ is the target value. The agent accumulates experiences (s_t, a_t, r_t, s_{t+1}) in a replay buffer $\mathcal{D}$ and samples mini-batches in training.

Bootstrapped DQN (Osband et al., 2016; 2018) is a non-parametric posterior sampling method and maintains K estimates of Q-values to represent the posterior distribution of the randomized value function. Bootstrapped DQN uses a multi-head network that contains a shared convolution network and K heads. Each head defines a Q^k-function. Bootstrapped DQN diversifies different Q^k by using different random initialization and individual target networks. The loss for training Q^k with (s_t, a_t, r_t, s_{t+1}) is

$$L(\theta^k) = \mathbb{E}\Big[\big(r_t + \gamma \max_{a'} Q^k(s_{t+1}, a'; \theta^{k-}) - Q^k(s_t, a_t; \theta^k)\big)^2 \Big| (s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}\Big]. \quad (1)$$

The k-th head $Q^k(s, a; \theta^k)$ is trained with its own target network $Q^k(s, a; \theta^{k-})$. If k-th head is sampled at the start of an episode when interacting with the environment, the agent will follow Q^k to choose actions in the whole episode, which provides temporally-consistent exploration for DRL.

2.2 Optimistic LSVI

Optimistic LSVI (Jin et al., 2020b) uses an optimistic Q-value with LSVI in linear MDP. We denote the feature map of the state-action pair as $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$. The transition kernel and

Algorithm 1 Optimistic LSVI in linear MDP

1: **Initialize:** $\Lambda_t \leftarrow \lambda \cdot \mathbf{I}$ and $w_h \leftarrow 0$
2: **for** episode $m = 0$ **to** $M - 1$ **do**
3: Receive the initial state s_0
4: **for** step $t = 0$ **to** $T - 1$ **do**
5: Take action $a_t = \arg\max_{a \in \mathcal{A}} Q_t(s_t, a)$ and observe s_{t+1}.
6: **end for**
7: **for** step $t = T - 1$ **to** 0 **do**
8: $\Lambda_t \leftarrow \sum_{\tau=0}^{m} \phi(x_t^\tau, a_t^\tau)\phi(x_t^\tau, a_t^\tau)^\top + \lambda \cdot \mathbf{I}$
9: $w_t \leftarrow \Lambda_t^{-1} \sum_{\tau=0}^{m} \phi(x_t^\tau, a_t^\tau)[r_t(x_t^\tau, a_t^\tau) + \max_a Q_{t+1}(x_{t+1}^\tau, a)]$
10: $Q_h(\cdot,\cdot) = \min\{w_t^\top \phi(\cdot,\cdot) + \alpha[\phi(\cdot,\cdot)^\top \Lambda_t^{-1} \phi(\cdot,\cdot)]^{1/2}, T\}$
11: **end for**
12: **end for**

reward function are assumed to be linear to ϕ. There exists weight w_t for any policy π to make $Q_t(s,a) = w_t^\top \phi(s,a)$ (Jin et al., 2020b). Optimistic LSVI shown in Algorithm 1 consists of two processes. In line 3-6, the agent executes the policy according to Q_t for an episode. In line 7-11, the parameter w_t of Q-function is updated in closed-form by following the regularized least-squares problem

$$w_t \leftarrow \arg\min_{w \in \mathbb{R}^d} \sum_{\tau=0}^{m} \left[r_t(s_t^\tau, a_t^\tau) + \max_{a \in \mathcal{A}} Q_{t+1}(s_{t+1}^\tau, a) - w^\top \phi(s_t^\tau, a_t^\tau) \right]^2 + \lambda \|w\|^2, \tag{2}$$

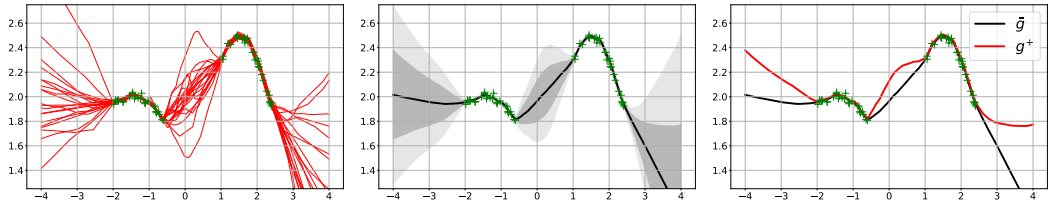
where m is the number of episodes, and τ is the episodic index. The least-squares method solves $w_t = \Lambda_t^{-1} \sum_{\tau=0}^{m} \phi(x_t^\tau, a_t^\tau) \left[r_t(x_t^\tau, a_t^\tau) + \max_a Q_{t+1}(x_{t+1}^\tau, a) \right]$ (line 9), where Λ_t is the Gram matrix. Then the value function $Q_t(s,a) \approx w_t^\top \phi(s,a)$. The $\min(\cdot, T)$ (line 10) represents the Q-value is bounded in $[0, T]$ because the reward is bounded in $[0, 1]$.

Optimistic LSVI uses a bonus $\alpha[\phi(s,a)^\top \Lambda_t^{-1} \phi(s,a)]^{1/2}$ (line 10) to measure the uncertainty of state-action pairs. We can intuitively consider $u := (\phi^\top \Lambda_t^{-1} \phi)^{-1}$ as a pseudo count of state-action pair by projecting the total features that have observed (Λ_t) to the ϕ direction. Then the bonus $\alpha/\sqrt{u}$ represents the uncertainty along the ϕ direction. Q-value added by the bonus becomes an optimistic value function Q^+, which serves as an upper bound of Q to encourage exploration. The bonus in each step is propagated from the end of the episode with the update of Q-value (line 7-11), which follows the principle of dynamic programming. The theoretical result shows optimistic LSVI suffers a near-optimal worst-case regret with the designed bonus and proper parameters α and λ.

3 Proposed Method

Optimistic LSVI (Jin et al., 2020b) provides valuable principles for efficient exploration but still challenging in developing a practical exploration algorithm for DRL. We propose OEB3 as an instantiation of optimistic LSVI in DRL by focusing on two aspects as follows.

- Proposing a general-purpose UCB-bonus for optimistic exploration. Optimistic LSVI uses a designed bonus $\alpha[\phi(s,a)^\top \Lambda_t^{-1} \phi(s,a)]^{1/2}$ for closed-form solution in linear MDP. The bonus projects the total features to a specific feature direction, which is limited in practical application because, due to the curse of dimensionality, state spaces in DRL of practical interest are enormous. There is a need for the bonus term to generalize across high-dimensional states while exploring intelligently within less computing overhead. We address this problem in Section 3.1 by using bootstrapped DQN to construct a general-purpose UCB-bonus for OEB3.
- Proposing a sample-efficient learning algorithm to integrate bootstrapped DQN and UCB-bonus into the backward update (line 7-11 in Alg. 1) of optimistic LSVI, which faithfully follows the principle of dynamic programming. We address this problem in Section 3.2 by extending Episodic Backward Update (EBU) (Lee et al., 2019) from ordinary Q-learning to bootstrapped Q-learning, and we denote this method as BEBU (Bootstrapped EBU). BEBU allows fast propagation of bonuses and rewards for sample-efficient learning. The action selection (line 3 in Alg. 1)



(a) (b) (c)

Figure 1: Illustration of UCB-bonus by using bootstrapped estimates in the regression task. The green dots represent 60 data points. (a) Regression curves of 20 neural networks. (b) Mean estimate (black) and uncertainty measurement (shadow). (c) The optimistic value (red) and mean value (black).

corresponds to take actions with respect to a randomized value function for temporally-extended exploration.

3.1 UCB-Bonus in OEB3

Optimistic exploration uses an optimistic action-value function Q^+ to incentive exploration by adding a bonus term to the ordinary Q-value. Thus Q^+ serves an upper bound of ordinary Q. The bonus term should represent the epistemic uncertainty that results from lacking experiences of the corresponding states and actions. The design of the bonus is crucial for optimistic exploration. In this paper, we propose an UCB-bonus $\mathcal{B}(s_t, a_t)$ by measuring the disagreement of the bootstrapped Q-values $\{Q^k(s_t, a_t)\}_{k=1}^K$ of the state-action pair (s_t, a_t) in bootstrapped DQN, as

$$\mathcal{B}(s_t, a_t) := \tilde{\sigma}\big(Q^k(s_t, a_t)\big) = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\Big(Q^k(s_t, a_t) - \bar{Q}(s_t, a_t)\Big)^2}, \tag{3}$$

where $\bar{Q}(s_t, a_t)$ is the mean of bootstrapped Q-values.

Theorem 1. *In linear function approximation, the UCB-bonus $\mathcal{B}(s_t, a_t)$ in OEB3 is equivalent to the bonus-term $[\phi_t^\top \Lambda_t^{-1} \phi_t]^{1/2}$ in optimistic-LSVI, where $\Lambda_t \leftarrow \sum_{\tau=0}^{m} \phi(x_t^\tau, a_t^\tau)\phi(x_t^\tau, a_t^\tau)^\top + \lambda \cdot \mathbf{I}$, and m is the current episode. (informal)*

The proof is given in Appendix A. In DRL, $\mathcal{B}(s_t, a_t)$ is reasonable for exploration because

- Bootstrapped DQN is a non-parametric posterior sampling method that maintains a randomized value function to represent the approximate posterior of the Q-value. The UCB-bonus measured by the difference between the bootstrapped Q-values captures the epistemic uncertainty of the optimal value function (Osband et al., 2019).
- Because the agent lacks experiences about the poorly-explored states and actions, different Q^k parametrized by deep neural networks will generalize differently due to the non-convexity of neural network and independent initialization, then the corresponding UCB-bonuses will be high. With more experiences are gathered, different Q^k will concentrate around the true value function and the corresponding UCB-bonuses will decrease. The UCB-bonuses become close to 0 if the agent explores the corresponding state-action pairs frequently.
- The UCB-bonuses are computed in a batch when performing experience replay, which is more efficient than other optimistic methods (Chen et al., 2017; Nikolov et al., 2019) that change the action-selection scheme in each timestep. We randomly sample a Q^k for action-selection in each episode as in bootstrapped DQN.

