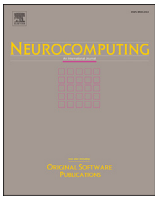
Neurocomputing 683 (2026) 131581

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Towards unbiased action value estimation in reinforcement learning

Yuan Xue [a,*], Daniel Kudenko [a], Megha Khosla [b]

[a] L3S Research Centre, Appelstraße 9A, Hanover, 30167, Lower Saxony, Germany
[b] Multimedia Computing Group in the Intelligent Systems Department Delft University of Technology, Van Mourik Broekmanweg 6, Delft, 2628 XE, the Netherlands

HIGHLIGHTS

- QV-learning with experience replay mitigates overestimation bias in RL.
- An optimal convergence analysis of QV-learning is presented.
- We propose Deep VQ-Networks (DVQN), a deep RL extension of QV-learning.
- DVQN performs strongly across ten Atari domains compared to baselines.

ARTICLE INFO

Communicated by D. Wang

Keywords:
Reinforcement learning
Value overestimation
Sample efficiency

ABSTRACT

Q-learning, as a well-known reinforcement learning algorithm, is prone to overestimation of action values. Such overestimation is mainly due to the use of the maximization operator when updating the Q function.

Although existing approaches attempt to reduce overestimation bias, they typically retain the maximization or minimization operator in the update process. Recognizing that these operators are the root cause of biased value estimation, we aim to eliminate these operators altogether. An existing tabular RL algorithm, QV-learning, jointly learns a state-value function and an action-value function without using the maximization or minimization operator; however, it leaves the analysis related to overestimation bias unaddressed. We fill this gap by conducting a targeted evaluation of QV-learning with experience replay applied, demonstrating its significant effectiveness in addressing overestimation bias and superior sample efficiency. Notably, we provide a theoretical analysis of the optimal convergence of QV-learning, which is absent from prior studies.

Moreover, we propose a novel deep RL extension of QV-learning, called Deep VQ-Networks (DVQN). Given the noisy learning environment in the deep RL setting, DVQN accounts for the exploration policy's bias towards the overestimated actions, thereby reducing the collection of poor data caused by overestimation and improving training efficiency. We evaluate DVQN across ten Atari game domains and demonstrate that it achieves performance that is either superior to or comparable with baselines including: Deep Q Networks, Deep SARSA, Deep Double Q Networks, Clipped Deep Double Q Networks, Averaged DQN, Dueling DQN and DQV-learning.

1. Introduction

Sample efficiency, long recognized as one of the major challenges in reinforcement learning, has seen substantial progress through a spectrum of recent approaches. These span multiple paradigms, including experience imagination via model-based RL [1–4], offline RL [5–9], sample prioritization [10,11], representation learning [12–16], reward shaping [17–20], employing more capable policy representations [7–9,21,22], among others. Beyond these directions, a line of studies specifically targets one inherent problem in reinforcement learning – estimation bias. In this study, we likewise focus on estimation bias and introduce a novel approach to address it.

Q-learning [23] has been one of the most widely applied reinforcement learning algorithms. Combined with function approximation through neural networks, Deep Q Networks (DQN) [24] show strong adaptability to complex discrete control tasks. However, Q-learning is prone to overestimation bias [25,26], which arises from the use of the maximization operation when computing the update targets. When the overestimation of the Q-function is uniform over all actions, the relative preferences among action values remain unchanged, and therefore

* Corresponding author.
Email address: xue@l3s.de (Y. Xue).

https://doi.org/10.1016/j.neucom.2025.131581
Received 2 January 2025; Received in revised form 26 August 2025; Accepted 14 September 2025
Available online 18 September 2025

the policy remains the same. However, when overestimation is not uniform, it can be detrimental to policy learning [25]. This phenomenon is particularly problematic with Deep Q Networks. SARSA and Expected SARSA also utilize the maximization operator for updates, making them susceptible to the overestimation bias as well.

One classic solution to the problem of overestimation is Double Q-learning [26]. It introduces two independent unbiased Q-functions to estimate action values. To update the first Q-function, the action with the highest value associated with Q-function one is selected and evaluated by the Q-function two, it is just the opposite when updating the second Q-function. Despite its efficacy against overestimation, Double Q-learning brings underestimation of the maximum expected value at the same time. Deep Double Q Networks (DDQN) [27] extend this idea to the setting of Deep Q Networks, in that they treat the target Q-function in DQN as the second independent unbiased action value estimator. DDQN has been shown to alleviate the overestimation problem and improve the reward performance in a wide range of domains. However, the target Q-function in DQN is introduced to stabilize the training process and is synchronized with the Q-function regularly. Therefore, the two Q functions in DDQN are not fully independent, in some cases DDQN still suffers from overestimation. Clipped Double Q-learning (CDDQN) [28] addresses this concern by taking the minimum between the two independent Q-functions in an actor-critic setting [29]. CDDQN is applied to tackle overestimation in TD3 [28] and SAC [30], two of the most popular RL algorithms mainly for continuous control tasks. However, CDDQN is still not a complete answer, it can effectively suppress the overestimation bias, but also tends to underestimate at the same time, since the employed minimum operation over action value evaluation is not lower bounded. Moreover, [31] propose another solution to the overestimation phenomenon, called Averaged DQN, which averages across the estimates of K previously learned Q functions to compute the target Q values. Averaged DQN helps reduce the variance of target approximation error, resulting in improved performance and stability. However, the number of Q functions, K, is a task-sensitive hyperparameter, often resulting in unstable performance across tasks and prolonged search for its optimal value.

While the aforementioned methods have managed to reduce overestimation bias to some degree, they continue to rely on maximization and minimization operators during value updates, leaving the root cause of both over- and underestimation bias intact. This raises a natural and intriguing research question: can we remove the very source of estimation bias altogether during value updates—by abandoning maximization and minimization—and still achieve superior performance compared to existing approaches?

An earlier tabular algorithm, QV-learning [32], proposes to jointly learn a state-value function V and an action-value function Q, both of which share the same TD target, achieving updates without using the maximization and minimization operators. In this work, we offer a targeted and comprehensive evaluation of QV-learning with experience replay incorporated, to examine how eliminating the maximization and minimization operators contributes to its effectiveness in addressing the overestimation bias – which goes significantly beyond the scope of the prior work. Moreover, we provide a theoretical analysis of QV-learning's optimal convergence, filling a gap left in previous studies. Our convergence analysis remains valid regardless of whether experience replay is used.

Building on QV-learning, we propose a novel deep RL algorithm, Deep VQ-Networks (DVQN). Beyond eliminating the root cause of overestimation bias during the update process, DVQN also accounts for the exploration policy's bias towards overestimated actions in the deep RL setting, which reduces the collection of poor data caused by overestimation and enhances training stability and efficiency. To this end, DVQN incorporates two independently initialized Q networks and derives the exploration policy from their mean estimates, distinguishing it from QV-learning, which uses a single Q-function. Our experimental evaluation demonstrates the superior performance of DVQN in Atari domains.

To summarize, we make the following main contributions.

1. We conduct a targeted evaluation of QV-learning with experience replay to demonstrate its remarkable effectiveness in addressing overestimation bias and improving sample efficiency.
2. We provide a theoretical analysis regarding QV-learning's optimal convergence.
3. We propose Deep VQ-Networks (DVQN), which build upon QV-learning. DVQN achieves performance that is on par with or surpasses that of the baselines.

The corresponding source code is released at https://github.com/xy9485/DVQN_RL.

2. Background and related work

Reinforcement learning (RL) involves the interaction between the agent and the environment. An RL task can be depicted by a Markov Decision Process (MDP) in the form of a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ indicate the state and action space respectively, $\mathcal{P}(s'|s,a)$ and $\mathcal{R}(s,a)$ denote the transition probability function and the reward function. The goal is to learn a policy $\pi(s)$ that maximizes the expected discounted cumulative reward for any given state. Given a policy π, we can define the discounted accumulative reward when taking action a in state s:

$$Q_\pi(s,a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \mid s_0 = s, a_0 = a \right] \tag{1}$$

where γ is the discount factor. The goal is to find the optimal value function $Q^*(s,a) = \max_\pi Q_\pi(s,a)$ so that the optimal policy can be induced by:

$$\pi^*(a|s) = \begin{cases} 1 & \text{if } a = \underset{a \in \mathcal{A}}{\operatorname{argmax}} Q^*(s,a) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$Q^*(s,a)$ is also the unique solution of Bellman optimality equation as: $Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} \mathcal{P}^a_{ss'} \max_{a'} Q_* \left(s', a'\right)$. Q-learning is a classic model-free RL algorithm to compute $Q^*(s,a)$, which updates $Q(s,a)$ iteratively by:

$$Q(s,a) = Q(s,a) + \alpha \left[R(s,a) + \gamma \max_{a'} Q\left(s', a'\right) - Q(s,a) \right] \tag{3}$$

In Deep Q Networks (DQN) [24], $Q(s,a)$ is approximated by a neural network ϕ, which is iteratively updated by minimizing the loss function below using gradient descent:

$$\mathcal{L}(\phi) = \left[r + \gamma \max_{a' \in \mathcal{A}} Q_{\hat{\phi}}\left(s', a'\right) - Q_\phi(s,a) \right]^2 \tag{4}$$

where (s, a, s', r) is sampled from the replay buffer which stores a fixed amount of the latest experiences. $\hat{\phi}$ denotes the parameters of the target Q network which is synchronized with ϕ at regular intervals. To stabilize the training of Q_ϕ, regular synchronization of its parameters with those of the target network is crucial. DQN is prone to overestimation of Q values [27] when evaluating the selected action a', due to the maximization operator.

To tackle the overestimation, Double DQN (DDQN) [27] takes the target network $Q_{\hat{\phi}}(s,a)$ in DQN as another unbiased action value estimator to evaluate action a' selected by $Q_\phi(s,a)$, thus its loss function

becomes:

$$\mathcal{L}(\phi) = \left[r + \gamma Q_{\hat{\phi}}\left(s', \arg\max_{a'} Q_{\phi}(s', a')\right) - Q_{\phi}(s, a)\right]^2 \quad (5)$$

However, since $\hat{\phi}$ is synchronized with ϕ regularly, $Q_{\phi}(s,a)$ and $Q_{\hat{\phi}}(s,a)$ are not fully independent. Thus, the overestimation cannot be effectively alleviated.

Clipped Double DQN (CDDQN) [28] improves upon DDQN by employing two independent action value estimators: $Q_{\phi_1}(s,a)$ and $Q_{\phi_2}(s,a)$. CDDQN was first introduced to an actor-critic setting, it can be adapted in a critic-only setting by ensuring that the policy which interacts with the environment is induced by $Q_{\phi_1}(s,a)$. When computing the target value, action a' with the highest value on s' is selected according to $Q_{\hat{\phi}_1}(s',a')$ and then evaluated by the minimum between $Q_{\hat{\phi}_1}(s',a')$ and $Q_{\hat{\phi}_2}(s',a')$. The loss function optimized by CDDQN is given as:

$$\mathcal{L}(\phi_i) = \left[r + \gamma \min_{i\in\{1,2\}} Q_{\phi_i}\left(s', \arg\max_{a'} Q_{\hat{\phi}_1}(s', a')\right) - Q_{\phi_i}(s, a)\right]^2 \quad (6)$$

While CDDQN suppresses the overestimation, it is prone to underestimation [33], since the target value is not lower bounded.

Averaged DQN [31] also aims to solve the overestimation issue by averaging previously learned Q values estimates, more specifically, averaging across previous versions of target Q networks. The averaged target Q value is used to replace the original target Q value in the DQN loss function.

ES-DQN [34] addresses the issue of unstable behavior arising from overestimation bias by leveraging multi-step updates. Recognizing the pronounced sensitivity of performance to the choice of the update horizon n, ES-DQN is introduced to dynamically adjust this horizon, thereby more effectively leveraging the advantages of multi-step updates to enhance reinforcement learning performance. Since using multi-step updates is orthogonal to both our method and the aforementioned approaches addressing overestimation bias, ES-DQN can readily be integrated with them. Accordingly, we do not include a direct comparison in our evaluation, as this technique is complementary rather than competitive.

An approach closely related to our DVQN is DQV-learning [35], a prior deep RL extension of QV-learning. Distinct from DQV-learning, DVQN explicitly mitigates the tendency of an exploration policy induced from a single Q-network to favor overestimated actions, reducing the risk of collecting poor data caused by overestimation. Accordingly, DVQN incorporates two independently initialized Q-networks and derives the exploration policy based on their mean estimates, in contrast to DQV-learning, which uses only one Q-network. We provide a performance comparison between DVQN and DQV-learning in Appendix A.7.

DVQN also bears resemblance to Dueling DQN [36] in incorporating a state-value function. Dueling DQN is proposed to improve the sample efficiency of DQN by replacing the original Q-network with the sum of a state-value function and an advantage function. However, since Dueling DQN still follows the standard Q-learning update rule, it does not address the issue of overestimation bias. See Section 4.1 for further analysis. Our results demonstrate that DVQN outperforms Dueling DQN on the Atari game benchmarks.

