**Deep Reinforcement Learning**

**Assignment 3 – Meta and Transfer Learning**

Or Gindes 201640042

Avi Tetruk 205419377

Submission Date: 15/03/24 (extended to 22/03/24)

Code instructions -

Set up an environment from the ./environment.yaml and **run**:

* individual\_networks.py for **section 1.**
* fine\_tune\_models.py for **section 2.**
* progressive\_networks.py for **section 3.**

from the IDE

1. **Section 1 – Training individual networks**

Architecture and training parameters used for all networks -   
MAX\_EPISODES = 5000  
DISCOUNT\_FACTOR = 0.99  
LEARNING\_RATE = 1e-3

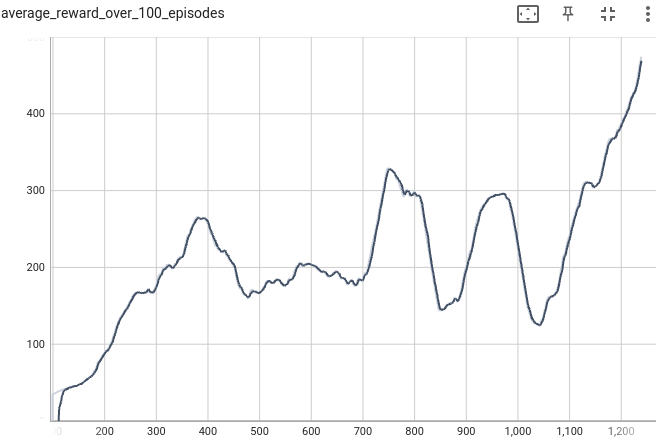
This learning rate refers to the actors lr, while the critics was set at a ratio of 10\*actor\_lr (in this case 1e-2).

The standardized state\_size for all of the environments in this problem was set to 6 which is the largest between them and the states of the other two environments were padded with zeros to fit. Similarly, for the action distribution, the size was set to 3 and for environments with less action options the remaining actions were masked so they would not be selected.

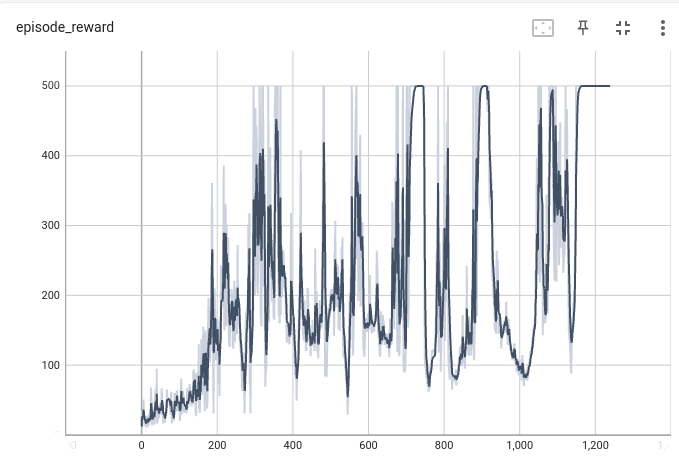
To train the individual networks simply run the script individual\_networks.py

* **CartPole -** First we’ll retrain the actor-critic architecture for the CartPole environment. Convergence threshold was set as 475, as with previous assignments. With the aforementioned parameters, the problem was solved at episode 1240 -  
    
  Episode 1236 Reward: 500.0 Average over 100 episodes: 463.78  
  Episode 1237 Reward: 500.0 Average over 100 episodes: 467.53  
  Episode 1238 Reward: 500.0 Average over 100 episodes: 471.16  
  Episode 1239 Reward: 500.0 Average over 100 episodes: 474.57  
  Episode 1240 Reward: 500.0 Average over 100 episodes: 478.04  
  Solved at episode: 1240

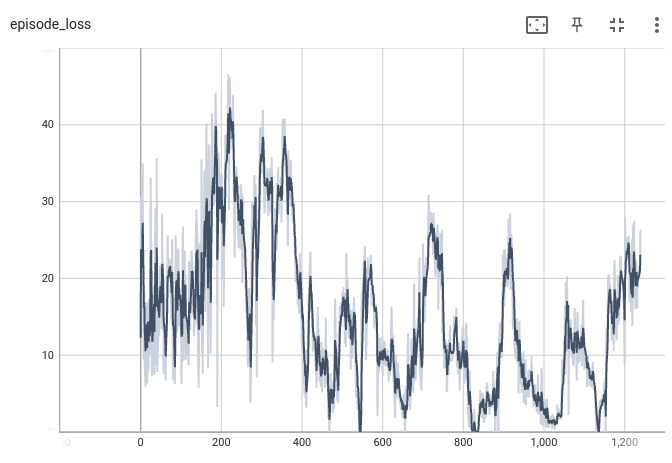
Average Reward of 100 consecutive episodes -



