Question 1: Tuning the DQN

1.1 Hyperparameters

Fine-tuning a DQN demands exploration of the numerous hyperparameters. In this case, it was chosen to explore a total of nine potential hyperparameters to find an optimised model. Table 1 provides a comprehensive list of each hyperparameter chosen to tune along with its respective range of values, accompanied by an explanation of the reasons behind their selection for exploration. Two additional hyperparameters referenced in the code were held constant and not subjected to tuning. The first hyperparameter is the number of episodes per run and was reduced from 300 to 250 to save some time at each run, while the second hyperparameter is the number of training and was fixed at a value of 10.

Hyperparameter	**Range of values**	**Explanation**
Epsilon (ϵ)	$0 < \epsilon < 1$	Adjusting the initial value of ϵ yields additional insights into the balance between exploration and exploitation and its impact on the model's performance
Epsilon Decay (ϵ_{decay})	$0.9 < \epsilon_{decay} < 0.99$	As the number of episodes increases, ϵ is decreased to better leverage the learned policy. By adjusting the decay rate, we can observe its effect on the learning curves.
Learning rate (α)	$0.0001 < \alpha < 0.1$	Adjusting the learning rate influences the speed of convergence and impacts the model's ability to find optimal solutions. Experimentation of different learning rates is done to achieve optimal training performance.
Neural Network Architecture	(4, 18, 2), (4, 32, 2), (4, 32, 18, 2), (4, 64, 32, 2), (4, 32, 128, 2), (4, 64, 128, 2), (4, 128, 64, 2)	Different architectures are tested between 1 or 2 hidden layers to select the one that is producing the best performance.
Activation Function	[Sigmoid, ReLU, LeakyReLU]	Experimenting with different activation function helps in identifying the most effective of them.
Optimiser	SGD, Adam	Testing also with the Adam optimiser that shown good results in deep learning (1), possibly enhancing the performance.
Buffer Size	[100, 500, 1000, 10000, 15000, 20000]	Different buffer size have an impact on the learning curve and therefore important to test them to select the best one.
Batch size	[1, 2, 5, 10, 15, 20]	A range of batch size were used to see how it impacts the performance of the model.
Update Frequency	[1, 2, 5, 10, 20, 40]	The update frequency tested varies from 1 to 40 to see which one has greater performance on the model.

Table 1: Hyperparameters chosen to tune with their range of values.

Given the multitude of hyperparameters and their associated search spaces, it was used Optuna to facilitate the exploration of optimal hyperparameters and establish a preliminary model. The objective function set to be minimised by Optuna was: Total number of episodes - (Total number of episodes > threshold), where here threshold = 100. A lower objective function value indicates a greater number of episodes surpassing the threshold, this means that our goal is to maximise the number of episodes where the agent has an average return higher then the threshold.A total of 200 runs were conducted using the range of values outlined in Table 1. The hyperparameters corresponding to the best model were chosen. The parameters found by Optuna were: ϵ: 0.743, ϵ_{decay}: 0.99, α: 0.00474, Activation Function: LeakyReLu, Neural Network Architecture: [4, 128, 64, 2], Optimizer: Adam, Buffer Size: 1000, Batch size: 20, Update Frequency: 2. Diverse combinations of these hyperparameters were tested to ascertain the most effective final configuration chosen, provided in Table 2. The testing was done varying only one parameter while keeping the other constant to see the impact this parameter has on the learning curves and to select the optimal one.

