ECE 448 FALL 2020
Assignment 5:

Reinforcement Learning & Deep Learning

Dec. 20, 2020

Yucheng Jin

Yiqing Xie

Hangtao Jin

Part 1

1. Implementation of agent snake:
   Train phase: During the train phase, the agent updates Q table and N table, then selects next action of the snake.

   Test phase: During the test phase, the agent only selects the next action of the snake.

2. Training and testing result:
   Best parameter:
   Ne: 15
   C: 25
   Gamma: 0.9
   Training time: 41.4156s
   Average point: 23.771

```
Training takes 41.41558861732483 seconds
Test Phase:
Loaded model successfully
Testing takes 4.32096266746521 seconds
Number of Games: 1000
Average Points: 23.771
Max Points: 51
Min Points: 1
```

3. Changes made to this MDP
   (1) State configuration:
   We find that wall adjoining parameter and body adjoining parameter can be combined, which means we can use 4 parameters to represent 6 parameters in the original setting. If we only change the state configuration, the average point of test increases by approximately 0.5. The performance is better with this setting because as the state space is smaller, the average training time for each state increases, which leads to an increase of performance.

   (2) Exploration policy:
   As the snake will die once it hit its body or the walls, we can ignore all actions that lead to death when making decision. By set this rule, we can find a remarkable increase in average point (increase by approximately 2-12). The performance is really unstable.

   (3) Reward model:
   The reason for changing the reward model is similar to the reason for the adjustment of exploration policy. We should punish the dead action severely. Here we set the reward for dead to be -100 instead of -1. The increase of average point is quite similar to that of last modification, increase by about 2-15. The performance is also unstable.

Part 2

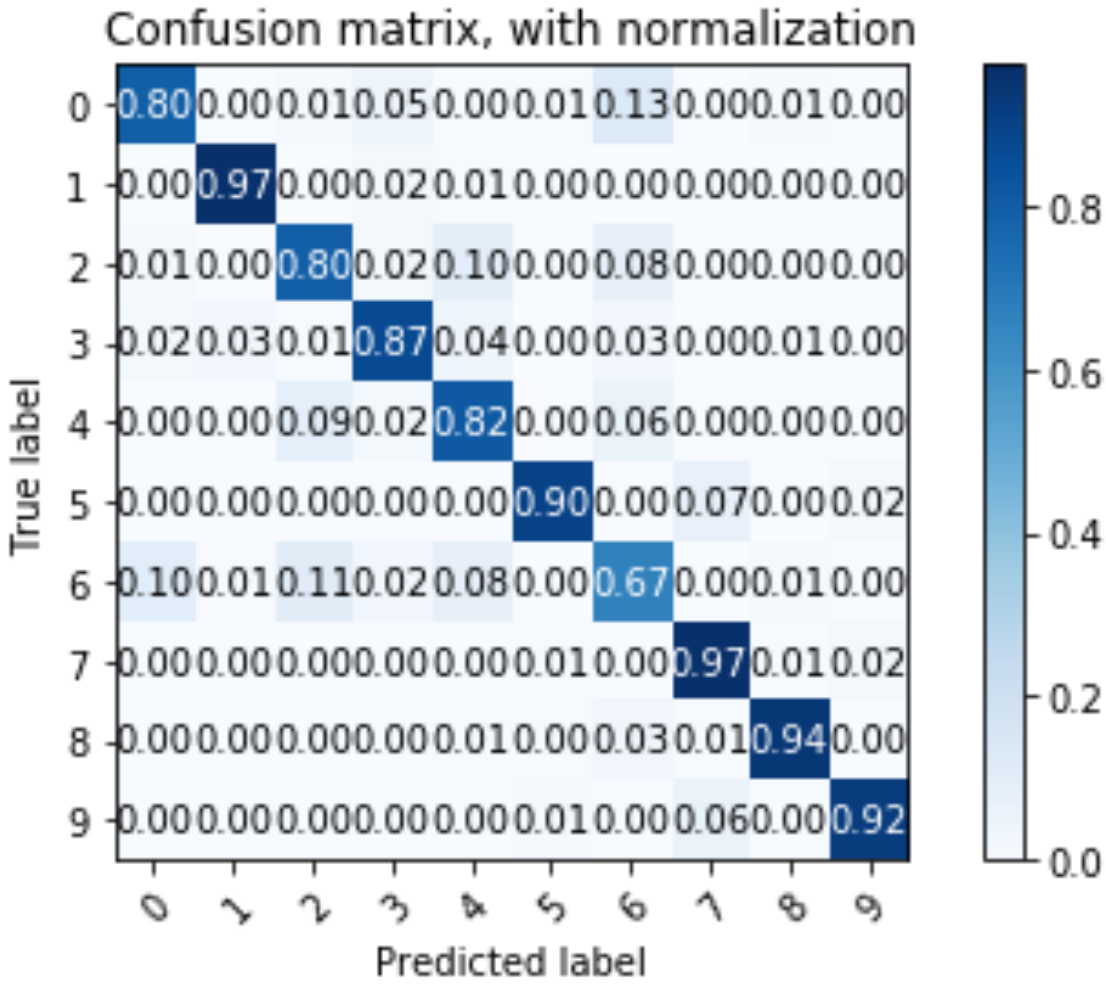
1. Briefly describe any optimizations you performed on your functions for faster runtime.

