**Table of Contents**

**1 INTRODUCTION**

* 1. Background to dissertation

1.2 Aims and Objectives

**2 LITERATURE REVIEW**

**2.1 Explanation of Multifactor Models**

2.1.1 Multifactor process

* + 1. Factor selection

**2.3 Non-Linear Multifactor Models in the Literature**

**2.4 Neural Networks**

2.4.1 Feed Forward Neural Networks

2.4.2 Deep Learning

2.4.3 Recurrent Neural Networks

2.4.4 Backpropagation

2.4.5 Gradient Descent

2.4.6 Long Short-Term Memory

* 1. **LSTMs in Multifactor Investment Models**

**3 DESIGN AND IMPLEMENTATION**

**3.1 Financial Data – Explanation of the financial data used**

**3.2 Python packages – Explanation of the Python packages that will be used to implement Neural Nets**

3.2.1 Implementation

3.2.2 Training the RNN models

**4 TESTING AND RESULTS**

**4.1 Comparison with accuracy metrics**

**5 ANALYSIS AND DISCUSSION**

5.1.1 Analysis of all comparisons

5.1.2 Summary of results/ preferred model

**6 CONCLUSION**

**7 REFERENCES**

**1 INTRODUCTION**

**1.1 Background to dissertation**

The project was devised from an internal training course at Aberdeen Standard Investments

given by the Quantitative Investment Strategy (QIS) group. The QIS group use what are called Multifactor Investment models, models that use factors to drive investment decisions. Factors can be described as collections of metrics around the performance of a company that have a similar theme. The main factors or themes in multifactor investing are traditionally but not always; company value, company quality, company size, momentum of company stock price and volatility of company stock price. Often these factors are considered in a linear manner, by this we mean it often assumed that factors determine the worth of a company independent of each other. My motivation was to investigate the use of machine learning techniques, specifically neural networks, to investigate the interactions of these factors in in a way to improve investment decisions.

ASI currently uses machine learning in a limited way in one of their products, one investment model uses random forests to pinpoint which specific factors are likely to perform well in specific economic conditions.

**1.2 Aims and Objectives**

My aim is to see if the use of neural networks in factor selection would be beneficial to the multifactor investment process. The project will look to construct a portfolio that will use a multifactor model to make investment decisions based on the use of deep learning factor selection and compare these to the equivalent linear model. I will specifically look to use Recurrent Neural Networks (RNNs) in the form of Short Long-Term Memory (LSTM) or if more suitable a Gated Recurrent Units (GRUs.) The project will require the adaption, training and validation of an RNN for factor selection. The factors or metrics will be used in combination with the stock returns (the change in stock prices either positively or negatively) over a period to train the RNN to predict factor combinations that will improve upon currently used linear models and their ability to predict returns.

**2 LITERATURE REVIEW**

**2.1 Explanation of Multifactor Models**

2.1.1 Multifactor process

The process of using multifactor models is widespread in quantitative finance. The arbitrage pricing theory outlined in a paper by Ross, 1976 is often the attributed starting point for multifactor models, the paper introduced a method of extracting factors by analysis of past stock returns, this mainly used macroeconomic factors. This was further developed by Fama and French, 1992 where the multifactor models were related more to finding stock returns using the attributes of companies. This was done by looking into companies accounting metrics, in particular valuation ratios such the ‘price-earnings ratio’ and the ‘price-to-book-ratio.’ The ‘price-earnings ratio’ (P/E) is the ratio for valuing a company that measures its current share price relative to its per-share earnings. The ratio shows how much investors are willing to pay per dollar (or currency of the country of origin) of earnings. If a company were currently trading at a multiple P/E of 20, the interpretation is that an investor is willing to pay $20 for $1 of current earnings. In general, a high P/E suggests that investors are expecting higher earnings growth in the future compared to companies with a lower P/E. A low P/E can indicate either that a company may currently be undervalued or that the company is doing exceptionally well relative to its past trends.  Analysts use the ‘price-to-book ratio’ (P/B) to compare a firm's stock market value to book value by dividing price per share by book value per share. P/B ratio = stock market price per share / book value per share. In this equation, book value per share =  (total assets - total liabilities) / number of shares outstanding. A lower P/B ratio could mean the stock is undervalued. However, it could also mean something is fundamentally wrong with the company.

P/E and P/B were the foundations of accounting metric use in factor models but they are by no means the only metrics, studies have shown there to be in access of 300 factors used in academic papers on multifactor models (Harvey et al. 2016.)

The core assumption behind factor analysis is that historical information has some forecasting value. These independent variables that describe a firm’s past financial performance may serve as leading indicators of stock prices. In these models stock returns are explained by common factors; if one is able to predict the likely future value of a factor, a higher return can be achieved by constructing a portfolio that tilts toward “good" factors and away from “bad" ones.

* + 1. Factor selection

For these models the relationship between the stock returns and the factors is linear. Traditional investment approaches (Elton and Gruber, 1991) assume that the return of a security can be described by a multifactor linear model.

