Assignment 3: Applied Deep Learning

Jan Janiszewski, UCNumber:2023198681*
Yorick Scheffler, UCNumber: 2023176104
jasiekj1@gmail.com
yorick.scheffler@student.hpi.com
University of Coimbra
Coimbra, Coimbra, Portugal

ABSTRACT

This report accompanies the Colab-Notebooks that are created to solve the tasks given in this assignment. This is a practical introduction to work with Reinforcement learning.

KEYWORDS

CNN, Neural Networks, OpenGym, Reinforcement Learning, Car Racing

1 CREATING & TRAINING (PART1)

General Implementation

The first task of the assignment is the implementation of the network architecture. The architecture consists of a feature extractor that is implemented using convolutional layers. These are followed by linear layers that help the network fulfill the task. This architecture is depicted in figure 1. Additionally, for the implementation of the dueling layers, some changes to the network were required.

The next task is to train the network with different loss functions or dueling layers. In total, there are 4 scenarios to test:

- DQN-Training
- Double DQN Training
- DQN-Training with Dueling
- Double DQN Training with Dueling

1.1 Performance Comparision

The different approaches archived in the end similar good results but they differed in how fast they got there and also how stable they remained after reaching that level. That is best to be seen in the plot of the average scores during the training runs. Another helpful evaluation is checking in qualitatively through the videos that were generated every 50 epochs. In the following, we go through the different approaches and analyze them in more detail.

DQN-Training is the basic approach for training the reinforcement learning algorithm for car racing. This average score with some additional information is shown in the plot 2. It follows in general the learning curve we could observe with all approaches. They differ a bit in how fast the network improved its results drastically. Also what can differ is the amount the average scores drop late in the training process. We are not sure what is causing that, but we think it is most likely that the model becomes a bit overconfident, which results in way worse results.

The next training approach we evaluated is the Double DQN. The results are depicted in figure 3. The differences are mainly that the model starts to improve a bit quicker and it drops more often

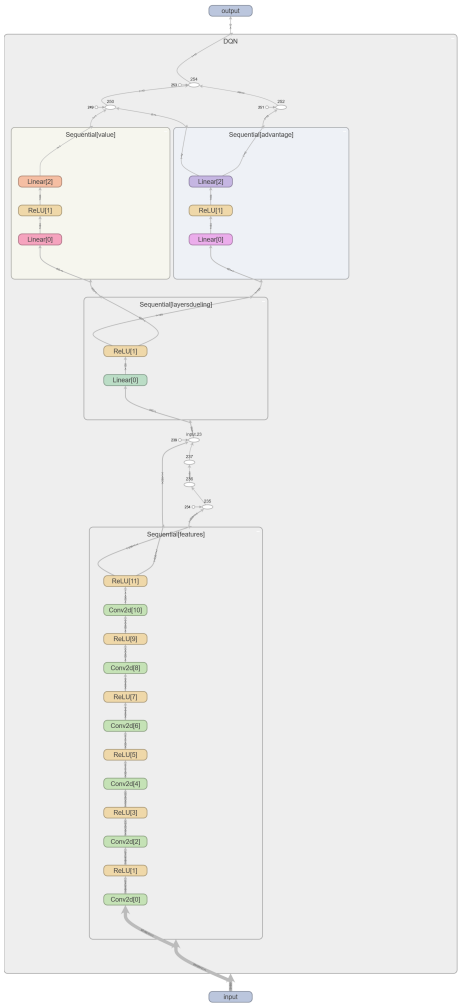

Figure 1: Architecture Part 1

off again. Compared to the first training the process had 4 drops in between, where the performance quickly went down but the model recovered fairly well. In the end, the overall performance is also a bit worse compared to the first approach, but the difference is not significant. This can be seen in the Colab notebook, which accompanies this report.

In the following, we evaluate the DQN training with dueling layers. The introduction of those layers helped to smoothen the

*Both authors contributed equally to this research.

