Names: Priel Kaizman (315627448), Adir Serruya (316472455)

DRL – Assignment 3

Section 1

The architectures we implemented in this section are different between the environments, for the CartPole and Acrobot environments we used an actor network with a single hidden layer that has 32 units and a single output and for the MountainCarContinuous environment we used a single hidden layer that has 256 units and 2 outputs (mu and sigma). For the critic network we used the same architecture for all environments, 2 hidden layers with 256 units each.

In addition, we used different hyper-parameters, for the CartPole and Actobot environments we used a learning rate of 0.0004 and we limited each episode to 501 steps, compared to a learning rate of 0.0001 and a limit of 1000 steps per episode for the MountainCarContinuous environment.

Lastly, we modified the loss function for the MountainCarContinuous environment to match the continuous action space.

The reason for the differences in architecture and parameters between the environments is to accommodate for the difference in discrete versus continuous action spaces and to prevent the car in the MountainCarContinuous problem from getting stuck at the bottom (local minimum close to 0 reward).

We used a limit of 1000 episode and a discount factor of 0.99 for all 3 environments.

**We stopped the MountainCarContinuous training at episode 500 since it was already stuck*

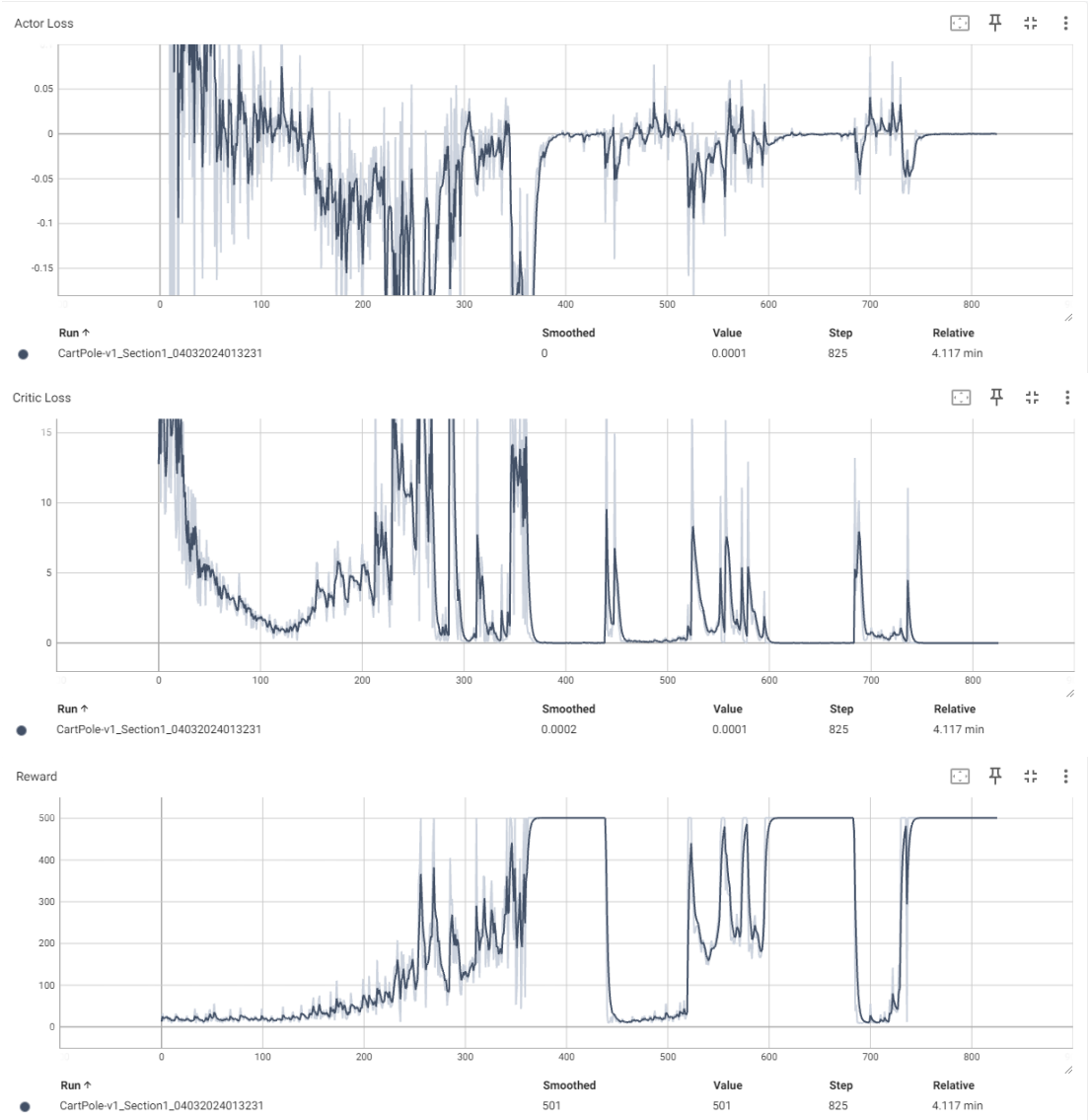
Results:

