Self-driving cars Exercise 2 – Reinforcement learning

Nuri Benbarka – Robert Herzig

2.1

a)

Our network consists of 3 convolution layers for the input images and a fully connected layer for the sensor inputs, then the outputs of both nets are connected to a fully connected network of 4 layers. All convolution layers have a kernel size of 5 and use ReLU activation. The number of input channels are 3 (RGB input), 12 , 24 respectively and the output has 32 channels, so each time the network gets deeper. To reduce the dimensions of the input striding is used in every layer and max pooling in the second layer. The fully connected network at the end starts with an input of 640 (4 * 4 * 32 + 128), then it reduces its dimension gradually until it reaches the number of actions.

As for the sensor values, it would be a problem if we don't include both the sensor values and the part of the image that has them. This is because the cropped image itself doesn't have all of the information of the world which we need to make good decisions, and therefor our system will not be a Markov decision process. The system has the Markov property when the current state completely characterizes the state of the world.

b)

We use both experience replay and iterative update to reduced the instability of reinforcement learning. Experience reply removes the correlation between the inputs by taking a random sample from a large database of previous runs, therefor it is more generalized. On the other hand, Iterative update reduced the chance of big sudden change in the policy which leads to instabilities.

c)

In order to understand why we should balance exploration and exploitation, we should see the two extremes of using only one. If only exploration is used, the agent will only give random actions and the whole system is not used. On the other hand, if only exploitation is used, the agent will not take new actions and will only take actions its comfortable with, so learning will not be effective. Therefor there should be a balance between both actions and this is what the e-greedy algorithm is used for. In this algorithm, depending on a value 'e' , which is a probability of taking a certain action, the agent takes either an exploration action or an exploitation action. So the plan is to change 'e' from a high value at the start to a low value at the end. This means the agent at the start will take actions with a high exploration probability and explore new states then at the end it will refine it knowledge by using exploitation with low exploration probability.

d)

At training the agent learns to get positive points very quickly. But it well not get really high scores until the probability of performing an exploration becomes low. Figure 1 below shows the rewards during training of agents with an exploration fraction of 10% and 50% respectively.

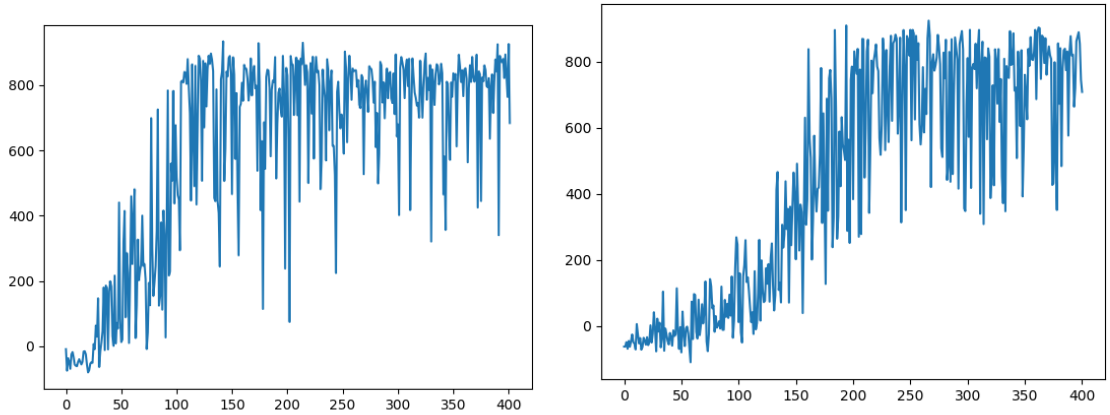

Figure 1: the rewards during training of agents with an exploration fraction of 10% (left) and 50% (right)

As seen from the figure the agent with 10% exploration fraction gets to higher rewards more quickly. But when evaluating both agents, the agent with 50% exploration fraction performed better and you can feel it has more experience.

