Question 1: Basic Q-learning performance (DQN)

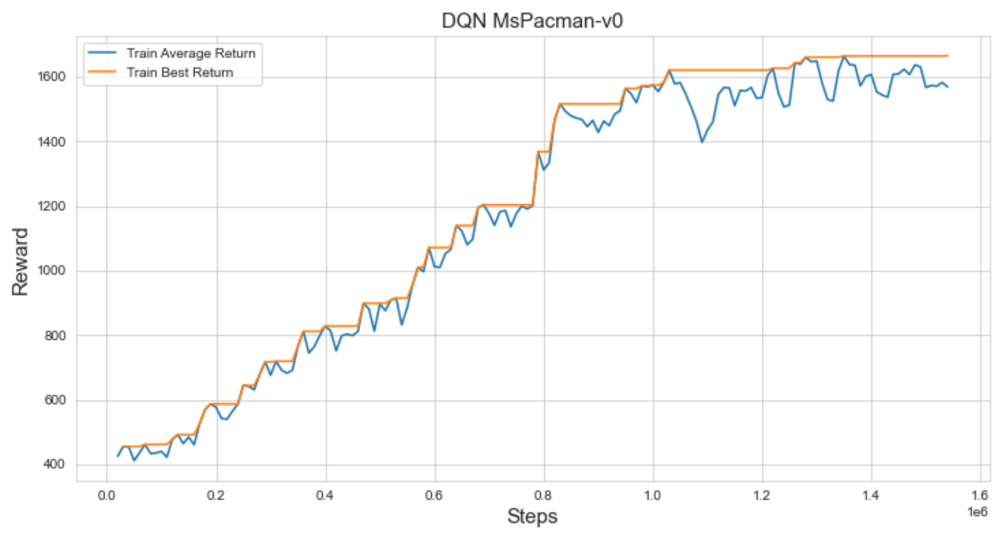

Figure 1: Learning curve showing the average training reward and best training reward of DQN for MsPacman-v0 for ~ 1.5 million steps.

Question 2: Double Q-learning (DDQN)

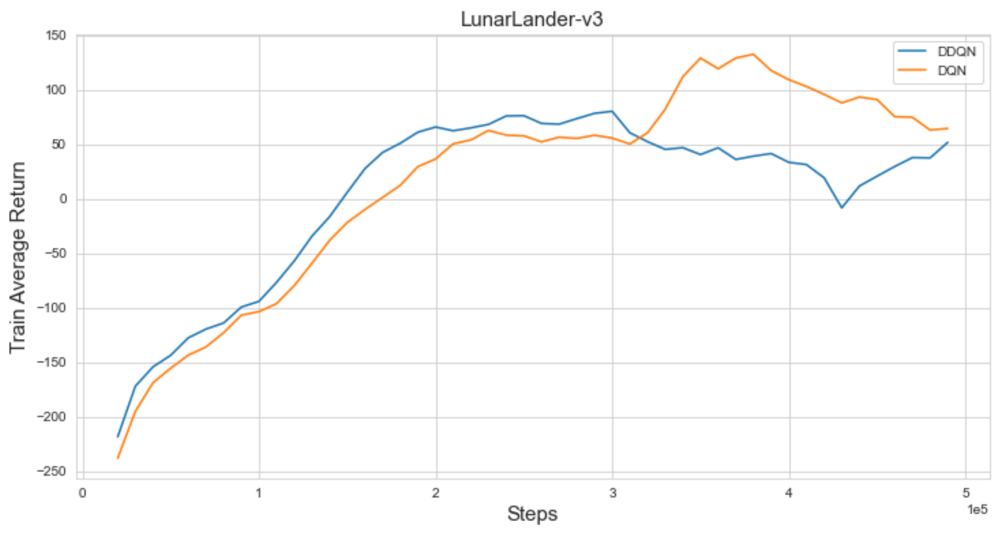

Figure 2: Learning curves of the average training reward across three different runs of DQN and DDQN respectively on the LunarLander-v3 environment. The DDQN outperforms the DQN in the first $30,0000$ iterations, hereafter the DDQN has a slight drop in performance and the DQN has an increase in performance.

Question 3: Experimenting with hyperparameters.

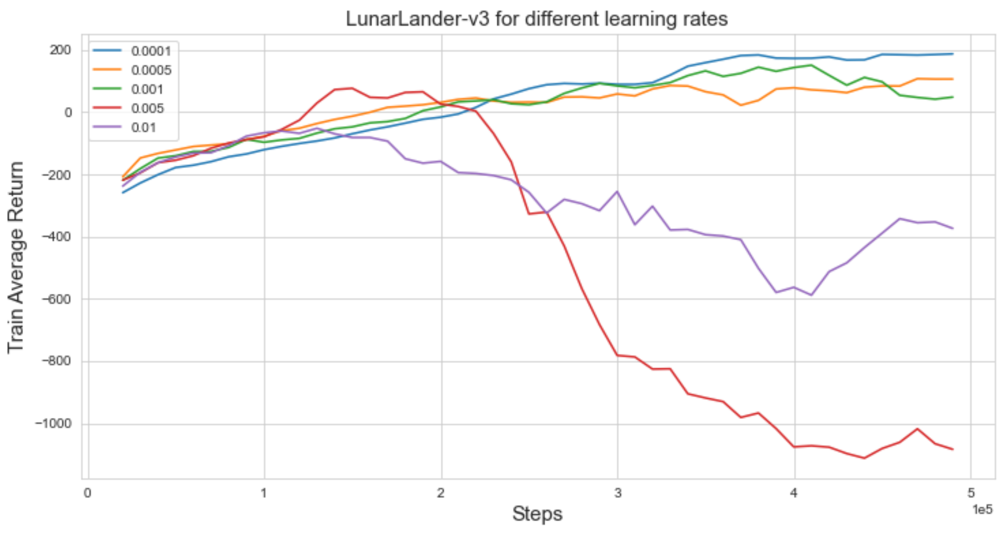

Figure 3: Learning curves showing the average training reward of DQN for five different learning rates in the LunarLander-v3 environment. The learning rates control the size of the steps in the gradient descent, in other words, how big a change is made to the weights in each update. Having a too large learning rate results in rapid changes and can hurt performance, as it can not reach optimality. A too low learning rate requires more training epochs for the model to converge, causing it sometimes not to. The effects of increasing the learning rate above the original setting of 0.001 for LunarLander-v3 hurt performance significantly, on the other hand, performance increases when the learning rate is lowered. A learning rate of 0.0001 seems to outperform the original setting, meaning that the environment benefits from taking smaller steps when updating.

Question 4: Sanity check with Cartpole

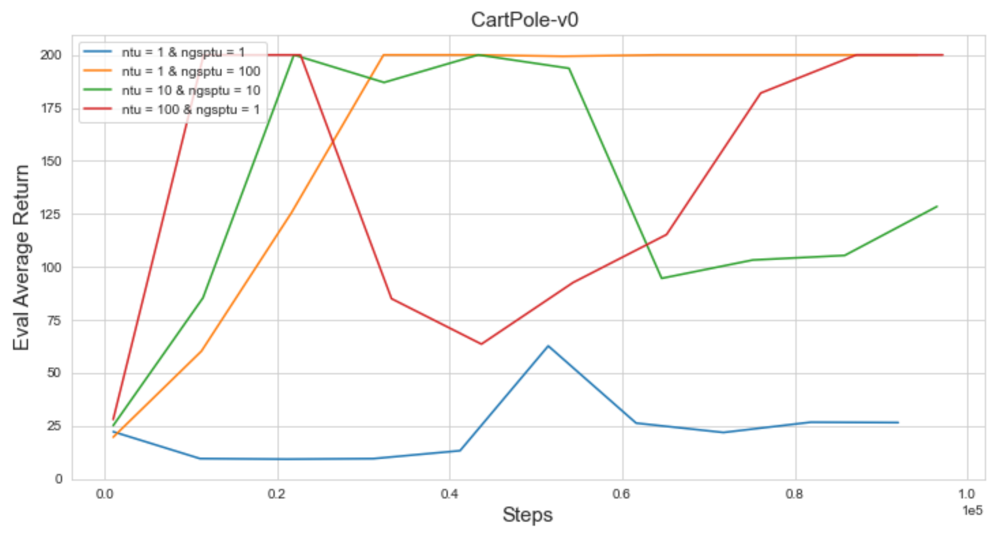

Figure 4: Learning curves of average evaluation reward of Actor-Critic on CartPole-v0 for different parameter settings of number of targets updates and number of gradient steps per target updates. The best performance of the Actor-Critic algorithm is found with few target updates and with a high number of gradient steps per target updates i.e. $ntu = 1$ and $ngsptu = 100$.

Question 5: Run actor-critic with more difficult tasks

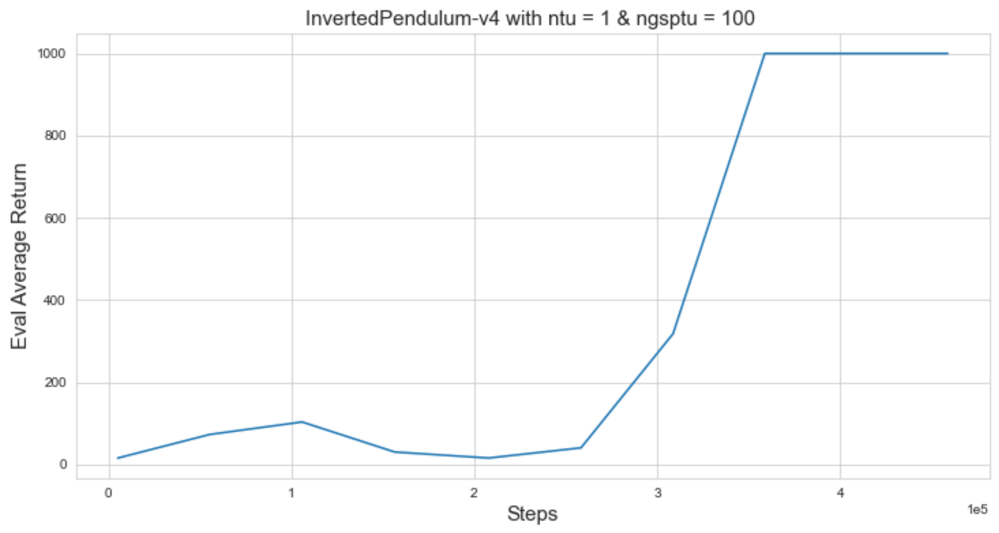

Figure 5: Learning curve of average evaluation reward of Actor-Critic on InvertedPendulum-v4 with $ntu = 1$ and $ngsptu = 100$.

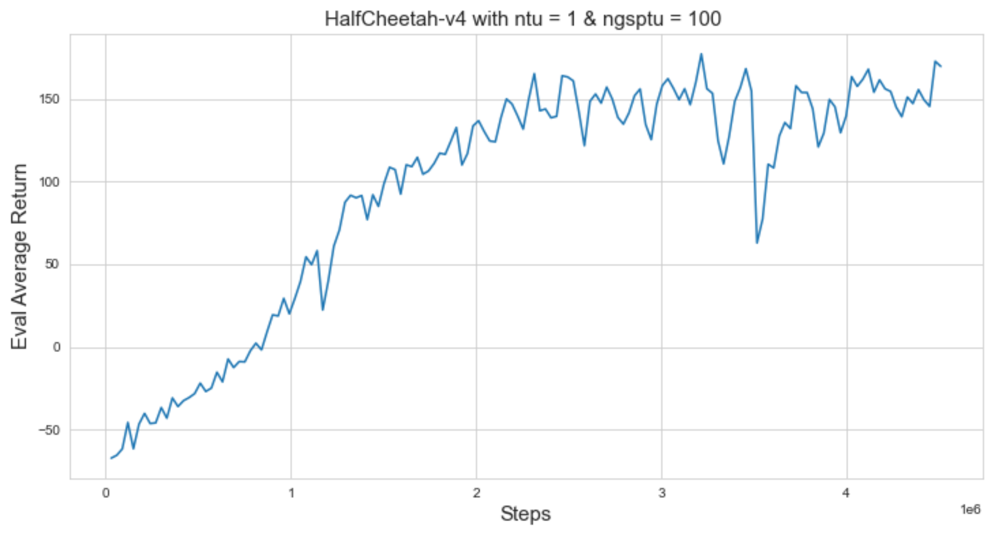

Figure 6: Learning curve of average evaluation reward of Actor-Critic on HalfCheetah-v4 with $ntu = 1$ and $ngsptu = 100$.

