EARIN Exercise 6

Reinforcement Learning

Oliwia Pęczkowska 303878
Piotr Skibiński 303840

Warsaw, 5.06.2022

1. Introduction

In our task we need to set up an environment using open AI gym for ‘CarRacing-v0'. Then we choose and algorithm and we train with it our model with varying policies and compare obtained results.

2. Algorithms

There are ton of Reinforcement Learning Algorithms. We are going to use model freebase reinforcement learning algorithms. They are only using the current state values to try to make a prediction. The model-based reinforcement learning algorithms try to make a prediction about the future state of the model to generate the best possible action.

2.1. PPO

In our task we need to use algorithm with varying policies. We decided to use Proximal Policy Optimalization (PPO). It is a policy gradient method where policy is updated explicitly. PPO can be used for environments with either discrete or continuous action spaces. PPO is easy to implement and tune. It tries to compute an update at each step which minimizes the cost function while ensuring the deviation from the previous policy is relatively small. As a policy of our algorithm, we will use Convolutional Neural Network (CNN) policy model It can be used for complicated image classification. CNN takes tensor as input. It can understand relation between nearby pixels and is designed to work for images or videos classification. We also decided to use Multilayer Perceptron (MLP) policy model; however, it is not suitable for our task. MLPs are suitable for classification prediction problems where inputs are assigned a class or label.

2.2. DQN

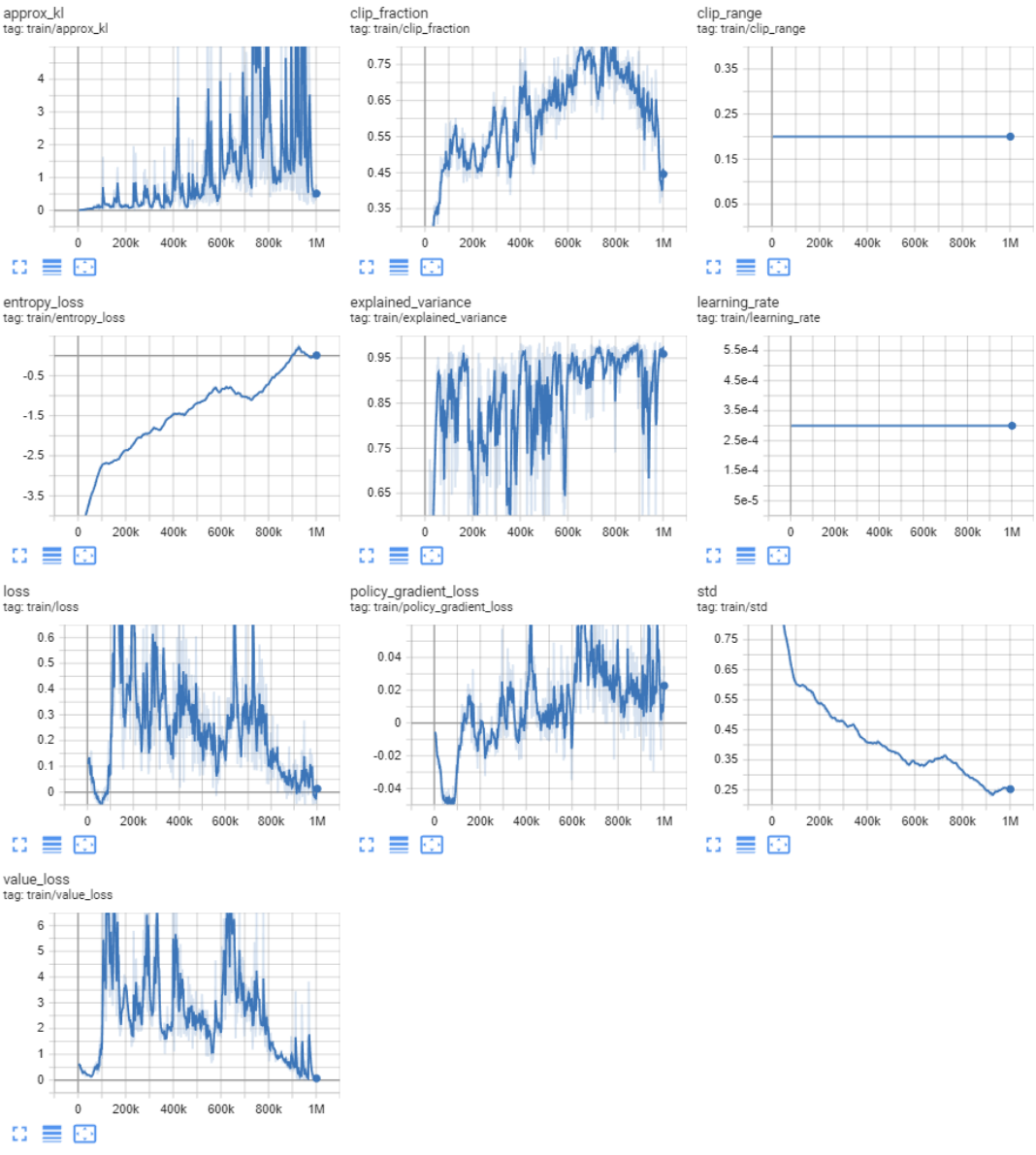
To compare we will use the Deep-Q Networks (DQN) algorithm, which is a variation of Q Learning. It is value-based algorithm. Using this algorithm, we create the neutral network acting like Q table. However, it does not contain Q values for every state and action. This algorithm only applies to discrete action and state spaces. In our game it is sufficient to use algorithm with discrete values because all variables of our car – speed, turning, breaking can be described in state form. DQN is usually slower to train but is the most sample efficient because of its replay buffer.

The Q value is $Q(s,a) = r(s,a) + \gamma(maxQ(s',A))$, where s-current state, s'-future state, a-particular action, A-action space. The Q value is calculated by adding rewards given for action during the state and the multiplication of the maximum Q value for given action space and future space by the discount rate. To maximize the rewards, we need to choose the highest possible Q value.

