# TRAINING A NEURAL NETWORK CAPABLE OF PREDICTING A FINANCIAL MARKET.

A DISSERTATION SUBMITTED TO COVENTRY UNIVERSITY FOR THE DEGREE OF BACHELOR OF ENGINEERING IN THE FACULTY OF ENGINEERING, ENVIRONMENT AND COMPUTING

By

**Mr Paul J. ARU**
Computer Hardware and Software Engineering
SID – **7155312**

Supervisor – **Mr Christopher J. BASS**

April 2021

## Abstract

This project set out with the lofty goal of examining the efficacy of Deep Learning models for forecasting financial markets. After completing a thorough background research in the form of a literature review, we pursued to apply the best know practises. Following the development of our project's tools, we continued by testing 8 key parameters that could affect the performance of our neural networks. Through the development of 139 different models, we were able to analyse several theories and learned about some of the aspects that influence a NN's performance. While we succeeded in our goal to predict a financial market to some extent, we also discussed why this probably will not be the get-rich-quick scheme many might hope.

# Table of Contents

**ABSTRACT** ..... **2**

**TABLE OF CONTENTS** ..... **3**

**1. INTRODUCTION** ..... **5**

1.1 BACKGROUND CONTEXT ..... 5

1.1 AIMS ..... 5

1.2 OBJECTIVES ..... 5

**2. LITERATURE REVIEW / BACKGROUND RESEARCH** ..... **7**

2.1 FEED-FORWARD TOPOLOGIES ..... 7

*2.1.1 Multi-Layer Perceptron (MLP)* ..... *7*

*2.1.2 Bayesian Neural Network* ..... *8*

2.2 RECURRENT TOPOLOGIES ..... 8

*2.2.1 Long Short-Term Memory (LSTM)* ..... *8*

*2.2.2 Convolutional Neural Network (CNN)* ..... *9*

*2.2.3 Unspecified* ..... *9*

2.3 UNSPECIFIED TOPOLOGIES ..... 9

2.4 CONCLUSION ..... 9

**3. PROJECT TOOLS** ..... **11**

3.1 BUILDING A DATASET SCRIPT ..... 11

*3.1.1 Data Source* ..... *11*

*3.1.2 Data Processing* ..... *11*

3.2 CREATING A MODEL TRAINING SCRIPT ..... 12

*3.2.1 Structure Design* ..... *12*

*3.2.2 Performance Measurement* ..... *12*

**4. PROJECT TESTS** ..... **14**

4.1 DATA POINT ADJUSTMENTS ..... 14

*4.1.1 Window Period* ..... *14*

*4.1.2 Number of Symbols*....15

4.2 Dataset Tweaks....16

*4.2.1 Number of Classes*....16

*4.2.2 Size/Age of Dataset*....18

*4.2.3 Financial Market*....19

4.3 Model Training Modifications....20

*4.3.1 Number of Epoch*....20

*4.3.2 Size of Mini-Batch*....22

*4.3.3 Number of Hidden Units*....22

**5. SUMMARY....24**

5.1 Critical Appraisal....24

5.2 Conclusion....24

5.3 Further Work....25

5.4 Personal Reflections....26

**6. GLOSSARY....27**

**7. LIST OF REFERENCES....28**

**8. APPENDICES....30**

8.1 Literature Table....30

8.2 Complete Test Results Table....32

8.3 Project Repository....35

8.4 Microsoft Project Initial Timeline....36

8.5 Logbook....37

8.6 Project Meeting Notes....42

*8.6.1 Meeting 1*....42

*8.6.2 Meeting 2*....43

*8.6.3 Meeting 3*....44

# 1. Introduction

## 1.1 Background Context

Machine Learning, and Deep Learning in particular, have seen a rapid growth of interest in recent years (Google Ngram, 2020). This is most likely thanks to some of the recent significant improvements in the field, which have led us to a world of self-driving cars, autonomous drones, Deepfake clips and Orwellian surveillance states. One of these advancements could be the development of AlexNet, a Deep Convolutional Neural Network that significantly outperformed previous ImageNet recognition challengers (Krizhevsky et al., 2017, p. 84). That, in turn, renewed peoples' interest in Deep Learning as a viable and promising field of Machine Learning.

Whether due to the lockdowns of the *COVID-19* pandemic or as a result of increased accessibility through mobile trading apps, Google recorded an all-time record interest in search terms related to stock market in March 2020 (Google Trends, 2020). With potentially many new and amateur traders gambling their savings away, seasoned stockbrokers stand to make a nice profit. Predicting a stock market in your favour, however, might not require years of learning and experience.

The concept of stock forecasting with the help of Machine Learning is nothing new. A lot of the research in the field can be traced back to the early 1990s, if not earlier. However, with technological advancements bringing us enormously increased compute power, much of that research has now moved from theorising to applying it in practice.

## 1.1 Aims

The aim of the project will be to evaluate the feasibility and profitability
of using a Deep Learning model for speculating a financial market. While this is by no means the first project to use financial data for Deep Learning, one of the distinguishing factors is the goal to produce an accessible report to readers with no background in finance or Machine Learning. The rationale behind constraining the project to just the Deep Learning field of Machine Learning lies within the recent advancements and achievements in not just Deep Learning in general but also in conjunction with financial data (Sujatha & Sundaram, 2010, p. 59).

## 1.2 Objectives

- Selecting Sources and Creating a Dataset – While the first objective might seem quite straightforward, it involves several sub-objectives. First of all, we have to select symbols that we will use for observing and investing. The second sub-objective is to find an approach to download all of the related data. Last, but certainly not least, that data will have to be processed to a format that our model can read as input.
- Designing and Training a Model – There is a wide variety of commonly used neural network structures, each with numerous factors to tweak. As a result, the first sub-objective would be to select an appropriate architecture for our model (CNN, LSTM, etc…). Next, we will have to determine the optimal configuration (number of layers, batch size, etc…).

- Measuring Performance – First, since there are many ways to assess how well a model is working, an appropriate metric will have to be selected. The next sub-objective is to conduct numerous tests to come up with quantifiable results. Lastly, if there is time we could develop a simulator to compare the proposed model against real-world benchmarks.

In case the outcome of the project indicates that a Deep Learning model can profitably predict a financial market, it could serve as a starting point for a trading bot. A significant breakthrough in this field could have a wider impact on how the financial markets operate.

# 2. Literature Review / Background Research

The following paragraphs will analyse a selection of recent scholarly literature on the application of Deep Learning for Stock Market forecasting. The research papers selected for our literature review were sourced from *IEEE*, *IOP* & *Google Scholar*; with 10, 5 & 5 documents respectively.

While there are many ways to classify Deep Learning, the approaches taken in the aforementioned literature can roughly be split in two based on the node connection structure. That is to say, a Neural Network model can be designed to have either feed-forward connections or recurrent connections. In layman's terms, this indicates whether the information provided to the network input travels in a single direction towards the output or both directions, affecting neurons that it has already passed through.

## 2.1 Feed-forward Topologies

### 2.1.1 Multi-Layer Perceptron (*MLP*)

One of the most common feed-forward topologies is Multi-Layer Perceptron or MLP for short. Out of the 20 scholarly literature studied, 7 were about, or included a comparison to, some form of a Multi-Layer Perceptron. That being said, more often than not the topology makes an appearance in older documents, possibly due to the advancements with certain recurrent topologies.

Both Sujatha & Sundaram (2010, p. 59) and Yousif & Elfaki (2017, p. 1) compared an MLP NN to classical statistical models. In their investigation of the Bombay stock index, Sujatha & Sundaram (2010, p. 62) concluded that their model is better than moving average, linear regression, quadratic curve, cubic curve and non-parametric linear regression with Sen's slope estimator. In addition to the quadratic curve, Yousif & Elfaki (2017, p. 6) put their MLP Artificial Neural Network also against Holt's trend model and AutoRegressive Integrated Moving Average (*ARIMA*) on the Doha market. Confirming the earlier findings, they found that their model "shows a better forecasting than linear time series models".

Moghaddam et al. (2015, p. 92) created MLP models for predicting the Nasdaq index. They focused their investigation on the inner structure of the network and looked at the effect of different layer and neuron configurations. They deduced that models with 40-40 and 20-40-20 configurations returned the best results.

Guresen et al. (2011, p. 10390) also used the Nasdaq index to compare their Multi-Layer Perceptron against another feed-forward model called DAN2 and hybrid models based on the two. Their conclusion showed that a basic MLP outperforms the other tested approaches.

The first study from our list to compare an MLP against a recurrent Neural Network predates the creation of AlexNet. In 2010, Naeini et al. (2010. p. 132) compared their Multi-Layer Perceptron against an Elman recurrent NN. They reached an interesting conclusion in that the MLP came out on top in value prediction, however, Elman performed better in forecasting the direction of change. More recently, Akita et al. (2016, p. 1) and Pang et al. (2018, p. 2098) compared the Multi-Layer Perceptron against another recurrent topology, the Long Short-Term Memory or LSTM for short. These two documents also stand out for another reason. They are in the tiny minority of 3 papers that not only include numeric financial data as the input but also text-based news. When simulating the Tokyo stock exchange, Akita et al. (2016, p. 5) found that not only did their LSTM model return higher profits than their MLP model but that they could also achieve higher profits / smaller losses with a simple recurrent NN and even a Support Vector Regression model. The latter of which is not a Deep Learning model, but rather a supervised Machine Learning model. Pang et al. (2018, p. 2112) manage to partially support this claim, by demonstrating that their LSTM models have higher accuracy than their MLP model with a Shanghai stock index. However, when it comes to forecasting a single stock, the results are mixed.

#### 2.1.2 Bayesian Neural Network

However, before moving on to Long Short-Term Memory and other Recurrent Neural Networks, there is one more example of applying a feed-forward network for stock market forecasting. Ticknor (2013, p. 5501) created a Bayesian NN for predicting individual American stocks and compared it against a fusion model with weighted average and ARIMA. Like the previous attempt at predicting individual stocks, this paper also presents mixed results where the proposed model accuracy slightly outperforms other candidates with one stock but falls a little short with another. While the author claims that their approach "reduces the potential for overfitting and local minima solution", it seems a bit inconclusive considering the limited dataset.

### 2.2 Recurrent Topologies

#### 2.2.1 Long Short-Term Memory (*LSTM*)

Even though it might by now seem that Multi-Layer Perceptron is the most used topology in our literature, then the Long Short-Term Memory actually has it beat. The recurrent architecture makes an appearance in 8 of our 20 papers. Like the name hints, the unique element of LSTM lies within its ability to memorise points with time-series data. Our selection of scholarly documents also seems to point that this is a newer approach in the field, with half of the research being from the current or previous year.

The first mention of Long Short-Term Memory in our list dates back to 2015 when Chen et. al. (2015, p. 2823) used LSTM models on Chinese stock markets. They focused on investigating the accuracy of the model based on the inputs provided and found two key components that resulted in a significant increase. Normalising the input data resulted in a 3.6% increase and including a stock index boosted the accuracy by 4%. Li et. al. (2019, p. 551) also investigated just the LSTM model from a Chinese perspective, albeit with a data set that covers a time period of five times as long. The authors summarise by stating that their model "has a good predictive effect on P/E ratio sequence", however, it is hard to conclude much from that as they have not compared their model against other prediction methods.

Sachdeva et. al. (2019, p. 1) investigated the effect of different optimisers, time steps and layer count on a Long Short-Term Memory model in the Indian stock exchange. They found that the RMSprop optimiser with 60-time-steps and 1-layer resulted in better accuracy than combinations with the Adam optimiser, 20-time-steps and 4-layers.

Pedrozo et. al. (2020, p. 1) took a very unique approach by creating a Foreign Exchange currency-pairs correlation analysis and used that as one of the inputs for their LSTM model. While no investigation was performed into the exact effect it had on the accuracy of the model, it is certainly an interesting factor to consider.

In 2016, Deng et. al. (2016, p. 661) conducted research that compared a wide variety of Deep Learning architectures. Aside from Long Short-Term Memory; they also took a look at a simple RNN, a Deep Convolutional Neural Network and proposed a hybrid model; combining Deep Learning with Reinforcement Learning. The authors' profitability analysis portrays an intriguing image. Their hybrid model seems to be more consistent in producing profits, however, its profitability rate is below the other Neural Networks. This could be due to the more conservative number of trades done by the hybrid model. Also worth noting is that the comparison indicates that their LSTM is more profitable than their simple RNN, which in turn is more profitable than their DCNN.

Guo & Tuckfield (2020, p. 1) also compared a Long Short-Term Memory against a Deep Convolutional Neural Network, with the addition of 3 Machine Learning models and news headlines input data. As somewhat expected by this point; Random Forests, Naive Bayes and Gradient Boosting Machine all had lower accuracies than the 2 Deep Learning models. However, when comparing LSTM with DCNN, they found that while the accuracies were fairly similar for predicting stock indexes, then with individual stocks LSTM had a noticeable lead.

#### 2.2.2 Convolutional Neural Network (*CNN*)

Chen & He (2018, p. 1) focused solely on using a Deep Convolutional Neural Network when looking into the prediction of a Chinese stock market. Their investigation focused on the accuracy of the model with different sources. Unfortunately, the sources are not disclosed and as a result, the only notable conclusion is that "conv1d" function can be used for converting 1-dimensional data to 2 dimensions for the DCNN.

#### 2.2.3 Unspecified

Srinivasan & Lakshmi (2017, p. 1) used an unspecified Recurrent Neural Network for predicting various individual Indian stocks with a varied number of input features. However, like the previous study, since the features are not explicitly revealed it is difficult to draw any applicable conclusions.

### 2.3 Unspecified Topologies

This leaves us with 4 papers that have not specifically declared whether the models used are based on a feed-forward or recurrent structure. In 2016, Billah et. al. (2016, p. 1) proposed an improved Levenberg-Marquardt algorithm and compared it against a traditional Levenberg–Marquardt trained Neural Network and an Adaptive Neuro-Fuzzy Inference System. The author's results indicate that not only did their algorithm reduce error, but also the time consumed, and memory required. Also in that year, Chong et. al. (2016, p. 187) conducted an in-depth investigation comparing data representation methods alongside different Machine Learning models with a large selection of individual stocks from the Korean stock exchange. In their goal to just forecast the direction of change, Chong et. al. (2016, p. 202) conclude by stating that "DNNs perform better than a linear autoregressive model in the training set, but the advantage mostly disappears in the test set." In 2019, Loayza et. al. (2019, p.1) investigated the possibilities of applying a Neural Network for financial forecasting in Peru and found it helpful. Last of all, Li et. al. (2020, p.1) proposed a BackPropagating Adaboost Neural Network for predicting a Chinese securities index. Compared to a regular BackPropagating model, they were able to establish slightly improved results.

### 2.4 Conclusion

While this literature review covers a wide selection of approaches to using Deep Learning in financial market predictions, there are a few trends that we can pick up from all of this.

First of all, most, if not all, sources that compared a Deep Learning model up against Machine Learning models or statistical models found that Deep Learning was able to outperform the other two fields. This confirms that we should focus our project aim on the Deep Learning field.

Secondly, feed-forward topologies, or at least Multi-Layer Perceptron, seem(s) to be an older approach and often results in less accurate outputs. Long Short-Term Memory appears to be the go-to architecture in recent studies and while there were not many comparisons to Deep Convolutional Neural Networks, the few that did noticed similar or slightly worse performance. As a result, we have decided to look into the latter two as possible architectures for our model.

Several papers have also noted that there is a discrepancy in the prediction accuracy between market indexes and individual stocks. Whether it is due to the number of shareholders or some other factor, market indexes seem to be easier to predict. Therefore, when selecting the symbols for our dataset, we will make sure to include some indexes where possible as we work through model revisions.

Last of all, it is difficult to draw any concrete conclusions from the data used. There were data sets that consisted of only 88-days of data, all the way up to 11-years. That being said, a small time period did not necessarily result in a small sample sum, since the frequency of data points varied from days to minutes as well. However, there was one aspect of data that had very little coverage in general. Namely the number of previous data points used for the prediction. As a result, this might be a good knowledge gap to investigate further and consider in our model revisions.

# 3. Project Tools

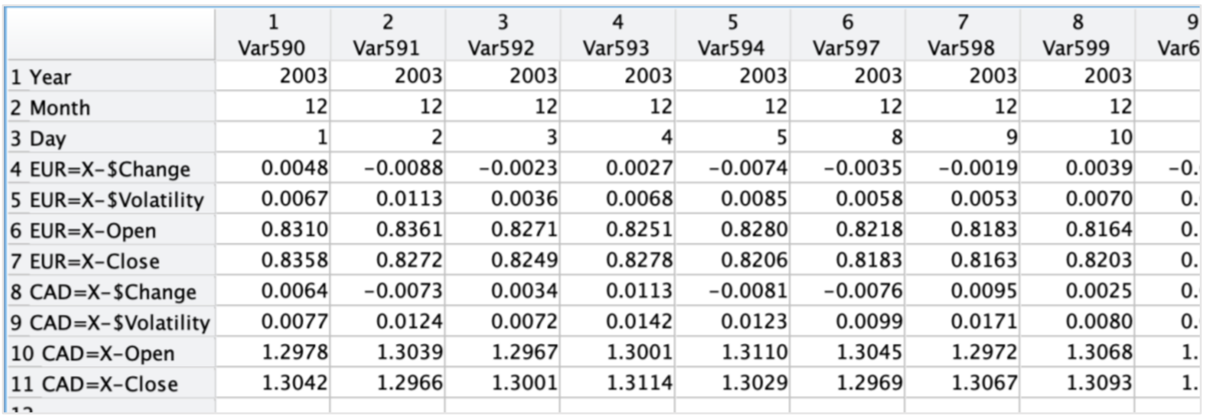
## 3.1 Building a Dataset Script

### 3.1.1 Data Source

A good dataset is a key component to training a great Neural Network model; data is also the most valuable commodity these days. While searching for a suitable source of market information, we encountered several platforms that offered this for exorbitant fees. In the end, we found a third-party MATLAB library that enables us to access prices for a given symbol from **Yahoo Finance**. While this does limit the interval of available data from a minute-by-minute to a day-by-day basis, it is free to use.

