Introduction

In this homework, we are going to train a Deep Reinforcement learning network using Gym environment. The game used in this homework is cart pole. In the first part, we compare the effects of two different exploration profiles, Soft_max and Epsilon_greedy policies, in the learning curve of the RL agent.

In the second part of this homework, we trained the RL agent for the same problem, *CartPole*, but with a different *observation space.* In this part, the observation space will be the screen pixels of the environment.

Model

The network used for studying the exploration profile is defined with named DQN, a very simple fully connected neural network with two layers with 128 neurons in each layer. Because the observation space has four elements, which are shown in Fig 1, the model has an input layer with four neurons

Softmax Action Selection

This selection method is defined in a function named *choose_action_softmax. In the s*oftmax method, the network output is turned into the probability and with that probability, the action is selected not just by selecting the index of the maximum element. This is done by the formula shown in Equation 1.

In this formula, τ is called temperature. Choosing a high temperature causes the actions to be nearly equal. In the case the temperature is zero it becomes greedy action selection.

Greedy action selection

In greedy model selection, with probability $(1 - \varepsilon)$ the action is the best possible action and with probability ε the action is randomly chosen from other possible actions (in this case other possible action is only one). This method is defined in a function named *choose_action_epsilon_greedy*.

Training loop

The training is done in a *for* loop. In this loop first state is the initial state and then an action is chosen using one of the two above selection methods and then the current state, the next state (the state that is achieved by the chosen action), reward, and Q values are saved in the memory. Updating the network weights starts after there are 1000 samples in memory. During the update process which is done in the function named *update_step,* q-values corresponding to the actions taken are calculated by the policy network and then q-values of the next states are calculated using the target network, and according to formula 2 the expected q-values are computed and by comparing the *q-values* and *expected q-values* the loss can be calculated. Scores of all trials

are saved in a variable and at the end, the training process stops if last 30 trials achieved the maximum.

Using screen pixels for observation space

In this part, we changed the observation space and used screen pixels for the training process. The code for this part is written in a different Jupyter notebook named *HW3Part2_cartpole_screen*. The first thing that comes to mind for training a neural network on images is that the best way is to use a convolutional neural network (CNN). Therefore, I defined a CNN with three convolutional layers. The architecture used is shown in Table 1. By using the command *env.render(mode='rgb_array')* the screen is returned. After that using the function *get_center* the location of the cart pole is found. For finding the location of the cart pole first some rows of the image from the bottom are cropped and in the first row from the bottom, we search for a pixel with value one (black pixel) then add its index to 30 (half of the width of the cart pole). To decrease the computation we should crop the image around the cart pole and get rid of excessive parts, this is done in the function *CroppImage.*

Results

Comparison of Softmax and greedy selection methods

In this section, we trained the agent using two different policies. For the epsilon greedy policy, two different methods can be used first one is to use a constant value for epsilon and the second one is to use a decaying value for epsilon. For the latter, I used the normalized values of the *exploration* variable which is used in Softmax as temperature. The score curves for these three methods are shown respectively in figures 2, 3, and 4.

Running average of all of these methods is shown in Figure 5. As it can be seen by using constant epsilon value agent can learn in fewer trials.

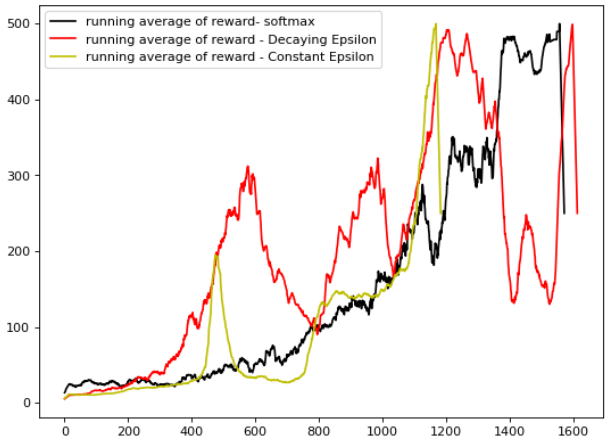

Fig 5. Comparison of Softmax and Epsilon greedy policy

Reward function

In the previous part, the reward that was used was only the reward given by the environment but here we added another term to the reward to see it compact on the training process. In the cart pole game, the best output is that the cart pole stays in middle and does not go away too much from the middle so proportional to the cart location we added a negative term to the reward function. We trained the network with this new reward function using both Softmax end epsilon greedy policies and the learning curve for these two policies are shown respectively in Fig 6 and 7 and in Figure 8 running average of both of them are shown. Both methods show a better convergence compared to the previous one in which the reward was only the output of the Gym. Again with this new reward, the epsilon greedy policy performed better than