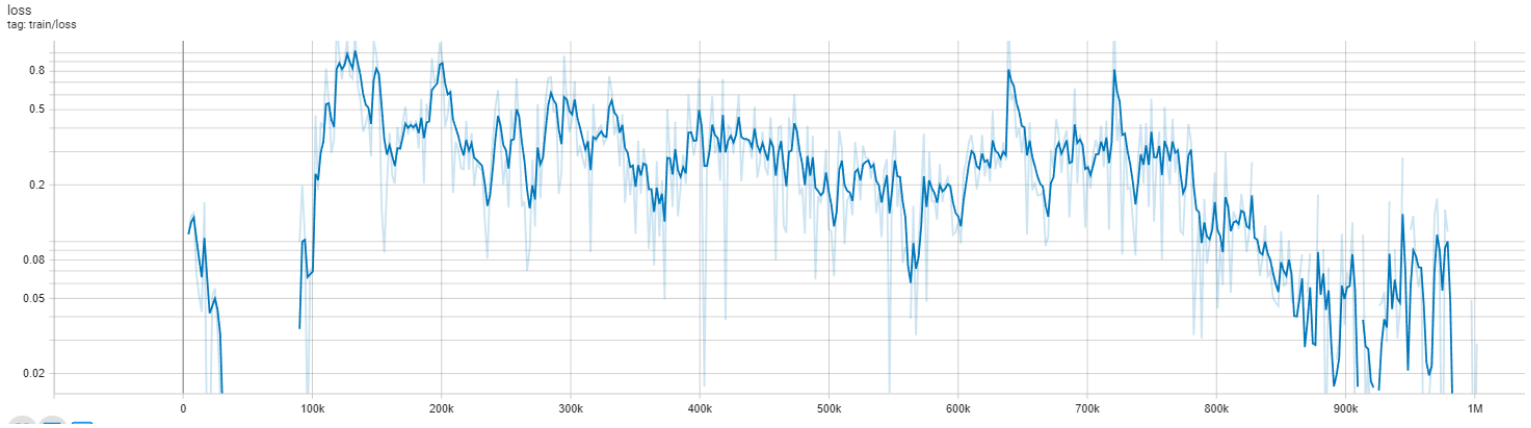
3. Comparison

Firstly, we have decided to train our PPO model with both MLP and CNN. Although we are aware that model train with MLP policy, will not be able to play the Car Racing game. We wanted to generate charts for both policies to better understand the behaviour of the algorithm. As it turned out later the CNN model for one million samples fell into local minimum in which it could not make a move, because it stopped using gas. As a result, the model with which we raised our greatest hopes turned out to be useless.

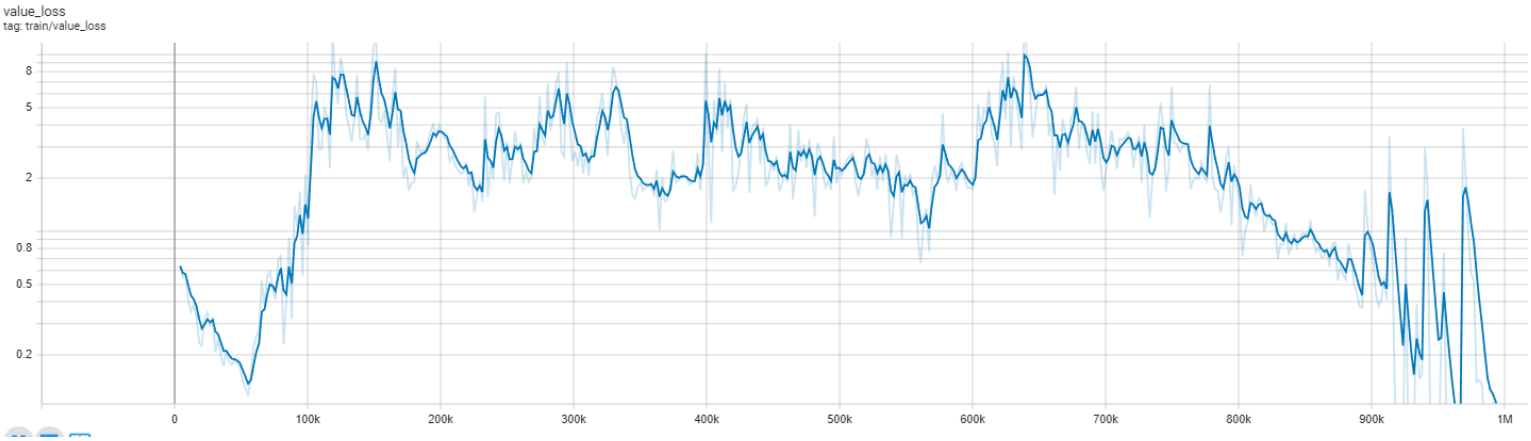
3.1. CNN model



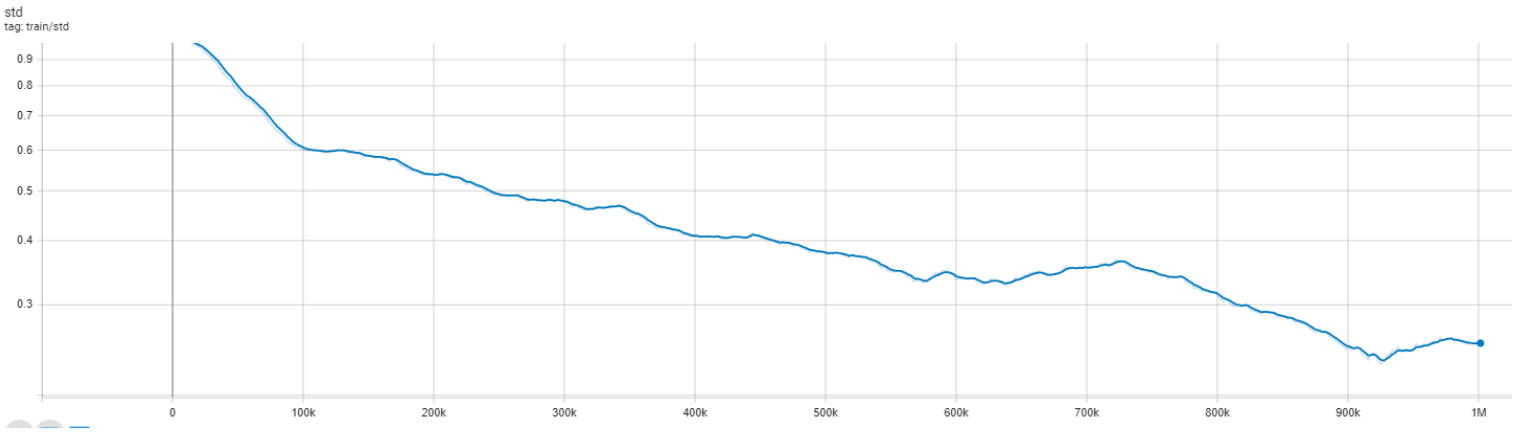
Loss function:



Value loss

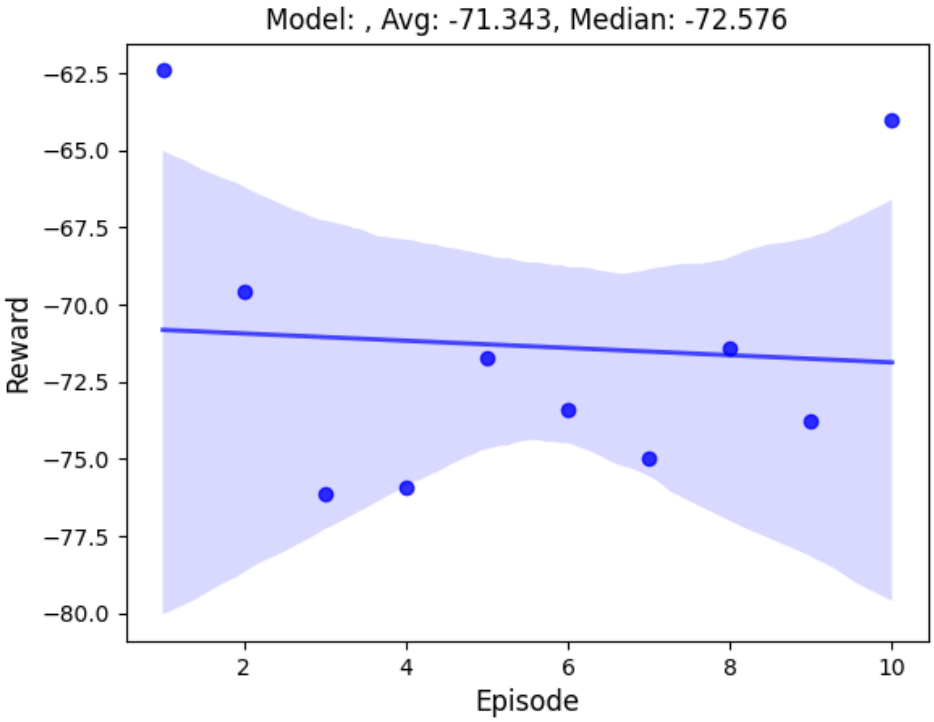


Standard deviation

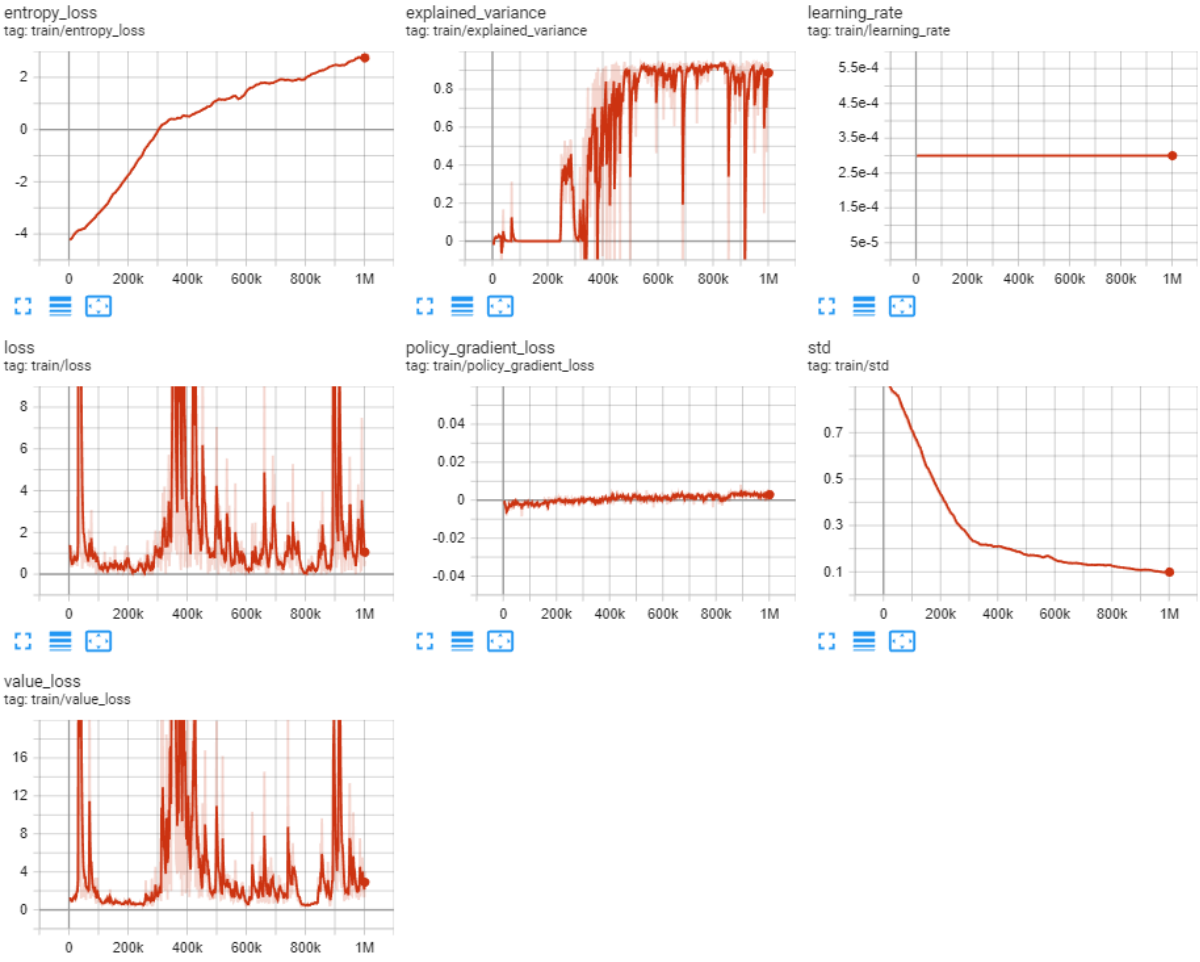


Results

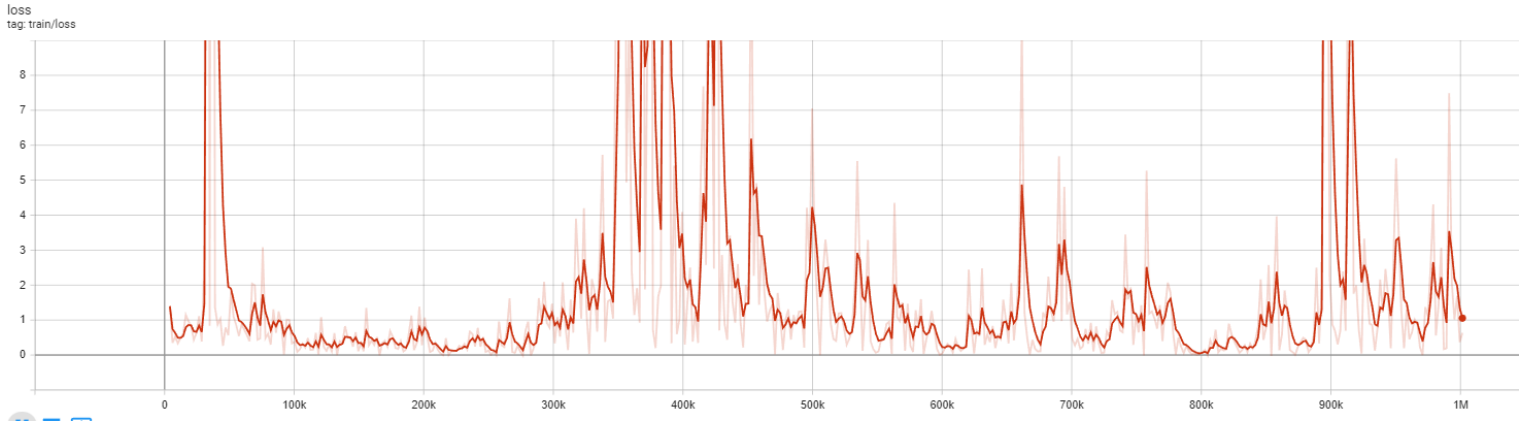