Environment	Convergence time	Episodes until solved
CartPole	247	825
Acrobot	70	103
MountainCarContinuous	N/A	N/A

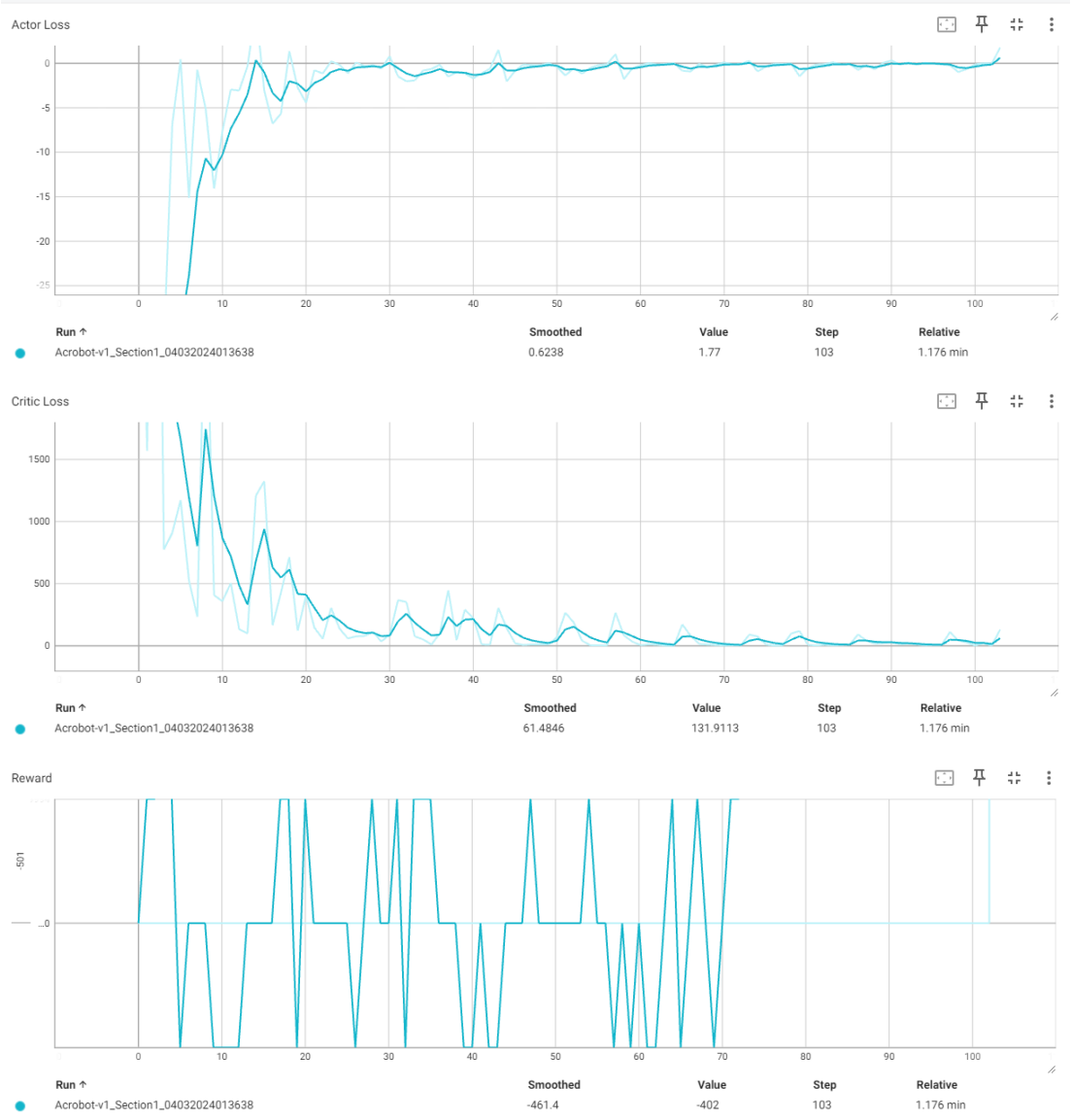
We were unable to solve the MountainCarContinuous problem, after trying many combinations of different hyper-parameters and architectures we were able to overcome the local minimum at the bottom of the hill (stuck at a reward of 0), but the car was still unable to reach the top of the mountain even after learning to

swing from side to side, even when we attempted to implement intrinsic rewards.

