Reinforcement Learning

Coursework 2

Imperial College London

Department of Bioengineering
MEng Biomedical Engineering

Nabeel Azuhar Mohammed

CID: 01701146

November, 2023

1 Tuning the DQN

1.1 Hyperparameters

Many hyperparameters of the DQN had to be tuned to ensure high performance and good learning efficiency in the Cartpole environment. Keeping in mind that sensible values for the hyperparameters have to chosen to complete the training in 300 episodes and not be too computationally demanding, the following values for the hyperparameters were explored (Table 1).

Hyperparameters	**Values explored**
Discount factor (γ)	[0.5, 0.8, 0.9, 0.99, 1]
Exploration rate (ϵ) start value	[0.8, 0.9, 1]
Exploration decay rate	[0.9, 0.95, 0.99, 0.992, 0.995, 0.999]
Replay buffer size	[5000, 10000, 15000, 20000]
Batch size	[16, 32, 64, 128]
Neural network architecture	[{4, 128, 2}, {4, 16, 16, 2}, {4, 32, 32, 2}, {4, 64, 64, 2}, {4, 128, 128, 2}]
Target network update frequency	[2, 4, 8, 16, 32, 64]
Optimisation frequency	[1, 2, 4, 8, 16]
Optimiser choice	[SGD, Adam]
Learning rate	[0.002, 0.001, 0.0005]
Loss function	[MSE, Huber]
Activation function	[ReLU, Leaky ReLU]

Table 1: Hyperparameter values that were explored.

Compared to other hyperparameters, the optimiser choice, loss function and activation functions were not explored as much. This is because the Cartpole environment is a relatively simple environment and changing these hyperparameters do not have a significant impact in the learning of the agent. The other hyperparameters had to be more carefully tuned to ensure that the agent learns fast, with a good stability and does not converge to a sub-optimal policy.

In the exploration, an exploration rate (ϵ) end value was not specified as this was controlled by setting the ϵ start value and decay rate to appropriate values such that the desired ϵ end value was achieved.

The hyperparameters that optimised the learning curve are as shown in Table 2.

Hyperparameter	Value	Justification
Discount factor (γ)	0.99	Represents the weight given to future rewards relative to immediate rewards. Multiple values of γ were explored but it made no noticeable difference to the performance of the agent. Regardless, this value was chosen to ensure that the agent prioritises future rewards and develops a suitable long term strategy while still giving importance to the immediate reward.
Exploration rate (ϵ) start value	1	Denotes the balance between exploration and exploitation. High exploration is initially required so that the agent experiences the consequences of more state-action pairs and can exploit better in the future.
Exploration decay rate	Linear (0.992)	Determines how fast ϵ will decay. It ensures that exploration is done for half of each run so that more state-action pairs can be explored by the agent and it can later be exploited to achieve higher returns. It also causes ϵ to approach 0.1 at the end of each run, thus allowing the agent to still explore higher return state-action pairs while predominantly acting greedily.
Replay buffer size	20000	Stores previous experiences that the agent can randomly sample from and learn from. This buffer size strikes a balance between allowing the agent to store and sample a suitably diverse range of experiences without being too exhaustive and demanding too much memory resources. A smaller buffer size would reduce the diversity of experiences while bigger sizes would reduce the memory efficiency.
Batch size	32	Data samples used to optimise and update the neural network. It is large enough to ensure stability by offering a smoother gradient estimation while still being small enough to be computationally efficient. It therefore strikes a good balance between learning efficiency and computational efficiency.
Neural network architecture	{4, 64, 64, 2}	Defines the structure and arrangement of neurons and layers within the neural network. This architecture is large enough to capture the nuances in the Cartpole environment without being overly complex. The chosen number of layers and neurons allow for a better representation of the state space and allows for the handling of non-linear transformations without causing overfitting or being too computationally demanding.