(1)

**Figure 1. Where denotes the return on security , are a set of factor values and are security exposure to factor l, is an intercept term (which under the Capital Asset Pricing Model([[1]](#footnote-1)) framework is assumed to be equal to the risk free rate([[2]](#footnote-2)) of return)**

**and is a random term with mean zero which is assumed to be uncorrelated across securities. (Levin, 1996, p 966)**

The factors may consist of any set of variables deemed to have explanatory power for security returns, however traditionally these factors are related to the themes of value, quality, size, momentum and risk. But these could be aspects of macroeconomics, fundamental security analysis, technical attributes or a combination of the above. The value of a factor is the expected excess return above risk free rate of a security with exposure to the factor, and no exposure to all other factors. Once the factors are set, the model assumption is that on average, two securities with similar factor weights () will behave in a similar manner. The factor model was not originally developed as a predictive model (Levin, 1996), but rather as an explanatory model, with the returns and the factor values assumed to be concurrent. To change this to a predictive model, each factor value must be replaced by an estimate, resulting in the model

(2)

**Figure 2. Whereis a security's future return and ^ is an estimate of the future value of factor l, based on currently available information. (Levin, 1996, p 967)**

The estimation of^ can be approached in different ways from a simple use of the historical mean to estimate the factor value, to more complex approaches such as attempting to construct a time series model for predicting the factor values (Levin, 1996.) Factor models in this form are used both to control the risk and to enhance the returns. Stocks chosen in this way are used to construct portfolios, or groupings of stocks. The grouping in the first case, is based on capturing the major sources of correlation among security returns, then constructing a balanced portfolio of stocks that can diversifies specific risk away. For the latter, stocks can be grouped based on factors that are more likely to delivery good returns.

**2.3 Non-Linear Multifactor Models in the Literature**

While linear factor models have proven to be very useful tools for financial analysis and investment management, the assumption of linear relationship between factor values and expected return is quite restrictive. The use of linear models means that each factor affects the return independently and therefore they ignore any possible interaction between different factors. To overcome these shortcomings of linear models, we must consider more complex models that allow for nonlinear relationship among factor values and expected returns. Considering the complexity of the financial markets, it is more appropriate to assume a nonlinear relationship between the stock returns and the factors.

(3)

**Figure 3. Where is introduced as a non-linear function and is now considered as the noise unexplained by the model (Levin, 1996, p 967)**

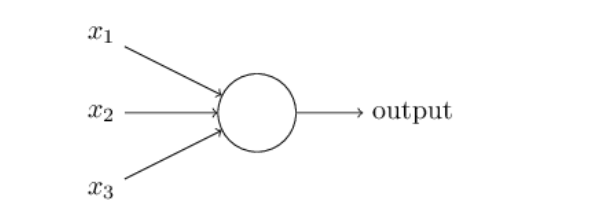
The prediction task for the nonlinear model is substantially more complex than in the linear case since it requires both the estimation of future factor values as well as a determination of the unknown function

To estimate the unknown function a family of models needs to be selected, from which a model is to be identified. Options for modelling the relationship between factor exposures and future returns would include using the class of multilayer feedforward neural networks.

A paper by Olson and Mossman 2003 compared linear models to non-linear models using neural networks. Their paper compares the forecasting performance of neural networks with traditional linear regression techniques, using fundamental accounting data as the metrics to forecast 1-year-ahead Canadian stock returns. They judge their forecasts based on the profit the model would generate, and on forecasting accuracy. Profitability is what matters to most investors so, trading profit is an interesting measure of the usefulness of a forecasting model. This accuracy was also captured in statistical measures of forecast error (e.g. mean squared error, mean absolute error, or mean absolute percentage error). The paper used a data set of 61 annual accounting ratios and financial variables for each company traded on the Toronto Stock Exchange during each of the 18 years in the sample. This includes three ratios commonly used in the finance literature price-to-book ratio, price-earnings ratio, and the price-to-sales ratio.[[3]](#footnote-3) These are the independent variables the dependent variable is the stock’s market-adjusted ‘abnormal’ return. Abnormal meaning the return the stock price movement measured over the next year minus the calculated return of all the entire Toronto Stock Exchange stocks (the change in all stock prices either positively or negatively normally expressed as a percentage.) The linear model compared is the Ordinary Least Squares (OLS) regression model, OLS is used to estimate the relationship between the metrics and the abnormal return of the stock market. This is done by minimising the sum of the squares in the difference between the observed and predicted values of the dependent variable configured as a straight line. Olson et Al concluded that “value can be further enhanced by using neural networks rather than regression techniques. Similar conclusions where linear models were outperformed by neural network models were reached by Balkin and Ord (2000) and Qi (2001.)

**2.4 Neural Networks**

Neural Networks were developed from artificial neural networks called perceptron networks (Rosenblatt, 1962) A perceptron, has a very simple structure

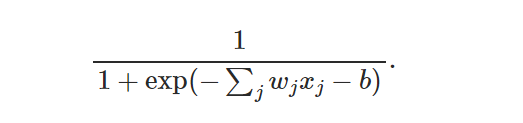
(4)

**Figure 4. Perceptron structure where it takes several binary inputs, x1,x2,… to the neuron and produces a single binary output (Nielsen** [**http://neuralnetworksanddeeplearning.com/index.html**](http://neuralnetworksanddeeplearning.com/index.html)**)**

In Rosenblatt’s model, weights: w1,w2,…were introduced as part of a simple rule to compute the output. Weights were real numbers expressing the importance of the respective inputs to the output. The neuron's output, 0 or 1, is determined by whether the weighted sum is less than or greater than some threshold value. Just like the weights, the threshold is a real number which is a parameter of the neuron.

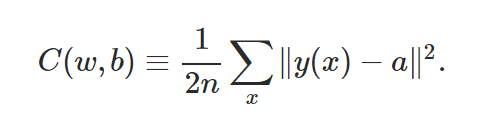
To make complicated decisions, like investment decisions based on a series of metrics, a network of perceptrons would be required. However, perceptron output is binary and changes to the weights and threshold (bias) can create a large change in the output which is not optimal in the creation of a neural network, a large change can cause the behaviour of the rest of the network to completely change in a very complicated way.