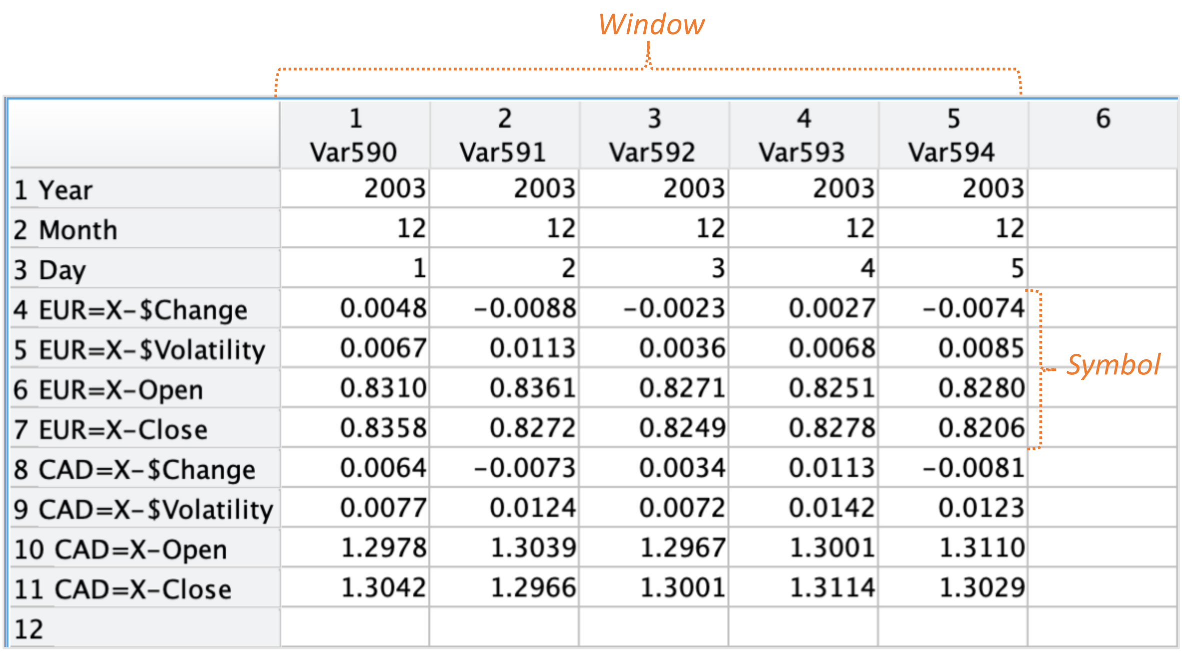
### 3.1.2 Data Processing

However, the construction of a dataset for training a Neural Network does not end there. The table of information that Artem Lensky's function provides to us from Yahoo Finance API still has to be sorted, classified, and stored. Our MATLAB function takes the valuable elements from a symbol's trading day and adds them to a temporary master table that contains all the symbols. Next, depending on the window size and number of classes that master table gets divided into sub-tables, which can be thought of as individual data points of our dataset. After that, these data points will be stored as *.csv* files in directories that correspond to their class value. Last, but not least, we have introduced dataset normalisation in the form of equalising the number of data points in each class. This is done by removing files from class directories that contain more samples than the class with the least number of data points.



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| | Var590 | Var591 | Var592 | Var593 | Var594 | Var597 | Var598 | Var599 | Var6 |
| 1 Year | 2003 | 2003 | 2003 | 2003 | 2003 | 2003 | 2003 | 2003 | |
| 2 Month | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | |
| 3 Day | 1 | 2 | 3 | 4 | 5 | 8 | 9 | 10 | |
| 4 EUR=X-$Change | 0.0048 | -0.0088 | -0.0023 | 0.0027 | -0.0074 | -0.0035 | -0.0019 | 0.0039 | -0. |
| 5 EUR=X-$Volatility | 0.0067 | 0.0113 | 0.0036 | 0.0068 | 0.0085 | 0.0058 | 0.0053 | 0.0070 | 0. |
| 6 EUR=X-Open | 0.8310 | 0.8361 | 0.8271 | 0.8251 | 0.8280 | 0.8218 | 0.8183 | 0.8164 | 0. |
| 7 EUR=X-Close | 0.8358 | 0.8272 | 0.8249 | 0.8278 | 0.8206 | 0.8183 | 0.8163 | 0.8203 | 0. |
| 8 CAD=X-$Change | 0.0064 | -0.0073 | 0.0034 | 0.0113 | -0.0081 | -0.0076 | 0.0095 | 0.0025 | 0. |
| 9 CAD=X-$Volatility | 0.0077 | 0.0124 | 0.0072 | 0.0142 | 0.0123 | 0.0099 | 0.0171 | 0.0080 | 0. |
| 10 CAD=X-Open | 1.2978 | 1.3039 | 1.2967 | 1.3001 | 1.3110 | 1.3045 | 1.2972 | 1.3068 | 1. |
| 11 CAD=X-Close | 1.3042 | 1.2966 | 1.3001 | 1.3114 | 1.3029 | 1.2969 | 1.3067 | 1.3093 | 1. |

*Figure 1 - The Master Table*


Window
Symbol

| | 1 Var590 | 2 Var591 | 3 Var592 | 4 Var593 | 5 Var594 | 6 |
|---|---|---|---|---|---|---|
| 1 Year | 2003 | 2003 | 2003 | 2003 | 2003 | |
| 2 Month | 12 | 12 | 12 | 12 | 12 | |
| 3 Day | 1 | 2 | 3 | 4 | 5 | |
| 4 EUR=X-$Change | 0.0048 | -0.0088 | -0.0023 | 0.0027 | -0.0074 | |
| 5 EUR=X-$Volatility | 0.0067 | 0.0113 | 0.0036 | 0.0068 | 0.0085 | |
| 6 EUR=X-Open | 0.8310 | 0.8361 | 0.8271 | 0.8251 | 0.8280 | |
| 7 EUR=X-Close | 0.8358 | 0.8272 | 0.8249 | 0.8278 | 0.8206 | |
| 8 CAD=X-$Change | 0.0064 | -0.0073 | 0.0034 | 0.0113 | -0.0081 | |
| 9 CAD=X-$Volatility | 0.0077 | 0.0124 | 0.0072 | 0.0142 | 0.0123 | |
| 10 CAD=X-Open | 1.2978 | 1.3039 | 1.2967 | 1.3001 | 1.3110 | |
| 11 CAD=X-Close | 1.3042 | 1.2966 | 1.3001 | 1.3114 | 1.3029 | |
| 12 | | | | | | |

*Figure 2 - Example Data Point with a Window Size 5.*

## 3.2 Creating a Model Training Script

### 3.2.1 Structure Design

Based on the Literature Review and thanks to previous research done in the field, **Long Short-Term Memory** was selected as the architecture that we would be focusing on. Not only is it an ideal choice due to its design around time-series data processing, but also in the aforementioned studies LSTM was often the best performing approach. Fortunately, MATLAB's Deep Learning Toolbox has made the development of a Neural Network a breeze. Our Long Short-Term Memory model is based on a MATLAB tutorial and its structure can be seen in the illustration below.

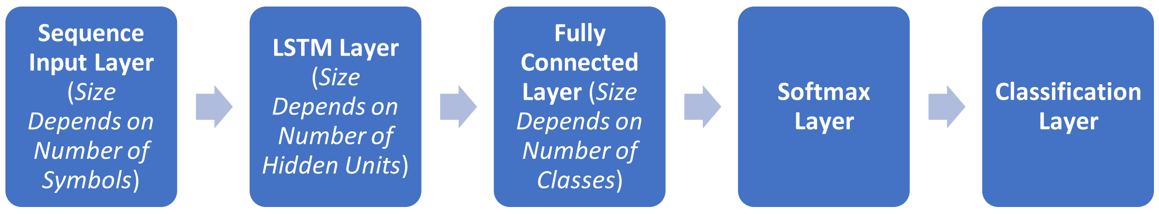

Sequence Input Layer (Size Depends on Number of Symbols)
LSTM Layer (Size Depends on Number of Hidden Units)
Fully Connected Layer (Size Depends on Number of Classes)
Softmax Layer
Classification Layer

*Figure 3 - Overview of Our Model's Layers*

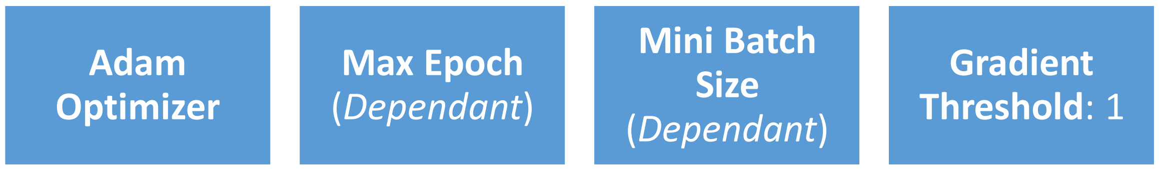


*Figure 4 - Overview of Our Model's Options*

### 3.2.2 Performance Measurement

A vital part of creating Neural Network models is assessing how well they operate. We began with a simple accuracy calculator, where a section of the dataset that was not used for training is processed through the model. The total number of hits and misses is averaged together to find a percentage representation of the **Precise Accuracy**.

As the project evolved onwards and we began tweaking the number of classes in our dataset and model, we realised that a new accuracy calculator was required. This would enable us to compare the more widely classified models to the original 3-class models (*Up*, *No-Change*, *Down*). Our solution, the **Directional Accuracy** calculator expands on the previous solution by considering traditional misses as hits so long as the direction of the change was still correct. That is to say, if a model predicted that a symbol's price would decrease by $1 to $5 but in actuality, it decreased by $0.1 to $1, the Precise Accuracy would mark this as a miss whereas the Directional Accuracy would mark this as a hit.

Late into the project we also introduced a function that would **Simulate Earnings** with a provided model. In our configuration, the simulator would make fictional buy and sell decisions while going through the last 3-months of market data. In addition, it should be noted that the simulation starts off with $0 and goes into negative to purchase 5 shares. At the end of the simulation period, it sells off all the shares that it still has left.

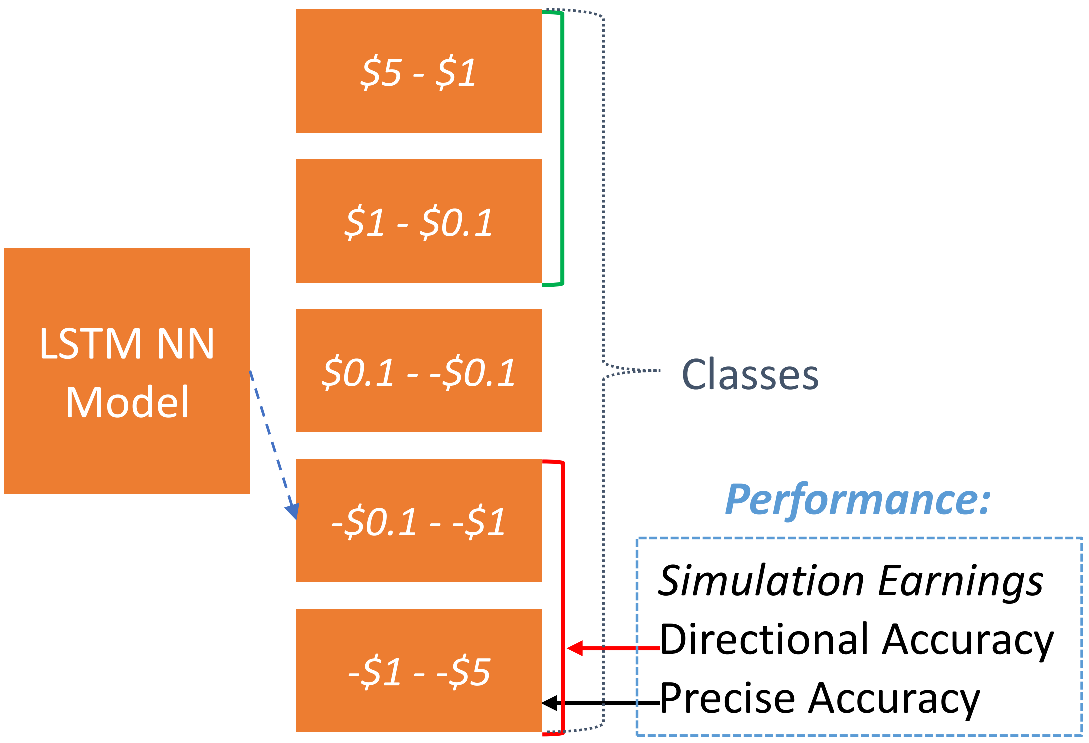


*Figure 5 - Performance Measurement Illustration*

# 4. Project Tests

## 4.1 Data Point Adjustments

### 4.1.1 Window Period

Like illustrated in *Figure 2*, the window size of a data point defines the number of days leading up to the prediction date. For example, a window period of 2 would be used to predict the following date's prices, based on information from current and previous days. In order to test the effect of window size on prediction accuracy, we devised 2 dedicated tests.

The first test used 20-Yearlong datasets of INTC (*Intel Corporation*) trading information, each split into 11 Classes. The models were trained with 1000 Hidden Units, an Epoch of 500 and a Mini-Batch Size of 2500. We tested Window Sizes of 1, 2, 3, 5, 7, 10, 14, 15, 20, 25 and 30 days. As the results below indicate, even though there **does not seem to be any noticeable correlation**, **the 14-day Window performed best**.

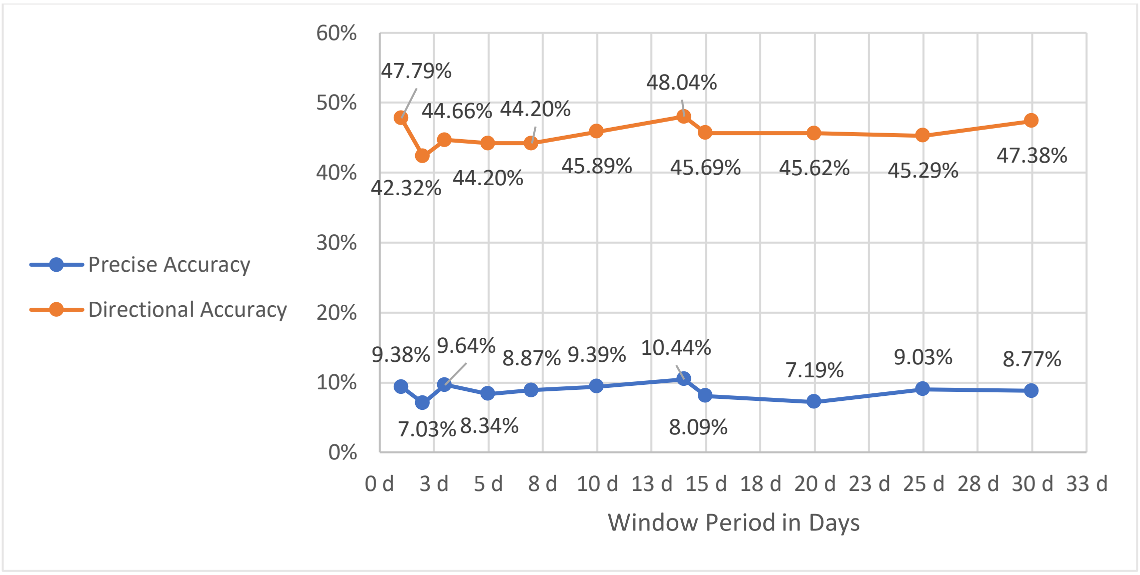


*Figure 6 - Dedicated Window Period Test 1*

This was followed up by a second test where most of the model parameters were increased, but the number of test samples was decreased. The datasets consisted of 50 Years of market info on IBM (*International Business Machines Corporation*) and ^GSPC (*S&P 500*) split into 3 Classes. The models were trained with 1500 Hidden Units, an Epoch of 5000 and a Mini-Batch Size of 1000. Once again; the tested Window Sizes of 3, 5, 10, 30 and 50 days **did not hint at any correlation**. However, **the highest result with a 10-day Window** was also closest to our previous top contender.

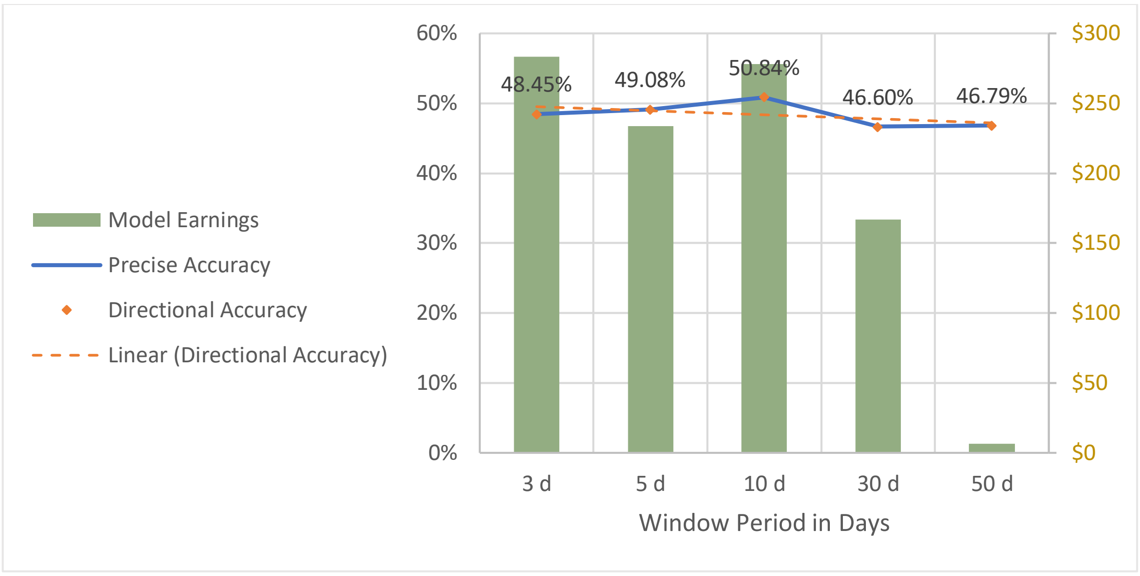


*Figure 7 - Dedicated Window Period Test 1*

However, before jumping to the conclusion that the ideal Window Size lies between 10 and 14 days, we should also take a look at the overall trend that covers all of the 139 models trained during this project. The results, seen below, seem to **conflict with our earlier findings** by suggesting that **the larger the Window Size, the more accurate our model** is.

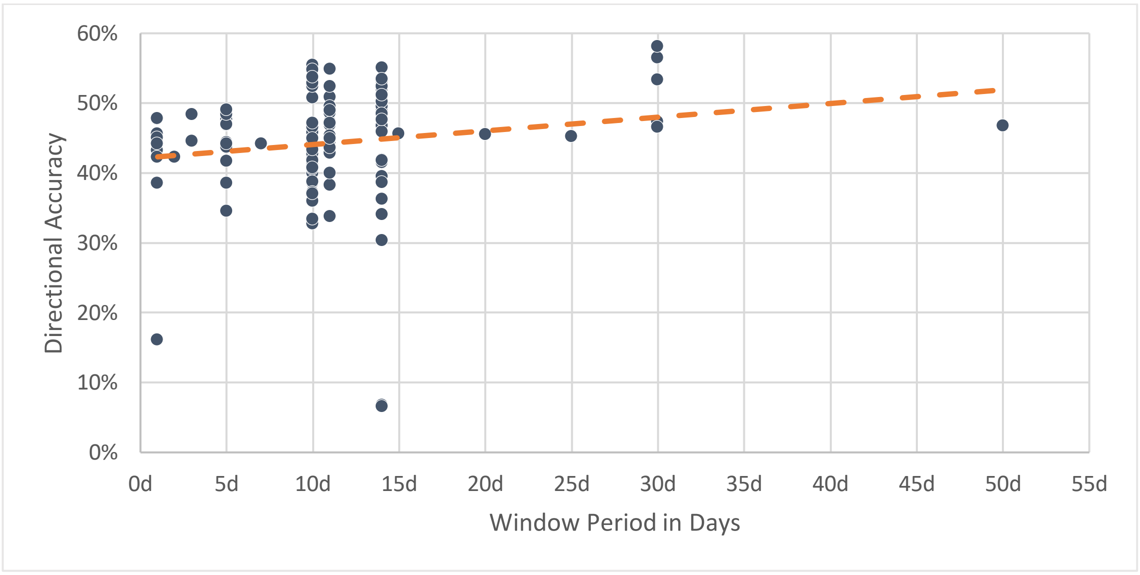


*Figure 8 - Window Period General Trend*

### 4.1.2 Number of Symbols

Also depicted in *Figure 2* is example information stored for a symbol and a portrayal of how additional symbols are handled. The rationale behind adding other symbols to our data point was to test whether the Neural Network could pick up any market segment or general economic trends by feeding it more data.