Target network update frequency	4	Determines how often the target network is updated using the Q-network. While updating it every 4 time steps can introduce some instability in the learning, we require the agent to quickly adapt to the environment and learn from its recent experiences so that it can perform as well as it could in a 300 episode run.
Optimisation frequency	4	Determines how often the Q-network is optimised. While updating it every 4 steps runs the risk of overfitting and may be a bit more computationally demanding, it allows the agent to quickly learn about the environment and it leads to faster convergence.
Optimizer choice and parameters	Adam	Adjusts the weights of the neural network to minimise the loss function. Adam was preferred over SGD because it converges faster owing to its ability to adapt the learning rates for each parameter based on the magnitudes of the gradients throughout the training process. This also allows it to be less sensitive to hyperparameters and therefore deliver a more robust performance.
Learning rate (α)	0.001	Determines the size of steps taken during the optimisation process. This moderate value strikes a balance between Adam's adaptability to different parameter updates and its stability during the training process. A lower value would increase stability but converge slower and be more sensitive to hyperparameters while a higher value would do the opposite.
Loss function	Mean Squared Error (MSE)	Quantifies the difference between predicted and actual values. MSE penalises larger errors more than smaller ones, leading to sharper updates in Q-values if there are any significant errors, thus allowing for a faster convergence than the Huber loss functions. The benefit of the other loss functions over MSE is their robustness to outliers but due to the simplicity of the environment, this functionality is not required.
Activation function	Leaky ReLU	Determines the output of a neuron by transforming the weighted sum of its inputs. Leaky ReLU was chosen over ReLU because it mitigates the issue of neurons becoming inactive due to negative inputs (dying ReLU problem) by allowing a small gradient for these inputs and thus preventing the neurons from becoming inactive. While this issue does not occur frequently in this environment due to its simplicity, this functionality does allow for a more robust performance.

Table 2: Hyperparameters for tuning a DQN for the Cartpole environment.

To effectively balance exploration with exploitation, it was important to consider the effects of ϵ and the ϵ decay. The ϵ and the ϵ decay hyperparameters were tuned in such a way that the agent explored for almost half the episode length and then began exploiting so that the agent could converge onto higher returns. These hyperparameters were also tuned to ensure that ϵ did not drop below 0.1, hence allowing the agent to act greedily to achieve more steps in an episode while allowing it to also explore a little at states that it would only reach in the later stages of the episodes.

The buffer uses a First-In-First-Out policy and therefore it had to be large enough to allow older experiences to be stored and sampled and this helps to reduce the variance in the returns. Both the buffer and batch sizes were chosen to be large enough so that the agent can sample many diverse experiences and optimise its policy better to converge onto higher returns. It is also important to note that the sampling from the buffer is being done randomly to allow for a more diverse sample.

1.2 Learning Curve

The learning curve produced through training the agent for 10 runs each containing 300 episodes is shown in Figure 1.

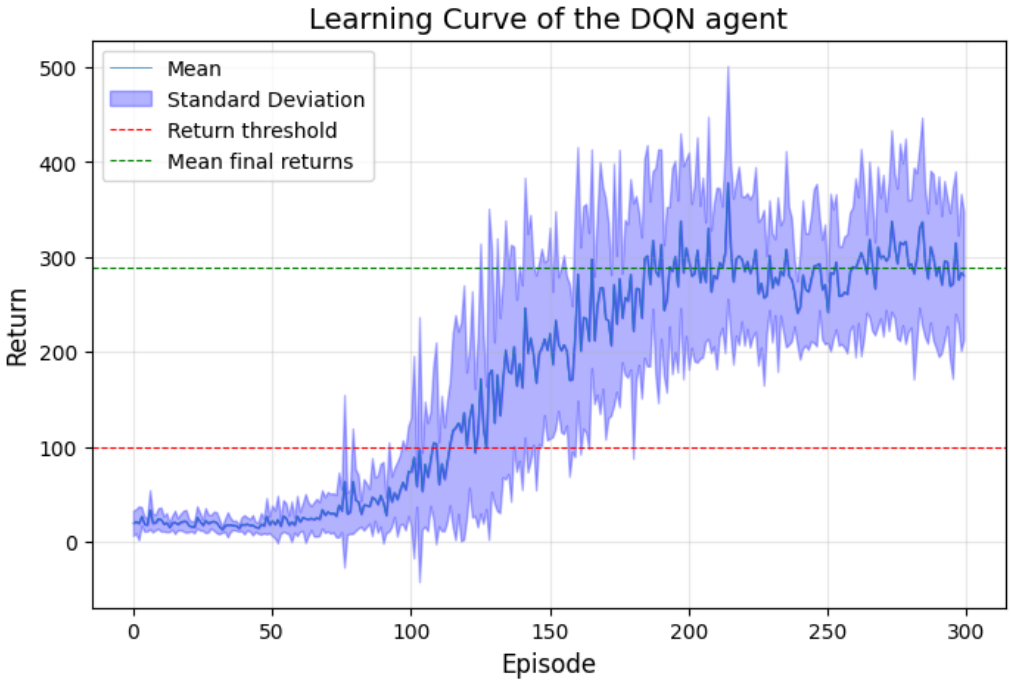

Figure 1: Learning curve of the DQN agent over 10 runs of 300 episodes each.