If a small change in a weight (or bias) causes only a small change in output, then it would be possible to modify the weights and biases to get the network to behave more in a more desirable manner. Sigmoid neurons developed by McCulloch and Pitts (1943) are similar to perceptrons but modified so that small changes in their weights and bias cause only a small change in their output and can be used in networks efficiently. Just like a perceptron, the sigmoid neuron has inputs but instead of being just 0 or 1, these inputs can also take on any values between 0 and 1. So, for instance, 0.638…0.638… is a valid input for a sigmoid neuron. Also, just like a perceptron, the sigmoid neuron has weights for each input and an overall bias, but the output is not 0 or 1. The output is defined by

(5)

**Figure 5. Output of a sigmoid neuron with inputs weightsand bias *b* (Nielsen** [**http://neuralnetworksanddeeplearning.com/index.html**](http://neuralnetworksanddeeplearning.com/index.html)**)**

The to generate the desired output an algorithm needs to find weights and biases so that the output from the network approximates *y(x)* for all training inputs *x*. To quantify how to achieve this a cost function is applied

(6)

**Figure 6. Here, *w* denotes the collection of all weights in the network, *b* all the biases, n is the total number of training inputs, *a* is the vector of outputs from the network when*x* is input, and the sum is over all training inputs, *x.***

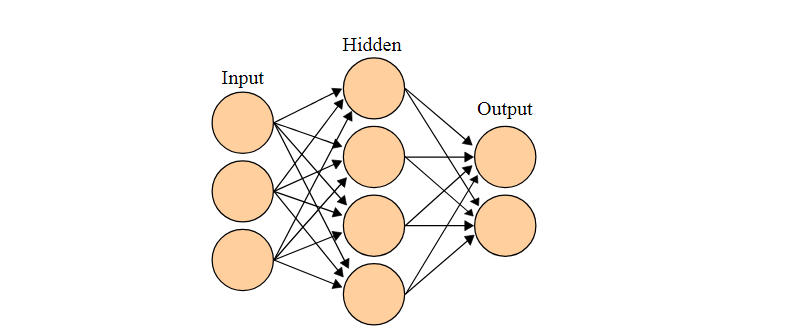
**sometimes known as the mean squared error.(Nielsen** [**http://neuralnetworksanddeeplearning.com**](http://neuralnetworksanddeeplearning.com)**)**

*C* here is the quadratic cost function; it's also sometimes known as the mean squared error,  minimising the cost function, the aim of our training algorithm will be to minimize the cost *C(w,b)* as a function of the weights and biases. In other words, we want to find a set of weights and biases which make the cost as small as possible.

Multiple layers of sigmoid neurons are the basis of neural networks in their many forms.

2.4.1 Feed Forward Neural Networks

Olson et Al (2003) for their multifactor model use the form of neural network called Feed Forward Neural Networks (FFNNs.) The main reason for using FFNNs is that there is some non-linear aspect to the forecasting problem being considered. The non-linearity may take the form of a complex non-linear relationship between the independent and dependent variables, the existence of upper or lower thresholds for the inﬂuence of independent variables, or differences between forecasting up or down movements of the dependent variable. The advantage of using an FFNNs is that the analyst need not know the type of functional relationship that exists between the independent and dependent variables when modelling the relationship between factor exposures and future returns (Hertz et al., 1991). Their universal estimation capabilities (Cybenko, 1989), as well as the existence of an effective parameter tuning method (the backpropagation algorithm (Rumelhart et al., 1986)) makes this family of models a powerful tool for the identification of nonlinear mappings and therefore a natural choice for modelling financial data.

(7)

**Figure 7. A simple FFNN with one hidden layer, hidden layer just being the terminology from layers that are not input or output layers. (Olah,** [**http://colah.github.io**](http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/)**)**

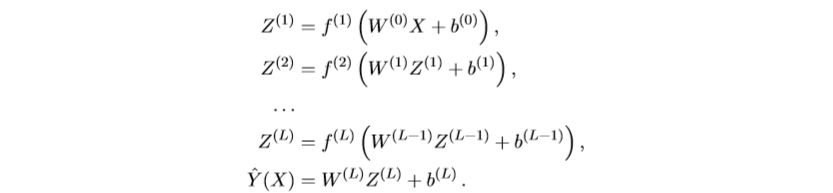
2.4.2 Deep Learning

Taking this initial step further introduces the concept of Deep Learning. Adeep neural network is simply a feed forward network with many hidden layers**.**  The data passes through many more layers than it does in conventional three-layer neural networks. By inputting data of multiple factors and passing them through many layers, deep learning could extract useful features, increase representational power, enhance performance, and improve the prediction accuracy for future stock returns.

Abe and Nakayama (2018) used deep learning in conjunction with multifactor models and investigated the performance of the method in the Japanese stock market. They showed that deep neural networks generally outperform shallow neural networks, and the best networks also outperform other machine learning models.

Deep learning trains a model on data to make predictions but is distinguished by passing learned features of data through different layers of abstraction. The deep approach employs hierarchical predictors comprising of a series of non-linear transformations applied to *x*. Each of the transformations is referred to as a layer, where the original input is *x*, the output of the ﬁrst transformation is the ﬁrst layer, and so on, with the output *y* as the *(L + 1)-th* layer. We use *l ∈ {l,...,L}* to index the layers from *1 to L*, which are called hidden layers. The number of layers *L* represents the depth of our architecture.

