Deep Quality-Value (DQV) Learning

Matthia Sabatelli, Gilles Louppe, Pierre Geurts, Marco A. Wiering

m.sabatelli@uliege.be, g.louppe@uliege.be, p.geurts@uliege.be, m.a.wiering@rug.nl

Contributions

We introduce Deep Quality-Value (DQV) Learning, a novel Deep Reinforcement Learning (DRL) algorithm which learns significantly faster and better than Deep Q-Learning and Double Deep Q-Learning. DQV uses temporal-difference learning to train a Value neural network and uses this network for training a second Quality-value network that learns to estimate state-action values.

Preliminaries and Motivation

We consider a set of possible states, $\mathcal{S}$, and a set of possible actions $\mathcal{A}$ that allow an agent to perform a state transition from state s_t at time t to s_{t+1} defined by a transition probability distribution $p(s_{t+1}|s_t, a_t)$. Associated to it there is an immediate reward function, $\Re(s_t, a_t, s_{t+1})$. Actions are taken based on a policy $\pi : s \rightarrow a$. For each state we can calculate its Value V with respect to a discount factor γ:

$$V^{\pi}(s) = E\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \middle| s_t = s\right],$$

whereas values of state-action pairs, $Q^{\pi}(s, a)$ can be maximized with a policy that satisfies

$$Q^{\pi}(s_t, a_t) = \sum_{s_{t+1} \in \mathcal{S}} p(s_{t+1}|s_t, a_t)\Big(\Re(s_t, a_t, s_{t+1}) + \gamma \max_{a_{t+1} \in \mathcal{A}} Q^{\pi}(s_{t+1}, a_{t+1})\Big).$$

Quality-Value QV(λ) Learning is an online tabular RL algorithm proposed in [1] which keeps track of both the Q function and the V function. First V is learned through temporal difference TD(λ) learning and the following update rule:

$$V(s) := V(s) + \alpha\big[r_t + \gamma V(s_{t+1}) - V(s_t)\big]e_t(s).$$

Then these estimates are used to learn the Q function with an update rule which is similar to one step Q-Learning.

$$Q(s_t, a_t) := Q(s_t, a_t) + \alpha\big[r_t + \gamma V(s_{t+1}) - Q(s_t, a_t)\big].$$

Goal: We aim to formulate these update rules as objective functions which can be minimized via gradient descent.

DQV-Learning

We use two neural networks parametrized as Φ and θ respectively, and formulate QV(λ)'s update rules in Mean Squared Error terms. We can now minimize the V function with:

$$L_{\Phi} = E\big[\big(\underbrace{r_t + \gamma V(s_{t+1}, \Phi)}_{y_t} - V(s_t, \Phi)\big)^2\big]$$

and the Q function with:

$$L_{\theta} = E\big[\big(\underbrace{r_t + \gamma V(s_{t+1}, \Phi)}_{y_t} - Q(s_t, a_t, \theta)\big)^2\big].$$

Note that both neural networks use the same target y_t when learning. We additionally improve the stability of DQV with:

- **Experience Replay**: a memory buffer, D, of size N, which stores experiences as $\langle s_t, a_t, r_t, s_{t+1} \rangle$ that we can uniformly sample from $\langle s_t, a_t, r_t, s_{t+1} \rangle \sim U(D)$ for optimizing the neural networks.
- **Target Neural Network:** a separate neural network, parametrized as Φ^- that is specifically designed for estimating the targets, y_t, that are necessary for computing the TD errors. This slightly modifies the original loss functions to:

$$L_{\Phi} = E\big[\big(r_t + \gamma V(s_{t+1}, \Phi^-) - V(s_t, \Phi)\big)^2\big]$$