Our only dedicated test investigating this parameter had 20-Yearlong datasets centred around INTC (*Intel Corporation*), with 1-day Window Sizes. However; one-by-one we would add ^GSPC (*S&P 500*), AMD (*Advanced Micro Devices, Inc.*), ^DJI (*Dow Jones Industrial Average*), NVDA (*NVIDIA Corporation*), ^IXIC (*NASDAQ Composite*), HPQ (*HP Inc.*), LNVGY (*Lenovo Group Limited*), MSFT (*Microsoft Corporation*) and AAPL (*Apple*

*Inc.*). The networks were trained with 1000 Hidden Units, an Epoch of 500 and a Mini-Batch Size of 2500. Once again, it is **hard to draw any clear correlations** from these results, but we can note that the **best performing model did not include any additional symbols**.

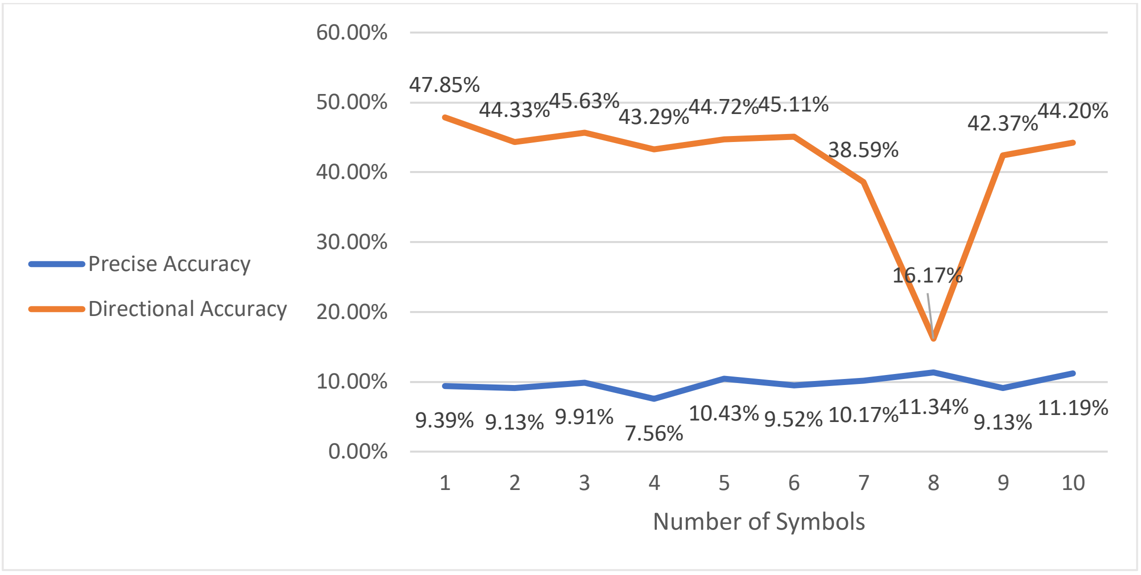


*Figure 9 - № of Symbols Dedicated Test*

Comparing the test results to the general trend (*from the 139 models*), one could say that it **somewhat confirms** our previous findings. The trend indicates that **the more additional symbols we include in a dataset the less accurate the model will be**.

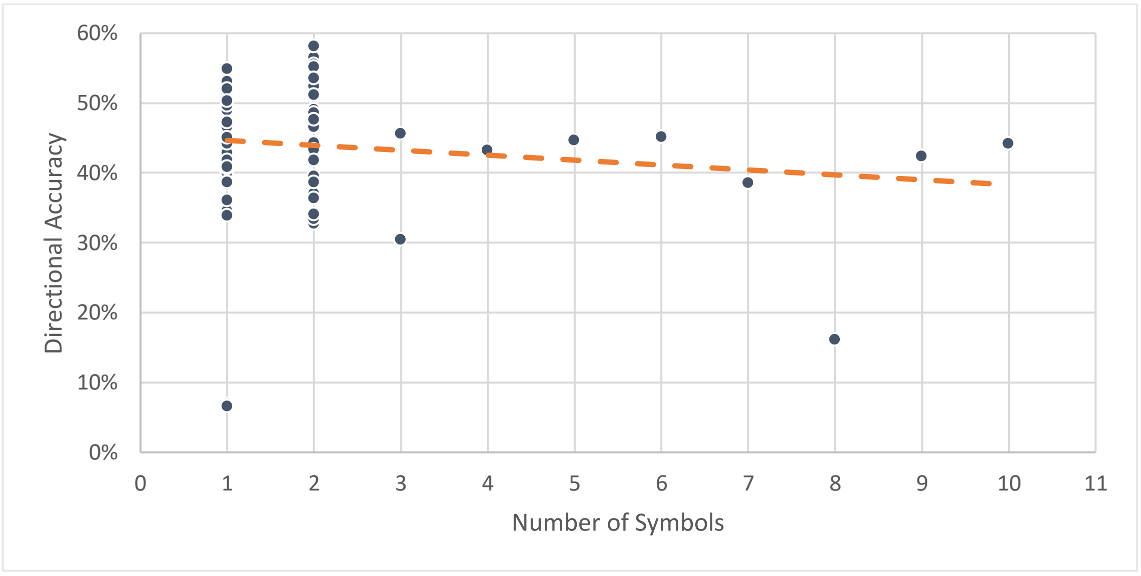


*Figure 10 - № of Symbols General Trend*

## 4.2 Dataset Tweaks

### 4.2.1 Number of Classes

As exhibited in *Figure 5*, in our project a Class defines a price change bracket. For example; with a Class size of 3 the dataset would normally be split into Up, No-Change and Down. As the value is increased, our script equally splits the dataset to reveal more precise price changes. At the same time, doing this decreases the number of data points each class has. This was the most extensively studied aspect, with 3 dedicated experiments.

Like a few other tests so far, we created datasets that contained 20-Years' worth of INTC (*Intel Corporation*) trading information, with Window Sizes of 5-Days. The models were trained with 1000 Hidden Units, an Epoch of 200 and a Mini-Batch Size of 5000. The results hinted at an interesting correlation, where the **accuracy increased until we reached a class size of 11**, at which point the improvement stagnated.

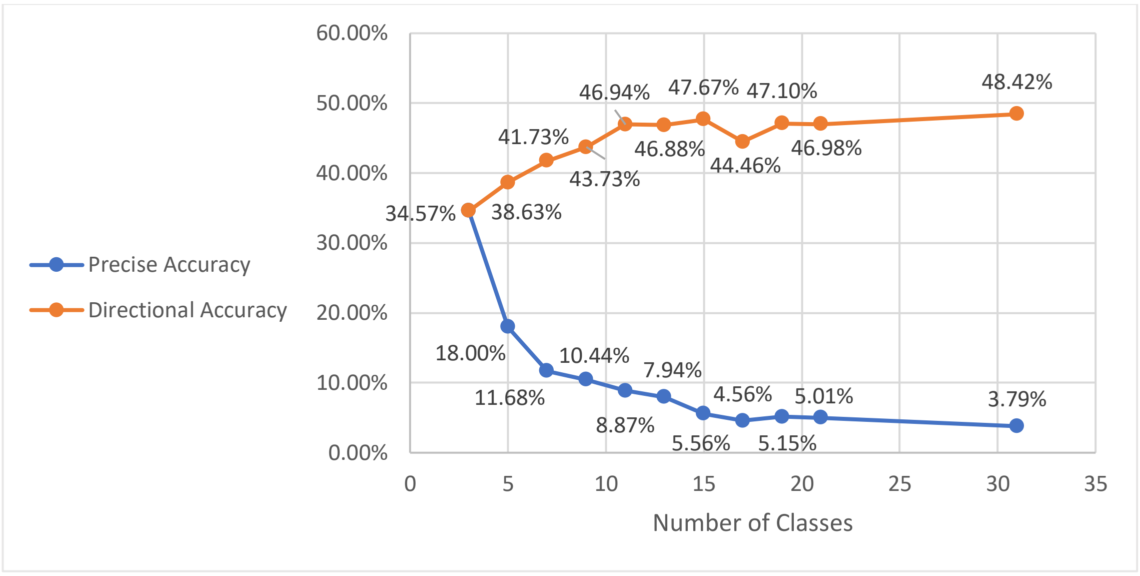


*Figure 11 - № of Classes Test 1*

Based on these results, we decided that it would be interesting to conceive a similar test where we also observe the changing number of samples in the classes. The only change we made to our datasets was to increase the Window Size to 11-Days. We also slightly adjusted the network training parameters by decreasing the Mini-Batch Size to 2500 and increasing the Epoch to 500. The results expanded on our earlier findings by revealing almost **logarithmic correlations between the number of classes and model accuracies**.

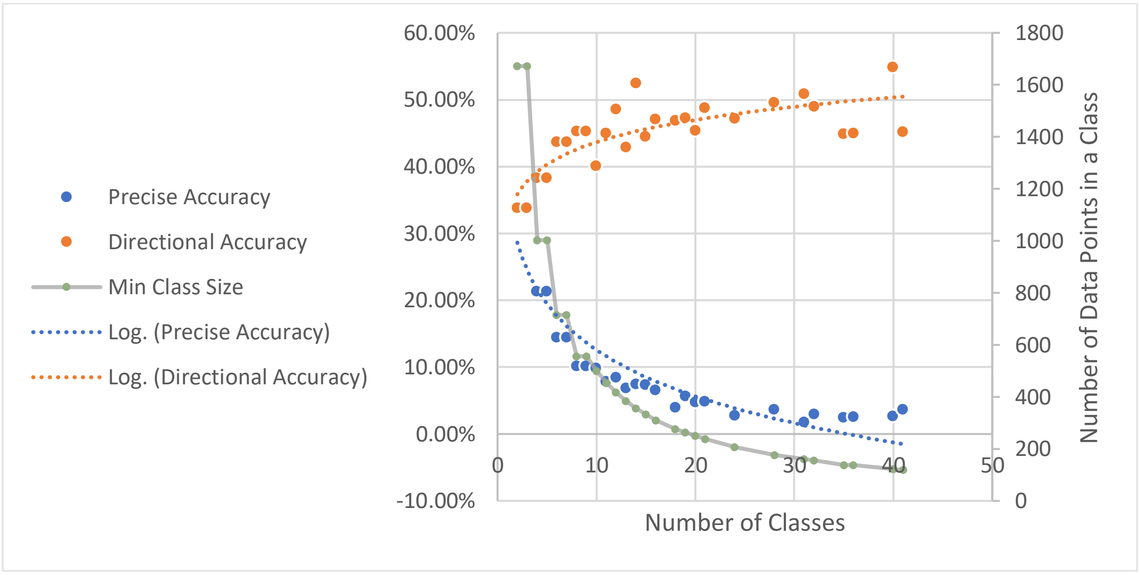


*Figure 12 - № of Classes Test 2*

The last test, or a comparison to be more accurate, was created to see the effect when all of the parameters are scaled up. This time we built 50-Yearlong datasets on MCD (*McDonald's Corporation*) with 30-Day Window Sizes and additional information on ^GSPC (*S&P 500*). Our models were trained with 1500 Hidden Units, and an Epoch limit of 5000 and a Mini-Batch size of 1500. This time we only compared Class Size 3 to 49. Interestingly enough, not only did the directional accuracy **favour the smaller class size**, but our earnings simulation also managed to produce a profit in this case.

| Results for Class Size 3 | |
|---|---|
| Precise Accuracy: | 58,17% |
| Directional Accuracy: | 58,17% |
| Earnings: | $125,33 |

| Results for Class Size 49 | |
|---|---|
| Precise Accuracy: | 6,85% |
| Directional Accuracy: | 53,39% |
| Earnings: | -$149,13 |

*Figure 13 - № of Classes Test 3*

By now we have already produced some conflicting results, so it is important to also look at the overall trend. While the **general trend complements our earlier findings**, we would not disregard the last experiment as an outlier. It is quite possible that with larger datasets the networks become better equipped to learn to distinguish large classes.

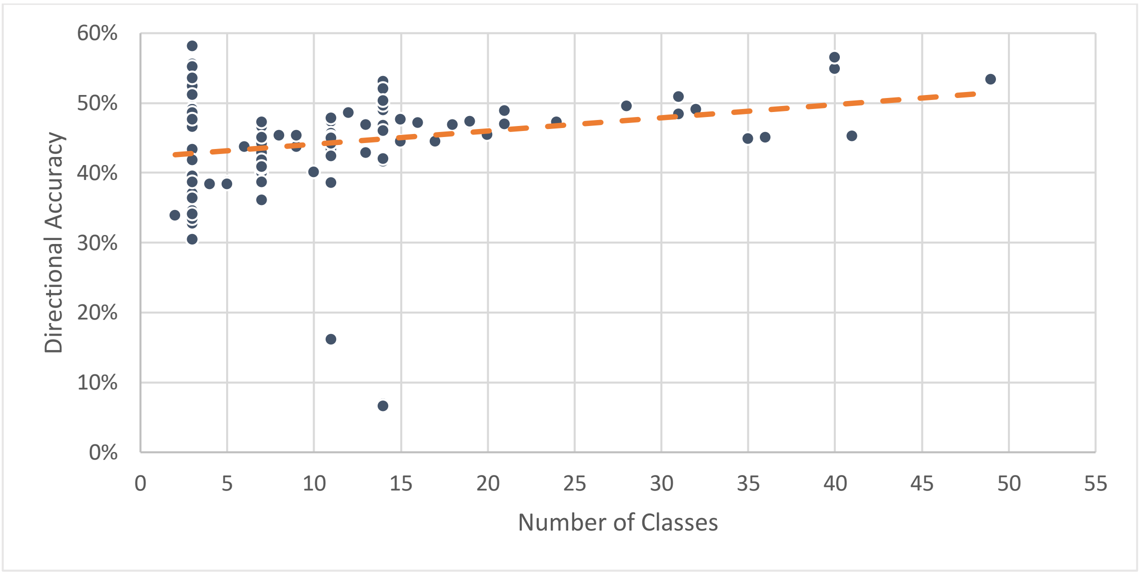


*Figure 14 - № of Classes General Trend*

### 4.2.2 Size/Age of Dataset

While this is practically common knowledge in the field, we decided to devise a single dedicated test to confirm that larger datasets improved the accuracy of our models. One thing to note here is that, since we could not increase the frequency of information from our source any further, we had to move the starting date back in time. This means that the results might be affected by long-term market trends that might not be visible otherwise.

While it took a while to find a suitable symbol with decades of trading information, we finally settled on creating our datasets on PG (*The Procter & Gamble Company*) with supporting information on ^GSPC (*S&P 500*). The data had a Window Size of 10 and was split into 3 Classes. The networks were trained with 1500 Hidden Units, a Mini-Batch Size of 1000 and an Epoch of 5000. While the earnings do not seem to indicate anything, the accuracy calculations affirm that **larger/older datasets produce more accurate models**.

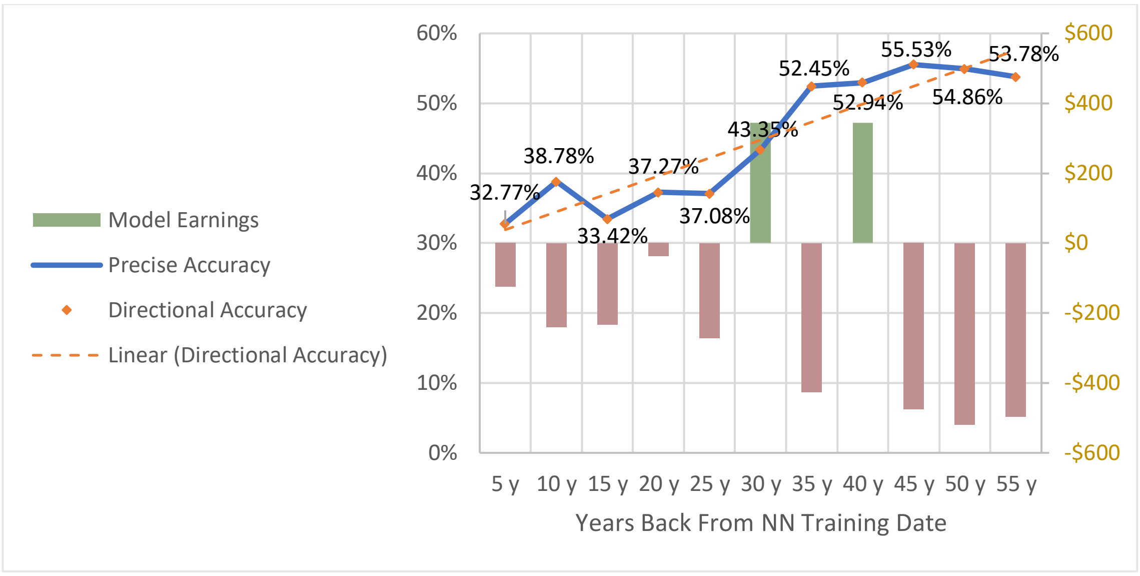


*Figure 15 - Dataset Size/Age Dedicated Test*

As always, we will compare the above results to the general trend in all of the 139 models designed during this project. Once again, we can indeed **confirm** that **the larger/older the dataset, the more accurate model** it can produce.