Speciﬁcally, a deep learning architecture can be described as, layers that are given univariate activation functions, activation functions are non-linear transformations of weighted data. Commonly used activation functions are sigmoidal, tanh or rectiﬁed linear units (ReLU) . The explicit structure of a deep prediction rule is then

 (8)

**Figure 8. denotes the *L-th* layer. The ﬁnal output is the response *Y* , which can be numeric or categorical.**

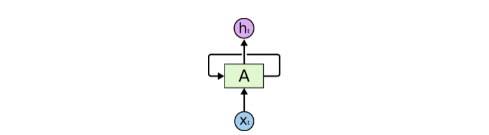
(**Heaton et al, 2017)**

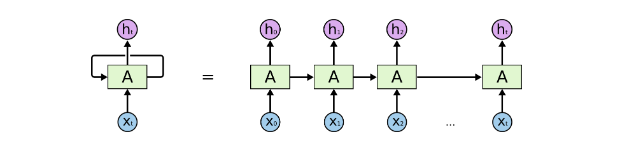
Here, are weight matrices, and are threshold or activation levels. Designing a good predictor depends crucially on the choice of univariate activation functions *.* The are hidden features (or factors) which the algorithm extracts.

2.4.3 Recurrent Neural Network

RNNs are an extension of a conventional neural networks that add a feedback connection to a feedback network. RNN algorithms it has been argued (Sang Il Le et al, 2018) are better suited for modelling multifactor ﬁnancial data. They argue that by feeding the network activations from a previous time step as inputs into the network, this inﬂuence helps predictions in the current step. In contrast, FFNNs are not appropriate for capturing these time-dependent dynamics as they operate on a ﬁxed-size time windows, and so they can provide only limited temporal modelling.

The benefit of RNNs is that they make use of sequential information, they can retain relevant information from earlier information that will help in the current decision making. A neural network representation:





(9)

**Figure 9. *A* has input  outputs a value . A loop allows information to be passed from one step of the network to the next.**

**(Olah,** [**http://colah.github.io**](http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/)**)**

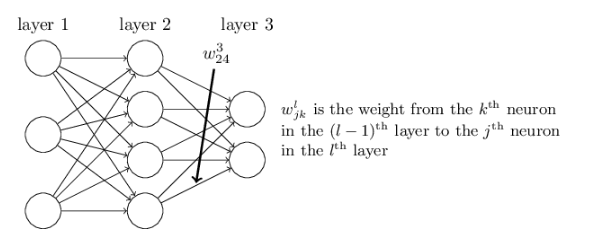
As it can be seen RNNs aren’t too different from NNs they represent a chain of NNs linking information from one to the next. Bengio et Al, 1994, found it is difficult to train simple RNNs to capture long-term dependencies, because the gradients tend to either vanish or explode. RNNs use gradient descent to minimise the cost function of the model and update the weights for the model. During backpropagation, the algorithm that allows the RNN to learn, the gradient value reduces as it goes back through time, if the gradient is small it does not contribute to learning.

2.4.4 Backpropagation

The chain rule of calculus is used to compute the derivatives of functions formed by composing other functions whose derivatives are known. Backpropagation is an algorithm that computes the chain rule, with a speciﬁc order of operations that is highly eﬃcient. (Goodfellow et Al, 2016)

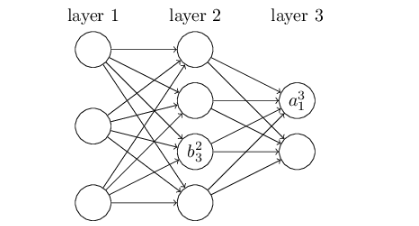
Backpropagation is about understanding how changing the weights and biases in a network changes the cost

function. Ultimately, this means computing the partial derivatives, the first being where

(10)

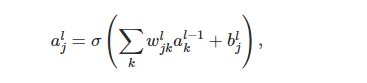
**Figure 10. (Nielsen** [**http://neuralnetworksanddeeplearning.com**](http://neuralnetworksanddeeplearning.com)**)**

The other partial derivative is  where it is the bias *b* that is computed related to the activation of the *j*th neuron of the *lth* layer.

 (11)

**Figure 11. (Nielsen** [**http://neuralnetworksanddeeplearning.com**](http://neuralnetworksanddeeplearning.com)**)**

Using these notations around the weights and biases we can determine the activations in the (*l* – 1)th  layer with the equation

 (12)

**Figure 12. (Nielsen** [**http://neuralnetworksanddeeplearning.com**](http://neuralnetworksanddeeplearning.com)**)**

In this equation the sum is over all neurons *k*  in the (*l* – 1)th layer. This is better expressed in matrix form, where the weights, biases and activations can be represented as vectors. Also, the function is better represented as a function. In the above equation this is σ (sigma) so it would be applying σ to every element in the vector.

The equation (12) can therefore be rewritten in a much more efficient form

 (13)

**Figure 11. (Nielsen** [**http://neuralnetworksanddeeplearning.com**](http://neuralnetworksanddeeplearning.com)**)**

This expression gives us a much more high-level way of thinking about how the activations in one layer relate to activations in the previous layer: just applying the weight matrix to the activations, then add the bias vector, and finally apply the σ function.

As mentioned earlier the goal of backpropagation is to compute the two partial derivatives of the cost function with respect to any weights or biases. There are still conditions required for the overall backpropagation algorithm to work, two assumptions about the cost function (Nielsen, 2016.) Firstly, the cost function can be written as an average over cost functions for individual training examples. This is because the backpropagation provides the partial derivatives  for a single training example and then recovers

 by taking the average over training examples. The second assumption we make about the cost is that it can be written as a function of the outputs from the neural network. The chosen cost function must satisfy this requirement. The network does not learn the cost function itself, so it is necessary to regard it as a function of the output activities alone. What backpropagation is about understanding is how changing the weights and biases in a network changes the cost function. This means computing the partial derivatives  and  but what is also required to compute these that has not been mentioned is the intermediate error rate. Backpropagation gives the process to compute the error then relate it to the partial derivatives above.