Episode Reward

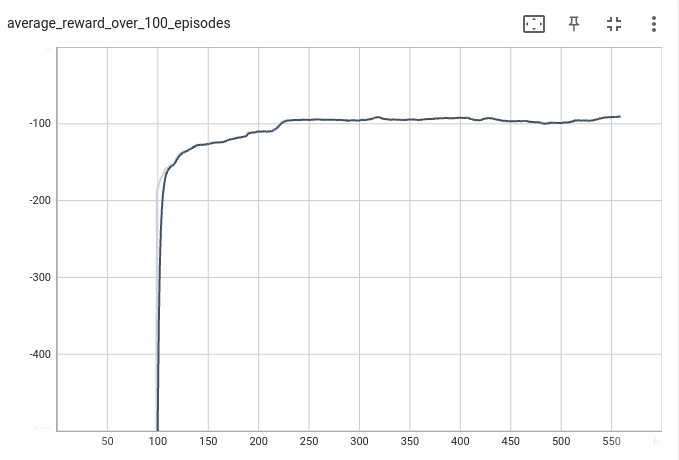


Episode Loss

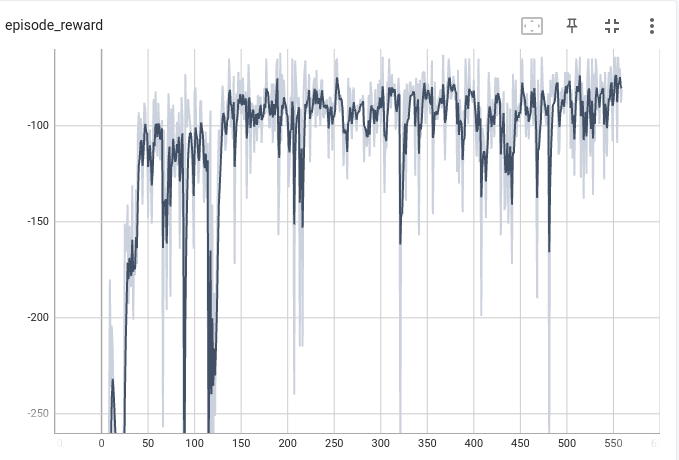


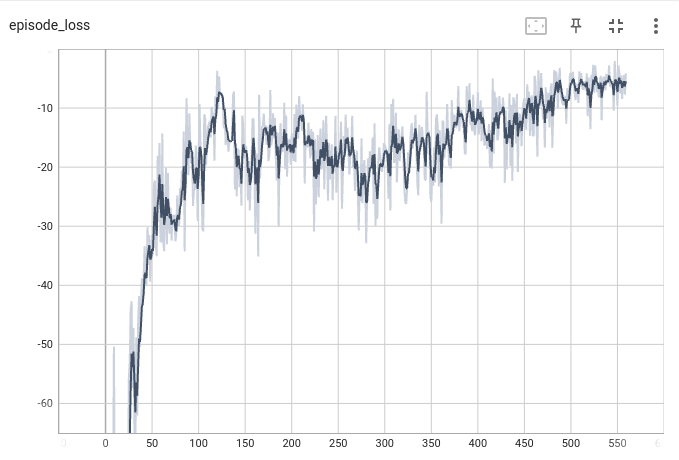
* **Acrobot -** Next, a fresh model was trained to solve the Acrobot environment with the same setup. Convergence threshold was set as -90 (reward limit threshold for the problem was defined as -100) and the problem was solved at episode 560 -  
    
  Episode 555 Reward: -64.0 Average over 100 episodes: -90.91  
  Episode 556 Reward: -77.0 Average over 100 episodes: -90.62  
  Episode 557 Reward: -70.0 Average over 100 episodes: -90.49  
  Episode 558 Reward: -88.0 Average over 100 episodes: -90.42  
  Episode 559 Reward: -81.0 Average over 100 episodes: -90.16  
  Episode 560 Reward: -72.0 Average over 100 episodes: -89.96  
  Solved at episode: 560

