Regulating Action Value Estimation in Deep Reinforcement Learning

Yuan Xue
L3S Research Centre
Hannover, Germany
xue@l3s.de

Megha Khosla
Intelligent Systems Department, Delft University of Technology
Delft, Netherlands
M.Khosla@tudelft.nl

Daniel Kudenko
L3S Research Centre
Hannover, Germany
Kudenko@l3s.de

ABSTRACT

Deep Q-learning is known to suffer from overestimation of action values, due to the maximization operation when computing the target values. Such overestimation can lead to substantial degradation of reward performance. In this work, we introduce a simple method based on DQN, named as Deep Value Q-learning, which regulates the estimation of action values and effectively tackles over- and underestimation. We evaluate our method on Atari-100k benchmark and demonstrate that DVQN consistently outperforms Deep Q-learning, Deep Double Q-learning and Clipped Deep Double Q-learning in terms of reward performance. Moreover, our experimental results show that DVQN serves as a better backbone network than DQN, when combined with an additional representation learning objective.

KEYWORDS

Deep Reinforcement Learning, Value Overestimation, Contrastive Learning

1 INTRODUCTION

Q-learning [23] has been one of the most widely applied reinforcement learning algorithms. Combined with function approximation through neural networks, deep Q-learning (DQN) [16] shows strong adaptability to complex discrete control tasks. However, Q-learning is known to suffer from overestimation bias [20] [8] due to the maximization operation when computing the temporal difference target. When the overestimation of the Q-function is uniform over all actions, the relative preferences among action values remain unchanged, and therefore the policy remains the same. However, when overestimations are not uniform, they can be detrimental to policy learning [20]. This phenomenon is particularly problematic with deep Q-learning (DQN) [16].

One classic solution to the problem of overestimation is Double Q-learning[8]. It introduces two independent unbiased Q-functions to estimate action values. To update the first Q-function, 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. Double Q-learning is guaranteed to underestimate the maximum expected action values. Deep Double Q-learning (DDQN) [21] extends this idea to the setting of deep Q-learning, in that it treats 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) [4] addresses this concern by taking the minimum between the two independent Q-functions in an actor-critic setting [18]. CDDQN is applied to tackle overestimation in TD3 [4] and SAC [6], 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. In this work, we propose a novel method, Deep Value Q-learning (DVQN), to regulate the estimation of action values. We introduce an unbiased state value function learned by a separate neural network, which can be used to regulate over- and underestimation biases of the action values through a knowledge distillation loss. We demonstrate with experimental results that DVQN outperforms DQN, DDQN and CDDQN in terms of reward performance.

While works like DDQN and CDDQN focused on overcoming the shortcomings of the Q-learning mechanism, several recent methods instead focus on improving reward performance by training better state representations with additional representation learning objectives. For example, Yarats et al. [26] propose using the reconstruction loss as an auxiliary loss alongside the RL loss. Zhang et al. [27] learn an invariant representation without reconstruction. In [15] and [19], momentum contrastive learning objectives are imposed on RL algorithms to improve reward and sample efficiency.

We explore and demonstrate that combining DVQN with an additional objective for representation learning leads to better performance compared to combining such a loss directly with DQN. Our experimental results support that DVQN can serve as a backbone architecture to realize the potential of an additional representation objective. In particular, we apply a temporal contrastive learning objective to the feature extraction module of DVQN. The temporal contrastive objective is constructed such that states that are temporally close to each other share similar representations in a learned latent space, while those that are temporally far from each other have dissimilar representations. The temporal contrastive objective stems naturally from the underlying markov decision process (MDP). Besides, works like [19, 22, 24] also show that the use of the underlying structure of the MDP in learning state representations leads to improvements in convergence and reward performance of RL algorithms in several domains.

We summarize our contributions as follows:

Proc. of the Adaptive and Learning Agents Workshop (ALA 2023), Cruz, Hayes, Wang, Yates (eds.), May 29-30, 2023, London, UK, https://alaworkshop2023.github.io/. 2023.

