Ultimate Tic-Tac-Toe

Neural Network

Team Name: Team JS

Team Members: Stephanie Yu, Justin Chin

Problem Statement
The problem we are tackling is that users can not play Ultimate Tic-Tac-Toe by themselves, so therefore we will create an AI for the users to play against. Therefore, there will be no need to find another player to play Ultimate Tic-Tac-Toe. There will potentially be different AI difficulties that the user may choose, or even a human opponent if chosen to do so.

Setting
The setting we are considering is using Tensorflow, because the library should be beneficial for creating our AI. Because TensorFlow is an open-source library known framework for AI applications, we can use and train deep neural networks, the moves of Tic-Tac-Toe, and use it to power up our AI in difficult settings.

Method
We will use Neural Network for this project because it is the most ideal in creating an AI that will make the next best possible move to win the game.

Expected Result
We expect to have a working UI that enables the user to create a new game that creates an empty Ultimate Tic-Tac-Toe board that allows the user to play a game of Ultimate Tic-Tac-Toe. We then will have the user be able to place their “X”s or “O”s on the board through an input on each of the user’s turns. On the AI’s turn, the AI should be able to place their piece optimally for it to win the game. We want to train the AI by playing a large amount of games with a computer that will always place their piece randomly. After playing many games, the AI should be able to learn from random playing how to make the best move to win. The end goal would be to have the AI to be able to almost always win.

Dataset
We will essentially create our own dataset for our neural network. Our AI will be trained to learn how to make the next best move through many played games (40,000~) with a player that will always place their piece randomly.

Plan
After our given preliminary results of all the games/data of Tie-Tac-Toe, we want to implement this set of data into our Ultimate board, where there would be 9 total boards of tic-tac-toe going at once. With the data we have, we want to train the AI to continuously play against each other to get better, and have the option to play against a user. These are the future steps we are considering.

Ultimate Tic-Tac-Toe Rule Overview

Ultimate Tic-Tac-Toe is played with a main bigger Tic-Tac-Toe board with a smaller Tic-Tac-Toe board in each of the nine sections in the bigger board. Players will take turns placing their 'X' or 'O' piece in any of the nine boards. The player does not need to place the piece in the same board. Once a three in a row has been achieved in any of the nine boards, the winning player will then win that board and have gained one win. When that section of the board is won, the remaining possible movements in the same section are then crossed out. When a player reaches five wins, they will win the game. If there are no more possible moves and neither person has won, the game results in a tie.

Chosen Algorithm: Neural Network

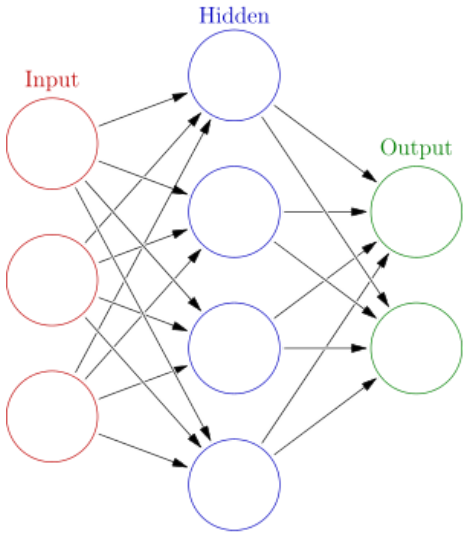
Neural Network provides the most effective model when given the best dataset for our project. It allows for efficiency and decision-making, providing for the best accuracy. The downsides however, is that it is slower than other algorithms, and it really does require a really large data set for it's best potential.

Why not other algorithms (Comparative Analysis)?

- **Linear Regression:** Linear Regression works better with less training data. With the amount of training data we will use for this project, linear regression will not be prefered. The accuracy would also be lower with linear regression than in neural networks.
- **Logistic Regression:** Logistic Regression works better with less training data. With the amount of training data we will use for this project, logistic regression will not be prefered. Logistic Regression also can not support non-linear solutions, which Neural Network can support.
- **Support Vector Machine (SVM):** SVM also can not support non-linear solutions. It is also not suitable for larger training data.
- **Decision Tree:** Neural Network will work better if there is a big enough training data.

Neural Network

Our neural network has an input layer of 81 neurons because we have 81 possible slots to put one's pieces onto. The output layer will have one neuron that will have a value between -1 and 1. A number closer to -1 will show that the AI will lose and a number closer to 1 will show that it will win.



Loss Function

We will use Mean Square Error since it is generic and is a popular loss function for regression.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

