Predicting Block Breaker Game Wins

Zachary Christensen
University of Missouri-St. Louis

April 12, 2022

Contents

Abstract

A block breaker game in the style of the classic game Breakout was simulated using an automated player. An example of the game in action can be found here. Because the game is simulated to represent level difficulty to a human player, the auto player is tuned to a given accuracy, generally between 30% and 40%. Through the data generated from thousands of complete simulations, I hope to understand the conditions for a player to win or lose.

1 Introduction

What are the rules for the game?

The game environment is surrounded by a top, bottom, left, and right wall and includes a paddle, a ball, and blocks . The player controls the horizontal axis of the paddle. Not all blocks are able to be destroyed. The game starts when the player launches the ball into motion. The ball reflects from the paddle, the blocks, and the walls. When a block reflects the ball, it takes damage and may be destroyed. Once all the blocks able to be destroyed are destroyed, the game is considered won. A player loses the game if they fail to reflect the ball with the paddle.

Why is this problem interesting?

Video games can represent very complex environments. Even a few elements interacting through the physics engine can create situations that are extremely difficult to predict and control. In games that depend on dynamic elements, such as the physics engine, there is no formula to predict exactly what will happen. An example would be striking a pool ball and trying to predict all the exact reflections and end positions for every other ball. If the game can be broken down into some basic measurements and basic goals, then tests can be run to produce data. That data is something tangible that can be used to start understanding the behaviour of the game, so it can be predicted and controlled.

Motivation

Characteristics of a good game are drama and depth. If a game has drama, then the player will have a fun, memorable experience. If the game has depth, the player will be able to get better at the game in numerous ways. These characteristics are only good when implemented in a balanced way that gives the player the most fun, memorable experience. For example, drama should engage the player but not be punishing, and depth should exist at different skill levels so it is not only engaging to a select group. To achieve this through manual playtesting is tedious if not ineffective. The motivation of this project is to learn how use statistics from gameplays to understand the elements that control the likeliness a player is be able to win a level.

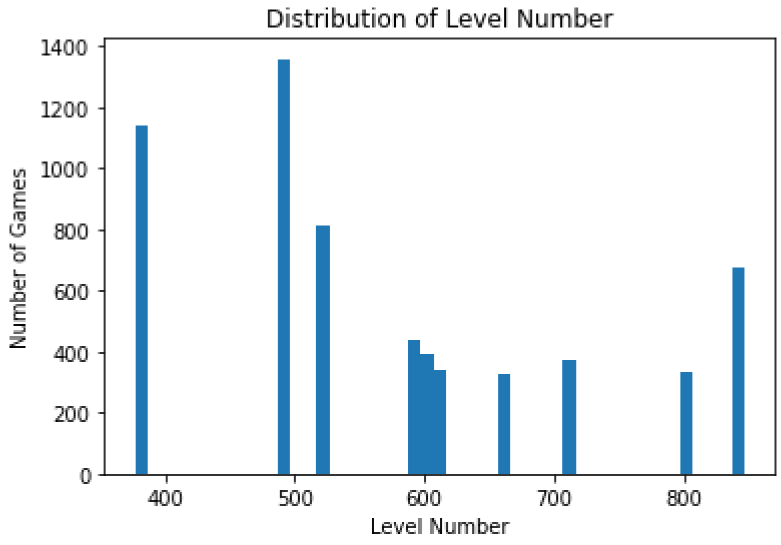
2 Dataset

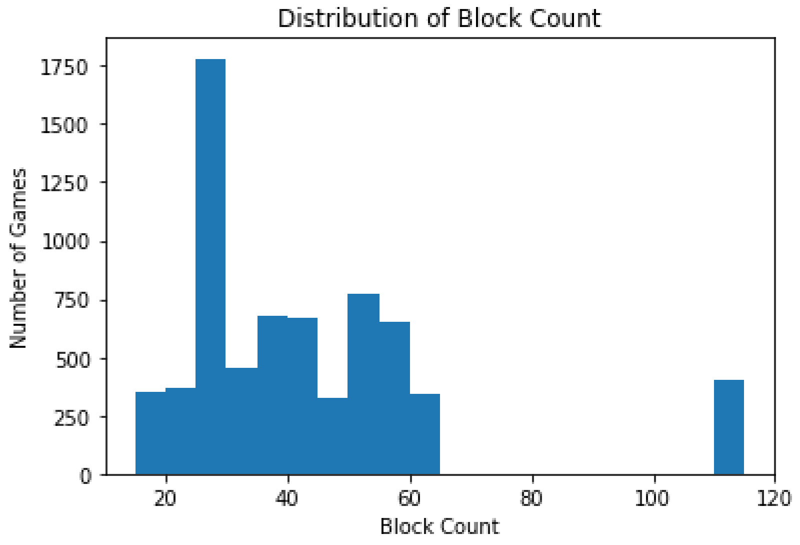
The ”Arcade Game Stats” dataset was found amonst the Kaggle datasets and can be found here [1]. The data was obtained from thousands of gameplay simulations using a robot for player input. The data contains 3266 games where the robot won and 3548 games where the robot lost. To account for a human player, the robot was assigned an accuracy that sets how well the robot performs. The output of the data is 0 for a loss and 1 for a win. The input features are as follows:

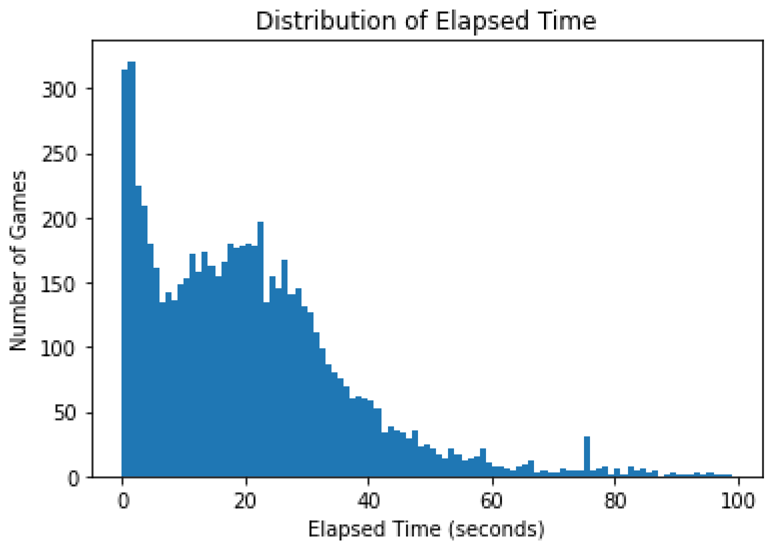
1. Level Number
2. Number of Blocks
3. Elapsed Time
4. Score
5. Accuracy

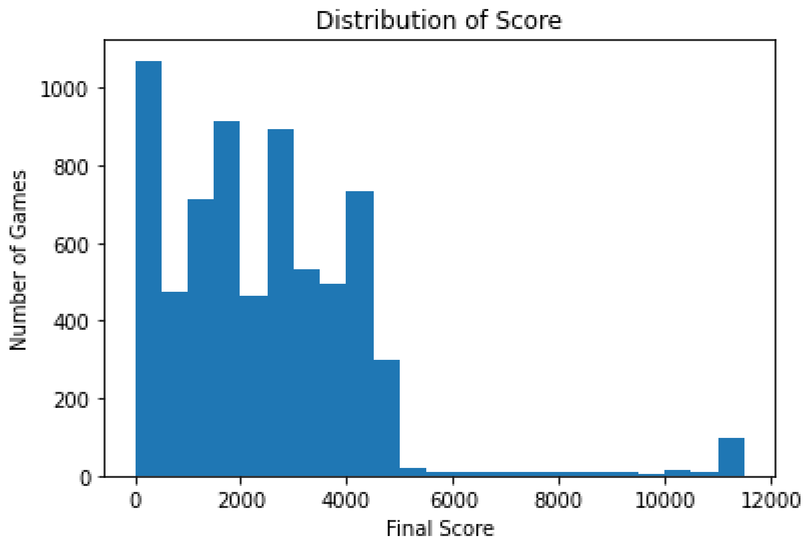
2.1 Data Visualization

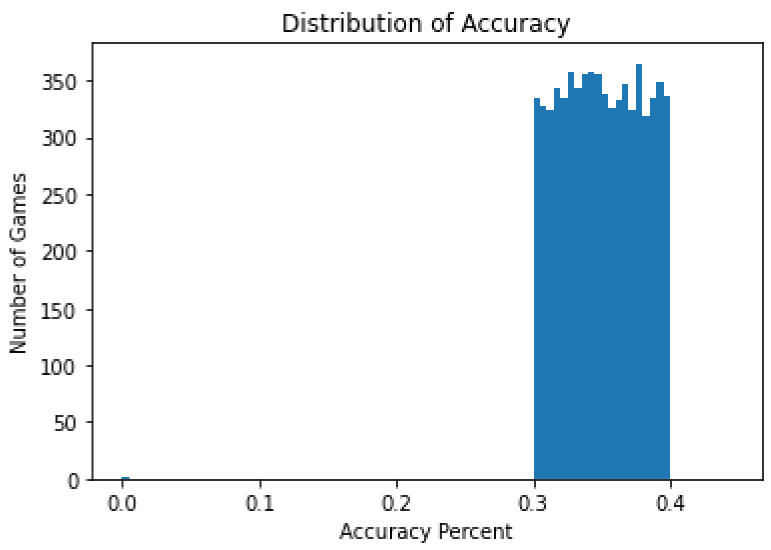
Level number appears to be grouped into several clusters between about 400 and 800. Number of blocks looks like it is normally distributed. Elapsed time is skewed to the right. Score looks like it is normally distributed. Accuracy is evenly distributed within its range.

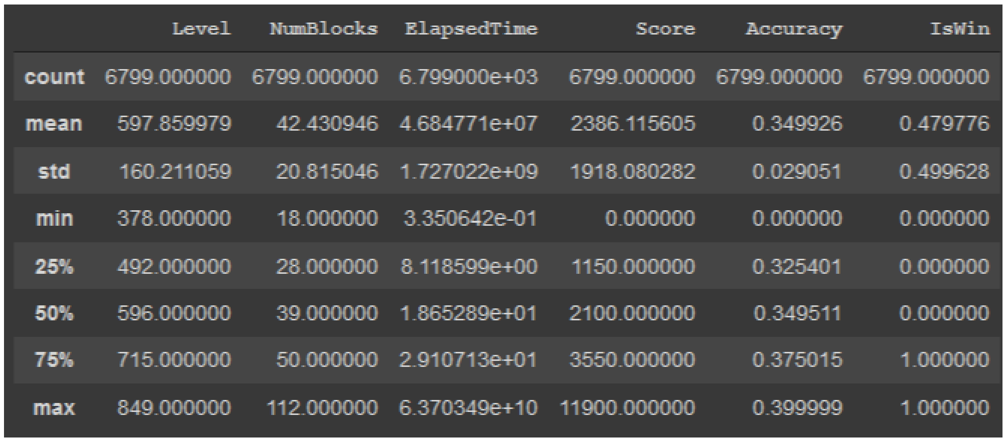



Distribution of Block Count
Number of Games
Block Count
0
250
500
750
1000
1250
1500
1750
20
40
60
80
100
120