Adding by the proposed UCB-bonus, the optimistic Q^+ gives higher value estimates than Q for poorly-explored state-action pairs and remains close to Q for frequently visited ones. Formally,

$$Q^+(s_t, a_t) := Q(s_t, a_t) + \alpha\mathcal{B}(s_t, a_t). \tag{4}$$

We use a simple regression task with neural networks to illustrate the proposed UCB-bonus, as shown in Figure 1. We use 20 neural networks with the same network architecture to perform

regression. Each network contains three residual blocks with two fully-connected layers each. By following (Osband et al., 2016), the difference between networks relies on different initialization. For a single input x, the networks provide different estimates as $\{g_i(x)\}_{i=1}^{20}$. According to the regression curves in Figure 1(a), all g_i perform well in fitting the data, but different networks lead to various generalizations in areas with no data. In Figure 1(b), we show the confidence bound of the regression result as $\bar{g}(x) \pm \tilde{\sigma}(g_i(x))$ and $\bar{g}(x) \pm 2\tilde{\sigma}(g_i(x))$, where $\bar{g}(x)$ and $\tilde{\sigma}(g_i(x))$ are the mean and standard deviation of estimates. $\tilde{\sigma}(g_i(x))$ measured by the disagreement of estimates captures the epistemic uncertainty of regression results. Figure 1(c) shows the optimistic estimation as $g^+(x) = \bar{g}(x) + \tilde{\sigma}(g_i(x))$ by adding the uncertainty. g^+ is close to $\bar{g}$ in areas with data points and higher than $\bar{g}$ in areas that lack data.

In DRL, the bootstrapped Q-functions $\{Q^k(s_t, a_t)\}_{k=1}^{K}$ that regresses the target Q-value perform similarly as $\{g_i(x)\}_{i=1}^{20}$ in the above regression task. A higher UCB-bonus $\mathcal{B}(s_t, a_t) := \tilde{\sigma}(Q^k(s_t, a_t))$ indicates a higher epistemic uncertainty of the action-value function with (s_t, a_t). Q-function augmented with the UCB-bonus yields Q^+ that produces optimistic estimates for novel state-action pairs and remains close to Q in areas that have been explored by the agent, which incentives the agent to explore the potentially informative states and actions efficiently.

3.2 Bootstrapped Episodic Backward Update

There are two major reasons for updating the action-value function in a backward manner.

- First, the backward propagation utilizes a complete trajectory from replay buffer. Such an approach allows OEB3 to infer long-term effect of decision making from the replay buffer. In contrast, DQN and Bootstrap DQN utilize the replay buffer by random sampling one-step transitions from the replay buffer, which loses the information containing such a long-term effect (Lee et al., 2019).
- Second, the backward propagation is required to propagate the future uncertainty to the estimated action-value function consistently within an episode. If the estimation is not conducted in a backward manner, it can occur that the uncertainty propagated into Q_{t_1} is not consistent with the uncertainty of Q_{t_2} with $t_2 > t_1$, where Q_{t_1} is updated before the update of Q_{t_2}.

To integrate the UCB-bonus into Q-value, we propose a novel Q-target in update by adding the bonus term in both the immediate reward and the next-Q value. The proposed Q-target is also suitable for BEBU for sample-efficient training. Formally, the Q-target for updating Q^k is defined as

$$y_t^k := \big[r(s_t, a_t) + \alpha_1 \mathcal{B}(s_t, a_t; \theta)\big] + \gamma\big[Q^k(s_{t+1}, a'; \theta^{k-}) + \alpha_2 \mathbb{1}_{a' \neq a_{t+1}} \tilde{\mathcal{B}}^k(s_{t+1}, a'; \theta^-)\big]. \quad (5)$$

where $a' = \arg\max_a Q^k(s_{t+1}, a; \theta^{k-})$. The choice of a' is determined by the target Q-value without considering the bonus. The immediate reward is added by $\mathcal{B}(s_t, a_t; \theta)$ with a factor α_1, where the bonus $\mathcal{B}$ is computed by bootstrapped Q-network with parameter θ. The next-Q value is added by $\mathbb{1}_{a' \neq a_{t+1}} \tilde{\mathcal{B}}^k(s_{t+1}, a'; \theta^-)$ with factor α_2, where the bonus $\tilde{\mathcal{B}}^k$ is computed by the target network with parameter θ^-. Different heads have different $\tilde{\mathcal{B}}^k$ because the choice of a' is different, while $\mathcal{B}$ for immediate reward is the same in all heads. The indicator function $\mathbb{1}_{a' \neq a_{t+1}}$ is used because the Q-values are backward updated. When updating step t, the $Q^k(s_{t+1}, a_{t+1})$ has already been updated as an optimistic estimate, thus we ignore the bonus when a' is equal to a_{t+1}.

We use an example to explain the process of backward update. The experiences are stored and sampled in episodes, and the update is also performed in episodes rather than uniformly sampled transitions. Considering a tabular MDP that an episode contains three time steps as $(s_0, a_0) \rightarrow (s_1, a_1) \rightarrow (s_2, a_2)$, then the Q-value in head k will be updated with backward order as $Q(s_2, a_2) \rightarrow Q(s_1, a_1) \rightarrow Q(s_0, a_0)$ from the end of the episode. We describe the process as follows. The superscript k is ignored for convenience.

1. Firstly, we update $Q(s_2, a_2) \leftarrow r(s_2, a_2) + \alpha_1 \mathcal{B}(s_2, a_2)$. Note that the next-$Q$ value does not exist in the last timestep.
2. Then, $Q(s_1, a_1) \leftarrow [r(s_1, a_1) + \alpha_1 \mathcal{B}(s_1, a_1)] + [Q(s_2, a') + \alpha_2 \mathbb{1}_{a' \neq a_2} \tilde{\mathcal{B}}(s_2, a')]$ by following (5), where $a' = \arg\max_a Q(s_2, a)$. Because $Q(s_2, a_2)$ has been updated as an optimistic value in step 1, we ignore the bonus-term $\tilde{\mathcal{B}}$ in next-Q value when $a' = a_2$. The UCB-bonus is augmented in update by adding $\mathcal{B}$ and $\tilde{\mathcal{B}}$ to the immediate reward and next-Q value, respectively.

3. The update of $Q(s_0, a_0)$ follows the same principle. The optimistic Q-value is $Q(s_0, a_0) \leftarrow [r(s_0, a_0) + \alpha_1 \mathcal{B}(s_0, a_0)] + [Q(s_1, a') + \alpha_2 \mathbb{1}_{a' \neq a_1} \tilde{\mathcal{B}}(s_1, a')]$, where $a' = \arg\max_a Q(s_1, a)$.

In practice, the episodic update usually leads to instability in DRL due to the strong correlation in consecutive transitions. A diffusion factor $\beta \in [0, 1]$ is proposed in EBU (Lee et al., 2019) to prevent instability, and we also use this factor for BEBU. Q-value is the weighted sum of the current value and back-propagated estimate with factor β. Considering an episodic experience that contains T transitions as $E = \{\mathbf{S}, \mathbf{A}, \mathbf{R}, \mathbf{S}'\}$, where $\mathbf{S} = \{s_0, \dots, s_{T-1}\}$, $\mathbf{A} = \{a_0, \dots, a_{T-1}\}$, $\mathbf{R} = \{r_0, \dots, r_{T-1}\}$ and $\mathbf{S}' = \{s_1, \dots, s_T\}$. A Q-table $\tilde{\mathbf{Q}} \in \mathbb{R}^{K \times |\mathcal{A}| \times T}$ initialized by $Q(\cdot; \theta^-)$ is used to store the next Q-values of all next-state $\mathbf{S}'$ and valid actions for K heads. We initialize $\mathbf{y} \in \mathbb{R}^{K \times T}$ for storing the Q-target for K heads and T steps. We use bootstrapped Q-network with parameters θ to compute the bonus $\mathbf{B} = [\mathcal{B}(s_0, a_0), \dots, \mathcal{B}(s_{T-1}, a_{T-1})]$ for immediate reward, and use the target network with parameters θ^- to compute bonus $\tilde{\mathbf{B}}^k = [\tilde{\mathcal{B}}^k(s_1, a'_1), \dots, \tilde{\mathcal{B}}^k(s_T, a'_T)]$ for next-Q value in each head, where $a'_t = \arg\max_a Q^k(s_t, a; \theta^{k-})$. The bonus vector $\mathbf{B} \in \mathbb{R}^T$ is the same for all Q-heads, while $\tilde{\mathbf{B}} \in \mathbb{R}^{K \times T}$ contains different values for different heads because the choices of a'_t are different.

In the training of head k, Q-target in the last step is initialized as $\mathbf{y}[k, T-1] = \mathbf{R}_{T-1} + \alpha_1 \mathbf{B}_{T-1}$, and there is no next Q-value exists. Then we perform a recursive backward update to get all Q-target values. The element of $\tilde{\mathbf{Q}}[k, a_{t+1}, t]$ for step t in head k is updated by using its Q-target $\mathbf{y}[k, t+1]$ with the diffusion factor, as

$$\tilde{\mathbf{Q}}[k, a_{t+1}, t] \leftarrow \beta \mathbf{y}[k, t+1] + (1 - \beta) \tilde{\mathbf{Q}}[k, a_{t+1}, t]. \tag{6}$$

Then $\mathbf{y}[k, t]$ in the previous time step is updated by taking the newly updated t-th column of $\tilde{\mathbf{Q}}[k]$, as

$$\mathbf{y}[k, t] \leftarrow \big(\mathbf{R}_t + \alpha_1 \mathbf{B}_t\big) + \gamma\big(\tilde{\mathbf{Q}}[k, a', t] + \alpha_2 \mathbb{1}_{a' \neq a_{t+1}} \tilde{\mathbf{B}}[k, t]\big), \tag{7}$$

where $a' = \arg\max_a \tilde{\mathbf{Q}}[k, a, t]$. In practice, we construct a matrix $\tilde{\mathbf{A}} = \arg\max_a \tilde{\mathbf{Q}}[\cdot, a, \cdot] \in \mathbb{R}^{K \times T}$ to gather all a' of next-Q. Then a mask matrix $\mathbf{M} \in \mathbb{R}^{K \times T}$ is constructed to represent whether $\tilde{\mathbf{A}}$ is identical to the executed action in the corresponding timestep. The bonus of next-Q is the element-wise product of $\mathbf{M}$ and $\tilde{\mathbf{B}}$ with factor α_2. After the backward update, we compute the Q-value of $(\mathbf{S}, \mathbf{A})$ as $\mathbf{Q} = Q(\mathbf{S}, \mathbf{A}; \theta) \in \mathbb{R}^{K \times T}$. The loss function is $L(\theta) = \mathbb{E}\big[(\mathbf{y} - \mathbf{Q})^2 | (s_t, a_t, r_t, s_{t+1}) \in E,\ E \sim \mathcal{D}\big]$, where the episodic experience E is sampled from replay buffer to perform gradient descent. The gradient of all heads can be computed simultaneously through the BEBU method.