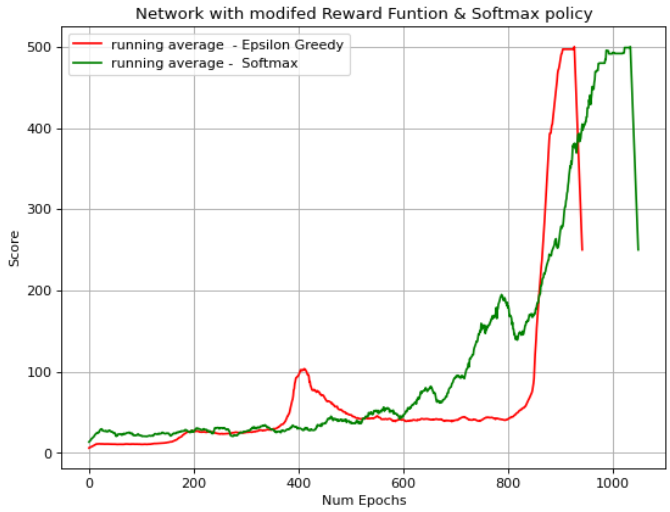

Fig 8. Score curve after modifying the reward

Screen pixels as observation space

Until here the observation space used for training the network was the four elements shown in Fig 1. Here we changed the observation space and instead of receiving the information given by the environment we just used the screen pixels of the environment. For achieving temporal context such as angular velocity we feed the network with the difference of two consecutive screens. For lowering the computation burden, we first did some preprocessing on the image. In the preprocessing step, first the location of the Cartpole is found and then an area with 30 pixels for height and 33 pixels for width. Next, the image is resized to 30*30.

In Fig 9 the scores are shown for two different learning rates (0.001 and 0.002). As it can be seen in some cases, the cartpole achieves a score higher than 500 but the mean score is nearly 400. (The videos related to this part are in the folder and also in the Jupyter notebook which are shown in Google Colab). Since the screens were cropped exactly around the cartpole, the information about the location of the cartpole is not given to the network and by paying attention to the output videos of this part it can be seen that in parts that the network does not achieve maximum score is because the cartpole goes out of the permitted range.

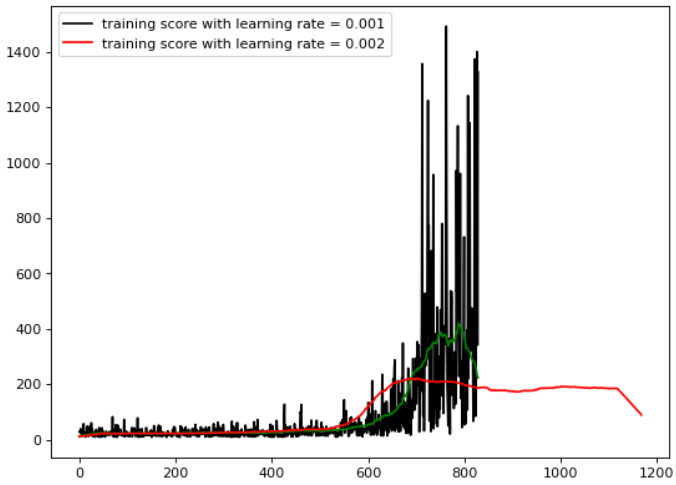

Fig 9. Learning curve of network with two different learning rate

Finally, another method of solving this problem was explored. In this part, we did some more preprocessing on the screen. Since the color of the images seems not to be so important we changed the images to gray and then using a *canny* edge detector we turned the images into binary images. Then for training the network we merged three consecutive frames of the environment into one frame and used it as the state of the environment. An example of an output of this preprocessing step is shown in Fig 10 and the learning curve of this part is shown in Fig 11. This Fig does not show a good result compared to previous sections for this idea.

