Solving the Cart Pole with Deep Reinforcement Learning

1 Tuning the DQN

1.1 Hyperparameters

Hyperparameter	**Value**
Number of neurons per hidden layer	50
Number of hidden layers	3
Learning rate	0.1
Size of Replay buffer	150000
Maximum number of episodes per run	150
Epsilon	0.1
Reward normalisation value	1
Memory sample size	64
Frequency of updating target network	4

Table 1: Identification and adjustment of the hyperparameters of the DQN.

We have decided to use a constant exploration schedule, as it has shown good practical results. The maximum possible reward was achieved with this approach, so there was no reason to make the model more exploratory.

1.2 Learning curve

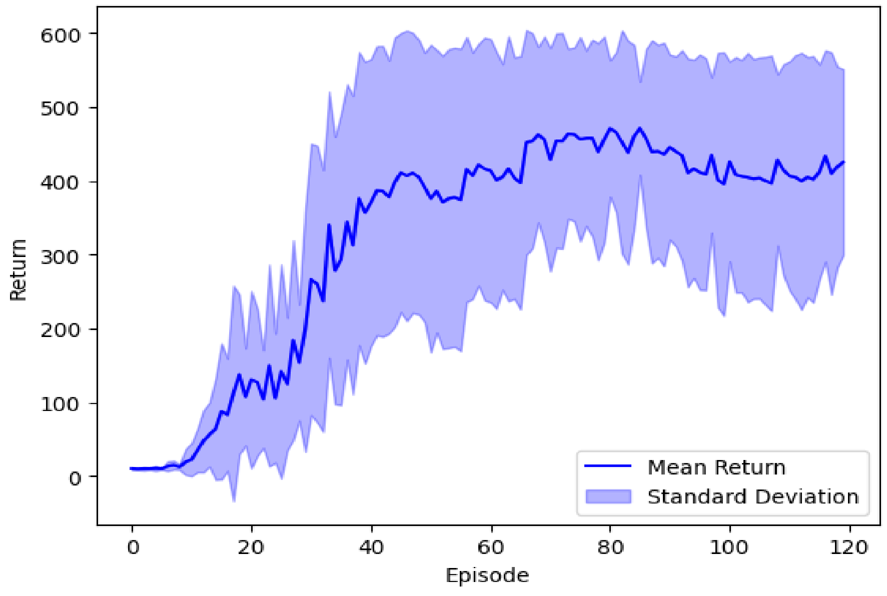

Figure 1: Learning curve of DQN after 10 trainings.

We can clearly see the model learns really fast, it always reaches a return of 100 from the episode 40 on. Even though standard deviation is a bit big, it has really good average returns of around 400, so we can say our agent has learned to control the pole.

2 Visualise your DQN policy

2.1 Slices of the greedy policy action

There are several possible scenarios, but generally, the optimal policy can be summarized as follows: when both the angle and angular velocity of the pole are positive, the agent should push the cart to the right. Conversely, if both values are negative, the cart should be pushed to the left. This action helps return the pole to an upright position. Additionally, the policy may need to adapt based on the magnitude of the pole's angle or the strength of its angular velocity, requiring different actions to achieve the same stabilization effect.

Given an optimal agent, its decision boundary should take the form of the function

$$f(x) = -a \cdot x + v,$$

where x represents the pole angle, $f(x)$ the angular velocity, v the velocity, and $a \in \mathbb{N}$. This formulation ensures the correct separation of the two decisions. The purpose of this function is to address situations where the pole angle is negative and the angular velocity is positive, and vice versa. These are scenarios where small changes become crucial in determining the action the model should take, as in the opposite cases, the action to take is quite clear. The values of a play a significant role in this context. The velocity factor creates a force that pushes the pole to the left, so as its magnitude increases, there are more states where the model must decide to push the cart to the left in order to counter the force being applied to the pole due to velocity.

When the velocity is zero, the symmetry of the actions should pass directly through the point $(0, 0)$ for the function we defined above, as velocity does not affect the pole. In this case, the agent should reason that no force needs to be applied when both the pole angle and angular velocity are also zero. This is the optimal position to avoid destabilizing the pole and achieve the maximum possible rewards, assuming the state remains unchanged.

As mentioned above, when velocity increases, the decision boundary should expand the range of values where the action is pushing the cart to the left, while reducing the range for pushing it to the right. This occurs because the pole is influenced by a force that causes a negative angle due to the velocity. Any push to the right would result in an even greater angle shift due to velocity, potentially leading to a terminal state.