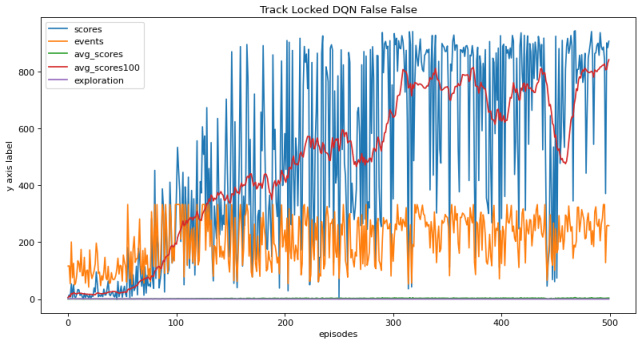

Figure 2: DQN Training

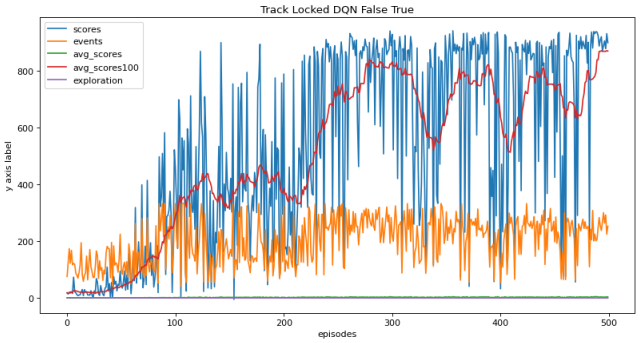

Figure 4: Dueling DQN Training

Randomization	DQN	DoubleDQN	DuelingDQN	DuelingDoub
1. False	881.34	654.23	844.07	885.66
2. False	647.57	833.73	857.69	873.48
3. False	906.99	595.96	896.02	932.19
4. False	714.73	892.97	850.13	869.21
5. False	163.82	839.30	942.39	653.95
1. True	-54.67	-35.26	-56.13	-83.69
2. True	-9.58	-69.98	-82.83	-78.22
3. True	-44.12	-80.84	23.43	-122.97
4. True	90.73	-59.45	-80.66	-40.04
5. True	264.38	247.06	-21.90	-27.59

Table 1: Evaluation Result Overview

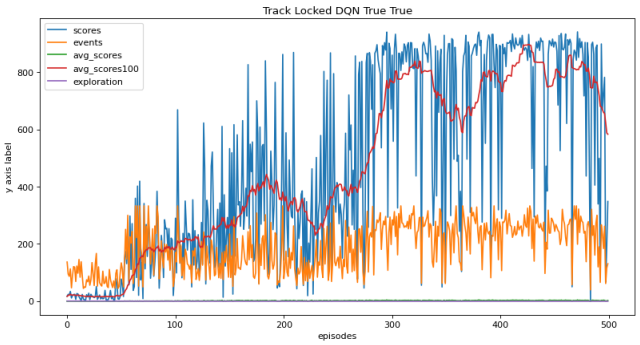

Figure 5: Dueling Double DQN Training

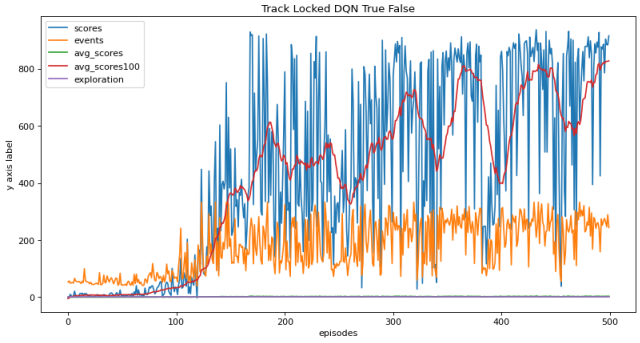

Figure 3: Double DQN Training

learning process. This can be seen in the plot we analyzed already for the other two approaches. It is depicted in figure 4. It needs to be noted that the learning process archives higher scores in a shorter training time. Additionally, the drops during the later stages of the training are less dramatic. The overall scores that the network scored are also a bit higher.

The last tested training method combines dueling layers with double DQN training. It scored the overall best results in our setup. The plot of the average score is shown in figure 5.

This curve is similar to the third approach. It needs, with the dueling layers, a bit longer to achieve good results. Noteworthy is that in this setup the drop in the late training process is not that steep. It also seems that the model is overfitting here for a bit because a steady downward trend shows in the last epochs.

Interesting insights can be found in table 1. Dueling Double DQN consistently achieves high scores across both randomized and non-randomized scenarios, suggesting robust performance. DQN and Double DQN show more variability in performance, with scores spanning a wider range. Randomization appears to have a significant effect on the performance of all architectures, especially noticeable in the negative scores for scenarios with randomization. Dueling Double DQN seems relatively more robust to randomization compared to the other architectures.

In addition, the test setup provides the option to train in discrete mode instead of the continuous one we used throughout the assignment. The qualitative of the plot doesn't change while using them. It is still a fairly steep increase in the beginning. Later on, drops are occurring for the average points achieved it can also be noted that the maximum is reached sooner in discrete control. It performs also a bit better in the evaluate method. It needs to be noted that it isn't as good for the random scenario as the network trained on different tracks we discuss in the next paragraph. The results for the known tracks are as good as with continuous control.