2.4.5 Gradient Descent

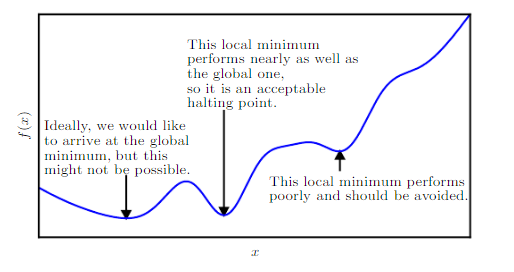
Gradient descent is used to minimise the cost function, it uses the derivatives of the cost function

The derivative is therefore useful for minimizing a function because it shows how to change *x* in order to make a small improvement in *y*. The derivative of the function is used, for a function *y = f(x)* the derivative

gives the slope of *f(x)* at the point *x.* It specifies how to scale a small change in the input to obtain the corresponding change in the output. We can thus reduce *f(x)* by moving *x* in small steps with the opposite sign of the derivative. This technique is called gradient descent.

When = 0, the derivative provides no information about which direction to move. Points where

these points are known as critical points, or stationary points. A local minimum is a point where *f(x)* is lower than at all neighbouring points, so it is no longer possible to decrease *f(x)* by making infinitesimal steps. Maxima is where it is higher than all neighbouring points. A point that obtains the absolute lowest value of *f(x)* is a global minimum. There can be only one global minimum or multiple global minima of the function. It is also possible for there to be local minima that are not globally optimal. In the context of deep learning, we optimize functions that may have many local minima that are not optimal, and many saddle points surrounded by very ﬂat regions. All of this makes optimization diﬃcult, especially when the input to the function is multidimensional. We therefore usually settle for ﬁnding a value off that is very low but not necessarily minimal in any formal sense.

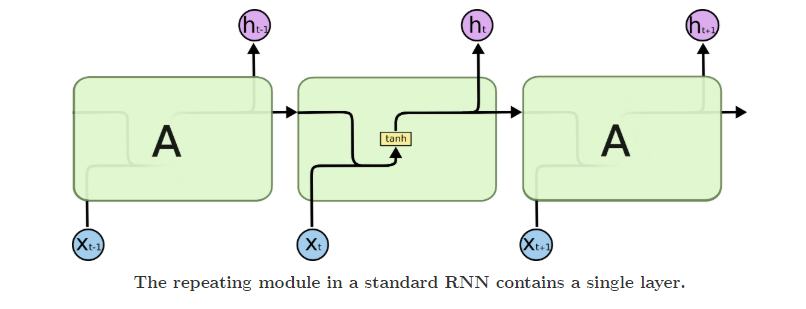
 (14)

**Figure 14. Goodfellow et al 2016**

Summing up, each of the neural network's weights receives an update proportional to the partial derivative of the error function with respect to the current weight in each iteration of training. The problem is that in some cases, the gradient will be vanishingly small, effectively preventing the weight from changing its value. In the worst case, this may completely stop the neural network from further training. In RNNs the initial layers suffer most from this issue.

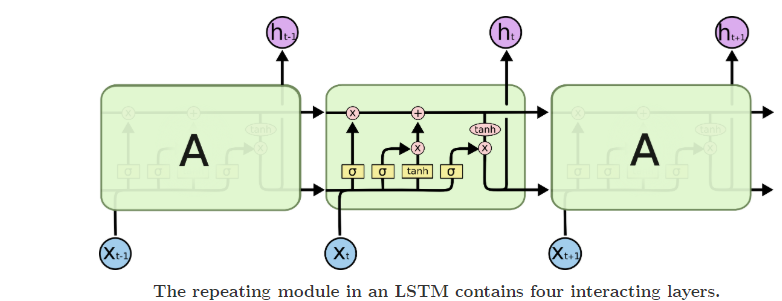
2.4.6 Long Short-Term Memory

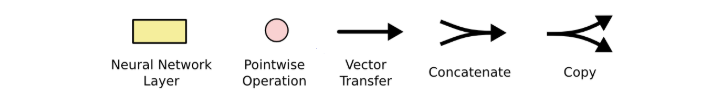
Long Short Term Memory (LSTM) networks are a special kind of RNN, capable of learning long- term dependencies. They were developed by [Hochreiter and Schmidhuber (1997)](http://www.bioinf.jku.at/publications/older/2604.pdf). All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single activation layer, the activation function in the below diagram is tanh.

(15)

**Figure 15. *A* has input  outputs a value . In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer. (Olah,** [**http://colah.github.io**](http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/)**)**

LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer there are four.