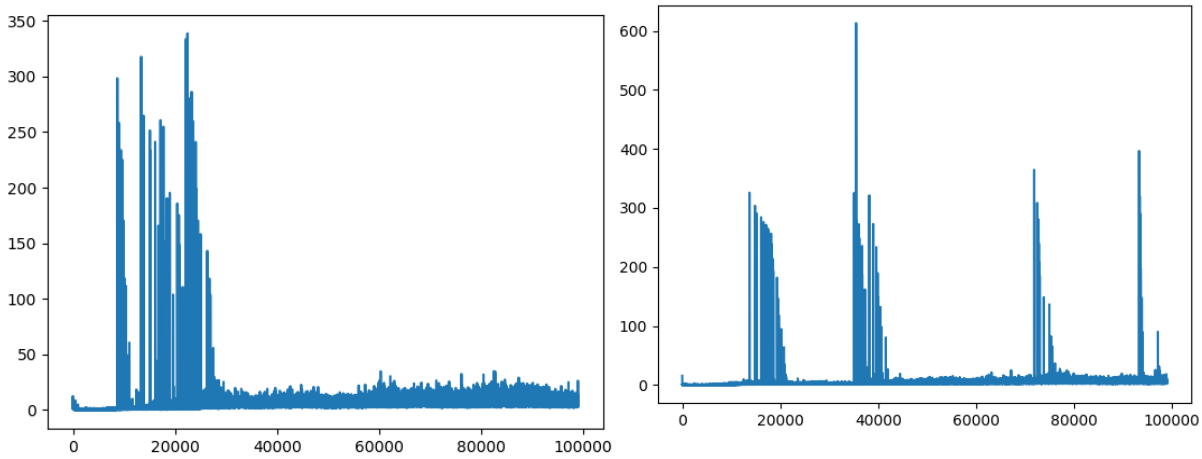

Figure 2: the training loss during training of agents with exploration fraction of 10% (left) and 50% (right)

In most of the scenarios the training loss is constant with some spikes in between. In contrast, in supervised learning the training loss will always decrease until the network reaches a local or a global minimum. This difference is caused by the fact that in supervised learning the network trains on a limited dataset so the error while go to a minimum eventually, while in reinforcement learning the network trains on the data collected while it is running which is almost like an infinite dataset. **(for the rest of the simulations the exploration fraction is 75%)**

e)

Our performance in the first assignment was poor, so it is better in every aspect especially robustness. The car trained using reinforcement learning can perform very well even in complex situations and sometimes it gets higher scores than us (humans). The reason for this robustness is the exploration which solves the problem of the unseen states in imitation learning.

2.2

a) Discount factor:

while running the system the environment will only give reward to the last action only. If our policy learns to choose the action with the highest immediate reward, this doesn't guaranty that we will get the highest total reward over the whole run. This is because sometimes an action which looks good now, would lead the agent into difficult or losing situation. For this reason, we should use the expected cumulative discounted reward as a evaluating metric, as shown here:

r_t + (γ * r_(t+1)) + (γ^2 * r_(t+2)) + ..

where r_t is the reward at time t and γ is the discount factor which has a range between 0 and 1. as γ increases the effect of future rewards will be stronger, which means the agent will learn to choose short term losses to get long term gains. With our system we tested the effect of the discount factor. We ran the program with discount factors of 0.5 , 0.75 , 0.999 as shown in figure 3. the agent with discount factor of 0.5 couldn't get a score higher than 150 x, and the agent with a value of 0.75 got higher rewards, but on average it just above 200. And if you see its driving behavior, it learned how to take corners and to drive forward but it didn't learn to brake before hard turns which made it drive into the grass repeatedly. The last one didn't work well maybe because of the higher complexity when looking way ahead in the future.

