Report

In the project, there are two methods to use Qlearning algorithm to train the dots and boxes game.

Description of Game:

Dots and Boxes Game need players take turns to draw edges, and if one of the player make an enclosed box, then he/she can do a continuous draw.
At last, when all the edges are draw, the winner would be the one who has most boxes.

Description of Q learning:

The Q learning algorithm in this project, is trained by a random player and also a Q learner.

There are two methods used in the project:
the first one is the Qtable. Every time player draw a line, the environment will give a reward for each user. After this, agent will pass the reward and also board status to Q_table file, which will generate and store all Q values.

Using equation as following to update the Q table.

$$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$$

For Deep Q learning. Here I used three hidden layers and one output layer. The input data is still combination of states and move. The output of the network is estimated Q value.
Still use the upward equation as target of the network.

Result of the Project:

For the Qtable:

As the following two figures shows , the Q table method can learn rapidly at beginning, but with the increase of iterations, the improvement becomes slow.

As the iteration goes up, the accuracy is growing.

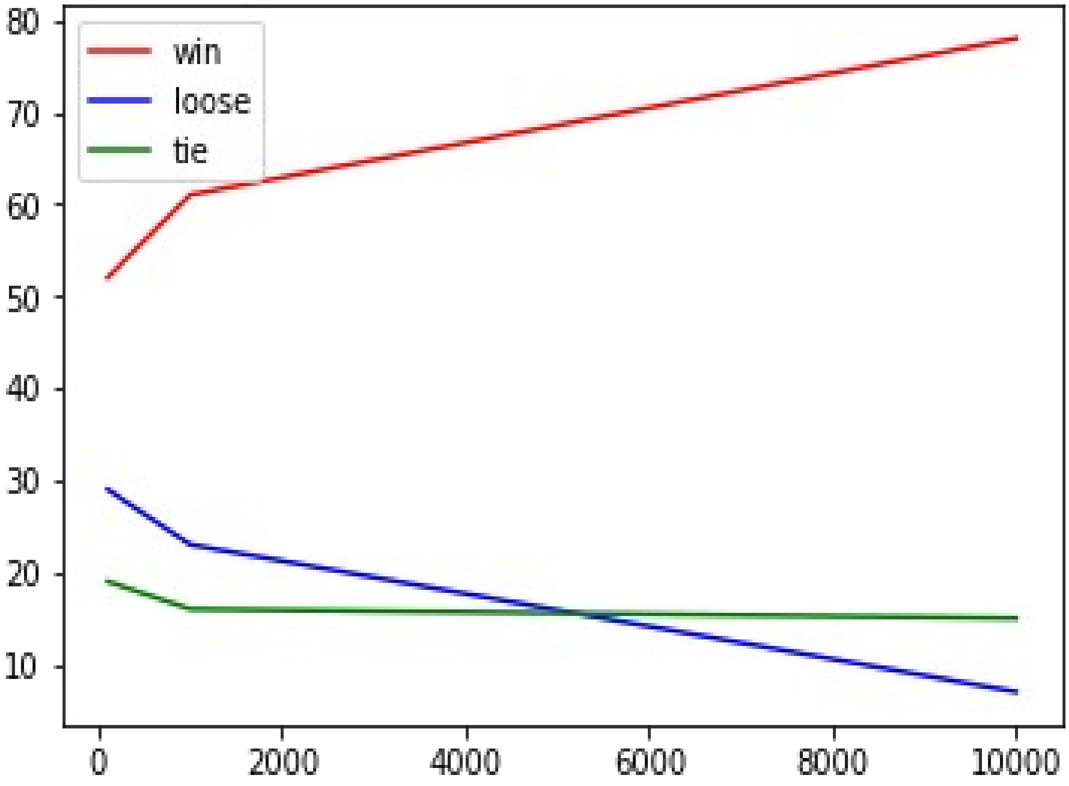

Figure of 2 grid size

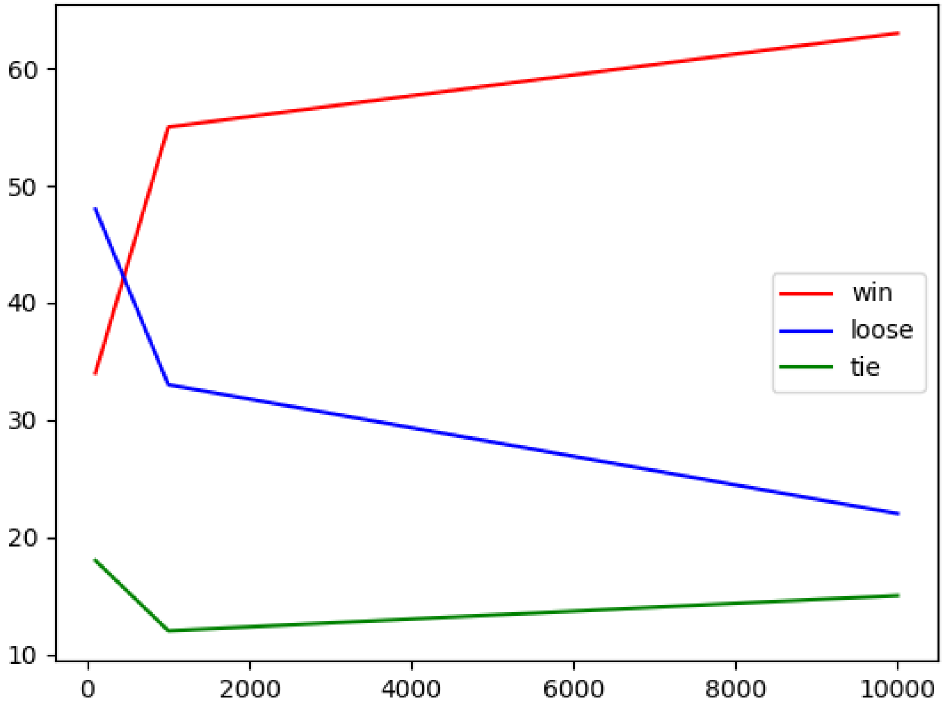

Figure of 3 grid size

For Deep Q learning:

As the following figure shows, with the iteration goes up, the learning efficiency is going down, which is because I used sgd with learning rate =1. It is a big value.

With the same iteration times, the neural network has a better performance than Q table. And the play rate is more than 80% after 10000 iterations.