UCB-bonuses are propagated from the end of the episode along with the update of Q-functions through BEBU, which faithfully follows the backward update of optimistic LSVI in Algorithm 1 to guarantee the theoretical consistency. The backward update significantly improves the sample-efficiency by allowing bonuses and delayed rewards to propagate directly through transitions of the sampled episode. The full algorithm of OEB3 with UCB-bonus and BEBU is given in Appendix B.

4 Related Work

One practical principle for exploration in DRL is maintaining epistemic uncertainty of action-value functions and learning to reduce this uncertainty. Epistemic uncertainty appears due to the missing knowledge of the environment, and tends to disappear with the progress of exploration. Bootstrapped DQN (Osband et al., 2016; 2018) samples Q-values from the randomized value functions to encourage exploration through Thompson sampling. Chen et al. (2017) shows optimistic action-selection gets better performance than randomly sampled Q-heads. Information-Directed Sampling (IDS) (Nikolov et al., 2019) is based on bootstrapped DQN and chooses actions by balancing the instantaneous regret and information gain. OAC (Ciosek et al., 2019) uses two Q-networks to get lower and upper bounds of Q-value to perform exploration in continuous control tasks. These methods seek to estimate the epistemic uncertainty in action-selection and choose the optimistic actions. In contrast, we focus on exploration with intrinsic reward by exploiting the uncertainty of value function. Other methods that utilizes the epistemic uncertainty of dynamics through Bayesian posterior (Ratzlaff et al., 2020) and ensembles (Pathak et al., 2019) are orthogonal to our contribution.

Methods for constructing intrinsic rewards in exploration often use pseudo count of state (Bellemare et al., 2016; Ostrovski et al., 2017; Choi et al., 2019; Rashid et al., 2020), episodic curiosity (Savinov

et al., 2019), representation changes (Raileanu & Rocktäschel, 2020), curiosity-bottleneck (Kim et al., 2019b), information gain (Houthooft et al., 2016) and random network distillation (RND) (Burda et al., 2019b) to measure the novelty of states. Never give up (Badia et al., 2020b) and Agent-57 (Badia et al., 2020a) combines pre-episode and life-long novelty. The 'curiosity' exploration based on prediction-error of environment models like ICM (Pathak et al., 2017; Burda et al., 2019a), EMI (Kim et al., 2019a), self-aware agent (Haber et al., 2018) and learning progress (Kim et al., 2020) enable the agents to do reward-free exploration (Jin et al., 2020a). However, according to an empirically research (Taiga et al., 2020) on intrinsic rewards, the promising results of bonus-based exploration in hard exploration tasks does not benefit the performance of the whole Atari suit. The research on intrinsic motivation should be carried on various tasks rather than a several domains. We follow this principle by evaluating our method on 49 Atari games.

5 Experimental Results

We evaluate the algorithms in Mnist Maze and 49 Atari games. The experiments about Mnist Maze are given in Appendix C, and we discuss the Atari experiments in the main body. Our method uses BEBU for training. Comparing our method with bootstrapped DQN-based baselines will have performance bias due to the difference between the basic learning algorithms. To eliminate such bias, we reimplement all bootstrapped DQN-based baselines with BEBU-based update. We compare the following methods in our experiment: (1) **OEB3**. The proposed optimistic exploration method with UCB-bonus and backward update. (2) **BEBU** (Osband et al., 2016). The re-implementation of bootstrapped DQN with BEBU-based training. (3) **BEBU-UCB** (Chen et al., 2017). BEBU with optimistic action-selection according to the upper bound of Q. (4) **BEBU-IDS** (Nikolov et al., 2019). Integrating homoscedastic IDS into BEBU without distributional RL (Bellemare et al., 2017).

We set α_1 and α_2 to the same value of 0.5×10^{-4} by searching coarsely. We use diffusion factor $\beta = 0.5$ for all methods by following (Lee et al., 2019). The hyper-parameters of BEBU-UCB and BEBU-IDS follow Chen et al. (2017) and Nikolov et al. (2019). The proposed OEB3 needs much less training time than BEBU-UCB and BEBU-IDS because the confidence bound used in BEBU-UCB and regret-information ratio used in BEBU-IDS are both computed in each time step when interacting with the environment, while the UCB-bonuses in OEB3 are calculated in episodes when performing batch training. The environmental interaction steps are usually much more than the training steps (e.g., in standard DQN, the interacting steps are four times the training steps), thus OEB3 needs less training time than BEBU-UCB and BEBU-IDS. The action-selection in OEB3 follows BEBU by randomly sampling a Q-head for an episode. All methods use the same network architecture as in bootstrapped DQN with $K = 10$ heads. Detailed specifications for all methods are described in Appendix D. The code is available at `https://github.com/review-anon/OEB3`.

According to EBU (Lee et al., 2019), the backward update is significantly more sample-efficient than ordinary Q-learning by using 20M training frames to achieve the mean human-normalized score of DQN with 200M training frames. We follow this setting by training all BEBU-based methods with 20M frames. In our experiments, 20M frames in OEB3 is enough to produce fairly strong empirical results and achieve competitive results with several baselines in 200M frames. After training, an ensemble policy by a majority vote of bootstrapped heads is used for evaluation (Osband et al., 2016). The agent is evaluated by 30 no-op evaluation. We use the popular human-normalized score $\frac{\text{Score}_{\text{Agent}} - \text{Score}_{\text{Random}}}{|\text{Score}_{\text{human}} - \text{Score}_{\text{random}}|}$ to perform the comparison. We report the mean and the median human-normalized scores in Table 1. The raw scores are reported in Appendix F. OEB3 outperforms BEBU, BEBU-UCB, and BEBU-IDS in both the mean and median of human-normalized scores. We provide the relative scores as $\frac{\text{Score}_{\text{Agent}} - \text{Score}_{\text{Baseline}}}{\max\{\text{Score}_{\text{human}}, \text{Score}_{\text{baseline}}\} - \text{Score}_{\text{random}}}$ to compare OEB3 with each baseline.

Table 1: Summary of human-normalized scores in 49 Atari games. BEBU, BEBU-UCB, BEBU-IDS and OEB3 are trained for 20M frames with RTX-2080Ti GPU for 5 random seeds.

Frames	200M				20M			
	DQN	BootDQN	NoisyNet	**BootDQN-IDS**	BEBU	BEBU-UCB	**BEBU-IDS**	**OEB3(ours)**
Mean	241%	553%	651%	**757%**	553%	610%	**622%**	**765%**
Medium	93%	139%	172%	**187%**	36%	38%	**44%**	**50%**

Table 2: Comparison of scores in *Montezuma's Revenge.*

Frames	200M				20M			
	DQN	BootDQN	NoisyNet	BootDQN-IDS	BEBU	BEBU-UCB	BEBU-IDS	OEB3
Scores	0	100	3	0	0	0	0	0

We report the performance of DQN (Mnih et al., 2015), NoisyNet (Fortunato et al., 2018), Bootstrapped DQN (Osband et al., 2016), and IDS (Nikolov et al., 2019) in 200M training frames. We choose NoisyNet as a baseline because Taiga et al. (2020) show NoisyNet performs substantially better than bonus-based methods (including CTS-counts (Bellemare et al., 2016), PixelCNN-counts (Ostrovski et al., 2017), RND (Burda et al., 2019b), and ICM (Pathak et al., 2017)) in the whole Atari suit (not in several hard exploration games). From Table 1, IDS performs better than NoisyNet thus IDS can be considered as a superior algorithm compared to bonus-based exploration. We reimplement IDS with BEBU-based update and the proposed OEB3 outperforms BEBU-IDS in mean and medium scores in the Atari suit. OEB3 achieves better performance than BEBU-IDS thus outperforms bonus-based methods. The results in Appendix G show OEB3 outperforms BEBU, BEBU-UCB, and BEBU-IDS in 36, 34, and 35 games out of all 49 games, respectively.

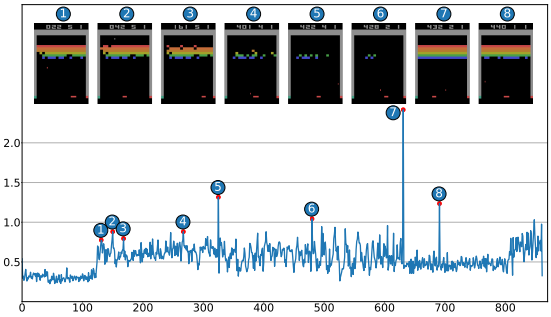


Figure 2: Visualization of UCB-bonus in Breakout

Our method does not have a good performance on Montezuma's Revenge (see Table 2) because the epistemic uncertainty-based methods are not particularly tweaked for this domain. Bootstrapped DQN, IDS, NoisyNet and BEBU-based methods all fail in this domain and achieves no more than 100 points. In contrast, the bonus-based methods achieve significantly higher scores in this tasks (e.g., RND achieves 8152 points). However, NoisyNet and IDS significantly outperforms all bonus-based methods evaluated by the whole Atari suit (Taiga et al., 2020).