The agent begins learning after 50 episodes as the gradient of the learning curve begins to increase and the learning curve crosses the return threshold value of 100 within 120 episodes. The learning speed is moderate as it learns for another 150 episodes before it converges at a return of 289. The convergence speed is fairly fast considering that it achieves its optimal policy in the space of 200 episodes. When observing the mean values of the returns, we notice that there are not many huge swings in the values of the mean and therefore we can discern that the learning was fairly stable. Considering that the learning curve is stable, converged

fast, learnt at a moderate speed and does not have a big standard deviation between runs, we can conclude that the hyperparameters were appropriately chosen and the agent has converged onto the optimal policy that is achievable after 300 episodes.

2 Visualising the DQN policy

2.1 Slices of greedy policy action

In the Cartpole environment, there are 4 variables that describe the state space - cart position (x), cart velocity (v), pole angle (θ) and pole angular velocity (ω). Moving right and moving clockwise are defined as the positive directions. Depending on the state of the environment, the agent chooses action 0 (move cart to the left) and action 1 (move cart to the right) to stabilise the pole between angles of -0.2094 to 0.2094 radians from the upright pole position.

When $v = 0$, no torque is being induced by the cart. The cart dynamics are explained through Table 3.

	$\theta < \mathbf{0}$	$\theta > \mathbf{0}$
$\omega > \mathbf{0}$	The pole is to the left of the center and turning clockwise. Depending on the magnitude of θ and ω, the agent will choose to perform action 0 to induce a positive torque to bring θ closer to 0 faster or action 1 to induce a negative torque to negate the clockwise movement. These actions will help stabilise the pole.	The pole is to the right of the center and falling right. Hence, the agent chooses action 1 to induce a negative torque and stabilise the pole.
$\omega < \mathbf{0}$	The pole is to the left of the center and falling left. Hence, the agent chooses action 0 to induce a positive torque and stabilise the pole.	The pole is to the right of the center and turning anti-clockwise. Depending on the magnitude of θ and ω, the agent will choose to perform action 0 to induce a positive torque to negate the anti-clockwise movement or action 1 to induce a negative torque to further reduce the value of θ. These actions will help stabilise the pole.

Table 3: Policy description when cart velocity is zero ($v = 0$)

Therefore, when $v = 0$, we observe a moderately steep negative gradient separation line between the two actions (Figure 2. This separation line can be modelled by the following function:

$$\omega = m\theta + \frac{v}{L}$$

where m is gradient of the separation line and L is the length of the pole. As v increases, the rightwards movement of the cart generates a counter-clockwise torque on the pole and that is

depicted by the constant term in the equation. Therefore, we would expect the separation line to shift to the left and that is what we observe in the policy (Figure 2).

As v increases, the cartpole system has more kinetic energy and according to the conservation of energy and conservation of angular momentum law, this energy can transfer into the pole and intensifying its angular momentum. Additionally, the higher kinetic energy in the system also amplifies the inertial effect of the pole. The combination of the two makes it challenging to maintain an equilibrium in the system as higher cart velocities will magnify the pole's resistance to changes in motion and applying action 0 in most situations (except for highly negative pole angles and pole angular velocities) could cause the angular momentum to dramatically increase and lead to the falling of the pole. Therefore, action 1 starts becoming the better action to perform.