Distribution of Elapsed Time
Number of Games
Elapsed Time (seconds)
0
50
100
150
200
250
300
0
20
40
60
80
100


Distribution of Score
Number of Games
Final Score
0
200
400
600
800
1000
0
2000
4000
6000
8000
10000
12000

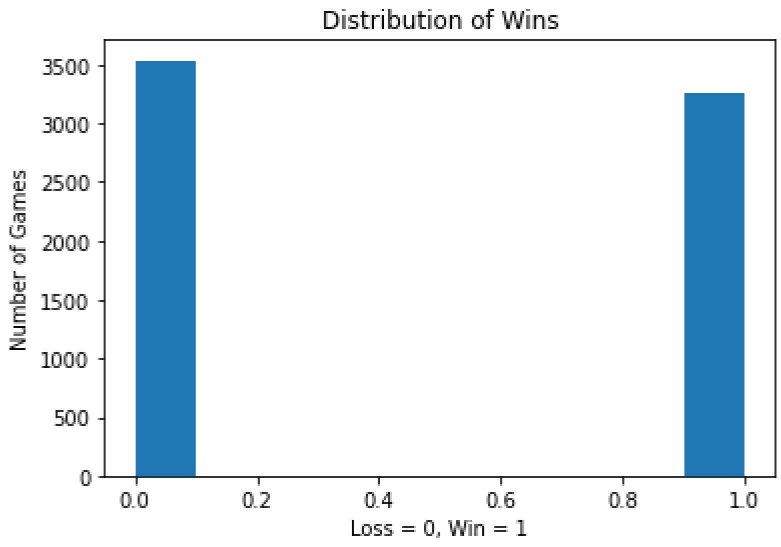




	Level	NumBlocks	ElapsedTime	Score	Accuracy	IsWin
count	6799.000000	6799.000000	6.799000e+03	6799.000000	6799.000000	6799.000000
mean	597.859979	42.430946	4.684771e+07	2386.115605	0.349926	0.479776
std	160.211059	20.815046	1.727022e+09	1918.080282	0.029051	0.499628
min	378.000000	18.000000	3.350642e-01	0.000000	0.000000	0.000000
25%	492.000000	28.000000	8.118599e+00	1150.000000	0.325401	0.000000
50%	596.000000	39.000000	1.865289e+01	2100.000000	0.349511	0.000000
75%	715.000000	50.000000	2.910713e+01	3550.000000	0.375015	1.000000
max	849.000000	112.000000	6.370349e+10	11900.000000	0.399999	1.000000

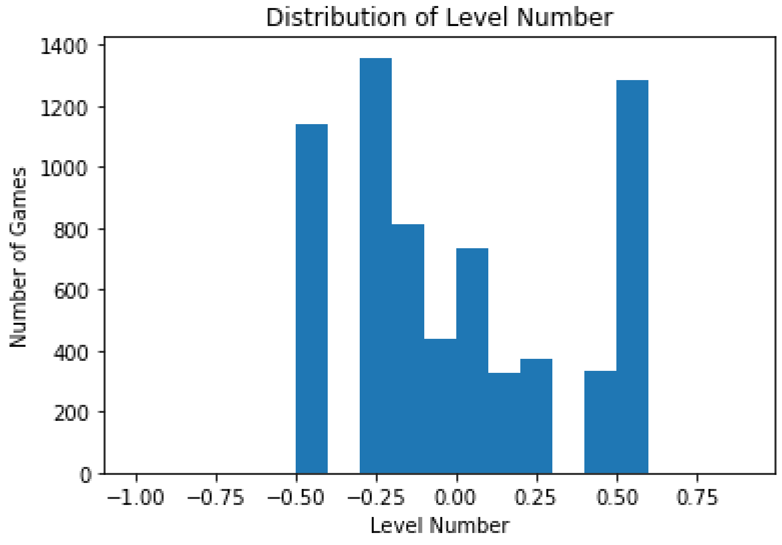
2.2 Distribution of Output

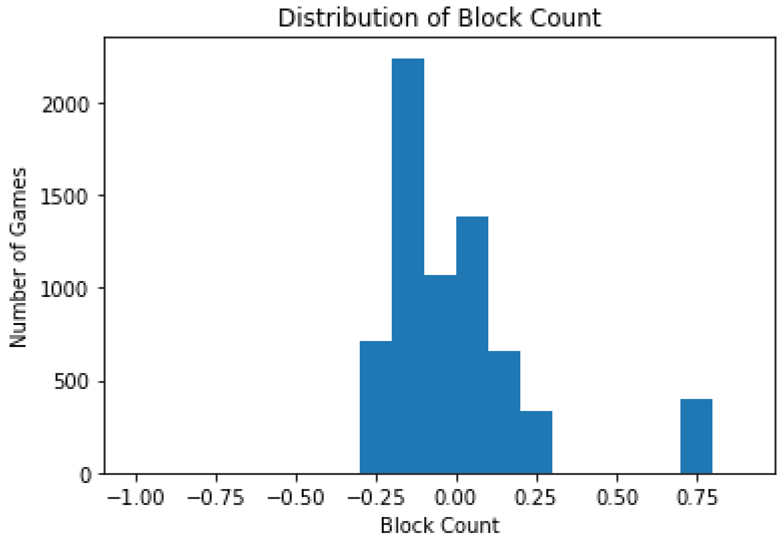
The number of wins and losses are almost perfectly balanced. There is an appropriate sample size for each sample.