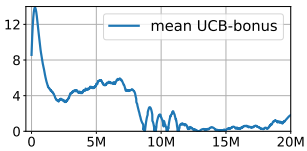


Figure 3: The change of mean UCB-bonus in Breakout

We use a trained OEB3 agent to interact with the environment for an episode in Breakout. The curve in Figure 2 shows the UCB-bonuses of the subsampled steps in the episode. We choose 8 spikes with high UCB-bonuses and visualize the corresponding frames. The events in spikes signify novel areas or meaningful experiences that are important for the agent to get rewards: digging a tunnel (1), throwing the ball to the top of bricks (2,3), rebounding the ball (4,5,6), eliminating all bricks and starting a new round (7,8). We provide more examples of visualization in Appendix E. We further record the change of mean UCB-bonus of the training batch in the learning process, as shown in Figure 3. The UCB-bonus is low at the beginning because the networks are randomly initialized. When the agent starts to explore the environment, the mean UCB-bonus rises rapidly to incentive exploration. As more experiences of states and actions are gathered, the mean UCB-bonus is reduced gradually, which indicates the bootstrapped value functions concentrate around the optimal value. But the UCB-bonuses are still relatively high with poorly-explored areas according to Figure 2, which makes the agent focus on exploring the meaningful parts of the environment.

6 Conclusion

The OEB3 with theoretical underpinnings from optimistic LSVI is proposed for DRL. The UCB-bonus captures the epistemic uncertainty of the action-value function. The optimistic value function augmented by UCB-bonus produces optimistic estimates for novel states and actions to incentive exploration. An efficient learning algorithm with the episodic backward update and bootstrapped Q-learning is derived. The empirically results in Mnist maze and Atari games show OEB3 outperforms several strong baselines with less computational cost. The visualization signifies high UCB-bonuses correspond to informative experiences for exploration. The future directions include adapting OEB3 to continuous control with policy gradient and integrating OEB3 with more expressive bonus schemes.

REFERENCES

Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, pp. 2312–2320, 2011.

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. In *Advances in Neural Information Processing Systems*, pp. 5048–5058, 2017.

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Peter Auer and Ronald Ortner. Logarithmic online regret bounds for undiscounted reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 49–56, 2007.

Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. In *International Conference on Machine Learning*, pp. 263–272, 2017.

Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. In *International Conference on Machine Learning*, 2020a.

Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martin Arjovsky, Alexander Pritzel, Andrew Bolt, and Charles Blundell. Never give up: Learning directed exploration strategies. In *International Conference on Learning Representations*, 2020b.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pp. 1471–1479, 2016.

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

Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. In *International Conference on Learning Representations*, 2019a.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019b.

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

Jongwook Choi, Yijie Guo, Marcin Moczulski, Junhyuk Oh, Neal Wu, Mohammad Norouzi, and Honglak Lee. Contingency-aware exploration in reinforcement learning. In *International Conference on Learning Representations*, 2019.

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

Christoph Dann and Emma Brunskill. Sample complexity of episodic fixed-horizon reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2818–2826, 2015.

Meng Fang, Tianyi Zhou, Yali Du, Lei Han, and Zhengyou Zhang. Curriculum-guided hindsight experience replay. In *Advances in Neural Information Processing Systems*, pp. 12602–12613, 2019.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, R'emi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy networks for exploration. In *International Conference on Learning Representations*, 2018.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pp. 1587–1596, 2018.

Nick Haber, Damian Mrowca, Stephanie Wang, Li F Fei-Fei, and Daniel L Yamins. Learning to play with intrinsically-motivated, self-aware agents. In *Advances in Neural Information Processing Systems*, pp. 8388–8399, 2018.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pp. 1109–1117, 2016.

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

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

Chi Jin, Akshay Krishnamurthy, Max Simchowitz, and Tiancheng Yu. Reward-free exploration for reinforcement learning. In *International Conference on Machine Learning*, 2020a.

Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. In *Proceedings of Thirty Third Conference on Learning Theory*, pp. 2137–2143, 2020b.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning for atari. In *International Conference on Learning Representations*, 2020.

Hyoungseok Kim, Jaekyeom Kim, Yeonwoo Jeong, Sergey Levine, and Hyun Oh Song. Emi: Exploration with mutual information. In *International Conference on Machine Learning*, pp. 3360–3369, 2019a.

Kun Ho Kim, Megumi Sano, Julian De Freitas, Nick Haber, and Daniel Yamins. Active world model learning in agent-rich environments with progress curiosity. In *International Conference on Machine Learning*, 2020.

Youngjin Kim, Wontae Nam, Hyunwoo Kim, Ji-Hoon Kim, and Gunhee Kim. Curiosity-bottleneck: Exploration by distilling task-specific novelty. In *International Conference on Machine Learning*, pp. 3379–3388, 2019b.

Su Young Lee, Choi Sungik, and Sae-Young Chung. Sample-efficient deep reinforcement learning via episodic backward update. In *Advances in Neural Information Processing Systems*, pp. 2110–2119, 2019.

Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

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

Nikolay Nikolov, Johannes Kirschner, Felix Berkenkamp, and Andreas Krause. Information-directed exploration for deep reinforcement learning. In *International Conference on Learning Representations*, 2019.

Ian Osband and Benjamin Van Roy. On lower bounds for regret in reinforcement learning. *arXiv preprint arXiv:1608.02732*, 2016.

Ian Osband and Benjamin Van Roy. Why is posterior sampling better than optimism for reinforcement learning? In *International Conference on Machine Learning*, pp. 2701–2710, 2017.

Ian Osband, Daniel Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*, pp. 3003–3011, 2013.

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

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

Ian Osband, Benjamin Van Roy, Daniel J Russo, and Zheng Wen. Deep exploration via randomized value functions. *Journal of Machine Learning Research*, 20(124):1–62, 2019.

Georg Ostrovski, Marc G Bellemare, Aäron van den Oord, and Rémi Munos. Count-based exploration with neural density models. In *International Conference on Machine Learning*, pp. 2721–2730, 2017.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, pp. 2778–2787, 2017.

Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *International Conference on Machine Learning*, pp. 5062–5071, 2019.

Roberta Raileanu and Tim Rocktäschel. Ride: Rewarding impact-driven exploration for procedurally-generated environments. In *International Conference on Learning Representations*, 2020.

Tabish Rashid, Bei Peng, Wendelin Boehmer, and Shimon Whiteson. Optimistic exploration even with a pessimistic initialisation. In *International Conference on Learning Representations*, 2020.

Neale Ratzlaff, Qinxun Bai, Li Fuxin, and Wei Xu. Implicit generative modeling for efficient exploration. In *International Conference on Machine Learning*, 2020.

Daniel Russo. Worst-case regret bounds for exploration via randomized value functions. In *Advances in Neural Information Processing Systems*, pp. 14410–14420, 2019.

Nikolay Savinov, Anton Raichuk, Raphaël Marinier, Damien Vincent, Marc Pollefeys, Timothy Lillicrap, and Sylvain Gelly. Episodic curiosity through reachability. In *International Conference on Learning Representations*, 2019.

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

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

Adrien Ali Taiga, William Fedus, Marlos C Machado, Aaron Courville, and Marc G Bellemare. On bonus based exploration methods in the arcade learning environment. In *International Conference on Learning Representations*, 2020.

Mike West. Outlier models and prior distributions in bayesian linear regression. *Journal of the Royal Statistical Society: Series B (Methodological)*, 46(3):431–439, 1984.

Andrea Zanette, David Brandfonbrener, Matteo Pirotta, and Alessandro Lazaric. Frequentist regret bounds for randomized least-squares value iteration. *arXiv preprint arXiv:1911.00567*, 2019.

A PROOF OF THEOREM 1

Recall that we consider the following regularized least-square problem,

$$w_t \leftarrow \arg\min_{w\in\mathbb{R}^d} \sum_{\tau=0}^{m} \big[r_t(s_t^\tau, a_t^\tau) + \gamma \cdot \max_{a\in\mathcal{A}} Q_{t+1}(s_{t+1}^\tau, a) - w^\top \phi(s_t^\tau, a_t^\tau)\big]^2 + \lambda \|w\|^2. \tag{8}$$

In the sequel, we consider a Bayesian linear regression perspective of (8) that captures the intuition behind the UCB-Bonus in OEB3. Our objective is to approximate the action-value function Q_t via fitting the parameter w, such that

$$w^\top \phi(s_t, a_t) \approx r_t(s_t, a_t) + \gamma \cdot \max_{a\in\mathcal{A}} Q_{t+1}(s_{t+1}, a),$$

where Q_{t+1} is given. We assume that we are given a Gaussian prior of the initial parameter $w \sim \mathcal{N}(0, \mathbf{I}/\lambda)$. With a slight abuse of notation, we denote by w_t the Bayesian posterior of the parameter w given the set of independent observations $\mathcal{D}_m = \{(s_t^\tau, a_t^\tau, s_{t+1}^\tau)\}_{\tau\in[m]}$. We further define the following noise with respect to the least-square problem in (8),

$$\epsilon = r_t(s_t, a_t) + \gamma \cdot \max_{a\in\mathcal{A}} Q_{t+1}(s_{t+1}, a) - w^\top \phi(s_t, a_t), \tag{9}$$

where (s_t, a_t, s_{t+1}) follows the distribution of trajectory. The following Theorem justifies the UCB-Bonus in OEB3 under the Bayesian linear regression perspective.

Theorem 2. *We assume that ϵ follows the standard Gaussian distribution $\mathcal{N}(0,1)$ given the state-action pair (s_t, a_t) and the parameter w. Let w follows the Gaussian prior $\mathcal{N}(0, \mathbf{I}/\lambda)$. We define*

$$\Lambda_t = \sum_{\tau=0}^{m} \phi(x_t^\tau, a_t^\tau)\phi(x_t^\tau, a_t^\tau)^\top + \lambda \cdot \mathbf{I}. \tag{10}$$

It then holds for the posterior of w_t given the set of independent observations $\mathcal{D}_m = \{(s_t^\tau, a_t^\tau, s_{t+1}^\tau)\}_{\tau\in[m]}$ that

$$\mathrm{Var}\big(\phi(s_t, a_t)^\top w_t\big) = \mathrm{Var}\big(\tilde{Q}_t(s_t, a_t)\big) = \sqrt{\phi(s_t, a_t)^\top \Lambda_t^{-1} \phi(s_t, a_t)}, \quad \forall (s_t, a_t) \in \mathcal{S}\times\mathcal{A}.$$

Here we denote by $\tilde{Q}_t = w_t^\top \phi$ the point estimation of action-value function obtained by solving (8).