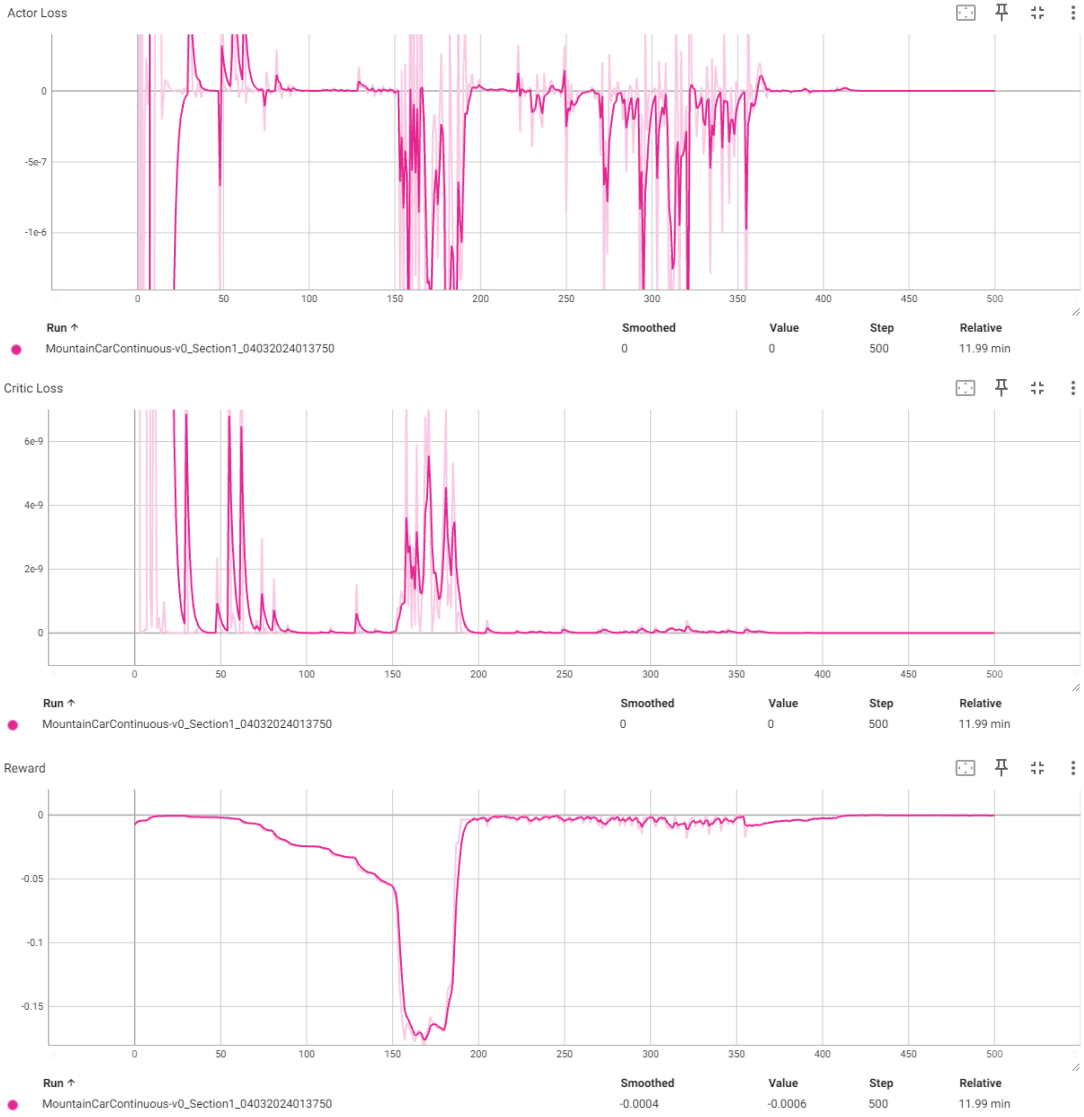
CartPole



Acrobot


Actor Loss
0
-5
-10
-15
-20
-25
0
10
20
30
40
50
60
70
80
90
100
Run
Smoothed
Value
Step
Relative
Acrobot-v1_Section1_04032024013638
0.6238
1.77
103
1.176 min
Critic Loss
1500
1000
500
0
0
10
20
30
40
50
60
70
80
90
100
Run
Smoothed
Value
Step
Relative
Acrobot-v1_Section1_04032024013638
61.4846
131.9113
103
1.176 min
Reward
-501
...0
0
10
20
30
40
50
60
70
80
90
100
Run
Smoothed
Value
Step
Relative
Acrobot-v1_Section1_04032024013638
-461.4
-402
103
1.176 min