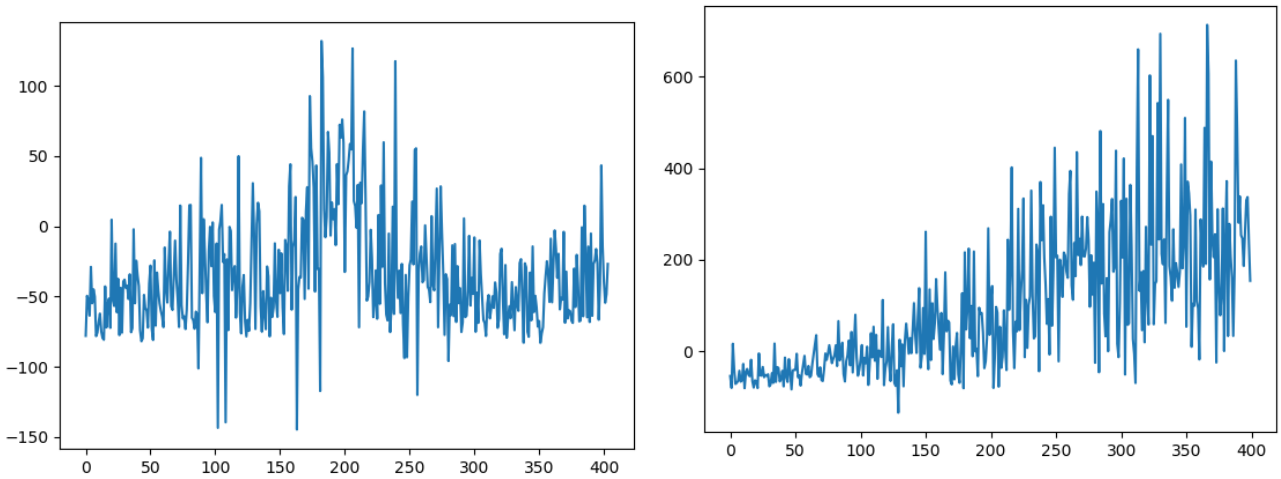

Figure 3: the rewards during training of agents with a discount factor of 0.5 (left) and 0.75 (right)

b) Action repeat parameter

The action repeat parameter is used to see the amplified effect of using an action, and also helps to get a policy with less alternating action decisions. A simulation was made with two different action repeat parameters 1 and 8. the average reward of using action repeat of 1 was high, but in the evaluation you can see that the car is often changing its driving direction even in straight roads. As for action repeat 8, the car couldn't learn anything because it is hard to get good results if you allowed only to use the same action 8 times.

c)Action space:

In the previous simulations 7 actions were used full and half accelerations, full and half turns to the left and right and a full brake. First we investigated adding the null command the reward curve looked the same, as shown in figure 4, but when when you see the evaluation the car decides to not to increase its speed so much compared to previous simulations. Secondly we decided to add another two turning commands to the left and right so there are 11 actions in total. In this case the duration of the exploration the rewards are mostly negative, as shown in figure 4, because the probability of getting a good action is lower with more commands. After the exploration time the agent starts to get high rewards. One observation during evaluation is that the agent learned to hold the brakes when a really hard turn is head and continues to hold the brake until the time is over. I think if the training is continued furthermore the agent will perform better.

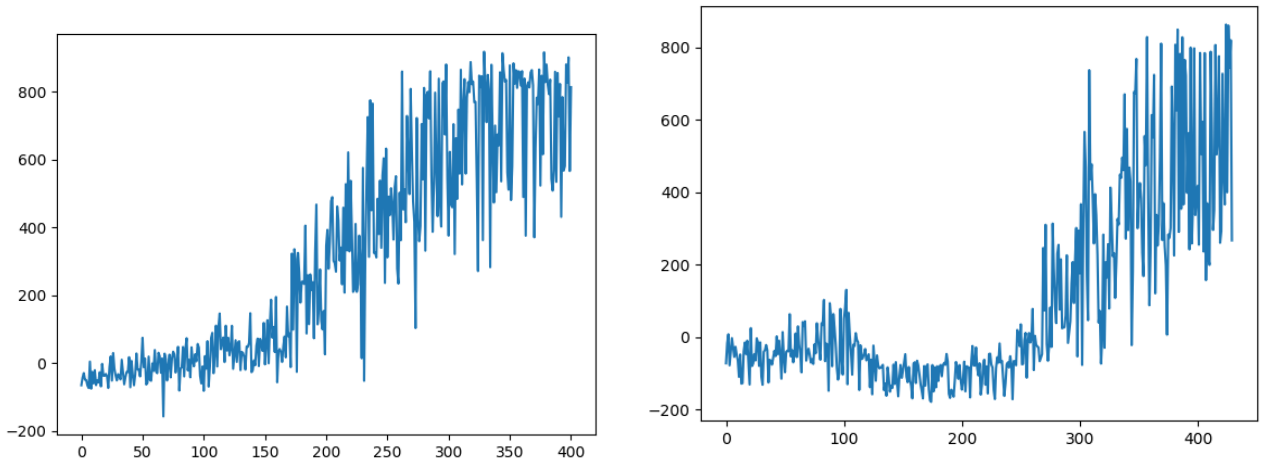

Figure 4: the rewards during training of agents with a null action (left) and 11 actions (right)

d) Double Q-learning:

the estimation errors during learning lead to a bias which causes overestimation. And the solution to this is to decouple the action selection and action evaluation, by using the policy network in the action selection and target network in the action evaluation. This will reduce the overestimation and will lead to better policies. In all of the previous simulations Double Q-Learning was used. The normal Q-learning network the results were not good so I decided to continue with Double Q-learning.

e) Best solution:

the best solution is the Double Q-learning with exploration fraction of 0.5 and discount factor of 0.99.