Proof. The proof follows the standard analysis of Bayesian linear regression. See, e.g., West (1984) for a detailed analysis. We denote the target of the linear regression in (8) by

$$y_t = r_t(s_t, a_t) + \gamma \cdot \max_{a\in\mathcal{A}} Q_{t+1}(s_{t+1}, a).$$

By the assumption that ϵ follows the standard Gaussian distribution, we obtain that

$$y_t \,|\, (s_t, a_t), w \sim \mathcal{N}\big(w^\top \phi(s_t, a_t), 1\big). \tag{11}$$

Recall that we have the prior distribution $w \sim \mathcal{N}(0,1)$. Our objective is to compute the posterior density $w_t = w \,|\, \mathcal{D}_m$, where $\mathcal{D}_m = \{(s_t^\tau, a_t^\tau, s_{t+1}^\tau)\}_{\tau\in[m]}$ is the set of observations. It holds from Bayes rule that

$$\log p(w \,|\, \mathcal{D}_m) = \log p(w) + \log p(\mathcal{D}_m \,|\, w) + Const., \tag{12}$$

where $p(\cdot)$ denote the probability density function of the respective distributions. Plugging (11) and the probability density function of Gaussian distribution into (12) yields

$$\begin{aligned}\log p(w \,|\, \mathcal{D}_m) &= -\|w\|^2/2 - \sum_{\tau=1}^{m} \|w^\top \phi(s_t^\tau, a_t^\tau) - y_t^\tau\|^2/2 + Const. \\ &= -(w-\mu_t)^\top \Lambda_t^{-1}(w-\mu_t)/2 + Const.,\end{aligned} \tag{13}$$

where we define

$$\mu_t = \Lambda_t^{-1} \sum_{\tau=1}^{m} \phi(s_t^\tau, a_t^\tau) y_t^\tau, \qquad \Lambda_t = \sum_{\tau=0}^{m} \phi(x_t^\tau, a_t^\tau)\phi(x_t^\tau, a_t^\tau)^\top + \lambda \cdot \mathbf{I}.$$

Thus, by (13), we obtain that $w_t = w \,|\, \mathcal{D}_m \sim \mathcal{N}(\mu_t, \Lambda_t^{-1})$. It then holds for all $(s_t, a_t) \in \mathcal{S}\times\mathcal{A}$ that

$$\mathrm{Var}\big(\phi(s_t, a_t)^\top w_t\big) = \mathrm{Var}\big(\tilde{Q}_t(s_t, a_t)\big) = \sqrt{\phi(s_t, a_t)^\top \Lambda_t^{-1} \phi(s_t, a_t)},$$

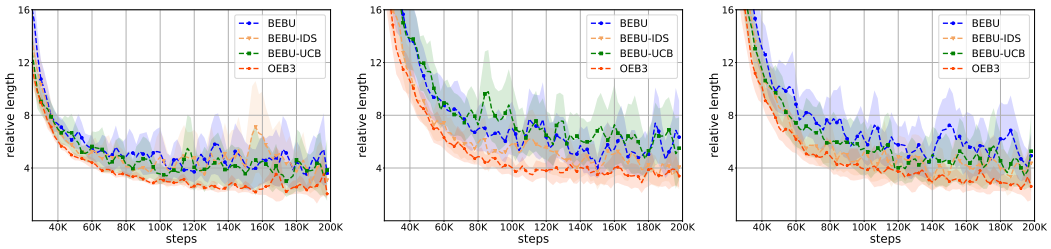
which concludes the proof of Theorem 1. □

B ALGORITHMIC DESCRIPTION

Algorithm 2 OEB3 in DRL

Initialize: replay buffer $\mathcal{D}$, bootstrapped Q-network $Q(\cdot;\theta)$ and target network $Q(\cdot;\theta^-)$
Initialize: total training frames $H = 20\text{M}$, current frame $h = 0$
while $h < H$ **do**
  Pick a bootstrapped Q-function to act by sampling $k \sim \text{Unif}\{1, \ldots, \text{K}\}$
  Reset the environment and receive the initial state s_0
  for step $i = 0$ **to** Terminal **do**
    With ϵ-greedy choose $a_i = \arg\max_a Q^k(s_i, a)$
    Take action and observe r_i and s_{i+1}, then save the transition in buffer $\mathcal{D}$
    if $h \,\%\, \text{training frequency} = 0$ **then**
      Sample an episodic experience $E = \{\mathbf{S}, \mathbf{A}, \mathbf{R}, \mathbf{S}'\}$ with length T from $\mathcal{D}$
      Initialize a Q-table $\tilde{\mathbf{Q}} = Q(\mathbf{S}', \mathcal{A}; \theta^-) \in \mathbb{R}^{K \times |\mathcal{A}| \times T}$ by the target Q-network
      Compute the UCB-bonus for immediate reward for all steps to construct $\mathbf{B} \in \mathbb{R}^T$
      Compute the action matrix $\tilde{\mathbf{A}} = \arg\max_a \tilde{\mathbf{Q}}[\cdot, a, \cdot] \in \mathbb{R}^{K \times T}$ to gather all a' of next-Q
      Compute the UCB-bonus for next-Q for all heads and all steps to construct $\tilde{\mathbf{B}} \in \mathbb{R}^{K \times T}$
      Compute the mask matrix $\mathbf{M} \in \mathbb{R}^{K \times T}$ where $\mathbf{M}[k, t] = \mathbb{1}_{\tilde{\mathbf{A}}[k,t] \neq \mathbf{A}_{t+1}}$
      Initialize target table $\mathbf{y} \in \mathbb{R}^{K \times T}$ to zeros, and set $\mathbf{y}[\cdot, T-1] = \mathbf{R}_{T-1} + \alpha_1 \mathbf{B}_{T-1}$
      for $t = T - 2$ **to** 0 **do**
        $\tilde{\mathbf{Q}}[\cdot, a_{t+1}, t] \leftarrow \beta \mathbf{y}[\cdot, t+1] + (1-\beta)\tilde{\mathbf{Q}}[\cdot, a_{t+1}, t]$
        $\mathbf{y}[\cdot, t] \leftarrow \big(\mathbf{R}_t + \alpha_1 \mathbf{B}_t\big) + \gamma\big(\tilde{\mathbf{Q}}[\cdot, a', t] + \alpha_2 \mathbf{M}[\cdot, t] \circ \tilde{\mathbf{B}}[\cdot, t]\big)$ where $a' = \tilde{\mathbf{A}}[\cdot, t]$
      end for
      Compute the Q-value of $(\mathbf{S}, \mathbf{A})$ for all heads as $\mathbf{Q} = Q(\mathbf{S}, \mathbf{A}; \theta) \in \mathbb{R}^{K \times T}$
      Perform a gradient descent step on $(\mathbf{y} - \mathbf{Q})^2$ with respect to θ
    end if
    Every C steps reset $\theta^- \leftarrow \theta$
    $h \leftarrow h + 1$
  end for
end while

C ADDITIONAL EXPERIMENT: MNIST MAZE



(a) Wall density of 30% (b) Wall density of 40% (c) Wall density of 50%

Figure 4: Results of 200K steps training of Mnist maze with different wall-density setup.

We use 10×10 Mnist maze with randomly placed walls to evaluate our method. The agent starts from the initial position $(0, 0)$ in the upper-left of the maze and aims to reach the goal position $(9, 9)$ in the bottom-right. The state of position (i, j) is represented by staking two randomly sampled images with label i and j from the Mnist dataset. When the agent steps to a new position, the state representation is reconstructed by sampling images. Hence the agent gets different states even stepping to the same location twice, which minimizes the correlation among locations. We further add some stochasticity in the transition probability. The agent has a probability of 10% to arrive in the adjacent locations when taking an action. For example, when taking action 'left', the agent has a 10% chance of transiting to 'up', and a 10% chance of transiting to 'down'. The agent gets a reward of -1 when bumping into a wall, and gets 1000 when reaching the goal.

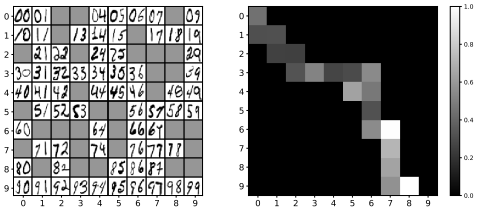


Figure 5: An example Mnist maze (left) and the UCB-bonuses in the agent's path (right).

We use the different setup of wall-density in the experiment. The wall-density means the proportion of walls in all locations. Figure 5 (left) shows a generated maze with wall-density of 50%. The gray positions represent walls. We train all methods with wall-density of 30%, 40%, and 50%. For each setup, we train 50 independent agents for 50 randomly generated mazes. The relative length defined by $l_{\text{agent}}/l_{\text{best}}$ is used to evaluate the performance, where l_{agent} is the length of the agent's path in an episode (maximum steps are 1000), and l_{best} is the length of the shortest path. The performance comparison is shown in Figure 4. OEB3 performs best in all methods, and BEBU-IDS is a strong baseline. We use a trained OEB3 agent to take action in the maze of Figure 5 (left). The UCB-bonuses of state-action pairs in the agent's path are computed and visualized in Figure 5 (right). The state-action pairs with high UCB-bonus show the bottleneck positions in the path. For example, the state-action pairs in location $(3, 3)$ and $(6, 7)$ produce high UCB-bonus to guide the agent to choose the right direction. The UCB-bonus encourages the agent to walk through these bottleneck positions correctly. We give more examples of visualization in Appendix E.

D IMPLEMENTATION DETAIL

D.1 MNIST MAZE

Hyper-parameters of BEBU. BEBU is the basic algorithm of BEBU-UCB and BEBU-IDS. BEBU uses the same network-architecture as Bootstrapped DQN (Osband et al., 2016). The diffusion factor and other training parameters are set by following EBU paper (Lee et al., 2019). Details are summarized in Table 3.