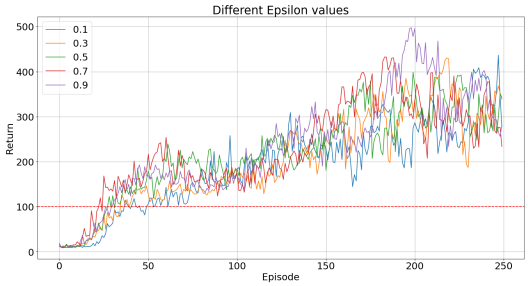

Figure 1: Different ϵ values

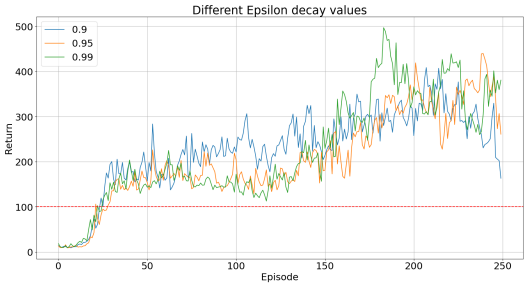

Figure 2: Different ϵ_{decay} values

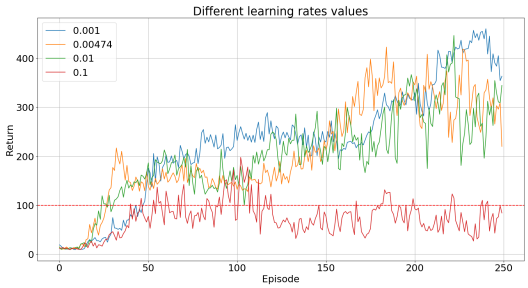

Figure 3: Different α values

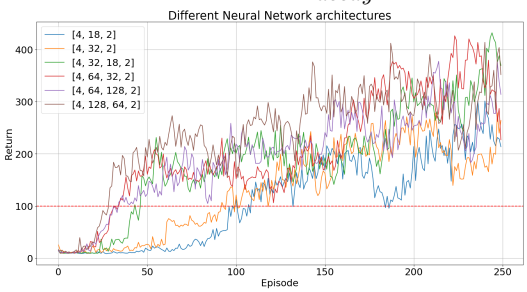

Figure 4: Different Neural Network Architectures

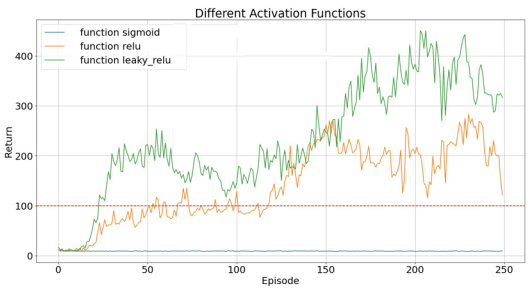

Figure 5: Different Activation Functions

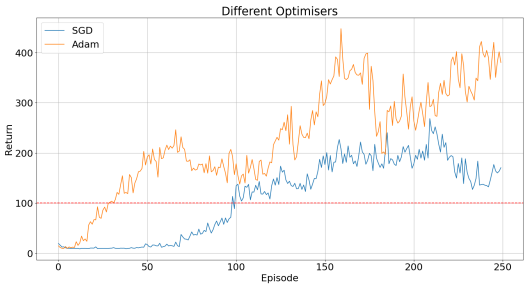

Figure 6: Different Optimisers

Hyperparameter	Value	Justification
Epsilon (ϵ)	0.74	The Figure 1 demonstrate that all ϵ values eventually reach the specified thresholds. Notably, values closer to 1 exhibit a quicker convergence to the threshold. Notably, the value 0.7 stands out as the one achieving the fastest convergence. Therefore, a value of 0.74 was chosen due to the fast convergence and create an initial exploratory phase, transitioning to exploitation due to the decay.
Epsilon Decay (ϵ_{decay})	0.99	It was chosen to apply an exponential decay to ϵ so that it uses the learned policy. By looking at Figure 2, it can be seen that all the value achieves the surpass the threshold, so any value could work but it was chosen 0.99 due to it's higher return.
Learning rate (α)	0.00474	The Adam optimizer demonstrates optimal performance when utilizing a learning rate within the range of 0.01 to 0.0001 (2). Looking at Figure 3 it becomes evident that a learning rate of 0.1 fails to reach the threshold. However, values within the range of 0.001 to 0.01 demonstrate successful convergence. A specific value, 0.00474, was selected for its rapid convergence in reaching the threshold.