The results have shown that the model is not capable of playing the game.



3.2. MLP model

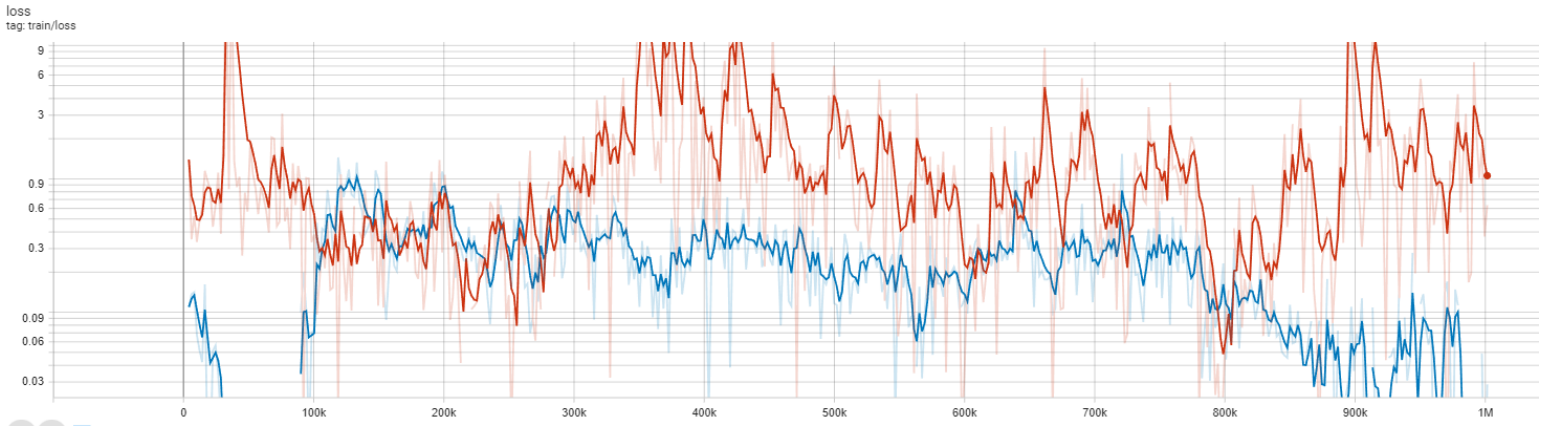


Loss function:

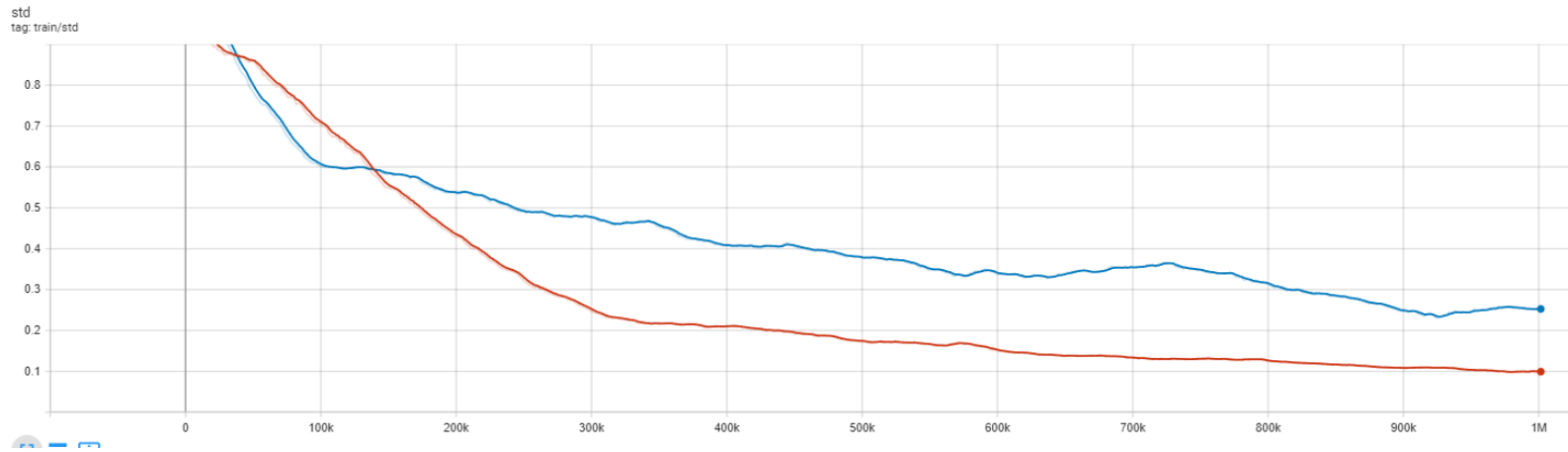


3.3. MLP vs CNN

Comparison the MLP(red) and CNN(blue) loss functions



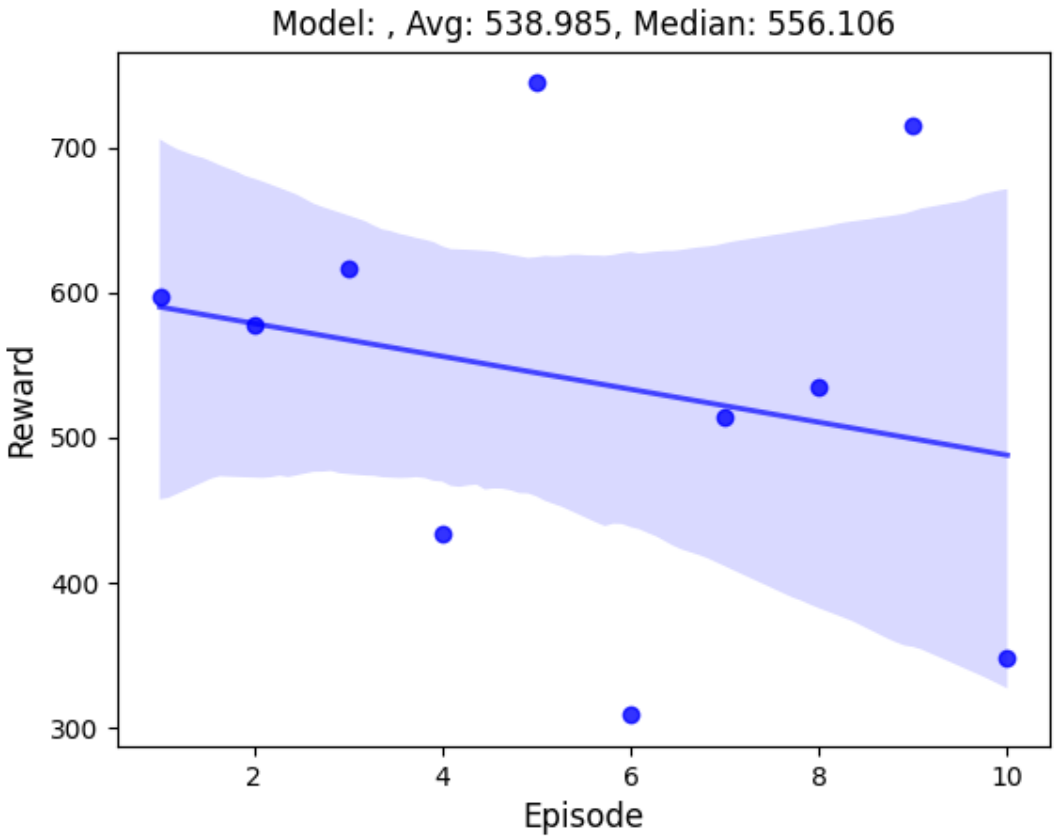
Standard deviation with comparison the MLP(red) and CNN(blue)



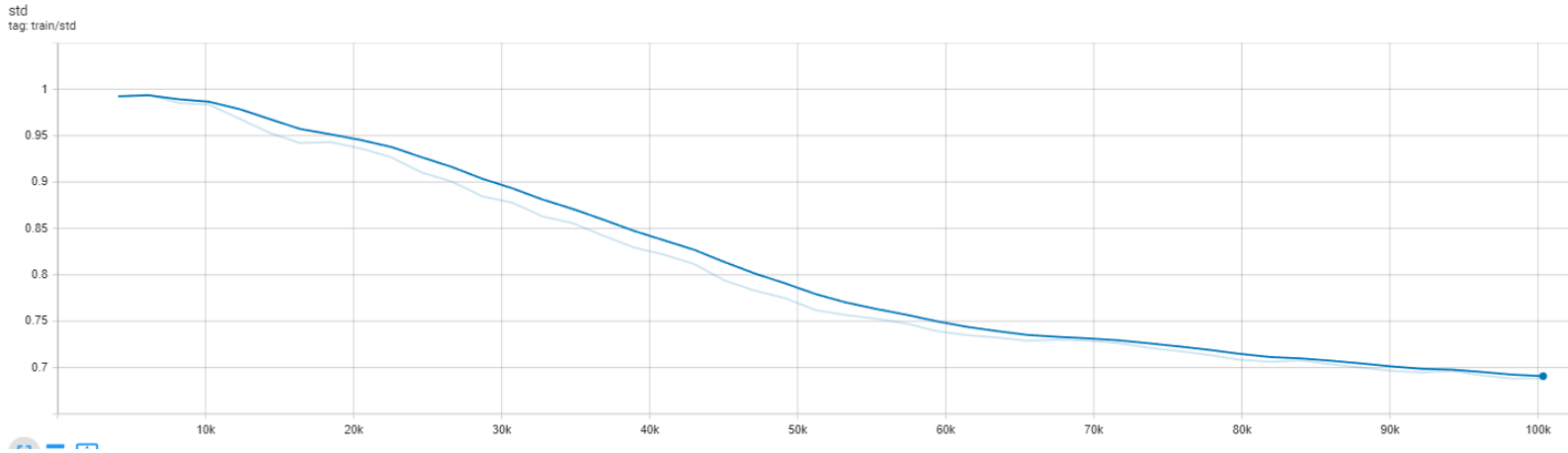
MLP has shown a similar result to the CNN model, because it was completely unable to play the Car Racing game. Something worth mentioning is that CNN policy (2h 12m) training took slightly longer than MLP(2h).

3.4. CNN 100k model

Unlike MLP, with CNN we have been capable of generating good models. Those are the results for a model with 100k samples.