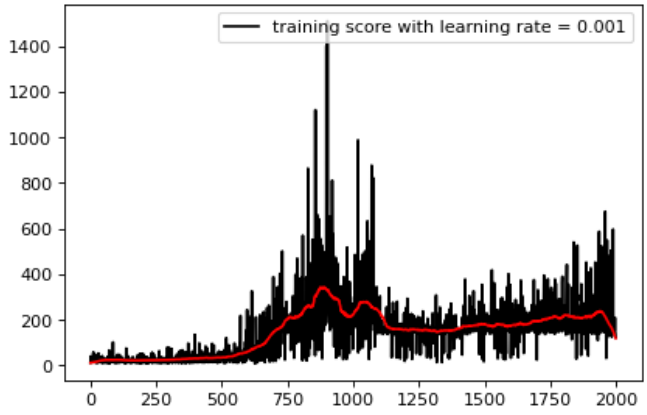

Fig 11. Learning curve of network trained with three consecutive frames

Appendix

$$P_t(a) = \frac{exp(Q_t(a)/\tau)}{\sum_{i=1}^{n} exp(Q_t(i)/\tau)}$$

Equation 1. Softmax

Num	Observation	Min	Max
0	Cart Position	-2.4	2.4
1	Cart Velocity	-Inf	Inf
2	Pole Angle	~ -41.8°	~ 41.8°
3	Pole Velocity At Tip	-Inf	Inf

Fig 1. Observation space of cart-pole

$$Q(s,a) = r(s,a) + \gamma \max_{a} Q(s',a)$$

Equation 2. Reward function

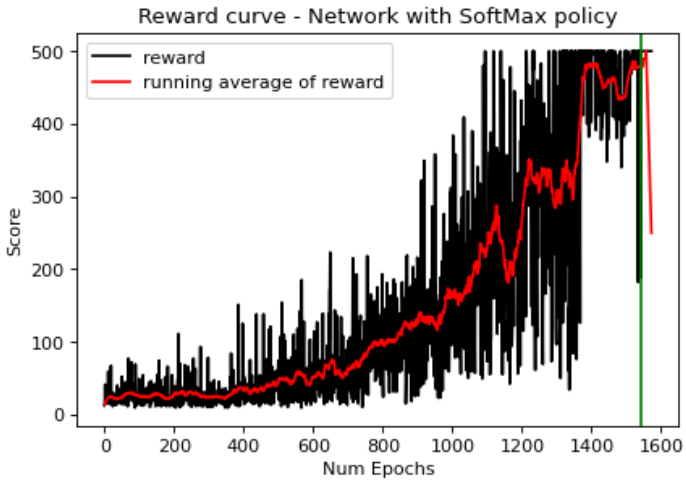

Fig 2. The learning curve of the network using the Softmax policy

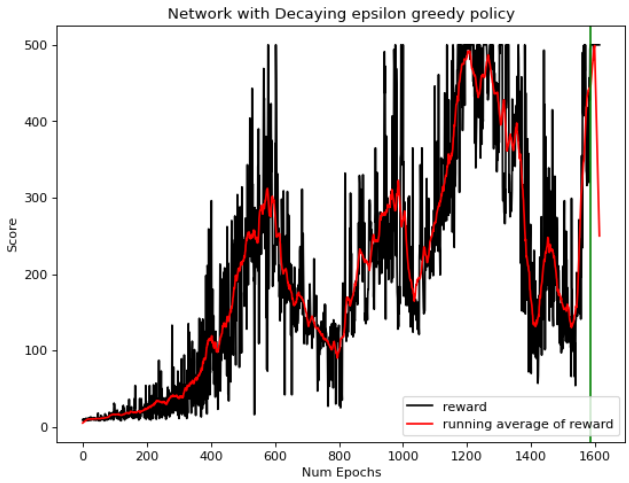

Fig 3. The learning curve of the network using the epsilon greedy policy (decaying value)