(1) We propose a novel method (DVQN) to tackle the over- and underestimation bias of action values. DVQN outperforms our baselines in 5 arbitrarily chosen domains of Atari-100k [12].
(2) Our experiments reveal that the temporal contrastive objective employed on DVQN leads to equal or greater performance. However, applying the same objective directly to DQN tends to be non-beneficial or even detrimental.
(3) Our DVQN method is easy to implement and requires little additional computational cost.

We open source our implementation 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 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 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 on state s:

$$Q_\pi(s,a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a, \pi \right] \tag{1}$$

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 Bellmann optimality equation as:

$$Q^*(s,a) = R(s,a) + \gamma \max_{a'} Q_*\left(s', a'\right) \tag{3}$$

Q-learning is a classic model-free RL algorithm to compute $Q^*(s,a)$, which update $Q(s,a)$ iteratively by:

$$Q(s,a) = Q(s,a) + \alpha \left[R\left(s,a,s'\right) + \gamma \max_{a'} Q\left(s',a'\right) - Q(s,a) \right] \tag{4}$$

In Deep Q-learning (DQN) [16], $Q(s,a)$ is approximated by a neural network ϕ, which is iteratively updated by minimizing the loss function below using gradient descent:

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

where (s,a,s',r) is sampled from the replay buffer which stores a fix amount of previous experiences dynamically. $\hat{\phi}$ indicated the target Q network which is synchronized with ϕ by an interval of timesteps, using the target network is crucial to stablize the training of Q_ϕ. Considering the stochasticity and noisy error of the neural network function approximator, DQN is prone to overestimate value estimation when evaluating the selected action a', due to the maximization operator.

To tackle the overestimation, Double DQN (DDQN) [21] takes the target network $Q_{\hat{\phi}}(s,a)$ as another unbiased action value estimator to evaluated a' selected by $Q_\phi(s,a)$, thus its loss function becomes:

$$\mathcal{L}(\phi) = \left[r + \gamma Q_{\hat{\phi}}\left(s', \operatorname{argmax} Q_\phi(s',a')\right) - Q_\phi(s,a) \right]^2 \tag{6}$$

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) [4] improves over DDQN by employing two independent action value estimators: $Q_{\phi_1}(s,a)$ and $Q_{\phi_2}(s,a)$. CDDQN was first introduced in an actor-critic setting, it can be adapted in a critic-only setting by ensuring that the policy which interact 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:

$$\mathcal{L}(\phi_i) = \left[r + \gamma \min_{i \in \{1,2\}} Q_{\phi_i}\left(s', \underset{a'}{\operatorname{argmax}} Q_{\hat{\phi}_1}(s',a')\right) - Q_{\phi_i}(s,a) \right]^2 \tag{7}$$

While CDDQN overcomes the overestimation problem it is still prone to underestimation [2], since the target value is not lower bounded. Improving over both DDQN and CDDQN, our approach tackles simultaneously over- and under-estimations by introducing a novel regularization term based on a value function approximation for states. The estimated value function behaves like a baseline which is used in a knowledge distillation framework to regulate the Q-values.

3 METHODOLOGY

3.1 Deep VQ-learning

To overcome the overestimation problem in deep Q-learning without resorting to underestimation, we propose Deep VQ-learning network (DVQN). We present the pseudocode for DVQN in Algorithm 1. In particular, along with the original Q function $Q_\phi(s,a)$, we learn additionally a state value function $V_\theta(s)$ according to temporal difference and use it to regulated the estimation of $Q_\phi(s,a)$. The rationale behind estimating an addition state value function is that it is not vulnerable to overestimation since unlike in Q-function estimation, no maximization operation is required to calculate the temporal difference target.