MountainCarContinuous



Section 2

In this section we used the same actor and critic network architectures and hyper-parameters as described in Section 1.

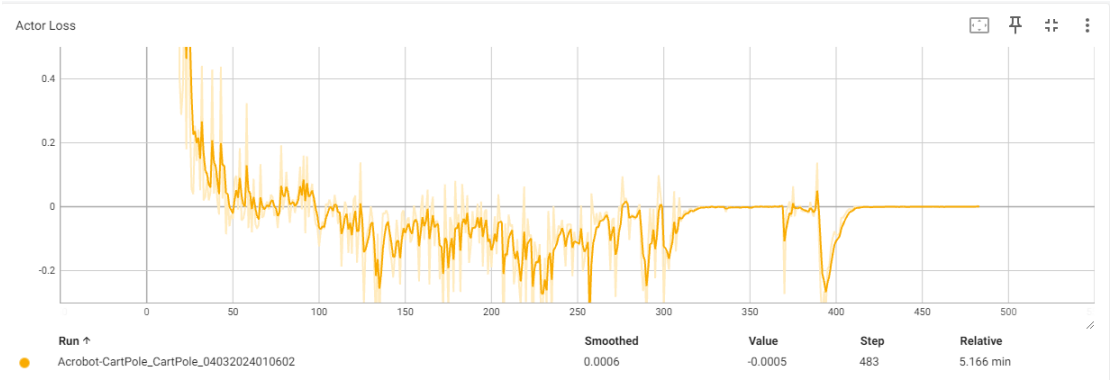
Results:

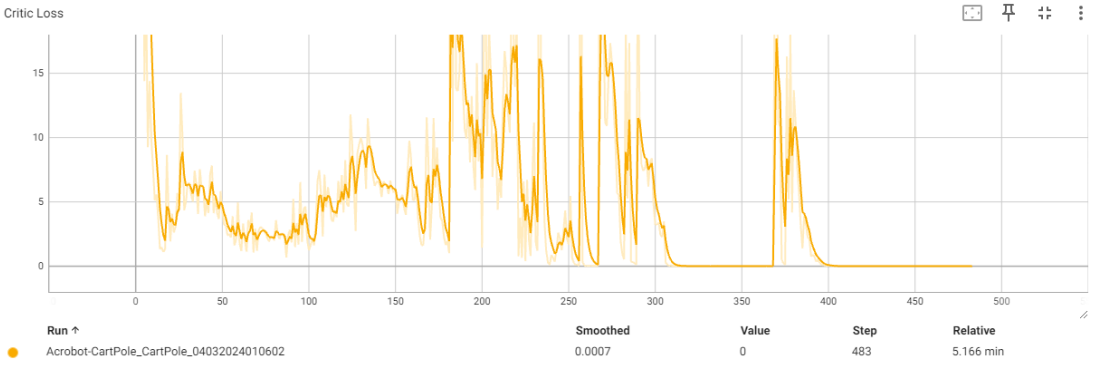
Environment	Convergence time	Episodes until solved
CartPole	309	483
MountainCarContinuous	N/A	N/A

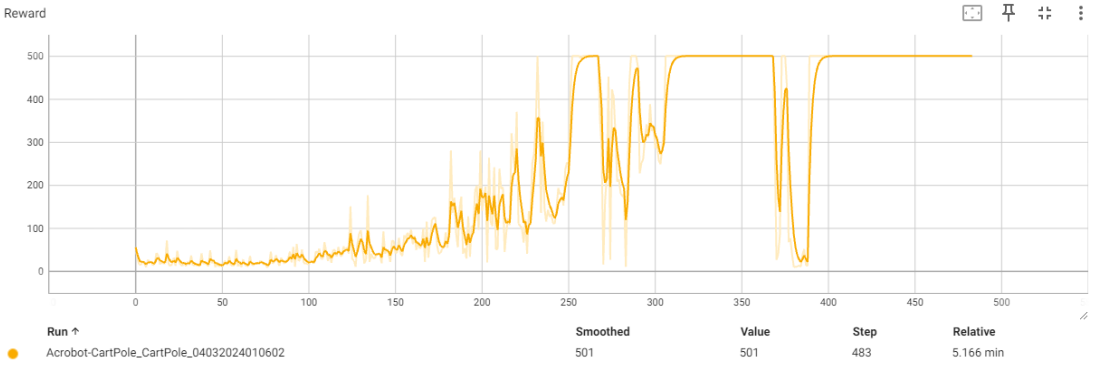
For the CartPole environment we saw drastic improvement, it took half as many episodes to solve (reach an average reward of 475 over 100 episodes), in addition, it reached the max reward of 501 within ~250 episodes, compared to ~350 episodes in the previous section.
We can conclude that using the pre-trained Acrobot model was beneficial since it learned generic features that were applicable to the CartPole environment, making the training of CartPole begin with optimized weights.

In this section again we were unable to solve the MountainCarContinuous environment, however the behavior in the training process was drastically different than the previous section. The car immediately learned to swing back and forth, it got high on the mountain but eventually got stuck picking a side and only trying to climb that specific side without swinging anymore, this happened in all training attempts we made. In this section again, we tried different hyper-parameters and architectures but eventually kept the same architecture as in Section 1 for clearer comparison and since we saw no improvement with other architectures.