Neural Network Architecture	[4, 128, 64, 2]	Various architectures were evaluated in Figure 4. Notably, configurations with a single hidden layer are slower to achieve the threshold compared to those with two hidden layers. Among the tested architectures, [4, 128, 64, 2] stands out for its good performance, demonstrating the fastest to reach the thresholds and achieving a high return.
Activation Function	LeakyReLU	LeakyReLU, as illustrated in Figure 5, reaches the threshold more rapidly than ReLU. In contrast, Sigmoid does not even reach the threshold.
Optimiser	Adam	The Figure 6 clearly depicts that Adam outperforms SGD for this particular task since Adam reaches the thresholds 61 episodes before SGD does. Adam also reaches a higher final return the SGD that has a slower growth rate
Buffer Size	10000	In Figure 7, it is shown that the buffer size has minimal impact on reaching the threshold. However, a smaller buffer size results in an overall lower return at the end of the 250 episodes. A buffer size of 10000 was chosen to strike a balance between accumulating a substantial amount of experience while avoiding excessive computational cost.
Batch size	15	Examining Figure 8, it can be seen that using a small batch size results in sub-optimal performance in reaching the threshold. The optimal range lies between 10 to 20, and a value of 15 was selected for its rapid increase in return.
Update Frequency	1	The Figure 9 indicates that the highest the update frequency, the poorest the performance is. Updating every 20 episodes doesn't make the model learn well from its experiences, causing it to reach the thresholds much later. In contrast, a value of 1 for the update frequency produces a quicker reaching of the threshold and stays constantly above it.

Table 2: Hyperparameters chosen with justification.