Specifically, in lines 13-16 of Algorithm 1, $V_\theta(s)$ is updated according to its own TD-loss, sharing the same data with the update to ϕ . In lines 18-22 of Algorithm 1, we construct the loss with respect to ϕ with two terms: the original Q-learning loss and the difference between $V_\theta(s)$ and $Q_\phi(s,a)$ weighted by an additional factor α.

The former $\mathcal{L}_Q$ ensures the learning is still towards the optimal policy and the latter prevents $Q_\phi(s,a)$ from deviating too far from $V_\theta(s)$. More specifically, when Q_ϕ happens to overestimate on (s_t,a_t) and results in an arbitrarily large y_q (line 18 in Algorithm 1), in comparison, $V_\theta(s_t)$ is more likely to estimate a rational state value, thus $V_\theta(s_t)$ is more promising for $Q_\phi(s_t,a_t)$ to approach in this case. On the other hand, when the target y_q for $Q_\phi(s,a)$ is rationally estimated without overestimation, $V_\theta(s)$ will not deviate too much from $Q_\phi(s, a)$ since the training data is shared between updates of both values function. Moreover, when the $Q_\phi(s, a)$ underestimates due to noisy error of the function approximator, $V_\theta(s)$ plays a role of cross validation to regulate the error.

To clarify more regarding $\mathcal{L}_{VQ}$ (line 20 of Algorithm 1), it only updates parameters of $Q_\phi(s, a)$. This ensures that the parameters of $V_\theta(s)$ will not be polluted by potential overestimation errors of Q values. The form of $\mathcal{L}_{VQ}$ is related to knowledge distillation loss [11] where $Q_\phi(s, a)$ is allowed to distill knowledge from $V_\theta(s)$ by approaching the estimation of $V_\theta(s)$, but not the other way around. The effect of this knowledge distillation can be adjusted by a factor α.

During the training of DVQN the data is collected by the ϵ-policy determined by $Q_\phi(s, a)$, which evolves towards the optimal Q function $Q_{\phi^*}(s, a)$ propelled by the loss term L_Q (line 19 of Algorithm 1). On the other hand, $V_\theta(s)$ is also learned over the same training data and thus $V_\theta(s)$ will also evolve towards the optimal value function $V_{\theta^*}(s)$. Therefore, $V_\theta(s)$ plays a role as a stable baseline for $Q_\phi(s, a)$ to reference along the whole training process.

In practice, as shown in Figure 1, $V_\theta(s)$ and $Q_\phi(s, a)$ are sharing the same feature extractor (CNN), referred to as Encoder. $V_\theta(s')$ and $Q_\phi(s, a)$ are both Multi-Layer Perceptron (MLP) modules. The former has a single output, and the latter has an output size equal to the action space. SG indicates *stop gradient* so that Q Net can approach the output of V net, but not vice versa. As an easy-to-implement extension of DQN, it does not sacrifice sample effiency since updates to V_θ and Q_ϕ use the same data sampled from the replay buffer.

We show that our method outperforms DQN, Double DQN and Clipped Double DQN as baselines in section 4.2.1.

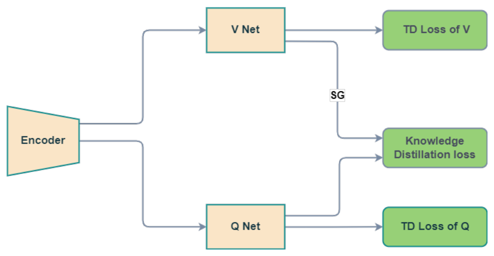


Figure 1: Architecture of DVQN. The encoder is shared between the V Net and the Q Net to improve training efficiency. V Net is learned on its own temporal difference loss, Q Net is learned by minimizing the original Q loss along with the discrepancy between estimations of V Net and Q Net in form of Knowledge Distillation. SG denotes *stop gradient*

.