Average Reward of 100 consecutive episodes -



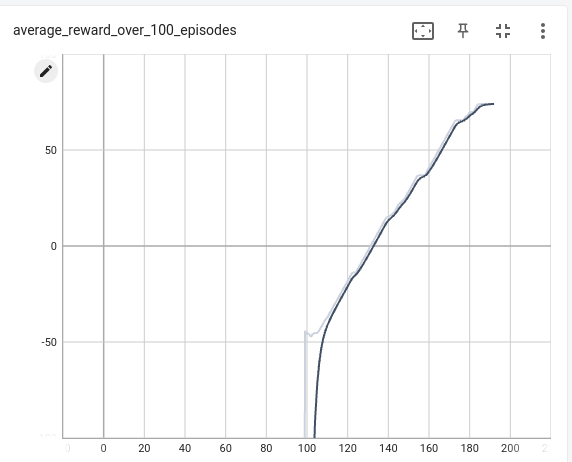
Episode Reward



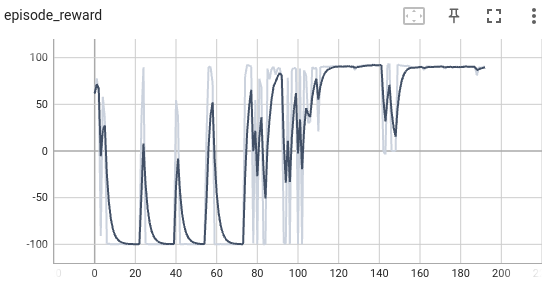
Episode Loss  


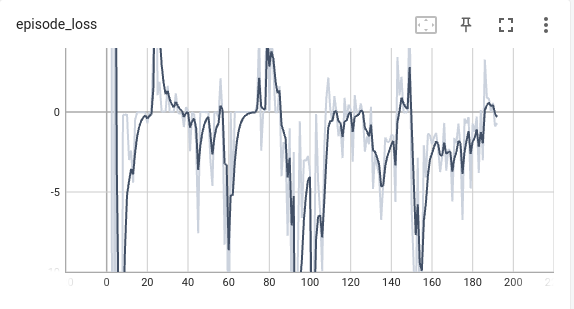
* **MountainCarContinuous** - For this environment, in order to standardize the continuous action space we applied a rounding function to the actions so the -1.0 to 1.0 range becomes an action space of [-1.0, 0, 1.0], similar to the discrete MountainCar environment. Additionally, the native rewards in the environment are very sparse (only positive reward for reaching the goal) and the model failed to converge. We augmented the environment with exploration rewards for reaching new areas based on distance from the ‘hole’ and increased rewards for reaching the goal. These rewards persist until the agent reaches the goal 20 times.   
  With max positive reward of +100 - (0.1 \* n\_actions), we experimented and eventually settled on a convergence threshold of 75. With this setup the network converged within 193 episodes -   
    
  Episode 188 Reward: 81.0 Average over 100 episodes: 73.89  
  Episode 189 Reward: 90.0 Average over 100 episodes: 73.97  
  Episode 190 Reward: 88.8 Average over 100 episodes: 73.98  
  Episode 191 Reward: 90.3 Average over 10 episodes: 73.99  
  Episode 192 Reward: 90.4 Average over 100 episodes: 74.11  
  Episode 193 Reward: 90.9 Average over 100 episodes: 76.01  
   Solved at episode: 193

Average Reward of 100 consecutive episodes -



Episode Reward



Episode Loss  


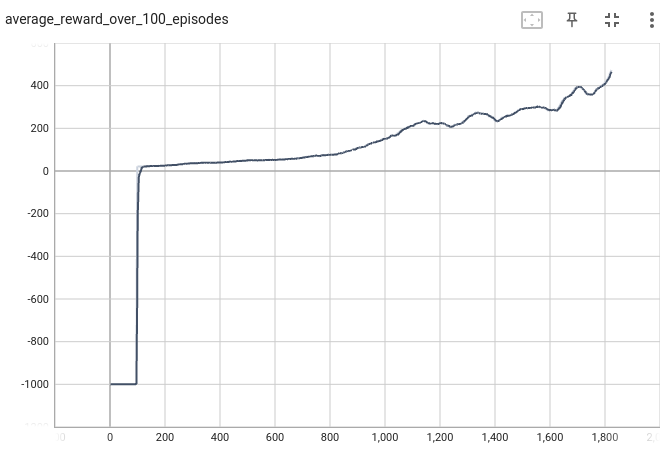
1. **Section 2 – Fine-tune an existing model**

In this section the individual networks from the previous section were fine tuned for a different environment. To run the code run the fine\_tune\_models.py script. To facilitate the fine-tuning of the existing models we freeze the weights of the initial layers to maintain the knowledge acquired in the training on the previous environment while re-initializing the weights between the single hidden layer and the output layer to achieve some flexibility for the fine-tuned model, without this step the fine-tuning process failed to converge. While we set out on the fine-tuning process with the same hyperparameters and reward schema that was used for the individual networks, some tuning was needed for the fine-tuning process to succeed.

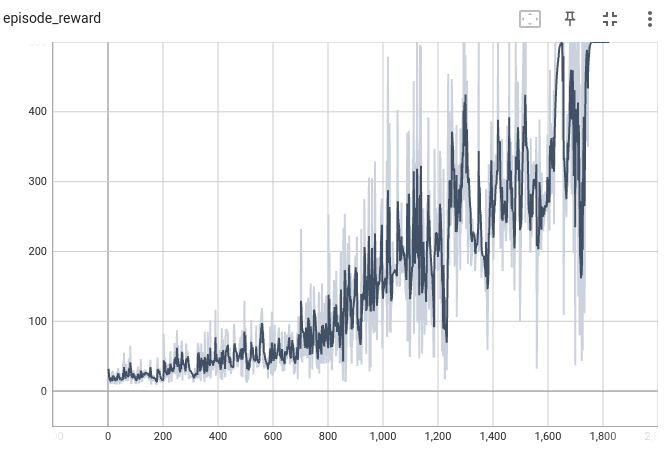
* **Acrobot -> CartPole -** Initially the Acrobot model failed to converge when fine-tuned to the CartPole environment when using the same parameters of the individual networks. After some experimentation we decided to normalize the rewards of the environments to the same range because the Acrobot agent is only trained on negative rewards (-500 to -90) and the CartPole agent is trained on a positive reward range (0 to 500). We train the Acrobot network again with and . With this configuration and learning speed of 1e-4, the fine-tuned model converged after 1824 training episodes. For comparison, we re-trained the CartPole model with the same reward schema and learning speed and found it failed to converge within the range of 5000 training episodes (with an increased learning rate of 5e-3 the model was able to converge in a similar number of episodes - 1598). In this instance the fine tuning process didn’t reduce the number of training episodes required in practice. If the fine-tuned model was able to converge with a higher learning rate it might represent an improvement but the pre-trained model seems to be less stable compared to training from scratch. It’s possible that this could be overcome by additional stabilizing methods such as gradient clipping or a different balance of actor-critic learning speeds but as it stands the fine-tuning.