UI

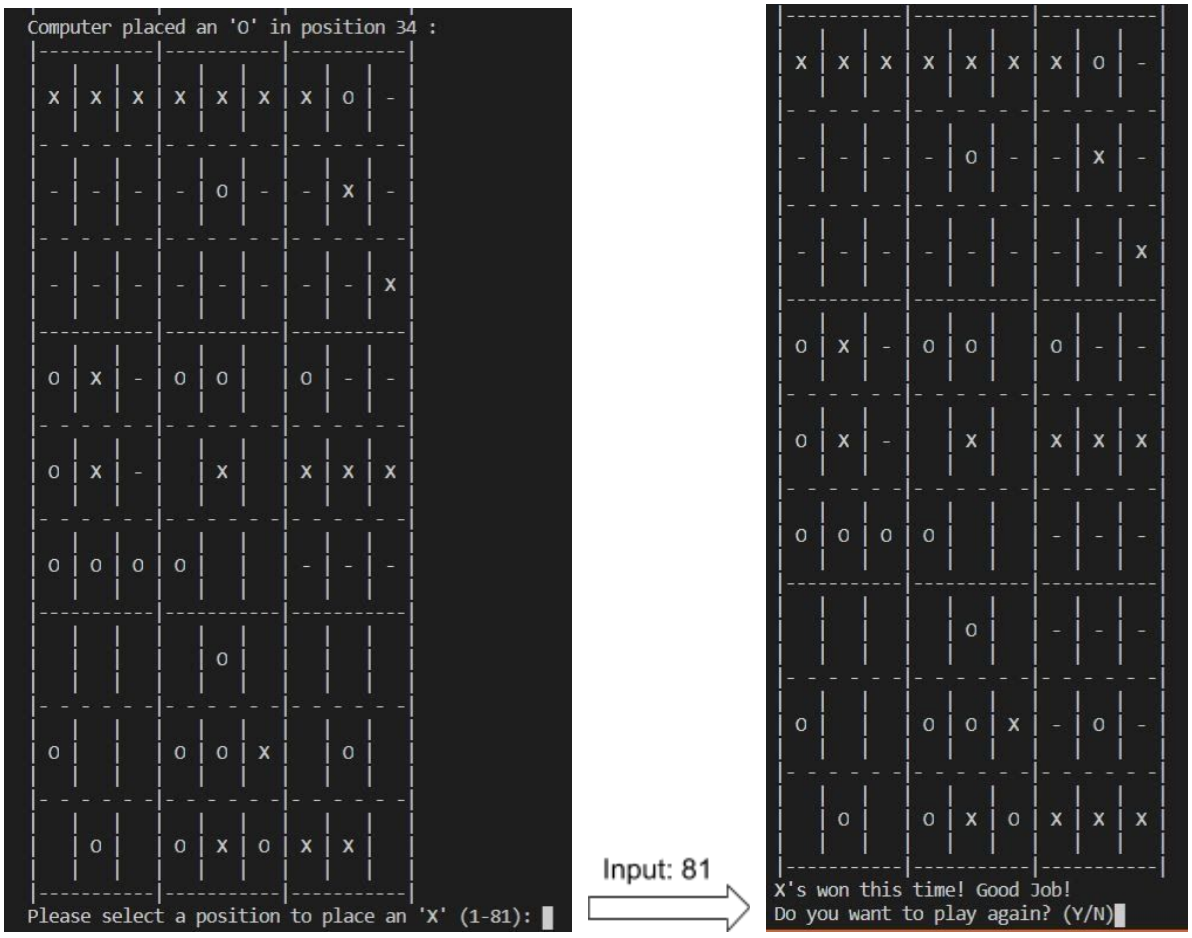
Our UI is shown in text in the terminal when you run the program. The grid layout will be similar to what is shown below. The user will be prompted to enter the location where they would like to place their piece in. The user will be able to enter a number between (1-81) and their piece (User will always place 'X') will be positioned in the same location that corresponds to the layout shown below. After the user enters their piece, the AI bot will then take its turn and then back to the user. When one of the tic tac toe games has been won, the rest of the box will be displayed with a '-'. The game will end when either the user or the AI bot wins five Tic-Tac-Toe games out of nine.

1	2	3	10	11	12	19	20	21
4	5	6	13	14	15	22	23	24
7	8	9	16	17	18	25	26	27
28	29	30	37	38	39	46	47	48
31	32	33	40	41	42	49	50	51
34	35	36	43	44	45	52	53	54
55	56	57	64	65	66	73	74	75
58	59	60	67	68	69	76	77	78
61	62	63	70	71	72	79	80	81

Layout

Results

We were able to create our game board and made it functional. It enables the user to initiate the game by entering ‘y’. After that, the user will be the first to move and input which location they want their ‘X’ piece to go to. The computer will then take its turn. This cycle will repeat until the user or computer wins five Tic-Tac-Toe games. A win for one Tic-Tac-Toe board is shown when the rest of that board is filled with ‘-’. Originally, we wanted the Neural Network AI to use this board to learn how to make the best move. The random computer player had been implemented, but we were not able to get the neural network AI working. We had ran into problems fully understanding how to utilize tensorflow and keras and was unable to create a neural network for this game.



Next Steps

To further continue on in this project, we would have to research and understand more on how to write the neural network in python. Some possible obstacles that had hindered our progress would be using a new programming language. Our team has never used python before and it has hindered us from fully understanding how to implement the neural network part of our project.

Resources

- https://techwithtim.net/tutorials/python-programming/tic-tac-toe-tutorial/?fbclid=IwAR04jmdJofFVzrLG4ju3LnzycnbluKyE2fVJYe0w-gQqwkESKxLF0X2ktPE
- https://medium.com/@shayak_89588/playing-ultimate-tic-tac-toe-with-reinforcement-learning-7bea5b9d7252
- https://www.codeproject.com/Articles/5160398/A-Tic-Tac-Toe-AI-with-Neural-Networks-and-Machine
- https://www.tensorflow.org/
- https://keras.io/
- https://numpy.org/
- **https://upload.wikimedia.org/wikipedia/commons/thumb/4/46/Colored_neural_network.svg/300px-Colored_neural_network.svg.png**
- **https://medium.com/@carsten.friedrich/part-4-neural-network-q-learning-a-tic-tac-toe-player-that-learns-kind-of-2090ca4798d**
- **https://www.freecodecamp.org/news/machine-learning-mean-squared-error-regression-line-c7dde9a26b93/**

Dependencies

TensorFlow
Keras
Numpy

Contribution

- Stephanie Yu
  - Report - 50%
  - Board implementation - 50%
  - Neural Network implementation - 50%
  - Demo Video - 50%
- Justin Chin
  - Report - 50%
  - Board implementation - 50%
  - Neural Network implementation - 50%
  - Demo Video - 50%

Github Link

https://github.com/syu25/cse474

TA

Junxuan