Algorithm 1 DVQN

```
1:  Initialize replay buffer $\mathcal{D}$
2:  Initialize $Q$ net parameters $\phi$, $V$ net parameters $\theta$
3:  Initialize target networks $\hat{\phi} \leftarrow \phi, \hat{\theta} \leftarrow \theta$
4:  Initialize weighting factor $\alpha$, replay period $n$, update target networks period $m$
5:  Initialize the environment
6:  for $t = 1$ to $T$ do
7:      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 $\mathcal{D}$
10:     if $t$ mod $n$ then
11:         Sample a batch of transitions $B = (s, a, s', r)$ from $\mathcal{D}$
12:         # Compute loss for $V_\theta$:
13:         $y_v(r, s', d) = r + \gamma V_{\hat{\theta}}(s')$
14:         $\mathcal{L}_V = \frac{1}{|B|} \sum_{(s,a,s',r,d)\in B} (V_\theta(s) - y_v(r, s', d))^2$
15:         Update $V_\theta$ by one step of gradient descent:
16:         $\nabla_\theta \mathcal{L}_V$
17:         # Compute loss for $Q_\phi$:
18:         $y_q(r, s', d) = r + \gamma \max_{a'} Q_{\hat{\phi}}(s', a')$
19:         $\mathcal{L}_Q = \frac{1}{|B|} \sum_{(s,a,s',r,d)\in B} \left(Q_\phi(s, a) - y_q(r, s', d)\right)^2$
20:         $\mathcal{L}_{VQ} = \frac{1}{|B|} \sum_{(s,a,s',r,d)\in B} \left(r + \gamma V_\theta(s') - Q_\phi(s, a)\right)^2$
21:         Update $Q_\phi$ by one step of gradient descent:
22:         $\nabla_\phi(\mathcal{L}_Q + \alpha \mathcal{L}_{VQ})$
23:     if $t$ mod $m$ then
24:         Update target netowrks:
25:         $\hat{\phi} \leftarrow \phi$
26:         $\hat{\theta} \leftarrow \theta$
```

3.2 Combine DVQN with Temporal Contrastive Learning

Based on the architecture of DVQN, we impose an additional temporal contrastive learning objective to the feature extraction (encoder) module of DVQN to investigate its effect on reward performance.

Contrastive learning [1, 7, 9, 17] involves generating achor, positive and negative samples in an unsupervised way and ensuring the anchor matches the positive samples more than the negative samples in a latent representation space. In practice, instead of sampling negative examples explicitly, we sample batches of data with anchor-positive pairs, and each anchor takes the positive samples corresponding to other anchors as its negative samples. We denote q_i, k_i as the codes of an achor and its positive sample respectively in a training batch $\mathcal{B}$ and $\mathcal{K} = \{k_1, k_2, k_3 ...\}$. The discrimination objective is in practice achieved by optimizing the InfoNCE loss [17] as:

$$\mathcal{L} = -\mathbb{E}_{q_i \in \mathcal{B}} \left[\log \frac{\exp\left(sim(q_i, k_i)\right)}{\sum_{k_j \in \mathcal{K}} \exp\left(sim(q_i, k_j)\right)} \right] \tag{8}$$

where $sim(q_i, k_i)$ denotes the similarity measured by bilinear product [10]: $q_i^T W k_i$.

We utilize two consecutive states in one transition as an anchor-positive pair to learn a rich temporal-level representation. Note that temporal similarity reflects the topological similarity in the Markov Decision Process describing the task. Hence, such a temporal contrastive learning objective can serve as an ideal task-agnostic auxiliary objective.

