Assignment 3: DeepQLearning

Gabriel Vazquez

917307919

Problem 1 - Problem Representation

Part 1

For the Q learner we must represent the game as a set of **states**, **actions**, and **rewards**. OpenAI offers two versions of game environments: one which offers the state as the game display (image) and one that offers the state as the hardware ram (array). Which do you think would be easier for the agent to learn from and why?

Solution

I think an agent would have an easier time learning from the game display rather than the hardware ram.

This is because of how the information of the game state is encoded into the screen. When we use the hardware ram, we are limited to a few registers that encode the game state. Also, ram for the atari 2600 is limited to only 128 bytes.

If we use the game display, our source of information is retrieved from the pixels on the screen. We can apply techniques such as Convolution in our neural network to retrieve more information about the game.

Also, when using the game display, it is easier to compare states as opposed to the ram approach since we would have to do extra work to accurately compare states.

Part 2

Use the starter code to answer this question. Describe the purpose of the neural network in Q-Learning. Neural networks learn a complicated function mapping their inputs to their outputs. What will the inputs and outputs for this neural network be? What do these variables represent? The neural network in the starter code is class QLearner(nn.Module).

Solution

The purpose of the neural network in Q-Learning is to learn a mapping from a given state to the Q-values of the possible actions we can take. The input of this neural network will be an order-3 tensor of size $1 \times 84 \times 84$. The output of this neural network will be a vector of size 6.

The input represents the game display of pong and the output represents the actions that we can take in pong. 3 values represent going down in pong, 1 value represents doing nothing, and the last 2 values represent moving up.

Part 3

What is the purpose of lines 48 and 57 of dqn.py (listed below)? Doesn't the q learner tell us what action to perform?

```
if random.random() > epsilon:
    ...
else:
    action = random.randrange(self.env.action_space.n)
```

Solution

The purpose of this if-else statement is to address the issue of exploration vs. exploitation. When the model trains to learn the Q-values of the game, it has two options. The first option is to exploiting the environment and the second option is to explore the environment. Exploiting the environment means to perform an action in which we know the Q-value. This is the largest known Q-value given the current state. However, it is not garunteed that the Q-value that we initially learn are the true Q-values. If we don't explore, we will restrict ourselves and prevent our agent from choosing the optimal move. To mitigate this, we should explore a percentage of the time so that we can learn Q-values for actions that we may have not done before. This will help our agent learn the best action given a state.

Problem 2 - Making the Q-Learner Learn

Part 1

Explain the objective function of Deep Q Learning Network for one-state lookahead below; what does each variable mean? Why does this loss function help our model learn? This is described in more detail in the Mitchell reinforcement learning text starting at page 383. We've provided the loss function below. This should be summed over the batch to get one value per batch.

$$Loss_i(\theta_i) = (y_i - Q(s, a; \theta_i))^2$$

Hint: θ is convention for "model parameters" and y is convention for "model output". The subscript i refers to "the i-th iteration". If you are stuck, research "mean squared error".

Solution

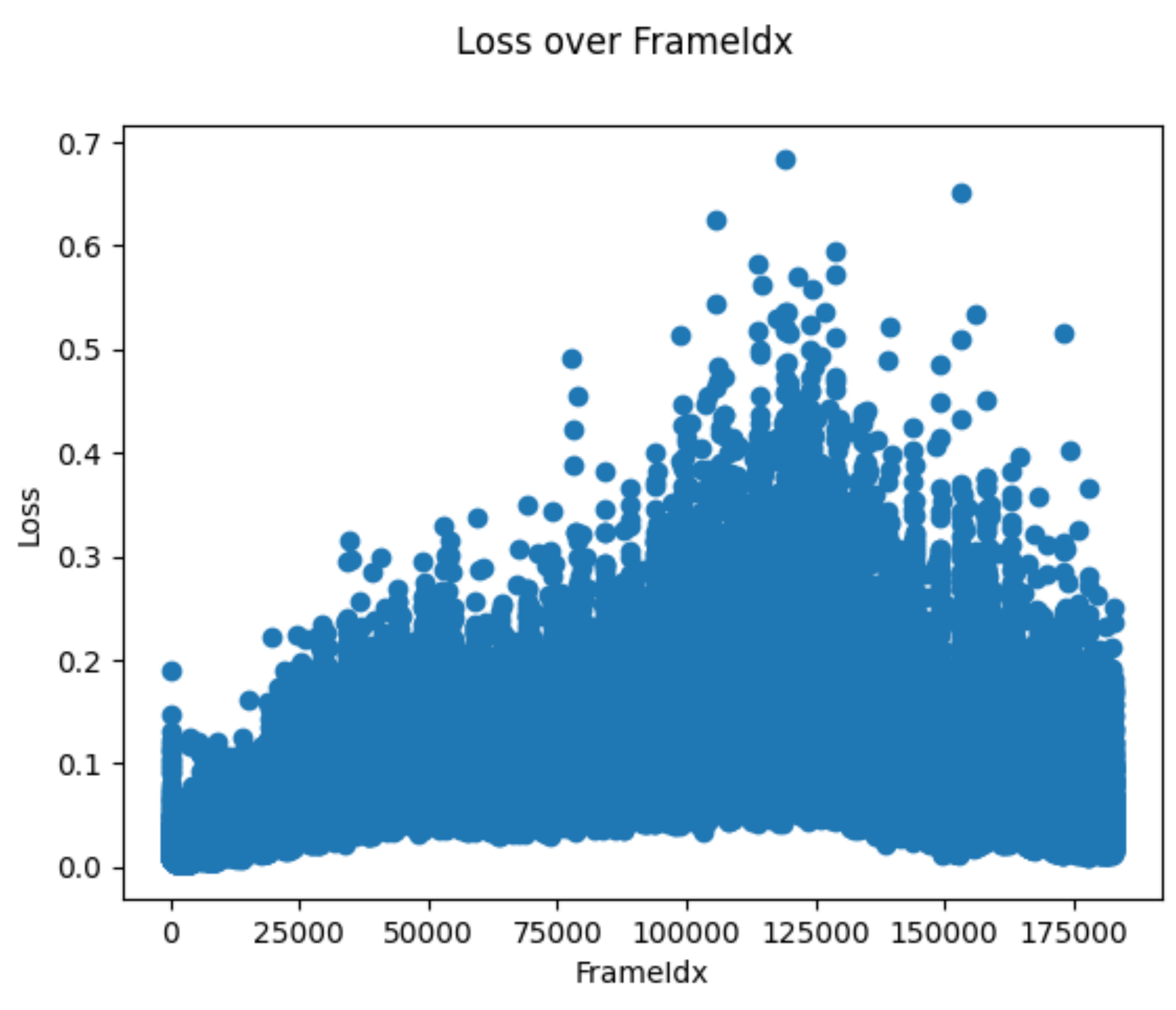
y_i represents the target_model value for a given action. s represents the current state and a represents the action our model took at state s. θ_i represents the parameters of our model. $Q(s, a; \theta_i)$ represents the Q value of an action a given the current state s.