Beyond improving sample efficiency by addressing overestimation bias, advancements in improving sample efficiency have emerged from diverse directions. In model-based RL, methods including [1–4], have achieved high-quality experience imagination via more accurate world models, without requiring additional online interaction. To address the cost and limited accessibility of online interaction, recent advancements in offline RL [5–9] have shown that policies with remarkable performance can be learned from offline datasets, thereby significantly enhancing the sample efficiency and extending RL's applicability to a broader range of scenarios. In model-free RL, experience replay with more sophisticated sampling prioritization [10,11] has also demonstrated effectiveness in improving sample efficiency. Furthermore, leveraging representation learning as auxiliary objectives, approaches [12–16] have substantially improved the learning efficiency of RL algorithms through employing unsupervised learning objectives. In sparse-reward scenarios, reward shaping [17–20] provides an effective means of injecting intrinsic rewards, yielding denser feedback and consequently enhancing sample efficiency. Moreover, recent studies [7–9,21,22] have utilized diffusion-based policy representations to address complex and multi-modal action distributions, resulting in significantly enhanced sample efficiency. Notably, our method (DVQN) can be seamlessly combined with the aforementioned approaches, as their objectives lie in directions orthogonal to our focus. Most of these methods still rely on the maximization or minimization operator during value updates, which causes estimation bias; in contrast, our method eliminates both operators while achieving outstanding performance.

Algorithm 1 QV-learning with experience replay.

Initialize $Q(s,a)$ arbitrarily for all $s \in \mathcal{S}, a \in \mathcal{A}$, except $Q(terminal,\cdot) = 0$
Initialize $V(s)$ arbitrarily for all $s \in \mathcal{S}$, except $V(terminal) = 0$
Initialize experience-replay buffer D
Initialize step size α and $\epsilon \in (0,1)$
for each episode **do**
  Initialize s
  repeat for each step of episode
    Take action a from s (ϵ-greedy), observe s', r
    Store transition (s,a,s',r) to D
    Randomly sample transitions $B = \{(s,a,s',r)\}$ from D
    $V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)]$
    $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma V(s') - Q(s,a)]$
    $s \leftarrow s'$,
  until s is terminal

3. QV-learning with experience replay

In this section, we conduct a targeted evaluation of QV-learning with experience replay to assess its effectiveness in addressing overestimation bias and sample efficiency. QV-learning jointly learns a state value function and an action value function, denoted by V and Q, respectively. The temporal-difference(TD) targets for both V and Q functions are identical and based on the estimates of the V function, achieving updates without employing the maximization or minimization operator. Algorithm 1 provides the pseudocode for QV-learning with experience replay. Notably, QV-learning was proposed as an on-policy RL algorithm. With the incorporation of experience replay, QV-learning's updates are performed over batches uniformly sampled from the replay buffer. As a result, the V function effectively reflects the state value function of a mixture or average of the past policies, rather than the current policy used to collect experiences. This makes QV-learning with experience replay an off-policy algorithm.

In Section 3.1, we evaluate QV-learning with experience replay along with four baselines, including Q-learning, Double Q-learning, SARSA and Expected SARSA. The evaluation is conducted across four distinct task settings. In Appendix A.3 and A.4, we augment the evaluation with more challenging tasks in two grid world environments.

By Theorem 1, we provide the optimal convergence analysis of QV-learning.

Theorem 1 (**Convergence of QV-learning**). *Consider a finite state-action MDP and apply GLIE (i.e., non-greedy actions are chosen with vanishing probabilities) learning policy π given as a set of probabilities $\Pr(a|s,t,n_t(s),Q)$. Assume that at time step t, action a_t is chosen according to π which uses $Q = Q_t$. V_t and Q_t are updated to V_{t+1} and Q_{t+1} as follows:*

$$V_{t+1}(s_t) = \left(1 - \alpha_t\right) V_t(s_t) + \alpha_t \left(r_t + \gamma V_t(s_{t+1})\right) \quad (7)$$

$$Q_{t+1}(s_t, a_t) = \left(1 - \alpha_t\right) Q_t(s_t, a_t) + \alpha_t \left(r_t + \gamma V_t(s_{t+1})\right) \quad (8)$$

Q_t converges w.p.1 to the optimal Q-function as long as:

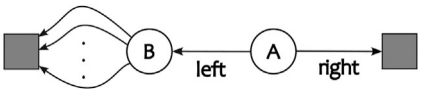


Fig. 1. Example task to illustrate the effectiveness of QV-learning. An episode starts from state A. The gray colored boxes represent terminal states.

1. *Q and V values are stored in lookup tables.*
2. $\mathrm{Var}[r(s,a)] \leq \infty$
3. $0 \leq \alpha_t \leq 1, \quad \sum_t \alpha_t(x,a) = \infty, \quad \sum_t \alpha_t^2(x,a) < \infty$
4. $\lim_{t\to\infty} V_t(s) = \lim_{t\to\infty} Q_t(s,a_t), a_t = \pi^{Q_t}(s)$, *where* $\pi^{Q_t}(s)$ *is the GLIE policy derived from* Q_t

We refer to the proof in Appendix A.1. The fourth condition is supported by empirical evidence (see estimates of the V function and Q function in Appendix A.3). Notably, Theorem 1 remains valid whether or not experience replay is used.

3.1. QV-learning with experience replay in four task settings

In this section, we evaluate QV-learning with experience replay applied alongside four baseline methods within the environment depicted in Fig. 1. This simple environment is introduced in [37] to demonstrate the performance of Q-learning and Double Q-learning against overestimation bias. To demonstrate the evaluation from various perspectives, the experiments are conducted across four different task settings, the original setting with stochastic rewards and three variants of it. A replay buffer of size 100 and a batch size of 8 are used across all task settings.

Based on experimental results across the four task settings, we show that when the reward or value update processes are stochastic, methods including Q-learning, SARSA and Expected SARSA exhibit pronounced overestimation bias. In contrast, Double Q-learning and QV-learning do not experience the issue, with QV-learning significantly outperforming the others. Even in the absence of stochasticity, QV-learning consistently achieves the best performance; while the performance is on par with Expected SARSA, QV-learning requires substantially fewer function evaluations for value updates. The advantage of QV-learning becomes even more pronounced with an expanded action space. In the remainder of this section, we provide detailed discussion for each task setting.

Stochastic rewards. As shown in Fig. 1, each episode starts from state A, which has two available actions. Taking the right action transitions to a terminal state with a reward of zero. Taking the left action transitions to state B, where 20 actions are available, each action leads to the other terminal state and results in a reward sampled from a Gaussian distribution $\mathcal{N}(-0.1, 1)$. Thus, the expected return of taking the left action is -0.1, implying that the optimal policy is to take the right action in state A and terminate the episode. The behavior policy is ϵ-greedy with ϵ decaying from 0.1 to zero. Value functions are tabular and initialized with zero values.

Fig. 2 (first column) illustrates that Q-learning, SARSA, and Expected SARSA initially learn to take the left action much more often than the right action and consistently select it significantly more often than 0% throughout the entire episode horizon. This indicates that $Q(A, \text{left})$ is overestimated by these three methods, which is also reflected in the illustration of $\max_a Q(A,a)$ in Fig. 2 (bottom row), where their Q estimates remain above the optimal value (equal to the true return of the best learned policy). The overestimation in these methods arises from the use of the maximum operation during Q value updates. Double Q-learning quickly corrects the overestimation and outperforms Q-learning, SARSA, and Expected SARSA. In contrast, QV-learning entirely avoids overestimating $Q(A, \text{left})$ since the beginning of learning and converges to the optimal policy, exhibiting a significant advantage in sample efficiency over Double Q-learning.

Noisy value updates. Based on the same setting as above, we remove the reward stochasticity and add independent Gaussian noise sampled from $\mathcal{N}(0, 0.01)$ to tabular-stored values after each episode. Actions from B now cause a deterministic reward of -0.1. This setting simulates the inherent noisy updates that occur when using neural networks for function approximation. As shown in the second column of Fig. 2, QV-learning again outperforms all baselines and converges to a near-optimal policy. Q-learning, SARSA and Expected SARSA still suffer from the overestimation of Q values, indicating that stochasticity in value updates can also

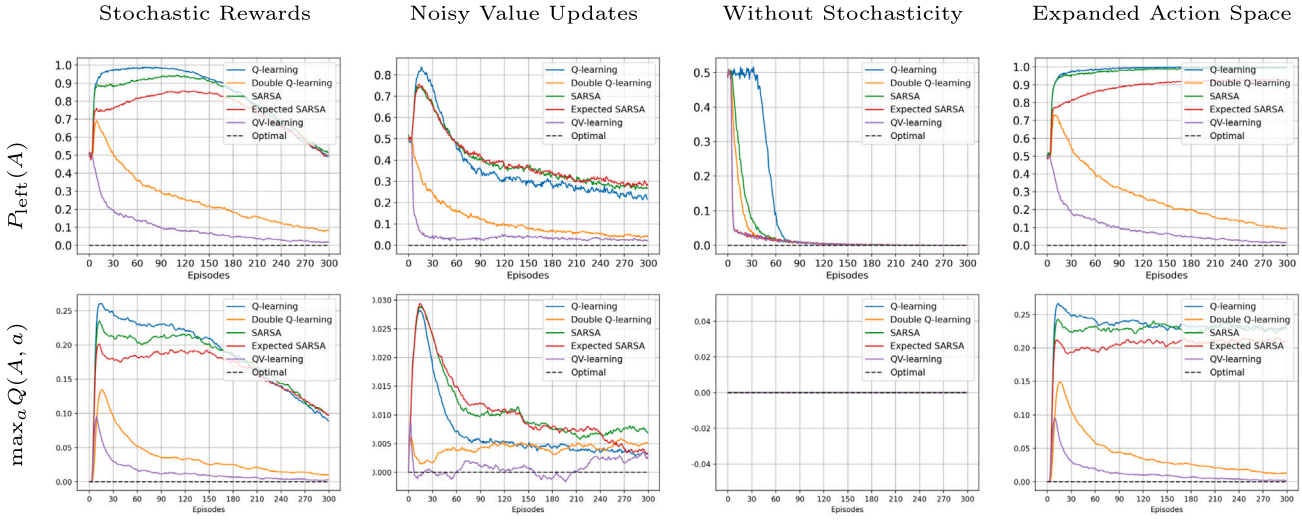


Fig. 2. Evaluation of QV-learning with experience replay across four different task settings, using a replay buffer size of 100 and a batch size of 8. The top row presents how frequent action left is taken from state A against number of timesteps. The bottom row presents the maximum of Q value estimates in state A, where the horizontal dashed line indicates the optimal value of zero. The results are averages across 1000 repetitions. In the setting of **Stochastic Rewards**, actions from B cause rewards sampled from Gaussian distribution $\mathcal{N}(-0.1, 1)$. In the setting of **Noisy Value Updates**, tabular values are added with independent Gaussian noises (sampled from $\mathcal{N}(0, 0.01)$) after each episode, actions from B cause deterministic reward of -0.1. The setting **Without Stochasticity** is based on **Stochastic Rewards** with the stochasticity in reward and updates being removed. The **Expanded Action Space** setting is identical to **Stochastic Rewards**, except that the available actions are increased from 20 to 100.

be a source of overestimation bias. The illustration of $max_a Q(A, a)$ also shows the overestimation with Q-learning, SARSA and Expected SARSA, whereas QV-learning estimates the value close to the optimal value.

Without stochasticity. Following the setup described in setting *Stochastic rewards*, we remove the stochasticity from both rewards and value updates. This effectively eliminates the sources of potential overestimation bias caused by the use of the maximization operator. As shown in the third column of Fig. 2, Q-learning, SARSA and Expected SARSA no longer overestimate Q values. QV-learning and Expected SARSA are the most sample-efficient methods with comparable performance. It is worth noting that when updating Q-values, QV-learning requires only a single function evaluation, as it uses the value function V to compute the target for both the V and Q estimates. In contrast, Expected SARSA requires evaluating all Q-values for a given state to compute its target. Therefore, despite achieving comparable performance, QV-learning is more computationally efficient. As the size of the action space increases, during updates, the cost of Q-value evaluations in Expected SARSA grows proportionally, while the evaluation expense for QV-learning remains constant, regardless of the number of actions.

For deeper insights behind the rationale of the superior learning efficiency of QV-learning: The key is to focus on how $Q(A, \text{left})$ evolves following different methods. Consider Q-learning, the target value for $Q(A, \text{left})$ is $r+\gamma\, max_{a'} Q(B, a')$ and all Q values are zero initialized. Hence, this target value will remain zero until all action values for $Q(B, *)$ are updated to at least a negative value (-0.1 with enough updates), so that $Q(A, \text{left})$ turns negative while $Q(A, \text{right})$ remains zero, therefore the optimal policy is derived. This means the number of steps before convergence is proportional to the number of actions in state B. QV-learning, on the other hand, uses $r+\gamma V(B)$ as the target value for both V and Q estimates. A single instance of any action taken from state B will result in $V(B)$ being updated to a negative value, which in turn makes $Q(A, \text{left})$ negative. Expected SARSA presents the same effect at the expense of function evaluations for all actions available in B to exactly compute the expected Q value involved in $r + \gamma \mathbb{E}_{a'}[Q(B, a')]$, while QV-learning only requires one evaluation of $V(B)$. SARSA uses sampling to approximate $\mathbb{E}_{a'}[Q(B, a')]$, which results in requiring more steps to converge, compared to Expected SARSA. In Double Q-learning, the target value for updating Q_1 is $r+\gamma Q_2(B, argmax_{a'} Q_1(B, a'))$; unlike Q-learning, $Q_2(B, argmax_{a'} Q_1(B, a'))$ is not guaranteed to remain zero even before all $Q_1(B, a')$ values are updated to negative; updates of Q_2 share the similar characteristics. As a result, Double Q-learning converges to the optimal policy in fewer steps than Q-learning.

Expanded action space. Based on the setting of *Stochastic rewards*, we increase the number of available actions in state B from 20 up to 100. As shown in the fourth column of Fig. 2, QV-learning continues to perform the best, achieving comparable performance as in the *Stochastic Rewards* setting, demonstrating robust sample efficiency in the presence of variable action spaces. In contrast, Q-learning, SARSA and Expected SARSA encounter substantial degradations in performance.