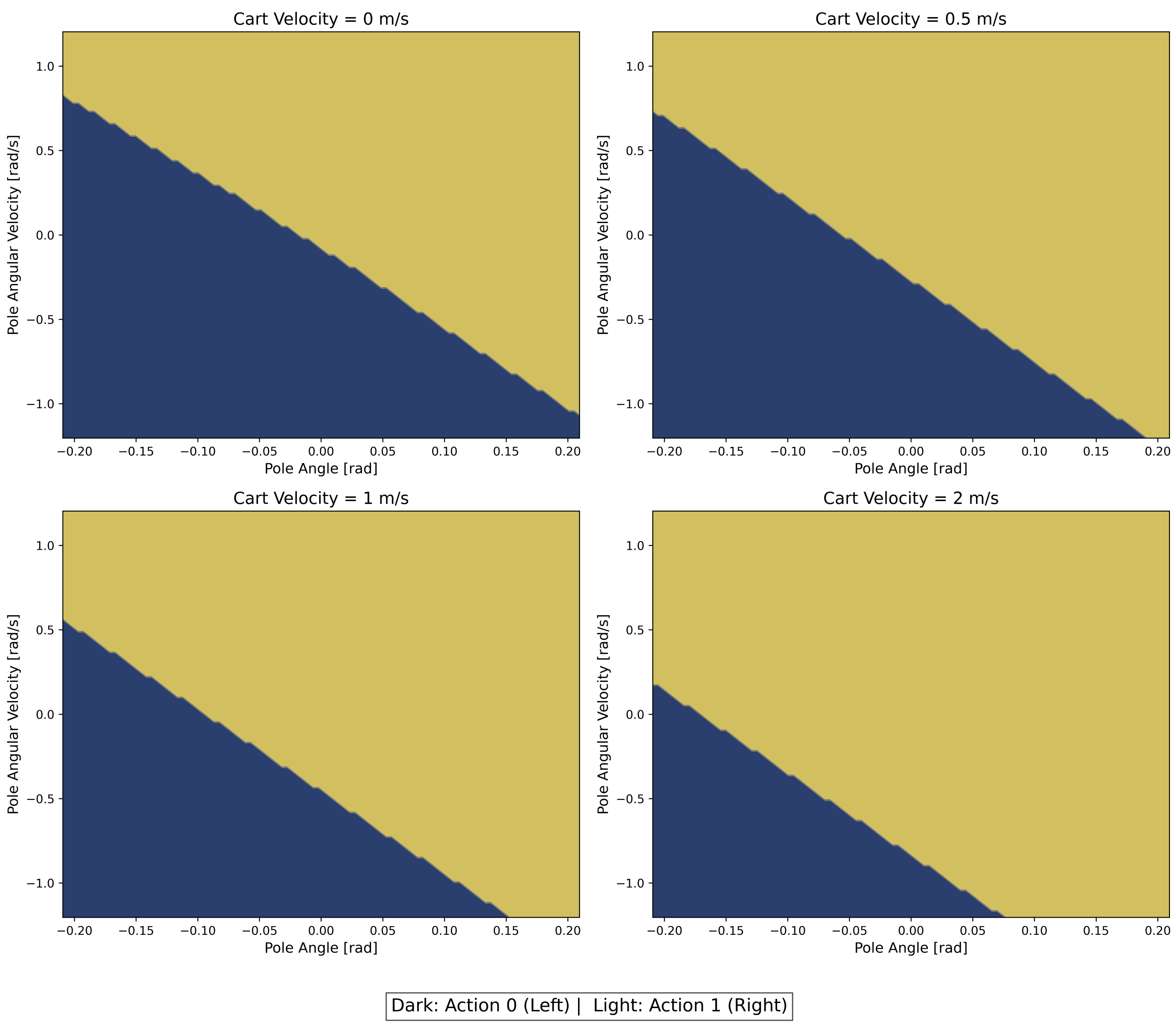

Figure 2: Greedy policy for different velocity values and cart position fixed to zero.

In our agent, we observe that it behaves as expected from an optimal agent. The shape of the regions in which it pushes to the right or left follows our predictions, and the decision boundary closely resembles the function $f(x) = -a \cdot x + v$. The symmetries of the action decision boundary pass exactly through the point $(0, 0)$, and it is clear that as velocity increases, the agent becomes much more likely to push the cart to the left.

2.2 Slices of the Q function

In this case, the plot for an optimal agent would exhibit two distinct and strongly contrasting colours. One colour would dominate in areas where the Q-values are relatively small, and the other in areas where they are relatively high. The first colour should emerge in two separate regions of the plot, indicating that the current state is relatively bad. As mentioned in the previous section, this occurs when both the pole angle and angular velocity are either negative or positive. Conversely, the second colour should appear between these regions, signifying that the current state is relatively good. As a result, these regions should align with the $f(x)$ function discussed in the previous section. The transition from the first colour to the second should be gradual, creating a blended mix of both colours in the intermediate region.

An optimal agent should have very high values in the regions far from the terminating states, as it should reason that it could remain in the same state, gaining rewards for a very long time. This would result in a value around 500, as it represents the maximum accumulated reward possible. On the other hand, close to the edge of the episode termination region, the reward should be much smaller than in the previous case, as in that state it is very likely that the model will reach a terminal state relatively soon.