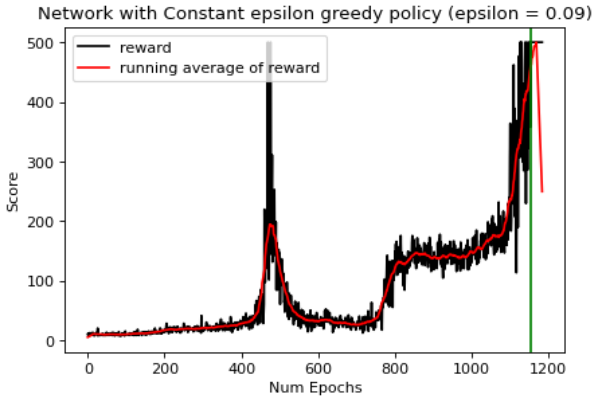

Fig 4. The learning curve of the network using the epsilon greedy policy (constant value)

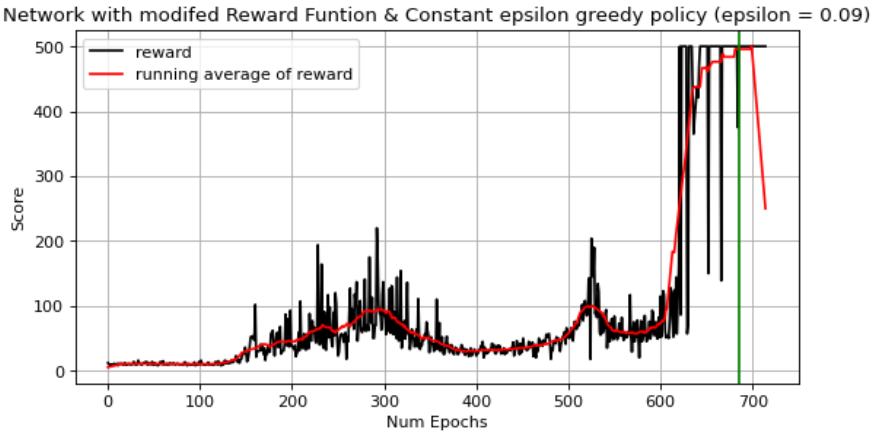

Fig 6. The learning curve of the network using modified reward & epsilon greedy policy (constant value)

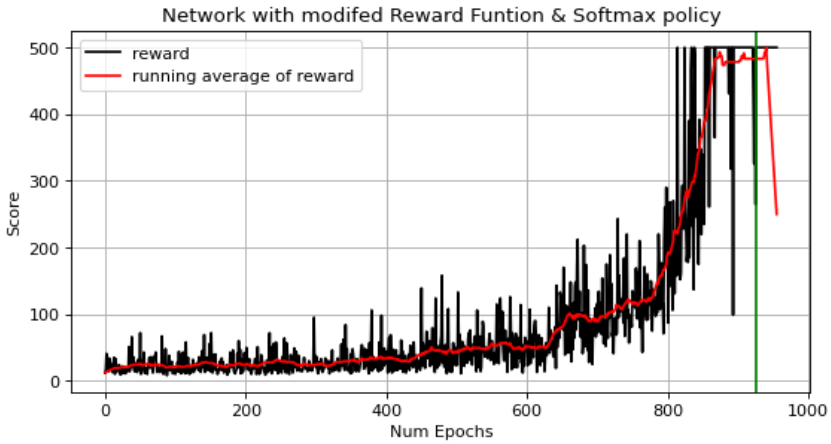

Fig 7. The learning curve of the network using modified reward & Softmax policy

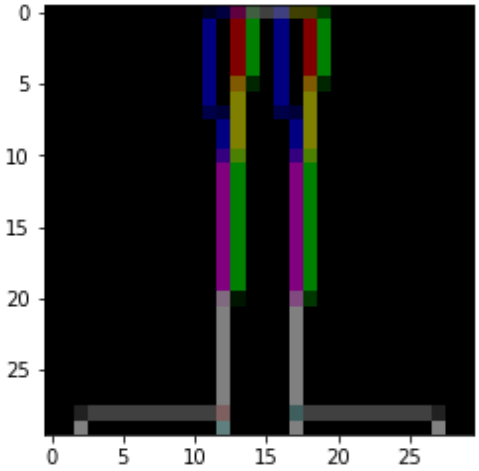

Fig 10. A sample of preprocessed section