Figure 2 illustrates the framework of our temporal contrastive learning. It involes three learnable components: encoder f, projection head g and transformation matrix W. Before input to the encoder, state s_t is first augmented by random shift [13, 14], which

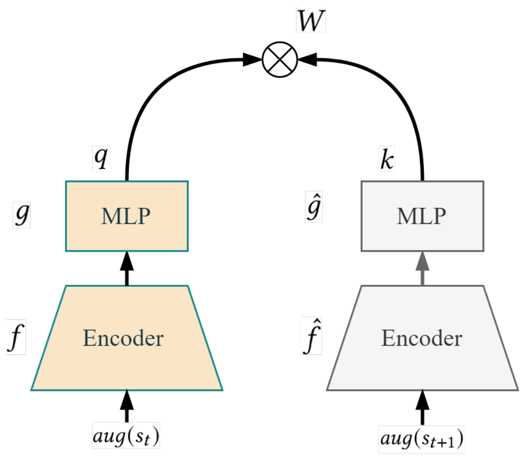


Figure 2: Network structure of temporal contrastive learning, f and g are encoder and projection head respectively, $\hat{f}$ and $\hat{g}$ are momentum conterparts. s_t and s_{t+1} are batch data sampled from the replay buffer, they are preprocess by random shift as data augmentation. Only encoder f is shared with DVQN. g is employed to prevent the output of the encoder from being overwhelmed by the discrimination objective and ignoring task-related knowledge.

has been testified to be a simple yet effective method to improve sample effciency and robustness of RL learning. The usage of projection head g can help avoid features from being tightly clustered [5], called feature collapse. More specifically, without the projection head, features learning tends to be overwhelmed by the contrastive objective and ignore task-related knowledge. Besides, since the data in RL settings is generated and stored dynamically, to stablize the temporal contrastive learning, we applied momentum encoder and momentum projection head [9, 15] to process s_{t+1}. $\hat{f}$ and $\hat{g}$ are synchronized with f and g respectively by a fix interval of timesteps. The encoder f is shared with the encoder in DVQN.

To facilitate a comparative analysis, we also apply the same objective to the encoder of DQN and illustrate with experimental results that the temporal contrastive learning objective applied to our method results in a better reward performance than the one applied to DQN.

The temporal contrasitive objective we use shares similarity with ATC[19] by sampling contrastive data pairs from temporally close states. However, we use projection heads for both anchor and positive samples, whereas ATC only uses one projection head for the anchor sample.

4 EXPERIMENTS AND ANALYSIS

4.1 Experimental Setup

In our experiments, we evaluate our method on Atari 100k[12], which is a sample-constrained benchmark for algorithms dealing with visual (raw pixel) inputs and discrete control. The agent is allowed to interact for 100k steps with the environment, roughly equivalent to 2 hours of gameplay by human.

Due to the complex observation space and constrained amount of data, there has been a number of works that conduct experiments on Atari 100k to evaluate data efficiency. For example, in [12], Kaiser et al. proposed a model-based method which is compared with Rainbow DQN and humen-level performance. CURL [15] and ATC [19] applied momentum contrastive learning to the feature extraction of RL agent to improve sample effcieny on Atari 100k. [13] and [25] achieves effective data efficiency improvement on the same benchmark by applying data augmentation.

Evaluation. Our experimental evaluation is divided into two parts. The first part involves comparing our method to DQN, DDQN, and CDDQN in terms of reward performance and value estimation control. In the second part, we analyze the extent to which our method can benefit when an additional temporal contrastive loss is applied to the feature extraction part of the network. We also compare it when DQN is combined with the same loss. Our experimental results are averaged across 15 random seeds along with confidence interval for one standard deviation.