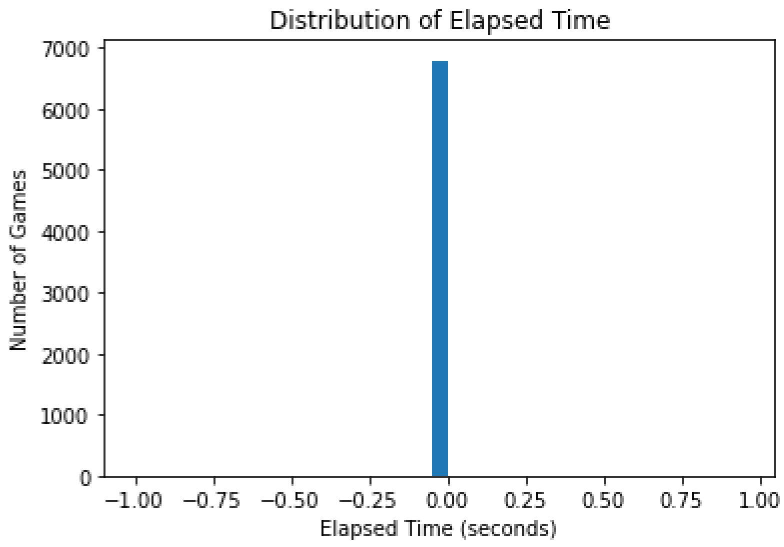


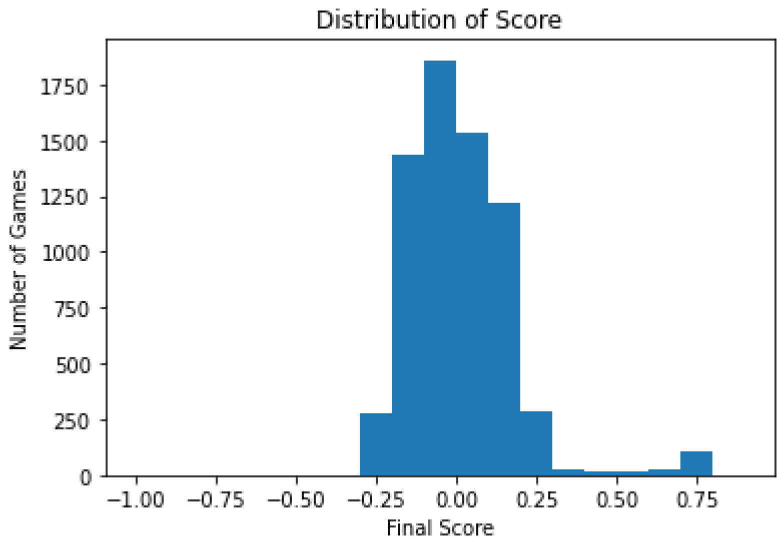
3 Data Normalization

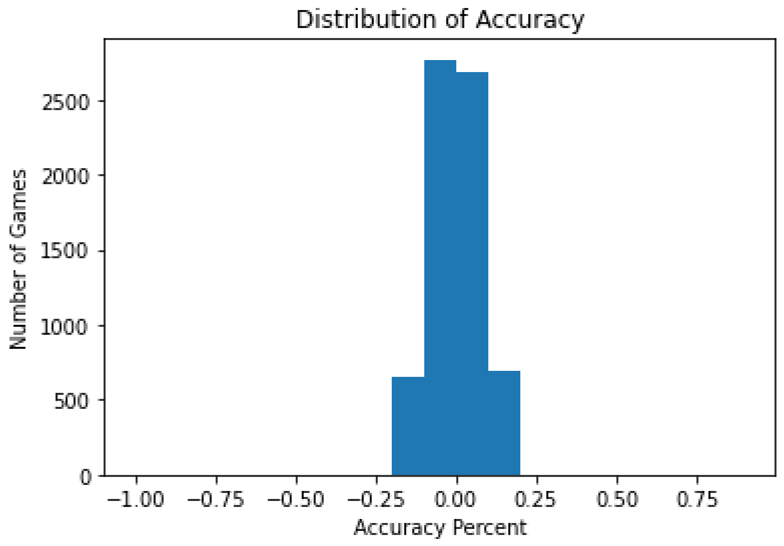
The normalization technique I chose was mean normalization. The only column I removed from the original data was the date column.