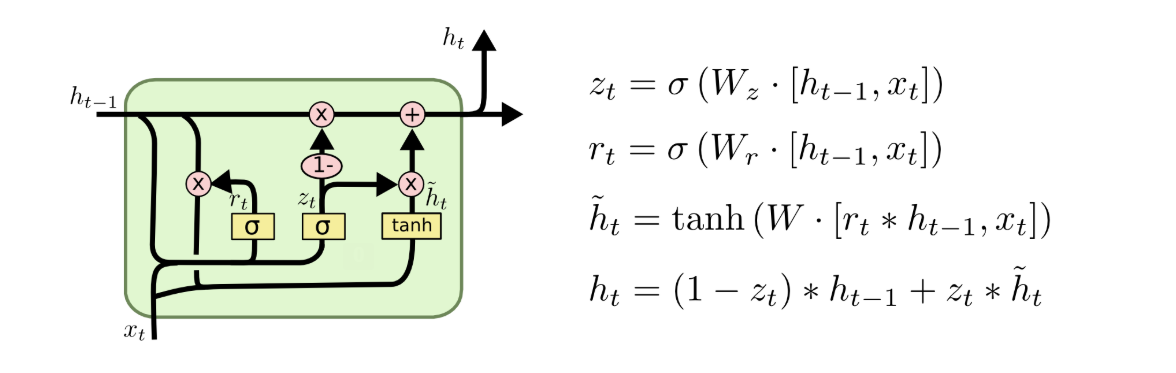
(16)

**Figure 16. *A* has input  outputs a value** **Instead of having a single neural network layer, there are four (Olah,** [**http://colah.github.io**](http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/)**)**

The LSTM does have the ability to remove or add information to the cell state, they do so by using structures called gates. Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation. The sigmoid layer outputs numbers between 0 and 1, describing how much of each component should be let through. A value of 0 means lets no information through while a value of 1 allows all information through. The first gate, to the left of all the layers above is the ‘forget gate’ this is a sigmoid layer that decides what information is to be thrown away from the cell. It looks at information from the previous layer and and outputs a number between 0 and 1 for each number in the previous cell state. Then it determines what new information will be stored in the cell state, this is done in two parts. Part one decided by an ‘input’ gate, a sigmoid layer decides which values to update. Part two creates a estimations of new candidate values that could be added to the current state, in the form of a vector.

In the next step, it combines these two to create an update to the state. In the the next step it multiplies the old cell state by the output of the ‘forget gate’ forgetting the things we decided to forget earlier. Then we add it to the new candidate values in the vector from the input gate. This is the new candidate values, scaled by how much we decided to update each state value. Finally, it decides what to output. This output will be based on current cell state but will be a filtered version. First, it runs a sigmoid layer which decides what parts of the cell it’s going to output. Then, we put the cell state through tanh (to push the values to be between −1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

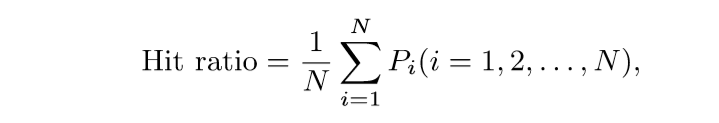
A variation on the LSTM is the Gated Recurrent Unit, or GRU, introduced by Cho, et al. (2014). It combines the forget and input gates into a single “update gate.” It also merges the cell state and hidden state and makes some other changes. The resulting model is simpler than standard LSTM models.

(17)

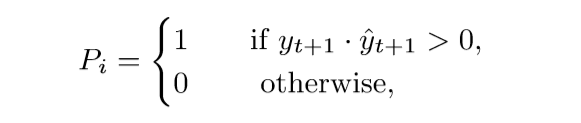
**Figure 16. (Olah,** [**http://colah.github.io**](http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/)**)**

* 1. **LSTMs in Multifactor Investment Models**

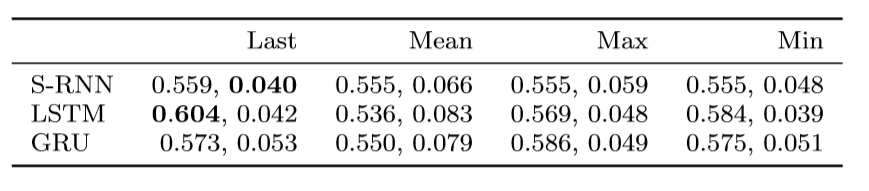
LSTMs can effectively learn important pieces of information that may be found at different positions in the ﬁnancial time series, by controlling what is added and removed from memory in the hidden layers. LSTMs were used by Sang Il Le et al, 2018 for time series analysis where they looked at stock price data for 10 stocks from January 1997 to December 2016. The dataset contained ﬁve attributes: open price, high price, low price, adjust close price, and volume of stocks traded. They attempted to predict one month ahead forecasts of stock returns using three models S-RNN (simple RNN), LSTM and GRU. They attempted to evaluate the predictive ability of the three models using the hit ratio, which is deﬁned as follows:

 (18)

**Figure 18. Sang Il Lee and Seong Joon Yoo, 2018 where *N* is total trading months and *Pi* is the prediction result for the trading day, deﬁned as:** **where +1 and ˆ +1 are the realized return at the last business day of month t+1 and the one-month-ahead return predicted at the last business day of month t, respectively**



The table below shows the mean and standard deviation (SD) of the hit ratios for the 10 assets and the use of the last business day and the LSTM model generates the best prediction accuracy value (0.604).

 (19)

**Figure 19. Sang Il Lee and Seong Joon Yoo, 2018**

Zhang et al 2018 focused on the CSI-300 stock index[[4]](#footnote-4) when developing a LSTM model combined with traditional multi factor analysis to build an improved multi factor stock selection model. They point out the main parameter change required in training that differs from RRN based multifactor models as the cyclical time dimension. The application and the results of the model are not clearly articulated so it is difficult to gain a full understanding of the results, however they do report “the result shows that the multi factor stock selection model based on LSTM has good profit forecasting ability and profitability.”

**3 DESIGN AND IMPLEMENTATION**

**3.1 Financial Data – Explanation of the financial data used**

The data the dissertation will be based on data supplied by financial data company Quandl. The data that will be used is called **Core US Fundamentals Data.** This data is Accounting Metrics published by the individual companies themselves, from stock exchanges where the companies are listed or metrics calculated from published data.