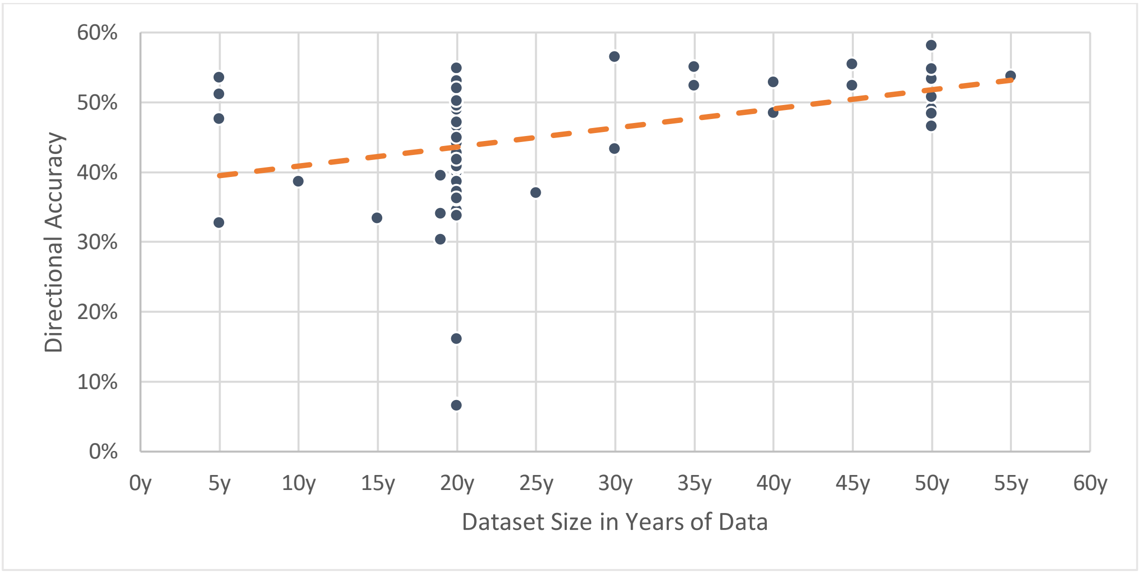


*Figure 16 - Dataset Size/Age General Trend*

### 4.2.3 Financial Market

We also decided to produce a simple comparison of the predictability of different markets. Since all of the 12 models created used different symbols, the available information also limited the size of the dataset. As a result, the only constant parameters of the Dataset were the Window Size of 14 and Class Size of 3. For training, we stuck with an Epoch of 5000 and 1500 for both the Hidden Units & Mini Batch Size. Like the preliminary results below indicate, **our models seem to perform better when trading with Stocks or Cryptocurrencies**. Interestingly enough, the dataset size varied the most between those two, with Cryptocurrencies only possessing 5 years of data when the Stocks averaged to 40 years.

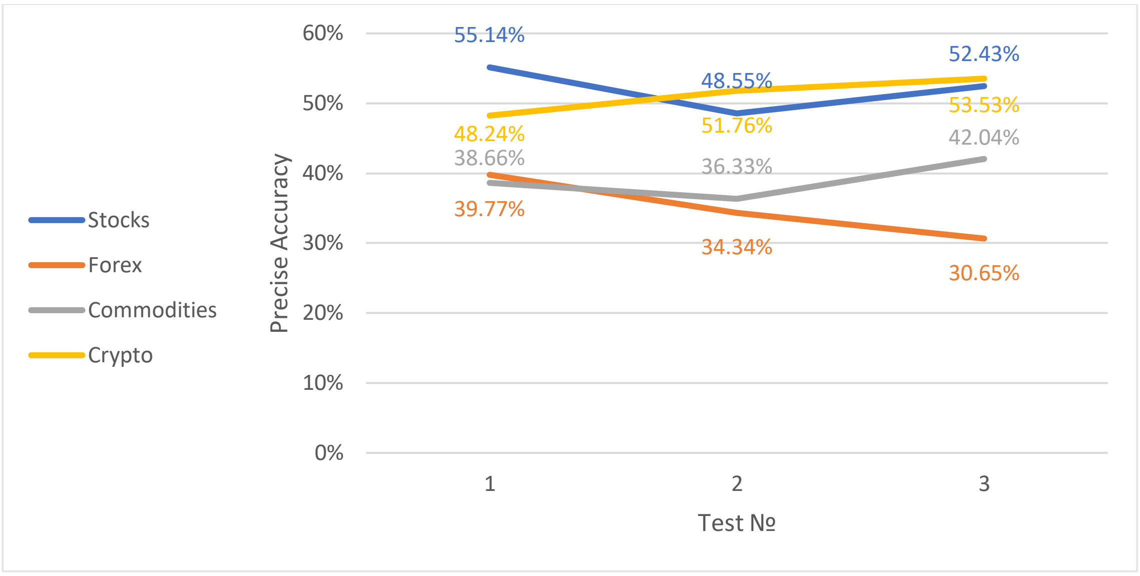


*Figure 17 - Financial Markets Comparison*

Unfortunately, since a large selection of our tests were devised around Stock market information, we were unable to produce general trends for all of the markets and as a result, **we can neither confirm nor refute the small-scale findings**.

## 4.3 Model Training Modifications

### 4.3.1 Number of Epoch

The first adjustable parameter in our model training section is Max Epoch. Since a single epoch indicates that the network has processed through the entire dataset once, this value can be thought of as how many loops through the dataset are done during the training process (MathWorks, 2021). This aspect will be covered in the form of a single dedicated test, an example model and finally the overall trend.

Our dedicated test datasets were built upon 20-Years of GC=F (*Gold futures*) trading information, with 10-Day Windows and split into 7 Classes. All the models were trained with a Hidden Unit value of 1500 and a Mini-Batch size of 1500. While the changing Epoch has not affected our accuracy readings, **from approximately 100 Epoch onwards half of the models produced a profit**.

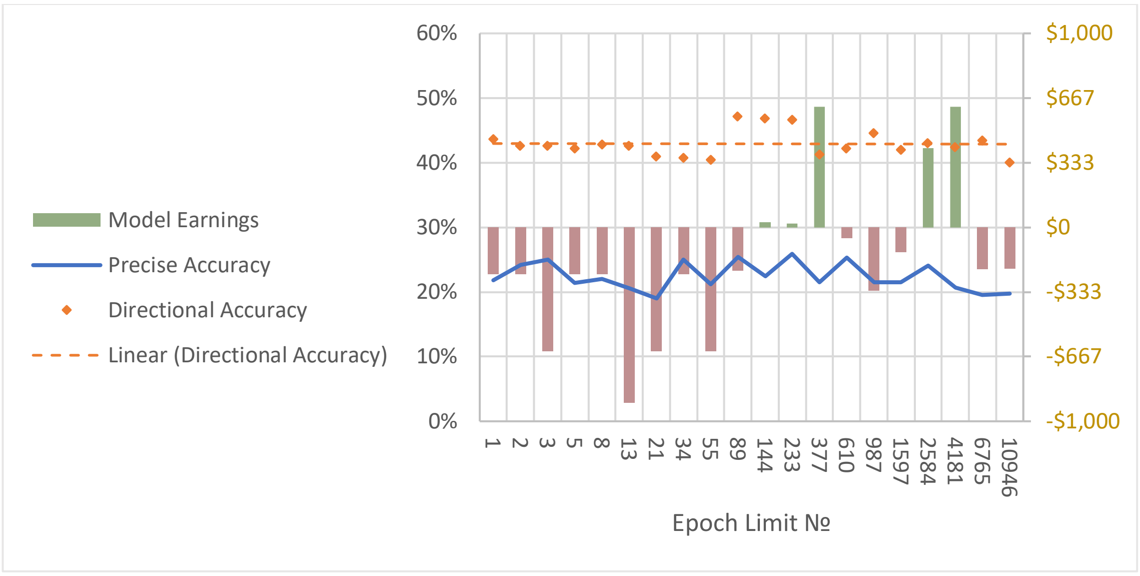
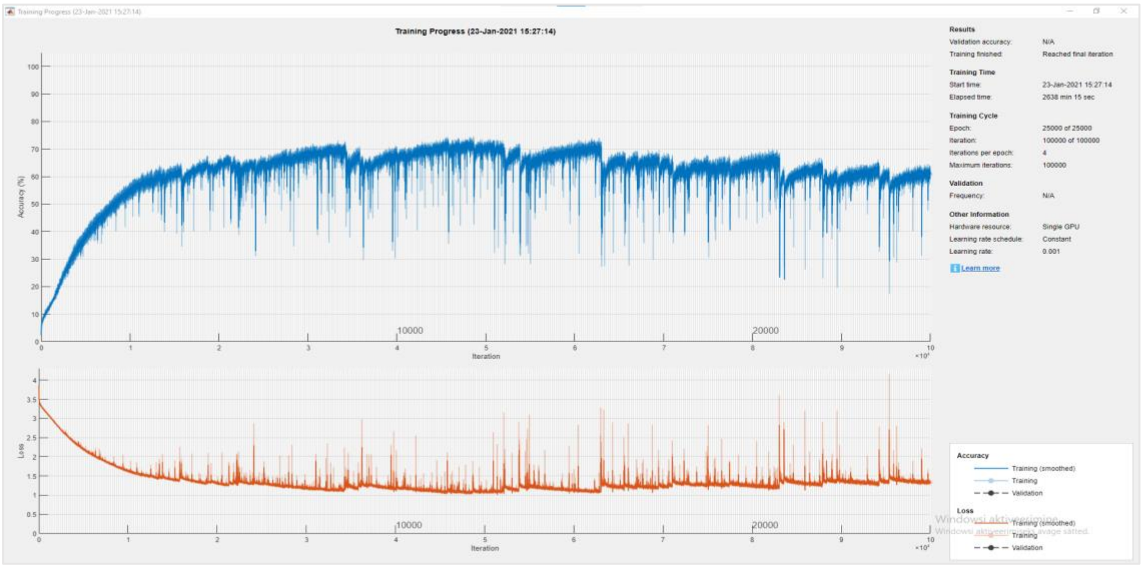


*Figure 18 - Epoch № Dedicated Test*

At this point, with the only seeming downside to an increased Epoch value being increased training time, we decided to investigate the effects of an even larger Epoch. We pushed the training system to its limit by setting the network training parameters to 1500 Hidden Units, a Mini-Batch Size of 1683, and an Epoch Limit of 25000. For the dataset, we used 30-Years of COKE (*Coca-Cola Bottling Co. Consolidated*) and ^GSPC (*S&P 500*), with 30-Day Windows and Split it all into 40 Classes. The Training Progress Window reveals that while the performance of the model does almost logarithmically keep increasing up to around 12500 Epochs, then after that point it begins degrading. This indicates that we **do not want to increase the value beyond a certain point**.



Results
Validation accuracy: N/A
Training finished: Reached final iteration
Training Time
Start time: 23-Jan-2021 15:27:14
Elapsed time: 2638 min 15 sec
Training Cycle
Epoch: 25000 of 25000
Iteration: 100000 of 100000
Iterations per epoch: 4
Maximum iterations: 100000
Validation
Frequency: N/A
Other Information
Hardware resource: Single GPU
Learning rate schedule: Constant
Learning rate: 0.001
Learn more

*Figure 19 - Super Run Training Progress*

While the previous findings do suggest that maxing out the Epoch can have a negative effect, then the overall trend does **confirm** that **larger Epoch limits produce better models**.

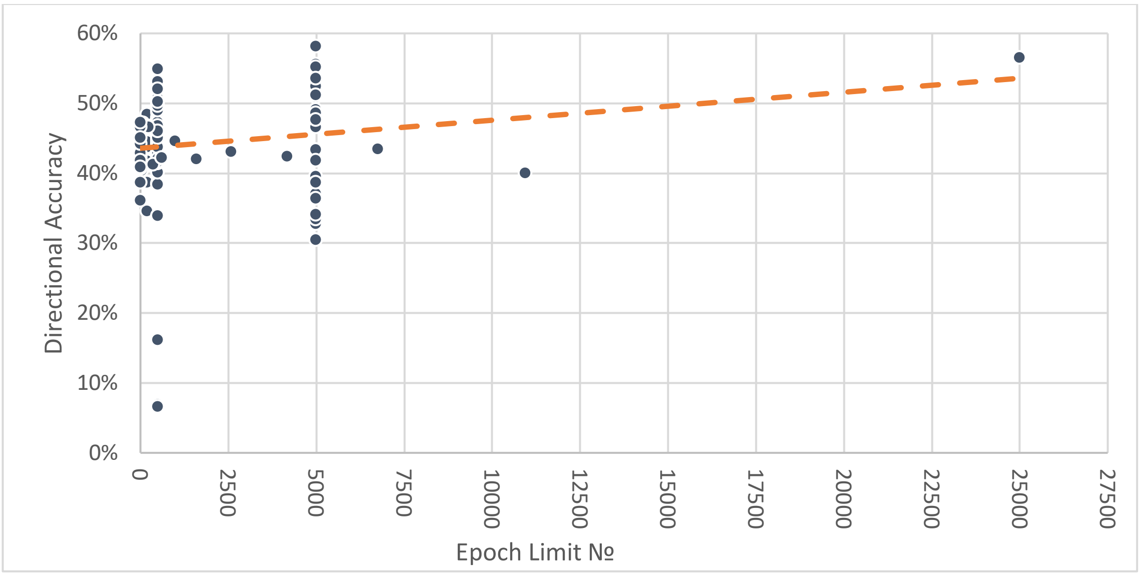


*Figure 20 - Epoch № Overall Trend*

### 4.3.2 Size of Mini-Batch

One of the lesser-studied aspects of the project is the Mini-Batch Size affect. MathWorks defines a Mini-Batch as "…a subset of the training set that is used to evaluate the gradient of the loss function and update the weights." (MathWorks, 2021) Aside from a few simple experiments, we can only observe the general trend here.

The results below indicate that there is **no visible correlation between Mini-Batch Size and the accuracy of a network**.

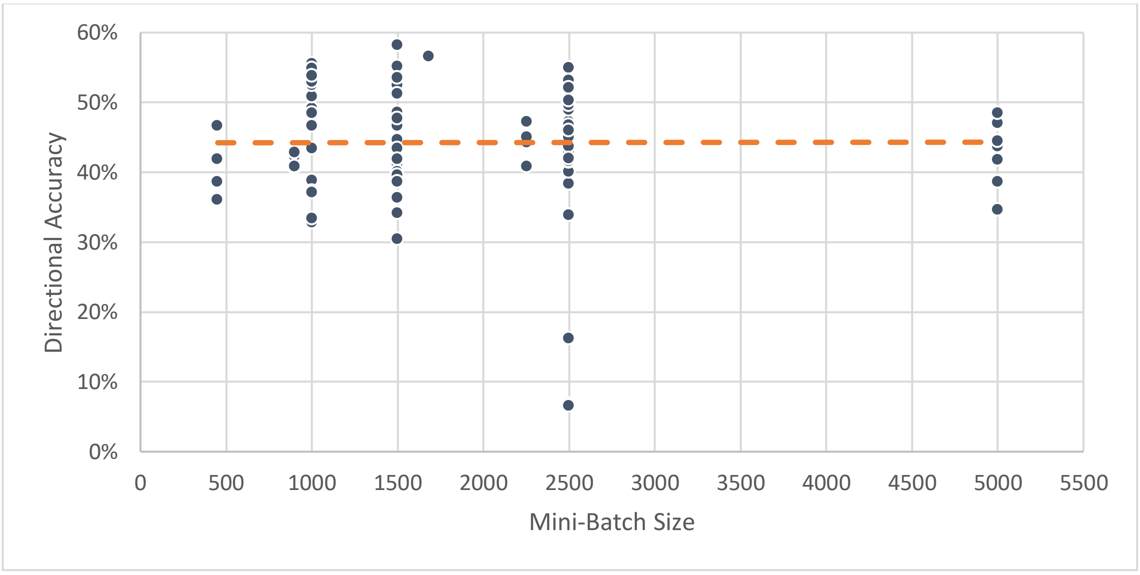


*Figure 21 - Mini-Batch Size General Trend*

### 4.3.3 Number of Hidden Units

Last, but not least, we tested the Long Short-Term Memory layer-specific Hidden Units. This value corresponds to the amount of data the layer can store within its memory cells (MathWorks, 2021). We conceived one dedicated test to observe changes and later compared those to the overall trend.

Our test models were trained with 20-Yearlong INTC (*Intel Corporation*) datasets with 14-Day Windows and split into 14 Classes. The network parameters were set to 500 Epoch and a Mini-Batch Size of 2500. The results below **do not seem to indicate any correlation** to model accuracy.

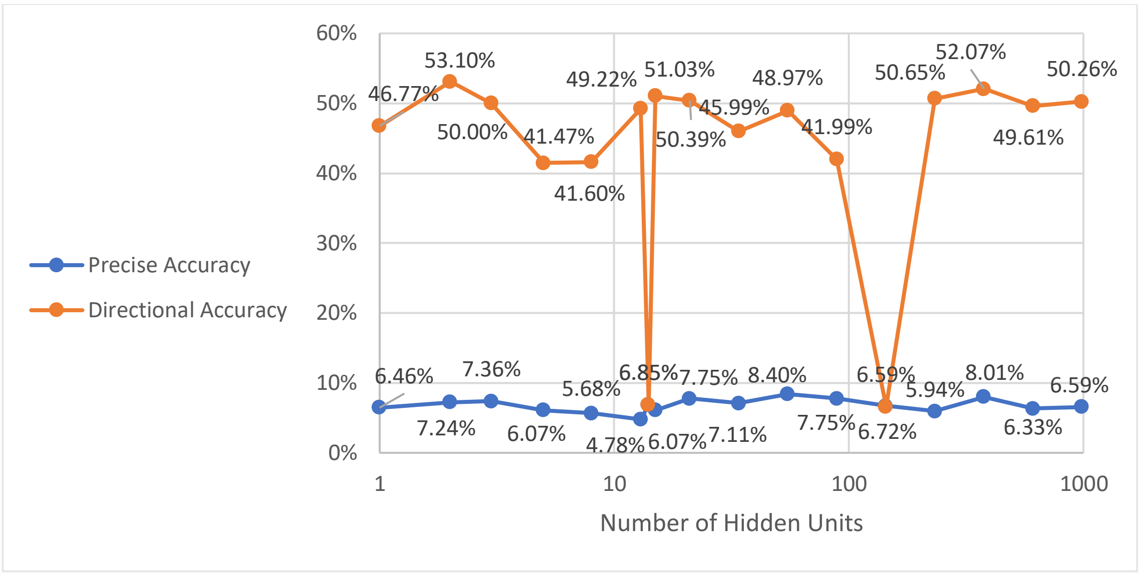


*Figure 22 - № of Hidden Units Dedicated Test*

The General Trend **somewhat conflicts** with our test, since we can see a very minor correlation emerging where **networks with a higher number of hidden units had higher accuracy**.