Distribution of Elapsed Time
Number of Games
7000
6000
5000
4000
3000
2000
1000
0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
Elapsed Time (seconds)


Distribution of Score
Number of Games
1750
1500
1250
1000
750
500
250
0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
Final Score


Distribution of Accuracy
Number of Games
2500
2000
1500
1000
500
0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
Accuracy Percent

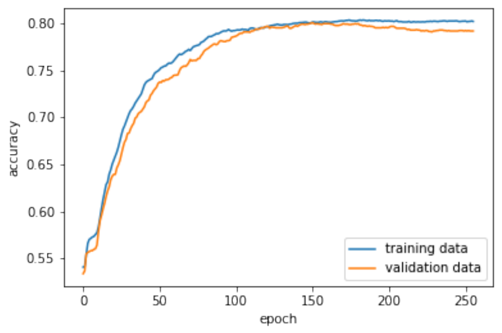
4 Data Modelling

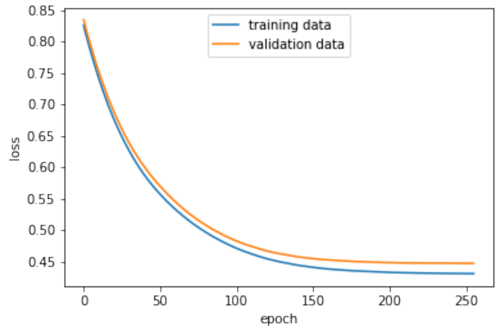
Following the instructions for project phase 2, I built a single layer model, several models of different neuron and layer count, an overfitting model, and several models where the last layer was changed to relu.

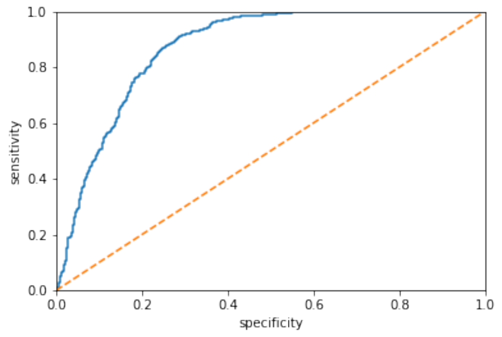
4.1 Single Layer Model

For the single layer model, I was only able to use a single neuron because otherwise there was an error. I ran the model for 256 epochs.

Accuracy	Loss	Precision	Recall	F1 Score	AUC
0.792074	0.446911	0.775571	0.796939	0.78611	0.869685







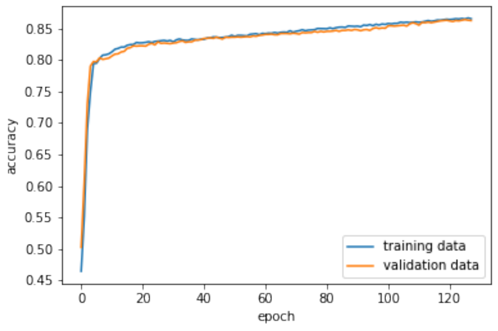
4.2 Multi Layer Models

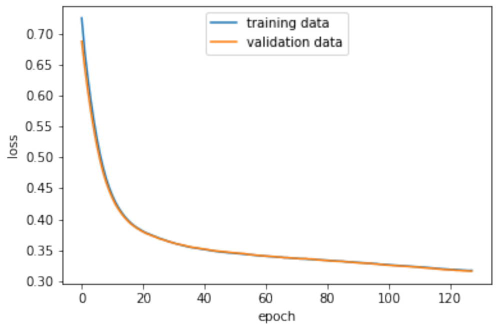
I correlated 2 and 3 layers with 32 and 128 neurons. This produced 4 different models. I ran the models for 128 epochs.

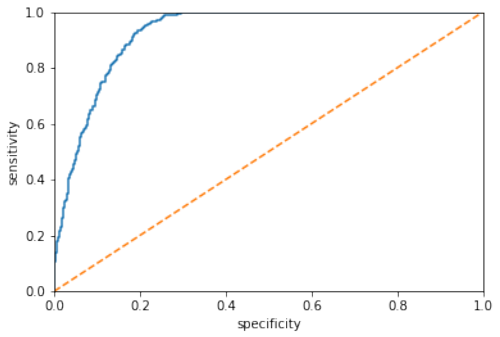
4.2.1 Model 1

This model uses 2 layers and 32 neurons.

Accuracy	Loss	Precision	Recall	F1 Score	AUC
0.862524	0.316271	0.804444	0.936853	0.865615	0.925784



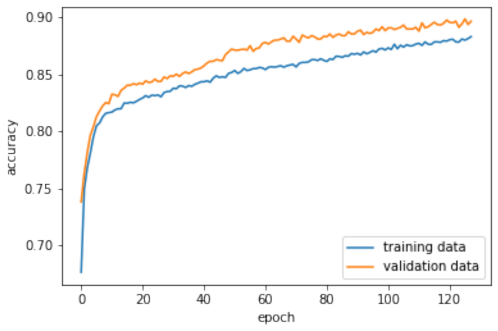


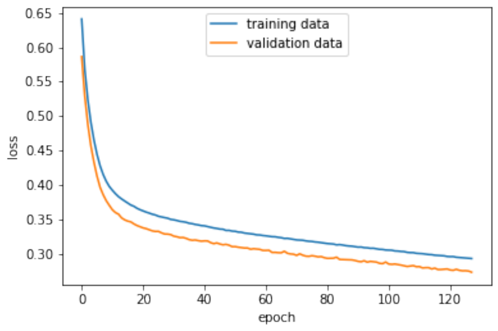


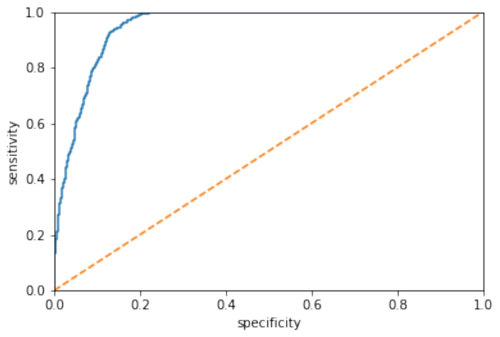
4.2.2 Model 2

This model uses 2 layers and 128 neurons.