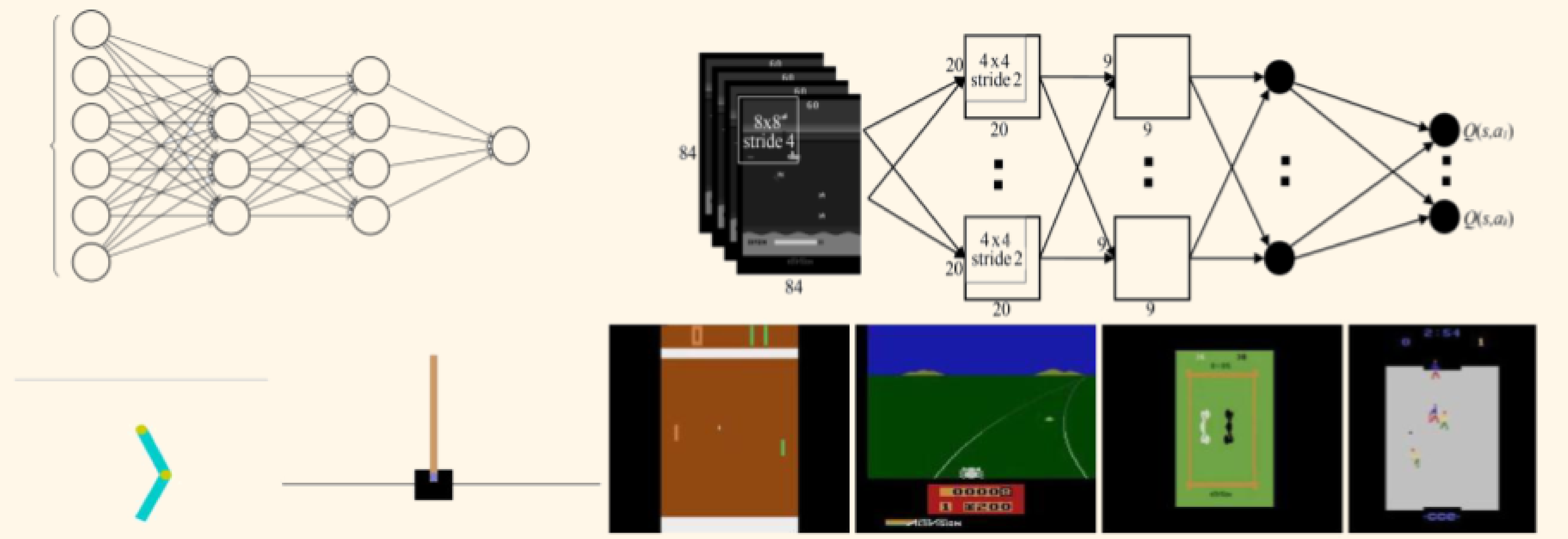
and

$$L_{\theta} = E\big[\big(r_t + \gamma V(s_{t+1}, \Phi^-) - Q(s_t, a_t, \theta)\big)^2\big].$$