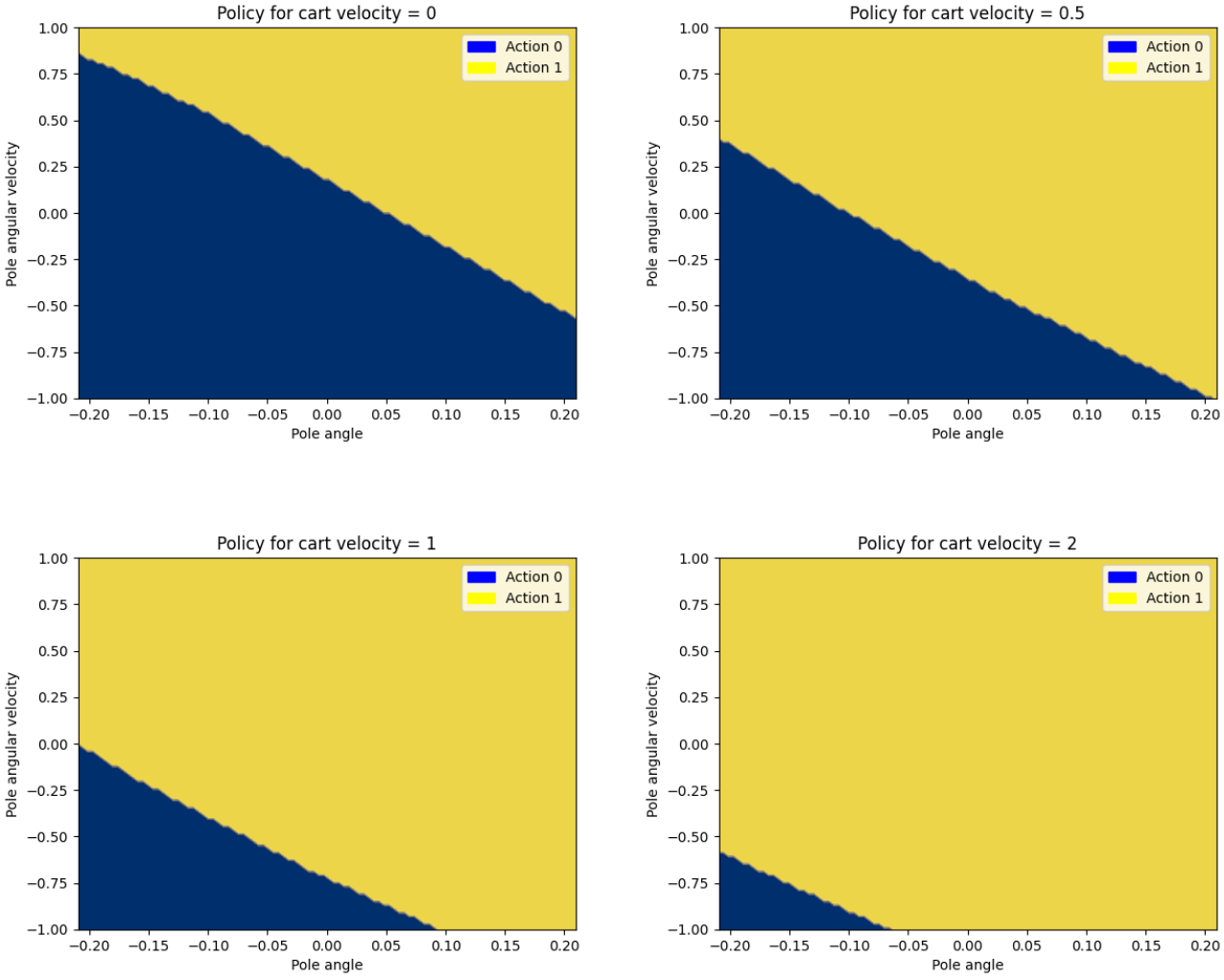

Figure 2: Policy generated by the DQN agents.

2.2 Slices of the Q function

The Q-values of the state space represents the cumulative expected returns an agent expects to receive when at those particular states. The Q-values for different velocities are depicted in Figure 3. Areas with a higher Q-value (in yellow) are states that the agent prefers to be in so that it can exploit higher returns whereas areas with low Q-values (in blue) are states that the agent does not expect to perform well in and it will try to avoid these states during training.

When $v = 0$, we observe that the yellow area is presented in the same area as the separation line that we observed in the policy figures (Figure 2). This is due to the equilibrium of the system being situated in those states and therefore, the agent has the highest chance of lasting longer in each episode if it manages to remain in that area. The blue areas are shown in the bottom left and top right corners of the plot because it is at these states that the pole has the highest risk of falling down.