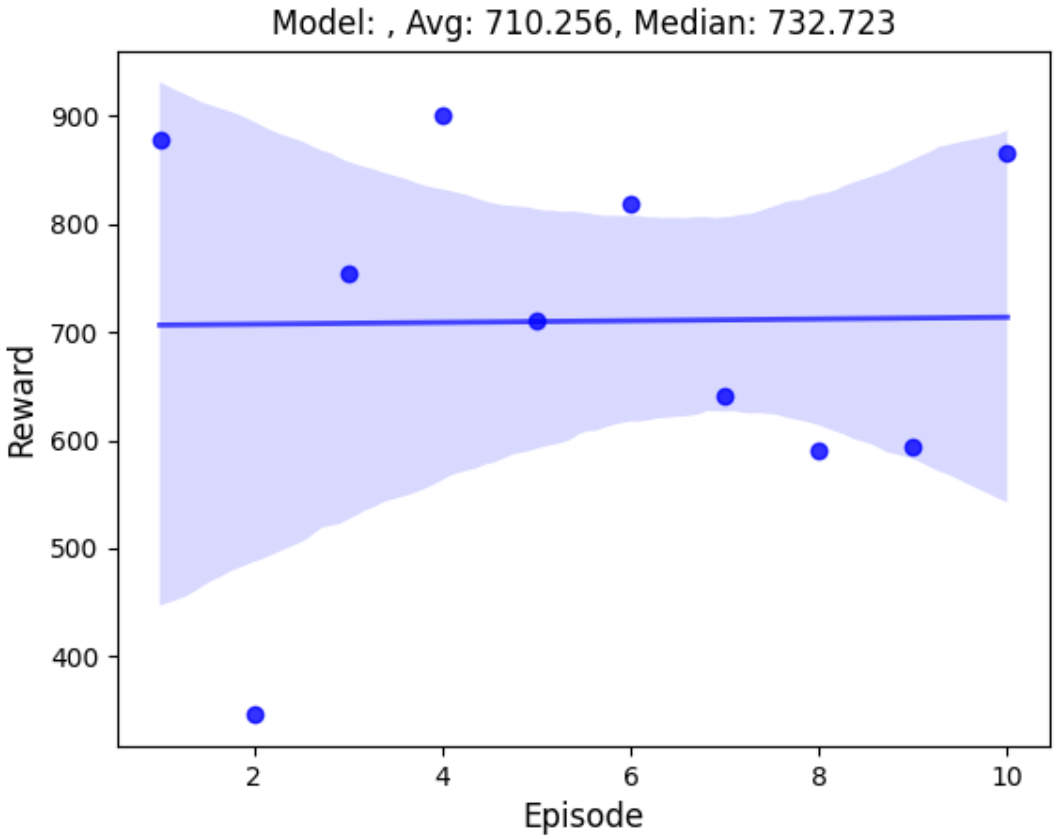


Standard deviation

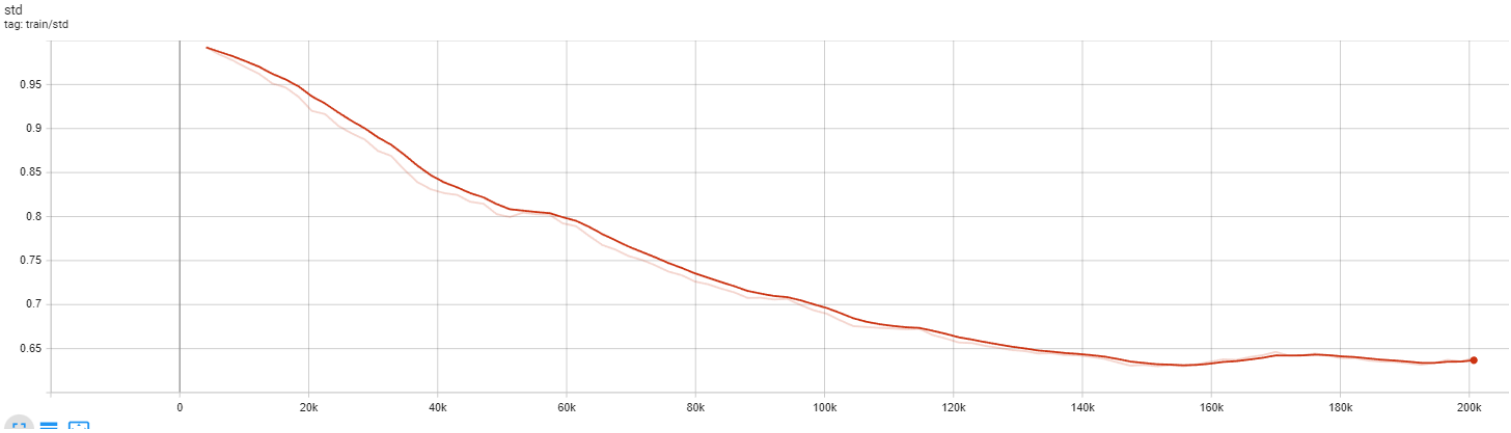


3.5. CNN 200k model

The 200k model has shown a significant improvements vs the 100k one. Results were much more stable and generally higher.



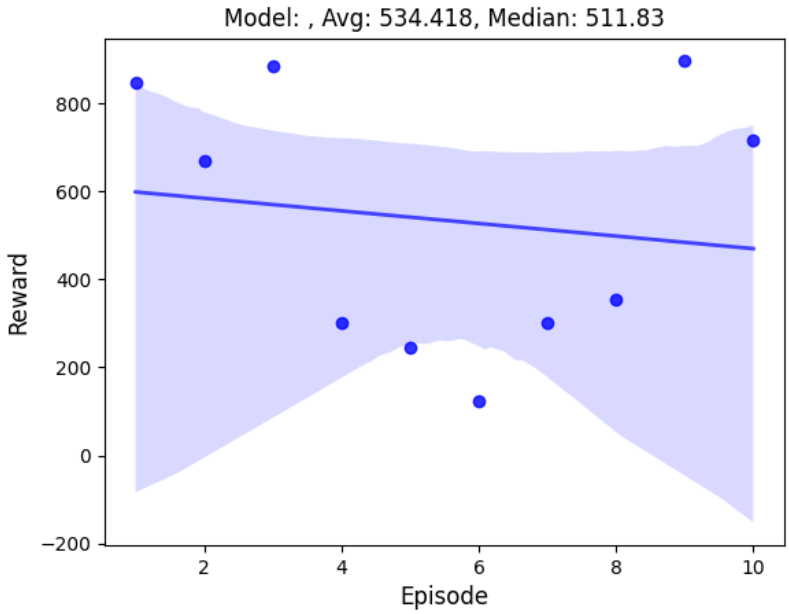
Loss function:



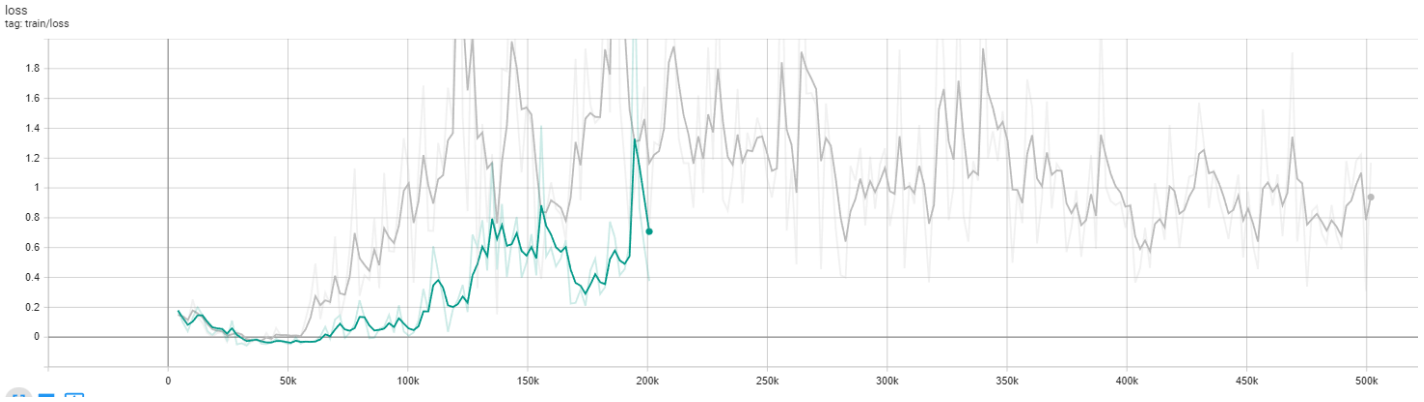
Train Time: 26min

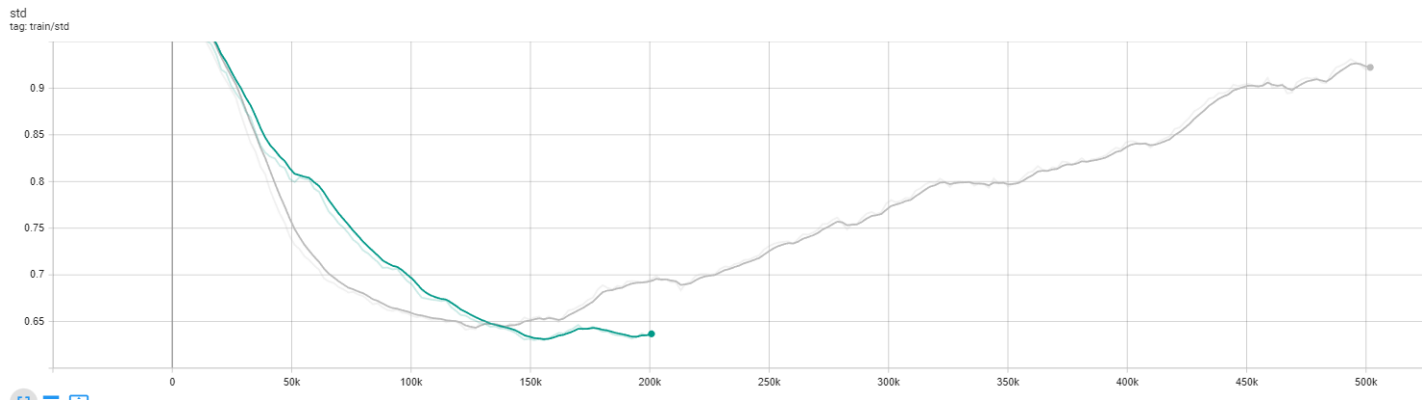
3.6. CNN 500k model

This model generated worst results in comparison to the 200k model.



Loss and standard deviation function (grey) vs the 200k model(green):





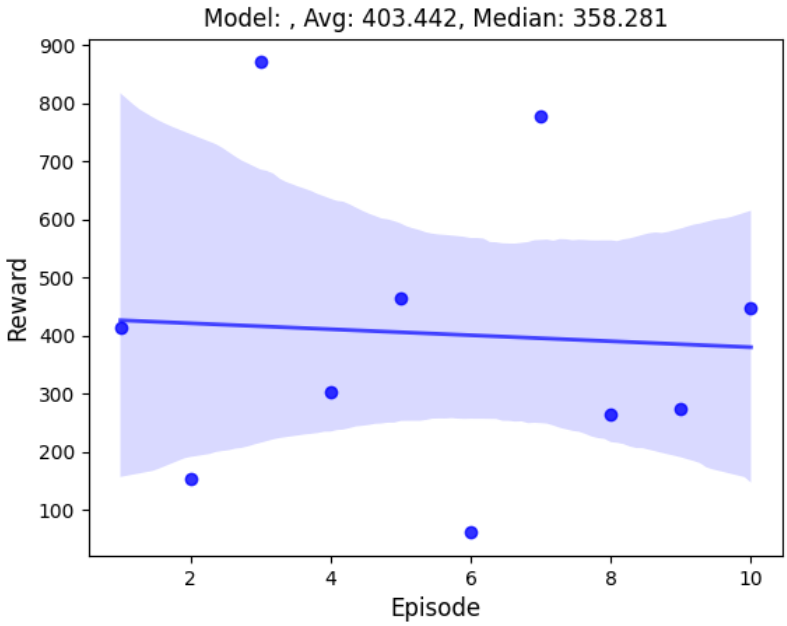
This time something strange started happening in our standard deviation graph

Evaluation output:
Timesteps: 500000, Evaluation output: (621.4161549799144, 254.38049481038493)

