Q1,
(1) the state as the game display (image) will be easier for the agent to learn from, because AI can learn from the image to identify if the game is “done”, and also the image gives information about where the ball is and where the paddles are. This allows AI to learn the motions and train as human being players.

(2) The inputs are the states, and the outputs are the Q value of each action, with actions as the index. Instead of building the Q table, here we use neuron network to take in a state and output the corresponding state’s q values of each action.

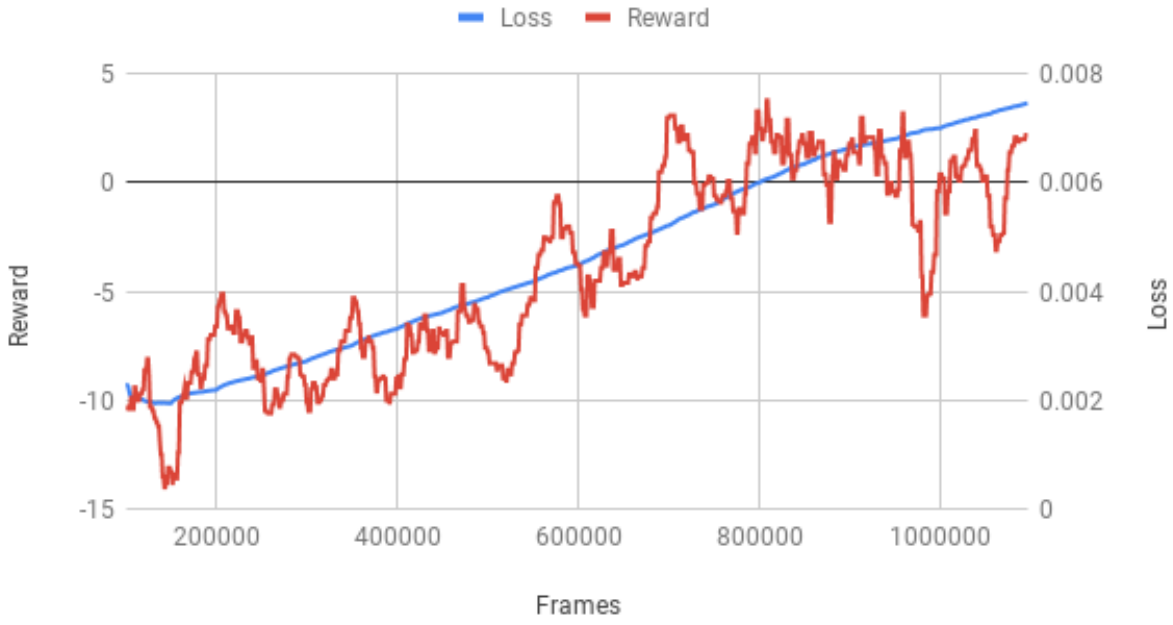
(3) Line 48 and 57 are from act(self, state, epsilon) function. This function has two parts, explore or exploit. At first, epsilon starts with a high value, because when the AI is immature, we want to explore more, so the following action is generated randomly; later, when AI trains more and becomes mature, it is more reliable to exploit, then we compute the q values from self.forward(state) and take the action which has the highest q value.

Q2,
(1) $Loss_i(\Theta_i) = (y_i - Q(s, a; \Theta_i))^2$
y_i means the i-th iteration of the q value for the current model, in the code, “model”. This takes in current state.
$Q(s, a; \Theta_i)$ is the i-th iteration of the q value for the target model, in the code, “target_model”, this takes in next state and uses the update rule: reward + gamma * maximum q value generated from the target model.
Loss is the mean squared error of these two variables.
Why this loss function in particular results in learning a good model? Because this function guarantees we learn the motion towards the maximum q value in the target model and converges to a local minimum of loss.

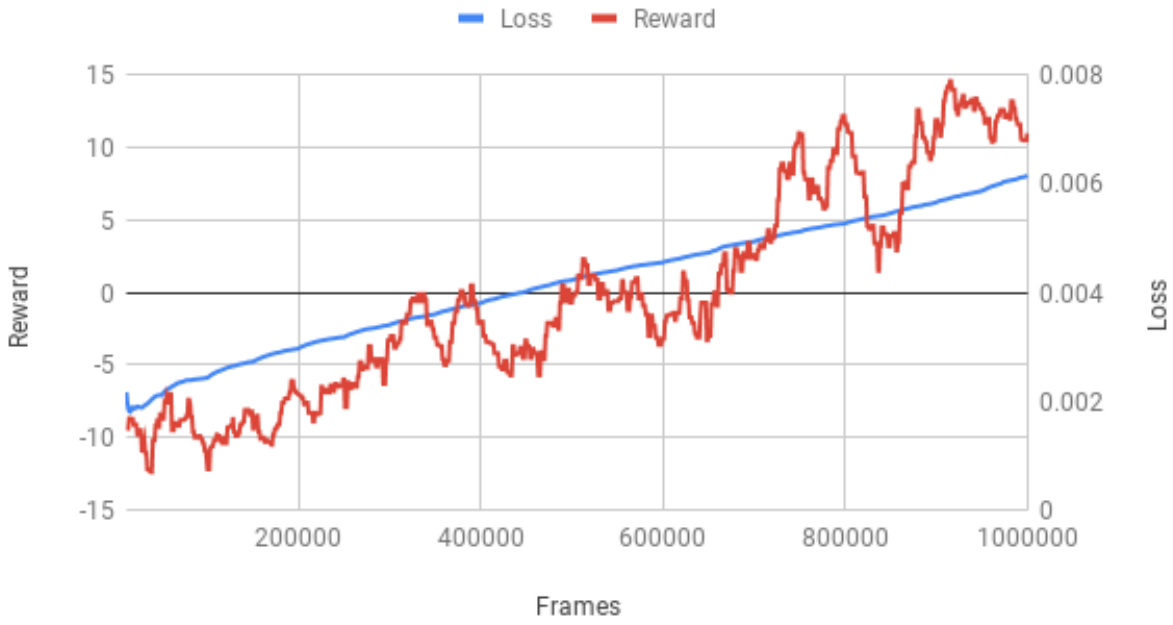
Q4,
(4) First run reward only went up to about -4.1 with the default setting, but the second run after adjusting hyperparameters is significant better. I tried to adjust the replay buffer size, gamma, and also batch size. It turns out changing the size of batch is the best solution. Big gamma (0.999) gets very close to 11.0. But for the big buffer size 1 million, 1,000,000, reward only went up to around 2.0. For the batch size 128, the reward increased to around 13.0, although the loss went up as well.

First Run (Default Hyperparameters)

Big Buffer Size - 1,000,000



Big Gamma - 0.999



Second Run (Batch size 128)