As v increases, the higher moment of inertia and induced counter-clockwise angular momentum shifts the yellow area to the bottom left of the graph. Again, it reflects the movement of the separation line seen in the policy plots. As the system equilibrium moves to these states, the agent will attempt to enter these states in order to maximise its returns. The top right corner of the plot becomes the dangerous states to be in and the agent will try to avoid taking actions that enter those states. Additionally, we can also observe that as v increases, the maximum Q-values achievable also decrease. This is due to the system dynamics becoming more complex and it becoming harder to remain in the equilibrium states for long.

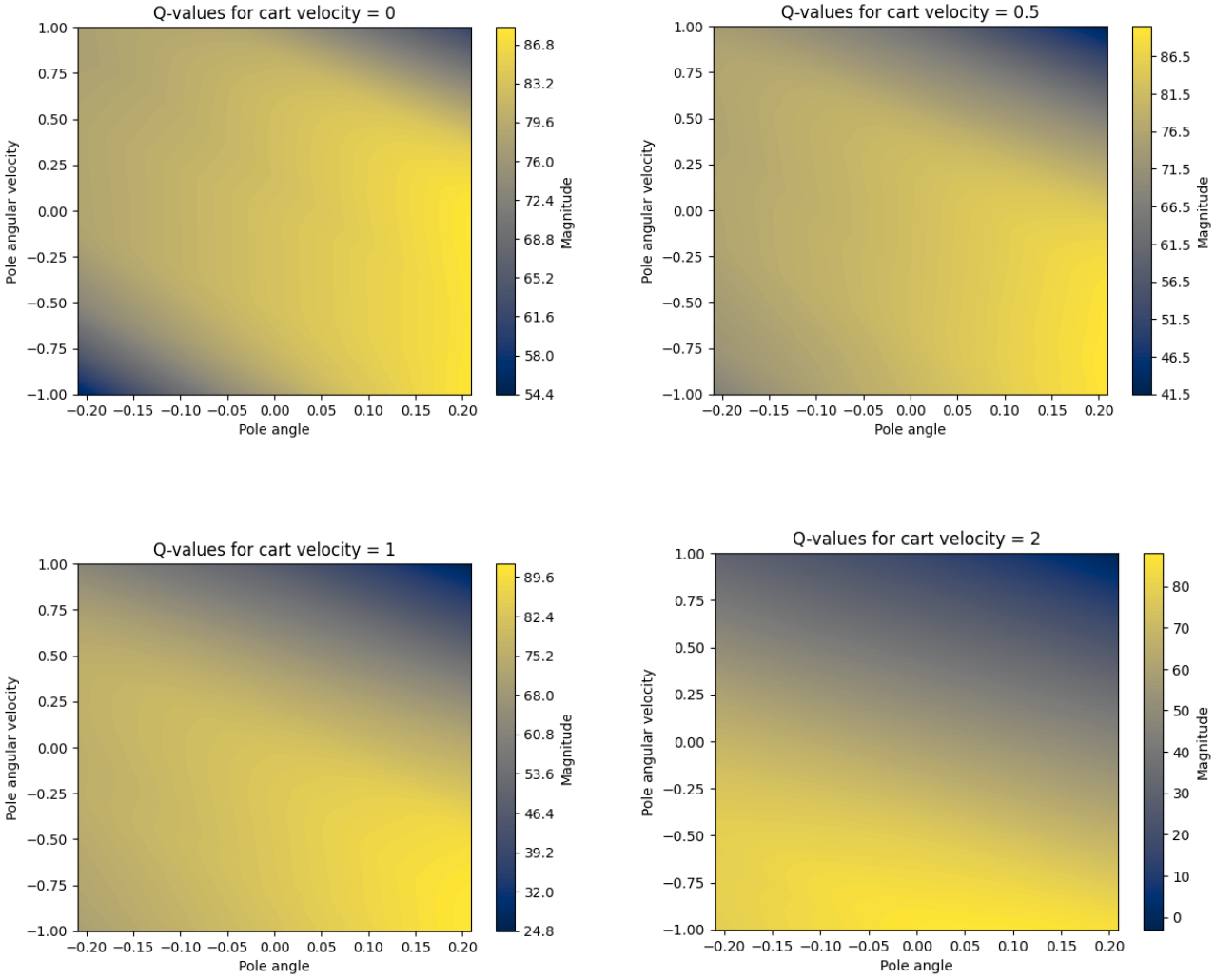

Figure 3: Q-Values generated by the DQN agent.