Accuracy	Loss	Precision	Recall	F1 Score	AUC
0.896771	0.272822	0.848673	0.95996	0.900892	0.947816



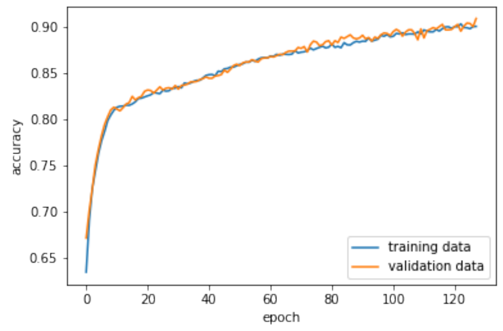


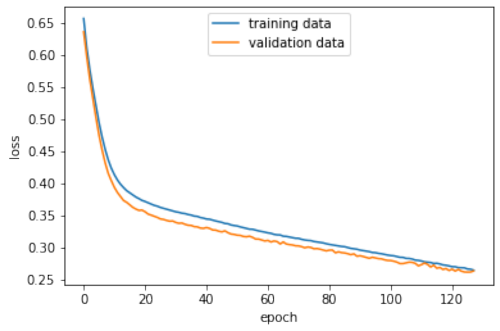


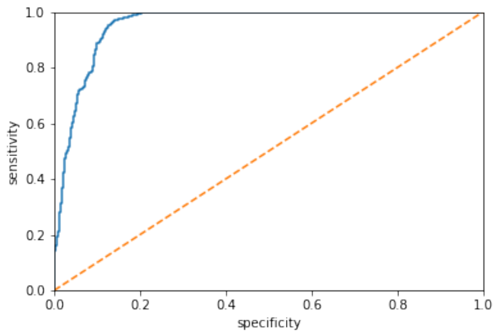
4.2.3 Model 3

This model uses 3 layers and 32 neurons. For the second layer, I used 25% of 32, or 8 neurons. This is also true for the next model using 3 layers.

Accuracy	Loss	Precision	Recall	F1 Score	AUC
0.908513	0.262939	0.855335	0.972251	0.910053	0.953967



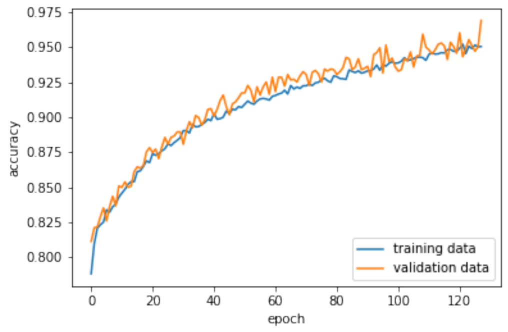


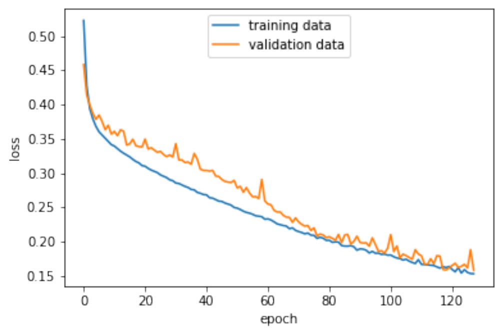


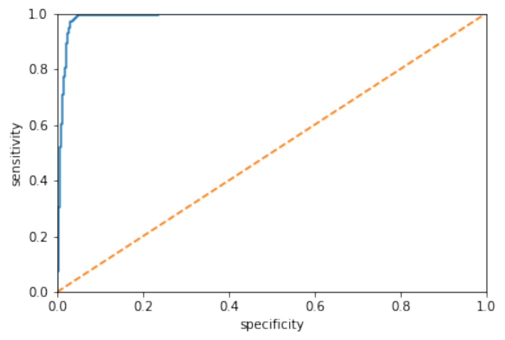
4.2.4 Model 4

This model uses 3 layers and 128 neurons. For the second layer, I used 32 neurons.