Table 3: Hyper-parameters of BEBU for Mnist-Maze

Hyperparameters	Value	Description
state space	$28 \times 28 \times 2$	Stacking two images sampled from Mnist dataset with labels according to the agent's current location.
action space	4	Including left, right, up, and down.
K	10	Number of bootstrapped heads.
network-architecture	conv(64,4,4) conv(64,3,1) dense$\{512, 4\}_{k=1}^{K}$	Using convolution (channels, kernel size, stride) layers first, then fully connected into K bootstrapped heads. Each head has 512 ReLUs and 4 linear units.
gradient norm	10	The gradient is clipped by 10. The gradient of each head is normalize by $1/K$ according to bootstrapped DQN.
learning starts	10000	The agent takes actions according to the initial policy before learning starts.
replay buffer size	170	A simple replay buffer is used to store episodic experience.
training frequency	50	Number of action-selection step between successive gradient descent steps.
H	200,000	Total timesteps to train a single maze.
target network update frequency	2000	The target-network is updated every 2000 steps.
optimizer	Adam	Adam optimizer is used for training. Detailed parameters: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon_{\text{ADAM}} = 10^{-7}$.
learning rate	0.001	Learning rate for Adam optimizer.
ϵ	$\frac{(h-H)^2}{H^2}$	Exploration factor. H is the total timesteps for training, and h is the current timestep. ϵ starts from 1 and is annealed to 0 in a quadratic manner.
γ	0.9	Discount factor.
β	1.0	Diffusion factor of backward update.
wall density	30%, 40%, and 50%	Proportion of walls in all locations of the maze.
reward	-1 or 1000	Reward is -1 when bumping into a wall, and 1000 when reaching the goal.
stochasticity	10%	Has a probability of 10% to arrive in the adjacent locations when taking an action.
evaluation metric	$l_{\text{rel}} = l_{\text{agent}}/l_{\text{best}}$	Ratio between length of the agent's path and the best length.

Hyper-parameters of BEBU-UCB. BEBU-UCB uses the upper-bound of Q-values to select actions. In particular, $a = \arg\max_{a \in \mathcal{A}} [\mu(s,a) + \lambda_{\text{ucb}}\sigma(s,a)]$, where $\mu(s,a)$ and $\sigma(s,a)$ are the mean and standard deviation of bootstrapped Q-values $\{Q^k(s,a)\}_{k=1}^{K}$. We use $\lambda_{\text{ucb}} = 0.1$ in our experiment.

Hyper-parameters of BEBU-IDS. The action-selection in IDS (Nikolov et al., 2019) follows the regret-information ratio as $a_t = \arg\min_{a \in \mathcal{A}} \frac{\hat{\Delta}_t(s,a)^2}{I_t(s,a)}$, which balances the regret and exploration. $\hat{\Delta}_t(s,a)$ is the expected regret that indicates the loss of reward when choosing a suboptimal action a. IDS uses a conservative estimate of regret as $\hat{\Delta}_t(s,a) = \max_{a' \in \mathcal{A}} u_t(s,a') - l_t(s,a)$, where $[l_t(s,a), u_t(s,a)]$ is the confidence interval of action-value function. In particular, $u_t(s,a) = \mu(s,a) + \lambda_{\text{ids}}\sigma(s,a)$ and $l_t(s,a) = \mu(s,a) - \lambda_{\text{ids}}\sigma(s,a)$, where $\mu(s,a)$ and $\sigma(s,a)$ are the mean and standard deviation of bootstrapped Q-values $\{Q^k(s,a)\}_{k=1}^{K}$. The information gain $I_t(a)$ measures the uncertainty of action-values as $I(s,a) = \log(1 + \frac{\sigma(s,a)^2}{\rho(s,a)^2}) + \epsilon_{\text{ids}}$, where $\rho(s,a)$ is the variance of the return distribution. $\rho(s,a)$ is measured by C51 (Bellemare et al., 2017) in distributional RL and becomes a constant in ordinary Q-learning. We use $\lambda_{\text{ids}} = 0.1$, $\rho(s,a) = 1.0$, and $\epsilon_{\text{ids}} = 10^{-5}$ in our experiment.

Hyper-parameters of OEB3. We set α_1 and α_2 to the same value of 0.01. We find that adding a normalizer to UCB-bonus $\tilde{\mathbf{B}}$ of the next Q-value enables us to get more stable performance. $\tilde{\mathbf{B}}$ is smoothed by dividing a running estimate of its standard deviation. Because the UCB-bonuses for next-Q are different for each Q-head, the normalization is useful in most cases by making the value function have a smooth and stable update.

D.2 ATARI GAMES

Hyper-parameters of BEBU. The basic setting of the Atari environment is the same as Nature DQN (Mnih et al., 2015) and EBU (Lee et al., 2019) papers. We use the different network architecture, environmental setup, exploration factor, and evaluation scheme from Mnist maze experiment. Details are summarized in Table 4.

Table 4: Hyper-parameters of BEBU for Atari games

Hyperparameters	Value	Description
state space	$84 \times 84 \times 4$	Stacking 4 recent frames as the input to network.
action repeat	4	Repeating each action 4 times.
K	10	The number of bootstrapped heads.
network-architecture	conv(32,8,4) conv(64,4,2) conv(64,3,1) dense$\{512, \lvert\mathcal{A}\rvert\}_{k=1}^{K}$	Using convolution(channels, kernel size, stride) layers first, then fully connected into K bootstrapped heads. Each head has 512 ReLUs and $\lvert\mathcal{A}\rvert$ linear units.
gradient norm	10	The gradient is clipped by 10, and also be normalize by $1/K$ for each head by following bootstrapped DQN.
learning starts	50000	The agent takes random actions before learning starts.
replay buffer size	1M	The number of most recent states stored in the replay buffer.
training frequency	4	The number of action-selection step between successive gradient descent steps.
H	20M	Total frames to train an environment.
target network update frequency	10000	The target-network is updated every 10000 steps.
optimizer	Adam	Adam optimizer is used for training. Detailed parameters: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon_{\mathrm{ADAM}} = 10^{-7}$.
mini-batch size	32	The number of training cases for gradient decent each time.
learning rate	0.00025	Learning rate for Adam optimizer.
initial exploration	1.0	Initial value of ϵ in ϵ-greedy exploration.
final exploration	0.1	Final value of ϵ in ϵ-greedy exploration.
final exploration frames	1M	The number of frames that the initial value of ϵ linearly annealed to the final value.
γ	0.99	Discount factor.
β	0.5	Diffusion factor of backward update.
ϵ_{eval}	0.05	Exploration factor in ϵ-greedy for evaluation.
evaluation policy	ensemble vote	The same evaluation method as in Bootstrapped DQN Osband et al. (2016).
evaluation length	108000	The policy is evaluated for 108000 steps.
evaluation frequency	100K	The policy is evaluated every 100K steps.
max no-ops	30	Maximum number no-op actions before an episode starts.

Hyper-parameters of BEBU-UCB. BEBU-UCB selects actions by $a = \arg\max_{a\in\mathcal{A}} [\mu(s,a) + \lambda_{\mathrm{ucb}}\sigma(s,a)]$. The detail is given in Appendix D.1. We use $\lambda_{\mathrm{ucb}} = 0.1$ in our experiment by searching coarsely.

Hyper-parameters of BEBU-IDS. The action-selection follows the regret-information ratio as $a_t = \arg\min_{a\in\mathcal{A}} \frac{\hat{\Delta}_t(s,a)^2}{I_t(s,a)}$. The details are given in Appendix D.1. We use $\lambda_{\mathrm{ids}} = 0.1$, $\rho(s,a) = 1.0$ and $\epsilon_{\mathrm{ids}} = 10^{-5}$ in our experiment by searching coarsely.

Hyper-parameters of OEB3. We set α_1 and α_2 to the same value of 0.5×10^{-4}. The UCB-bonus $\tilde{\mathbf{B}}$ for the next Q-value is normalized by dividing a running estimate of its standard deviation to have a stable performance.

E VISUALIZING OEB3

OEB3 uses the UCB-bonus that indicates the disagreement of bootstrapped Q-estimates to measure the uncertainty of Q-functions. The state-action pairs with high UCB-bonuses signify the bottleneck positions or meaningful events. We provide visualization in several tasks to illustrate the effect of UCB-bonuses. Specifically, we choose *Mnist-maze* and two popular Atari games *Breakout* and *Mspacman* to analyze.

E.1 MNIST-MAZE

Figure 6 illustrates the UCB-bonus in four randomly generated mazes. The mazes in Figure 6(a) and 6(b) have a wall-density of 40%, and in Figure 6(c) and 6(d) have a wall-density of 50%. The left of each figure shows the map of maze, where the black blocks represent the walls. We do not show the Mnist representation of states for simplification, and Mnist states are used in training. A trained OEB3 agent starts at the upper-left, then takes actions to achieve the goal at bottom-right. The UCB-bonuses of state-action pairs in the agent's path are computed and shown on the right of each figure. The value is normalized to $0 \sim 1$ for visualization. We show the maximal value if the agent appears several times in the same location.

The positions with UCB-bonuses that higher than 0 draw the path of the agent. The path is usually winding and includes positions beyond the shortest path because the state transition has stochasticity. The state-action pairs with high UCB-bonuses show the bottleneck positions in the path. In maze 6(a), the agent slips from the right path in position $(4, 7)$ to $(4, 9)$. The state-action in position $(4, 8)$ produces high bonus to guide the agent back to the right path. In maze 6(b), the bottleneck state in $(3, 2)$ has high bonus to avoid the agent from entering into the wrong side of the fork. The other two mazes also have bottleneck positions, like $(3, 3)$ in maze 6(c) and $(7, 6)$ in maze 6(d). Selecting actions incorrectly in these important locations is prone to failure. The UCB-bonus encourages the agent to walk through these bottleneck positions correctly.