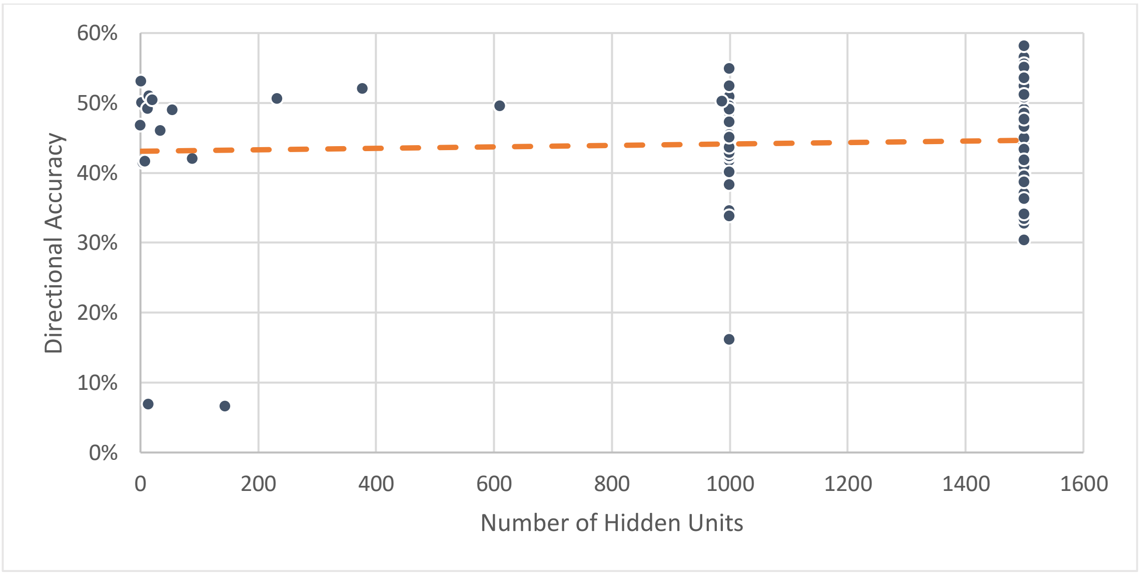


*Figure 23 - № of Hidden Units Overall Trend*

# 5. Summary

## 5.1 Critical Appraisal

By now we have introduced a collection of tests and detailed their outcomes. This is a good point to summarise and bring them all together. We will begin with our investigations into Data Point adjustments, starting with the Window size. While at first, it seemed that the sweet spot for this lies near 10-14 Days, then the overall trend conflicted with this suggesting that the longer the window size, the more accurate the model. As a result, while we recommend using larger **window sizes, it requires further investigation** due to the conflicting results. For the number of symbols, we generally found that including additional market information reduced the performance of our networks. Even though it is quite possible that we did not combine the right symbols or find the right balance for the information; since we did not encounter any disputing results, we **cautiously recommend sticking to just the symbol that the model is designed to predict**.

Moving on to the tests in the Dataset category; the examination of dataset classification proved to be the most intriguing part of our investigations. We managed to confirm that increasing the number of classes improves performance. However, a large-scale comparison conflicted with our previous results and general trend. Since the result could be an outlier or a sign that the correlation does not apply to large models, we suggest a **further examination of the effect at large-scale but recommend using many classes at small-scale**. As for the dataset size, the results confirmed our expectations in that the larger the dataset the more accurate the model. While there is the possibility that our results stemmed from increasing the dataset by time period rather than frequency, then the likelihood of that seems quite slim. Thus, we **advise using as large of a dataset as possible**. When it comes to the different financial markets our preliminary investigation indicates that forecasting Stocks and Cryptocurrencies is more viable than traditional currencies or commodities. However, we **certainly recommend investigating this further**.

The last set of tests could be categorised as Model Training Parameters, under which the Epoch Limit was the first presented. As we found through a test and the general trend, a larger Epoch trains better models. On the other hand, according to our training window, increasing it too far produces a negative result. However, since the ideal value for this seemed to be dependent on a multitude of other factors, we would **suggest a trial-and-error approach for selecting an Epoch**. Next, while our investigations into the Mini-Batch Size did not reveal any correlations to model performance, we did not conduct extensive research either. As a result, we **recommend further investigations to confirm that Mini-Batch Size does not affect performance**. Lastly, we examined the LSTM Layer specific number of Hidden Units. Even though our first findings did not indicate anything, we later found a very minor correlation in our general trend. Hence, **while we recommend further investigation, we would proceed with larger Hidden Unit values** when possible.

## 5.2 Conclusion

Returning to our aims and objectives, we set out to create an Artificial Neural Network designed to predict a financial market in a profitable manner for short-term investment purposes. We broke that aim down into 3 objectives and met those quite early in the project, basing some of the decisions on our wider literature review. This did not just offer a good overview of best practices in the field, but also comparisons of different technologies, leading us to conclude which direction to proceed in. Over time, as we explored through 8 different parameters in the 3 test fields, we learned and improved our techniques.

At this point, the readers who were lured in by the catchy title might be wondering what the conclusion is then. Have we succeeded in predicting a financial market and enabled anyone to make a quick buck? However cliché this might sound, Yes and No. **22 of the developed models (*out of the total 139*) had an accuracy above 50%**, indicating that they can outperform a simple coin flip when it comes to decision-making. Furthermore, **4 out of the 5 best-performing networks were among the last 30 we trained**, indicating that the more time we spent working on it the better we became at it. In addition, while not a very precise indicator, **19 out of the 63 simulated models earned a profit**.

| | *№* | *Precise Accuracy* | *Directional Accuracy* | *Model Earninigs* |
|---|---|---|---|---|
| *Top 5* | 110 | 58,17% | 58,17% | $125,33 |
| | 109 | 4,41% | 56,55% | -$112,55 |
| | 125 | 55,53% | 55,53% | -$475,45 |
| | 128 | 55,14% | 55,14% | $404,21 |
| | 59 | 2,65% | 54,92% | - |
| *Average* | **106,2** | 35,18% | 56,06% | -$14,62 |

*Figure 24 - Top 5 Most Accurate Models (Including the order № of when it was developed)*

If all of these statements begin to sound too promising, then the reality is not quite as rosy. First, we mustn't forget about brokerage fees. While these usually don't make a dent in long-term holdings, frequent trading and especially in day trading, **fees can often outweigh smaller profit margins**. In addition, it would be remiss of us to not include some investment wisdom from Warren Buffet's favourite investment book while concluding a project in connection to financial markets. As Benjamin Graham points out in The Intelligent Investor (2009, p. 167), "*... in stock-market affairs the popularity of* ***a trading theory has itself an influence on the market's behavior*** *which detracts in the long run from its profit-making possibilities.*" While we doubt that the market has reached that point yet, it does remind us of the dilemma that prevents a Neural Network from being the next mass-market get-rich-quick scheme.

## 5.3 Further Work

There are several parameters that could be researched further. In addition to the aspects mentioned in Critical Appraisal that require further investigation, there are elements that we did not investigate at all. When it comes to the Data Point adjustments, we did not quantitatively explore which **trading information components** (*i.e. High Price, Close Price, Trading Volume, etc…*) are more relevant and which are less. Expanding on this, we did not delve too much into **pre-processing or normalising** this data either. As for the Dataset, due to the limitations placed on us by our data source, we focused on creating datasets with a daily frequency. As a result, the effect of **more and less frequent trading information** should be studied. Similarly, we did not explore whether it would be easier and/or more accurate to **predict price changes for the following week/month/quarter/year** instead of the following day. Lastly, regarding the Model Training aspect, as stated earlier, our work focused on a basic LSTM architecture. However, not only would it be a good idea to **investigate the other less**

**studied structures, but furthermore mixing and matching layers** could consequentially improve performance.

## 5.4 Personal Reflections

When the author began the project, he had very limited knowledge both in the world of machine learning and finance. By overcoming various challenges throughout the development of the project, the author became well acquainted with both fields. For example, in the field of deep learning, he went from viewing neural networks as black boxes to understanding the intricate system of layers and nodes. On the investment side, the author learned of various other financial markets that exist alongside the stock market and became aware that the latter is not a singular centralised entity.

Looking back at all of the findings up to this point and all of the areas that could be investigated further, the author would be keen to keep working on this subject. However, unfortunately, projects have deadlines, and a line has to be drawn at some point. That being said, if the opportunity to build a follow-up on this project presented itself, he would gladly accept.

## 6. Glossary

| *Term* | *Definition* |
|---|---|
| **DCNN / CNN** | (Deep) Convolutional Neural Network. |
| **DNN / NN** | (Deep) Neural Network. |
| **ImageNet** | A huge database of images. |
| **Index** | An indicator of the price of a collection of stocks. |
| **LSTM** | Long Short-Term Memory. |
| **MLP** | Multi-Layer Perceptron. |
| **Node / Neuron** | A single cell in a Neural Network structure. |
| **RNN** | Recurrent Neural Network. |
| **Symbol** | Numbers or Characters that refer to a single trade item. |

## 7. List of References

Akita, R., Yoshihara, A., Matsubara, T., & Uehara, K. (2016). Deep Learning for Stock Prediction Using Numerical and Textual Information. *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, 1-6. https://doi.org/10.1109/ICIS.2016.7550882

Aru, P. J. (2020). *Training a Neural Network Capable of Predicting a Financial Market. - A Preparation Progress Report*. 305AAE - Individual Project Preparation.

Billah, M., Waheed, S., & Hanifa, A. (2016). Stock market prediction using an improved training algorithm of neural network. *2016 2nd International Conference on Electrical, Computer & Telecommunication Engineering (ICECTE)*, 1-4. https://doi.org/10.1109/ICECTE.2016.7879611

Chen, K., Zhou, Y., & Dai, F. (2015). A LSTM-based method for stock returns prediction: A case study of China stock market. *2015 IEEE International Conference on Big Data (Big Data)*, 2823-2824. https://doi.org/10.1109/BigData.2015.7364089

Chen, S., & He, H. (2018). Stock Prediction Using Convolutional Neural Network. *2018 2nd International Conference on Artificial Intelligence Applications and Technologies (AIAAT 2018)*, 435, 1-9. https://doi.org/10.1088/1757-899X/435/1/012026

Chong, E., Han, C., & Park, F. C. (2017). Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications*, 83, 187-205. https://doi.org/10.1016/j.eswa.2017.04.030

Deng, Y., Bao, F., Kong, Y., Ren, Z., & Dai, Q. (2017). Deep Direct Reinforcement Learning for Financial Signal Representation and Trading. *IEEE Transactions on Neural Networks and Learning Systems*, 28(3), 653-664. https://doi.org/10.1109/TNNLS.2016.2522401

Google Ngram. (2020). *deep learning,machine learning,artificial intelligence*. https://books.google.com/ngrams/graph?content=deep+learning%2Cmachine+learning%2Cartificial+intelligence&year_start=1969&year_end=2019&case_insensitive=on&corpus=26&smoothing=0&direct_url=t4%3B%2Cdeep%20learning%3B%2Cc0%3B%2Cs0%3B%3Bdeep%20learning%3B%2Cc0%3B%3BDeep%20Learning%3B%2Cc0%3B%3BDeep%20learning%3B%2Cc0%3B%3BDEEP%20LEARNING%3B%2Cc0%3B.t4%3B%2Cmachine%20learning%3B%2Cc0%3B%2Cs0%3B%3Bmachine%20learning%3B%2Cc0%3B%3BMachine%20Learning%3B%2Cc0%3B%3BMachine%20learning%3B%2Cc0%3B%3BMACHINE%20LEARNING%3B%2Cc0%3B.t4%3B%2Cartificial%20intelligence%3B%2Cc0%3B%2Cs0%3B%3BArtificial%20Intelligence%3B%2Cc0%3B%3Bartificial%20intelligence%3B%2Cc0%3B%3BArtificial%20intelligence%3B%2Cc0%3B%3BARTIFICIAL%20INTELLIGENCE%3B%2Cc0

Google Trends. (2020). *stock market,stocks,shares*. https://trends.google.com/trends/explore?date=all&q=stock%20market,stocks,shares

Graham, B., Zweig, J. (2009). *The Intelligent Investor* (Rev. ed.). HarperCollins e-books.

Guresen, E., Kayakutlu, G., & Daim, T. U. (2011). Using artificial neural network models in stock market index prediction. *Expert Systems with Applications*, 38(8), 10389-10397. https://doi.org/10.1016/j.eswa.2011.02.068

Guo, J., & Tuckfield, B. (2020). News-based Machine Learning and Deep Learning Methods for Stock Prediction. *4th International Conference on Artificial Intelligence Applications and Technologies (AIAAT 2020)*, 1642, 1-7. https://doi.org/10.1088/1742-6596/1642/1/012014

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet Classification with Deep Convolutional Neural Networks. *Communications of the ACM*, 60(6), 84–90. https://doi.org/10.1145/3065386

Li, G., Xiao, M., & Guo, Y. (2019). Application of Deep Learning in Stock Market Valuation Index Forecasting. *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*, 551-554. https://doi.org/10.1109/ICSESS47205.2019.9040833

Li, L., Xu, S., Liu, Y., & Yang, X. (2020). Predicting the Rise and Fall of Stock Prices based on the modified BP_AdaBoost. *2020 4th International Conference on Machine Vision and Information Technology (CMVIT 2020)*, 1518, 1-6. https://doi.org/10.1088/1742-6596/1518/1/012060

Loayza, R., Vidal V., & Murillo, M. (2019). Application of artificial intelligence in the development of a model of Neural Network for the financial forecast in the Stock Exchange of Lima. *2019 IEEE Sciences and Humanities International Research Conference (SHIRCON)*, 1-4. https://doi.org/10.1109/SHIRCON48091.2019.9024868

MathWorks. (2021). *Help Center - trainingOptions.* https://uk.mathworks.com/help/deeplearning/ref/trainingoptions.html

Moghaddam, A. H., Moghaddam, M. H., & Esfandyari M. (2016). Stock market index prediction using artificial neural network. *Journal of Economics, Finance and Administrative Science*, 21(41), 89-93. https://doi.org/10.1016/j.jefas.2016.07.002

Naeini, M. P, Taremian, H., & Hashemi, H. B. (2010). Stock market value prediction using neural networks. *2010 International Conference on Computer Information Systems and Industrial Management Applications (CISIM)*, 132-136. https://doi.org/10.1109/CISIM.2010.5643675

Pang, X., Zhou, Y., Wang, P., Lin, W., & Chang, V. (2018). An innovative neural network approach for stock market prediction. *J Supercomput*, 76, 2098-2118. https://doi.org/10.1007/s11227-017-2228-y

Pedrozo, D., Barajas, F., Estupiñán, A., Cristiano, K. L., & Triana D. A. (2019). Development and implementation of a predictive method for the stock market analysis, using the long short-term memory machine learning method. *VI International Conference Days of Applied Mathematics (VI ICDAM)*, 1514, 1-6. https://doi.org/10.1088/1742-6596/1514/1/012009

Sachdeva, A., Jethwani, G., Manjunath, C., Balamurugan, M., & Krishna, A. V. N. (2019). An Effective Time Series Analysis for Equity Market Prediction Using Deep Learning Model. *2019 International Conference on Data Science and Communication (IconDSC)*, 1-5. https://doi.org/10.1109/IconDSC.2019.8817035

Srinivasan, N., & Lakshmi, C. (2017). Stock prediction and analysis using intermittent training data with artificial neural networks. *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, 1-4. https://doi.org/10.1109/ICIIECS.2017.8276082

Sujatha, K. V., Sundaram, S. M. (2010). Stock index prediction using regression and neural network models under non normal conditions. *INTERACT-2010*, 59-63. https://doi.org/10.1109/INTERACT.2010.5706195

Ticknor, J. L. (2013). A Bayesian regularized artificial neural network for stock market forecasting. *Expert Systems with Applications*, 40(14), 5501-5506. https://doi.org/10.1016/j.eswa.2013.04.013

Yousif, A., & Elfaki, F. (2017). Artificial Neural Network versus Linear Models Forecasting Doha Stock Market. *4th International Conference on Mathematical Applications in Engineering 2017 (ICMAE'17)*, 949, 1-6. https://doi.org/10.1088/1742-6596/949/1/012014

# 8. Appendices

## 8.1 Literature Table

| Date | Researchers | Title | Approach(es) Used | Input Data | | | | Output Type | Performance Checking Approach |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Type | Time Period | Frequency | Sum of Samples | | |
| 2016 | Ryo Akita,<br>Akira Yoshihara,<br>Takashi Matsubara,<br>Kuniaki Uehara | Deep Learning for Stock Prediction Using Numerical and Textual Information | **LSTM**,<br>Support Vector Regression,<br>MLP,<br>RNN | **Nikkei Newspaper Articles**<br>**+ Price Data**<br>10 Companies from Nikkei 225 | **7 years**<br>(2001-2006 Training,<br>2007 Validation,<br>2008 Testing) | 1-Day | ≈2555 | Day Close | Market Simulation (Largest Profits) |
| 2016 | Yue Deng,<br>Feng Bao,<br>Youyong Kong,<br>Zhiquan Ren,<br>Qionghai Dai | Deep Direct Reinforcement Learning for Financial Signal Representation and Trading | **Fuzzy Deep Direct Reinforcement**,<br>LSTM,<br>RNN,<br>DCN | 1 Stock Index /<br>1 Commodity Future | **10/20 Days**<br>(Latest 15000 Ticks)<br>**+ 8 years**<br>(Latest 2000 Days for Training) | 1-Minute<br>+ 1-Day | 17000 | Next Tick Price | Market Simulation (Largest Profits) |
| 2015 | Kai Chen,<br>Yi Zhou,<br>Fangyan Dai | A LSTM-based method for stock returns prediction: A case study of China stock market | **LSTM** | **High, Low, Open, Close, Volume**<br>Companies from Shanghai and Shenzhen | **2 years**<br>(1/7/2013-12/11/2014 Training,<br>12/11/2014-31/5/2015 Validation) | 1-Day | 1211361 | - | Accuracy Calculation (Prediction Difference From Actual) |
| 2010 | Mahdi Pakdaman Naeini,<br>Hamidreza Taremian,<br>Homa Baradaran Hashemi | Stock Market Value Prediction Using Neural Networks | **MLP**,<br>Elman RNN | **High, Low, Average of past 10 days**<br>1094 Companies from Tehran SM | **5 years** | 1-Day | 1825 | Next Day Price | Mean Absolute Deviation,<br>Mean Absolute Percentage Error,<br>Mean Squared Error,<br>Root Mean Squared Error |
| 2010 | K. V. Sujatha,<br>S. Meenakshi Sundaram | Stock Index Prediction Using Regression and Neural Network Models under Non Normal Conditions | **MLP** | **Close**<br>Stock Index (BSE Sensex) | **88 days**<br>(2/1/2010-31/3/2010) | 1-Day | 300 | Day Close | Mean Absolute Percentage Error |
| 2019 | Ge Li,<br>Ming Xiao,<br>Ying Guo | Application of Deep Learning in Stock Market Valuation Index Forecasting | **LSTM** | **P/E Ratio**<br>Stock Index<br>(China's Growth Enterprise Market) | **10 years**<br>(1/10/2009-2/7/2018 Training,<br>3/7/2018-28/6/2019 Testing) | 1-Day | 3546 | P/E Ratio | Forecast Trend Accuracy,<br>Average Forecast Deviation Rate,<br>Root Mean Squared Error |
| 2016 | Mustain Billah,<br>Sajjad Waheed,<br>Abu Hanifa | Stock Market Prediction Using an Improved Training Algorithm of Neural Network | **Levenberg-Marquardt Neural Network,**<br>Adaptive Neuro Fuzzy Inference System | **High, Low, Open, Close, Volume**<br>Individual Stock<br>(Grameenphone GP) | **2 years**<br>(2013-2015) | 1-Day | 730 | - | Root Mean Squared Error,<br>Coefficient of Multiple Determinations ($R^2$) |
| 2017 | N. Srinivasan,<br>C. Lakshmi | Stock Prediction and Analysis Using Intermittent Training Data With Artificial Neural Networks | Neural Network | Individual Stocks<br>(Axis Bank, Mahindra, Hindalco, Ramco) | - | - | - | Month High | Accuracy Calculation (Prediction Difference From Actual) |
| 2019 | Akshay Sachdeva,<br>Geet Jethwani,<br>Chinthakunta Manjunath,<br>Balamurugan M,<br>Adapalli V N Krishna | An Effective Time Series Analysis for Equity Market Prediction Using Deep Learning Model | **LSTM** | **High, Low, Open, Close, Date**<br>**60 Previous Days**<br>1 Stock (Infosys Ltd) &<br>1 Index (NSE NIFTY 50) | **11 years**<br>(11/12/2007-11/12/2017 Training,<br>12/12/2017-12/12/2018 Testing) | 1-Day | 4015 | Next Day Price | Mean Absolute Deviation,<br>Mean Absolute Percentage Error,<br>Mean Squared Error |
| 2019 | Raul Loayza,<br>Victor Vidal,<br>Margarita Murillo | Application of Artificial Intelligence in the development of a model of Neural Network for the financial forecast in the Stock Exchange of Lima | Backpropagating Neural Network | Individual Stocks<br>(Southern Peru Copper Corporation, Credicorp Ltd, Telefonica del Peru S.A.A., Compania de Minas Buenaventura S.A.A., Yahoo Inc, Peru Amazon Group S.A.C.) | - | - | 2020 | - | - |