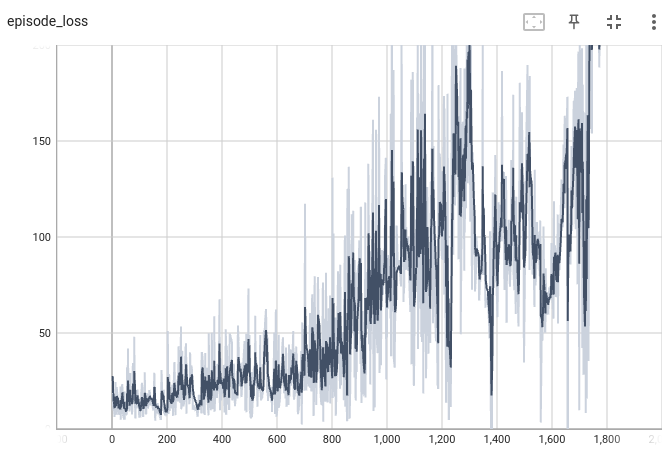
Episode 1820 Reward: 500.0 Average over 100 episodes: 465.12  
Episode 1821 Reward: 500.0 Average over 100 episodes: 465.73  
Episode 1822 Reward: 500.0 Average over 100 episodes: 468.62  
Episode 1823 Reward: 500.0 Average over 100 episodes: 473.19  
Episode 1824 Reward: 500.0 Average over 100 episodes: 476.26  
 Solved at episode: 1824

Average Reward of 100 consecutive episodes -



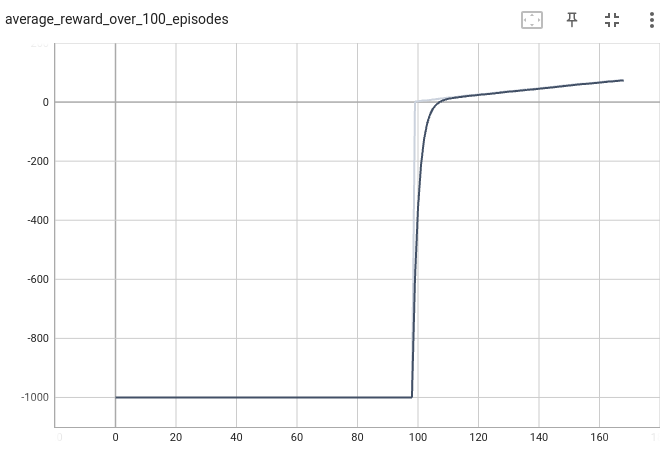
Episode Reward



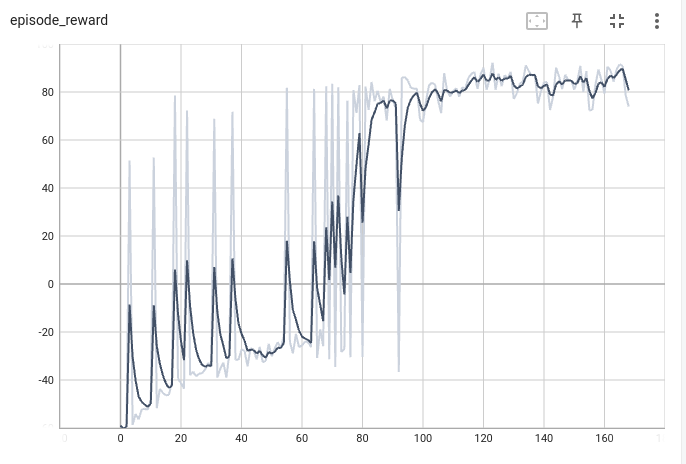
Episode Loss  


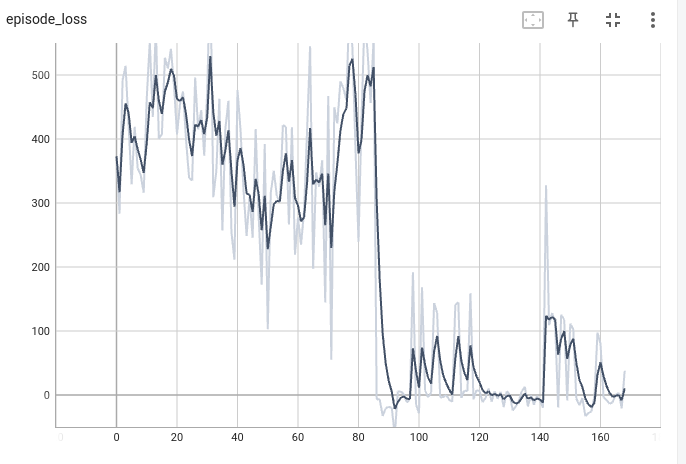
* **CartPole -> MountainCar** - In this scenario, the model initially converged very quickly to zero, i.e. the agent standing in place at the bottom and not pushing to either side, despite the exploration rewards added to the individual network. As a first step we reduced the learning rate to 5e-5 (reduced from 1e-3) which allowed the agent to “discover” the goal and the large success reward but this still converged to the same solution after one or two goal discoveries. In the next attempt we increased the exploration rewards slightly. This is a careful balance, because we cut off the exploration reward after 20 goals are reached, too high of a reward leads the agent to adopt an erratic behavior of pushing hard to the opposite side on each step without ever reaching the goal. In the individual network training the exploration reward schema was while for the fine-tuning it was replaced with .   
  This led the model to converge within 176 episodes compared to 193 of the individual network. To be fair, we went back and tested the original mountainCar model with the same reward schema and it converged with a similar number of episodes (169), albeit with a much higher learning rate so it's hard to say the fine-tuning process achieved improved performance in this instance.  
    
  Episode 171 Reward: 54.23 Average over 100 episodes: 73.26  
  Episode 172 Reward: 74.99 Average over 100 episodes: 73.21  
  Episode 173 Reward: 89.80 Average over 100 episodes: 74.38  
  Episode 174 Reward: 85.00 Average over 100 episodes: 74.39  
  Episode 175 Reward: 90.20 Average over 100 episodes: 74.47  
  Episode 176 Reward: 73.99 Average over 100 episodes: 75.44  
   Solved at episode: 176

Average Reward of 100 consecutive episodes -



Episode Reward



Episode Loss  


1. **Section 3 – Transfer learning**

In this section, we have implemented a simple version of the Progressive Networks that were presented in class. According to the definitions given to the network (two source tasks and one target task) we used the weights of fully trained networks of the environments of the sources in order to solve the target task. In the source networks we had a single hidden layer; we connected the **element-wise addition of the weights** of the source to the output of the target network. In this way, it is possible to profit **both** from the **characteristics learned in the previous tasks**, which can be relevant for the target task, and from the training of **new parameters dedicated to the target task**.

Implementing this mechanism prevents “catastrophic forgetting” - the tendency of the model to forget previously learned information when it learns new tasks or experiences. This can severely impact the performance of the model, especially in sequential learning scenarios like in our environments.

On the other hand, the size of the network and the number of weights increases significantly with each addition of a task, which can make training difficult and harms its scalability (tasks that require very large networks as policy networks).

Moreover, it is not possible to know directly which of the source tasks is indeed relevant to the target task. Due to this, the explanatory power of the model is impaired.

**Setting: {cartpole, acrobot} -> mountainCar**

**learning\_rate = 0.0005**

**discount\_factor = 1**

Episode 254 Reward: 89.8 Average over 100 episodes: 73.59

Episode 255 Reward: 89.5 Average over 100 episodes: 73.6

Episode 256 Reward: 89.7Average over 100 episodes: 73.63

Episode 257 Reward: 88.9 Average over 100 episodes: 73.63

Episode 258 Reward: -99.8 Average over 100 episodes: 71.74

Episode 259 Reward: 89.3 Average over 100 episodes: 71.77

Episode 260 Reward: 55.6Average over 100 episodes: 71.44

Episode 261 Reward: 77.9 Average over 100 episodes: 71.34

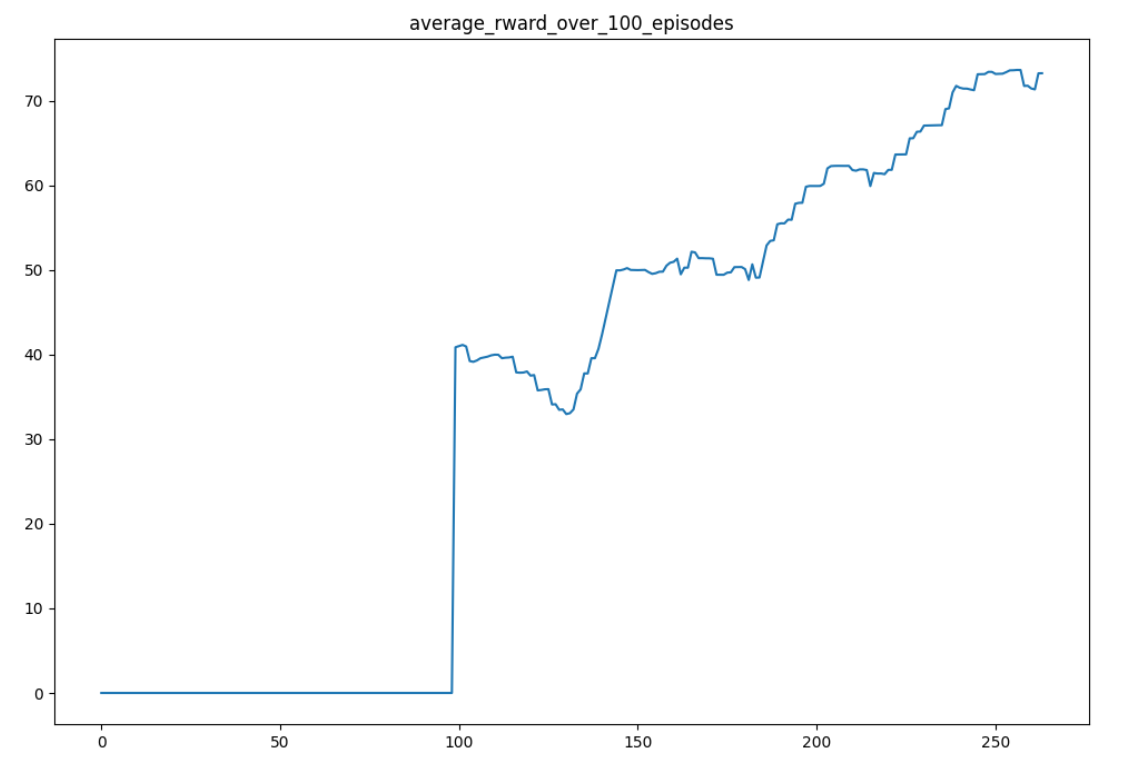
Episode 262 Reward: 90.2 Average over 100 episodes: 73.24

