Jackie Au 914158394

1. Problem Representation

1.

I believe it will be easier for the agent to learn from the game display (images) as inputs rather than hardware ram (array). The model will learn through sub symbolic learning as combinational convolutions will be performed on the given game display -- i.e. filters will be passed over the raw image/pixels thus it is more beneficial for the inputs to be the game display.

2.

The purpose of neural networks in Q-Learning is to take in inputs that are states and output actions based on q values. A q learning model can learn to make the best action based on the best q value. For this neural network, the inputs are env, num_frames, batch_size, gamma, and replay_buffer -- representing the game environment, the capped number of frames a model can train, the batch size is the number o from buffer, gamma is the discount factor which effects immediate and future rewards, and the replay buffer that stores all results of a game, respectively. The output of this neural network is x which represents q values for each action.

3.

Lines 48 and 57 of dqn.py describes the exploration vs exploitation strategy. In line 48, random.random() returns a random floating point number in the range [0.0, 1.0) and tests it against epsilon. If this random value is greater than epsilon then the model will perform an action through exploitation -- choosing an action by taking max q-value for its current state. Otherwise line 58 will be true and the model will explore the environment and choose the next action randomly. Actions are based on the strategy mentioned before -- initially the model will choose an action randomly and the value of epsilon will decay over time and be tested against random.random().

2. Making the Q-Learner Learn

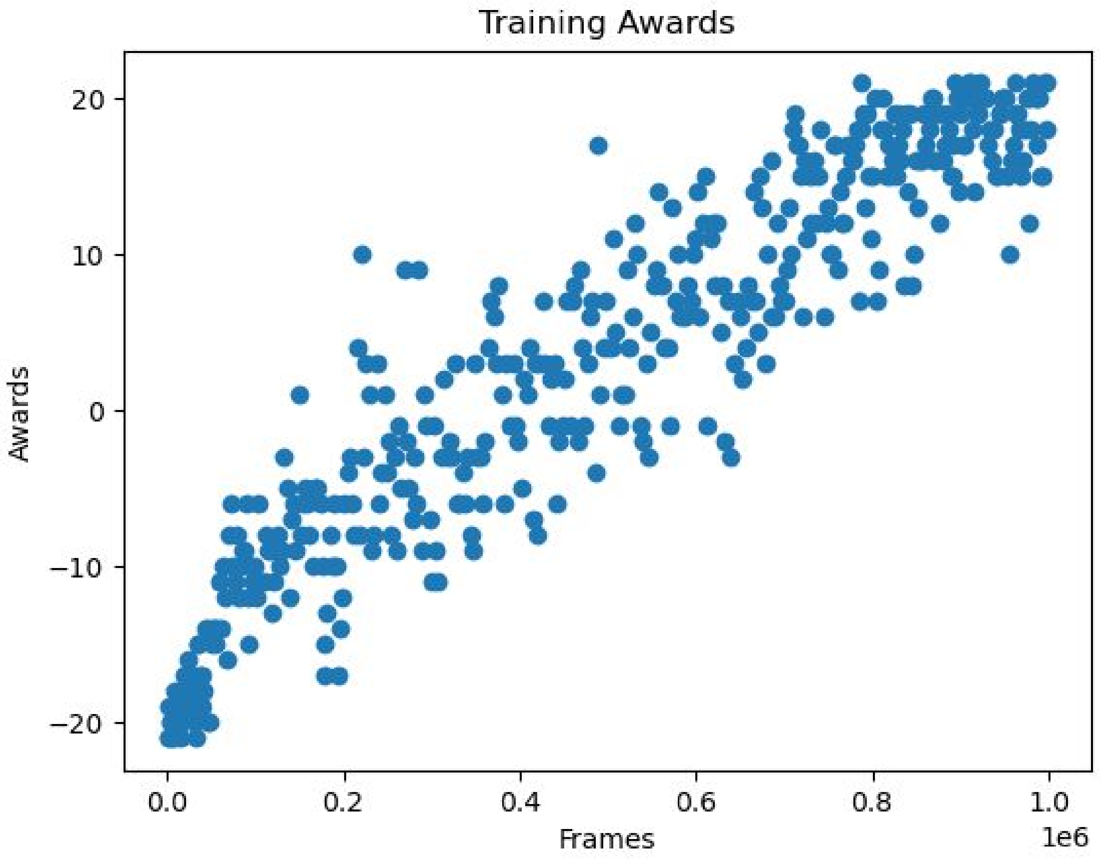
1.