| Date | Researchers | Title | Approach(es) Used | Input Data | | | | Output Type | Performance Checking Approach |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Type | Time Period | Frequency | Sum of Samples | | |
| 2020 | D Pedrozo, F Barajas, A Estupiñán, K L Cristiano, D A Triana | Development and implementation of a predictive method for the stock market analysis, using the Long Short-Term Memory Machine Learning method | **LSTM** | Forex Currencies (EUR/USD, GBP/USD, USD/JPY, USD/CHF, AUD/USD, USD/CAD, EUR/GBP, EUR/CHF, EUR/JPY, GBP/CHF, GBP/JPY, EUR/CAD) | **1 year** (2019) | 1-Minute | 525600 | Next Tick Price | Accuracy Calculation (Prediction Difference From Actual) |
| 2018 | Sheng Chen, Hongxiang He | Stock Prediction Using Convolutional Neural Network | **DCN** | **High, Low, Open, Close, Volume** Stocks from Chinese Stock Market | **≈3 years** (5/1/2015-29/12/2017, 80% Training, 20% Testing) | 5-Minute | 313632 | Rise/Fall After 10 Days | Accuracy Calculation (Prediction Difference From Actual) |
| 2020 | Junjie Guo, Bradford Tuckfield | News-based Machine Learning and Deep Learning Methods for Stock Prediction | **LSTM**, DCN | **High, Low, Open, Close, Volume** Stock Indexes (DJIA, S&P500), Individual Stocks (IBM, JPM) **News**: 25 Daily Headlines | **≈7 years** (8/8/2008-22/7/2015, 1400d Training, 350d Testing) | 1-Day | 1750 | - | Accuracy Calculation (Prediction Difference From Actual) |
| 2017 | Adil Yousif, Faiz Elfaki | Artificial Neural Network versus Linear Models Forecasting Doha Stock Market | **MLP** | **Close, Date** Qatar General Closing Index | **8 years** (2007-2014 Training, 2015 Validation) | 1-Day | 2920 | - | Accuracy Calculation (Prediction Difference From Actual) |
| 2020 | Lu Li, Shizhan Xu, Yanhong Liu, Xiaorong Yang | Predicting the Rise and Fall of Stock Prices based on the modified BP_AdaBoost | Backpropagating Neural Network | **Rates of Return, Ranges of Prices, Volatility, Amplitude, Total Value, Circulation Market Value, Valuation Factors, Growth Factors, Turnover Rate, Ten Day Closing Price** Stock Index (CSI800) | **≈6 years** (4/1/2010-31/12/2015, 70% Training, 15% Validation, 15% Testing) | - | - | Rise/Fall Monthly Average | Accuracy Calculation, Root Mean Squared Error, Average Monthly Closing Price, F1 |
| 2011 | Erkam Guresen, Gulgun Kayakutlu, Tugrul U. Daim | Using Artificial Neural Network models in Stock Market Index Prediction | **MLP**, Dynamic Artificial Neural Network, Generalised AutoRegressive Conditional Heteroscedasticity, | **4 Previous Days** Stock Index (NASDAQ) | **≈8 months** (7/10/2008-26/6/2009, 146d Training, 36d Testing) | 1-Day | 262 | Next Day Price | Mean Squared Error, Mean Absolute Deviate |
| 2013 | Jonathan L. Ticknor | A Bayesian Regularized Artificial Neural Network for Stock Market Forecasting | **Bayesian Levenberg-Marquardt Neural Network** | **High, Low, Open, Exponential Moving Average, Relative Strength Index, Williams R%, Stochastic K%, Stochastic D%** Individual Stocks (Microsoft, Goldman Sachs) | **≈3 years** (4/1/2010-31/12/2012, 80% Training, 20% Testing) | 1-Day | 734 | Next Day Closing Price | Mean Absolute Percentage Error |
| 2015 | Amin Hedayati Moghaddam, Moein Hedayati Moghaddam, Morteza Esfandyari | Stock Market Index Prediction Using Artificial Neural Network | **MLP** | **4/9 Previous Days Historic Price, Day of Week** Stock Index (NASDAQ) | **≈5 months** (28/1/2015-18/6/2015, 70d Training, 29d Testing) | 1-Day | 141 | - | Mean Squared Error, Determination Coefficient ($R^2$) |
| 2016 | Eunsuk Chong, Chulwoo Han, Frank C. Park | Deep Learning networks for Stock Market analysis and prediction: Methodology, Data Representations, and Case Studies | AutoEncoder, Restricted Boltzmann Machine | Individual Stocks (38 from KOSPI) | **≈5 years** (4/1/2010-30/12/2014, 80% Training, 20% Testing and Validation) | 5-Minute | 73041 | | Normalised Mean Squared Error, Root Mean Squared Error, Mean Absolute Error, Mutual Information |
| 2018 | Xiongwen Pang, Yanqiang Zhou, Pan Wang, Weiwei Lin, Victor Chang | An innovative neural network approach for stock market prediciton | LSTM with Automatic Encoder, **LSTM with Embedded Stock Vector Layer**, MLP | **High, Low, Open, Close, Volume, Amplitudes** Individual Stocks **News** | **≈10 years** (1/1/2006-19/10/2016, 70% Training, 20% Testing, 10% Validation) | - | 7902686 | Next Price | Accuracy Calculation (Prediction Difference From Actual) |

Literature_Table.xlsx

## 8.2 Complete Test Results Table

| № | Precise Accuracy | Directional Accuracy | Model Earnings | Window Size | Data Period | Classes | Epoch | Hidden Units | Mini Batch Size | Symbols |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 34,57% | 34,57% | | 5 | 20 | 3 | 200 | 1000 | 5000 | INTC |
| 2 | 18,00% | 38,63% | | 5 | 20 | 5 | 200 | 1000 | 5000 | INTC |
| 3 | 11,68% | 41,73% | | 5 | 20 | 7 | 200 | 1000 | 5000 | INTC |
| 4 | 10,44% | 43,73% | | 5 | 20 | 9 | 200 | 1000 | 5000 | INTC |
| 5 | 8,87% | 46,94% | | 5 | 20 | 11 | 200 | 1000 | 5000 | INTC |
| 6 | 7,94% | 46,88% | | 5 | 20 | 13 | 200 | 1000 | 5000 | INTC |
| 7 | 5,56% | 47,67% | | 5 | 20 | 15 | 200 | 1000 | 5000 | INTC |
| 8 | 4,56% | 44,46% | | 5 | 20 | 17 | 200 | 1000 | 5000 | INTC |
| 9 | 5,15% | 47,10% | | 5 | 20 | 19 | 200 | 1000 | 5000 | INTC |
| 10 | 5,01% | 46,98% | | 5 | 20 | 21 | 200 | 1000 | 5000 | INTC |
| 11 | 3,79% | 48,42% | | 5 | 20 | 31 | 200 | 1000 | 5000 | INTC |
| 12 | 9,38% | 47,79% | | 1 | 20 | 11 | 500 | 1000 | 2500 | INTC |
| 13 | 7,03% | 42,32% | | 2 | 20 | 11 | 500 | 1000 | 2500 | INTC |
| 14 | 9,64% | 44,66% | | 3 | 20 | 11 | 500 | 1000 | 2500 | INTC |
| 15 | 8,34% | 44,20% | | 5 | 20 | 11 | 500 | 1000 | 2500 | INTC |
| 16 | 8,87% | 44,20% | | 7 | 20 | 11 | 500 | 1000 | 2500 | INTC |
| 17 | 9,39% | 45,89% | | 10 | 20 | 11 | 500 | 1000 | 2500 | INTC |
| 18 | 10,44% | 48,04% | | 14 | 20 | 11 | 500 | 1000 | 2500 | INTC |
| 19 | 8,09% | 45,69% | | 15 | 20 | 11 | 500 | 1000 | 2500 | INTC |
| 20 | 7,19% | 45,62% | | 20 | 20 | 11 | 500 | 1000 | 2500 | INTC |
| 21 | 9,03% | 45,29% | | 25 | 20 | 11 | 500 | 1000 | 2500 | INTC |
| 22 | 8,77% | 47,38% | | 30 | 20 | 11 | 500 | 1000 | 2500 | INTC |
| 23 | 9,39% | 47,85% | | 1 | 20 | 11 | 500 | 1000 | 2500 | INTC |
| 24 | 9,13% | 44,33% | | 1 | 20 | 11 | 500 | 1000 | 2500 | INTC,^GSPC |
| 25 | 9,91% | 45,63% | | 1 | 20 | 11 | 500 | 1000 | 2500 | INTC,^GSPC, AMD |
| 26 | 7,56% | 43,29% | | 1 | 20 | 11 | 500 | 1000 | 2500 | INTC,^GSPC, AMD, ^DJI |
| 27 | 10,43% | 44,72% | | 1 | 20 | 11 | 500 | 1000 | 2500 | INTC,^GSPC, AMD, ^DJI, NVDA |
| 28 | 9,52% | 45,11% | | 1 | 20 | 11 | 500 | 1000 | 2500 | INTC,^GSPC, AMD, ^DJI, NVDA, ^IXIC |
| 29 | 10,17% | 38,59% | | 1 | 20 | 11 | 500 | 1000 | 2500 | INTC,^GSPC, AMD, ^DJI, NVDA, ^IXIC, HPQ |
| 30 | 11,34% | 16,17% | | 1 | 20 | 11 | 500 | 1000 | 2500 | INTC,^GSPC, AMD, ^DJI, NVDA, ^IXIC, HPQ, LN |
| 31 | 9,13% | 42,37% | | 1 | 20 | 11 | 500 | 1000 | 2500 | INTC,^GSPC, AMD, ^DJI, NVDA, ^IXIC, HPQ, LN |
| 32 | 11,19% | 44,20% | | 1 | 20 | 11 | 500 | 1000 | 2500 | INTC,^GSPC, AMD, ^DJI, NVDA, ^IXIC, HPQ, LN |
| 33 | 33,86% | 33,86% | | 11 | 20 | 3 | 500 | 1000 | 2500 | INTC |
| 34 | 21,34% | 38,34% | | 11 | 20 | 5 | 500 | 1000 | 2500 | INTC |
| 35 | 14,47% | 43,68% | | 11 | 20 | 7 | 500 | 1000 | 2500 | INTC |
| 36 | 10,16% | 45,31% | | 11 | 20 | 9 | 500 | 1000 | 2500 | INTC |
| 37 | 7,84% | 44,97% | | 11 | 20 | 11 | 500 | 1000 | 2500 | INTC |
| 38 | 6,91% | 42,89% | | 11 | 20 | 13 | 500 | 1000 | 2500 | INTC |
| 39 | 7,37% | 44,50% | | 11 | 20 | 15 | 500 | 1000 | 2500 | INTC |
| 40 | 5,69% | 47,35% | | 11 | 20 | 19 | 500 | 1000 | 2500 | INTC |
| 41 | 4,88% | 48,84% | | 11 | 20 | 21 | 500 | 1000 | 2500 | INTC |
| 42 | 1,78% | 50,89% | | 11 | 20 | 31 | 500 | 1000 | 2500 | INTC |
| 43 | 2,54% | 44,91% | | 11 | 20 | 35 | 500 | 1000 | 2500 | INTC |
| 44 | 3,71% | 45,24% | | 11 | 20 | 41 | 500 | 1000 | 2500 | INTC |
| 45 | 33,86% | 33,86% | | 11 | 20 | 2 | 500 | 1000 | 2500 | INTC |
| 46 | 21,34% | 38,34% | | 11 | 20 | 4 | 500 | 1000 | 2500 | INTC |
| 47 | 14,47% | 43,68% | | 11 | 20 | 6 | 500 | 1000 | 2500 | INTC |
| 48 | 10,16% | 45,31% | | 11 | 20 | 8 | 500 | 1000 | 2500 | INTC |
| 49 | 9,92% | 40,08% | | 11 | 20 | 10 | 500 | 1000 | 2500 | INTC |
| 50 | 8,50% | 48,63% | | 11 | 20 | 12 | 500 | 1000 | 2500 | INTC |

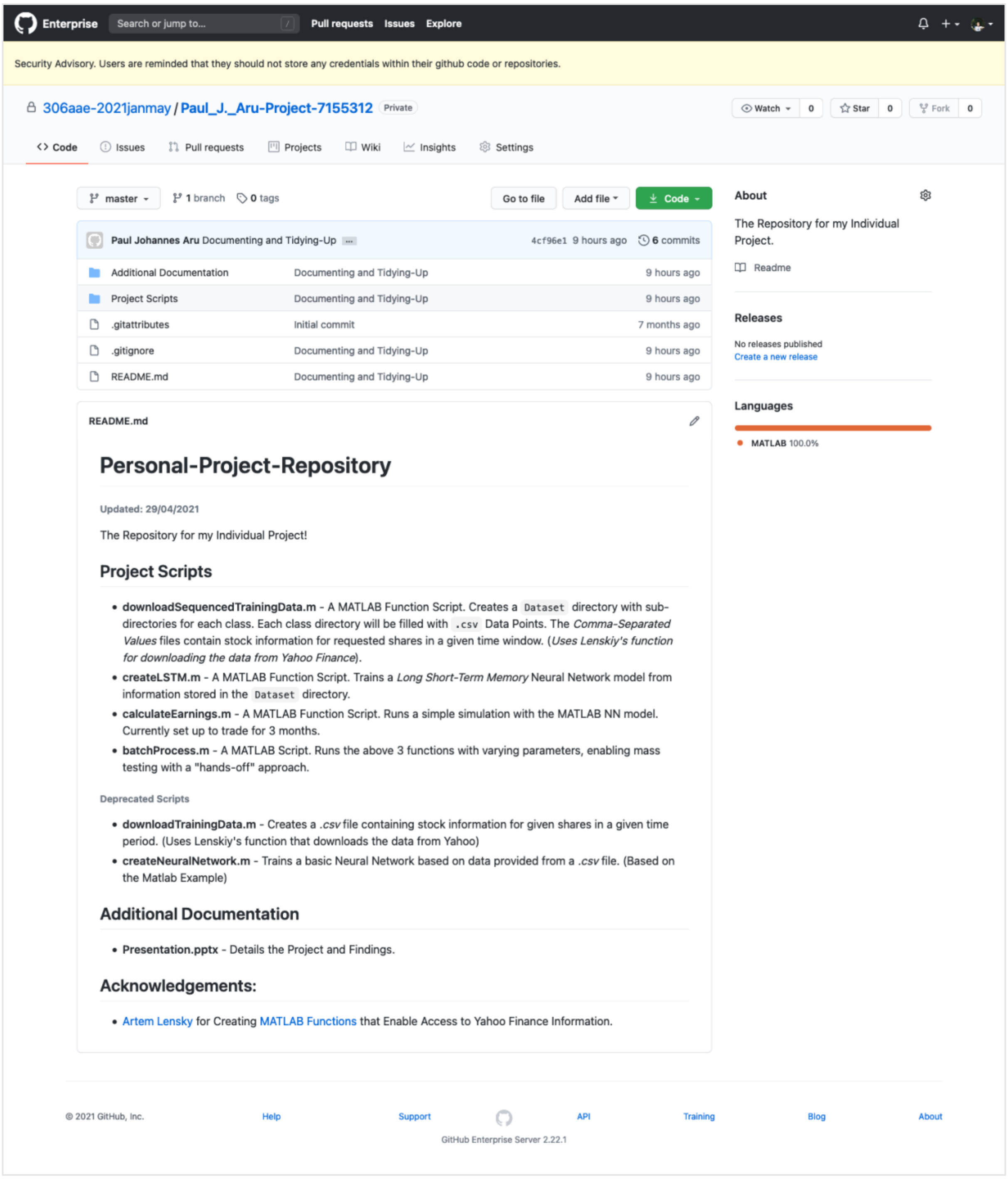
| № | Precise Accuracy | Directional Accuracy | Model Earnings | Window Size | Data Period | Classes | Epoch | Hidden Units | Mini Batch Size | Symbols |
|---|---|---|---|---|---|---|---|---|---|---|
| 51 | 7,49% | 52,45% | | 11 | 20 | 14 | 500 | 1000 | 2500 | INTC |
| 52 | 6,62% | 47,14% | | 11 | 20 | 16 | 500 | 1000 | 2500 | INTC |
| 53 | 4,02% | 46,89% | | 11 | 20 | 18 | 500 | 1000 | 2500 | INTC |
| 54 | 4,77% | 45,42% | | 11 | 20 | 20 | 500 | 1000 | 2500 | INTC |
| 55 | 2,83% | 47,23% | | 11 | 20 | 24 | 500 | 1000 | 2500 | INTC |
| 56 | 3,68% | 49,56% | | 11 | 20 | 28 | 500 | 1000 | 2500 | INTC |
| 57 | 3,04% | 49,05% | | 11 | 20 | 32 | 500 | 1000 | 2500 | INTC |
| 58 | 2,64% | 45,03% | | 11 | 20 | 36 | 500 | 1000 | 2500 | INTC |
| 59 | 2,65% | 54,92% | | 11 | 20 | 40 | 500 | 1000 | 2500 | INTC |
| 60 | 6,46% | 46,77% | | 14 | 20 | 14 | 500 | 1 | 2500 | INTC |
| 61 | 7,24% | 53,10% | | 14 | 20 | 14 | 500 | 2 | 2500 | INTC |
| 62 | 7,36% | 50,00% | | 14 | 20 | 14 | 500 | 3 | 2500 | INTC |
| 63 | 6,07% | 41,47% | | 14 | 20 | 14 | 500 | 5 | 2500 | INTC |
| 64 | 5,68% | 41,60% | | 14 | 20 | 14 | 500 | 8 | 2500 | INTC |
| 65 | 4,78% | 49,22% | | 14 | 20 | 14 | 500 | 13 | 2500 | INTC |
| 66 | 6,85% | 6,85% | | 14 | 20 | 14 | 500 | 14 | 2500 | INTC |
| 67 | 6,07% | 51,03% | | 14 | 20 | 14 | 500 | 15 | 2500 | INTC |
| 68 | 7,75% | 50,39% | | 14 | 20 | 14 | 500 | 21 | 2500 | INTC |
| 69 | 7,11% | 45,99% | | 14 | 20 | 14 | 500 | 34 | 2500 | INTC |
| 70 | 8,40% | 48,97% | | 14 | 20 | 14 | 500 | 55 | 2500 | INTC |
| 71 | 7,75% | 41,99% | | 14 | 20 | 14 | 500 | 89 | 2500 | INTC |
| 72 | 6,72% | 6,59% | | 14 | 20 | 14 | 500 | 144 | 2500 | INTC |
| 73 | 5,94% | 50,65% | | 14 | 20 | 14 | 500 | 233 | 2500 | INTC |
| 74 | 8,01% | 52,07% | | 14 | 20 | 14 | 500 | 377 | 2500 | INTC |
| 75 | 6,33% | 49,61% | | 14 | 20 | 14 | 500 | 610 | 2500 | INTC |
| 76 | 6,59% | 50,26% | | 14 | 20 | 14 | 500 | 987 | 2500 | INTC |
| 77 | 21,80% | 43,60% | -$241,10 | 10 | 20 | 7 | *1* | 1500 | 1500 | GC=F |
| 78 | 24,20% | 42,60% | -$241,10 | 10 | 20 | 7 | *2* | 1500 | 1500 | GC=F |
| 79 | 25,00% | 42,60% | -$637,30 | 10 | 20 | 7 | *3* | 1500 | 1500 | GC=F |
| 80 | 21,40% | 42,20% | -$241,10 | 10 | 20 | 7 | *5* | 1500 | 1500 | GC=F |
| 81 | 22,00% | 42,80% | -$241,10 | 10 | 20 | 7 | *8* | 1500 | 1500 | GC=F |
| 82 | 20,60% | 42,60% | -$904,30 | 10 | 20 | 7 | *13* | 1500 | 1500 | GC=F |
| 83 | 19,00% | 41,00% | -$637,30 | 10 | 20 | 7 | *21* | 1500 | 1500 | GC=F |
| 84 | 25,00% | 40,80% | -$241,10 | 10 | 20 | 7 | *34* | 1500 | 1500 | GC=F |
| 85 | 21,20% | 40,40% | -$637,30 | 10 | 20 | 7 | *55* | 1500 | 1500 | GC=F |
| 86 | 25,40% | 47,20% | -$223,60 | 10 | 20 | 7 | *89* | 1500 | 1500 | GC=F |
| 87 | 22,40% | 46,80% | $26,30 | 10 | 20 | 7 | *144* | 1500 | 1500 | GC=F |
| 88 | 25,90% | 46,61% | $21,10 | 10 | 20 | 7 | *233* | 1500 | 1500 | GC=F |
| 89 | 21,51% | 41,24% | $620,20 | 10 | 20 | 7 | *377* | 1500 | 1500 | GC=F |
| 90 | 25,30% | 42,23% | -$54,30 | 10 | 20 | 7 | *610* | 1500 | 1500 | GC=F |
| 91 | 21,51% | 44,62% | -$325,70 | 10 | 20 | 7 | *987* | 1500 | 1500 | GC=F |
| 92 | 21,51% | 42,03% | -$126,90 | 10 | 20 | 7 | *1597* | 1500 | 1500 | GC=F |
| 93 | 24,10% | 43,03% | $408,10 | 10 | 20 | 7 | *2584* | 1500 | 1500 | GC=F |
| 94 | 20,72% | 42,43% | $620,00 | 10 | 20 | 7 | *4181* | 1500 | 1500 | GC=F |
| 95 | 19,51% | 43,43% | -$214,90 | 10 | 20 | 7 | *6765* | 1500 | 1500 | GC=F |
| 96 | 19,72% | 40,04% | -$212,90 | 10 | 20 | 7 | *10946* | 1500 | 1500 | GC=F |
| 97 | 20,52% | 38,65% | -$1 118,70 | 10 | 20 | 7 | *1* | 1500 | *451* | GC=F |
| 98 | 17,73% | 42,23% | -$625,00 | 10 | 20 | 7 | *1* | 1500 | *902* | GC=F |
| 99 | 22,51% | 40,84% | -$625,00 | 10 | 20 | 7 | *1* | 1500 | *2255* | GC=F |
| 100 | 21,71% | 46,61% | -$625,00 | 10 | 20 | 7 | *2* | 1500 | *451* | GC=F |