Episode 263 Reward: 89.8 Average over 100 episodes: 73.25

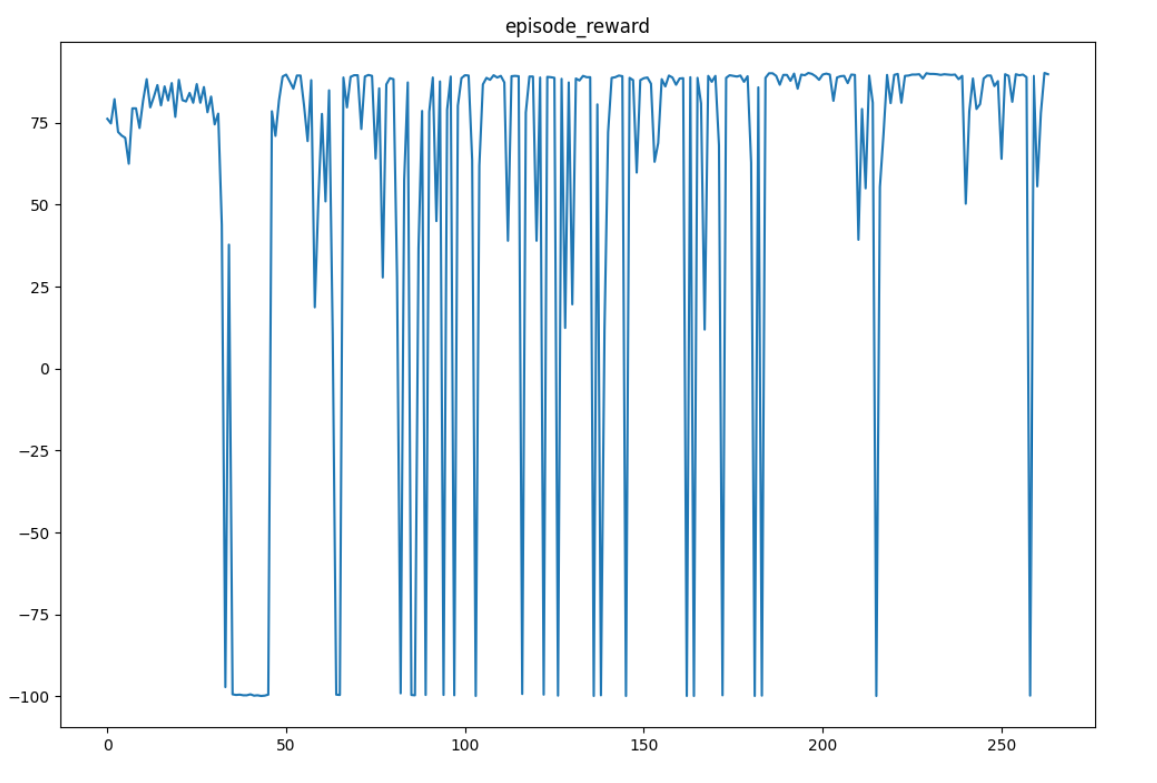
Episode 264 Reward: 88.8 Average over 100 episodes: 75.13