In addition, we also conduct an evaluation under the same settings described above, but without using experience replay. As shown in Appendix A.2, the results are similar to those in Fig. 2.

3.2. Extensive evaluation in grid world environments

To broaden and enhance the validations from Section 3.1, we also conduct experimental evaluations in two grid world environments. We devise a 2-room grid world (Fig. A.9) inspired by the MDP design in Fig. 1. We also run experiments in a 3x3 grid world, introduced in Double Q-learning [26]. Additionally, we expand the action spaces with Do-Nothing actions to investigate more regarding learning efficiency. The two grid worlds offer more challenging tasks for QV-learning and baselines, revealing more insights in the presence of an enlarged policy search space. Details of experiments in grid worlds can be found in Appendix A.3 and A.4.

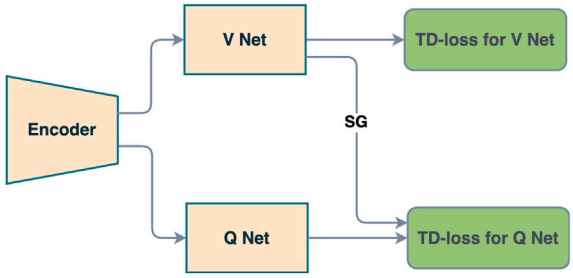


Fig. 3. Architecture of DVQN. The encoder is shared between the state value network (V Net) and the action value network (Q Net) to improve training efficiency. SG denotes *stop gradient*. V Net is learned on its own temporal difference loss. TD-loss for Q Nets takes target values based on the output of V Net, yet V Net is not to be updated due to the SG operator. The two Q networks are plotted as one for simplicity of illustration.

Our experimental results demonstrate that QV-learning outperforms all other baselines in reward performance and speed of convergence to the optimal policy. In addition, we present Q-value estimates for all methods. The observed superior performance of QV-learning over the baselines becomes more evident in the setting with an expanded action space.

Moreover, we present estimation of the V function and Q function in Appendix A.3, showing that the discrepancy between V and Q converges to near zero. This provides the empirical validation for the fourth condition outlined in Theorem 1. A similar observation for DVQN is provided in Appendix A.6.

4. Deep VQ-networks

To overcome the overestimation problem in Deep Q Networks (DQN), we propose Deep VQ-Networks (DVQN) based on QV-learning. The pseudocode for DVQN is presented in Algorithm 2. A schematic diagram of DVQN with a brief description of the architecture is provided in Fig. 3.

In particular, DVQN learns a state value function $V_\theta(s)$ and uses it to construct target values for the estimation of the Q networks $Q_{\phi_{1,2}}$. Specifically, in lines 12–13 of Algorithm 2, $V_\theta(s)$ is updated based on its own temporal-difference loss without involving a maximization operator. In lines 14–15 of Algorithm 2, the loss is constructed with respect to ϕ_1 and ϕ_2 by using the same target value in line 13.

Algorithm 2 DVQN.

1: Initialize the environment and replay buffer D
2: Initialize Q-function parameters ϕ_1 and ϕ_2
3: Initialize V-function parameters θ
4: Initialize target networks $\hat{\theta} \leftarrow \theta$
5: Initialize replay period n, update target networks period m
6: **for** $t = 1$ to T **do**
7: Based on $(Q_{\phi_1} + Q_{\phi_2})/2$, select action a given state s ($\epsilon-$greedy)
8: Observe reward r and next state s'
9: Store transition (s, a, s', r) to replay buffer D
10: **if** t mod n **then**
11: Randomly sample transitions $B = \{(s, a, s', r)\}$ from D
12: Update V_θ by gradient:
13: $\nabla_\theta \frac{1}{|B|} \sum_{(s,a,s',r)\in B} \left(r + \gamma V_{\hat{\theta}}(s') - V_\theta(s)\right)^2$
14: Update Q_{ϕ_1} and Q_{ϕ_2} by gradient:
15: $\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,s',r)\in B} \left(r + \gamma V_{\hat{\theta}}(s') - Q_{\phi_i}(s, a)\right)^2$ for $i = 1, 2$
16: **if** t mod m **then**
17: Update target networks:
18: $\hat{\theta} \leftarrow \theta$

Distinct from QV-learning, DVQN employs two Q networks instead of one. In the deep RL setting, an exploration policy (e.g., ϵ-greedy) derived from a single Q network tends to favor overestimated actions, resulting in the collection of poor data caused by overestimation, which in turn impairs training stability and sample efficiency. To mitigate this, DVQN derives the exploration policy from the mean estimates of the two independently initialized Q-networks ($Q_{\phi_{1,2}}$), both of which are updated while sharing the same TD-target as depicted in line 15 of Algorithm 2. It is worth noting that for each Q function in DVQN, the convergence to the optimality described in Theorem 1 still remains valid. Even if there were transitions collected with the occurrence of overestimation, the updates on these transitions in DVQN do not apply the maximization operator, which could otherwise lead to the accumulation of value overestimation.

DVQN employs a shared convolutional encoder for feature extraction, followed by one MLP head for the V function and two MLP heads for the two Q functions. This design reduces the total number of parameters while ensuring that the V and two Q networks operate over a consistent feature space, which improves stability.

Fig. 3 illustrates the architecture of DVQN, the *encoder*, with convolutional neural networks as the feature extractor, is shared between V and Q networks. Both networks have MLP (Multi-Layer Perceptron) modules to estimate state and action values respectively; the former has a single output and the latter has an output size equal to the size of the action space. SG indicates *stop gradient*. Compared to DQN, the additional training weights in DVQN arise from the MLP components of the V network and the additional Q network. As an easy-to-implement

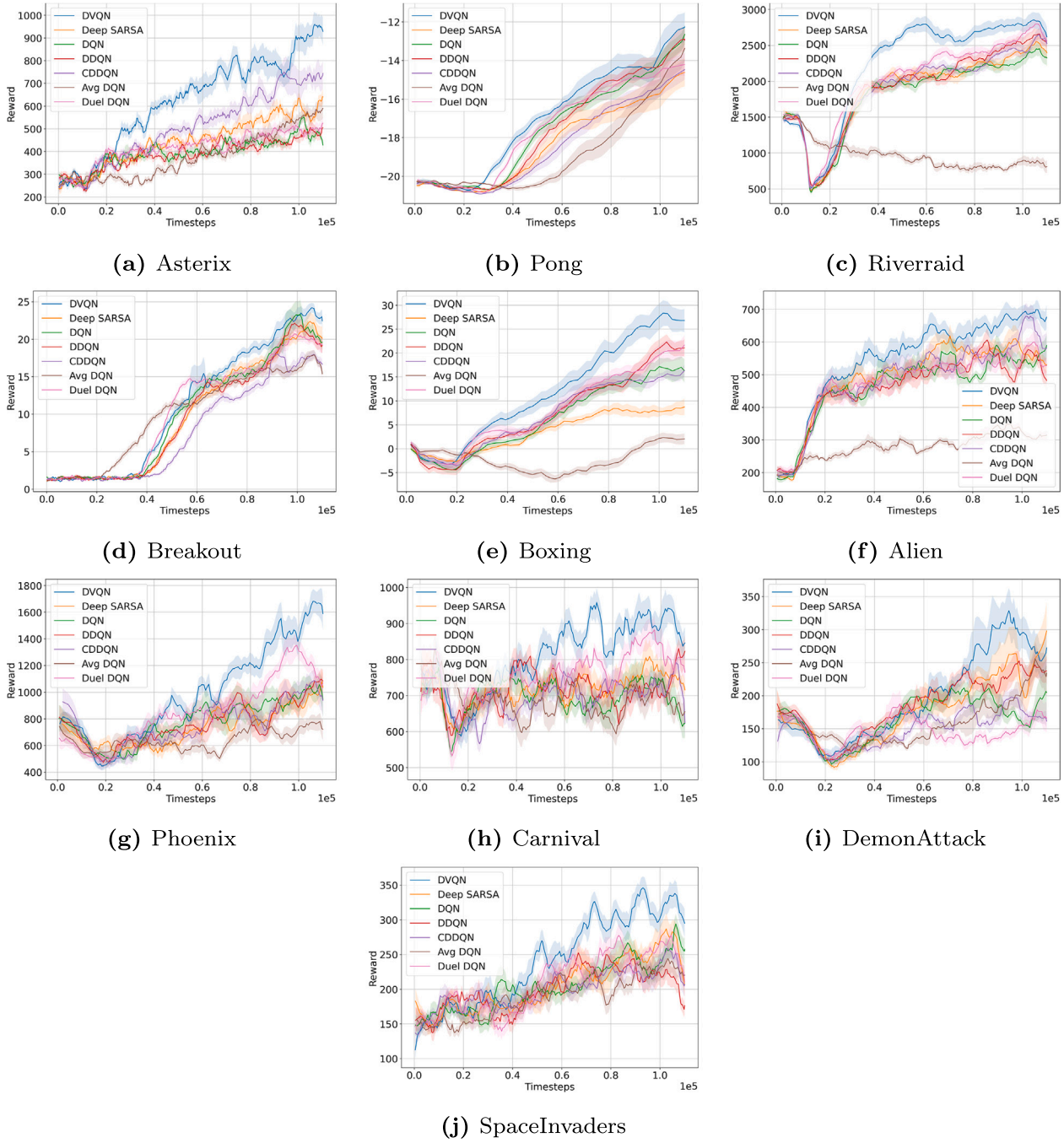


Fig. 4. Reward performance against 100k timesteps.

extension of DQN, DVQN does not sacrifice sample efficiency since V_θ and $Q_{\phi_{1,2}}$ are updated over the same data.

4.1. Comparison with dueling DQN

In this section, we compare DVQN with Dueling DQN [36], due to the resemblance in learning a state value function.

Dueling DQN is a technique designed to enhance the sample efficiency of DQN. It modifies the internal architecture of the Q function by splitting it into two separate network streams: one for the state value function and another for the advantage value function. The outputs of these streams are combined to construct the Q values estimates. Dueling DQN operates under the assumption that for many states, the choice of actions is irrelevant to the task, the value of each action does not need to be precisely learned; updating the state value network causes a corresponding shift in the Q values for all actions, thereby enhancing learning efficiency under this assumption. However, this assumption may be overly strong and is not guaranteed to hold across all tasks. In contrast, DVQN does not rely on this assumption and still enhances learning efficiency by using the state value function estimates to construct the target values for Q function predictions. More insights regarding DVQN's superior learning efficiency are clarified in Section 3.1, setting *Without Stochasticity*.

The loss function in Dueling DQN remains the same as in DQN, therefore, the overestimation issue of DQN remains untouched. Additionally, a zero-mean constraint is applied to the advantage value network to prevent the state value network from collapsing into a constant zero and the advantage value network from degenerating into a standard Q

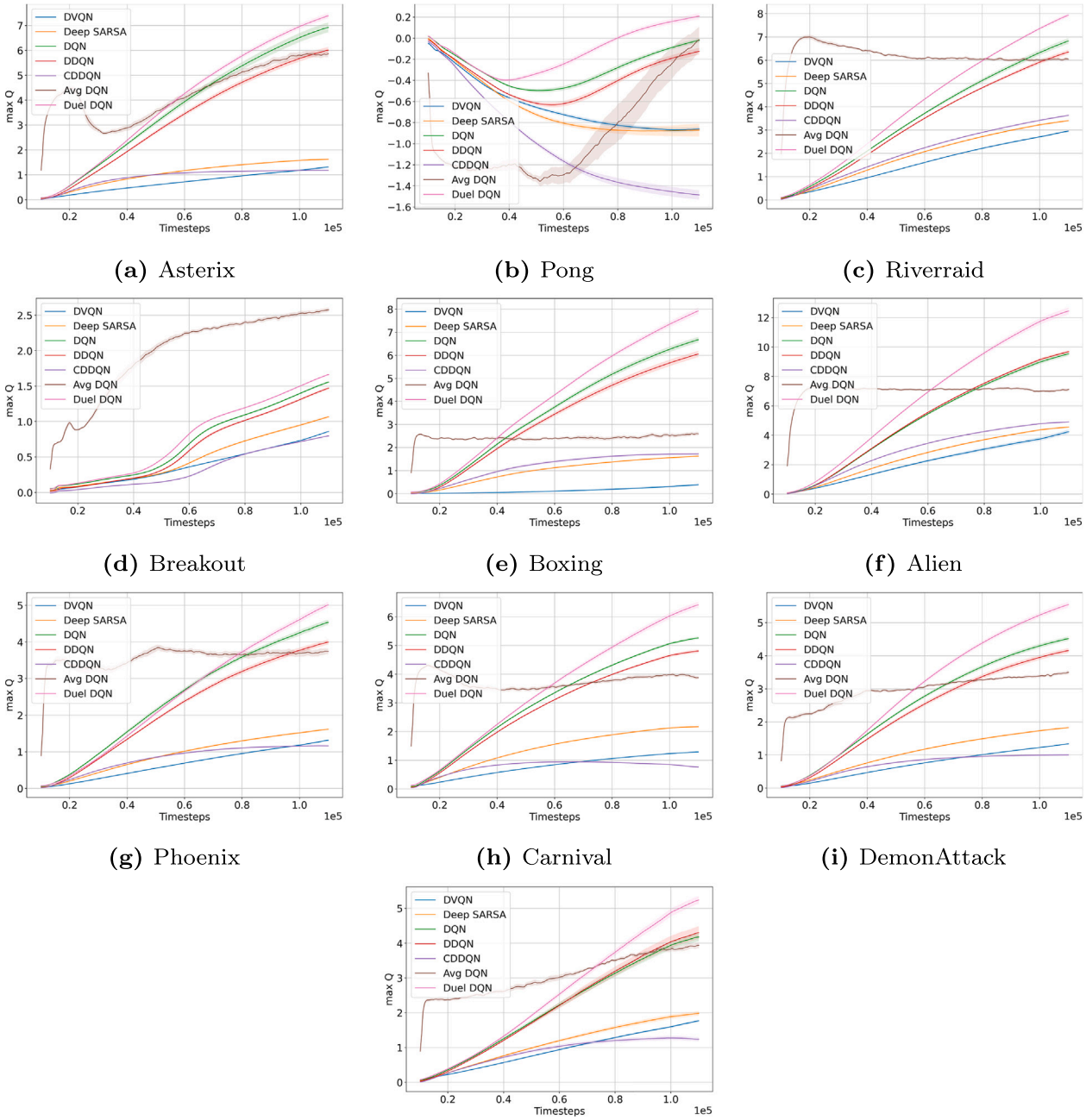


Fig. 5. Max Q value estimation against 100k timesteps.

value network. This constraint could sacrifice the learning efficiency of the Q function. On the other hand, DVQN proposes to learn a separate network alongside the Q networks to estimate the state value function with its own loss function. It tackles overestimation effectively, while Dueling DQN does not.