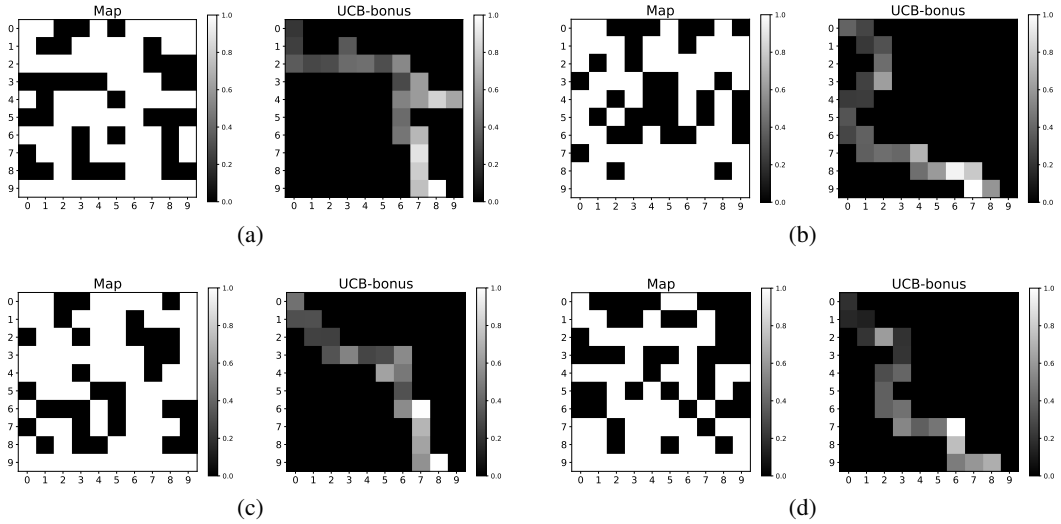


Figure 6: Visualization of UCB-bonus in Mnist-maze

E.2 BREAKOUT

In Breakout, the agent uses walls and the paddle to rebound the ball against the bricks and eliminate them. We use a trained OEB3 agent to interact with the environment for an episode in Breakout. The whole episode contains 3445 steps, and we subsample them every 4 steps for visualization. The curve in Figure 7 shows the UCB-bonus in 861 sampled steps. We choose 16 spikes with high UCB-bonuses and visualize the corresponding frames. The events in spikes usually mean meaningful experiences that are important for the agent to get rewards. In step 1, the agent is hoping to dig a tunnel and get

rewards faster. After digging a tunnel, the balls appear on the top of bricks in steps 2, 3, 4, 5, 6, 9, and 12. Balls on the top are easier to hit bricks. The agents rebound the ball and throw it over the bricks in steps 7, 8, 10, and 11. In state 13, the agent eliminates all bricks and then comes to a new round, which is novel and promising to get more rewards. The agents in steps 14, 15, and 16 rebound the ball and try to dig a tunnel again. The UCB-bonus encourages the agent to explore the potentially informative and novel state-action pairs to get high rewards. We record 15 frames after each spike for further visualization. The video is available at `https://youtu.be/VptBkHyMt8g`.

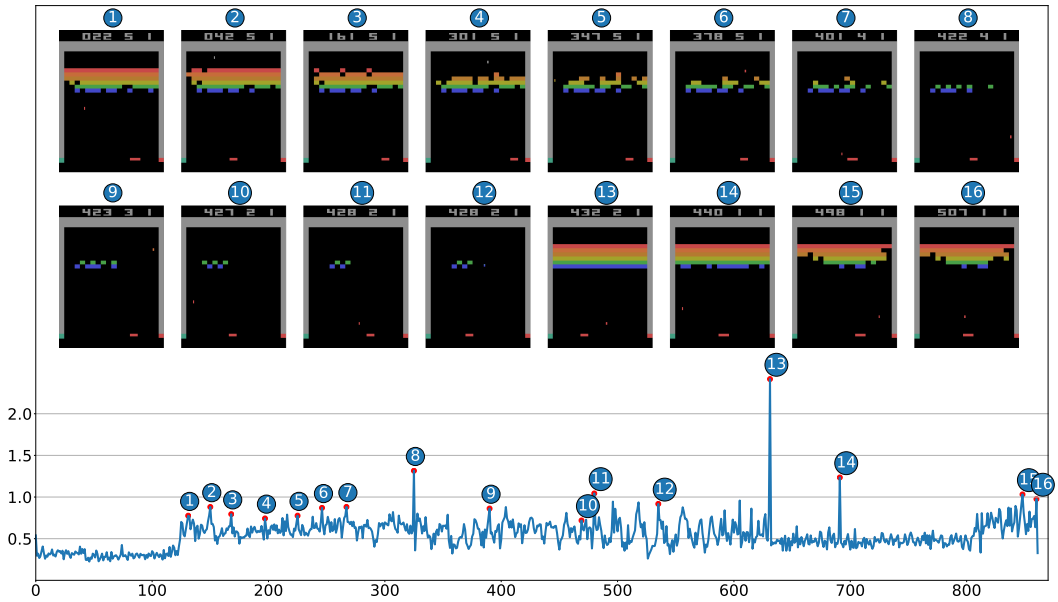


Figure 7: Visualization of UCB-bonus in Breakout

E.3 MsPacman

In MsPacman, the agent earns points by avoiding monsters and eating pellets. Eating an energizer causes the monsters to turn blue, allowing them to be eaten for extra points. We use a trained OEB3 agent to interact with the environment for an episode. Figure 8 shows the UCB bonus in all 708 steps. We choose 16 spikes to visualize the frames. The spikes of exploration bonuses correspond to meaningful events for the agent to get rewards: starting a new scenario (1,2,9,10), changing direction (3,4,13,14,16), eating energizer (5,11), eating monsters (7,8,12), and entering the corner (6,15). These state-action pairs with high UCB-bonuses make the agent explore the environment efficiently. We record 15 frames after each spike, and the video is shown at `https://youtu.be/C_8NHKpBNXM`.

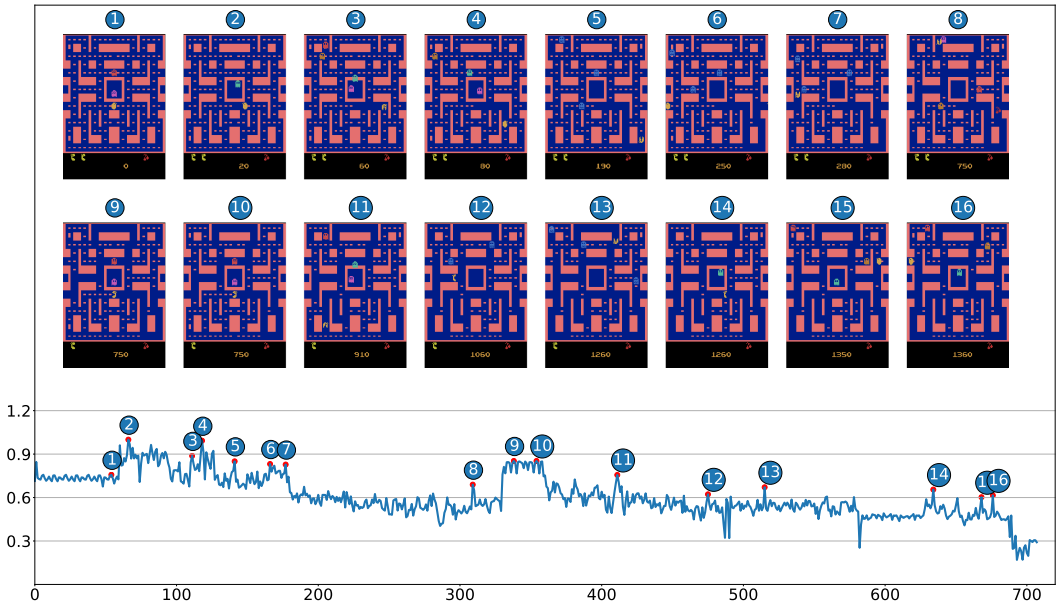


Figure 8: Visualization of UCB-bonus in MsPacman

F RAW SCORES OF ALL 49 ATARI GAMES

Table 5: Raw scores for Atari games. Each game is trained for 20M frames with a single RTX-2080Ti GPU. Bold scores signify the best score out of all methods.

