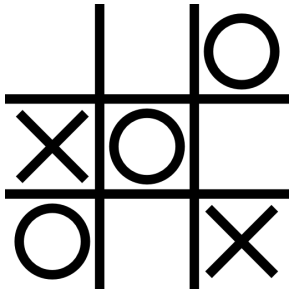
Final Report

Tic Tac Toe Using Artificial Neural Networks



EN250 Introductory Quantitative Biology

"I pledge my honor that I have abided by the Stevens Honor System"

Trent Berrien

March 11, 2021

Table of Contents:

Introduction:

The purpose of this research is to provide insight into the historical origins, mathematical principles, and applications of artificial neural networks (ANNs). The artificial neural network model is inspired by how a biological brain operates. Most artificial neural networks can be broken up into three sections; the input layer, the hidden layer(s), and the output layer. Within each layer is a select number of "neurons". These neurons are able to take in several inputs and send their output to several other neurons, similar to how biological neurons behave. Each layer is connected to the layer adjacent, typically in a forward feeding manner, where information flows from one end to the other.

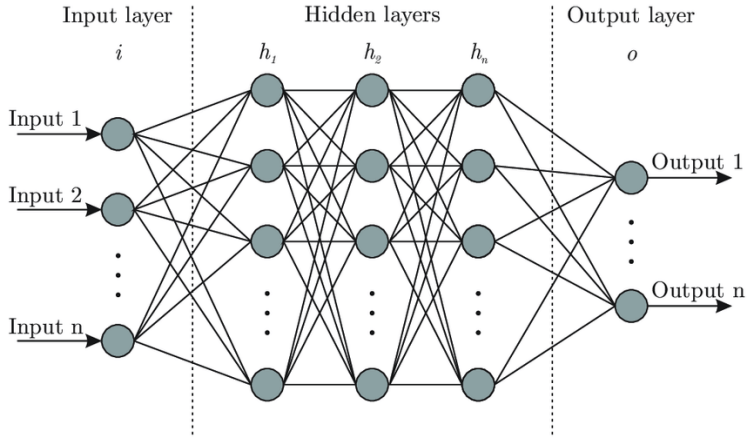


Figure 1.1 Artificial Neural Network Diagram

The figure above is a diagram showing a sample of a possible neural network configuration. Each circle in the figure represents a neuron, and each line represents a connection between two neurons. In order to create a useful model, the strengths of each connection can be tweaked, along with other constants within the network. This is how the artificial neural network model transforms from an interesting concept to a highly powerful tool that can perform many tasks.

History:

The history of this model starts in 1943, when Warren McCulloch and Walter Pitts are credited with making the first computational model based on organic neural networks. Later in 1958, Frank Rosenblatt created what was known as the “perceptron”.

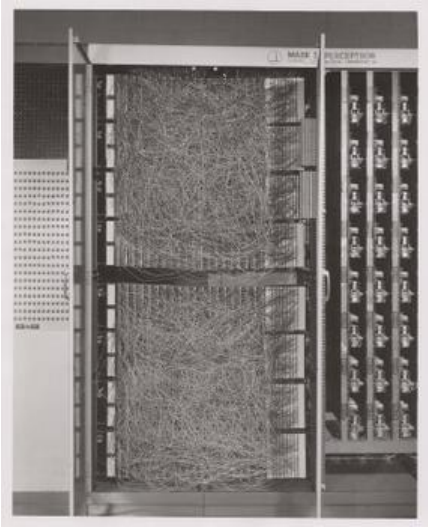


Figure 2.1 Perceptron Machine

This device was trained to be able to scan photos of people and decipher between males and females with relatively high accuracy. This led to much excitement surrounding the technology into the 1960’s. Due to the lack of processing power at the time, this excitement dwindled in the 1970’s. Although progress was made in the theory of how more complex versions of these networks could be taught in this time, progress mostly stagnated as the technology could not be practically used. This excitement was reignited in the 1980’s when new advancements had increased transistor counts in electronics. With the advancement of GPU’s in 1990 to the present day, ANNs continue to become increasingly complex, capable of performing increasingly difficult tasks. This model has now been implemented into many applications, even finding its way into everyday use in smartphones, and continues to push boundaries.

Mathematical Properties:

To understand what is happening mathematically within this model, a slice can be taken looking at the interaction between a select few neurons.

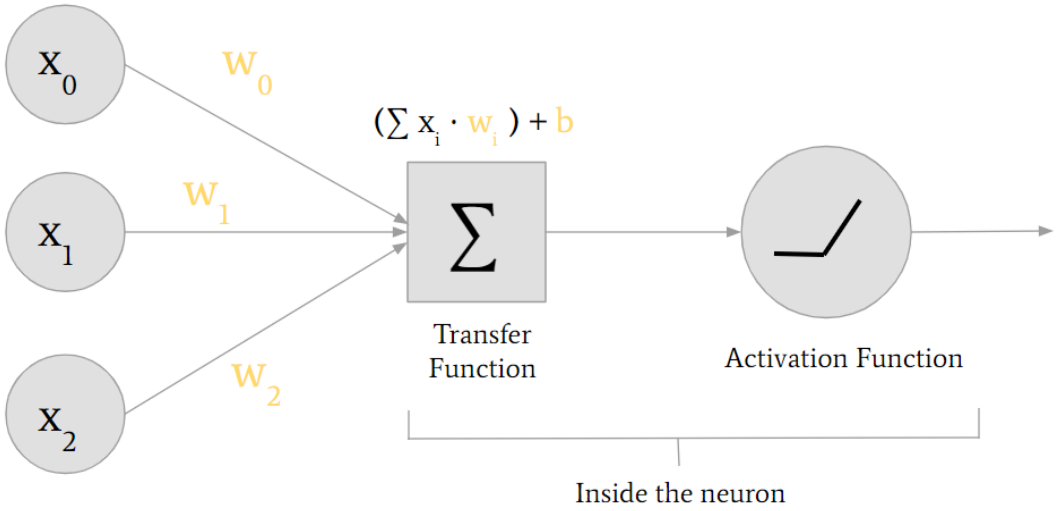


Figure 3.1 Example Slice of ANN

The figure above shows the interaction between 3 input neurons and one neuron of a hidden layer. Each input is denoted by x_i, and their respective connection weight is denoted by w_i. These values are passed to the transfer function of the connected neuron, which sums the product of each x_i with its respective w_i. Also within the transfer function, a constant known as the bias (a predetermined constant denoted as “b”) is added to the resulting sum. The function of this bias constant is to shift the sum required to activate the neuron. This final summed value from the transfer function is then passed to the activation function within the neuron to determine the level by which the neuron is activated or deactivated. This activation function can vary depending on the design of the network. Many networks, especially earlier ones use the sigmoid function, s(x) = 1/(1+e^-x). What this function does is take values on the high end to only

approach 1, and values on the low end to approach 0. Many functions used for the activation serve a similar purpose of limiting the range of outputs. Another function used in more modern networks is the Rectified Linear Unit function, otherwise known as the ReLU function. For values less than zero, the output is zero, and for values greater than zero, the function has a slope of 1 with a y intercept of zero.

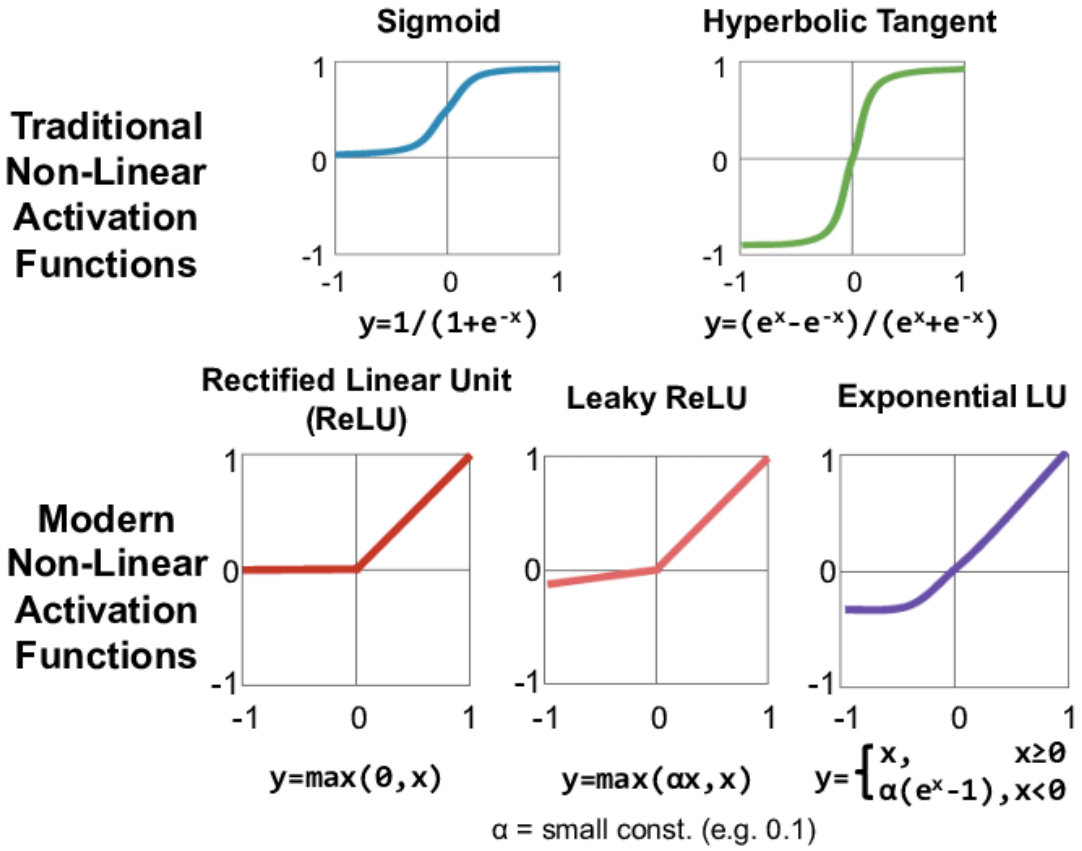


Figure 3.2 Plots of Activation Functions

Above is a figure showing the aforementioned functions, along with some other iterations. The value output from this function is then the value associated with the neuron. With this new calculated value, the neuron in the hidden layer can be treated just as the input neuron was. This process of passing values to the transfer and activation functions is continued through all layers until the final values are calculated in the output layer.

Applications:

While the origins of this model stem from the fields of biology and neurology, it's applications are quite vast and can be used in many industries. The first use for artificial neural networks, that still remains a frequent use of this model, is for identification and recognition. This technology performs exceptionally well in the area of recognizing patterns. Some modern examples of this include facial identification software, as well as voice recognition software. A subset of this recognition is pattern recognition and predictions. For example, an artificial neural network can be trained to make predictions on the stock market given data on previous days. Another example of this is to predict natural disasters such as forest fires. Given data on time of year, humidity, temperature, recent rainfall, and other relevant data, a neural network could be trained to determine the likelihood of a forest fire in an area. And lastly, another area that neural networks are being used for is optimization.

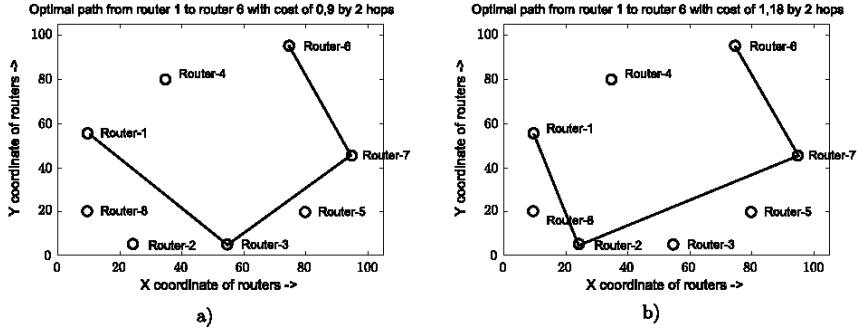


Fig. 6. (a) Optimal path from router 1 to router 6, including link capacity and traffic density matrices

Figure 4.1 Route Optimization Using ANN

In the figure above what is being shown is an artificial neural network being implemented to find the most optimal connection route between a series of Wi-Fi routers to make the most stable connection. These examples are only scratching the surface of the countless applications of artificial neural networks. As this technology continues to improve and become more complex as computing power improves, more applications will continue to be found for this technology.

Original Research:

Objective:

The objective of this research is to train an Artificial Neural Network to play Tic-Tac-Toe. Additionally, to gather more insight into this model's ability to find solutions to unsolved problems, the network will not be trained in such a way where the network is told what a “correct” or “incorrect” move is. For this research, we will assume that there are no objectively right or wrong moves, but we are aware of the rules of the game. The model will only ever be told when it is its turn to make a move, if the move it has decided to make is legal, and if it has won or lost.

Methodology:

In order for this network to improve, a method of training must be selected. In this case of trying to solve an “unsolved” problem, the method of Neuroevolution is very applicable. Neuroevolution is the combination of the Artificial Neural Network model, as well as the Evolutionary algorithm model. The Evolutionary Algorithm is a model which takes inspiration from the findings of Charels Darwin, who theorized that species had evolved over time through the process of natural selection where certain mutations would make a species more likely to survive and therefore pass down their genes.

In order to implement this model, multiple python scripts were developed, primarily using the numpy library. Additionally, Google Sheets was used to analyze the results. The code for this project can be found here.

Neural Network design:

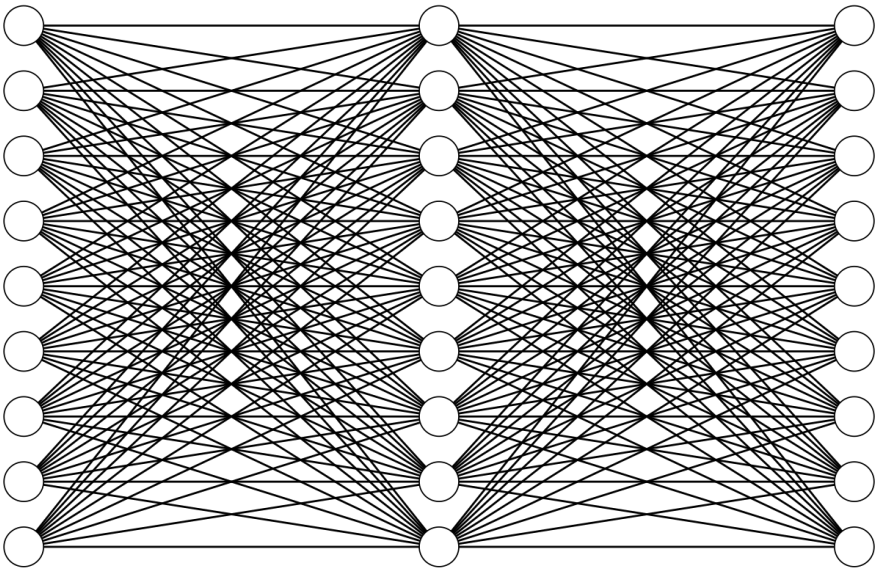


Figure 5.3.1 Designed Neural Network Diagram

Above what is shown is a diagram depicting the design of the neural network used for this research. For the input layer, each node represents an individual space on the board, making a total of 9 nodes. For an unoccupied space the input value is a 0, 1 for a marker of the network, and -1 for a marker of the opponent. Again in the hidden layer, there is another set of 9 nodes. The decision to add a hidden layer with 9 nodes was made to let the network potentially develop some complex form of abstraction, and potentially recognize more complex patterns. Additionally, for this layer the hyperbolic tangent function is used to limit the values to a range of -1 to 1. As for the last layer, again there are another 9 nodes, each representing a space on the board. The node containing the highest value in this layer is selected to place a marker in the corresponding space on the board that the node represents. If the selected move is already occupied, the next heist value will be selected.

Evolutionary Algorithm Design:

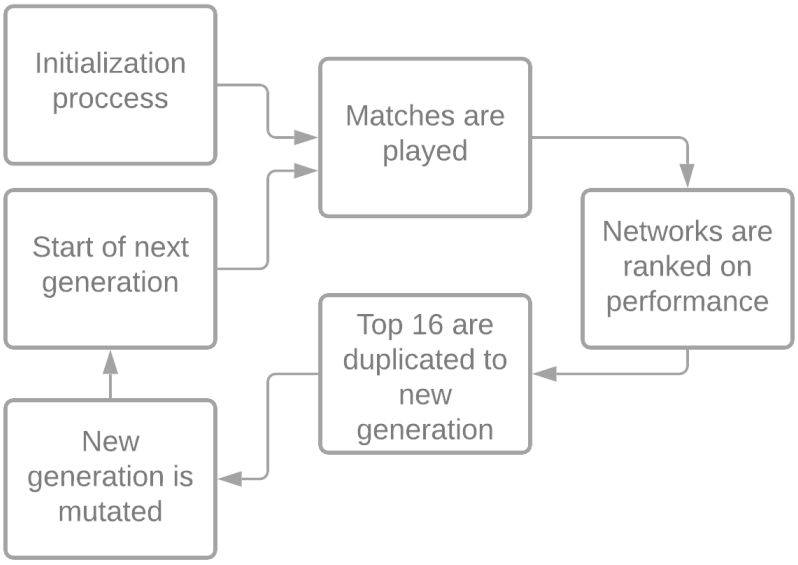


Figure 5.4.1 Evolutionary Algorithm Flow Chart

Above what is shown is a flow chart describing the process of the evolutionary algorithm. To start, a population of 32 networks are made. Each individual network is composed of 180 weights and biases, which are randomly generated values ranging from -.5 to .5. From there, matches are played, allowing the networks to be ranked on their performance. The top 16 networks are then selected to “reproduce”. Each of these 16 networks are then duplicated to create a new generation. From there, the new generation is brought through the mutation process. In the mutation process, each weight and bias has a 50% chance of being mutated. If a weight or bias is selected to be mutated, then the value will be changed by a random value between -.25 and .25. The process is then repeated with the newly mutated networks. In theory, this should allow positive mutations to the network to be passed down to the next generation, ultimately moving the networks closer to the optimal values for their weights and biases.

Method 1, Network vs. Network:

For the first test, the 32 networks were randomly paired against each other at the start of each generation to play one match. For determining who will start the match, the starting player is randomly selected. The winner of the match is then selected to be in the top 16 networks who will "reproduce". In the case of a tie, the winner is randomly selected. For this test, this process was repeated for 3500 generations. To summarize the results of this experiment, graphs of the win rate of each player and the lengths of the matches are shown below.

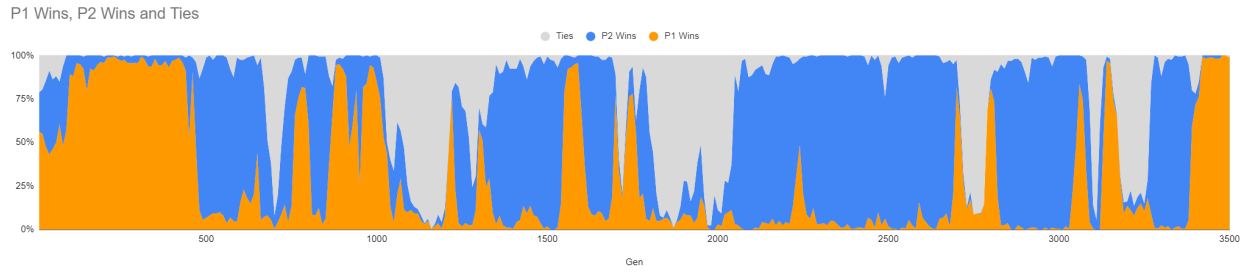


Figure 5.5.1 Win Rate Statistics For Test 1

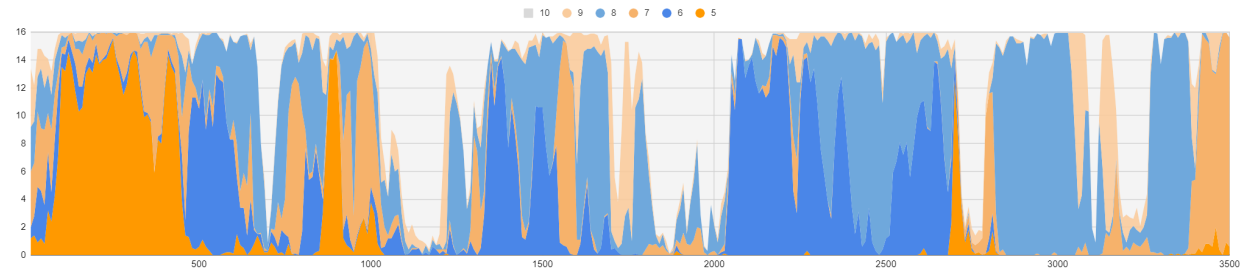


Figure 5.5.2 Match Length Statistics For Test 1

Although from first glance these graphs may look chaotic, there are many interesting patterns to observe. To start in the first 100 generations what we see is what we would expect if two players were making moves randomly. This makes sense as our networks are initialized with random values. What we see in around the 100th generation is a shift to a developed strategy, where the network has learned as player one how to efficiently place it's markers in a straight

line, giving player two less chances to block. What we see around the 500th generation is that the network as player 2 learns how to block player 1's "B-lining" strategy, and to efficiently complete a line itself. This is a somewhat consistent pattern where the network descoves new areas to form a quick 3 in a row, and then learns how to block this method as the other player. Additionally what we can observe as a general trend throughout the generations, is that as the generations increase, the match lengths tend to get longer. This would suggest that the network can no longer get away with the simple strategy of B-lining, and is developing more complex strategies

Although these results look promising, there are some concerns with this method. The first concern is that the results may not necessarily be showing objectively better play. It is very possible that the networks are potentially optimizing to lose as a certain player, in order to increase its chances of winning as the other player, but if a network deviates from that strategy, they are able to beat the deviator. The effect will not allow to networks to achieve perfect play. Additionally, with the instability shown in the results, it is unclear that even if the networks were to find perfect play, that they would be able to stabilize around those optimal numbers, or if they would instead devolve away from optimal strategy.

Method 2, Network vs. Random Moves:

In order to address the concerns from the first test, the second test conducted had each of the 32 networks play 5 matches against an opponent who played randomly selected moves. To rank their performance, a fitness score was given to each network. A win would add 2 points, a tie 1 point, and a loss 0 points. To summarize the results of this experiment, below is a graph showing the maximum, minimum, first quartile, third quartile, and average fitness scores for the generations over time.

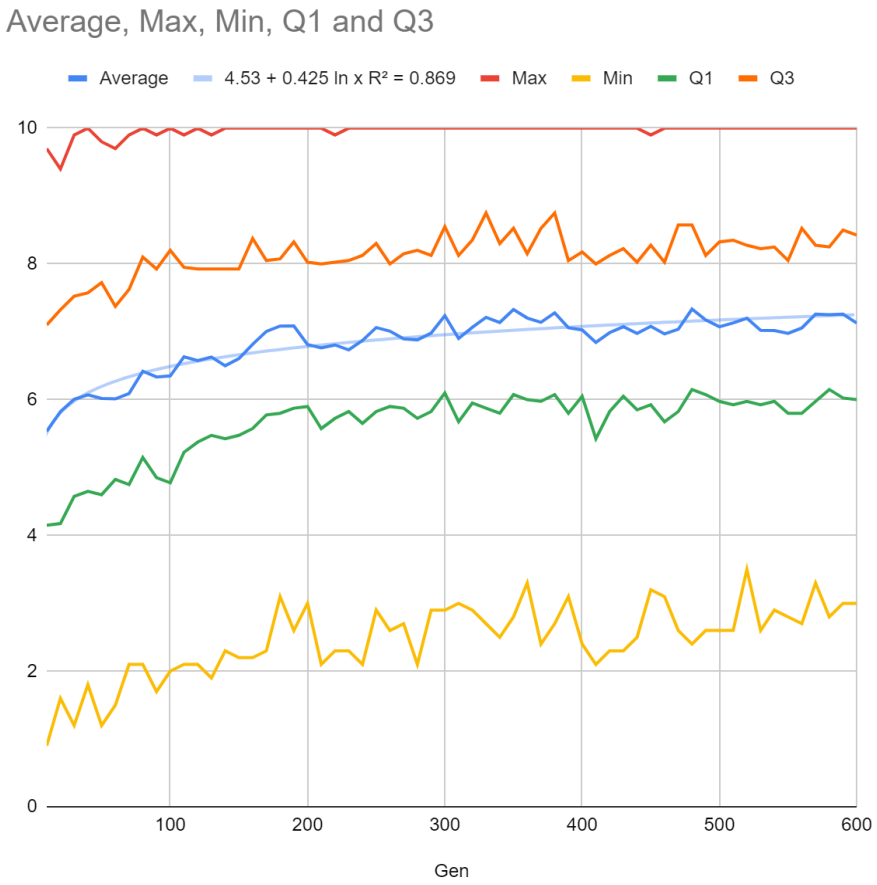


Figure 5.6.1 Fitness Score Distribution Over Time

What we see from this graph is that the network was able to significantly improve its fitness score. Starting from roughly the expected average fitness score of slightly over 5, the model was able to increase the average fitness score to over 7 points. Additionally, what is also shown is a level of stability that was not found in the last test. Not only were the networks able to improve their performance, but they were also able to keep a "memory" of some sort, where they did not lose what had worked for them previously. With this method, what was proven is that the evolutionary algorithm is able to continuously improve the network, and that the networks are not evolving away from an improved level of play.

Conclusion:

In conclusion, what has been shown from this research is that the method of Neuroevolution does have the potential to solve problems in which we do not have the answer to. By only knowing what a good outcome was, the network was able to modify itself in order to give it a higher chance of success. Although the level of intelligence achieved by this network was not the level required to play a perfect game, it did show the potential of being able to reach this level of play, given some modifications to the model.

With the amount of parameters in this model, there is plenty of room for optimization and improvement. The first area to start would be to test the effects of varying mutation rates, as well and mutation magnitudes. Another area could be the number of matches played per generation, whether that be network vs another network or a random moving opponent. By increasing the number of matches, the rankings of the networks would be more accurate, giving good mutations an even higher chance of survival. Another aspect of the evolutionary algorithm that can be modified is the population size in order to increase diversity. Additionally, the method of creating a new generation can be changed to vary the amount of children each network has to also increase diversity. As for the network itself, the area with the most room for modification is the typology of the network.

In summary, although this experiment did not result in a neural network which can perfectly play tic tac toe, it was a success in it's ability to show the great potential for the applications of this technology.

References:

"Artificial neural network" Retrieved from:

https://en.wikipedia.org/wiki/Artificial_neural_network

"Artificial neural networks: fundamentals, computing, design, and application", I.ABasheer,

Retrieved from: https://www.sciencedirect.com/science/article/abs/pii/S0167701200002013

"Artificial neural network architecture" Retrieved from:

https://www.researchgate.net/figure/Artificial-neural-network-architecture-ANN-i-h-1-h-2-h-n-o_fig1_321259051

"Various forms of non linear activation functions" Retrieved from:

https://www.researchgate.net/figure/Various-forms-of-non-linear-activation-functions-Figure-adopted-from-Caffe-Tutorial_fig3_315667264

"Perceptron" Retrieved from: https://en.wikipedia.org/wiki/Perceptron

"Neural network for optimization of routing in communication networks" N. Kojic, I. Reljin, B. Reljin, 2006, Retrieved from:

https://www.semanticscholar.org/paper/Neural-network-for-optimization-of-routing-in-Kojic-Reljin/fd565315acea095405db9411b328c5251acd0a1e

Apendix:

Code for project:

https://github.com/trentberrien/Tic-tac-toe-Neuroevolution

Sheets for initialization and data analytics:

https://docs.google.com/spreadsheets/d/1GajIU_botDzREoZ12Tf5CfP9LBN66Lmd8zOBjTZu448/edit?usp=sharing