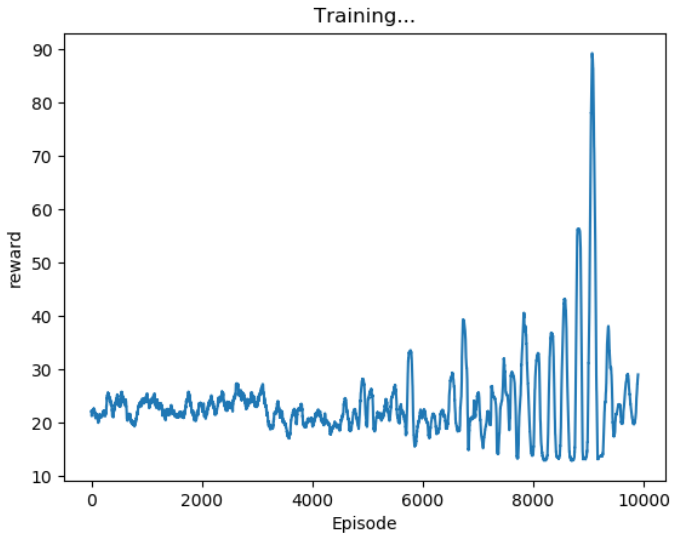
CS886 Assignment Three

Connor Raymond Stewart

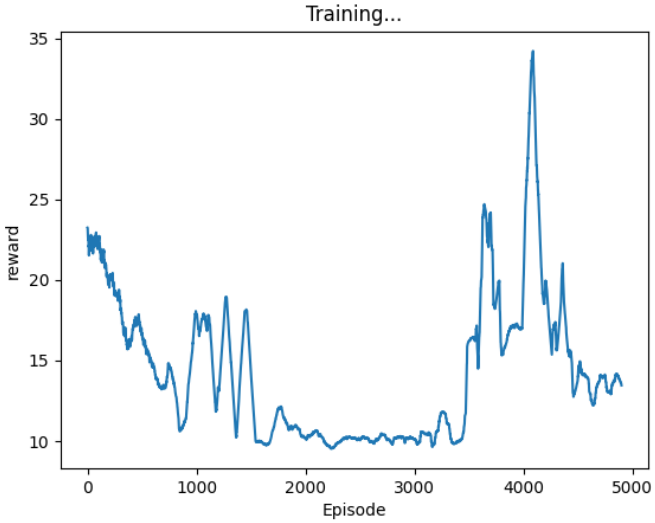
Student ID: 20673233

Part I – Partially Observable RL

The DQN results are as follows:



The DRQN results are as follows:



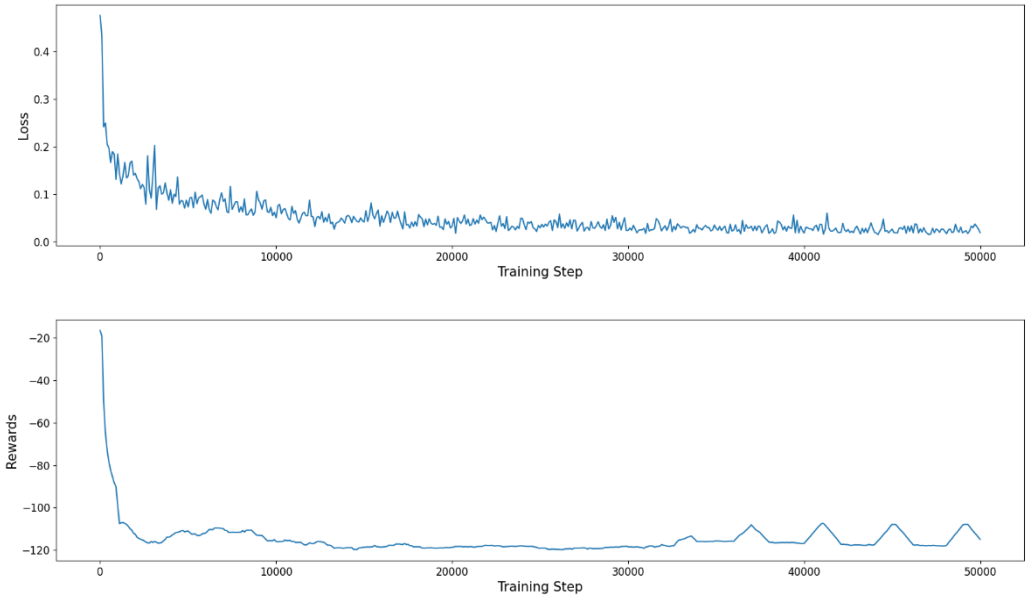
The four DQRN results show a trend where the DQRN algorithm has a lower overall oscillation than the DQN program. The DRQN program tends to drop its reward midway through execution when compared to the program start but recovers near the end. An LSTM layer is a long short-term memory, which is a recurrent neural network.