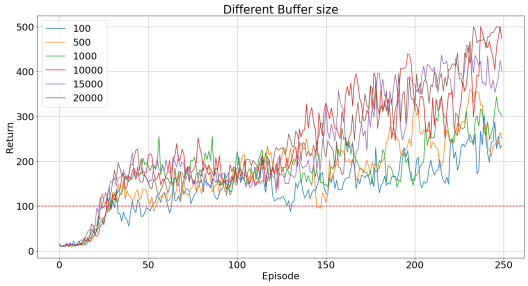

Figure 7: Different Buffer Size values

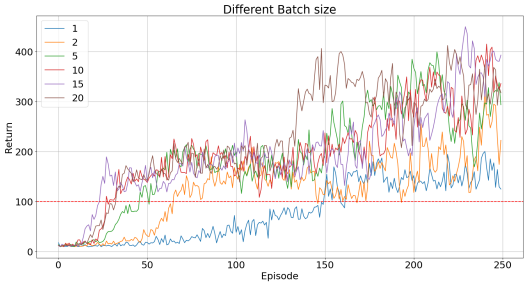

Figure 8: Different Batch Size values

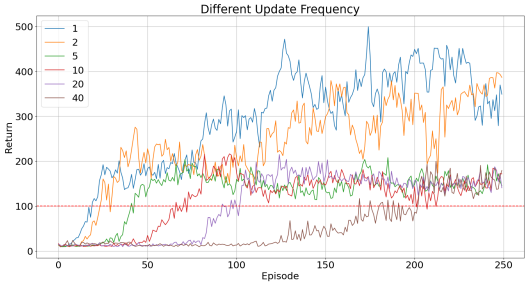

Figure 9: Different Update Frequency

1.2 Learning curve

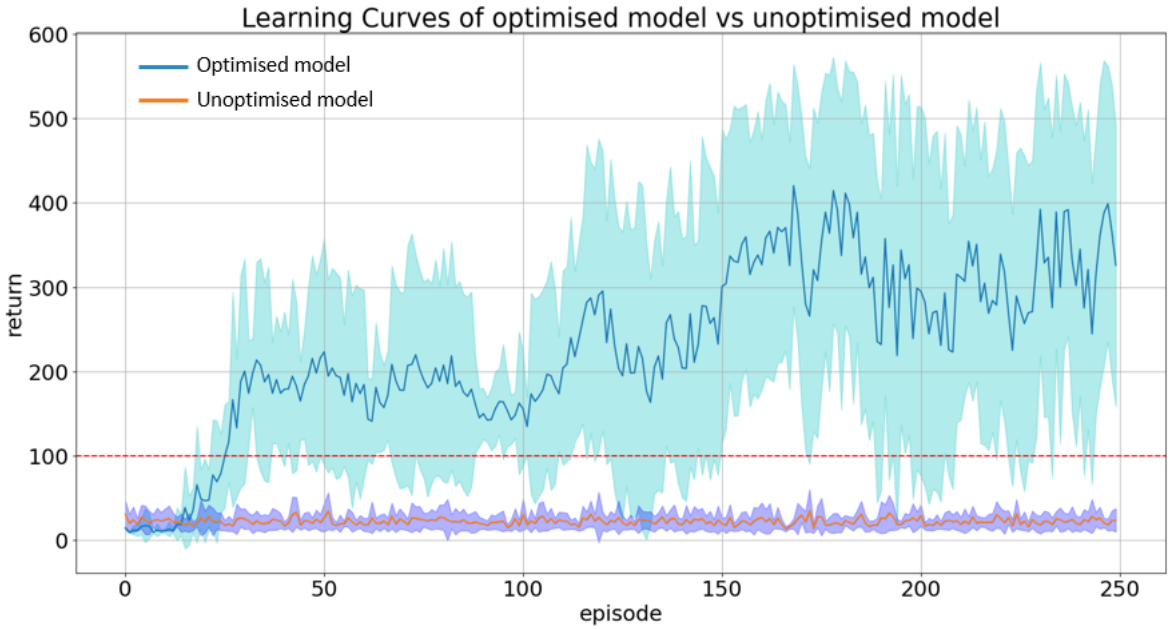

Figure 10: Comparison of the learning curves between the optimized and unoptimized models, based on the average results from 10 runs using parameters specified in Table 2 along with their standard deviation. the optimised model has 223 consecutive episodes with a higher average return then the threshold

Question 2: Visualise your DQN policy

2.1 Slices of the greedy policy action

In this scenario, where the pole is initially positioned between -0.05 to 0.05 radians, the sign of the angular velocity determines the direction in which the pole is falling. A positive angular velocity indicates that the pole is tilting to the right, while a negative angular velocity suggests a tilt to the left. To maintain stability, corrective actions are required. Specifically, if the pole is tilting to the right (positive angular velocity), the cart should be pushed to the right to counteract the fall. Conversely, if the pole is tilting to the left (negative angular velocity), pushing the cart to the left is necessary to counterbalance the tilt and maintain stability.