| № | Precise Accuracy | Directional Accuracy | Model Earnings | Window Size | Data Period | Classes | Epoch | Hidden Units | Mini Batch Size | Symbols |
|---|---|---|---|---|---|---|---|---|---|---|
| 101 | 24,10% | 42,83% | -$212,90 | 10 | 20 | 7 | *2* | 1500 | *902* | GC=F |
| 102 | 20,72% | 44,22% | -$212,90 | 10 | 20 | 7 | *2* | 1500 | *2255* | GC=F |
| 103 | 21,12% | 36,06% | -$625,00 | 10 | 20 | 7 | *5* | 1500 | *451* | GC=F |
| 104 | 22,31% | 40,64% | -$625,00 | 10 | 20 | 7 | *5* | 1500 | *902* | GC=F |
| 105 | 25,90% | 45,02% | -$625,00 | 10 | 20 | 7 | *5* | 1500 | *2255* | GC=F |
| 106 | 20,52% | 41,83% | -$625,00 | 10 | 20 | 7 | *10* | 1500 | *451* | GC=F |
| 107 | 23,51% | 40,84% | -$625,00 | 10 | 20 | 7 | *10* | 1500 | *902* | GC=F |
| 108 | 23,11% | 47,21% | -$780,00 | 10 | 20 | 7 | *10* | 1500 | *2255* | GC=F |
| 109 | 4,41% | 56,55% | -$112,55 | 30 | 30 | 40 | 25000 | 1500 | 1683 | COKE,^GSPC |
| 110 | 58,17% | 58,17% | $125,33 | 30 | 50 | 3 | 5000 | 1500 | 1500 | MCD,^GSPC |
| 111 | 6,85% | 53,39% | -$149,13 | 30 | 50 | 49 | 5000 | 1500 | 1500 | MCD,^GSPC |
| 112 | 49,16% | 49,08% | $233,67 | 5 | 50 | 3 | 5000 | 1500 | 1000 | IBM,^GSPC |
| 113 | 46,79% | 46,79% | $6,45 | 50 | 50 | 3 | 5000 | 1500 | 1000 | IBM,^GSPC |
| 114 | 48,45% | 48,45% | $283,25 | 3 | 50 | 3 | 5000 | 1500 | 1000 | IBM,^GSPC |
| 115 | 46,68% | 46,60% | $167,03 | 30 | 50 | 3 | 5000 | 1500 | 1000 | IBM,^GSPC |
| 116 | 50,84% | 50,84% | $277,88 | 10 | 50 | 3 | 5000 | 1500 | 1000 | IBM,^GSPC |
| 117 | 32,77% | 32,77% | -$124,84 | 10 | *5* | 3 | 5000 | 1500 | 1000 | PG,^GSPC |
| 118 | 38,78% | 38,78% | -$240,48 | 10 | *10* | 3 | 5000 | 1500 | 1000 | PG,^GSPC |
| 119 | 33,42% | 33,42% | -$233,04 | 10 | *15* | 3 | 5000 | 1500 | 1000 | PG,^GSPC |
| 120 | 37,27% | 37,27% | -$37,54 | 10 | *20* | 3 | 5000 | 1500 | 1000 | PG,^GSPC |
| 121 | 37,08% | 37,08% | -$271,51 | 10 | *25* | 3 | 5000 | 1500 | 1000 | PG,^GSPC |
| 122 | 43,35% | 43,35% | $344,57 | 10 | *30* | 3 | 5000 | 1500 | 1000 | PG,^GSPC |
| 123 | 52,45% | 52,45% | -$425,92 | 10 | *35* | 3 | 5000 | 1500 | 1000 | PG,^GSPC |
| 124 | 52,94% | 52,94% | $344,57 | 10 | *40* | 3 | 5000 | 1500 | 1000 | PG,^GSPC |
| 125 | 55,53% | 55,53% | -$475,45 | 10 | *45* | 3 | 5000 | 1500 | 1000 | PG,^GSPC |
| 126 | 54,94% | 54,86% | -$519,49 | 10 | *50* | 3 | 5000 | 1500 | 1000 | PG,^GSPC |
| 127 | 53,78% | 53,78% | -$496,02 | 10 | *55* | 3 | 5000 | 1500 | 1000 | PG,^GSPC |
| 128 | 55,14% | 55,14% | $404,21 | 14 | 35 | 3 | 5000 | 1500 | 1500 | AAPL, MSFT |
| 129 | 48,55% | 48,55% | $544,27 | 14 | 40 | 3 | 5000 | 1500 | 1500 | JPM,BAC |
| 130 | 52,43% | 52,43% | -$179,80 | 14 | 45 | 3 | 5000 | 1500 | 1500 | JNJ, PFE |
| 131 | 39,77% | 39,54% | $0,69 | 14 | 19 | 3 | 5000 | 1500 | 1500 | EURUSD=X,EURGBP=X |
| 132 | 34,34% | 34,11% | -$0,05 | 14 | 19 | 3 | 5000 | 1500 | 1500 | EUR=X,CAD=X |
| 133 | 30,65% | 30,41% | -$0,02 | 14 | 19 | 3 | 5000 | 1500 | 1500 | JPYCNY=X,JPYUSD=X, |
| 134 | 38,66% | 38,66% | $62,36 | 14 | 10 | 3 | 5000 | 1500 | 1500 | BZ=F,CL=F |
| 135 | 36,33% | 36,33% | -$992,10 | 14 | 20 | 3 | 5000 | 1500 | 1500 | GC=F,SI=F |
| 136 | 42,04% | 41,84% | $88,65 | 14 | 20 | 3 | 5000 | 1500 | 1500 | KC=F,CC=F |
| 137 | 48,24% | 47,65% | -$196 269,13 | 14 | 5 | 3 | 5000 | 1500 | 1500 | BTC-USD, ETH-USD |
| 138 | 51,76% | 51,18% | $219 505,39 | 14 | 5 | 3 | 5000 | 1500 | 1500 | BTC-USD, BTC-EUR |
| 139 | 53,53% | 53,53% | -$402 412,21 | 14 | 5 | 3 | 5000 | 1500 | 1500 | BTC-USD, GC=F |

Individual_Project_
Tests.xlsx

## 8.3 Project Repository

https://github.coventry.ac.uk/306aae-2021janmay/Paul_J._Aru-Project-7155312



Security Advisory. Users are reminded that they should not store any credentials within their github code or repositories.

306aae-2021janmay / Paul_J._Aru-Project-7155312 Private

Paul Johannes Aru Documenting and Tidying-Up ... 4cf96e1 9 hours ago 6 commits

| | | |
|---|---|---|
| Additional Documentation | Documenting and Tidying-Up | 9 hours ago |
| Project Scripts | Documenting and Tidying-Up | 9 hours ago |
| .gitattributes | Initial commit | 7 months ago |
| .gitignore | Documenting and Tidying-Up | 9 hours ago |
| README.md | Documenting and Tidying-Up | 9 hours ago |

README.md

**Personal-Project-Repository**

Updated: 29/04/2021

The Repository for my Individual Project!

**Project Scripts**

- **downloadSequencedTrainingData.m** - A MATLAB Function Script. Creates a `Dataset` directory with sub-directories for each class. Each class directory will be filled with `.csv` Data Points. The *Comma-Separated Values* files contain stock information for requested shares in a given time window. (*Uses Lenskiy's function for downloading the data from Yahoo Finance*).
- **createLSTM.m** - A MATLAB Function Script. Trains a *Long Short-Term Memory* Neural Network model from information stored in the `Dataset` directory.
- **calculateEarnings.m** - A MATLAB Function Script. Runs a simple simulation with the MATLAB NN model. Currently set up to trade for 3 months.
- **batchProcess.m** - A MATLAB Script. Runs the above 3 functions with varying parameters, enabling mass testing with a "hands-off" approach.

Deprecated Scripts

- **downloadTrainingData.m** - Creates a *.csv* file containing stock information for given shares in a given time period. (Uses Lenskiy's function that downloads the data from Yahoo)
- **createNeuralNetwork.m** - Trains a basic Neural Network based on data provided from a *.csv* file. (Based on the Matlab Example)

**Additional Documentation**

- **Presentation.pptx** - Details the Project and Findings.

**Acknowledgements:**

- Artem Lensky for Creating MATLAB Functions that Enable Access to Yahoo Finance Information.

About

The Repository for my Individual Project.

Readme

Releases

No releases published
Create a new release

Languages

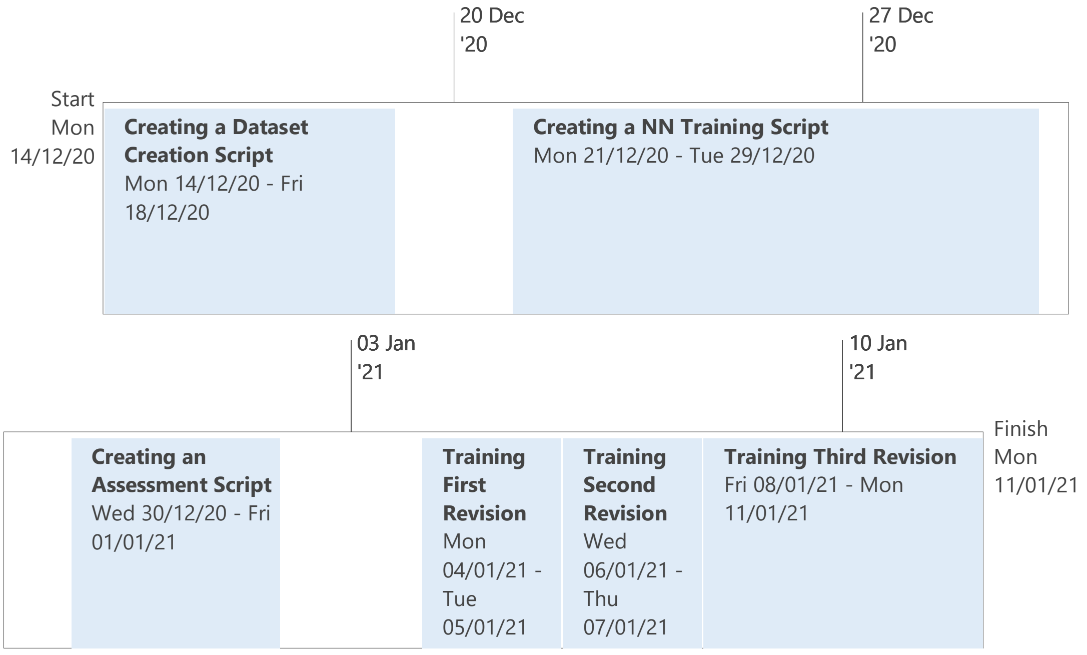
● MATLAB 100.0%

© 2021 GitHub, Inc. Help Support API Training Blog About

GitHub Enterprise Server 2.22.1

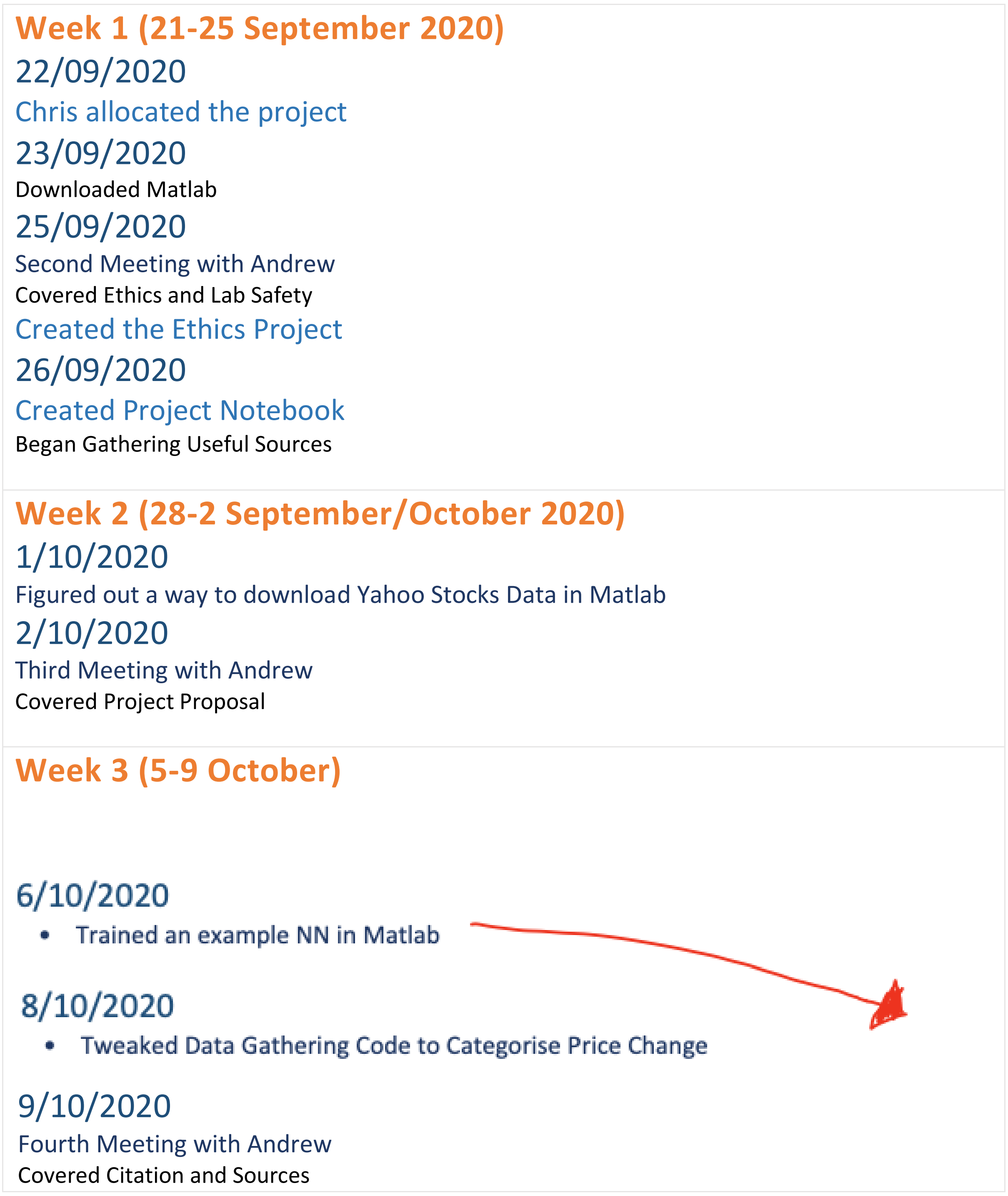
*Capture 1 - GitHub Repository Page*

## 8.4 Microsoft Project Initial Timeline



## 8.5 Logbook

[Link to Microsoft OneNote Logbook]