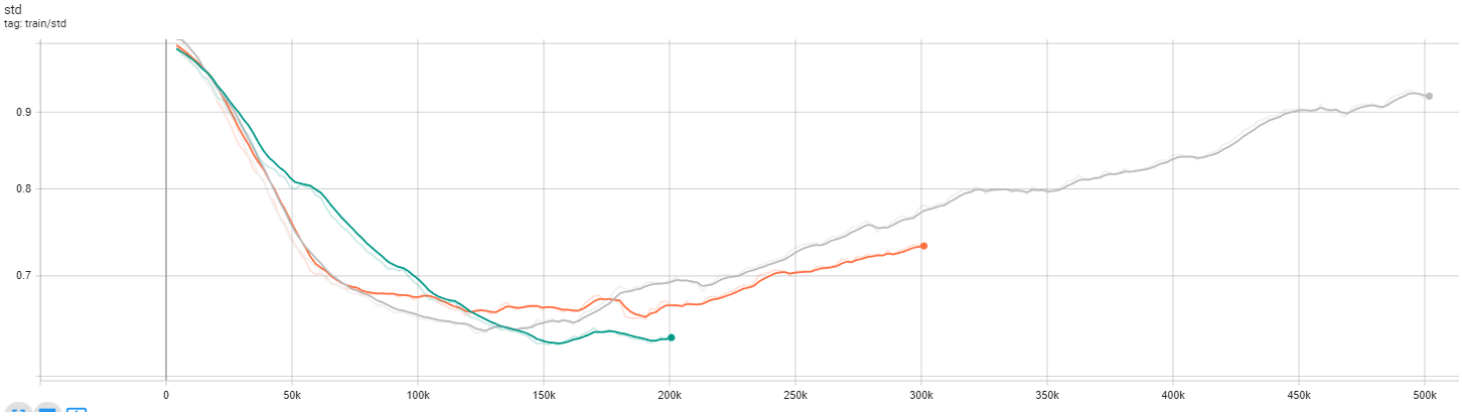
Learn time: 1h7m

3.7. CNN 300k model

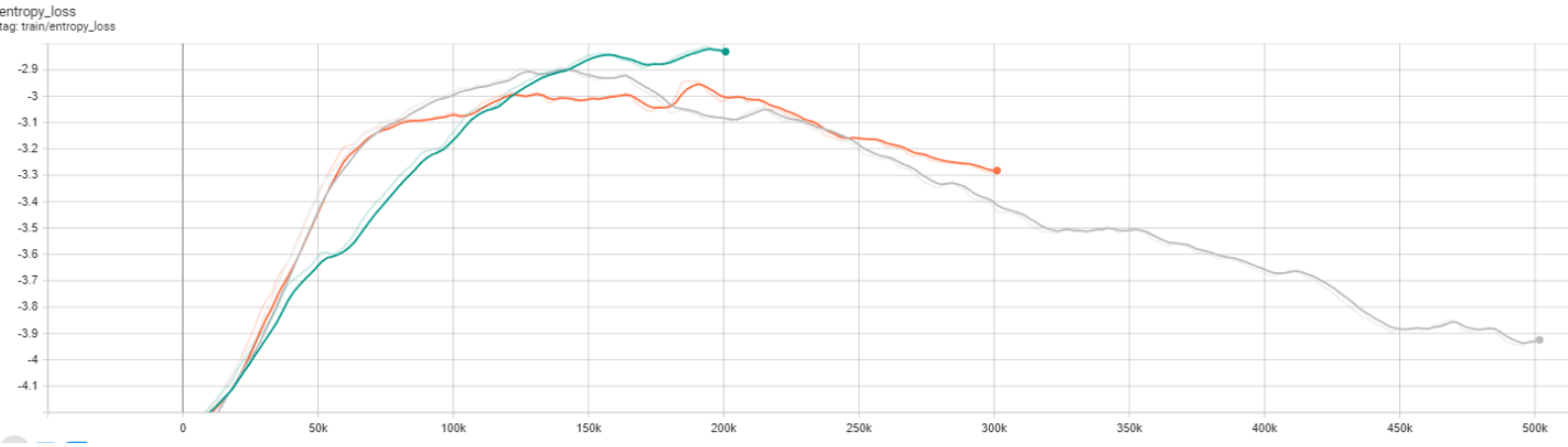
To our surprise, the 300k model did show a significant drop in performance against the 200k and 500k models. Probably around that point there is a local minimum.



Standard deviation 200k vs 300k vs 500k



Finally, the entropy loss function.

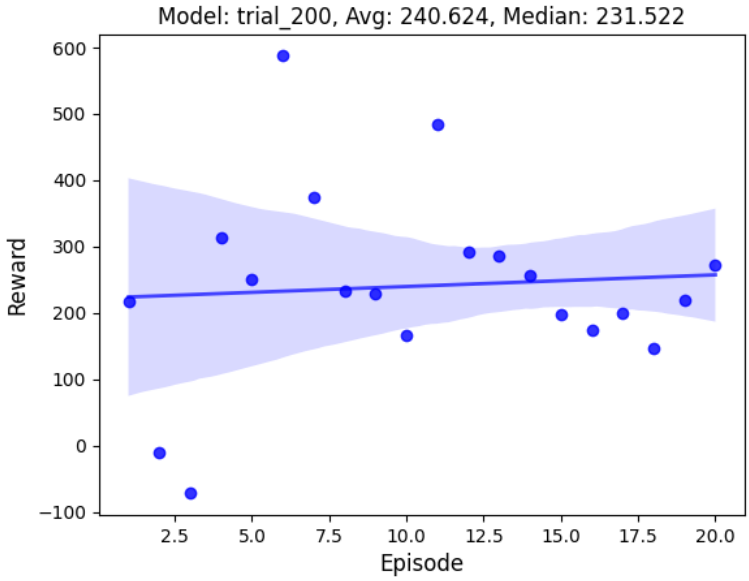


Evaluation output:
Timesteps: 300000, Evaluation output: (498.7045661523938, 239.83417406354712)

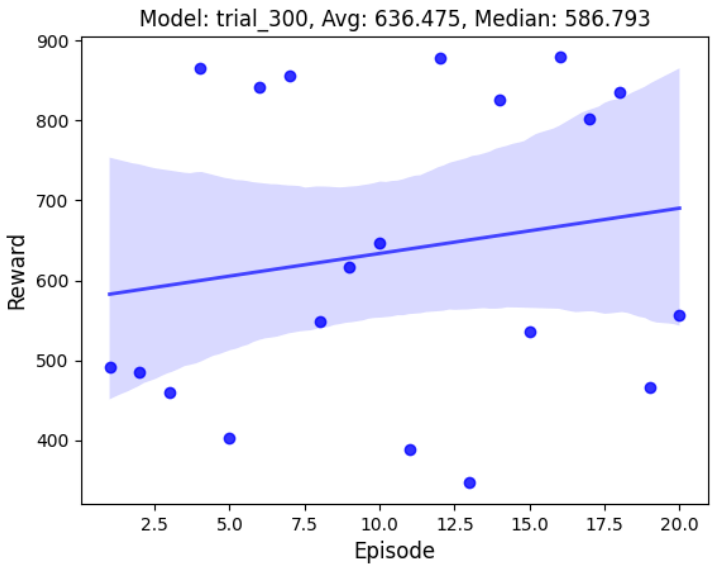
3.8. DQN

Lastly, we also tried using the DQN algorithm. Unfortunately, the stable version baseline did not work for us. We implemented the algorithm in a slightly more manual manner. Training for the 300 episodes took around 6 hours. The results were satisfactory.

The graphs that show the 200 episodes of learning. Each episode consists of batch size of 16:



Improvement after 300 episodes:



4. Conclusion

4.1. Proximal Policy Optimization

4.1.1. MLP Policy

As we suspected this policy did not manage to train the model. It is not designed for this type of problem.

4.1.2. CNN Policy

Training of this model turned out to be slightly counterintuitive. In the end this policy turned out to give the best results. With the average score of 710 points. After analysis the best model was the one with 200k steps.

4.2. Deep Q-learning

The DQN is significantly slower than the PPO. For the 200-300 episode we have received a model capable of playing the Car Racing game with results of average 636 points.