Solved at episode: 264

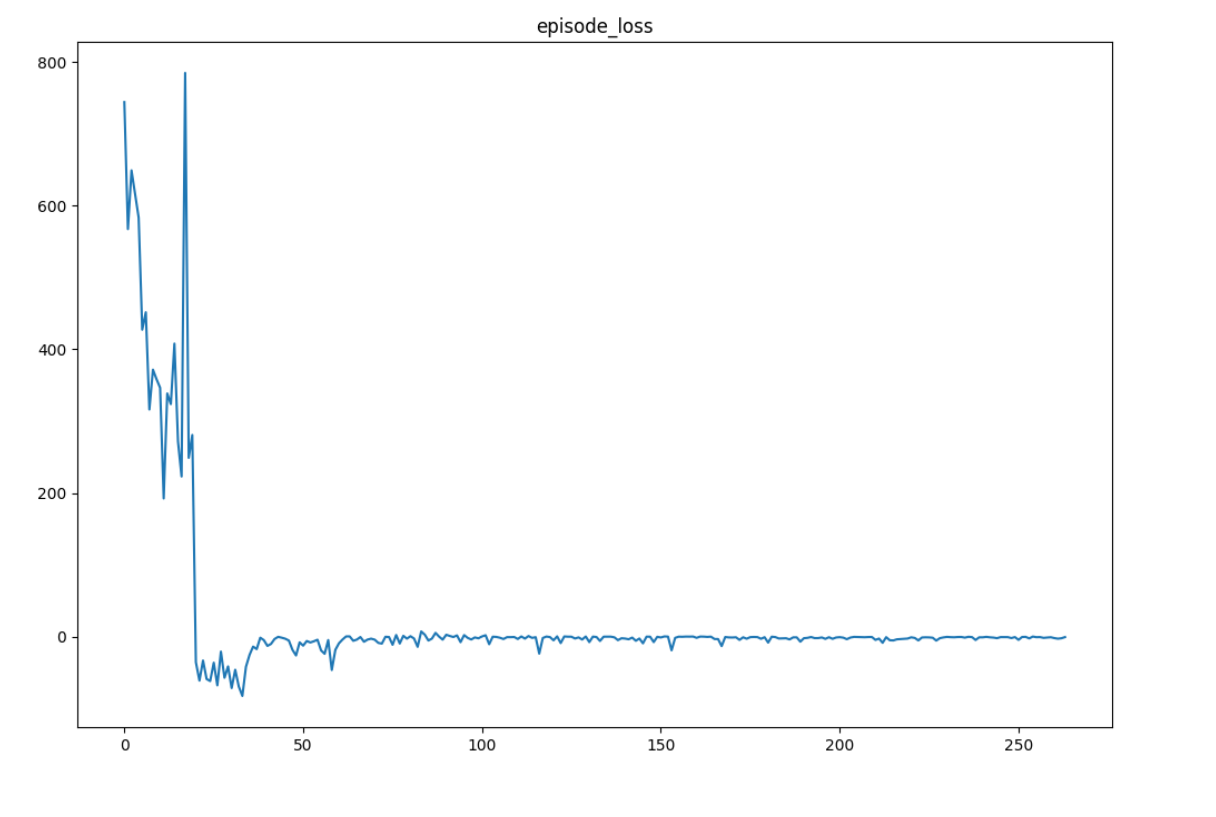
Average Reward of 100 consecutive episodes -



