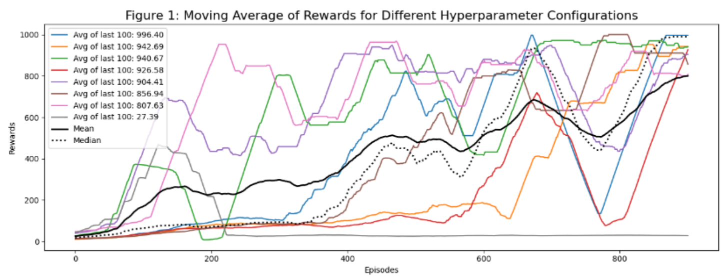
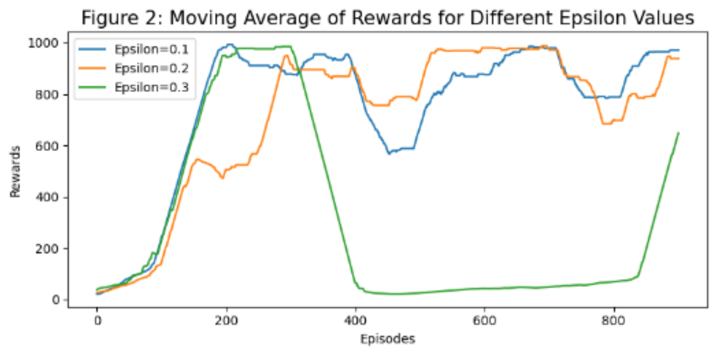
Part III – PPO Implementation:

In this report we have implemented Proximal Policy Optimization (PPO) algorithm on Mujoco's simulated "InvertedPendulum-v4" environment. We ran our PPO algorithm using these various hyperparameters learning rates: 1×10^{-3} and 1×10^{-4}, clipping parameters: 0.8 and 0.5, number of epochs: 1 and 5, discount factors: 0.95 and 0.90, and a minibatch size of 64, and lastly, training over 1000 episodes.



The graph (Figure 1) illustrates how different hyperparameters affect the PPO algorithm's performance. Initially, the learning process is highly volatile with widely varying rewards, but as more episodes pass, the agent learns, and the rewards generally improve. We experimented with different learning rates and found that a low learning rate of 1×10^{-4} more often resulted in poor rewards. A potential reason for this could be due to the low learning rate, the agent finds the local minimum but is not able to further explore to find the global and optimal action. The black line represents the mean of all runs with all parameters, showing an overall upward trend. We ran the algorithm for 1000 episodes due to time constraints. If we had more time, we would have extended this to 1 million episodes, as done in the referenced paper, to observe if the rewards what the trends would look like over time as it stabilizes more. The figure highlights that some hyperparameter combinations, like the blue and orange lines, perform exceptionally well, achieving rewards close to 1000 on its average results on the last hundred episodes, while others, like the grey line, perform poorly with an average reward of around 27 in the final episodes, which indicates that choosing the right hyperparameters is important.

In the paper titled "Proximal Policy Optimization Algorithms" by John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov, OpenAI, they implemented the PPO algorithm using these specific hyperparameters: a learning rate of 3×10^{-4}, 10 epochs, a discount factor of 0.99 and a GAE parameter of 0.95. In addition, they have tested it with a clipping parameter of 0.1, 0.2 and 0.3. The paper also ran it using a 1 million timestep benchmark. We tried to compare using the same parameters, but only running up to 1000, which resulted in the rewards shown in figure 2. In general, the PPO agent seem quickly learn the best action, but sometimes chooses the wrong actions. In general, over the 1000 episodes, it is unstable, but generally gets to the 1000 cumulative rewards. We hypothesis this would occur more frequently if we had time to let it run longer, then we could also see how the green line develops, as it seems to be getting better at the end there.



Comparing the results across different hyperparameter settings, as shown in Figures 1 and 2, with those presented in the paper reveals key insights into the PPO algorithm's performance. The graph (figure 2) indicates that lower epsilon values of 0.1 and 0.2 resulted in more stable and higher rewards, suggesting a well-balanced exploration-exploitation trade-off. However, with an epsilon of 0.3, the agent initially achieved high rewards but experienced a significant drop between 400-800 episodes, indicating over-exploration that disrupted the learning process.

Our results show that moderate epsilon values provided stability and high rewards, while the paper also emphasizes the importance of the clipping parameter in stabilizing training and avoiding excessive updates. Overall, both our results and the paper highlight the critical role of hyperparameter tuning in PPO's performance. For further exploration, we would have explored more hyperparameters and ran it over a higher timestep benchmark to see how it develops and possibly stabilizes over time.