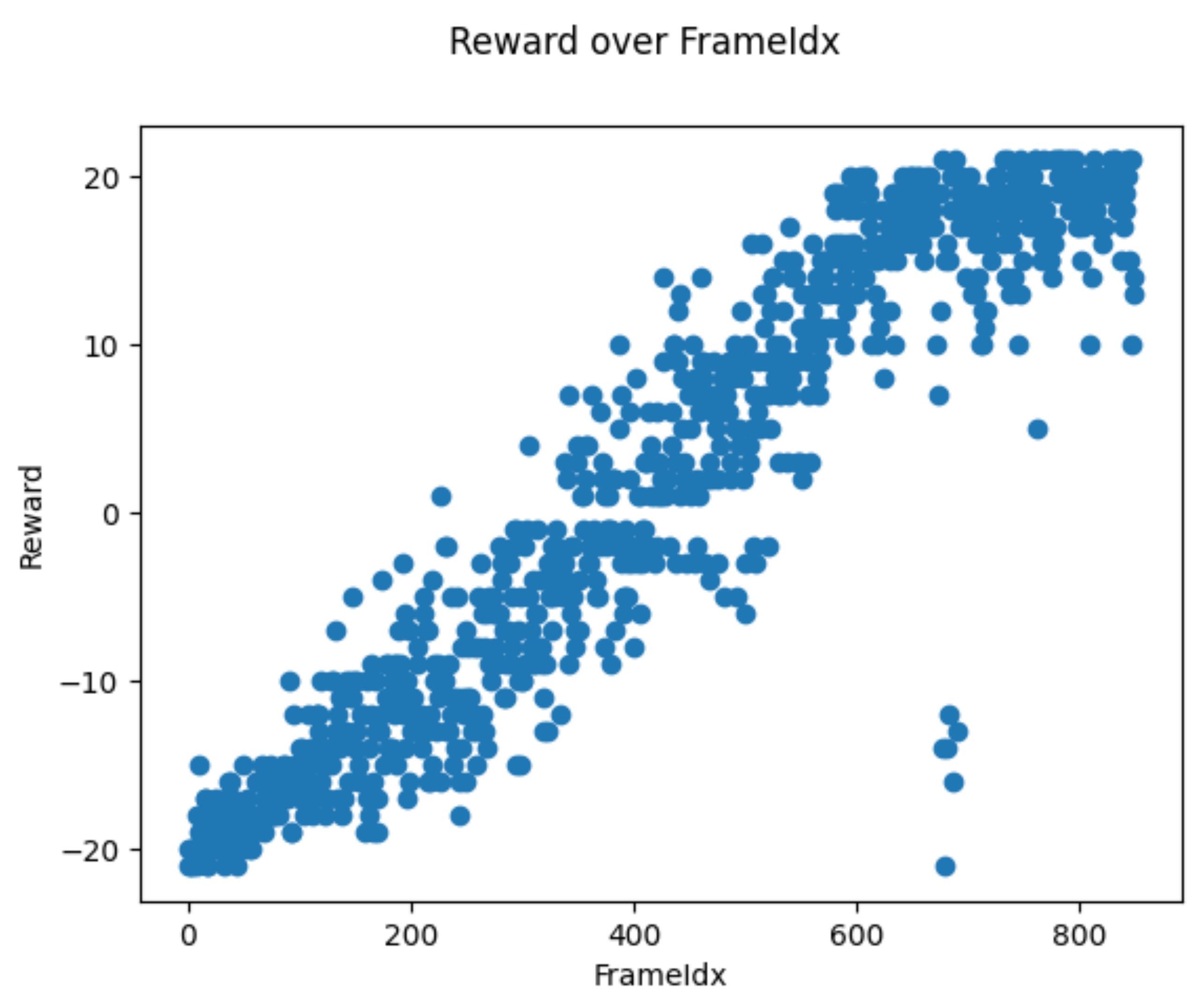
Our model learns from this loss function because it approxiately tells how far we are from the desired Q-values (from our target_model). Since our Q-values monotonically increase, by minizing our loss, we can approach the actual real Q-values without overestimating the value for a given action.

Problem 4 - Learning to Play Pong

Part 4

Plot how the loss and reward change during the training process. Include these figures in your report.


Loss over FrameIdx
Loss
FrameIdx


Reward over FrameIdx
Reward
FrameIdx