**Accounting Metrics**

There are six rows and 111 columns. There are six rows because of the different dimensions, dimensions refer to how the data is reported, the distinctions are outlined below.

As Reported view (AR):

* excludes restatements
* point-in-time view with data time-indexed to the date the form 10 regulatory filing was submitted to the SEC
* presents data for the latest reporting period at that filing date
* may include multiple observations in a quarter if more than one filing is made during the quarter
* on limited occasion may not have any observations in a particular quarter. Sometimes companies are delayed in reporting for up to 18 months. On such occasions they may report multiple documents on the same date to catch up, in which case these datasets will only provide date for the most recent reporting period.
* typically suitable for back-testing

Most-Recent Reported view (MR):

* includes restatements
* time indexed to the financial/report period
* presents the most recently reported data for that reporting period
* typically suitable for assessing business performance after restatements for mergers/divestitures

The Quandl documentation states that ARY (as-reported annual) dimension is typically suitable for back-testing, which is what the dissertation is based on.

These are annual observations of one year duration. Using the as-reported dimension will more closely align with the date that information was disseminated to the market, and the corresponding market impact.

In addition there are 3 time dimensions:

Annual (Y): Annual observations of one year duration

Trailing Twelve Months (T): Quarterly observations of one year duration

Quarterly (Q): Quarterly observations of quarterly duration (available only for US domestic companies, unavailable for foreign companies)

These Accounting Metrics are only available on a monthly basis.

Below is an example of the latest Accounting Metrics for the stock **Apple (APPL)**

|  | **ticker** | **dimension** | **calendardate** | **datekey** | **reportperiod** | **lastupdated** | **accoci** | **assets** | **assetsavg** | **assetsc** | **...** | **sharesbas** | **shareswa** | **shareswadil** | **sps** | **tangibles** | **taxassets** | **taxexp** | **taxliabilities** | **tbvps** | **workingcapital** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **None** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **0** | AAPL | ARQ | 2018-09-30 | 2018-11-05 | 2018-09-29 | 2018-11-05 | -3454000000 | 365725000000 | NaN | 131339000000 | ... | 4745398000 | 4801588000 | 4847547000 | 13.100 | 365725000000 | 0 | 2296000000 | 0 | 76.168 | 14473000000 |
| **1** | AAPL | ART | 2018-09-30 | 2018-11-05 | 2018-09-29 | 2018-11-05 | -3454000000 | 365725000000 | 3.723045e+11 | 131339000000 | ... | 4745398000 | 4955377000 | 5000109000 | 53.597 | 365725000000 | 0 | 13372000000 | 0 | 76.168 | 14473000000 |
| **2** | AAPL | ARY | 2018-12-31 | 2018-11-05 | 2018-09-29 | 2018-11-05 | -3454000000 | 365725000000 | 3.723045e+11 | 131339000000 | ... | 4745398000 | 4955377000 | 5000109000 | 53.597 | 365725000000 | 0 | 13372000000 | 0 | 76.168 | 14473000000 |
| **3** | AAPL | MRQ | 2018-09-30 | 2018-09-29 | 2018-09-29 | 2018-11-05 | -3454000000 | 365725000000 | NaN | 131339000000 | ... | 4829926000 | 4801588000 | 4847547000 | 13.100 | 365725000000 | 0 | 2296000000 | 0 | 76.168 | 14473000000 |
| **4** | AAPL | MRT | 2018-09-30 | 2018-09-29 | 2018-09-29 | 2018-11-05 | -3454000000 | 365725000000 | 3.723045e+11 | 131339000000 | ... | 4829926000 | 4955377000 | 5000109000 | 53.597 | 365725000000 | 0 | 13372000000 | 0 | 76.168 | 14473000000 |

**Accounting Metrics Explanation**

Below is a table with descriptions for the Accounting Metric data.

**Table** - Quandl source table

**Indicator** - Accounting Metric code

**Isfilter** - Whether the search criteria can be applied to the Accounting Metric

**Title** - Full name of the Accounting Metric

**Description** - Full description of the Accounting Metric

**Unit Type** - Unit of measurement for the Accounting Metric

A sample of the full descriptions dataset

|  | **table** | **indicator** | **isfilter** | **isprimarykey** | **title** | **description** | **unittype** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **None** |  |  |  |  |  |  |  |
| **0** | SF1 | revenue | N | N | Revenues | [Income Statement] Amount of Revenue recognized from goods sold; services rendered; insurance premiums; or other activities that constitute an earning process. Interest income for financial institutions is reported net of interest expense and provision for credit losses. | currency |
| **1** | SF1 | cor | N | N | Cost of Revenue | [Income Statement] The aggregate cost of goods produced and sold and services rendered during the reporting period. | currency |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **109** | SF1 | reportperiod | N | Y | Report Period | [Entity] The Report Period represents the end date of the fiscal period. | date (YYYY-MM-DD) |
| **110** | SF1 | lastupdated | Y | N | Last Updated Date | [Entity] Last Updated represents the last date that this database entry was updated; which is useful to users when updating their local records. | date (YYYY-MM-DD) |