5. Experiments and analysis

5.1. Experimental setup

Experimental domains. We perform our experiments on Atari games, considering both short and long horizons. For evaluation with a short-term horizon, the Atari 100k [38] benchmark is employed, a sample-constrained benchmark for algorithms dealing with high-dimensional observations (raw pixels) and discrete control. The agent is allowed to interact for 100k steps in total with the environment, roughly equivalent to 2 hours of gameplay by a human. To investigate the longer-term performance of DVQN and baseline approaches, we also conduct additional experiments with horizons of 1 million training steps. The evaluations are carried out in ten Atari game domains, namely Asterix, Pong, Riverraid, Breakout, Boxing, Alien, Phoenix, Carnival, DemonAttack and SpaceInvaders.

Baseline approaches. We compare DVQN (our method) with baseline approaches, including DQN [24], DDQN [27], CDDQN [28], Averaged DQN [31], Deep SARSA, Dueling DQN [36] and DQV-learning [35].

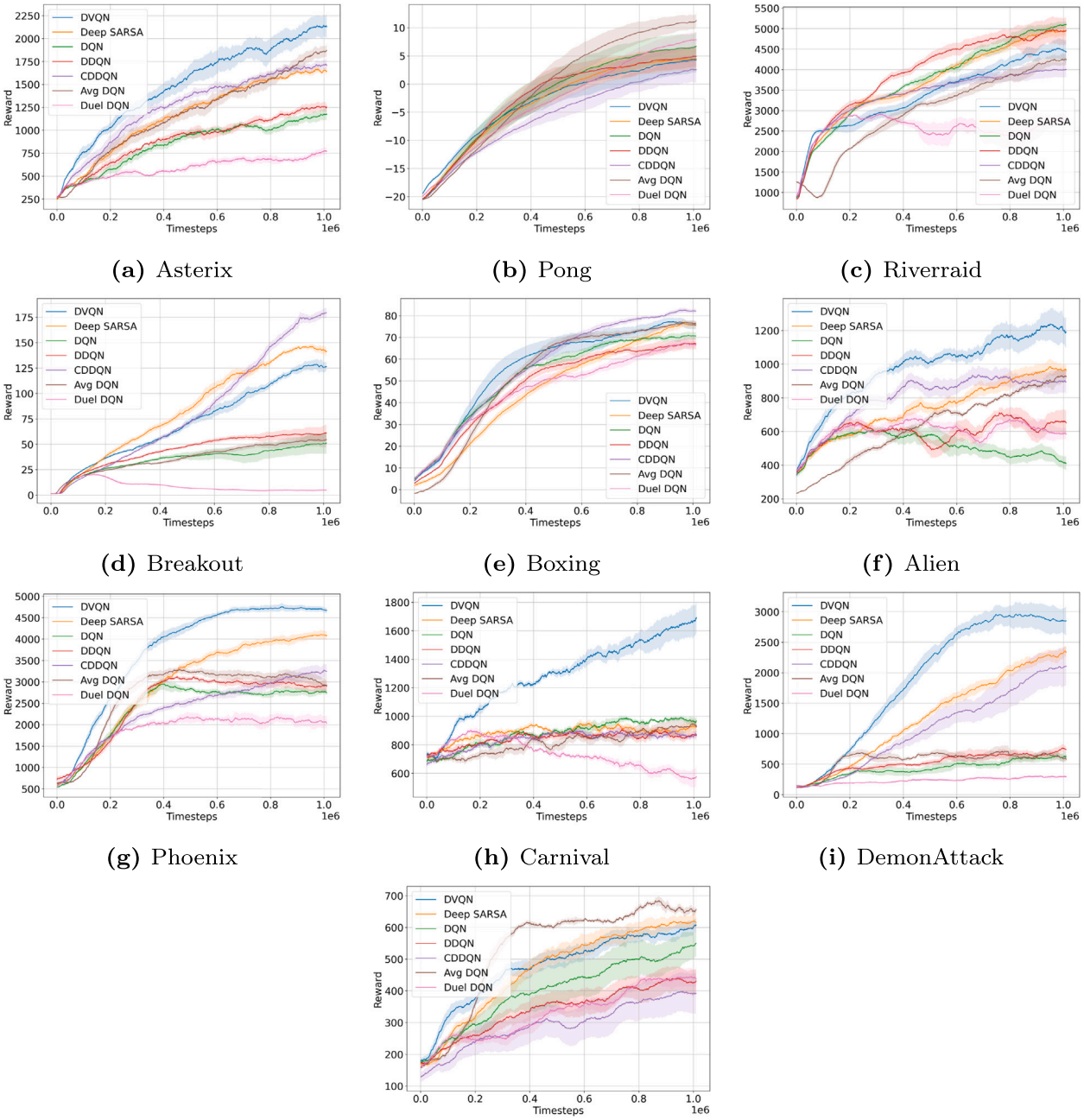


Fig. 6. Reward performance against 1000k timesteps.

Evaluation. Our experimental evaluation is divided into two parts: We **first** compare our method to the baseline approaches in terms of reward performance with short-term and long-term horizons. **Second**, we investigate the robustness and sample efficiency of DVQN when expanding the action spaces with redundant actions.

Observations augmentation. Observations need to be augmented by random shift (DrQ) [39] before feeding them into the networks. DrQ has been shown on the Atari 100k setting to be a simple yet effective method to improve the sample efficiency and robustness of DQN [39]. Instead of solely integrating DrQ into DQN, we apply DrQ to each method included in our evaluation to ensure the fairness of our evaluation. We also conduct experiments without using DrQ, the results are reported in Appendix A.8.

Our experimental results on the Atari 100k setting are averaged across 15 random seeds along with the confidence interval represented as the standard error of the mean. The results for experiments with one million steps are illustrated as averages over 6 random seeds. Hyper-parameters for the experiments and network architecture are detailed in Appendix A.5. We keep the common hyper-parameters shared across all methods the same for fair comparisons.

5.2. Result analysis

5.2.1. Performance over short-term and long-term horizons

In Fig. 4, we assess the performance of the reward against 100k time steps. Deep SARSA, Dueling DQN, DDQN, CDDQN, and Averaged

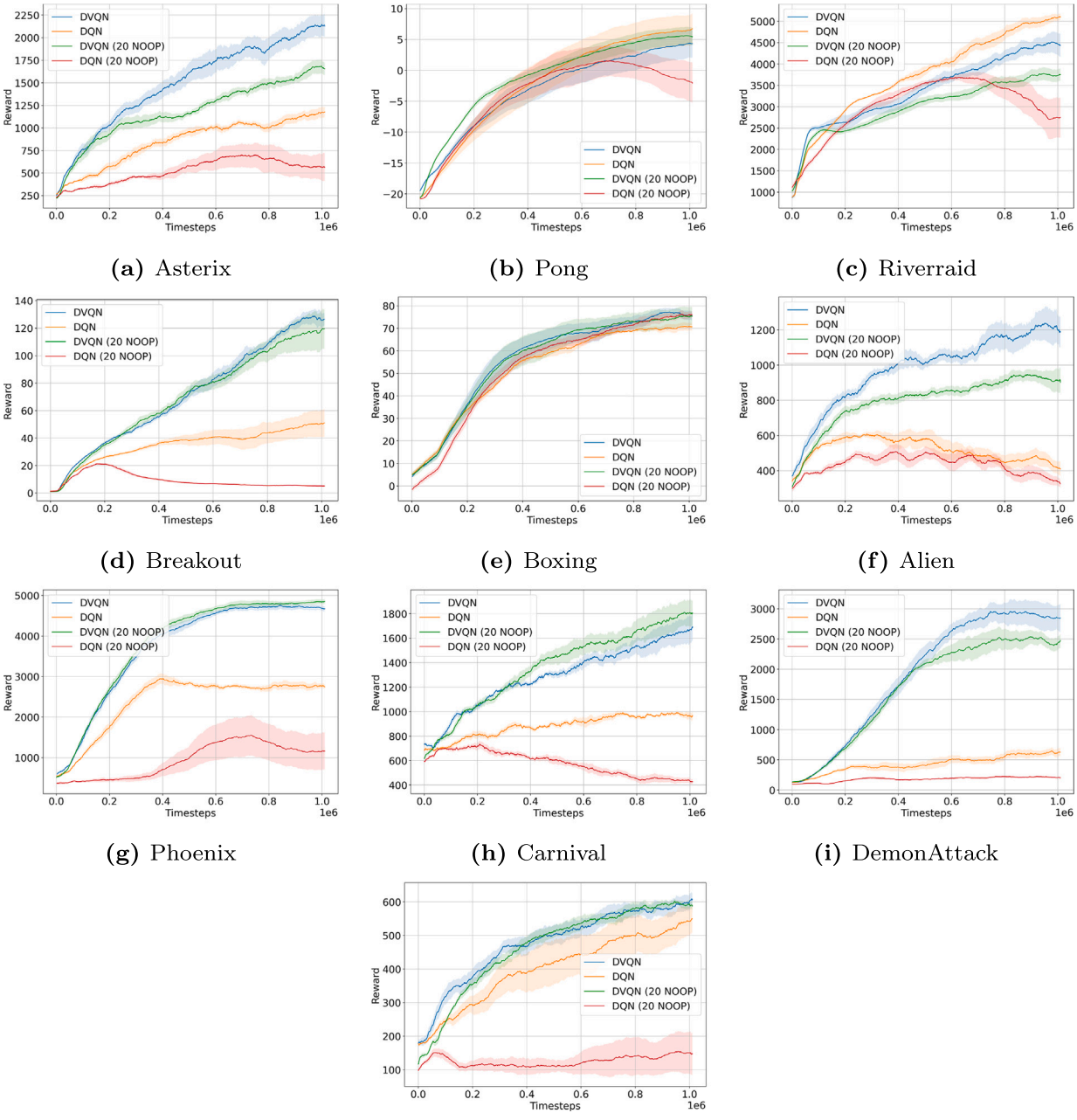


Fig. 7. Reward performance of DVQN and DQN using expanded action space (adding 20 redundant no-operation actions), respectively compared with DVQN and DQN using the original action space.

DQN enhance performance over DQN in certain domains, but none consistently maintain these improvements across all domains. In contrast, our method, DVQN, delivers performance that is significantly superior to all baselines across all domains, demonstrating robust and stable performance. Our empirical findings indicate that the performance of Averaged DQN in Atari games is sensitive to the interval choices for updating the target network. However, we could not identify a single value that yields consistent performance across all domains.

Moreover, we track the estimation of Q value for each method, as shown in Fig. 5. Given the fact that DVQN demonstrates the best reward performance in our experiments, the baseline methods tend to produce a higher bias in Q estimation.

The reward performance over the long-term horizon of 1 million steps is demonstrated in Fig. 6. DVQN outperforms all baselines significantly in five Atari domains. In the other five domains, DVQN's performance remains comparable to the best ones. Moreover, there isn't a single method that consistently excels across all the other methods in these five domains; the one that performs the best in one domain can perform much worse in another. DVQN, in contrast, exhibits more stable and robust performance across all domains.

In Appendix A.6, we demonstrate that the discrepancy in estimations between the V network and Q network is empirically shown to remain close to zero, providing empirical evidence in support of the fourth condition outlined in Theorem 1. Furthermore, we evaluate DVQN against DQV-learning, a prior deep reinforcement learning extension of QV-learning; DVQN demonstrates superior or comparable performance across all ten Atari domains (see Appendix A.7).

5.2.2. Experiments with redundant actions

DVQN enhances learning efficiency compared to DQN not only by addressing biased Q-value estimation but also by incorporating an additional state value function to construct target values for updates (rationale explained in Section 3.1). To verify this, we expand the action spaces of all domains by adding 20 additional no-operation actions, making each domain more challenging to solve.

Fig. 7 presents the performance over one million time steps. With the redundant action space, DQN experiences a significant performance decline in 8 out of 10 domains compared to its performance with the original action space, failing to learn in 4 of them. In contrast, DVQN maintains a comparable performance level between the redundant and original action spaces in 7 domains, while performance reductions in the remaining 3 domains are limited to within 20 percent.

6. Conclusion

In this study, we focus on addressing the overestimation bias of action values in reinforcement learning and aim to examine how eliminating the maximization and minimization operators during the update process contributes to the effectiveness in addressing this issue. QV-learning is a tabular RL algorithm that jointly learns a state-value function and an action-value function without using the maximization and minimization operators. We conduct a targeted evaluation of QV-learning to examine its remarkable effectiveness in addressing overestimation bias–an investigation going well beyond the scope of the prior work. Furthermore, we provide a theoretical analysis of the optimal convergence of QV-learning, filling a gap left in prior studies.

Moreover, we introduce DVQN, a novel deep RL extension of QV-learning. In addition to the unbiased value update process, DVQN also takes into account the exploration bias toward overestimated actions under the deep RL setting. We present empirical evidence demonstrating that DVQN either surpasses or performs on par with DQN, Deep SARSA, DDQN, CDDQN, Averaged DQN, Dueling DQN and DQV-learning in reward performance across ten Atari games. Additionally, we highlight the significant advantage of DVQN in enhancing sample efficiency, as it maintains robust performance even under stress tests with additional redundant actions.

CRediT authorship contribution statement

Yuan Xue: Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Conceptualization. **Daniel Kudenko:** Writing – review & editing, Supervision, Resources, Project administration, Methodology, Formal analysis. **Megha Khosla:** Writing – review & editing, Supervision, Resources, Project administration, Methodology, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was partially funded by the Federal Ministry of Education and Research (BMBF), Germany under the project LeibnizKILabor (grant no. 01DD20003).

Appendix A

A.1. Convergence and optimality of QV-learning

To prove Theorem 1 we will need the following lemma by Singh et al. [40], which is an extension of Theorem 1 from the work of Jaakkola et al. [41].

Lemma 1. *Consider a stochastic process* $(\alpha_t, \Delta_t, F_t)$, $t \geq 0$ *where* $\alpha_t, \Delta_t, F_t : X \rightarrow \mathbb{R}^n$ *satisfy the equations:*

$$\Delta_{t+1}(x) = \left(1 - \alpha_t(x)\right)\Delta_t(x) + \alpha_t(x)F_t(x)$$

where $x_t \in X$ *and* $t = 0, 1, 2, \ldots$. *Let* P_t *be a sequence of increasing* $\sigma - fields$ *such that* α_0 *and* Δ_0 *are* P_0*-measurable and* α_t, Δ_t *and* F_t *are* P_t*-measurable,* $t=1,2,\ldots$. Δ_t *converges to zero w.p.1 when the following conditions are satisfied*

1. *the set* X *is finite*
2. $0 \leq \alpha_t \leq 1, \sum_t \alpha_t(x) = \infty$ *and* $\sum_t \alpha_t^2(x) < \infty$ $w.p.1$
3. $\left\|\mathbb{E}\left[F_t(x) \mid \mathcal{F}_t\right]\right\|_W \leq \gamma \left\|\Delta_t\right\|_W + C_t$, *with* $\gamma \in [0, 1)$ *and* C_t *converges to zero w.p.1*
4. $\mathrm{Var}\left[F_t(x) \mid \mathcal{F}_t\right] \leq K\left(1 + \left\|\Delta_t\right\|_W^2\right)$, *for* $K > 0$

Theorem 1. *Consider a finite ergodic MDP and apply a GLIE learning policy* π^Q *(i.e., non-greedy actions according to the Q function are chosen with vanishing probabilities). Assume that at time step* t, *action* a_t *is chosen according to* π^Q, $Q = Q_t$. V_t *and* Q_t *are updated to* V_{t+1} *and* Q_{t+1} *as follows:*

$$V_{t+1}(s_t) = \left(1 - \alpha_t\right)V_t(s_t) + \alpha_t\left(r_t + \gamma V_t(s_{t+1})\right) \tag{A.1}$$

$$Q_{t+1}(s_t, a_t) = \left(1 - \alpha_t\right)Q_t(s_t, a_t) + \alpha_t\left(r_t + \gamma V_t(s_{t+1})\right) \tag{A.2}$$

Q_t *converges w.p.1 to the optimal Q-function as long as:*

1. *Q and V values are stored in a lookup table.*
2. $\mathrm{Var}[r(s, a)] \leq \infty$
3. $0 \leq \alpha_t \leq 1, \quad \sum_t \alpha_t(x, a) = \infty, \quad \sum_t \alpha_t^2(x, a) < \infty$
4. $\lim_{t\rightarrow\infty} V_t(s) = \lim_{t\rightarrow\infty} Q_t(s, a_t)$, $a_t = \pi^{Q_t}(s)$, *where* $\pi^{Q_t}(s)$ *is the GLIE policy derived from* Q_t

Proof. Let Q^* denote the target Q values. By subtracting $Q^*(s_t, a_t)$ from both sides of Eq. (A.2) we obtain:

$$\begin{aligned} &Q_{t+1}\left(s_t, a_t\right) - Q^*\left(s_t, a_t\right) \\ &= (1 - \alpha)Q_t\left(s_t, a_t\right) + \alpha(r_t + \gamma V_t(s_{t+1})) - (1 - \alpha + \alpha)Q^*(s_t, a_t) \\ &= (1 - \alpha)Q_t\left((s_t, a_t) + \alpha\left(r_t + \gamma V_t(s_{t+1})\right) - (1 - \alpha)Q^*(s_t, a_t) - \alpha Q^*(s_t, a_t)\right. \end{aligned}$$

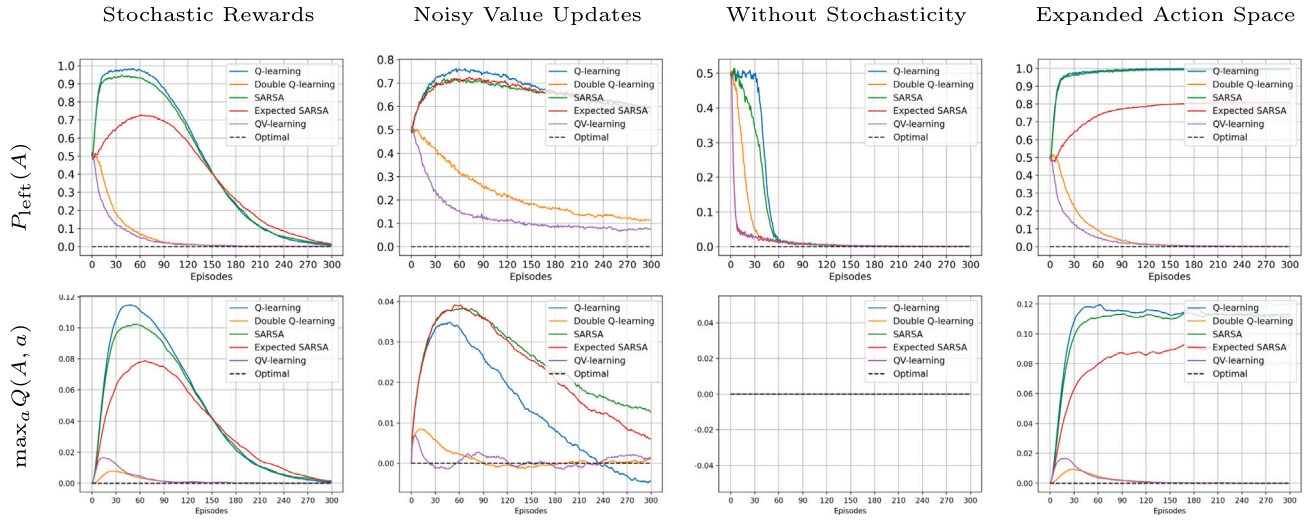


Fig. A.8. Results without applying experience replay. The configurations of the four task variants remain unchanged as described in Fig. 2.

$$= (1-\alpha)\left[Q_t(s_t,a_t) - Q^*(s_t,a_t)\right] - \alpha\left[r_t + V_t(s_{t+1}) - Q^*(s_t,a_t)\right] \quad \text{(A.3)}$$

To use Lemma 1 we define $x_t = (s_t, a_t)$ and denote:

$$\Delta_t(s_t,a_t) := Q_t(s_t,a_t) - Q^*(s_t,a_t) \quad \text{(A.4)}$$

$$F_t(s_t,a_t) := r_t + \gamma V_t(s_{t+1}) - Q^*(s_t,a_t) \quad \text{(A.5)}$$

Adding and subtracting $\gamma Q_t(s_{t+1}, a_{t+1})$ from $F_t(s_t, a_t)$, we obtain:

$$F_t(s_t,a_t) = r_t + \gamma Q_t(s_{t+1},a_{t+1}) - Q^*(s_t,a_t) + \gamma\left[V_t(s_{t+1}) - Q_t(s_{t+1},a_{t+1})\right] \quad \text{(A.6)}$$

where $a_{t+1} = \pi^{Q_t}(s_{t+1})$, π^{Q_t} is a *GLIE* policy derived from Q_t. Let:

$$F_t(s_t,a_t) = F_t^{\text{Sarsa}} + C_t^{VQ} \quad \text{(A.7)}$$

where

$$F_t^{\text{Sarsa}} = r_t + \gamma Q_t(s_{t+1},a_{t+1}) - Q^*(s_t,a_t) \quad \text{(A.8)}$$

$$C_t^{VQ} = \gamma\left[V_t(s_{t+1}) - Q_t(s_{t+1},a_{t+1})\right] \quad \text{(A.9)}$$

F_t^{Sarsa} would correspond to F_t in Lemma 1 if the Q values were updated according to SARSA. To apply Lemma 1 we denote the σ-algebra generated by the random variables $\{s_t, \alpha_t, a_t, r_{t-1}, \ldots, s_1, \alpha_1, a_1, Q_0\}$ by P_t. Note that $Q_t, Q_{t-1}, \ldots, Q_0$ are P_t-measurable and, thus, both Δ_t and F_t are P_t-measurable, satisfying the measurability conditions of Lemma 1.

According to Theorem 1 in Singh et al. [40] we have that:

$$\begin{aligned}\left\| E\left\{F_t^{\text{Sarsa}} \mid P_t\right\}\right\|_\infty &= \|E\{F_t^{\text{Q}} + C_t | P_t\}\|_\infty \\ &= \|E\{F_t^{\text{Q}}|P_t\} + E\{C_t|P_t\}\|_\infty \\ &\le \|E\{F_t^{\text{Q}}|P_t\}\|_\infty + \|E\{C_t|P_t\}\|_\infty \\ &\le \gamma\|\Delta_t\|_\infty + \|E\{C_t|P_t\}\|_\infty \end{aligned} \quad \text{(A.10)}$$

for all t, where $F_t^{\text{Q}} = r_t + \gamma \max_{b\in A} Q_t(s_{t+1}, b) - Q^*(s_t, a_t)$ and $C_t = \gamma[Q_t(s_{t+1}, a_{t+1}) - \max_{b\in A} Q_t(s_{t+1}, b)]$, $\|E\{C_t|P_t\}\|_\infty$ converges to zero w.p.1 under the GLIE policy.

Then we have:

$$\begin{aligned}\|E\{F_t(s_t,a_t)|P_t\}\|_\infty &= \|E\{F_t^{\text{Sarsa}} + C_t^{VQ}|P_t\}\|_\infty \\ &= \|E\{F_t^{\text{Sarsa}}|P_t\} + E\{C_t^{VQ}|P_t\}\|_\infty \\ &\le \|E\{F_t^{\text{Sarsa}}|P_t\}\|_\infty + \|E\{C_t^{VQ}|P_t\}\|_\infty \\ &\le \gamma\|\Delta_t\|_\infty + \|E\{C_t|P_t\}\|_\infty + \|E\{C_t^{VQ}|P_t\}\|_\infty \end{aligned} \quad \text{(A.11)}$$

To satisfy condition 3 of Lemma 1, we are now left with showing $\|E\{C_t^{VQ}|P_t\}\|_\infty$ converges to zero w.p.1, which is satisfied by condition 4 in Theorem 1. An intuitive rationale supporting condition 4 in Theorem 1 is that during each update, both the V function and Q function use the same target value as their supervision; in Appendix A.3 and A.6, we also demonstrate this with empirical results.

Condition 4 in Lemma 1 can be established from the similar property of F_t^{Sarsa}. □

A.2. Evaluation in Section 3.1 without experience replay

We also conduct the same evaluation as in Section 3.1, but without using experience replay. Compared to the results in Fig. 2, similar results are observed in Fig. A.8.

A.3. Evaluation in 2-room grid world

We devise a 2-room grid world to evaluate QV-learning against Q-learning, Double Q-learning, SARSA and Expected SARSA, as shown in Fig. A.9. The grid world has two rooms separated by a wall in the middle while being connected by a door in the wall. The red cell denotes the starting state and overlaps the door. The two green cells denote goals that terminate an episode, the goal state (*Goal*#1) located in the lower room causes a reward sampled from $\mathcal{N}(-0.1, 1)$. The other goal state (*Goal*#2) in the upper room causes a reward of 0.1. Therefore, the optimal policy is to reach *Goal*#2. An ordinary step causes a reward of 0. This 2-room grid world can be viewed as an intensified version of the toy example in Section 3.1, where a single node is now expanded into a room.