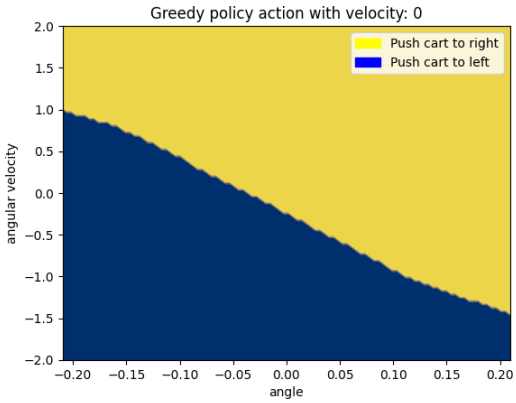

Figure 11: Velocity = 0

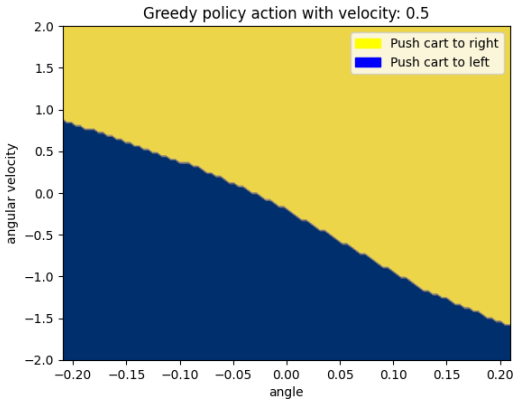

Figure 12: Velocity = 0.5

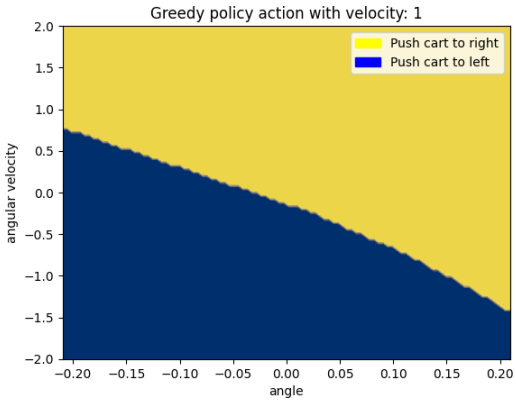

Figure 13: Velocity = 1

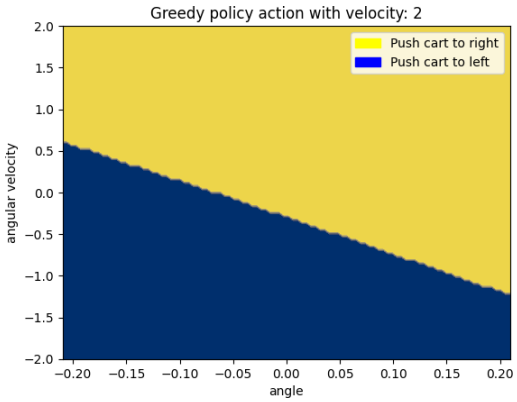

Figure 14: Velocity = 2

From Figure 11 to 14, it is shown the fluctuations in the greedy policy action as velocity changes, with the cart positioned at the initial position 0. For example, looking at Figure 11 when the velocity is 0, if the angular velocity exceeds 1 (indicating a tilt to the right), the best action is to push the cart to the right, regardless of the pole's angle. This shows that the policy learned the corrective action to restore stability

and bring the pole back to an angle of 0°. On the other hand, when the angular velocity falls below -1.5 (indicating a tilt to the left), the policy learned shows that, regardless of the angle position the pole is, the best action is to push the cart to the left, aiming to realign the pole's angle back towards 0°. As velocity increases, the nonlinear separation line, showing the boundaries for the two different push directions, shows a less negative slope. This implies that with increasing velocity, the line tend to flatten, allowing for an expanded region on the top left where the action to push the cart to the right and expanded region on the bottom right for pushing the cart to the left (going from -1.5 to around -1). This is expected since with the velocity increased, it is highlighting the need to start moving the cart sooner in anticipation of a possible fall, showing a quicker response with the learned policy. However, it can be seen that with increasing velocity, the region favoring the action of pushing the cart to the right expands in comparison to the other one, making the best action more frequently align with pushing the cart to the right. This could be due to the pole having a higher probability of falling to the right, therefore more the area corresponding to pushing the cart to the right increases.

2.1 Slices of the greedy policy action

The angle of 0 will be referred to as the central angle. Between Figure 15 and 16, specific portions of the Q-value function reveal yellow areas indicating a high Q-value, signifying a desirable state for the pole. Reciprocally, as the color shifts towards blue, the Q-value decreases, indicating a less favorable state. On Figure 15, the function assigns a high Q-value to the state inside the diagonal going from the top left to the bottom right. This is expected because, in these states, the angular velocity could be going towards the central angle, and the cart having the action of going the same direction of the angular velocity, making it easy for the cart to stabilise the pole or alternatively, the angular velocity is diverging from the central angle (the pole is falling away from the central angle and towards the maximum angle which is 12°, but the angle is not far from the central one and the action of going in the same way to stabilise it. On the other side, when the angular velocity is moving away from the central angle and the angle is approaching its maximum limit (12 degrees here), a low Q-value is assigned to this state due to the high probability that the pole is not being able to be stabilised anymore. This is evident in Figure 15 at the top right and bottom left corners, where stabilizing the cart becomes harder.