Accuracy	Loss	Precision	Recall	F1 Score	AUC
0.969178	0.158653	0.94702	0.993056	0.969492	0.987966



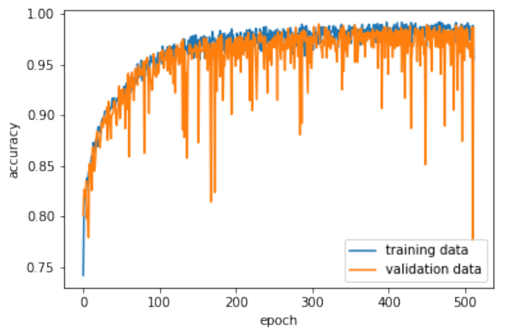


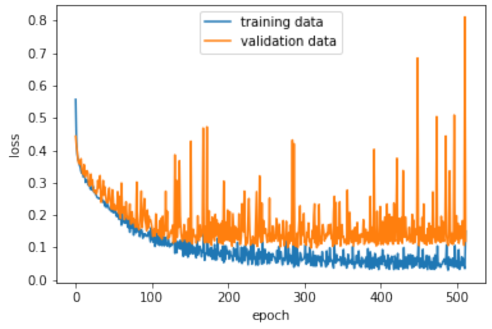


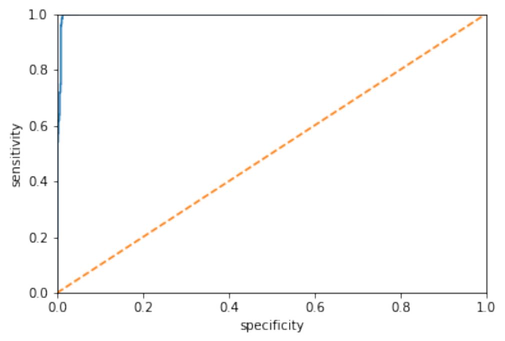
4.3 Overfit Model

The models I have tested on this data all seem to perform well. In order to create an overfitting model, I used a 5 layer model. The model used 512 epochs, and each descending layer contained 25% of the neurons of the previous. It was difficult to find a model that showed an overfit because most models seem to be able to generalize the data. In this model, the validation loss curve was above the training loss curve.

Accuracy	Loss	Precision	Recall	F1 Score	AUC
0.986301	0.065453	0.97549	0.996994	0.986125	0.995604







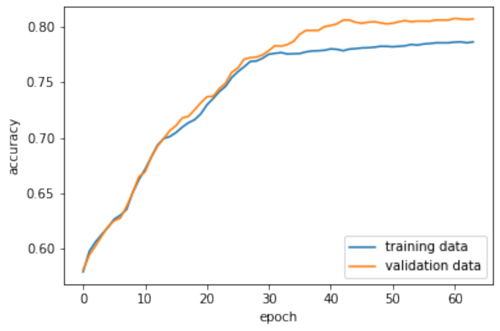
4.4 Last Layer Changed to Relu

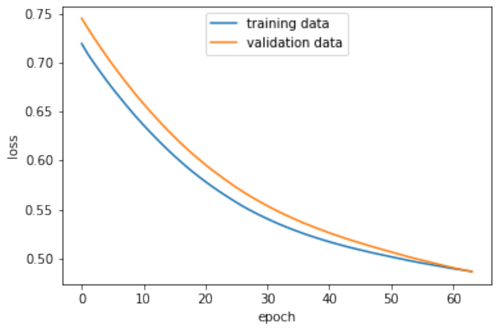
Because the problem is binary classification, the last layer should be sigmoid; however, these next models used relu instead. I trained these models for 64 epochs.

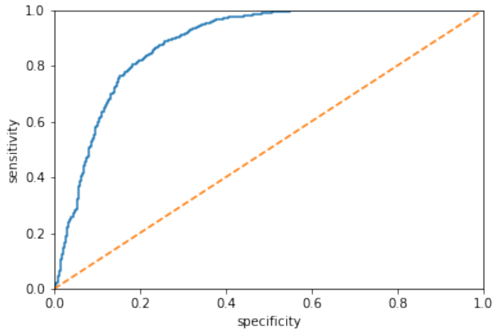
4.4.1 Model 1

This model uses 1 layer and 8 neurons.

Accuracy	Loss	Precision	Recall	F1 Score	AUC
0.806751	0.486973	0.786398	0.826788	0.806087	0.880884



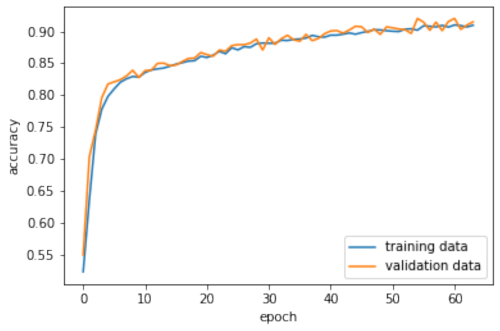


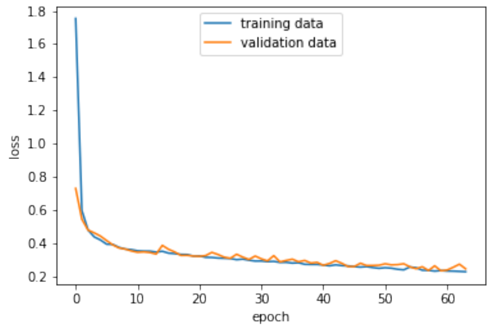


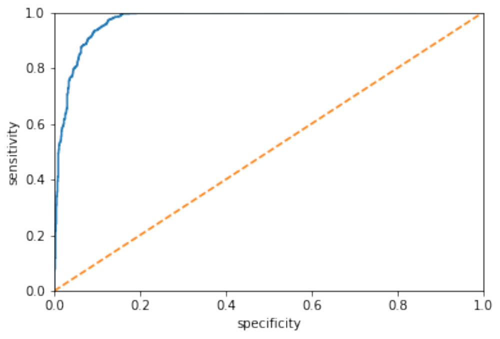
4.4.2 Model 2

This model uses 3 layers and 64 neurons in layer 3, 16 neurons in layer 2, and 1 neuron in layer 1.