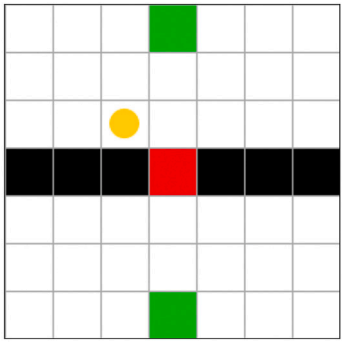
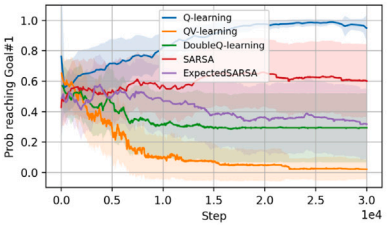
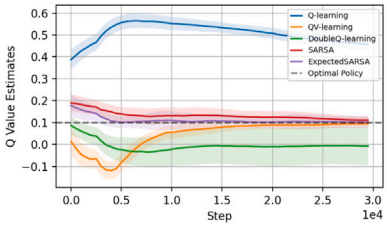


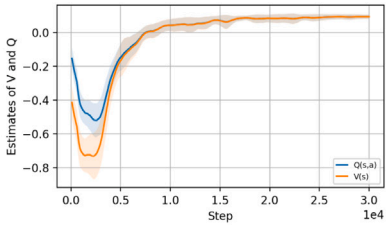
Fig. A.9. 2-room grid world.



(a) Probability of reaching $Goal\#1$. The sooner it drops to zero, the better.

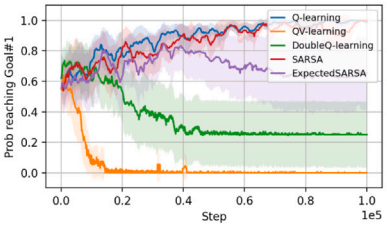


(b) The average Q values over a 100-step window of actions selected using the epsilon-greedy policy, which is derived from the current Q function.

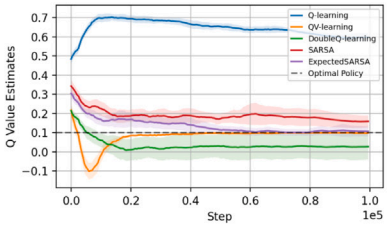


(c) $V(s)$ and $Q(s,a)$ in QV-learning, the discrepancy between two values stay converges to near-zero.

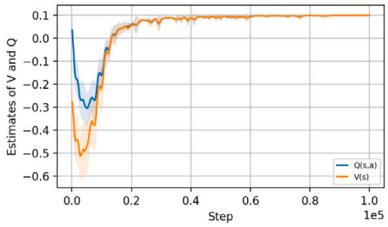
Fig. A.10. Evaluation in 2-room grid world.



(a) Probability of reaching $Goal\#1$. The sooner it drops to zero, the better.



(b) The average Q values over a 100-step window of actions selected using the epsilon-greedy policy, which is derived from the current Q function.



(c) $V(s)$ and $Q(s,a)$ in QV-learning, the discrepancy between two values stay converges to near-zero.

Fig. A.11. Evaluation in 2-room grid world, action space expanded with six additional Do-Nothing actions.

All methods share the same hyperparameter settings: a replay buffer of size 1000, a batch size of 1, a learning horizon of 30,000 steps, a learning rate of 0.1 and an exploration rate that exponentially decays from 0.5 to 0. The results are averaged across 20 repetitions.

In Fig. A.10(a), we evaluate the learning efficiency in convergence to the optimal policy. In Fig. A.10(b), we present a comparison between the estimated Q values and the ground truth value (the straight horizontal red line). The Q value estimates are computed regularly during training with full evaluation phases of length $T = 300$ as:

$$\frac{1}{T}\sum_{t=1}^{T} \underset{a}{\text{argmax}} Q\left(S_t, a\right)$$

The ground truth values are calculated by running the best learned policy and averaging the actual discounted return obtained from each visited state. Without biased value estimation, we would expect these curves to align with the straight horizontal red line on the right side of the figure.

As shown in Fig. A.10(a), QV-learning exhibits the best performance, achieving the near-optimal policy. Q-learning starts with a probability of reaching $Goal\#1$ exceeding 50%, stemming from Q-learning's susceptibility to overestimation (as reflected in Fig. A.10(b). Double Q-learning, SARSA, and Expected SARSA outperform Q-learning; however, they still fail to achieve near-optimal convergence. All baselines employ either maximization or minimization operators during value updates, which inevitably introduce a certain degree of bias into the value estimates.

Fig. A.10(c) illustrates that the discrepancy between $V_t(s_t)$ and $Q_t(s_t, a_t)$ converges to near zero in QV-learning, providing empirical support for the fourth condition in Theorem 1. Action a_t is chosen given state s_t according to the epsilon-greedy policy based on Q_t.

In addition, we make the 2-room grid world evaluation even more challenging by introducing six additional Do-Nothing actions to the action space and extending the total timesteps to 100k. This setting underscores the superior sample efficiency of QV-learning compared to other baselines. As shown in Fig. A.11(a), QV-learning requires less than 20k steps to converge to the optimal policy. In contrast, all the baselines fail to achieve convergence within 100k steps. As presented in Fig. A.11(c), the discrepancy between $V_t(s_t)$ and $Q_t(s_t, a_t)$ converges to near zero in QV-learning, empirically supporting the fourth condition of Theorem 1.

A.4. Evaluation in 3x3 grid world

We also assess QV-learning alongside baseline methods in a 3×3 grid world environment, originally introduced in Double Q-learning by Hasselt [26]. As illustrated in Fig. A.12, each state offers four possible actions, corresponding to the four cardinal directions in which the agent can move. The state marked S denotes the starting position, while G indicates the goal states. Each time the agent selects an action that would move it off the grid, it remains in the same state.

Aligning with the original setting, each non-terminating step causes a random reward of -12 or $+10$ with equal probability. Reaching the goal yields $+5$ and immediately terminates an episode. Exploration follows an ϵ-greedy strategy with $\epsilon(s) = 1/\sqrt{n(s)}$, where $n(s)$ denotes the number of visits to state s. The learning rate is defined as $\alpha = 1/n(s,a)^{0.8}$,

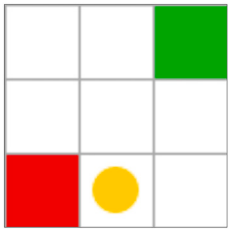


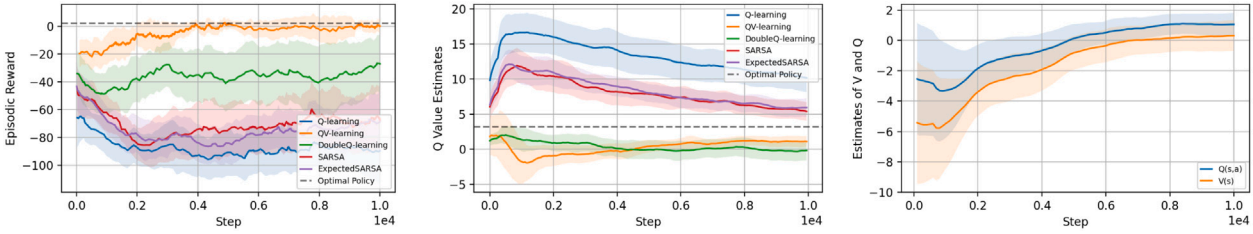
Fig. A.12. 3x3 grid world.

based on the count of state-action visits. Different from the original setting, a replay buffer of size 500 and a batch size of 1 is employed across all methods.

The optimal policy completes the task in 4 actions, achieving an expected episodic reward of 2. The optimal Q value in the starting state is $5\gamma^3 - \sum_{k=0}^{2} \gamma^k \approx 1.43$, with $\gamma = 0.95$. The optimal Q value averaged over all four steps in a trajectory is $1/4 \sum_{n=0}^{3} 5\gamma^n - \sum_{k=0}^{n-1} \gamma \approx 3.19$.

Fig. A.13(a) presents the episodic reward performance, QV-learning exhibits the best performance across all baselines and is the only method that converges to the optimal policy. Note that the average Q-value estimates in QV-learning do not closely approach those of the optimal policy (Fig. A.13(b). This is due to the use of a dynamic learning rate based on state-action visit counts. Since QV-learning converges to the optimal policy within only 4000 steps, the learning rates for state-action pairs along the optimal trajectory quickly decay to near zero. As a result, the Q-function does not fully converge to the optimal Q-function, even though the resulting policy is already optimal. The same reason explains the discrepancy between estimates of V and Q not converging to zero (see Fig. A.13(c).

We also further investigate the learning efficiency by expanding the action space with six additional Do-Nothing actions, keeping all other settings unchanged. As shown in Fig. A.14(a), QV-learning still converges to the optimal policy within 10,000 steps, while other baselines fail to do so. This highlights QV-learning's significant advantage in learning efficiency, particularly in the presence of an enlarged policy search space.

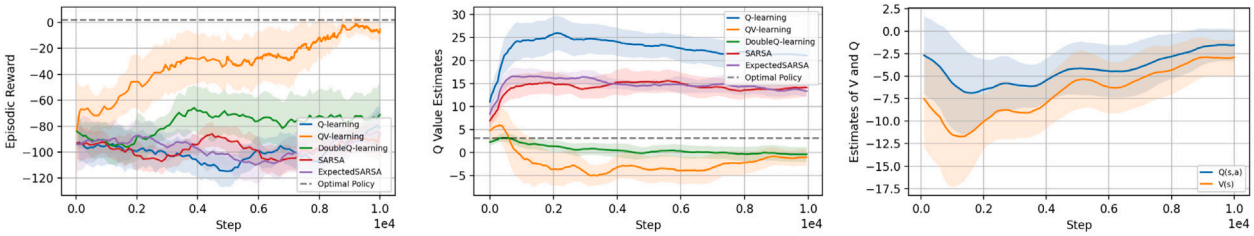


(a) Episodic reward performance. The horizontal dashed line indicates the expected episodic reward of 2 achieved by the optimal policy.

(b) Q-value estimates averaged over steps in an evaluation trajectory. The horizontal dashed line represents the optimal value ($\approx$ 3.19) averaged over steps following the optimal policy.

(c) Estimates of $V(s)$ and $Q(s, a)$ in QV-learning

Fig. A.13. Evaluation in 3x3 grid world.



(a) Episodic reward performance. The horizontal dashed line indicates the expected episodic reward of 2 achieved by the optimal policy.

(b) Q-value estimates averaged over steps in an evaluation trajectory. The horizontal dashed line represents the optimal value ($\approx$ 3.19) averaged over steps following the optimal policy.

(c) Estimates of $V(s)$ and $Q(s, a)$ in QV-learning

Fig. A.14. Evaluation in 3x3 grid world, action space expanded with 6 additional Do-Nothing actions.

A.5. Experimental settings and implementation details for DVQN

In Section 5, we compare our method with baseline methods in ten Atari game domains, namely Asterix, Pong, Riverraid, Breakout, Boxing, Alien, Phoenix, Carnival, DemonAttack and SpaceInvaders. All ten domains share the same settings (as below) for preprocessing observations.

- Number of no-operation actions on reset: 30
- Frameskip (Number of frames skipped between steps): 4
- Framestack: 4
- Max-pooling (Pooling over the most recent two observations from the frame skips): True
- Resize: 84×84
- Grayscale observation: True
- Normalization: [0, 1)

Additionally, before feeding into the networks, the preprocessed observations need to be augmented by random shift, also known as Data-regularized Q [39], which has been testified to be a simple yet effective method to improve sample efficiency and robustness of deep reinforcement learning algorithms. Furthermore, sizes of action space for each domain are listed in Table A.1.

All methods involved in our evaluation share the deep reinforcement learning related hyperparameters listed below:

- Number of warm-up steps with random actions before the start of learning: 10000
- Number of total training steps: 100k/1000k
- Replay buffer size: 100k/1000k
- Batch size: 256
- γ: 0.99
- Exploration rate ϵ: 0.1, decayed by $0.1^{\frac{1}{100k}}$ / $0.1^{\frac{1}{1000k}}$ per step
- Learning rate: 0.0001
- Frequency of updating target networks: 1 step
- Rate of updating target networks τ: 0.001
- Training frequency: 1
- Clip reward: $[-1, 1]$
- Optimizer: RMSprop

The configuration of the Q Net in DVQN is outlined below; the Encoder module is a block of convolutional layers, and the Critic module is an MLP (the size of the output depends on the size of the action space).

Table A.1
Action space sizes for each domain.

Domain	Action Space Size
Asterix	9
Pong	6
Riverraid	18
Breakout	4
Boxing	18
Alien	18
Phoenix	8
Carnival	6
DemonAttack	6
SpaceInvaders	6

```
QNet(
  Encoder(
      (0): Conv2d(4, 32, kernel_size=(8, 8), stride=(4, 4))
      (1): ReLU()
      (2): Conv2d(32, 64, kernel_size=(4, 4), stride=(2, 2))
      (3): ReLU()
      (4): Conv2d(64, 64, kernel_size=(3, 3), stride=(2, 2))
      (5): ReLU()
      (6): Flatten(start_dim=1, end_dim=-1)
  )
  Critic(
      (0): Linear(in_features=1024, out_features=256, bias=True)
      (1): ReLU()
      (2): Linear(in_features=256, out_features=256, bias=True)
      (3): ReLU()
      (4): Linear(in_features=256, out_features=18, bias=True)
  )
)
```

V Net in DVQN shares the Encoder module with Q Net; the Critic module of V Net is detailed as follows:

```
Critic(
    (0): Linear(in_features=1024, out_features=256, bias=True)
    (1): ReLU()
    (2): Linear(in_features=256, out_features=256, bias=True)
    (3): ReLU()
    (4): Linear(in_features=256, out_features=1, bias=True)
)
```

A.6. Discrepancy between v net and q net of DVQN

As shown in Fig. A.15, we monitor the discrepancy between the estimates of $V(s)$ and $Q_1(s, a)$, which remains consistently near zero throughout all Atari domains. This observation provides empirical support for the fourth condition stated in Theorem 1. (s, a) represents batch data sampled from the replay buffer.