For function affine_forward(A, W, b), at first I calculated the product by some “for loops”, but later I used np.matmul(A, W) to accelerate calculation. In practice, np functions are much more faster than “for loops”.

2. Report Confusion Matrix, Average Classification Rate, and Runtime of Minibatch gradient for 10, 30, 50 epochs. This means that you should have 3x3=9 total items in this section.

For 10 epochs:

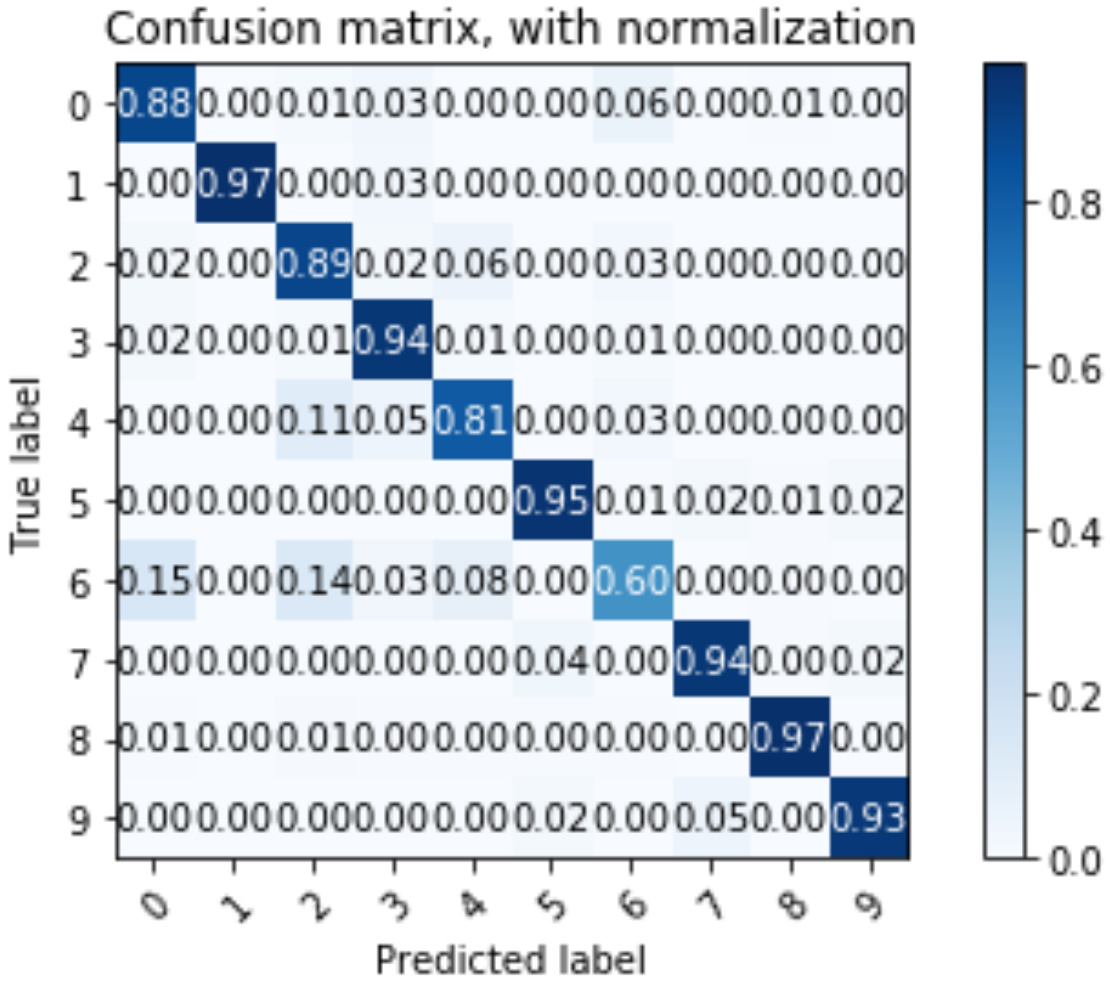
Confusion Matrix:



