User Manual: A Neural Net based Implementation of Tic-Tac-Toe

When discussing the game of tic-tac-toe, one would think that the goals of the game and how we arrive at these goals are relatively simple. Two players take turns penciling in X's and O's until one player has three of them in a row. In the event that neither player achieves three in a row, the game results in a draw. When our team first started discussing ideas and whether we were going to implement it for our final project, we acknowledged that tic-tac-toe is a widely known game that has been played throughout history in all parts of the world. We also acknowledged that the implementation of the game would not be too far fetched in difficulty and would provide us with a solid foundation in the event we had more time to expand on it. Everyone in the group was also familiar with the game and the different strategies that could be used to win, which prompted curiosity in how a neural net based implementation of tic-tac-toe would behave toward a human player.

As we continued to research more about tic-tac-toe, there were clear indicators from various sources that the simplicity of the game allowed for behavioral observation regarding the two players; The player who goes first will win more often than not unless a mistake is made on their part, in which case the chances of player two winning drastically increases as well as the chances of the game resulting in a draw. This power dynamic between the two players became more interesting as we questioned if/how a neural net would become aware of it, and how it would behave differently as it learned the pre planned dynamic within the game. The other behavioral aspect of tic-tac-toe is the strategy that is used to win the game. Generally speaking, two people that have just learned the game would seek to achieve three in a row, blocking the opponent when they need to, and then returning back to their efforts of putting three X's or O's in a row. However, when a player achieves two opposite corner spaces with the middle being

available, it is impossible for that player to lose unless a mistake is made. In the event that a mistake is made and both players have almost perfect strategies, the result of the game is more often than not a draw.

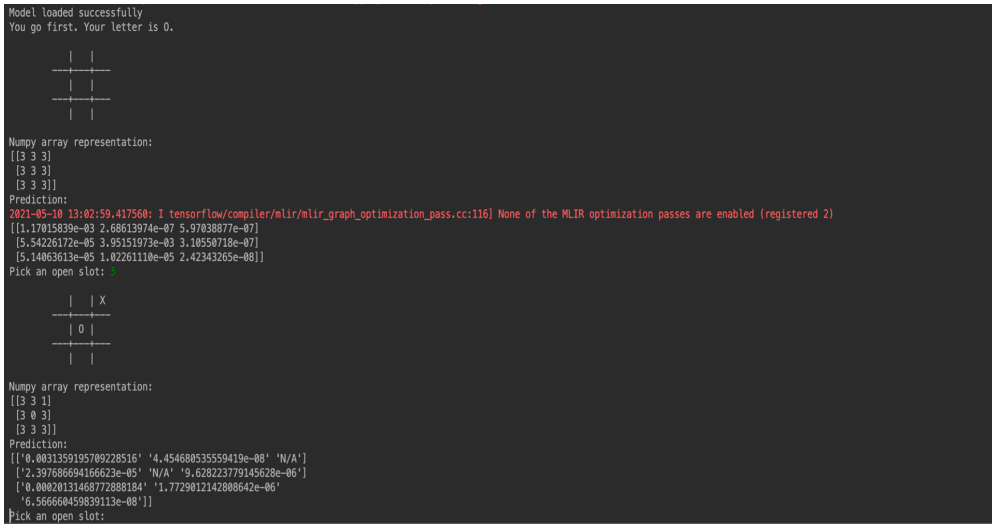
We ultimately decided to create a neural net based implementation of tic-tac-toe to observe and recognize the humanistic behaviors within the game. We do this by having the neural net output predictions for each cell that it thinks the human may choose as well as outputting a move of its own so that the game continues as it predicts. We chose a long short-term memory (LSTM) neural net as it is very helpful in predicting the next action while having past sequences. The general process begins with an empty board printed to the terminal with a statement that asks you to pick an open slot (1 through 9). In this case, the board and the potential slots picked is represented a such:

[1 , 2 , 3

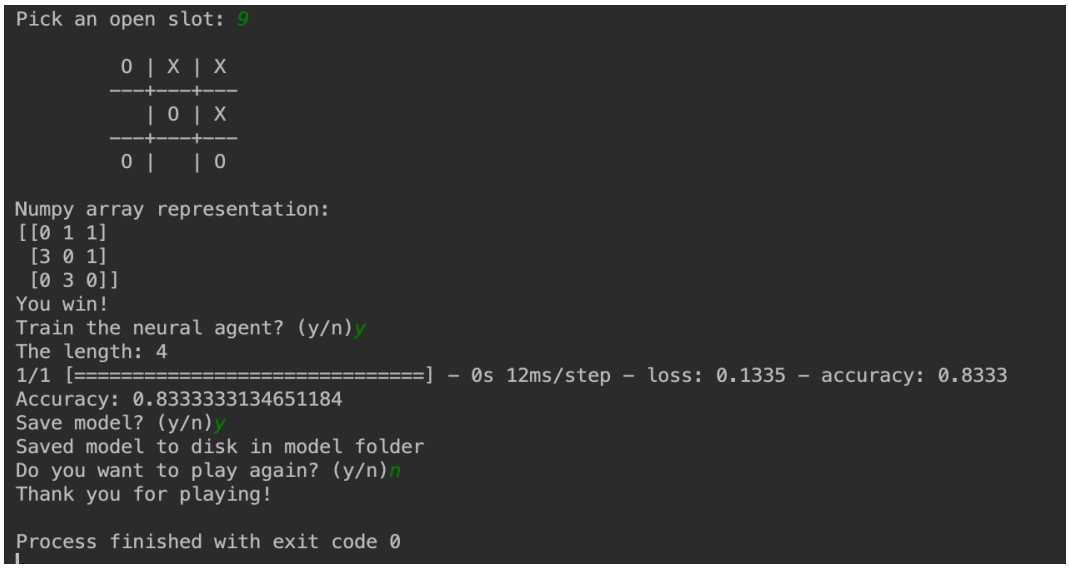
4 , 5 , 6

7 , 8 , 9]

With each move that is played, a human readable tic-tac-toe board is printed to the terminal, along with a move from our AI and the likelihood of each individual cell being played by the human for our next move. The first board you see below is the start of the game, and in this case we were picked to go first (who goes first is picked randomly each time as we want to make the power dynamic as obvious as possible for our net). The example then shows that the human chose slot 5, which represents the middle square as seen above. Then the program outputs the same game board with both of our moves, as well as predictions for the humans next move.



As the game comes to an end, the program offers a few options to the human player. The player can respond to these options by typing “y” for yes and “n” for no. The first option is to train the neural agent, which essentially takes each state of the game we just previously played, inputs them into the neural agent, and compares its predictions during the game with what actually happened. In this specific example, we can see that our neural agent had an accuracy of 0.83 in predicting what we were going to do. The last two questions regard saving the model for future learning and playing again.



Here is the input and output diagram of what a neural model is training: