CS 4300 Final Report

Ryan Kelly

May 13th, 2021

Contents

Abstract

Connect Four is a two-player board game for players ages 6 plus. As the game is designed for children, the rules are very simple. Despite the overall simplicity of the game, there is an extensive amount of possible combinations of game states that can be made while playing Connect Four. John Tromp, a dutch computer scientist, did research to record each possible combination of Connect Four pieces.[1] After completing his findings, he created a subset of his data set that contained all possible combinations of Connect Four games before the final move. Using that specific data set, a neural network model will be trained to attempt to accurately predict the outcome of any given Connect Four game. A simple baseline model was created using a linear activation function. After concluding that more layers would benefit the model, new activation functions were tested on the model. The final design used a sigmoid activation function for each layer. Then, the model was improved even further with model checkpointing. Feature reduction was not successful in further improving the model. The final model achieved an accuracy of 89.08%, which is an increase of 16.38% from the baseline accuracy of 72.7%. The attempt to train a neural network model to accurately predict the outcome of any given Connect Four game was successful.

1 Introduction

1.1 What is Connect Four?

Connect Four is a two-player connection board game.[2] Players face off against each other by taking turns dropping colored discs into a seven by six vertical game grid. The pieces always fall to the lowest available level on the grid. The objective is to be the first player to make a vertical, horizontal, or diagonal line of four of your respectively colored game pieces.

1.2 What is so interesting about it?

Connect Four was always one of my favorite board games growing up. As I have grown and learned about different card games, video games, and sports, I always appreciated the simplicity of Connect Four. But when looking for a data set for my semester long project, I was blown away by the size of this Connect Four data set. It had 67,557 instances of different Connect Four rounds, every possible combination before the winning move.[1]

1.3 How does this apply to Artificial Intelligence?

Artificial Intelligence is commonly used in video games to control characters or enemies that are not another player. The motivation for this project is to try and write learning algorithms to have the AI predict a winner through the use of a neural network.

2 Data Set

The "Connect-4 Opening Database" data set was downloaded from the UCI Machine Learning Repository. John Tromp spent 40,000 hours creating a database of every possible combination of Connect-4 games. This data set is a subset of that data set, and contains each unfinished and unforced position. Tromp created this subset of data for machine learning experiments.[1] Each row lists each of the respective positions for pieces on a Connect Four board, described by what piece is occupying it. A single class label as the final entry shows if the game was a win, a loss, or a draw. The input features are for the following fields:

1. A1
2. A2
3. A3
4. A4
5. A5
6. A6
7. B1
8. B2
9. B3
10. B4
11. B5
12. B6
13. C1
14. C2
15. C3
16. C4
17. C5
18. C6
19. D1
20. D2
21. D3
22. D4
23. D5
24. D6
25. E1
26. E2
27. E3
28. E4
29. E5
30. E6
31. F1
32. F2
33. F3
34. F4
35. F5
36. F6
37. G1
38. G2
39. G3
40. G4
41. G5
42. G6

2.1 Visualization of the distribution of each input feature

Distribution of each blank, player one, or player two input value across all 61,108 games.

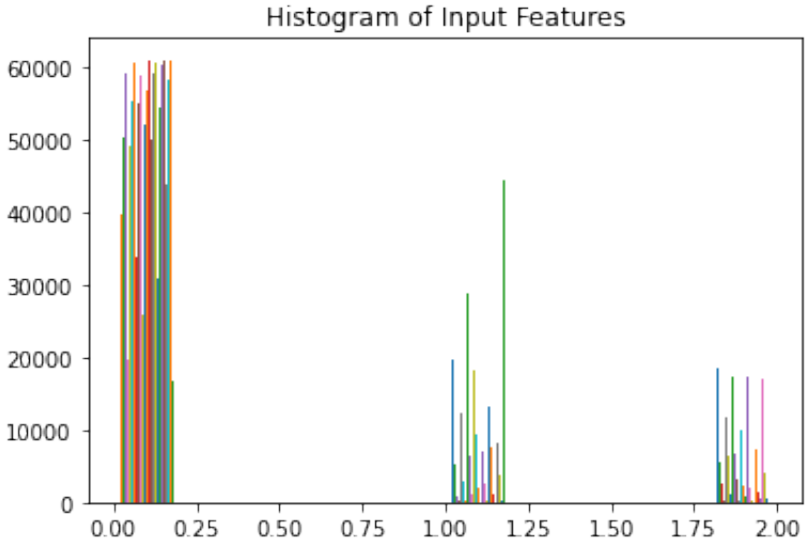


Figure 1: Distribution of Connect 4 Input Features

In the figure, the blank spaces (labeled 0.00) make up the majority of most games. The player one pieces (labeled 1.00) appears to be distributed unevenly, but still has more overall input than the player two pieces (labeled 2.00). The difference in input features could explain the distribution of output features in section 2.3.

2.2 Input Feature Statistics

Input Feature	Mean	Standard Deviation
A1		%
A2		%
A3		%
A4		%
A5		%
A6		%
B1		%
B2		%
B3		%
B4		%
B5		%
B6		%
C1		%
C2		%
C3		%
C4		%
C5		%
C6		%

Input Feature	Mean	Standard Deviation
D1		%
D2		%
D3		%
D4		%
D5		%
D6		%
E1		%
E2		%
E3		%
E4		%
E5		%
E6		%
F1		%
F2		%
F3		%
F4		%
F5		%
F6		%
G1		%
G2		%
G3		%
G4		%
G5		%
G6		%

2.3 Distribution of Output Features

Player one won 44,473 of the 61,108 games, or 72.7 percent.[1] John's study revealed that, if the first player makes the right moves, they can always win, which would explain this imbalance.

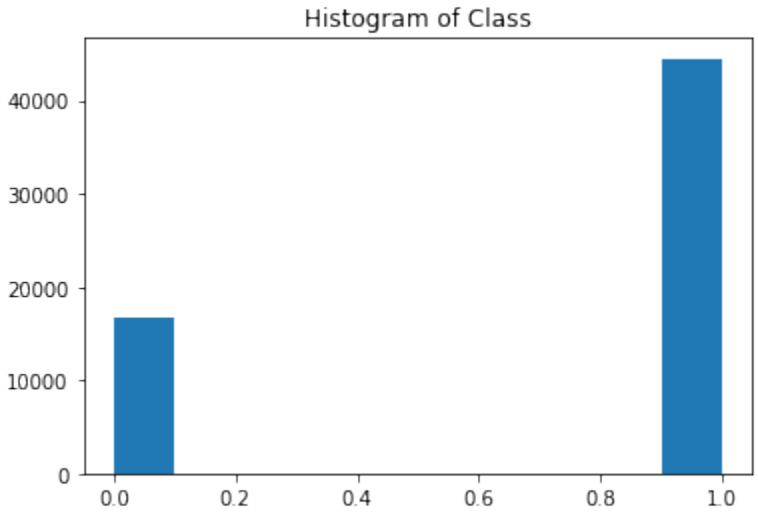


Figure 2: Output Features Distribution Histogram.

3 Data Processing

3.1 Data Normalization

In terms of input and output, this data set is very simple. Input can either be a blank space B, a player one piece O, or a player two piece X. Data normalization is not necessary for this data set, as all of these values can just be converted to a number for the neural network. For input, B was changed to 0, O was changed to 1, and X was changed to 2. Output was divided into win or lose, with win being converted to 0 and lose being converted to 1.

4 Modeling

A tensorflow feed forward neural network architecture was used to create this model. The data was shuffled, meaning that results will vary. These models were compiled on April 14th, 2021.

4.1 Baseline Accuracy

The baseline accuracy of the model can be found by dividing the number of wins by the total number of data entries. With 44,473 wins and 61,108 total data entries, this model has a baseline accuracy of 72.7778%.

4.2 Selecting a Baseline Neural Network Model

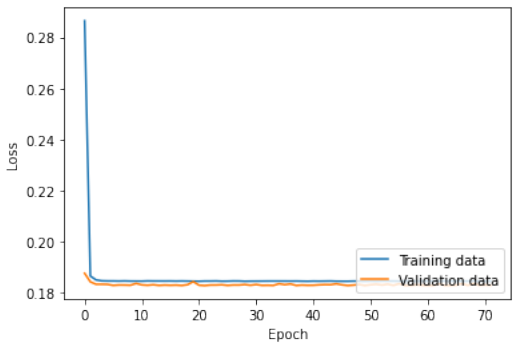
The first model was a baseline to act as our control. It used a logistic regression model with one layer for input, one layer for output, and a linear activation function. The loss and accuracy were observed while adding hidden layers, to see which ultimately preforms the best on the training set.

4.2.1 Baseline Model Performance

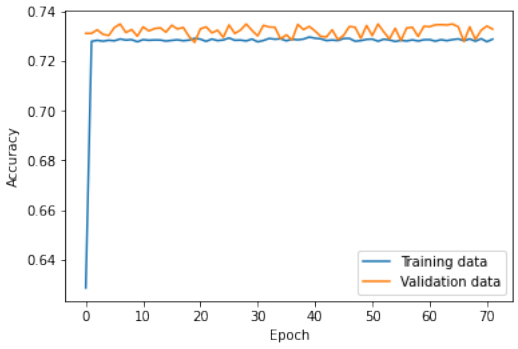
Hidden	Loss	Accuracy
0 Layer	0.18395	72.97%
1 Layer	0.18532	72.75%
2 Layer	0.18288	73.29%

By referring to the table, we can see the model with two hidden layer outperforms both other models in terms of accuracy. Going forward, two hidden layers will be applied to other architecture.

4.3 Learning Curve of Baseline Model



(a) Loss/Epoch of 0 Layer Model



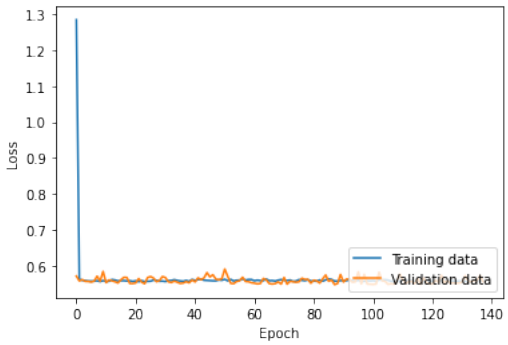
(b) Accuracy/Epoch of 0 Layer Model

4.4 Activation Function

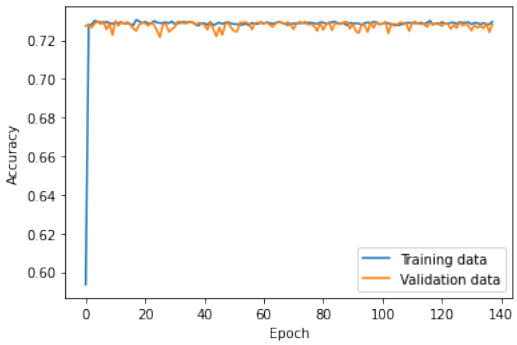
The linear activation function is generally used for instances of logistic regression. As our model is an instance of binary classification, the sigmoid activation function should perform better [3], as it is designed for outputs of zero and one. To compare performance, tests will be run with both linear and sigmoid activation functions. Both activation functions will have a test where they are only applied in the last layer, and another where the activation function is applied to every layer. To help increase the performance of the model, the first layer was given a density of 4, the second layer was given a density of 2, and the final output layer only needs a density of 1.

Activation Function	Accuracy	Loss
Linear (All)	72.82%	0.54698
Linear (Last)	76.9%	0.4716
Sigmoid (All)	77.89%	0.47962
Sigmoid (Last)	77.52%	0.47183

4.5 Learning Curve of Linear Activation Model (All Neurons)

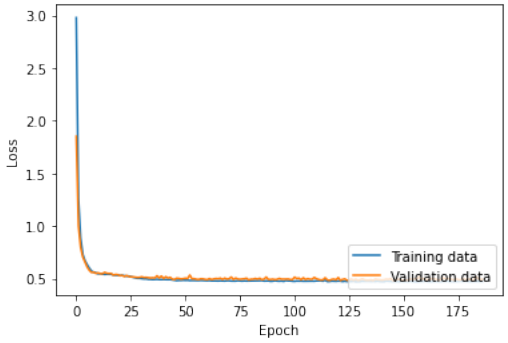


(a) Loss/Epoch of Linear(All) Model

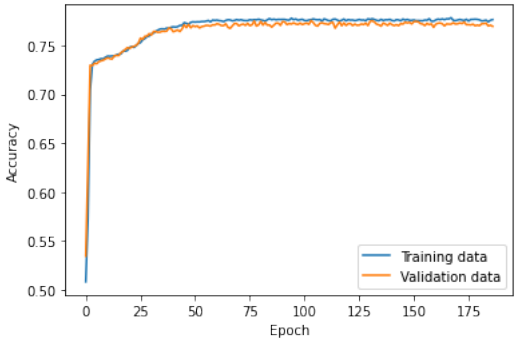


(b) Accuracy/Epoch of Linear(All) Model

4.6 Learning Curve of Linear Activation Model (Last Neuron)

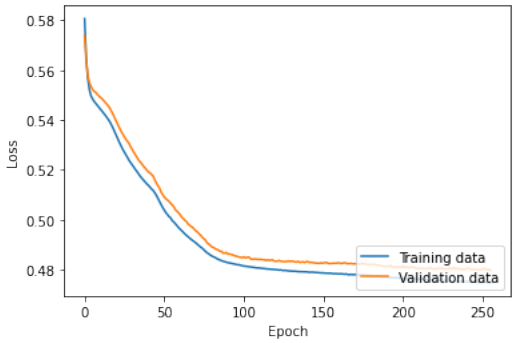


(a) Loss/Epoch of Linear(Last) Model

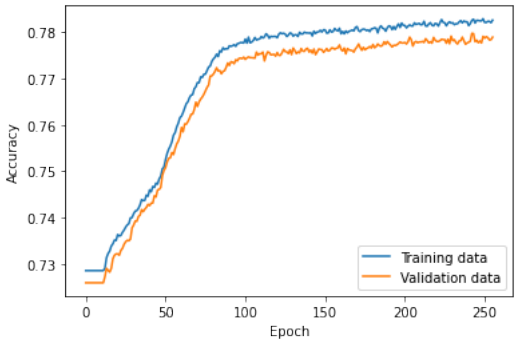


(b) Accuracy/Epoch of Linear(Last) Model

4.7 Learning Curve of Sigmoid Activation Model (All Neurons)

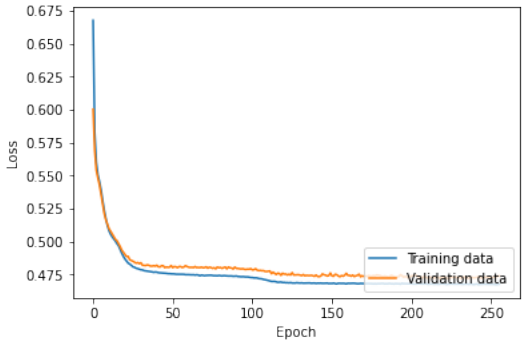


(a) Loss/Epoch of Sigmoid(All) Model

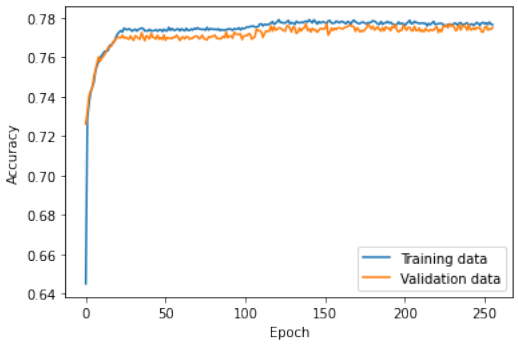


(b) Accuracy/Epoch of Sigmoid(All) Model

4.8 Learning Curve of Sigmoid Activation Model (Last Neuron)



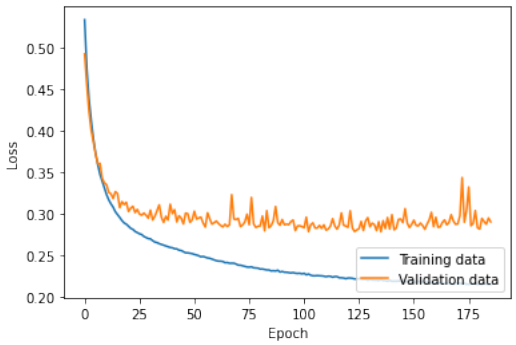
(a) Loss/Epoch of Sigmoid(Last) Model



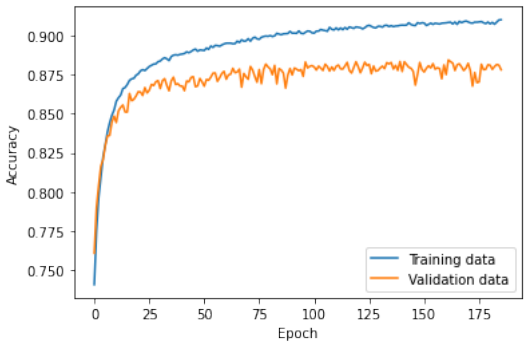
(b) Accuracy/Epoch of Sigmoid(Last) Model

4.9 Learning Curve of Overfit Model

The number of neurons in each layer was increased by a factor of 10 to overfit the model.



(a) Loss/Epoch of Overfit Model



(b) Accuracy/Epoch of Overfit Model

	Accuracy	Loss
Overfitting	87.82%	0.27855

5 Model Evaluation

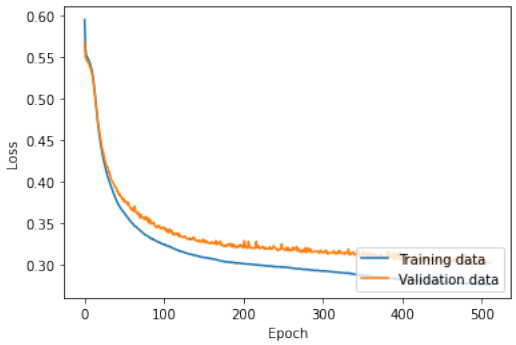
The three classification model metrics used to evaluate the neural network models were precision, recall, and f1 score.

1. Precision: The quantity of positive identifications that was correct.
2. Recall: The quantity of positives correctly identified.
3. F1-Score: Evaluation metric determined using precision and recall; the highest possible value is 1.0, the lowest possible value is 0.0.

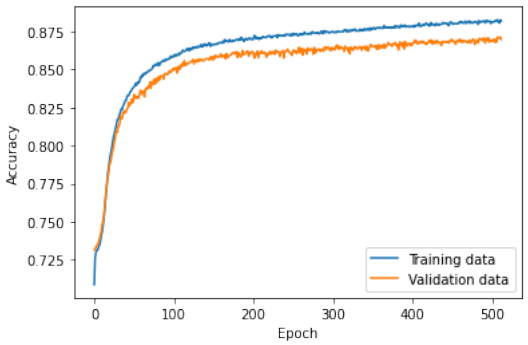
Model	Precision	Recall	F1-Score
Baseline	74.12%	97.66%	0.84
Linear (All)	73.41%	98.28%	0.84
Linear (Last)	78.90%	93.19%	0.85
Sigmoid (All)	79.72%	93.27%	0.86
Sigmoid (Last)	79.21%	93.66%	0.86
Overfitting	91.81%	91.38%	0.92

5.1 Increasing the Number of Layers and Neurons

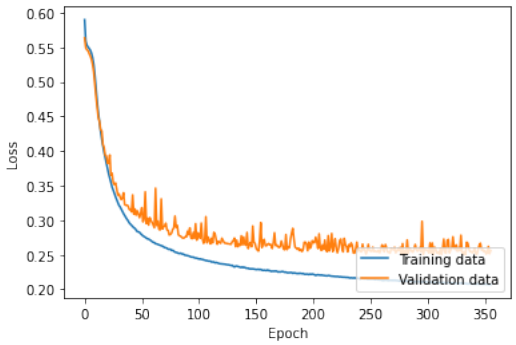
After evaluating the previous models, we can see the model with all sigmoid activation functions out performed the others slightly in terms of accuracy, precision, and F1-score. As the baseline accuracy was 72.7%, an increase to 79.21% is not significant. To further increase the accuracy of the model, the all sigmoid model will be tested with 4, 5, and 6 hidden layers. The 4th layer will have 16 neurons, and the number of neurons will double as the number of layers increases.



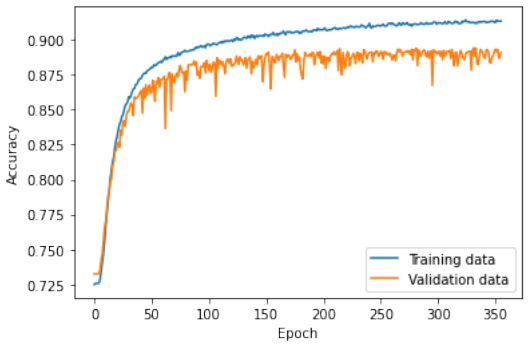
(a) 4 Layers Loss/Epoch



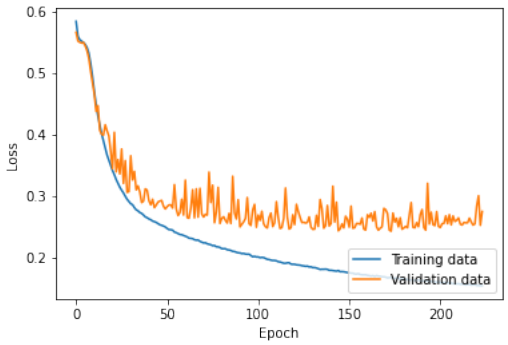
(b) 4 Layers Accuracy/Epoch



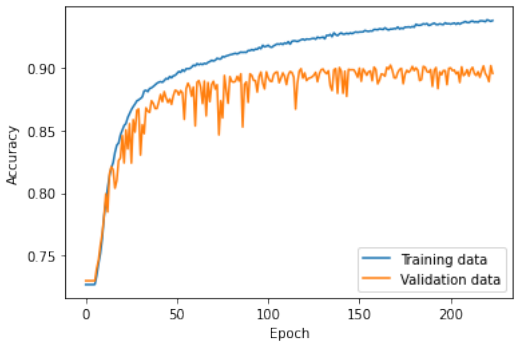
(a) 5 Layers Loss/Epoch



(b) 5 Layers Accuracy/Epoch



(a) 6 Layers Loss/Epoch



(b) 6 Layers Accuracy/Epoch

Hidden Layers	Loss	Accuracy	Precision	Recall	F1-Score
4	0.30045	86.99%	89.56%	93.06%	0.91
5	0.25012	89.08%	93.41%	91.55%	0.92
6	0.24217	89.57%	90.62%	95.61%	0.93

By increasing the number of layers an neurons, the loss was reduced to almost half of what is was from the best model (Sigmoid All) during the initial Model Evaluation. The 6 Hidden Layer model reached an accuracy of 89.57%, an increase of 16.8% from the baseline accuracy. The graph of the 6 Layer model shows signs of overfitting, as the training accuracy is much higher than the validation accuracy. The overall increase in accuracy across all models is statistically significant, and shows how the model is starting to understand the game of Connect 4 better.

6 Model Checkpointing

Model Checkpointing was used after Baseline Models to save time on data processing. By implementing Checkpointing early, models that could not improve after fifty epochs are stopped early. It was not as important with models like the Overfit Model, which improved its loss so much that Model Checkpointing never had to terminate early. By looking at the epoch of the graphs in Section 5.1, you can see as the layers increased, the model needed less time to reach its maximum accuracy or minimum loss.

7 Feature Reduction

7.1 Feature Importance

All 42 inputs were divided into groups of three, run through a training model, and ranked their accuracy against each other. The range is less than 0.0351. Testing larger input feature groups could be more useful.

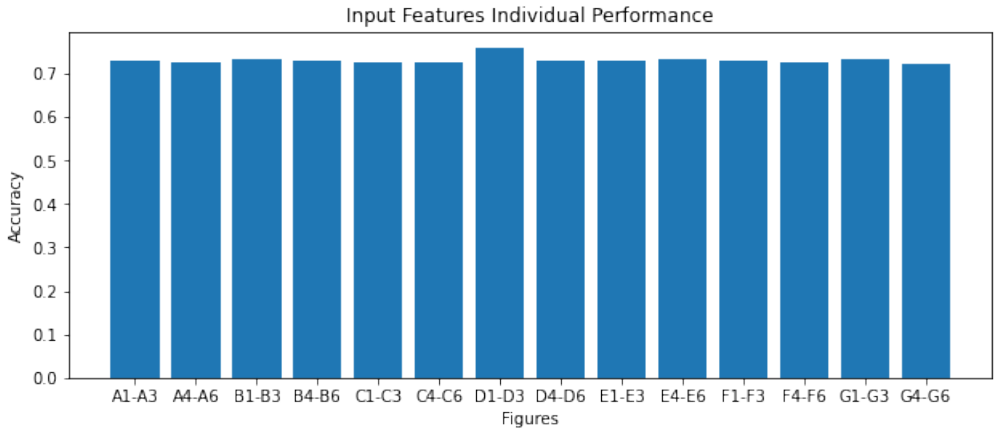


Figure 12: Accuracy of predicted output with three input features.

Of the 14 input feature groups of size three, 13 had an accuracy of around 72% to 73%. One input feature, D1-D3, achieved an accuracy of 75.8%. As any slot on the Connect-4 Board can be filled and be part of a win state, it makes sense that the importance of features is relatively balanced. It is important to note that column D is the center of all 7 rows. This could explain why D1-D3 have the highest accuracy, as the center point could be involved in more winning states. D1-D3 are also located on the three bottom levels of the board, and the first pieces always start on the bottom of the board.

7.2 Performance of Model After Feature Removal

The removal of input features only decreased the accuracy of our model. The 4 hidden layer model has an accuracy of 86.99%. The lowest accuracy model after feature removal was the model that removed C1-C3, with an accuracy of 82.44%. This is a decrease of 4.55%. All model with removed features were within a range of 3.61%, with the maximum being A4-A6 with an accuracy of 86.05% and the previously mentioned minimum of C1-C3.

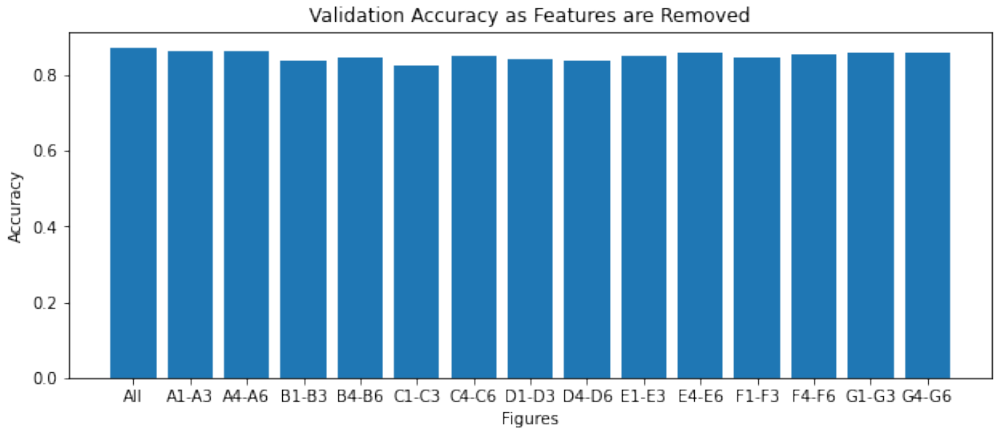
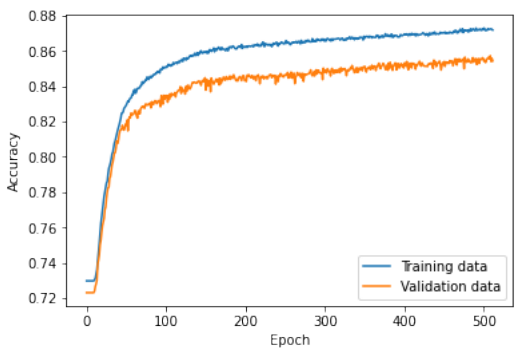


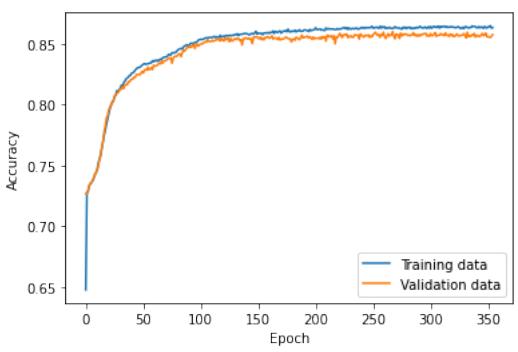
Figure 13: Accuracy of model with three features removed is very balanced.

The model does not seem to benefit from the simple Recursive Feature Elimination technique I applied. The individual input features performance histogram does not correlate with the validation accuracy as features are removed histogram. As D1-D3 had the highest accuracy in the individual performance, we would predict the model to take the largest drop in accuracy when the feature was removed. However, D1-D3 has three other models with worst accuracy in the removed features histogram.

As the game of Connect-4 is very dynamic, the removal of any given input feature cannot benefit our model. By removing features, the AI is blinded from a portion of the game board. For example, the removal of A1-A3 makes it very difficult to tell who won the game of Connect-4 when three of the winning pieces are inside of A1-A3. This does not make up a lot of the data, and explains why we only see a drop of about 4%. Overall, Recursive Feature Elimination is not effective for the accuracy of the Connect-4 neural network model.



(a) F4-F6 Feature Removal



(b) G4-G6 Feature Removal

8 Conclusion

The motivation for this project and report was to test and record how effective a neural network model would be at predicting the outcome of a Connect 4 game, using a binary classification model. Binary classification was chosen as it is designed for placing data into one of two categories, in our case win or lose. During training, several different activation functions and different sized models were tested to compare performance. Ultimately, the sigmoid activation function on each layer out performed the linear and relu activation functions during the model evaluation portion. By increasing the number of layers and neurons, we saw the model began to overfit itself. Model checkpointing and feature reduction were implemented to try and make the model more general. Feature reduction was unsuccessful in improving the model, but the model did see a lot of improvement overall. The baseline accuracy of 72.7% was increased to 89.08% without overfitting the model. My project was successful, as a neural network model that can accurately predict the winner of a Connect 4 game was created.

References

[1] John Tromp. John's Connect Four Playground. 1995.

[2] Milton Bradley Co. Connect four (rules). (4430-X1), 1990.

[3] Jason Brownlee. How to Choose an Activation Function for Deep Learning. 2021.