**Week 1 (21-25 September 2020)**

22/09/2020

Chris allocated the project

23/09/2020

Downloaded Matlab

25/09/2020

Second Meeting with Andrew

Covered Ethics and Lab Safety

Created the Ethics Project

26/09/2020

Created Project Notebook

Began Gathering Useful Sources

**Week 2 (28-2 September/October 2020)**

1/10/2020

Figured out a way to download Yahoo Stocks Data in Matlab

2/10/2020

Third Meeting with Andrew

Covered Project Proposal

**Week 3 (5-9 October)**

6/10/2020

- Trained an example NN in Matlab

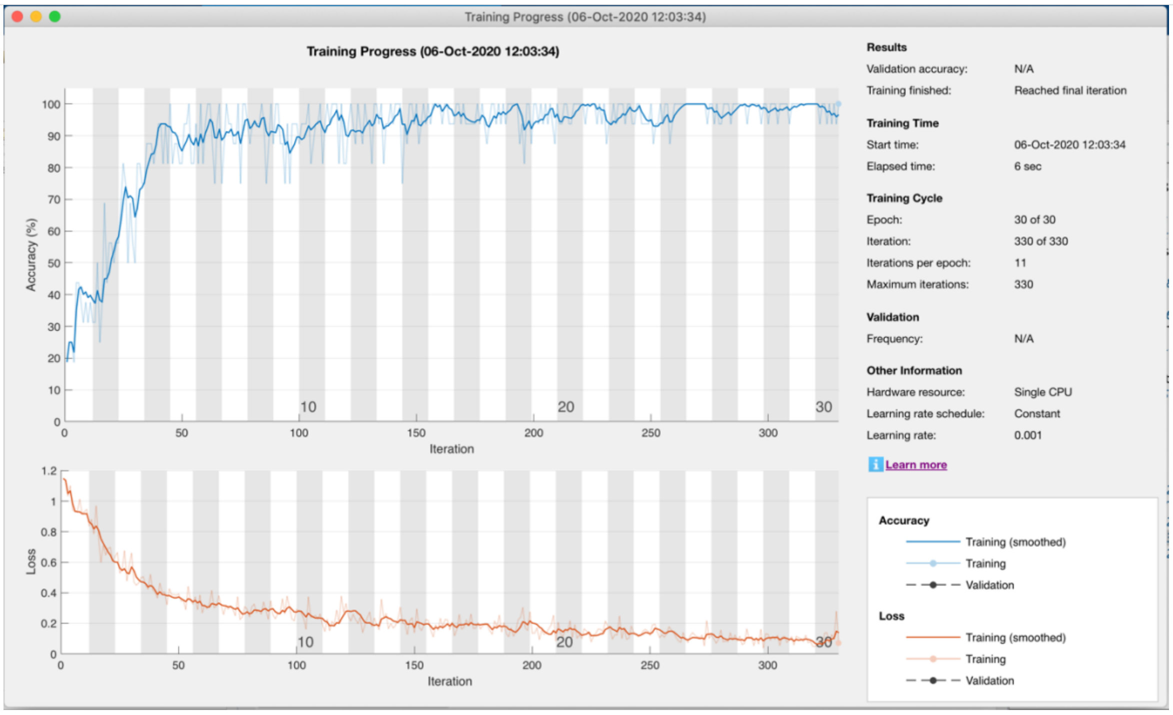
8/10/2020

- Tweaked Data Gathering Code to Categorise Price Change

9/10/2020

Fourth Meeting with Andrew

Covered Citation and Sources



**Results**
Validation accuracy: N/A
Training finished: Reached final iteration

**Training Time**
Start time: 06-Oct-2020 12:03:34
Elapsed time: 6 sec

**Training Cycle**
Epoch: 30 of 30
Iteration: 330 of 330
Iterations per epoch: 11
Maximum iterations: 330

**Validation**
Frequency: N/A

**Other Information**
Hardware resource: Single CPU
Learning rate schedule: Constant
Learning rate: 0.001

## Week 4 (12-16 October)

### 6/10/2020

Trained a custom NN in Matlab using the Stock Market data from Yahoo (Used INTC as an example, daily measurements for the past year) (Training settings same as on example) [accuracy = 0.6316]

### 15/10/2020

Created GitHub repo

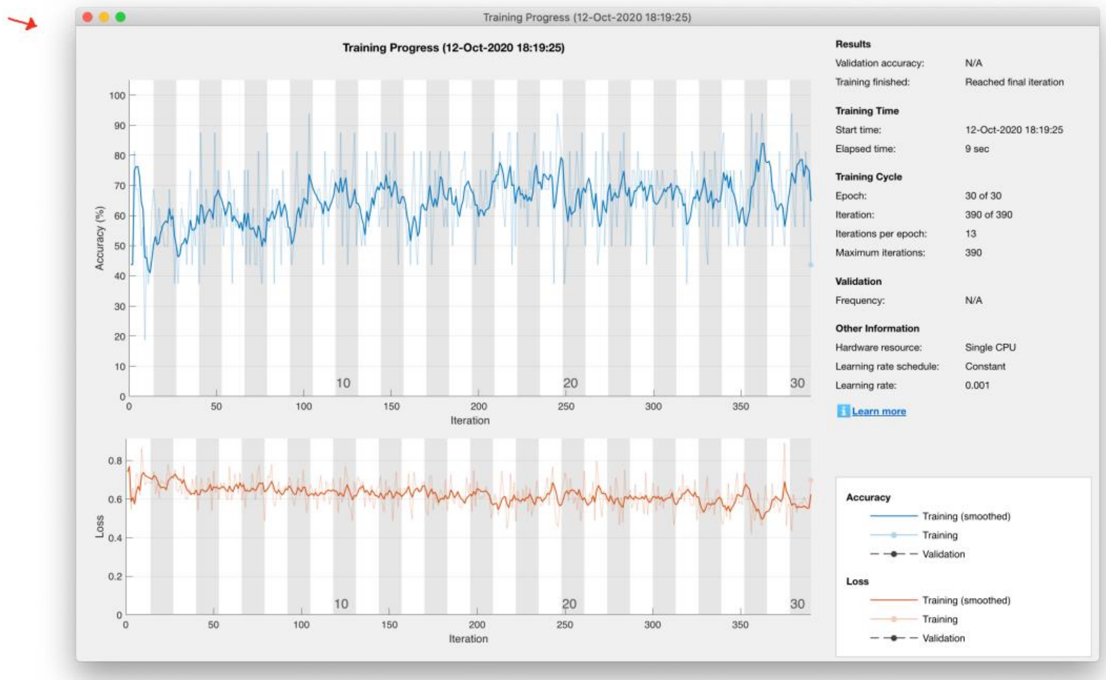
### 16/10/2020

Fifth Meeting with Andrew

Covered Roles & Responsibilities

### 17/10/2020

Submitted Signed Project Brief to Aula



| | |
|---|---|
| **Results** | |
| Validation accuracy: | N/A |
| Training finished: | Reached final iteration |
| **Training Time** | |
| Start time: | 12-Oct-2020 18:19:25 |
| Elapsed time: | 9 sec |
| **Training Cycle** | |
| Epoch: | 30 of 30 |
| Iteration: | 390 of 390 |
| Iterations per epoch: | 13 |
| Maximum iterations: | 390 |
| **Validation** | |
| Frequency: | N/A |
| **Other Information** | |
| Hardware resource: | Single CPU |
| Learning rate schedule: | Constant |
| Learning rate: | 0.001 |

## Week 5 (19-23 October)

### 21/10/2020

Began putting together Literature List for Literature Review

### 23/10/2020

Sixth Meeting with Andrew

Covered Aim Developing

## Week 6 (26-30 October)

### 29/10/2020

First Project meeting with Chris

<<Project Meeting Form.docx>>

### 30/10/2020

Seventh Meeting with Andrew

Covered final dissertation structure

## Week 7 (2-6 November)

### 2/11/2020

Finished reading literature

### 6/11/2020

Eighth Meeting with Andrew

Covered assessment rubrics

## Week 8 (9-13 November)

### 9/11/2020

Started writing literature review

### 12/11/2020

Second Project meeting with Chris and Arfan

<<Project Meeting Form.docx>>

### 13/11/2020

Ninth Meeting with Andrew

Covered Developing Objectives and Choosing Methods

## Holiday Week 1 (21-27 December)

### 27/12/2020

Installed Matlab on Desktop Computer alongside GPU acceleration toolbox

Investigated LSTM Layers in Matlab with an example

## Holiday Week 2 (28-3 December/January)

### 28/12/2020

Started Writing a new Data Downloading Script that would Store Sequential Data with Adjustable Window Sizes

### 29/12/2020

Finished Writing the Sequential Data Downloading Script

Started Writing a LSTM Neural Network Training Script

### 30/12/2020

Finished Writing the LSTM Neural Network Training Script

Modified the Sequential Data Downloading Script to add additional classes

### 31/12/2020

Tweaked Statistics and Class Allocation on the Sequential Data Downloading Script

### 3/1/2021

Started Work on Batch Processing Script

## Holiday Week 3 (4-10 January)

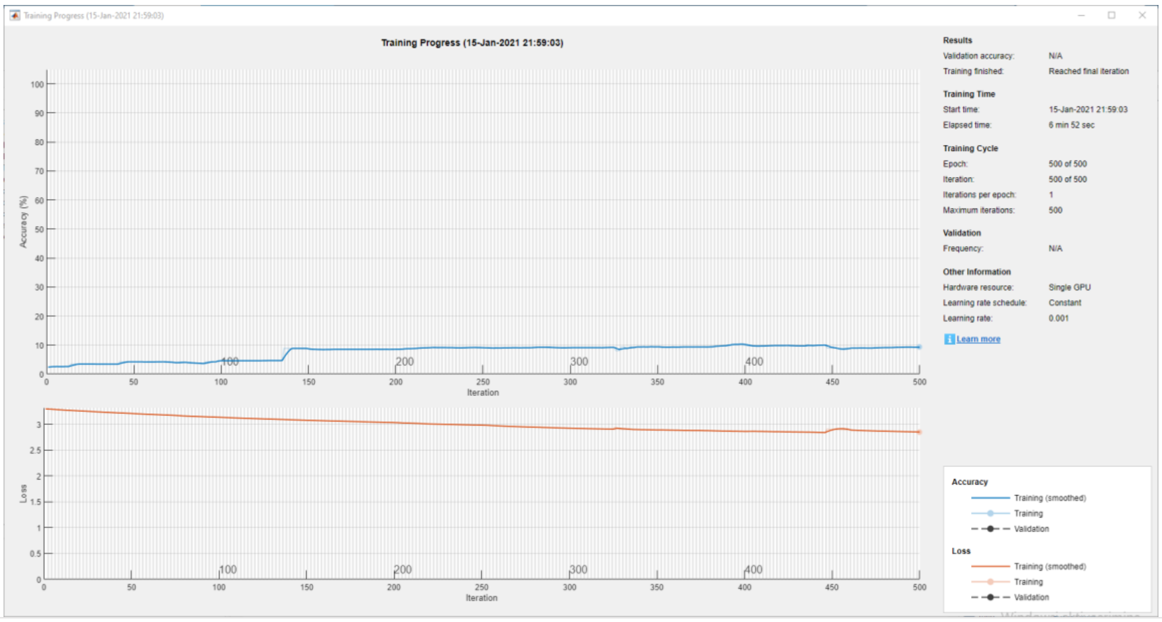
Finished Batch Processing Script

Automated Class Allocation

Began Running Tests

## Holiday Week 4 (11-17 January)

- Continued Running Tests
- Added Class Sample Size Normalisation
- Possible Breakthrough After Removing Trade Volume



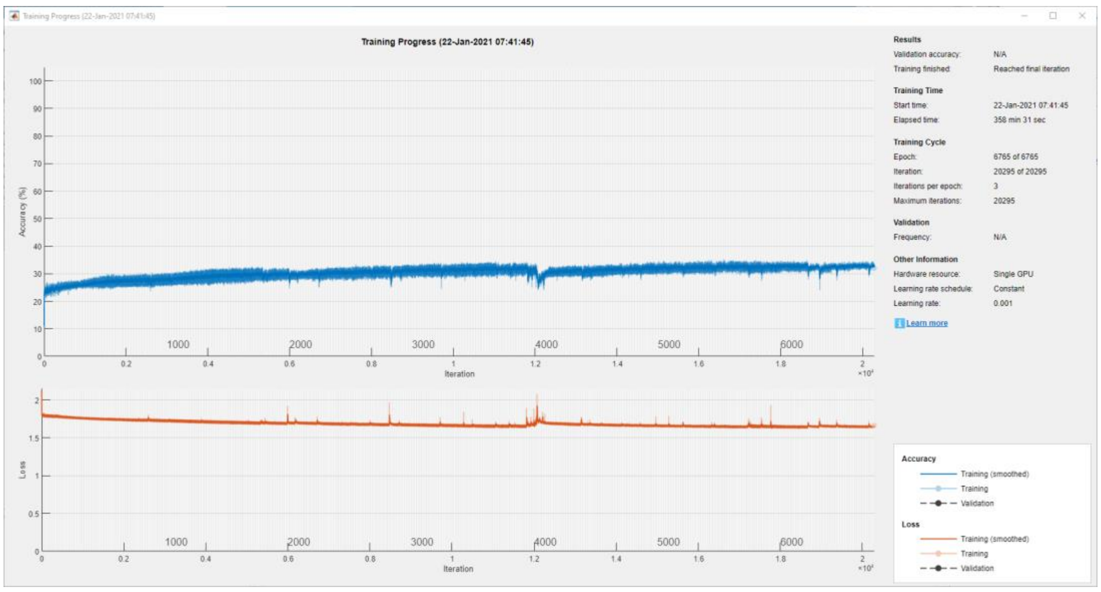
**Results**

| | |
|---|---|
| Validation accuracy: | N/A |
| Training finished: | Reached final iteration |
| **Training Time** | |
| Start time: | 15-Jan-2021 21:59:03 |
| Elapsed time: | 6 min 52 sec |
| **Training Cycle** | |
| Epoch: | 500 of 500 |
| Iteration: | 500 of 500 |
| Iterations per epoch: | 1 |
| Maximum iterations: | 500 |
| **Validation** | |
| Frequency: | N/A |
| **Other Information** | |
| Hardware resource: | Single GPU |
| Learning rate schedule: | Constant |
| Learning rate: | 0.001 |

Learn more

## Week 13 (18-24 January)

- Increased Epoch breakthrough



**Results**

| | |
|---|---|
| Validation accuracy: | N/A |
| Training finished: | Reached final iteration |
| **Training Time** | |
| Start time: | 22-Jan-2021 07:41:45 |
| Elapsed time: | 358 min 31 sec |
| **Training Cycle** | |
| Epoch: | 6765 of 6765 |
| Iteration: | 20295 of 20295 |
| Iterations per epoch: | 3 |
| Maximum iterations: | 20295 |
| **Validation** | |
| Frequency: | N/A |
| **Other Information** | |
| Hardware resource: | Single GPU |
| Learning rate schedule: | Constant |
| Learning rate: | 0.001 |

Learn more

## 8.6 Project Meeting Notes

### 8.6.1 Meeting 1

**Face-To-Face Student/Supervisor Meeting Record**

| | | | |
|---|---|---|---|
| **Project Title:** | | **Photo:** | |
| **Student Name:** | Paul J. Aru | **Student ID:** | 7155312 |
| **Supervisor:** | Chris Bass | **Student UID:** | Arup |
| **Supervisor UID:** | | **Department:** | AAEEE |
| **Course Code:** | ECU134 | **Module Code:** | 305AAE / 306AAE |
| **Date Today:** | 29/10/2020 | **Time:** | |

**Current Progress and Issues:**

*(Include challenges encountered and actions taken to overcome them)*

- *Found a Matlab library that can be used for pulling Market Data from Yahoo.*
- *Trained a few basic Neural Networks with the data in Matlab.*
- *Read through 17 white papers on applying Machine Learning to Financial Markets.*
- *Set-up a Project Repository in GitHub.*
- *Created Project Brief.*
- *Created Ethics Proposal.*

**Agreed Key Action Points:**

*(Include dates of any deadlines)*

- *Meet again in 2 weeks.*
- *Speak with Arfan about the technical side (Different Machine Learning models).*
- *Finish Reading Literature.*

| | |
|---|---|
| **Date and Time of next meeting:** | *Roughly in 2 weeks.* |

### 8.6.2 Meeting 2

**Face-To-Face Student/Supervisor Meeting Record**

| | | | |
|---|---|---|---|
| **Project Title:** | | **Photo:** | |
| **Student Name:** | Paul J. Aru | **Student ID:** | 7155312 |
| **Supervisor:** | Chris Bass | **Student UID:** | Arup |
| **Supervisor UID:** | | **Department:** | AAEEE |
| **Course Code:** | ECU134 | **Module Code:** | 305AAE / 306AAE |
| **Date Today:** | 12/11/2020 | **Time:** | |

**Current Progress and Issues:**

*(Include challenges encountered and actions taken to overcome them)*

- *Finished reading through 21 white papers on applying Machine Learning to Financial Markets.*
- *Started writing the Literature Review.*
- *Created Literature Table, condensing all the relevant information from literature.*

**Agreed Key Action Points:**

*(Include dates of any deadlines)*

- *Look into Deep Convolutional Neural Networks with GoogleNet or ResNet structures.*
- *Finish writing the literature review.*

| | |
|---|---|
| **Date and Time of next meeting:** | *Next Academic Term.* |

## 8.6.3 Meeting 3

**Face-To-Face Student/Supervisor Meeting Record**

| | | | |
|---|---|---|---|
| **Project Title:** | | **Photo:** | |
| **Student Name:** | Paul J. Aru | **Student ID:** | 7155312 |
| **Supervisor:** | Chris Bass | **Student UID:** | Arup |
| **Supervisor UID:** | | **Department:** | AAEEE |
| **Course Code:** | ECU134 | **Module Code:** | 305AAE / 306AAE |
| **Date Today:** | 22/01/2021 | **Time:** | |

**Current Progress and Issues:**

*(Include challenges encountered and actions taken to overcome them)*

- *Developed Tools for Training LSTM NNs.*
- *Collected a Selection of Results.*
-

**Agreed Key Action Points:**

*(Include dates of any deadlines)*

- *Look into Deep Convolutional Neural Networks with structures.*

| | |
|---|---|
| **Date and Time of next meeting:** | - |