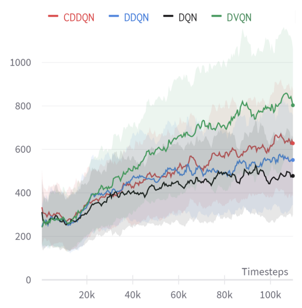
4.2 Result Analysis

4.2.1 DVQN improves over DQN, DDQN, CDDQN. In Figure 3, the performance of the reward against time steps is evaluated. DVQN (our method) outperforms DQN, DDQN, and CDDQN in all 5 Atari domains. This indicates that DVQN regulates the Q-value estimation in a positive way. At the same time, Figure 4 illustrates a comparison of the Q value estimations between DVQN and those of DQN, DDQN and CDDQN. As expected, DQN estimates $Q_\phi(s, a)$ the highest and performs the worst. On the contrary, DVQN consistently suppresses overestimation in all 5 Atari domains by keeping $Q_\phi(s, a)$ not far from the estimation of $V_\theta(s)$. CDDQN estimates higher than DVQN in three domains. In Pong and Breakout, CDDQN estimates lower than DVQN, which is also in our expectation, since the estimation of CDDQN is not lower bounded. However, DVQN is superior to CDDQN in reward performance in all domains, this supports that DVQN can tackle both over- and underestimation. CDDQN tends to be either too aggressive or too conservative when suppressing overestimation, as a result, errors of either overestimation or underestimation are still being backpropagated to some extent. In addition, as discussed in Section 2 that DDQN alleviates the overestimation but still suffers from it, our experiments demonstrate that DDQN tends to be conservative in reducing the overestimation across all domains. As a result, it leads to equal or slightly better reward performance compared with DQN.

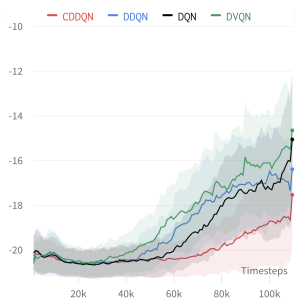
Moreover, DVQN demonstrates consistently lower-level noise in value estimation across all domains when compared to other methods. This characteristic could benefit DVQN, as it achieves the highest reward performance in our evaluations.

Furthermore, DVQN addresses the problem of overestimation on the granularity of individual state-action pairs, while DDQN and CDDQN concentrate on introducing an additional Q function to mitigate the problem on the level of the entire action value function. Additionally, DVQN can also regulate underestimated Q values, which is not possible with DDQN and CDDQN.

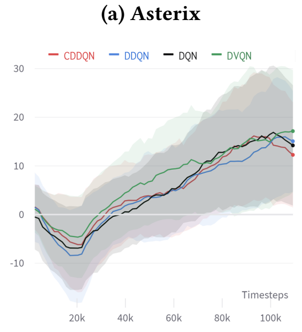
4.2.2 Effect of contrastive learning. Due to the fragility of RL algorithms [3], it is possible that incorporating an extra representation objective into an RL algorithm and learning it end-to-end could



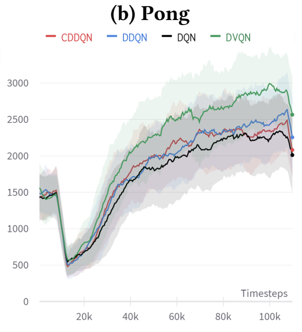
(a) Asterix



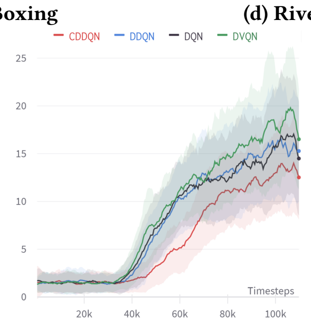
(b) Pong



(c) Boxing

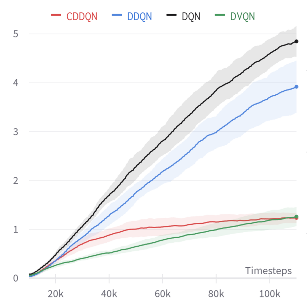


(d) Riverraid

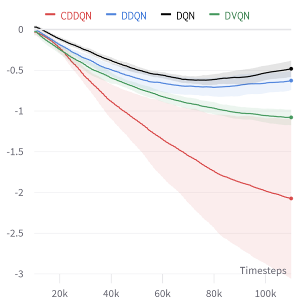


(e) Breakout

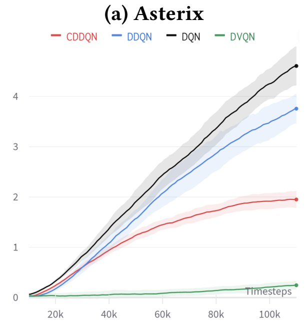
Figure 3: Reward performance against timesteps on 5 atari domains