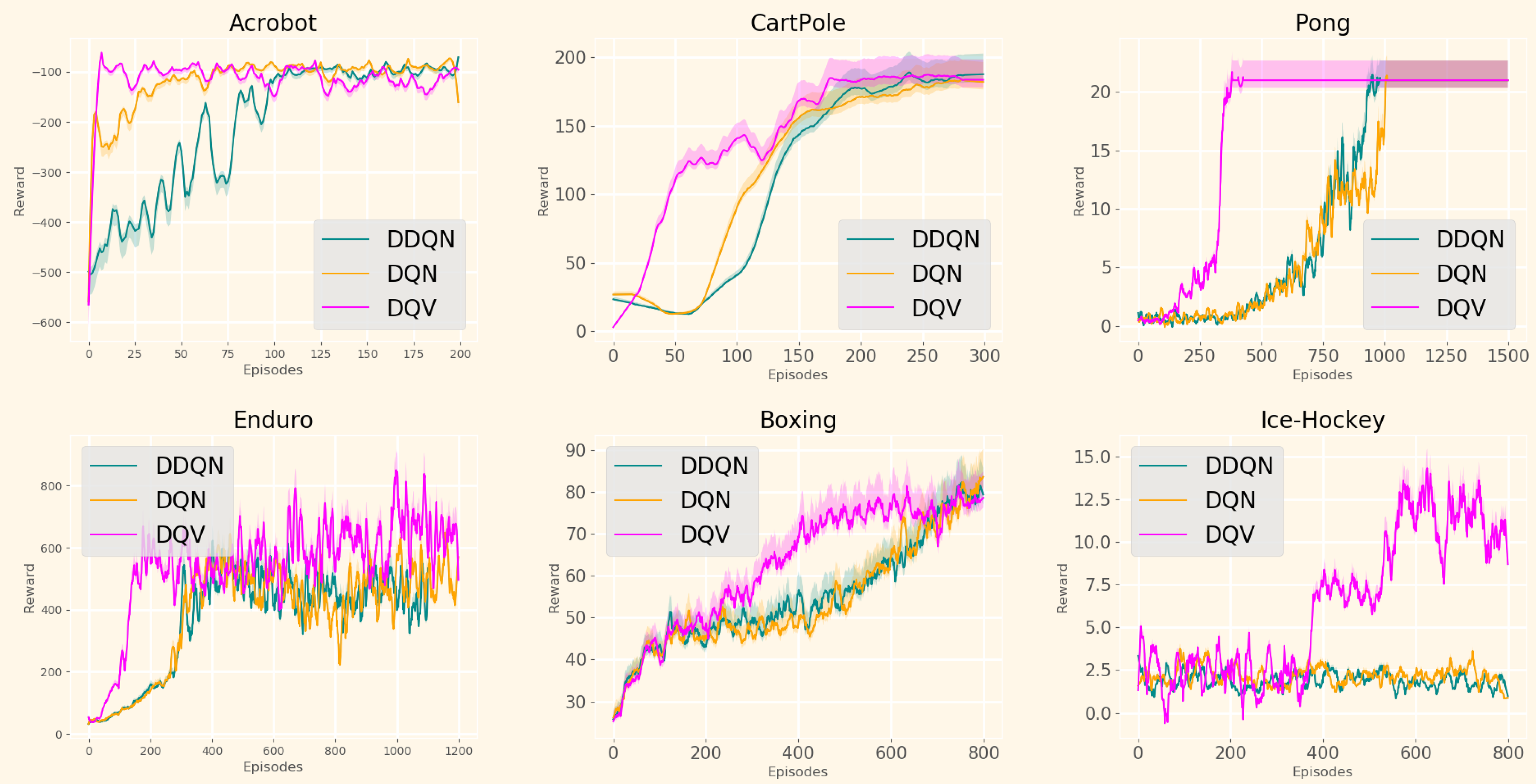
We update the Value-Target-Network Φ^- with the weights of our original Value Network Φ every 10,000 actions as defined by the hyperparameter c.

Experimental Setup

We use Multilayer Perceptrons for approximating the V and the Q function on two classic RL problems, and Deep Convolutional Neural Networks on four Atari games. The environments that have been used are from left to right: *Acrobot, Cartpole, Pong, Enduro, Boxing and Ice-Hockey.*



Results and Discussion



When used in combination with a Multilayer Perceptron on two classic RL problems (Acrobot and Cartpole) DQV learns significantly faster when compared with DQN [2] and DDQN [3]. Similarly, when used in combination with Deep Convolutional Neural Networks, Experience Replay and Target Neural Networks on four Atari games, DQV learns faster on the games *Pong* and *Boxing* while it also yields better results on the games *Enduro* and *Ice-Hockey*.

Acknowledgements

Matthia Sabatelli acknowledges the financial support of the Federal Public Planning Service Science Policy, Belgium, within the BRAIN-be project.

References

[1] Marco A. Wiering QV(λ)-learning: A new on-policy reinforcement learning algorithm.

[2] Mnih et al. Human-level control through deep reinforcement learning.

[3] Van Hasselt et al. Deep Reinforcement Learning with Double Q-Learning.