Average Classification Rate: 0.8662

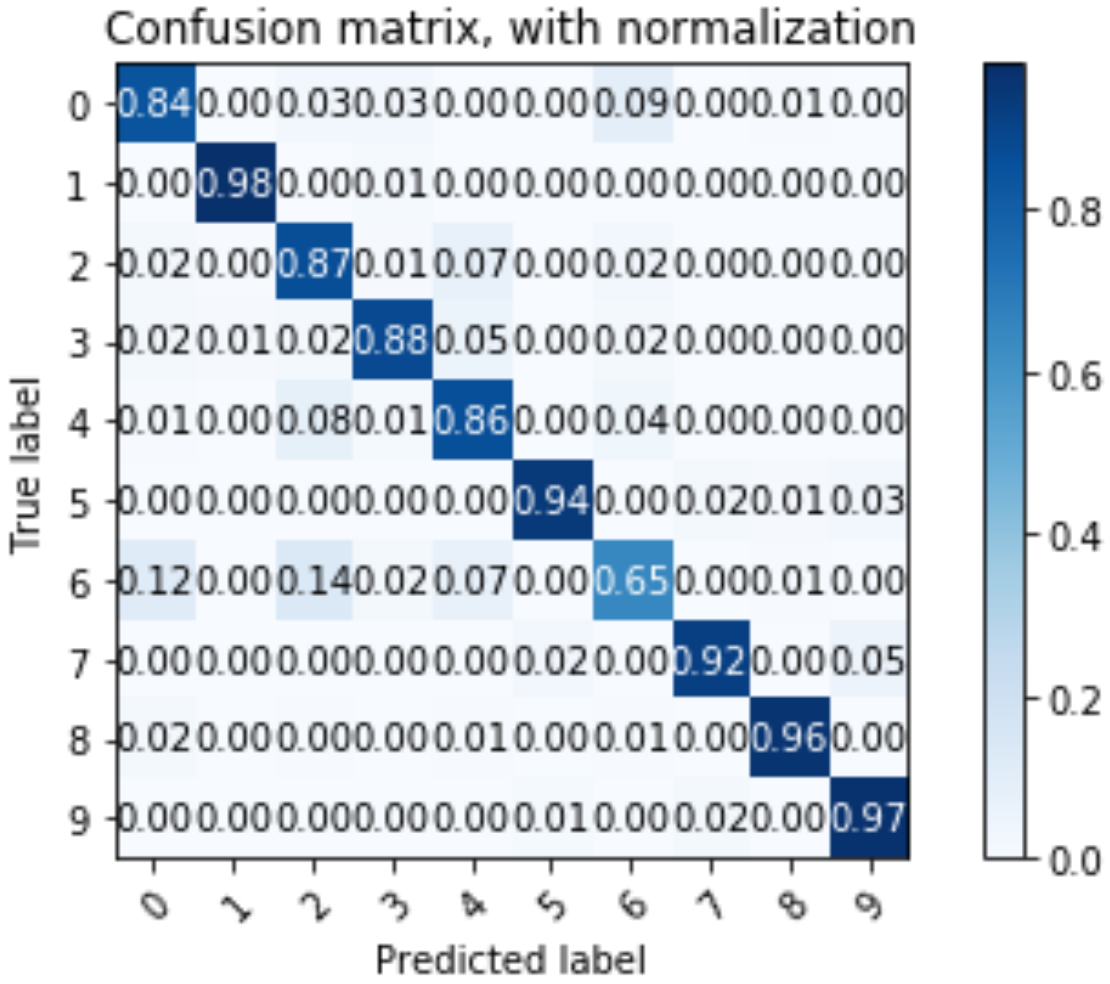
Runtime of Minibatch Gradient: 20.721543788909912 s

For 30 epochs:
Confusion Matrix:



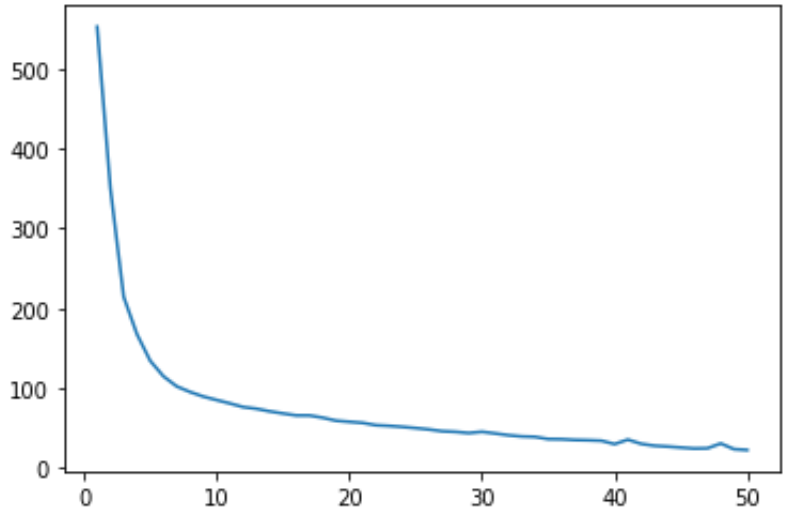
Average Classification Rate: 0.8872
Runtime of Minibatch Gradient: 64.49502491950989 s

For 50 epochs:
Confusion Matrix:



Average Classification Rate: 0.8876
Runtime of Minibatch Gradient: 111.7649519443512 s

3. Add a graph that plots epochs vs losses at that epoch. (For 50 epochs)



4. Describe any trends that you see. Is this expected or surprising? What do you think is the explanation for the observations?

As the number of epochs increases, the loss decreases, this is what we expected. Because the training process makes the neural network more accurate, which makes the loss smaller, also, the overfitting problem has not occurred when there are only 50 epochs.