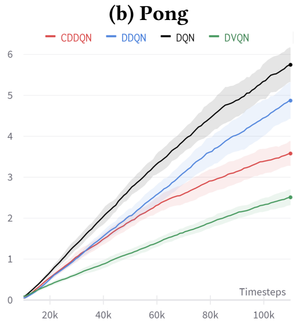
(a) Asterix



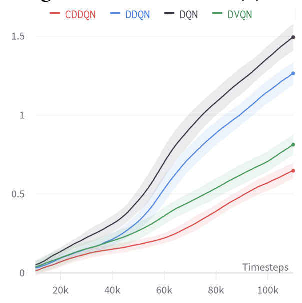
(b) Pong



(c) Boxing



(d) Riverraid

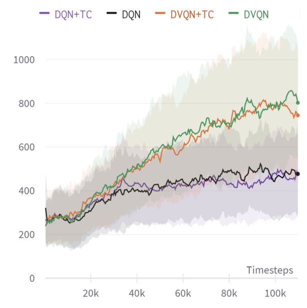


(e) Breakout

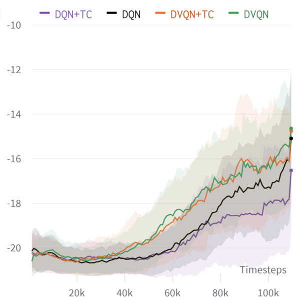
Figure 4: Estimation of Q-value against timesteps on 5 atari domains

have a negative impact. This is also observed in our experimental results. As DVQN helps alleviate the negative impact caused by overestimation or underestimation errors on network updates, we aim to investigate the impact on reward performance when an extra representation learning objective is introduced to DVQN. To be more precise, we utilize DVQN and DQN as the backbone architecture and incorporate an extra temporal contrastive learning objective (TC) to the feature extraction (encoder). The temporal contrastive loss ensures that states that are temporally close to each other should have similar representations in a learned latent space, and those that are far apart should have dissimilar representations. As shown in Figure 5, the combination of DVQN with TC results in improved reward performance in three domains compared to DVQN alone, while maintaining equivalent reward performance in the other two domains. In comparison, combining DQN with TC leads to a significant degradation in reward performance in two domains and no improvement in one domain. Despite enhancing reward performance in the Riverraid and Breakout, it still underperforms DVQN with TC. Based on the presented results, we conclude that DVQN is a better backbone network than DQN for leveraging the potential benefit of TC objective. We attribute it to the fact that DVQN provides a circumstance with reduced noisy error caused by over-and underestimation, thereby allowing for more efficient learning of temporal-level knowledge.

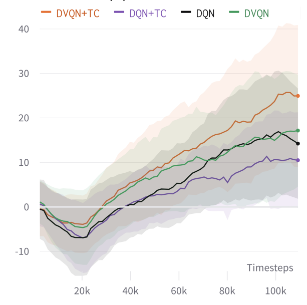
We also conducted experiments incorporating the contrastive objective introduced by Curl [15] to DVQN and DQN. Curl involves using two random augmentations of a single state as a contrastive pair. We find in our results that training DVQN and DQN along with the Curl loss results in inferior reward performance compared to using backbone networks alone. Eventhough DVQN with Curl is superior than DQN with Curl in reward performance.



