Deep Q-Network Project

TCSS 435 Artificial Intelligence and Knowledge Acquisition

Huy Huynh | Amanda Mannas | Jayden Fausto | Caleb Carroll

Prepared for Ankur Teredesai, May 7th, 2025, University of Washington Tacoma

Deep Q Network

The Deep Q-Network is an algorithm that is an enhancement of a classic reinforcement learning algorithm, the Q-Learning algorithm, by combining it with DNNs (Deep Neural Networks) and experience replay. The Deep Q network requires a neural network to estimate Q-values. This is often shown by the function

$$Q(s, a; \theta) \approx Q*(s, a)$$

Where the theta represents the weights on the network. Q-learning updates the Q-value by using the Bellman equation which is as follows:

$$Q(s, a) = r + \gamma maxQ(s', a')$$

for all a' where a' represents all possible actions the agent can perform from the next state s'. Where *s* and *a* represent all current possible actions and current state, *r* represents the immediate reward, and γ represents a discount factor where $0 \leq \gamma < 1$.

Deep Q-Networks or DQNs use deep neural networks to generate an approximation of the Q-function:

$$Q(s, a; \theta) \approx r + \gamma maxQ(s', a'; \theta^{-})$$

In this Q-function, inverse theta represents the weights in the target network which is a delayed copy of the main training network which is also called the policy network. Both networks are identical at instantiation and are composed of 3 main components: The input layer, any number of 'hidden layers', and an output layer. The input layer is just the state of the environment, the hidden layers are composed of weights and biases, and the output layer is the action that the neural network decides is optimal.

Chosen Extension

The extension that our group chose to implement was the Deuling DQN extension. The dueling DQN is a model-free reinforcement learning neural network architecture. The

dueling DQN architecture splits into two estimators for state value function and state-dependent action advantage respectively. The dueling DQN architecture still utilizes aspects of standard DQN functions and outputs Q functions for training in a similar way.

In the research paper published by Google DeepMind that was published on April 5th, 2016, in which this Dueling DQN architecture was first written of, the authors claim that this new dueling architecture "leads to better policy evaluation" and allows their reinforcement learning agent to "outperform the state-of-the-art" (Wang et al, 2016). Our findings substantiated their claims of increased performance:

```
Episode 997 | Reward: 286.99 | Return G: 106.49 | Success: 1
Episode 998 | Reward: 296.76 | Return G: 110.76 | Success: 1
Episode 999 | Reward: 283.83 | Return G: 105.82 | Success: 1
Episode 1000 | Reward: 269.42 | Return G: 90.65 | Success: 1

Summary Table (Last 100 Episodes):
                Metric DQN (Vanilla) DQN + Extension
Avg Episodic Reward         46.30          250.60
        Avg Return          46.30          250.60
   Success Rate (%)         9.00%          85.00%
```

Figure 1 Summary Table

The above image is the summary table of the last 100 episodes of our models training. The Dueling DQN architecture gave more than 5 times more episodic reward *and* average return over our original DQN implementation. The improvement in our Success Rate was even better. We saw an increase in Success Rate of over *nine* times going from the vanilla DQN architecture to the Dueling DQN implementation.

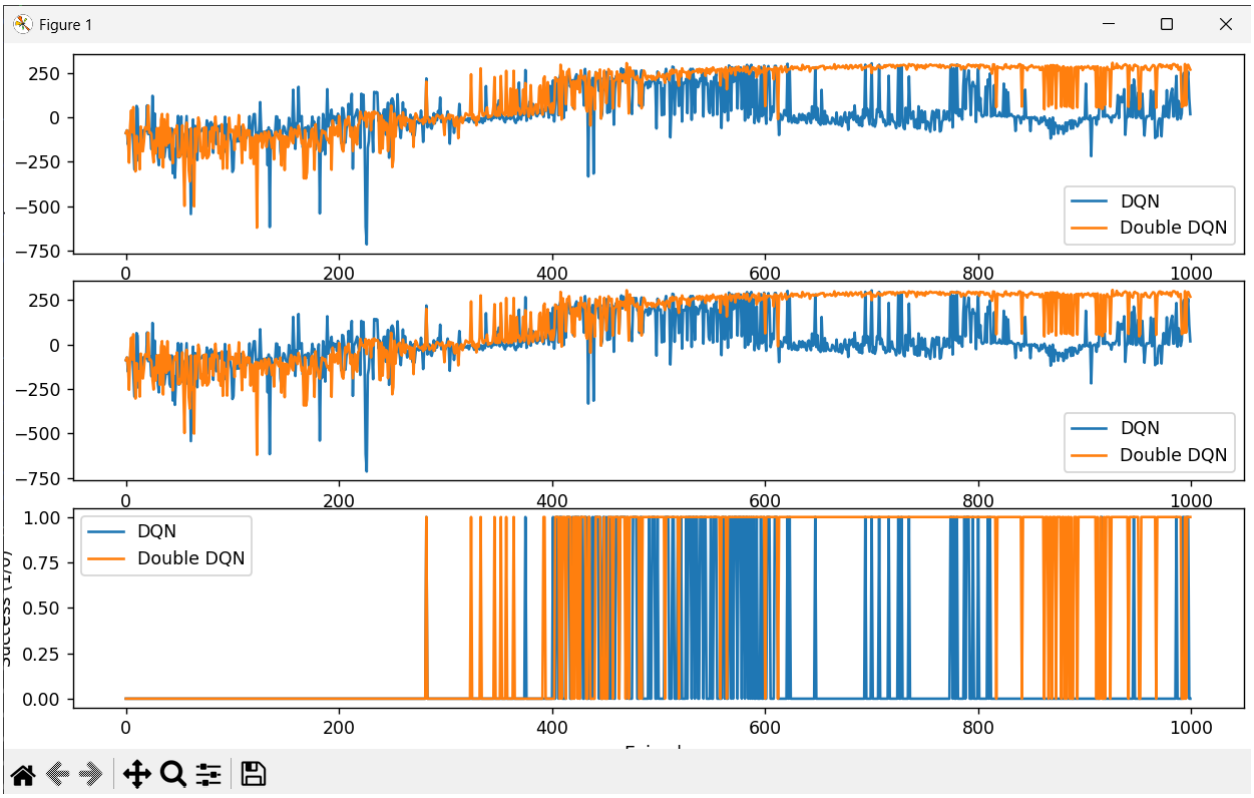

Figure 2 Performance Charts, (label 'Double DQN' meant to be Dueling DQN)

The above image here shows three different charts of which our vanilla DQN is drawn in blue, and our Dueling DQN is drawn in red. On top is Episodic Reward vs episode number, in the middle is Episodic return vs Episode number and finally on the bottom we have success rate of each implementation over time.

It is easy to see that in the later stages of training the Dueling DQN architecture outperforms our original implementation to a significant degree in all three measured metrics. This increase in performance starts to show after around episode 450 and as the training gets to episode 500, the Dueling DQN implementation takes a definitive lead in performance until the end of our 100 episodes. It also seems to experience a 100% success rate at a faster rate than the standard DQN implementation with its first success occurring around episode 230 as seen in the bottom graph of figure 2.

Contributions

Jayden started us off by setting up the GitHub Repository following this, each of the team members did a part of the project. Jayden did the Vanilla Deep Q-Network Implementation, Huy did the Dueling DQN extension implementation, and Amanda did the Evaluation and Analysis. Caleb wrote the report for the project and the readme on the GitHub repository.

GitHub Repository: https://github.com/NinjaPanda351/DQN_LL

Citation: https://arxiv.org/abs/1511.06581v3