	Random	Human	BEBU	BEBU-UCB	BEBU-IDS	OEB3
Alien	227.8	6,875.0	**1,118.0**	811.1	857.9	916.9
Amidar	5.8	1676.0	81.7	**166.4**	148.1	94.0
Assault	222.4	1,496.0	1,377.0	**3,574.5**	2,441.8	2,996.2
Asterix	210.0	8,503.0	2,315.0	2,709.3	2,433.9	**2,719.0**
Asteroids	719.1	13,157.0	962.8	**1,025.0**	868.8	959.9
Atlantis	12,850.0	29,028.0	3,020,500.0	3,191,600.0	3,144,440.0	**3,146,300.0**
Bank Heist	14.2	734.4	331.8	277.0	361.6	**378.6**
Battle Zone	2,360.0	37,800.0	5,446.4	**16,348.8**	10,520.0	13,454.5
BeamRider	363.9	5,775.0	2,930.0	3,208.3	3,391.0	**3,736.7**
Bowling	23.1	154.8	29.9	30.7	**40.2**	30.0
Boxing	0.1	4.3	72.4	68.3	69.8	**75.1**
Breakout	1.7	31.8	**473.2**	382.3	412.7	423.1
Centipede	2,090.9	11,963.0	2,547.2	2,377.9	**3,328.4**	2,661.8
Chopper Command	811.0	9,882.0	930.6	1,013.4	1,100.0	**1,100.3**
Crazy Climber	10,780.5	35,411.0	49,735.7	39,187.5	42,242.9	**53,346.7**
Demon Attack	152.1	3,401.0	6,506.3	6,840.4	**7,080.0**	6,794.6
Double Dunk	-18.6	-15.5	-18.9	**-16.5**	-17.0	-18.2
Enduro	0.0	309.6	504.1	697.8	513.6	**719.0**
Fishing Derby	-91.7	5.5	-56.7	-83.8	**-53.3**	-60.1
Freeway	0.0	29.6	21.5	21.6	21.3	**32.1**
Frostbite	65.2	4,335.0	393.4	470.4	466.2	**1,277.3**
Gopher	257.6	2,321.0	4,842.6	**7,211.8**	7,171.5	6,359.5
Gravitar	173.0	2,672.0	256.1	321.0	283.3	**393.6**
H.E.R.O	1,027.0	25,763.0	2,951.4	2,905.0	3,059.4	**3,302.5**
Ice Hockey	-11.2	0.9	-5.4	-6.5	-4.6	**-4.2**
Jamesbond	29.0	406.7	**650.0**	360.3	302.1	434.3
Kangaroo	52.0	3,035.0	3624.2	2,711.1	**4,448.0**	2,387.0
Krull	1,598.0	2,395.0	15,716.7	11,499.0	10,818.0	**45,388.8**
Kung-Fu Master	258.5	22,736.0	56.0	20,738.9	**26,909.7**	16,272.2
Montezuma's Revenge	0.0	4,376.0	0.0	0.0	0.0	**0.0**
Ms. Pacman	307.3	15,693.0	1,723.8	1,706.8	1,615.5	**1,794.9**
Name This Game	2,292.3	4,076.0	8,275.3	6,573.9	**8,925.0**	8,576.8
Pong	-20.7	9.3	18.1	18.5	17.2	**18.7**
Private Eye	24.9	69,571.0	1,185.8	1,925.2	1,897.1	1,174.1
Q*Bert	163.9	13,455.0	3,588.4	3,783.2	3,696.0	**4,275.0**
River Raid	1,338.5	13,513.0	3,127.5	**3,617.7**	3,169.1	2,926.5
Road Runner	11.5	7,845.0	11,483.0	20,990.7	17,281.4	**21,831.4**
Robotank	2.2	11.9	10.3	13.3	10.7	**13.5**
Seaquest	68.4	20,182.0	447.0	**492.3**	332.4	332.1
Space Invaders	148.0	1,652.0	814.4	782.2	794.7	**904.9**
Star Gunner	664.0	10,250.0	1,467.2	1,201.5	1,158.9	**1,290.2**
Tennis	-23.8	-8.9	-1.0	-2.0	-1.0	**-1.0**
Time Pilot	3,568.0	5,925.0	2,622.1	3,321.2	1,950.6	**3,404.5**
Tutankham	11.4	167.6	167.0	151.0	80.5	**297.0**
Up and Down	533.4	9,082.0	**5,954.8**	4,530.2	4,619.7	5,100.8
Venture	0.0	1,188.0	42.9	3.4	**150.0**	16.1
Video Pinball	16,256.9	17,298.0	26,829.6	48,959.1	58,398.3	**80,607.0**
Wizard of Wor	563.5	4,757.0	810.8	**1,316.7**	578.2	480.7
Zaxxon	32.5	9,173.0	1,587.5	2,104.8	1,594.2	**2,842.0**

G PERFORMANCE COMPARISON

We use the relative scores as

$$\frac{\text{Score}_{\text{Agent}} - \text{Score}_{\text{Baseline}}}{\max\{\text{Score}_{\text{human}}, \text{Score}_{\text{baseline}}\} - \text{Score}_{\text{random}}}$$

to compare OEB3 with baselines. The results of OEB3 comparing with BEBU, BEBU-UCB, and BEBU-IDS is shown in Figure 9, Figure 10, and Figure 11, respectively.

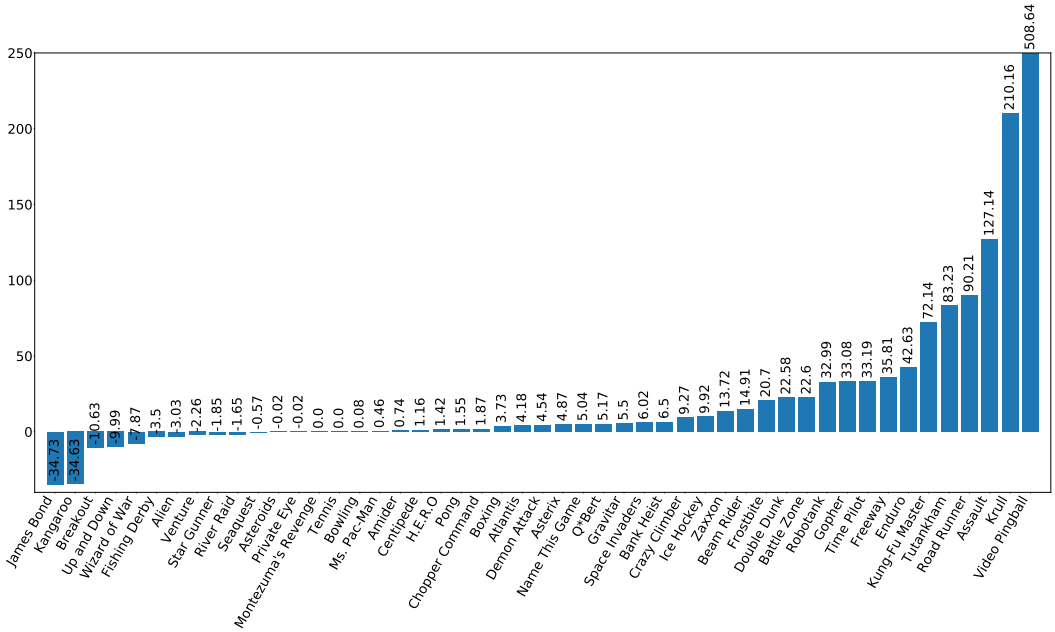


Figure 9: Relative score of OEB3 compared to BEBU in percents (%).

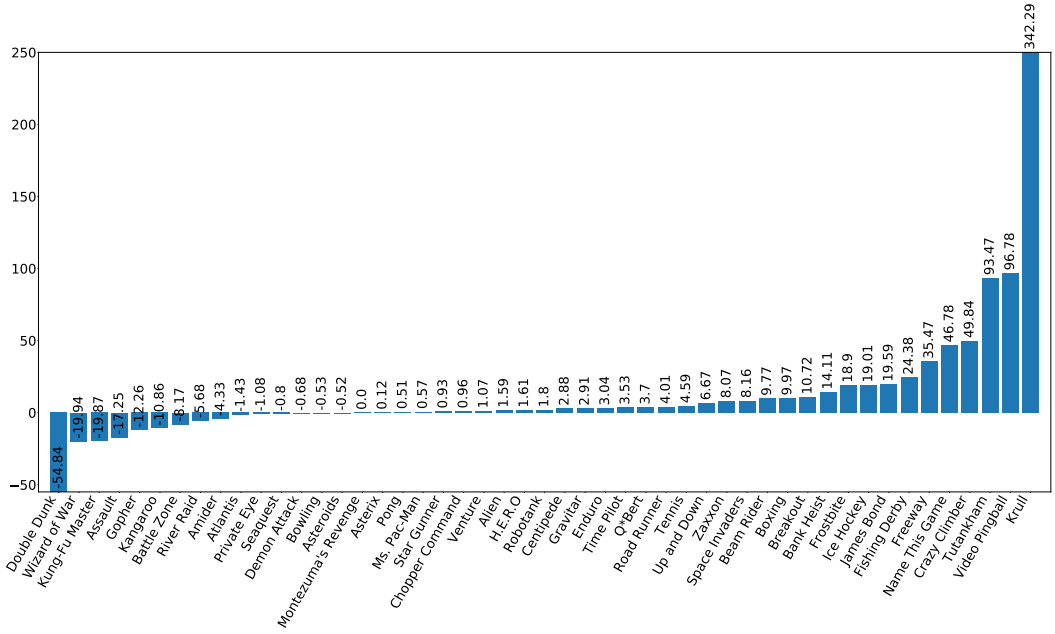


Figure 10: Relative score of OEB3 compared to BEBU-UCB in percents (%).

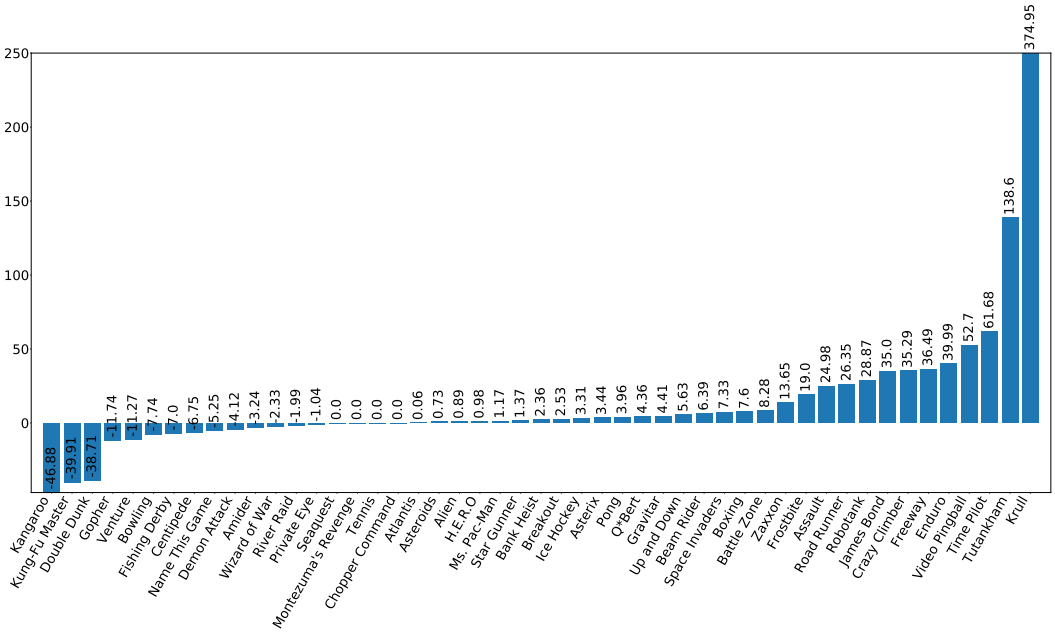


Figure 11: Relative score of OEB3 compared to BEBU-IDS in percents (%).