The LSTM layer helps learn long-term dependencies, a type of information where previously iterated information in the neural network can be used to solve a present task. This program helps with the modelling of future rewards by lowering the oscillation in the rewards across episodes. The LSTM does this by learning patterns in the Cartpole environment from previously iterated episodes in the neural network. The DRQN program runs slower than the DQN program. A reason for the slower execution speed of the DRQN program is that it adds an extra layer to the neural network – specifically an LSTM layer – which has a high degree of computational complexity compared to the original two layers (as it needs to keep track of long-term dependencies).

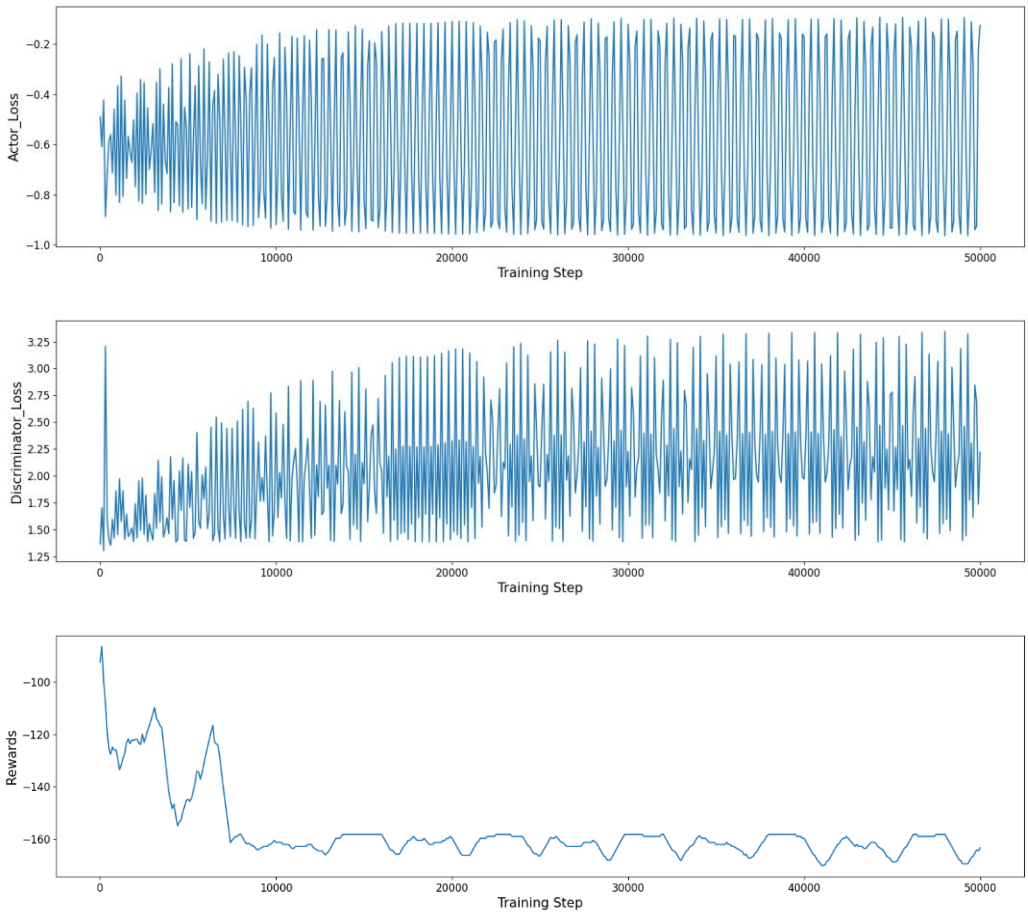
Rewards are lower than in the DQN model since the DRQN model is more complex with a more extensive neural network, meaning it may take more episodes to determine optimal values, or overfitting may be causing the data to show invalid patterns in the output. Furthermore, the DRQN may have been implemented poorly on my part and may have some minor bug or error in its hyperparameters. Stochasticity may be skewing the results as well in the DQN model. It is also possible that the DQN model is correct, and the results reflect a smoothed version of the DRQN model, despite the lower reward values.