Accuracy	Loss	Precision	Recall	F1 Score	AUC
0.914873	NaN	0.853913	0.993927	0.918616	0.970585







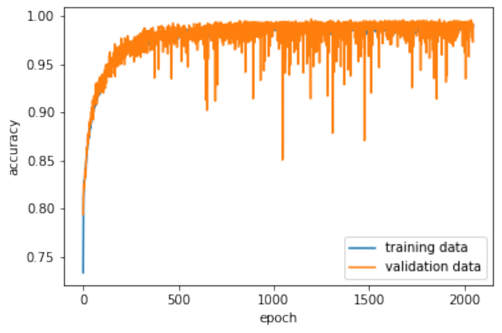
4.5 Best Model

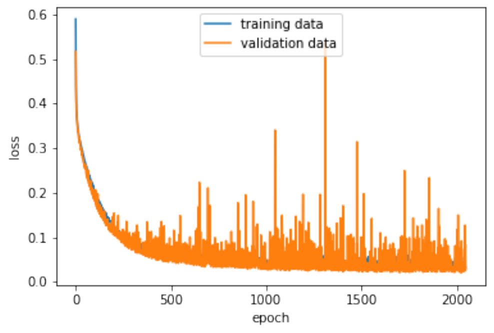
The best model found was model 4 from the tests in section 4.2. Model 4 was a 3 layer model, using 128 neurons for the 3rd layer, 32 neurons for the 2nd layer, and 1 neuron for the last layer. It was trained for 128 epochs and used a batch size of 256. The optimizer was rmsprop and the activation layers used were relu execpt for the final layer using sigmoid. The accuracy of the model was consistently similar for the training and validation data throughout the fitting of the model. The ROC curve produced was also very close to the top left corner.

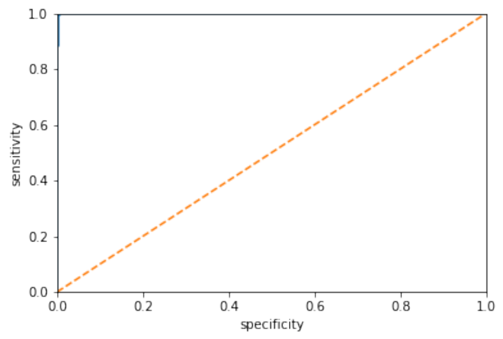
5 Early Stopping and Checkpointing

Using the values from the best model found from the tests in section 4, I created a model to used checkpointing and early stopping. I trained this model with a patience of 512 for 2048 epochs. It completed the epochs and found the best loss value at epoch 1701.

	Accuracy	Loss	Precision	Recall	F1 Score	AUC
0	0.993151	nan	0.991194	0.995088	0.993137	0.997627



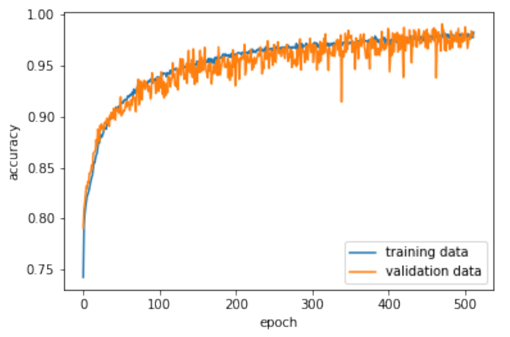


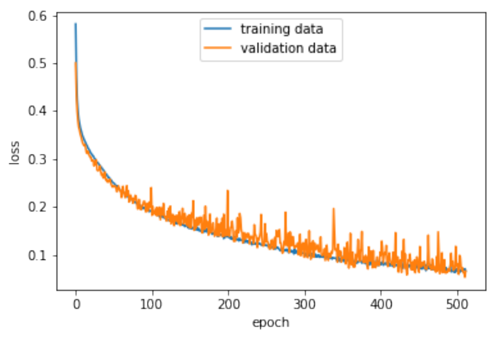


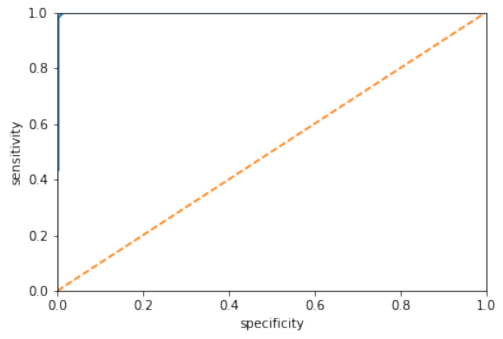
6 Best Fit Model with Highest Accuracy

Using the values from the best model found from the tests in section 4, I created the same model but trained it for more epochs. When I trained it for very large amounts of epochs, sometimes the results were unpredictable, and the model would become overfit to the training data. When I lowered the epochs to 512, or 4 times larger than the original test, then the model performed better and had a much lower validation loss score than the original test. This model did not perform as well as the model from the checkpointing test.

	Accuracy	Loss	Precision	Recall	F1 Score	AUC
0	0.978474	0.065543	0.958212	0.998987	0.978175	0.997898







References

[1] Vance, T. D. (2019, September 10). Arcade game stats. Kaggle. Retrieved February 21, 2022, from https://www.kaggle.com/depmountaineer/arcade-game-stats