Overall we concluded that the best results achieved were with a combination of both training methods. This is also what we used for part 2 of the assignment. This is also the one we chose for retraining with randomization. The plot is shown in figure 6. It is to be noted

Randomization	Dueling Double DQN with randomization
1. False	504.29
2. False	462.71
3. False	81.91
4. False	503.57
5. False	319.67
1. True	272.28
2. True	455.73
3. True	25.72
4. True	484.46
5. True	47.34

Table 2: Evaluation Results: Dueling Double DQN with randomization

that the achieved score is way lower than training on the same track. A small increase can still be observed.

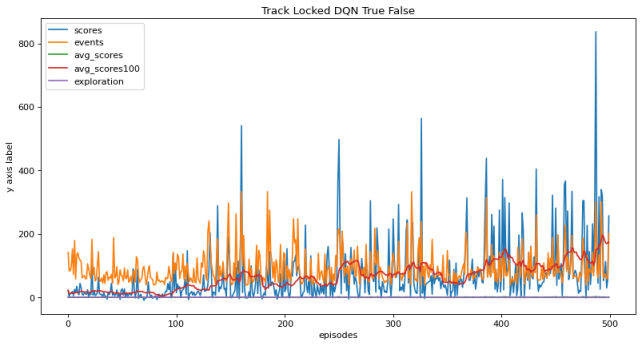

Figure 6: Dueling Double DQN with randomization

As expected it performs a bit better in the evaluation method especially with the tests on randomized tracks, because it was trained with them. This is probably caused by this training approach because in that more generalized features are learned. The score is shown in 2.

2 PART II:

In this part, Double DQN with Dueling was used. This type of model achieved the best results in the previous Part. Since we won't be providing images as an input to our model it only has Linear layers. This architecture is depicted in figure 7.

Figure 7: Architecture of the Network used in part II

This model was trained separately for two different scenarios:

(1) Scenario without obstacles
(2) Scenario with obstacles

Models trained on both scenarios were tested in multiple entry conditions (starting position and goal position). In the obstacle scenario, the input data was supplemented with an 11x11 local inline map. The provided code was modified only in one place in

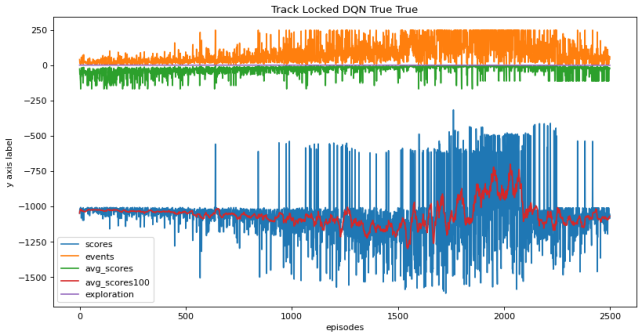

Figure 9: Dueling Double DQN Training on Scenario 2

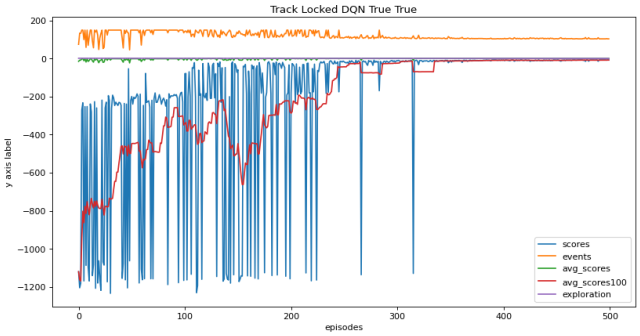

Figure 8: Dueling Double DQN Training on scenario 1

addition to the completion in the designated places - the value of *Learning Rate* was set to 0.0001.

2.1 Performance Comparision

As can be seen in the graph 8, the model managed the task of finding the way to the target in half of the episodes designated for it. Around episode 250 it almost got to the target position and for the next 50, he circled around it to finally get there. As with the scores in Part I, we're not sure what's causing the score to drop, probably due to the model becoming overconfident. On the other hand, it can achieve the goal of reaching the targeted position in each of the cases we tested. Both placing the launch position closer to the target and moving them away from each other were tested. As can be seen in the graph 9, the model failed to handle scenario 2.

The same pattern of behavior can be seen in each of the cases tested. The model correctly moves towards the target position while at a certain point, it starts to spin in circles. There may be possible explanations and suggestions to address this issue:

- Experience Replay Buffer Size is too small. It may affect the stability of learning as the model might not be learning from a diverse set of experiences.
- Learning Rate needs to be adjusted. If the learning rate is too high, the model may oscillate or overshoot the optimal values. If it's too low, the model may converge too slowly or get stuck in local minima.
- Exploration-Exploitation Tradeoff - the model may not be exploring enough to discover the optimal path to the target position. Adjusting exploration strategy may solve this problem.

3 GOOGLE COLAB WITH CODE

All code that was used can be found here: Assignment part I and Assignment part II