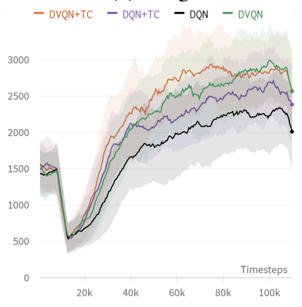
(a) Asterix



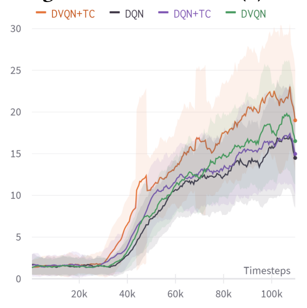
(b) Pong



(c) Boxing



(d) Riverraid



(e) Breakout

Figure 5: Reward evaluation against timesteps of DVQN and DQN combined with temporal contrastive objective

5 CONCLUSION

Our work introduces a new network, DVQN, designed to address the issues of over- and underestimation of action values by learning an additional state value function to regulate the original Q-function in DQN. We provide empirical evidence that our method surpasses DQN, DDQN, and CDDQN in terms of reward performance. Moreover, our experiments incorporating a temporal contrastive loss demonstrate that using DVQN as the backbone network allows for realizing greater potential benefits of the contrastive loss than when using DQN. Our method is easy to implement and increase limited computational cost. In particular, the additional state value function shares the same data with the Q-function to learn, thus it does not sacrifice data efficiency compared with DQN. In the future, it would be worthwhile to explore the adaption of DVQN in actor-critic settings and evaluate its effect.

ACKNOWLEDGMENTS

This research was funded by the Federal Ministry of Education and Research (BMBF), Germany under the project LeibnizKILabor with grant No. 01DD20003.

REFERENCES

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.

[2] Kamil Ciosek, Quan Vuong, Robert Loftin, and Katja Hofmann. 2019. Better exploration with optimistic actor critic. *Advances in Neural Information Processing Systems* 32 (2019).

[3] Benjamin Eysenbach, Tianjun Zhang, Ruslan Salakhutdinov, and Sergey Levine. 2022. Contrastive learning as goal-conditioned reinforcement learning. *arXiv preprint arXiv:2206.07568* (2022).

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

[5] Kartik Gupta, Thalaiyasingam Ajanthan, Anton van den Hengel, and Stephen Gould. 2022. Understanding and Improving the Role of Projection Head in Self-Supervised Learning. *arXiv preprint arXiv:2212.11491* (2022).

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

[7] Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, Vol. 2. IEEE, 1735–1742.

[8] Hado Hasselt. 2010. Double Q-learning. *Advances in neural information processing systems* 23 (2010).

[9] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9729–9738.

[10] Olivier Henaff. 2020. Data-efficient image recognition with contrastive predictive coding. In *International conference on machine learning*. PMLR, 4182–4192.

[11] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[12] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. 2019. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374* (2019).

[13] Ilya Kostrikov, Denis Yarats, and Rob Fergus. 2020. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649* (2020).

[14] Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. 2020. Reinforcement learning with augmented data. *Advances in neural information processing systems* 33 (2020), 19884–19895.

[15] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. 2020. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*. PMLR, 5639–5650.

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

[17] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

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

[19] Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. 2021. Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*. PMLR, 9870–9879.

[20] Sebastian Thrun and Anton Schwartz. 1993. Issues in using function approximation for reinforcement learning. In *Proceedings of the Fourth Connectionist Models Summer School*, Vol. 255. Hillsdale, NJ, 263.

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

[22] Vikram Waradpande, Daniel Kudenko, and Megha Khosla. 2020. Deep reinforcement learning with graph-based state representations. *arXiv preprint arXiv:2004.13965* (2020).

[23] Christopher John Cornish Hellaby Watkins. 1989. Learning from delayed rewards. (1989).

[24] Yuan Xue, Daniel Kudenko, and Megha Khosla. 2023. Graph learning-based generation of abstractions for reinforcement learning. *Neural Computing and Applications* (2023), 1–21.

[25] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. 2021. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645* (2021).

[26] Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. 2021. Improving sample efficiency in model-free reinforcement learning from images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 10674–10681.

[27] Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. 2020. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742* (2020).