When velocity is 0, the symmetries of the learned values should follow $f(x) = -x + k$, where x represents the pole angle, $k \in \mathbb{Z}$, and $f(x)$ represents the angular velocity. As mentioned earlier, in this scenario, the pole angle and angular velocity are the only influential factors, and the agent should learn that the state $(0, 0)$ is at least one of the best states. The function should have this shape because each line, as we vary the value of k, represents the states that have similar Q-values.

There is a force being applied to the pole once velocity increases as mentioned before, so it would be very reasonable for an optimal agent have higher Q-values for positive pole angel values, since it would end up in a straight pole. On the contrary, it should have lower Q-values for negative pole angle values. But all this would also vary depending on the power of angle velocity.

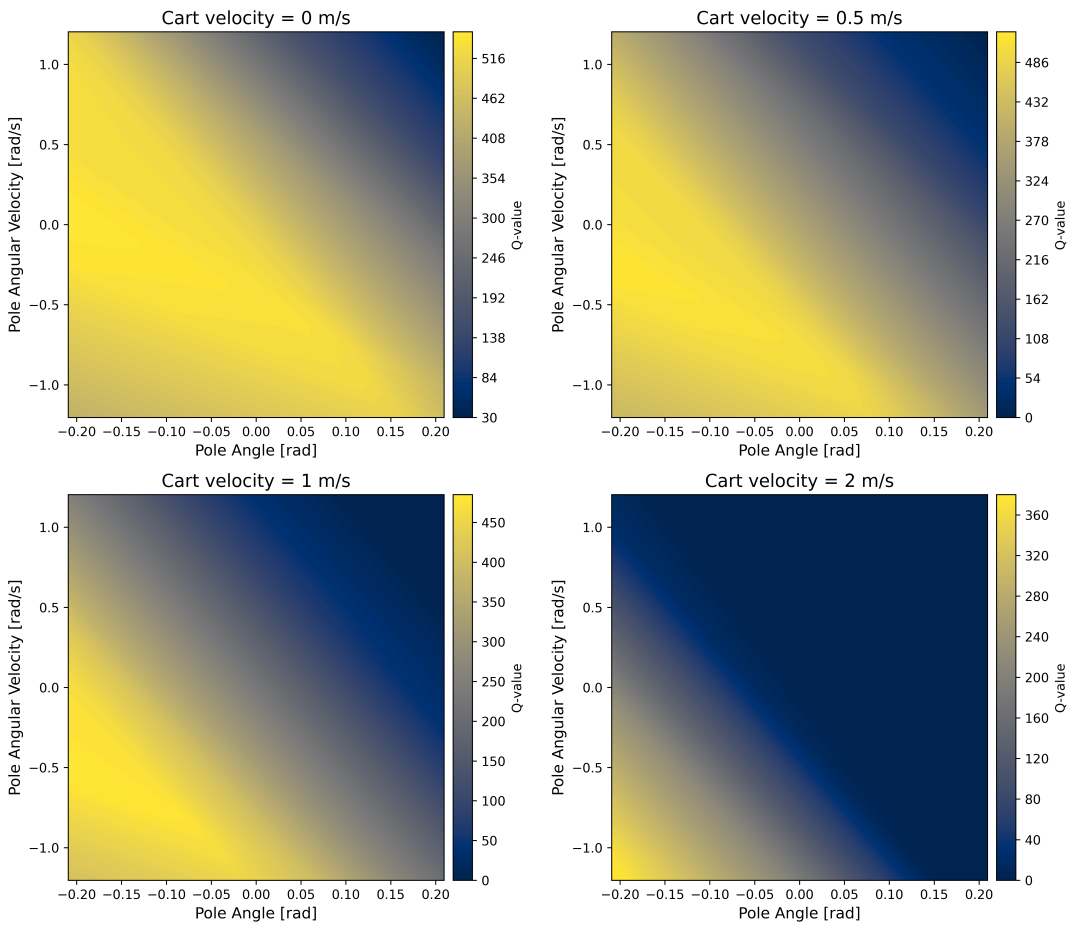

Figure 3: Greedy Q-values for different velocity values and cart position fixed to zero.

On the one hand, we can observe that our agent has learned the symmetries of the values it predicted, which are close to the optimal values for an agent.

On the other hand, we notice that the agent has learned very low values for states where both the pole angle and angular velocity are very high, but not for the opposite situation. The same behaviour is observed for values near the edge of the episode termination region and when velocity increases. This is counter-intuitive because, for example, a pole angle of -0.2 radians and an angular velocity of -1 rad/s represents a poor state, and it becomes even worse as the velocity increases. Beside, for higher velocity values, instead of having higher Q-values for higher pole angle and angular velocity, the opposite happens. The values learned by our agent align with our expectations in the range of values far from the edge of the episode termination region.

This issue arises because our agent is not perfect, and there are still situations where it cannot control the pole or it has not faced those situations too many time to learned the correct Q-values for them. To address this in future improvements, we would need to make the agent more exploratory by using a softer learning curve, allowing the agent to better handle such scenarios.

However, leaving aside the states where the pole angle and angular velocity are low, we can see that the values learned by our agent are very close to those we expected for an optimal agent and that is why we get really results in practice.