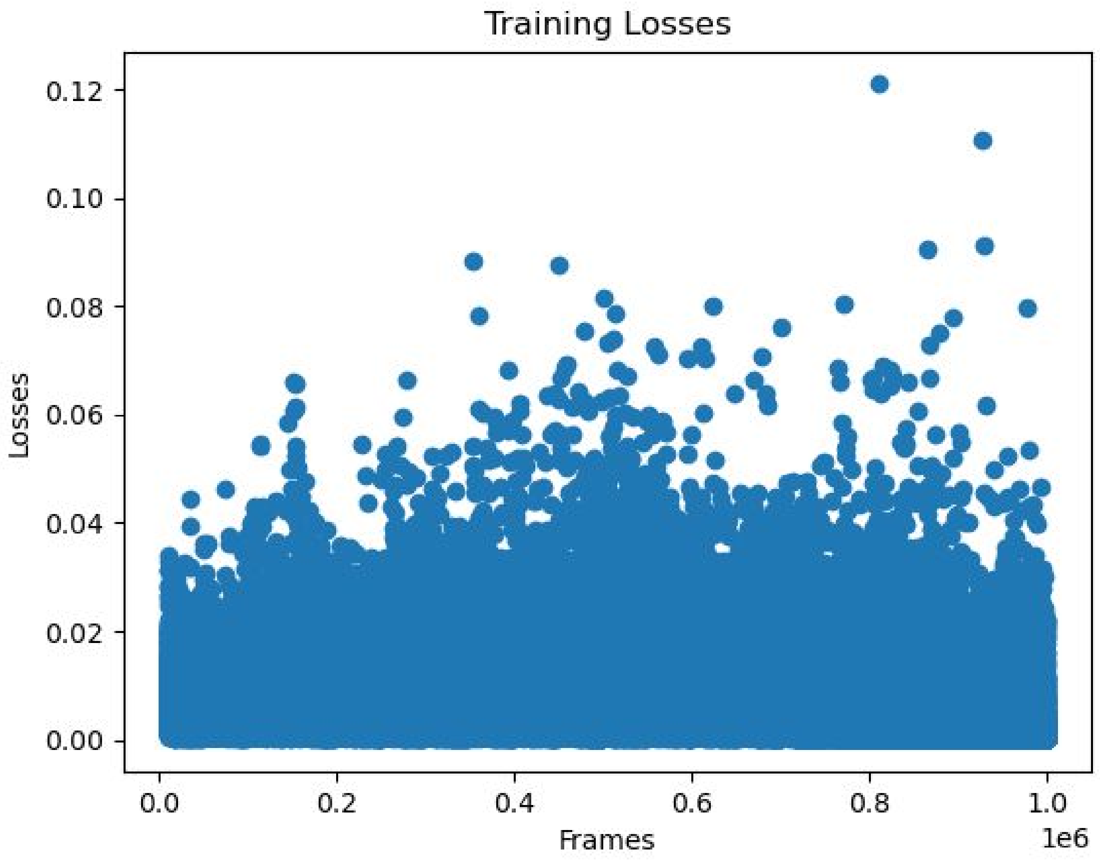
The objective function of Deep Q Learning Network for one-state lookahead reduces the difference between the current q value and the estimated target q values by recursively looking at more distant descendants -- reducing overall error and making training efficient.

Θ_i is a model's parameters at the i-th iteration.
y_i is the "model output" or simply the current q value at the i-th iteration.
Q is the "target output" or simply the next q value that is determined by Θ_i at i-th iteration.
s and a are the state and action respectively which correspond to specific state and actions at the i-th iteration.

This loss function helps our model learn by being more efficient in looking ahead into more distant descendants along the neural network when training. Returning loss, or simply the mean squared error of the current q value and target q value (based on update rule), allows the model to update network weights accurately.

4. Learning toPlay Pong.





Training Losses
Losses
Frames
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.0
0.2
0.4
0.6
0.8
1.0
1e6