As velocity increases, it was shown in Figure 14 that the action of moving the cart to the right takes more space, this is why that in the Figure 18, a bigger blue area (meaning less good state) can be seen on the top right. This happens due to the optimal action within the blue area involves pushing the cart to the right. In this region, the angular velocity is positive and the angle towards the right direction. Therefore, the cart is pushed to the right when the pole is falling in that direction, aiming to stabilize it. However, it is harder to prevent the pole from falling, leading to the assignment of a low Q-value in this blue area.This observation can be seen in Figure 15 to 18, where the region with low Q-values expands in the top right corner. This expansion is attributed due to the expansion of the area where the action of pushing the cart to

the right when velocity increases from Figure 11 to 14.

Furthermore, as velocity increases, the overall Q-value of each state decreases, possibly due to the instability and increased difficulty of the system to balance the pole. For example in Figure 15, the highest Q-value reaches 450, while in Figure 18, the highest Q-value is 405.

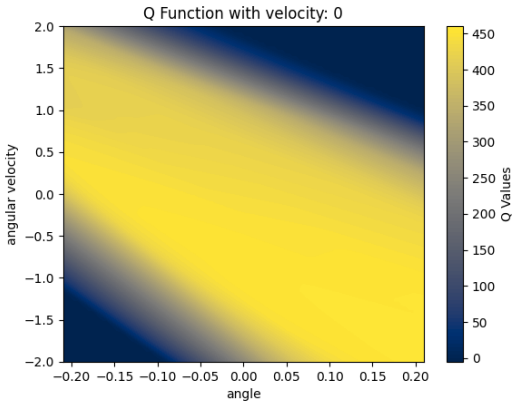

Figure 15: Velocity = 0

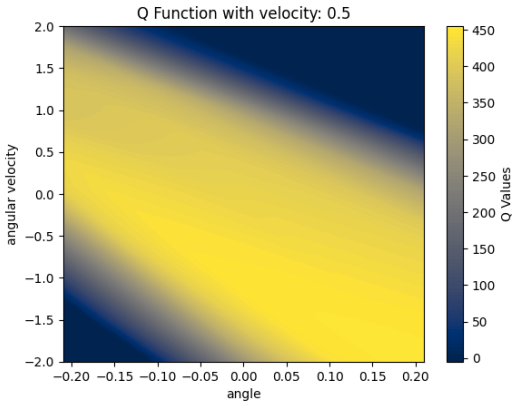

Figure 16: Velocity = 0.5

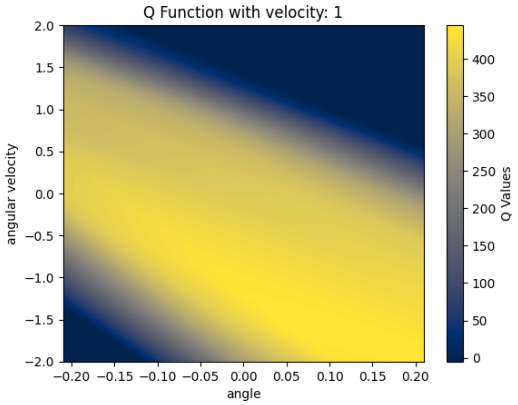

Figure 17: Velocity = 1

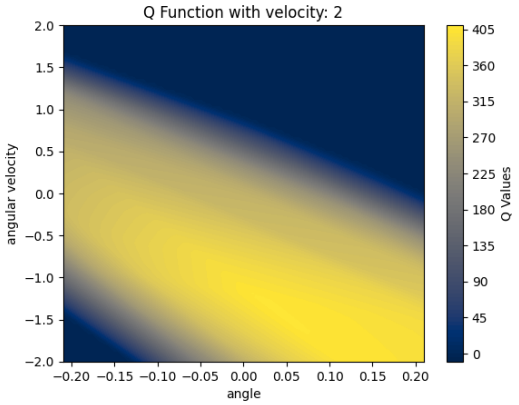

Figure 18: Velocity = 2

References

[1] Kingma DP, Ba J. Adam: A Method for Stochastic Optimization; 2017. pages 2

[2] Wilson AC, Roelofs R, Stern M, Srebro N, Recht B. The Marginal Value of Adaptive Gradient Methods in Machine Learning; 2018. pages 4