Question 6: Run soft actor-critic (SAC) more difficult tasks.

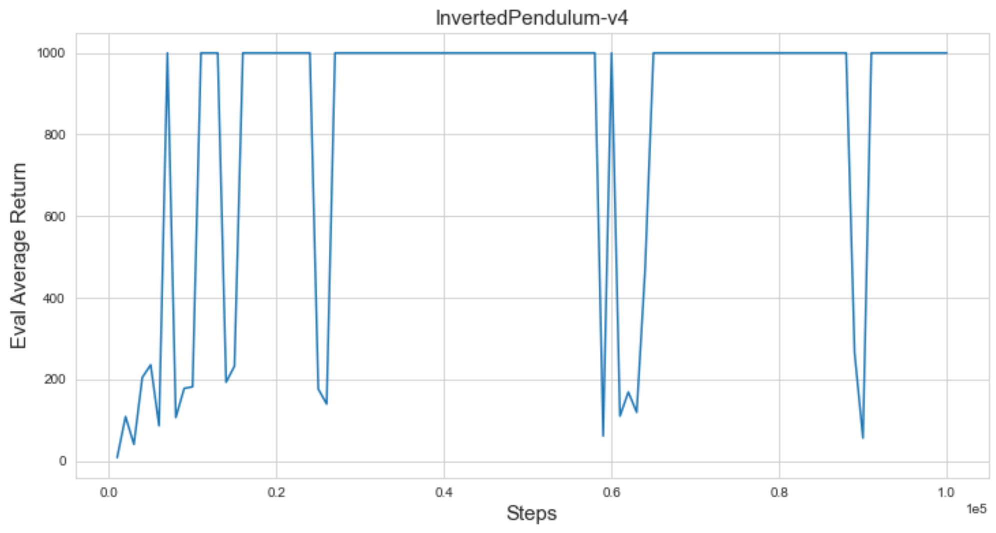

Figure 7: Learning curve of average evaluation reward of Soft-Actor-Critic on InvertedPendulum-v4. Soft-Actor-Critic algorithm performs significantly better compared to Actor-Critic, as it takes fewer steps to reach an evaluation reward of 1000.

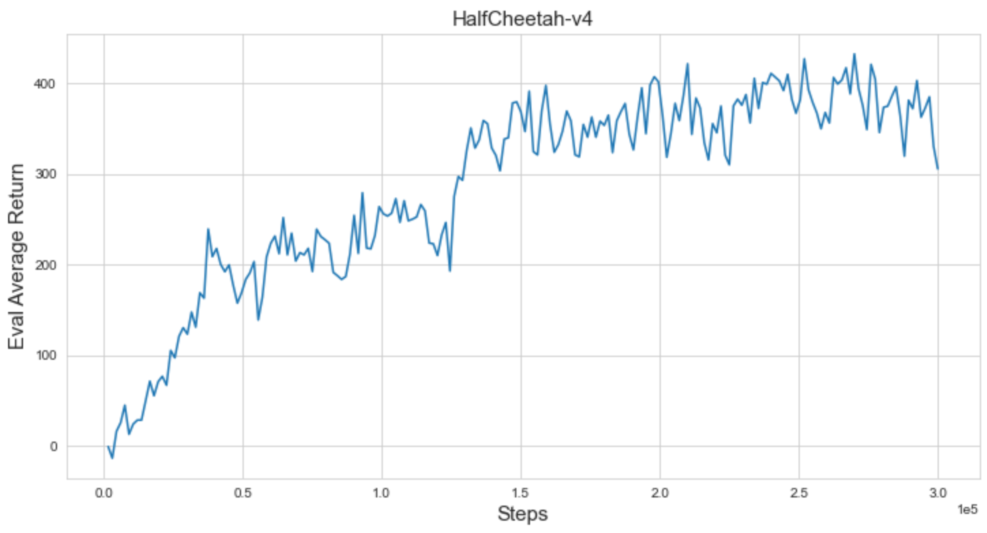

Figure 8: Learning curve of average evaluation reward of Soft-Actor-Critic on HalfCheetah-v4. Again, Soft-Actor-Critic performs significantly better compared to Actor-Critic, as it manages to reach double the rewards of Actor-Critic.