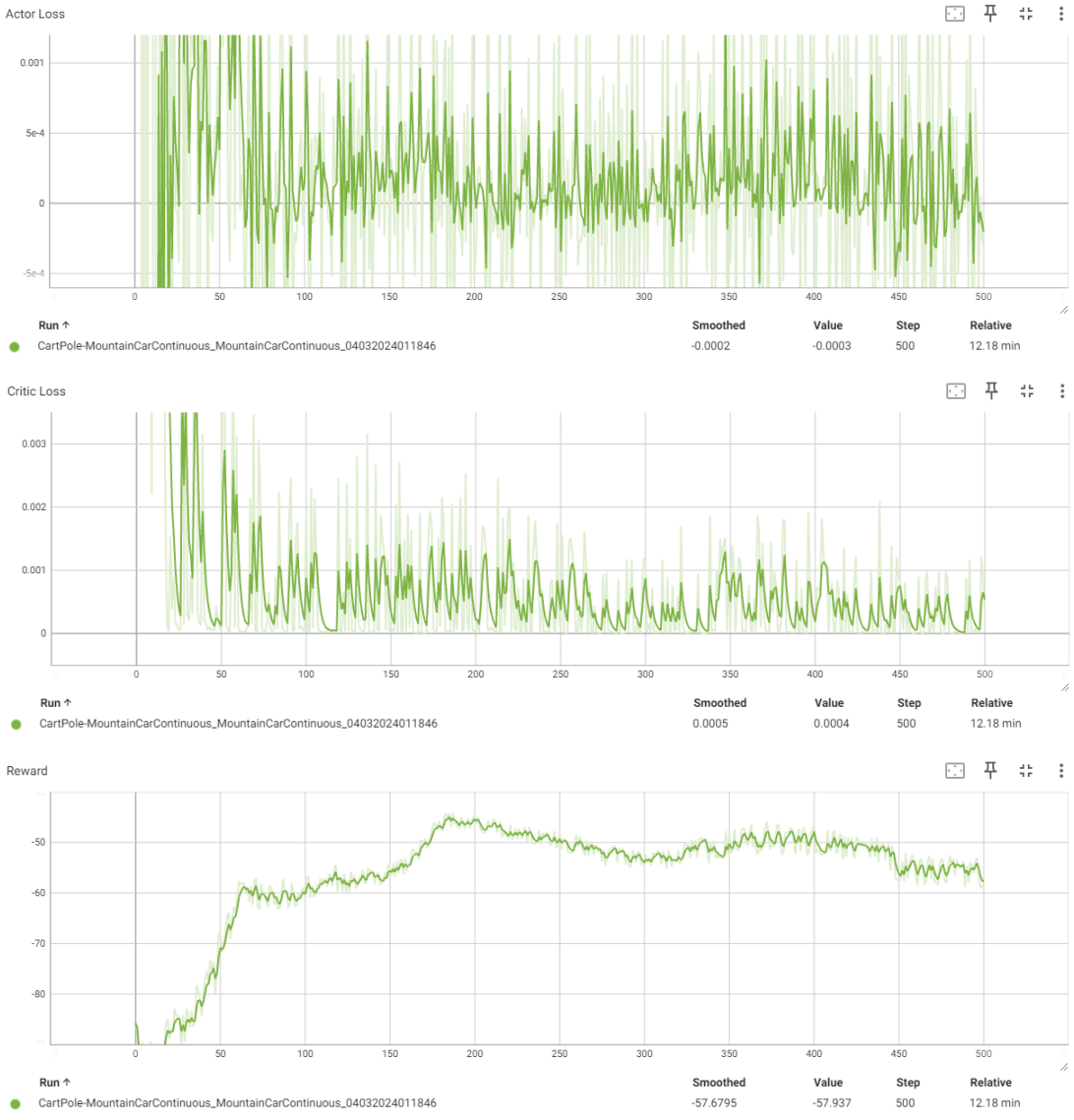
CartPole


Actor Loss
0.4
0.2
0
-0.2
0
50
100
150
200
250
300
350
400
450
500
Run
Smoothed
Value
Step
Relative
Acrobot-CartPole_CartPole_04032024010602
0.0006
-0.0005
483
5.166 min


Critic Loss
15
10
5
0
0
50
100
150
200
250
300
350
400
450
500
Run
Smoothed
Value
Step
Relative
Acrobot-CartPole_CartPole_04032024010602
0.0007
0
483
5.166 min


Reward
500
400
300
200
100
0
0
50
100
150
200
250
300
350
400
450
500
Run
Smoothed
Value
Step
Relative
Acrobot-CartPole_CartPole_04032024010602
501
501
483
5.166 min

MountainCarContinuous



Section 3

For this section we decided to use Tensorflow v2.x instead of v1.x, which was used in the previous sections. The reason for the change is that we encountered many issues trying to implement the architecture with the old tf version.
Since we changed the tf version, we will compare the results of the training with and without transfer learning on the models created using Tensorflow v2.x, instead of comparing to the results of previous sections.

The architecture of the network we used:
Actor – single hidden layer with 32 units for CartPole and Acrobot, 256 units for MountainCarContinuous.
Critic – 2 hidden layers, both with 128 units for CartPole and Acrobot, 256 units for MountainCarContinuous.
The structure of the progressive network follows the instructions in this section.

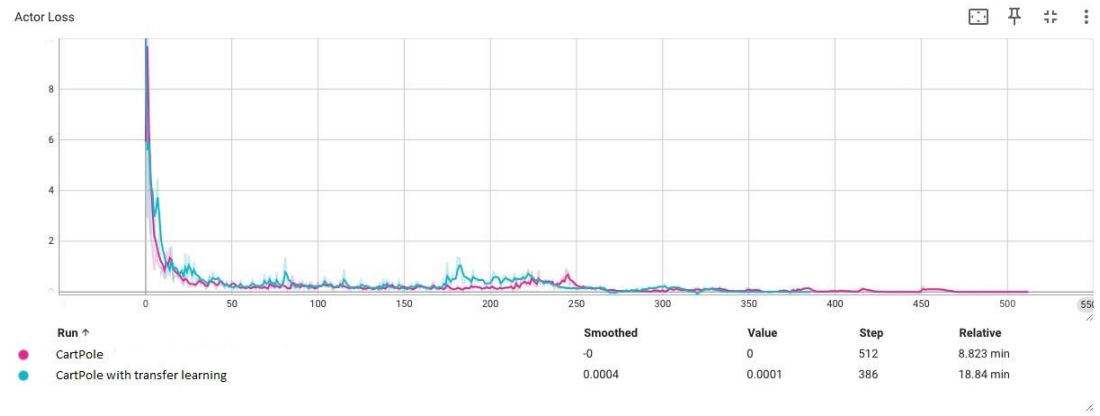
Hyperparameters:
learning rate = 0.0005
discount factor = 0.99

Results:

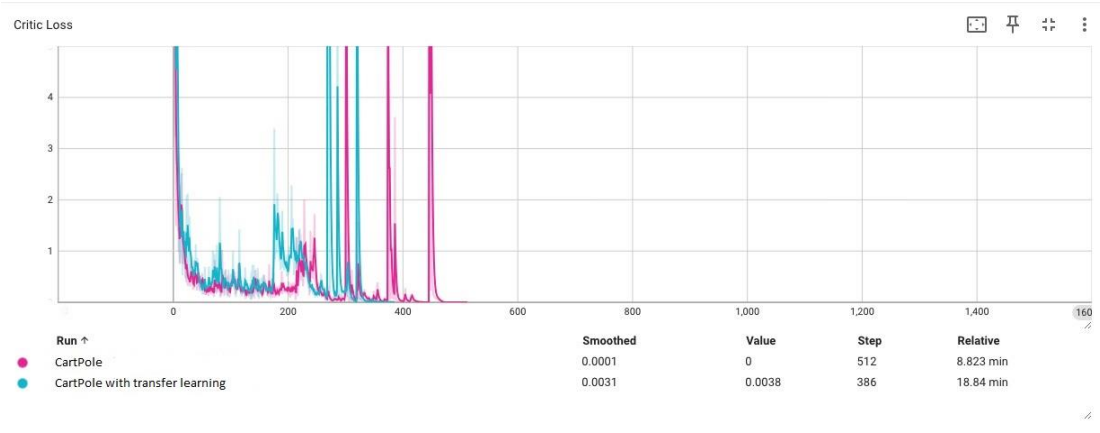
Environment	Convergence time	Episodes until solved
CartPole	8.82 minutes	513
CartPole with transfer learning	18.84 minutes	387
Acrobot	1.7 minutes	103
MountainCarContinuous	20.85 seconds	4
MountainCarContinuous with transfer learning	<20 seconds	1

Based on the results, it's clear that transfer learning has significantly improved the efficiency of solving the target environments, although with a longer convergence time for CartPole (due to the size of the network). The episodes until solved were much lower for both environments.

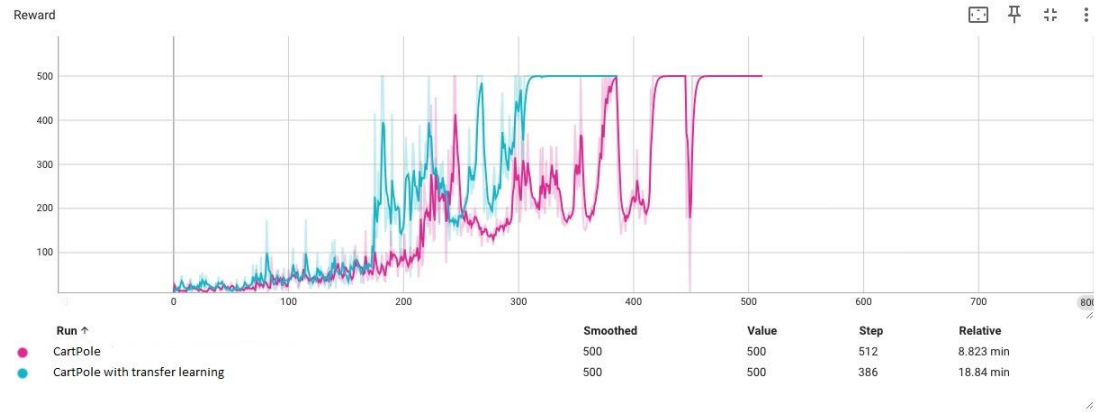
CartPole



Run	Smoothed	Value	Step	Relative
CartPole	-0	0	512	8.823 min
CartPole with transfer learning	0.0004	0.0001	386	18.84 min

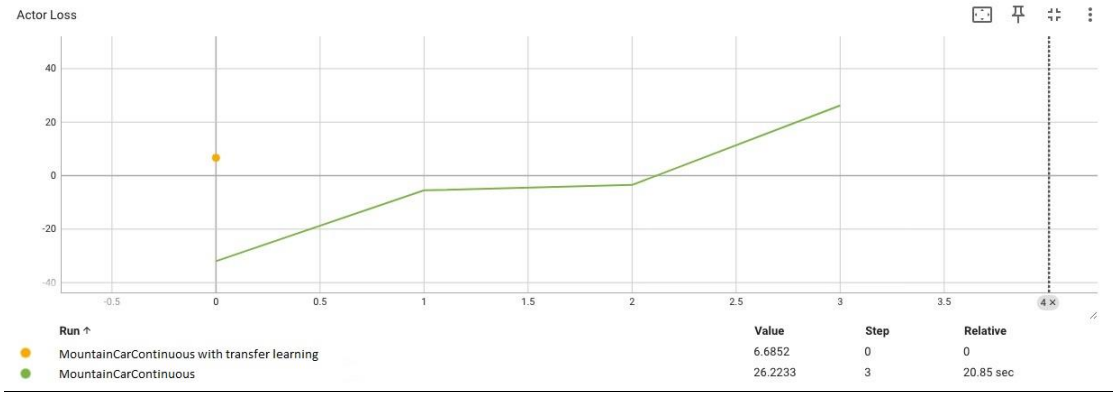


Run	Smoothed	Value	Step	Relative
CartPole	0.0001	0	512	8.823 min
CartPole with transfer learning	0.0031	0.0038	386	18.84 min