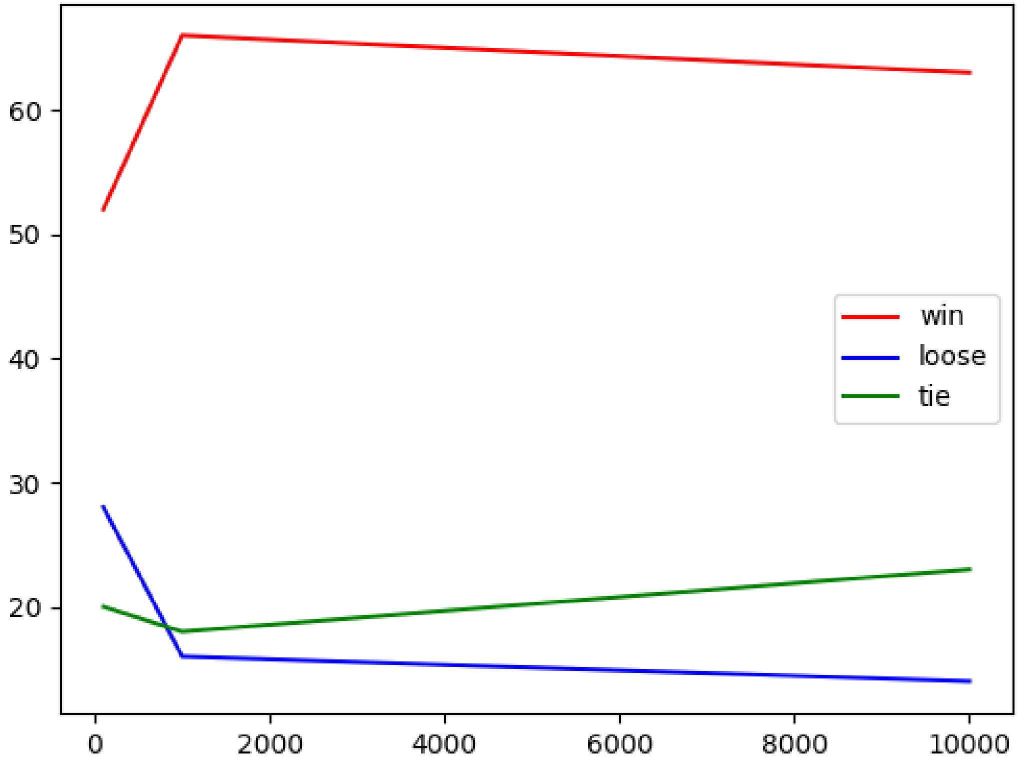

Figure of 2 grid size using DQL

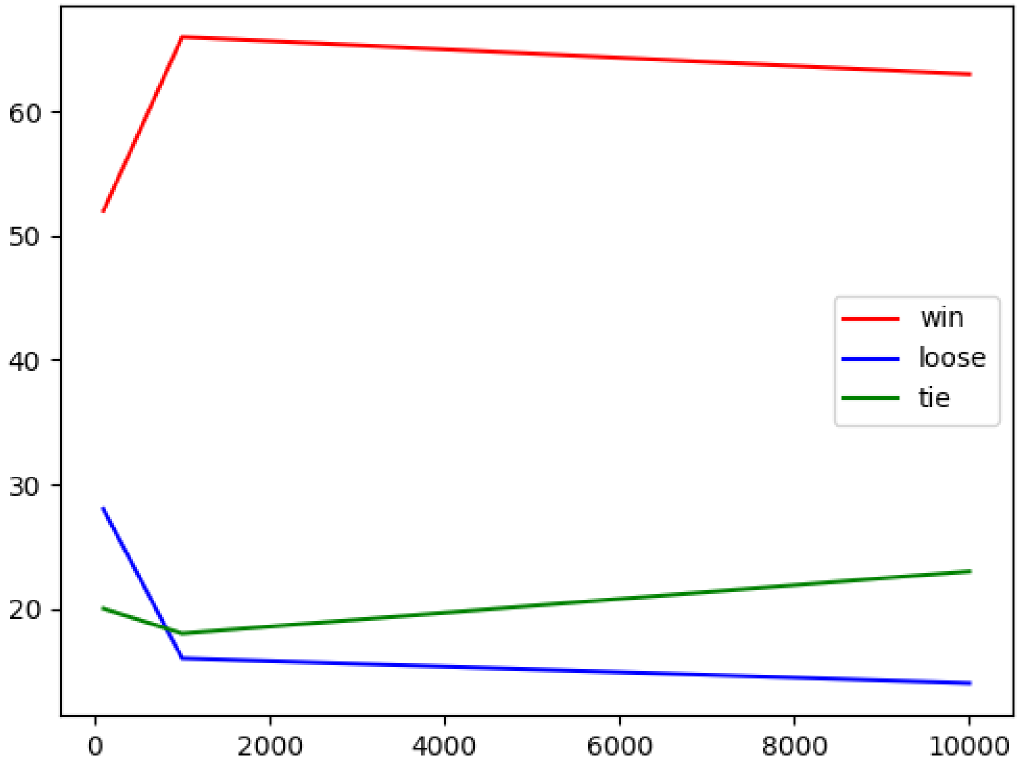

Figure of 3 grid size DQL