A.7. Comparison of DVQN with DQV-learning

Fig. A.16 presents the performance comparison between DVQN and DQV-learning [35] over a horizon of one million timesteps. DQV-learning is equivalent to a variant of DVQN using a single Q-network. DVQN achieves superior or comparable performance across all ten Atari domains.

A.8. Evaluation of DVQN without using data-regularized q augmentation (DrQ)

As shown in Fig. A.17, DVQN achieves significantly better performance than all baselines in six domains, while securing the second-best performance (or tying with other methods) in the remaining four domains.

Notably, Averaged DQN outperforms DVQN in three domains when DrQ is not utilized. Averaged DQN identifies that the overestimation bias of the Q function is also affected by the variance of the Target Approximation Error (TAE) [31], which arises from multiple factors including the sub-optimality of minimization, the limited representation capacity of neural networks and the generalization error for unseen state-action pairs. Hence, it aims at reducing the variance of TAE at the expense of K-fold more forward passes through a Q network. TAE is also inevitable for DVQN, but DVQN does not mitigate the variance of TAE in the same manner as Averaged DQN. Therefore, a plausible speculation is that, when the TAE exhibits overwhelmingly high variance, DVQN may suffer from inferior performance compared to Averaged DQN.

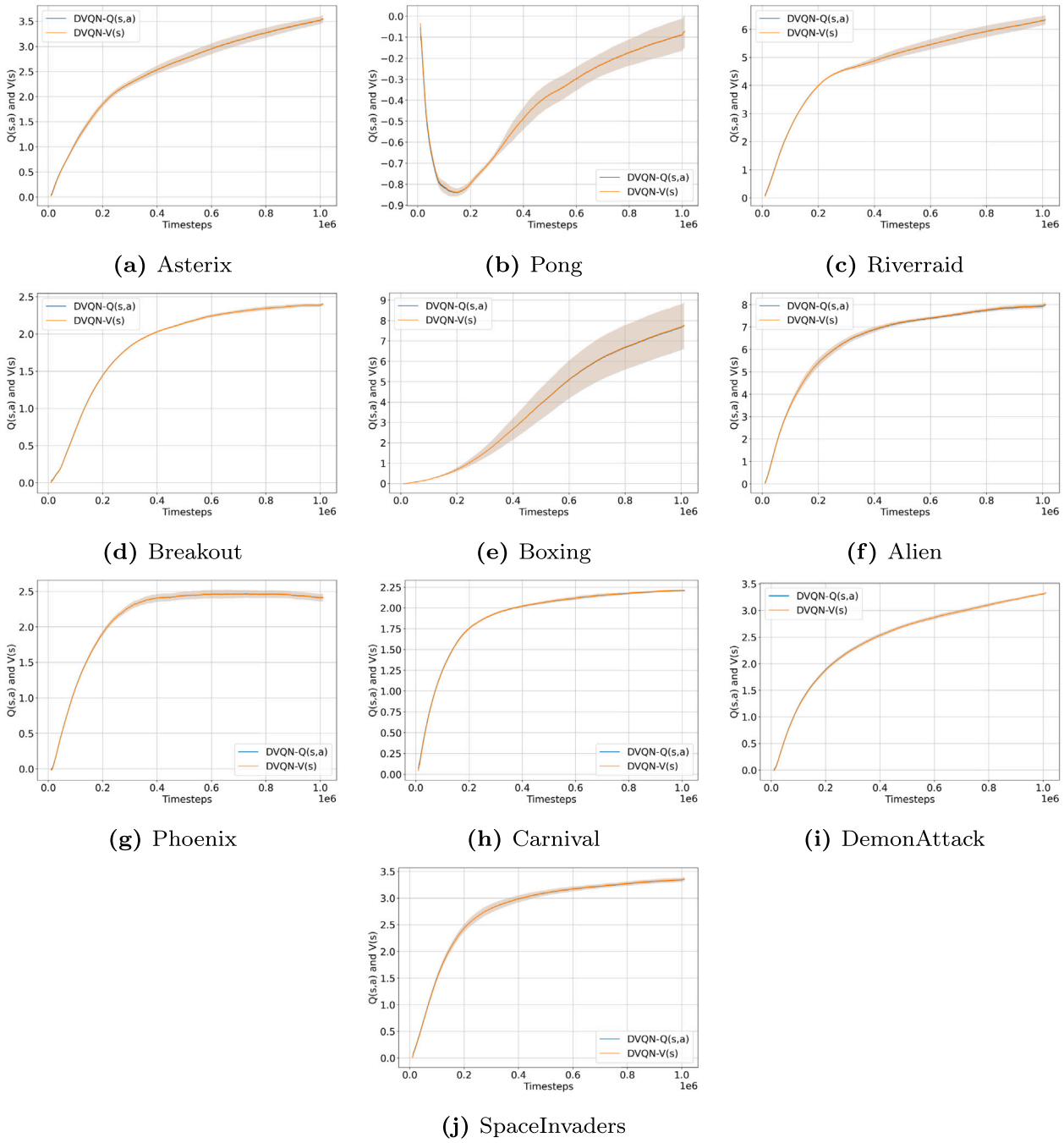


Fig. A.15. $V(s)$ and $Q_1(s,a)$ in DVQN. The discrepancies remain close to zero across all domains.

A.9. Comparison between q value estimates and true values

In this section, we provide a comparison between Q value estimates and true values for DVQN and DDQN. To approximate the true values, we evaluate the policies for both DVQN and DDQN with a fixed interval during training, repeated for 50 times; and average the actual discounted returns obtained from each visited state. Similarly, the Q-value estimates are averaged across all visited states as well. The evaluation is conducted with 1 million training steps.

As illustrated in Fig. A.18, Q value estimates and the corresponding approximated true values are plotted for both methods. In six domains, the discrepancy between the Q-value estimate and the true value is smaller for DVQN compared to DDQN. Notably, DVQN surpasses DDQN in reward performance across all ten domains. We attribute DVQN's superior performance to its unbiased value update procedures. In certain domains, such as Asterix, Phoenix, Carnival, and SpaceInvaders, the reward performance of DVQN improves rapidly, such that the updates

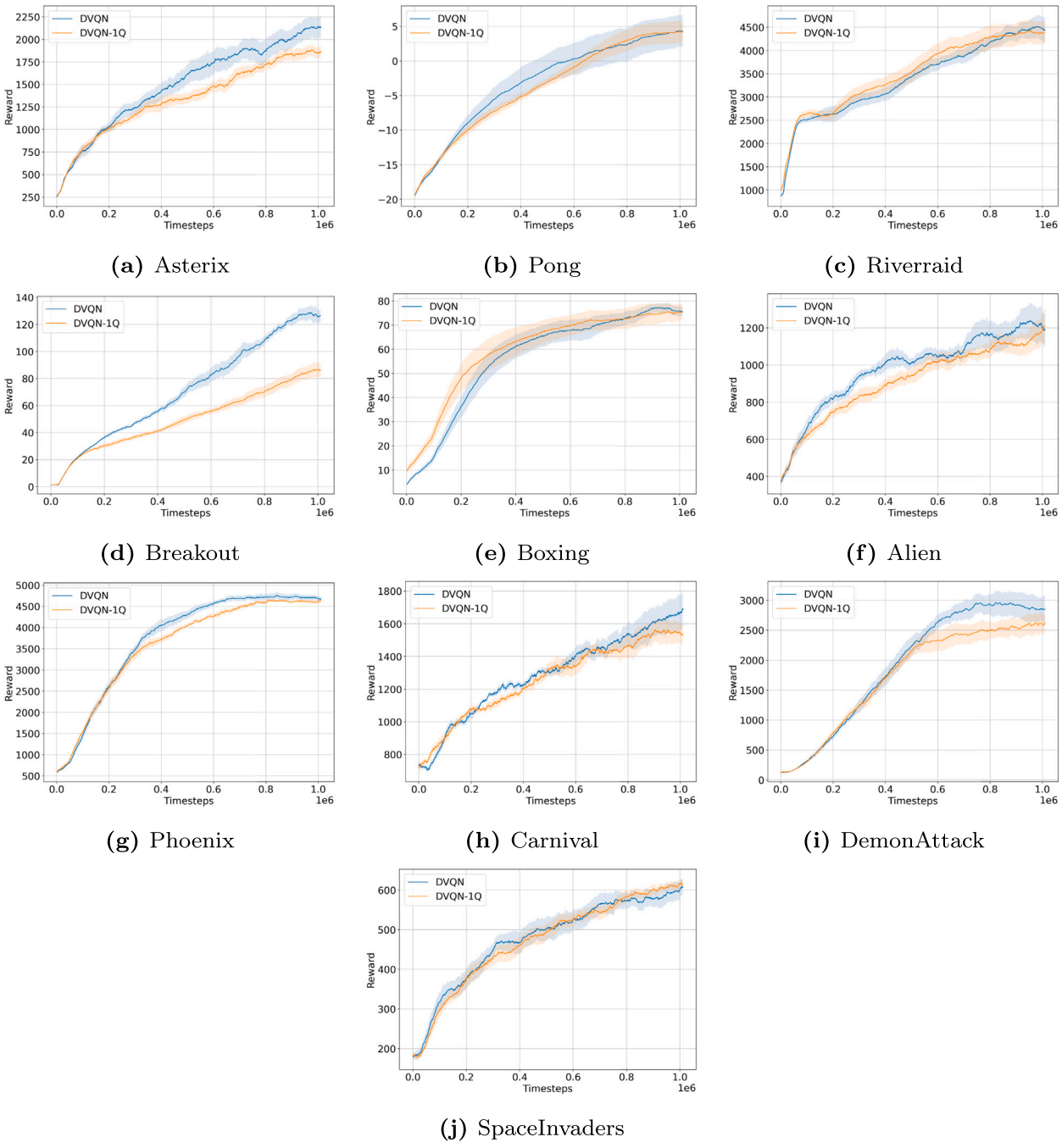
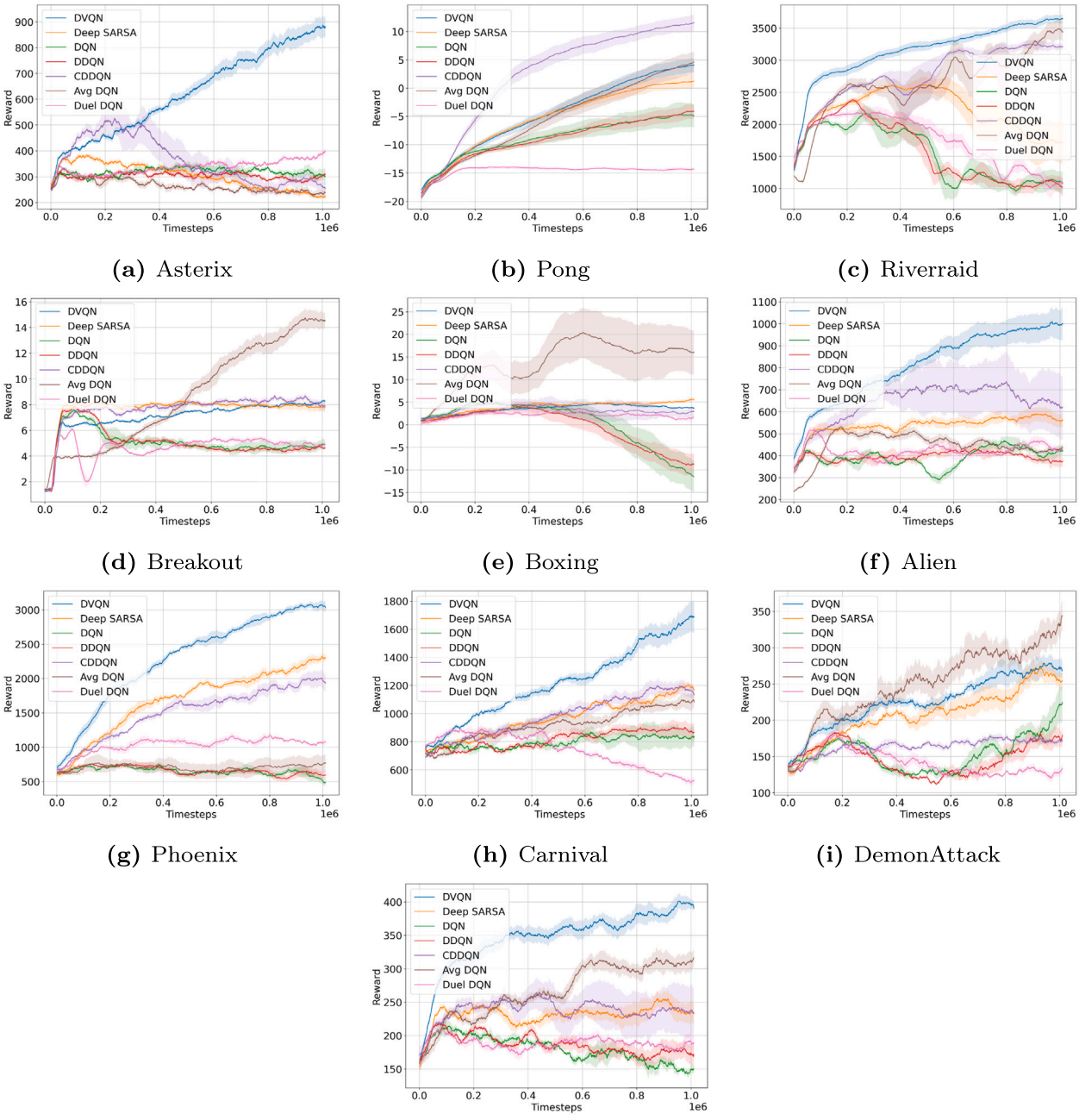


Fig. A.16. Compare DVQN against DQV-learning that is equivalent to a variant of DVQN using a single Q-network.

to Q-value estimates lag behind, even though high-performing policies have already been derived.