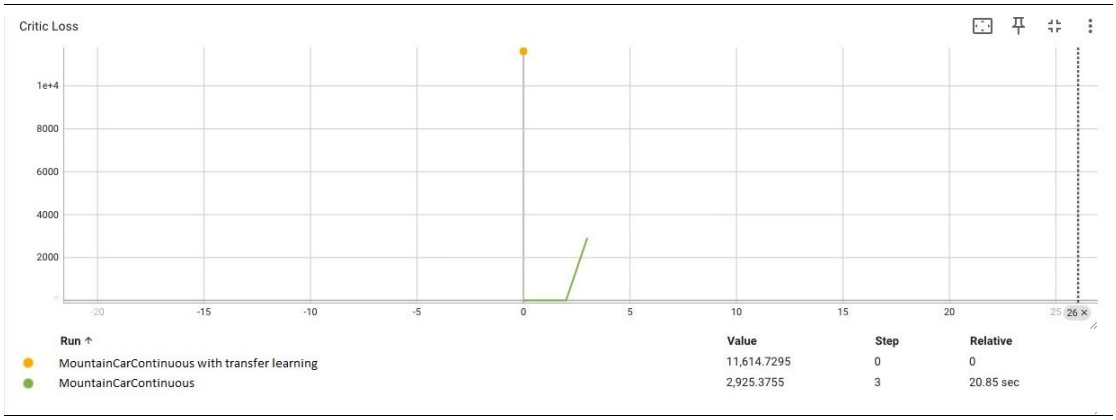


Run	Smoothed	Value	Step	Relative
CartPole	500	500	512	8.823 min
CartPole with transfer learning	500	500	386	18.84 min

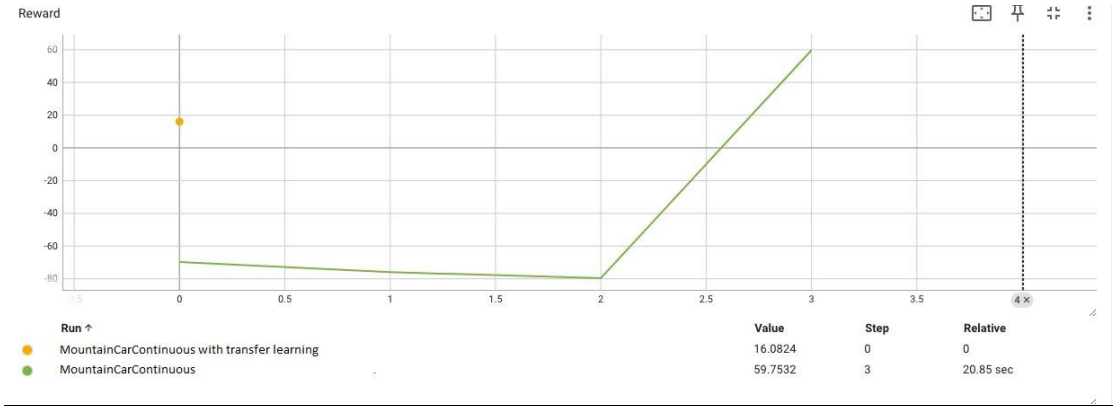
MountainCarContinuous



Run	Value	Step	Relative
MountainCarContinuous with transfer learning	6.6852	0	0
MountainCarContinuous	26.2233	3	20.85 sec



Run	Value	Step	Relative
MountainCarContinuous with transfer learning	11,614.7295	0	0
MountainCarContinuous	2,925.3755	3	20.85 sec



Run	Value	Step	Relative
MountainCarContinuous with transfer learning	16.0824	0	0
MountainCarContinuous	59.7532	3	20.85 sec

Instructions for running the scripts

1. Preliminary Setup:

Ensure that all required packages are installed in your environment, these dependencies are listed in the requirements.txt file.
You can install them using the command: pip install -r requirements.txt

2. Script Descriptions:

- Section1.py: the script for section 1.

- Section2.py: the script for section 2.

- Section3.py: the script for section 3.

3. Running the Scripts:

- To execute a script, simply run it in your Python environment. For example by using the command: python Section1.py

- Repeat the process for Section2.py and Section3.py as needed.