Episode Reward



Episode Loss



**Setting {acrobot, mountainCar} -> cartpole**

learning\_rate=0.001

discount\_factor=0.99

Episode 816 Reward: 500.0 Average over 100 episodes: 452.41

Episode 817 Reward: 500.0 Average over 100 episodes: 455.46

Episode 818 Reward: 500.0 Average over 100 episodes: 458.5

Episode 819 Reward: 500.0 Average over 100 episodes: 461.43

Episode 820 Reward: 500.0 Average over 100 episodes: 463.82

Episode 821 Reward: 500.0 Average over 100 episodes: 465.64

Episode 822 Reward: 500.0 Average over 100 episodes: 465.64

Episode 823 Reward: 500.0 Average over 100 episodes: 465.64

Episode 824 Reward: 500.0 Average over 100 episodes: 465.64

Episode 825 Reward: 500.0 Average over 100 episodes: 465.64

Episode 826 Reward: 500.0 Average over 100 episodes: 466.55

Episode 827 Reward: 500.0 Average over 100 episodes: 469.03

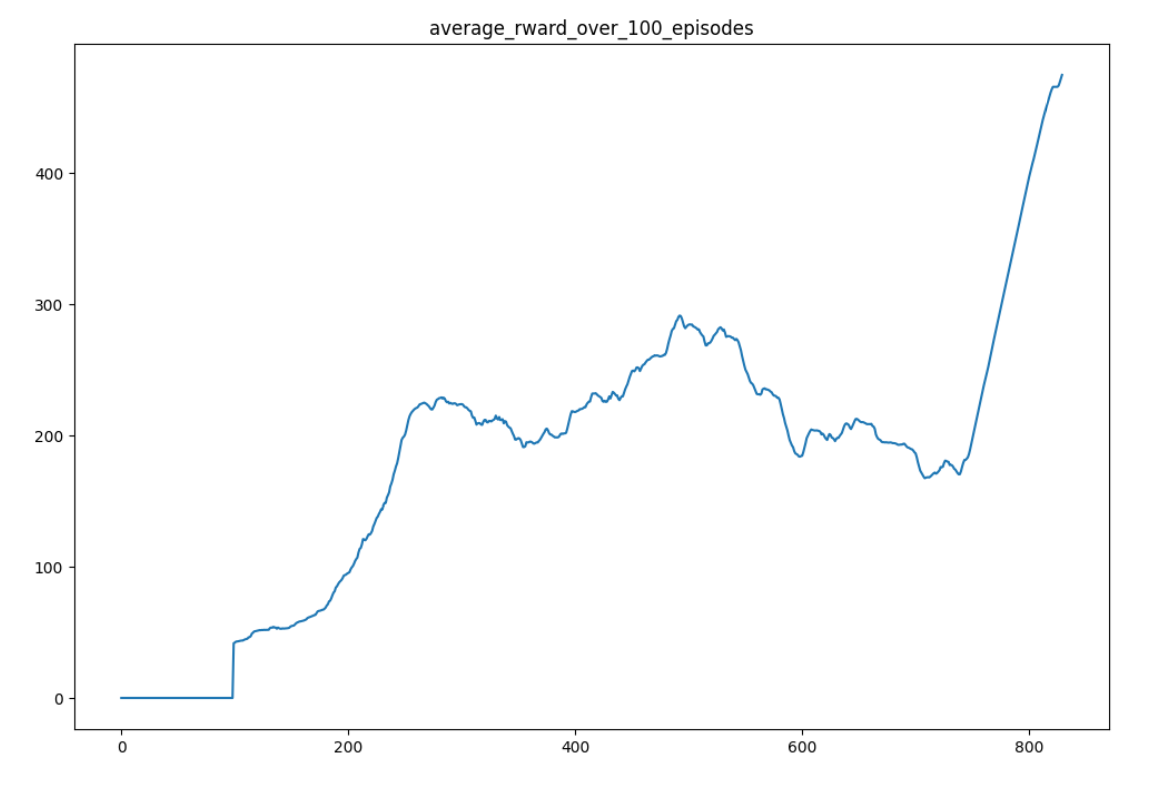
Episode 828 Reward: 500.0 Average over 100 episodes: 471.99

Episode 829 Reward: 500.0 Average over 100 episodes: 474.69

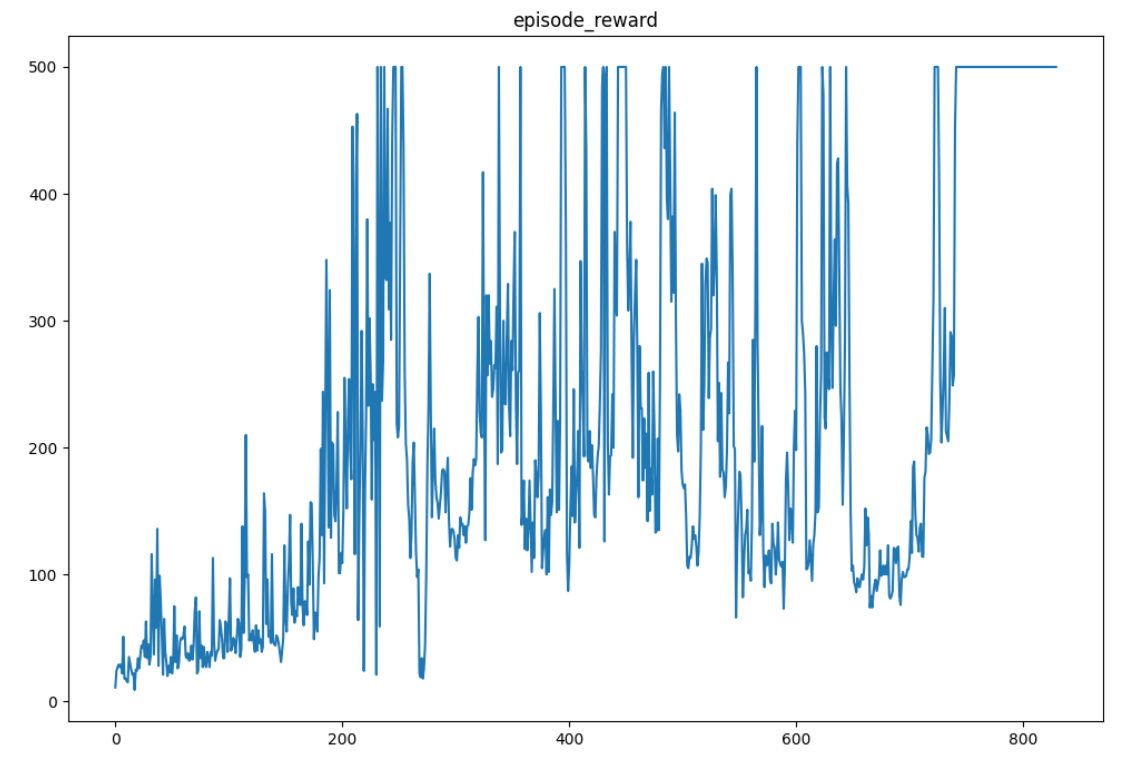
Episode 830 Reward: 500.0 Average over 100 episodes: 477.1

Solved at episode: 830

Average Reward of 100 consecutive episodes -



Episode Reward



Episode Loss