Part II – Imitation Learning

The results of the behavior cloning are as follows:



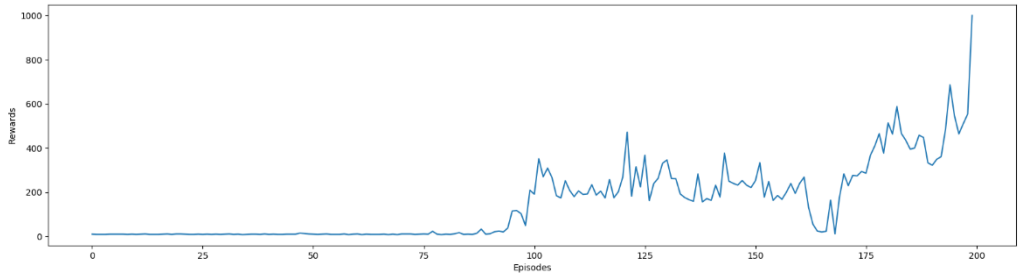
The results of the GAIL are as follows:



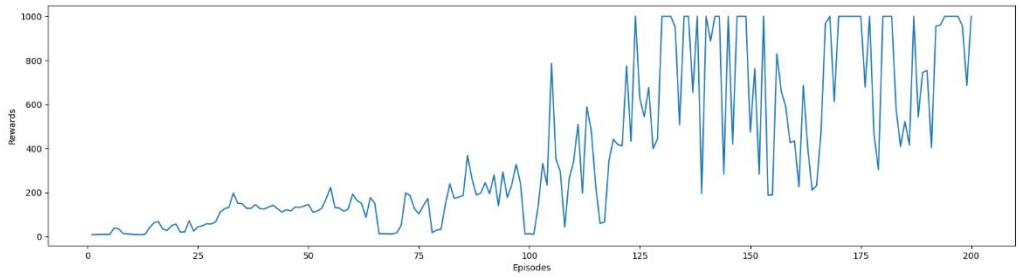
Both the BC and the GAIL algorithm produce logarithmically decaying rewards. In the BC algorithm, the loss value also decays logarithmically; however, in the GAIL algorithm, the discriminator and actor loss values oscillate with increasing amplitude as the number of training steps performed increases. Behaviour cloning (BC) focuses on learning a policy using supervised learning, a machine learning method where a function maps an input to an output based on input-output pairs. Essentially, behaviour cloning focuses on reconstructing a pattern from a behavioural trace using machine learning optimization. Behaviour cloning is thus characterized by an increase in reward over time, followed by drops in reward as the model makes mistakes. Eventually, the model reaches an optimum, where the reward peaks at maximum, as seen in the DQN algorithm below. The GAIL algorithm works using adversarial learning, a method where the machine learning model is intentionally fooled by providing deceptive supply input. The adversarial machine learning model can test susceptibility. A machine learning model has slight differences in input. Therefore, the GAIL algorithm shows sharp changes in reward at random intervals, even though the reward is stable, primarily on a horizontal plane. The sharp changes indicate that the GAIL algorithm finds weaknesses in the machine learning model and changes the value of the rewards while the program attempts to correct the model for future episodes.

Part III – Distributional RL

The results of the DQN algorithm are as follow



The results of the C51 algorithm are as follows:



The results of the DQN algorithm show a long period of no increase in reward followed by a sharp increase in rewards. The C51 algorithm shows an increase in reward overall, but it is marked by sudden drops in reward followed by fast recoveries. The DQN algorithm is deterministic, and therefore input can be mapped onto output with certainty. However, the C51 algorithm is stochastic, meaning randomness makes the outcome of the program random in nature. Stochasticity in the C51 algorithm changes the program's execution, contributing to the reward's random drops over time. However, the C51 algorithm tends to a higher reward as the number of episodes increases. The deterministic DQN algorithm seemed to have trouble picking up on patterns for the first half of the program execution and likely needed brute force to find meaningful increases in reward due to a lack of stochasticity. The C51 models' stochasticity provided a degree of randomness that allowed the program to occasionally happen upon new patterns, allowing small spikes in reward due to randomness to propagate across later episodes. Therefore, the C51 model is less stable than the DQN model but can faster convergence upon an optimal reward.