A.10. Discussion regarding adaptation of DVQN to continuous control domains

We highlight that, the core design of DVQN can be adapted to methods with actor-critic frameworks, such as TD3 [28] and SAC [30], which are state-of-the-art approaches for continuous control tasks. Notably, TD3 and SAC also utilize the Clipped Double Q-learning (CDDQN) technique to address overestimation of values, it is reasonable to expect that DVQN maintains superior performance on continuous control benchmarks. We believe it will be a valuable endeavor for further research to conduct a extensive evaluation of DVQN (the version adapted for continuous control) and related state-of-the-art baselines on continuous control benchmarks.



(a) Asterix (b) Pong (c) Riverraid

(d) Breakout (e) Boxing (f) Alien

(g) Phoenix (h) Carnival (i) DemonAttack

(j) SpaceInvaders

Fig. A.17. Evaluation of DVQN alongside baselines in reward performance without using DrQ.

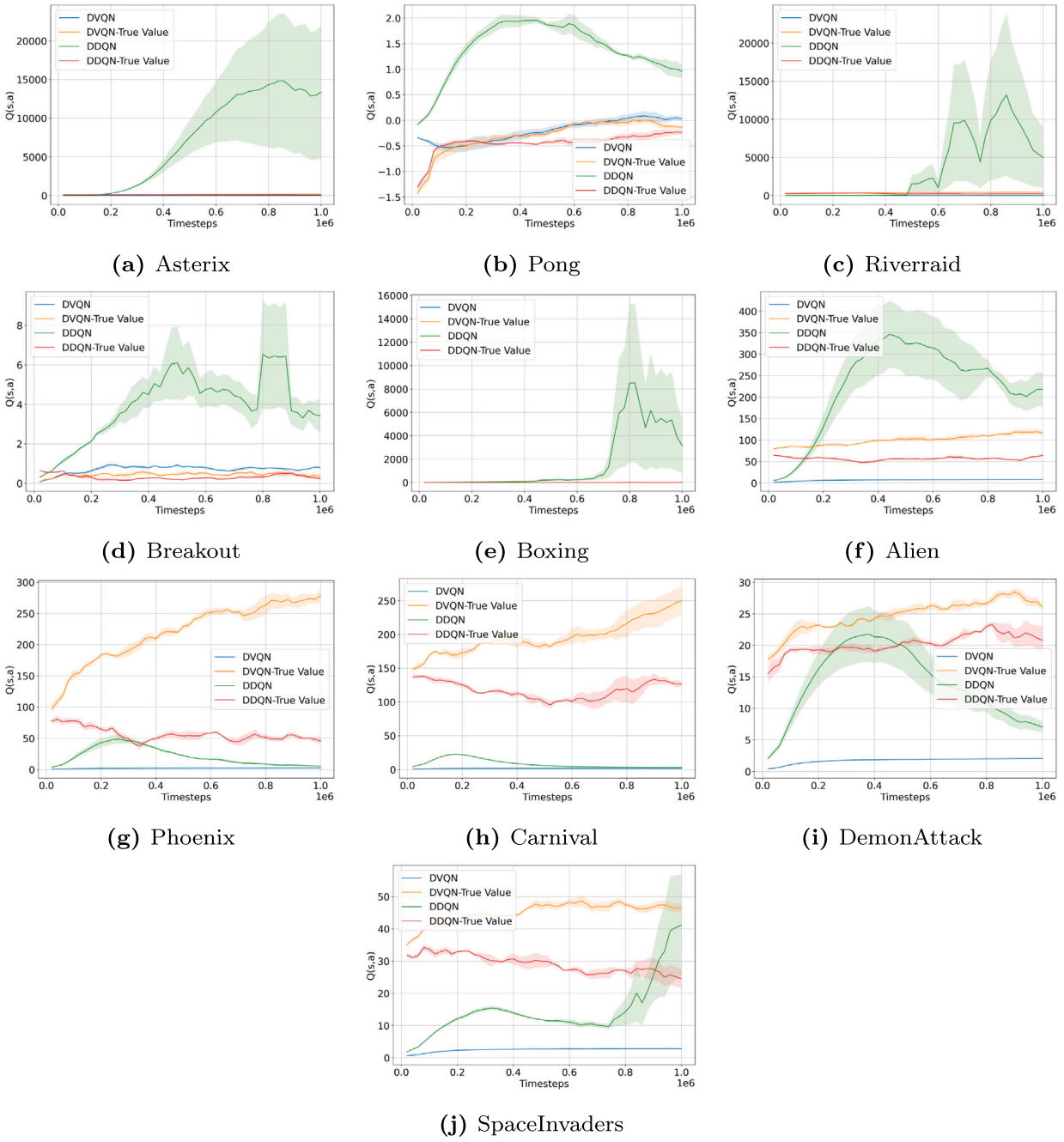


Fig. A.18. Comparison of Q value estimates and true values for DVQN and DDQN.

Data availability

The link to the source code is revealed by the end of Section 1.

References

[1] W. Ye, S. Liu, T. Kurutach, P. Abbeel, Y. Gao, Mastering atari games with limited data, arXiv preprint arXiv:2111.00210, 2021.
[2] V. Micheli, E. Alonso, F. Fleuret, Transformers are sample-efficient world models, arXiv preprint arXiv:2209.00588, 2022.
[3] S. Wang, S. Liu, W. Ye, J. You, Y. Gao, Efficientzero v2: Mastering discrete and continuous control with limited data, arXiv preprint arXiv:2403.00564, 2024.
[4] Q. Li, Y. Lin, Q. Luo, L. Yu, Dreamerv3 for traffic signal control: Hyperparameter tuning and performance, arXiv preprint arXiv:2503.02279, 2025.
[5] A. Kumar, A. Zhou, G. Tucker, S. Levine, Conservative q-learning for offline reinforcement learning, Adv. Neural Inf. Process. Syst. 33 (2020) 1179–1191.
[6] I. Kostrikov, A. Nair, S. Levine, Offline reinforcement learning with implicit q-learning, arXiv preprint arXiv:2110.06169, 2021.
[7] Z. Wang, J.J. Hunt, M. Zhou, Diffusion policies as an expressive policy class for offline reinforcement learning, arXiv preprint arXiv:2208.06193, 2022.
[8] L. He, L. Shen, L. Zhang, J. Tan, X. Wang, Diffcps: Diffusion model based constrained policy search for offline reinforcement learning, arXiv preprint arXiv:2310.05333, 2023.
[9] S.E. Ada, E. Oztop, E. Ugur, Diffusion policies for out-of-distribution generalization in offline reinforcement learning, IEEE Robot. Autom. Lett. 9 (2024) 3116–3123.
[10] S. Sujit, S. Nath, P. Braga, S. Ebrahimi Kahou, Prioritizing samples in reinforcement learning with reducible loss, Adv. Neural Inf. Process. Syst. 36 (2023) 23237–23258.
[11] H. Hassani, S. Nikan, A. Shami, Improved exploration–exploitation trade-off through adaptive prioritized experience replay, Neurocomputing 614 (2025) 128836.
[12] M. Laskin, A. Srinivas, P. Abbeel, Curl: contrastive unsupervised representations for reinforcement learning, in: International Conference on Machine Learning, PMLR, 2020, pp. 5639–5650.

[13] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, R. Fergus, Improving sample efficiency in model-free reinforcement learning from images, arXiv preprint arXiv:1910.01741, 2020.
[14] A. Zhang, R. McAllister, R. Calandra, Y. Gal, S. Levine, Learning invariant representations for reinforcement learning without reconstruction, arXiv preprint arXiv:2006.10742, 2021.
[15] M. Schwarzer, A. Anand, R. Goel, R.D. Hjelm, A. Courville, P. Bachman, Data-efficient reinforcement learning with self-predictive representations, arXiv preprint arXiv:2007.05929, 2020.
[16] M. Kemertas, T. Aumentado-Armstrong, Towards robust bisimulation metric learning, Adv. Neural Inf. Process. Syst. 34 (2021) 4764–4777.
[17] Y. Wang, M. Yang, R. Dong, B. Sun, F. Liu, et al., Efficient potential-based exploration in reinforcement learning using inverse dynamic bisimulation metric, Adv. Neural Inf. Process. Syst. 36 (2023) 38786–38797.
[18] G.C. Forbes, N. Gupta, L. Villalobos-Arias, C.M. Potts, A. Jhala, D.L. Roberts, Potential-based reward shaping for intrinsic motivation, arXiv preprint arXiv:2402.07411, 2024.
[19] M.İ. Bal, H. Aydın, C. İyigün, F. Polat, Potential-based reward shaping using state–space segmentation for efficiency in reinforcement learning, Futur. Gener. Comput. Syst. 157 (2024) 469–484.
[20] M. Li, J. Zhang, E. Bareinboim, Automatic reward shaping from confounded offline data, arXiv preprint arXiv:2505.11478, 2025.
[21] A.Z. Ren, J. Lidard, L.L. Ankile, A. Simeonov, P. Agrawal, A. Majumdar, B. Burchfiel, H. Dai, M. Simchowitz, Diffusion policy policy optimization, arXiv preprint arXiv:2409.00588, 2024.
[22] S. Park, Q. Li, S. Levine, Flow q-learning, arXiv preprint arXiv:2502.02538, 2025.
[23] C.J.C.H. Watkins, Learning from delayed rewards, PhD thesis, King's College, Cambridge, England, 1989.
[24] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, nature 518 (2015) 529–533.
[25] S. Thrun, A. Schwartz, Issues in using function approximation for reinforcement learning, in: Proceedings of the Fourth Connectionist Models Summer School, Hillsdale, NJ, 1993, pp. 263.
[26] H. Hasselt, Double q-learning, Adv. Neural Inf. Process. Syst. 23 (2010).
[27] H. Van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2016.
[28] S. Fujimoto, H. Hoof, D. Meger, Addressing function approximation error in actor-critic methods, in: International Conference on Machine Learning, PMLR, 2018, pp. 1587–1596.
[29] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic policy gradient algorithms, in: International Conference on Machine Learning, Pmlr, 2014, pp. 387–395.
[30] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: International Conference on Machine Learning, PMLR, 2018, pp. 1861–1870.
[31] O. Anschel, N. Baram, N. Shimkin, Averaged-dqn: variance reduction and stabilization for deep reinforcement learning, in: International Conference on Machine Learning, PMLR, 2017, pp. 176–185.
[32] M.A. Wiering, et al., Qv (lambda)-learning: a new on-policy reinforcement learning algrithm, in: Proceedings of the 7th European Workshop on Reinforcement Learning, 2005, pp. 17–18.
[33] K. Ciosek, Q. Vuong, R. Loftin, K. Hofmann, Better exploration with optimistic actor critic, Adv. Neural Inf. Process. Syst. 32 (2019).
[34] A. Ly, R. Dazeley, P. Vamplew, F. Cruz, S. Aryal, Elastic step DQN: a novel multi-step algorithm to alleviate overestimation in deep q-networks, Neurocomputing 576 (2024) 127170.
[35] M. Sabatelli, G. Louppe, P. Geurts, M.A. Wiering, The deep quality-value family of deep reinforcement learning algorithms, in: 2020 International Joint Conference on Neural Networks (IJCNN), IEEE, 2020, pp. 1–8.
[36] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, N. Freitas, Dueling network architectures for deep reinforcement learning, in: International Conference on Machine Learning, PMLR, 2016, pp. 1995–2003.
[37] R.S. Sutton, A.G. Barto, et al., Introduction to Reinforcement Learning, vol. 135, MIT Press Cambridge, 1998.
[38] L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R.H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine, et al., Model-based reinforcement learning for atari, arXiv preprint arXiv:1903.00374, 2019.
[39] I. Kostrikov, D. Yarats, R. Fergus, Image augmentation is all you need: Regularizing deep reinforcement learning from pixels, arXiv preprint arXiv:2004.13649, 2020.
[40] S. Singh, T. Jaakkola, M.L. Littman, C. Szepesvári, Convergence results for single-step on-policy reinforcement-learning algorithms, Mach. Learn. 38 (2000) 287–308.
[41] T. Jaakkola, M. Jordan, S. Singh, Convergence of stochastic iterative dynamic programming algorithms, Adv. Neural Inf. Process. Syst. 6 (1993).

Author biography

Yuan Xue is a PhD student at the L3S Research Center in Hannover, Germany. His primary research focuses on Reinforcement Learning, with an emphasis on improving learning efficiency of RL, abstraction in RL, and cross-domain applications of RL. He also actively investigates generative models and their combinations with RL.

Dr. Daniel Kudenko is research group leader at the L3S Research Center in Hannover, Germany, and Managing Director of the European Digital Innovation Hub for Artificial Intelligence and Cybersecurity (DAISEC). His research focuses on machine learning (especially reinforcement learning), and the foundations and security of generative artificial intelligence, in particular language models. He is also actively engaged in knowledge transfer to industry and the public sector.

Megha Khosla is an Assistant Professor in the Intelligent Systems department at TU Delft, Netherlands. Her main research area is Machine Learning on Graphs with focus on three key aspects of effectiveness, interpretability and privacy-preserving learning. She also actively explores the application of graph representation learning for improving efficiency of RL algorithms.