**Metric List**

Metric List A full list of all the 111 metrics is below

|  | **title** | **description** |
| --- | --- | --- |
|  |  |  |
| **0** | Revenues | [Income Statement] Amount of Revenue recognized from goods sold; services rendered; insurance premiums; or other activities that constitute an earning process. Interest income for financial institutions is reported net of interest expense and provision for credit losses. |
| **1** | Cost of Revenue | [Income Statement] The aggregate cost of goods produced and sold and services rendered during the reporting period. |
| **2** | Selling General and Administrative Expense | [Income Statement] A component of [OpEx] representing the aggregate total costs related to selling a firm's product and services; as well as all other general and administrative expenses. Direct selling expenses (for example; credit; warranty; and advertising) are expenses that can be directly linked to the sale of specific products. Indirect selling expenses are expenses that cannot be directly linked to the sale of specific products; for example telephone expenses; Internet; and postal charges. General and administrative expenses include salaries of non-sales personnel; rent; utilities; communication; etc. |
| **3** | Research and Development Expense | [Income Statement] A component of [OpEx] representing the aggregate costs incurred in a planned search or critical investigation aimed at discovery of new knowledge with the hope that such knowledge will be useful in developing a new product or service. |
| **4** | Operating Expenses | [Income Statement] Operating expenses represents the total expenditure on [SGnA]; [RnD] and other operating expense items; it excludes [CoR]. |
| **5** | Interest Expense | [Income Statement] Amount of the cost of borrowed funds accounted for as interest expense. |
| **6** | Income Tax Expense | [Income Statement] Amount of current income tax expense (benefit) and deferred income tax expense (benefit) pertaining to continuing operations. |
| **7** | Net Loss Income from Discontinued Operations | [Income Statement] Amount of loss (income) from a disposal group; net of income tax; reported as a separate component of income. |
| **8** | Consolidated Income | [Income Statement] The portion of profit or loss for the period; net of income taxes; which is attributable to the consolidated entity; before the deduction of [NetIncNCI]. |
| **9** | Net Income to Non-Controlling Interests | [Income Statement] The portion of income which is attributable to non-controlling interest shareholders; subtracted from [ConsolInc] in order to obtain [NetInc]. |
| **10** | Net Income | [Income Statement] The portion of profit or loss for the period; net of income taxes; which is attributable to the parent after the deduction of [NetIncNCI] from [ConsolInc]; and before the deduction of [PrefDivIS]. |
| **11** | Preferred Dividends Income Statement Impact | [Income Statement] Income statement item reflecting dividend payments to preferred stockholders. Subtracted from Net Income to Parent [NetInc] to obtain Net Income to Common Stockholders [NetIncCmn]. |
| **12** | Net Income Common Stock | [Income Statement] The amount of net income (loss) for the period due to common shareholders. Typically differs from [NetInc] to the parent entity due to the deduction of [PrefDivIS]. |
| **13** | Earnings per Basic Share | [Income Statement] Earnings per share as calculated and reported by the company. Approximates to the amount of [NetIncCmn] for the period per each [SharesWA]. |
| **14** | Earnings per Diluted Share | [Income Statement] Earnings per diluted share as calculated and reported by the company. Approximates to the amount of [NetIncCmn] for the period per each [SharesWADil]. |
| **15** | Weighted Average Shares | [Income Statement] The weighted average number of shares or units issued and outstanding that are used by the company to calculate [EPS]; determined based on the timing of issuance of shares or units in the period. |
| **16** | Weighted Average Shares Diluted | [Income Statement] The weighted average number of shares or units issued and outstanding that are used by the company to calculate [EPSDil]; determined based on the timing of issuance of shares or units in the period. |
| **17** | Capital Expenditure | [Cash Flow Statement] A component of [NCFI] representing the net cash inflow (outflow) associated with the acquisition & disposal of long-lived; physical & intangible assets that are used in the normal conduct of business to produce goods and services and are not intended for resale. Includes cash inflows/outflows to pay for construction of self-constructed assets & software. |
| **18** | Net Cash Flow - Business Acquisitions and Disposals | [Cash Flow Statement] A component of [NCFI] representing the net cash inflow (outflow) associated with the acquisition & disposal of businesses; joint-ventures; affiliates; and other named investments. |
| **19** | Net Cash Flow - Investment Acquisitions and Disposals | [Cash Flow Statement] A component of [NCFI] representing the net cash inflow (outflow) associated with the acquisition & disposal of investments; including marketable securities and loan originations. |
| **20** | Net Cash Flow from Financing | [Cash Flow Statement] A component of [NCF] representing the amount of cash inflow (outflow) from financing activities; from continuing and discontinued operations. Principal components of financing cash flow are: issuance (purchase) of equity shares; issuance (repayment) of debt securities; and payment of dividends & other cash distributions. |
| **21** | Issuance (Repayment) of Debt Securities | [Cash Flow Statement] A component of [NCFF] representing the net cash inflow (outflow) from issuance (repayment) of debt securities. |
| **22** | Issuance (Purchase) of Equity Shares | [Cash Flow Statement] A component of [NCFF] representing the net cash inflow (outflow) from common equity changes. Includes additional capital contributions from share issuances and exercise of stock options; and outflow from share repurchases. |
| **23** | Payment of Dividends & Other Cash Distributions | [Cash Flow Statement] A component of [NCFF] representing dividends and dividend equivalents paid on common stock and restricted stock units. |
| **24** | Net Cash Flow from Investing | [Cash Flow Statement] A component of [NCF] representing the amount of cash inflow (outflow) from investing activities; from continuing and discontinued operations. Principal components of investing cash flow are: capital (expenditure) disposal of equipment [CapEx]; business (acquisitions) disposition [NCFBus] and investment (acquisition) disposal [NCFInv]. |
| **25** | Net Cash Flow from Operations | [Cash Flow Statement] A component of [NCF] representing the amount of cash inflow (outflow) from operating activities; from continuing and discontinued operations. |
| **26** | Effect of Exchange Rate Changes on Cash | [Cash Flow Statement] A component of Net Cash Flow [NCF] representing the amount of increase (decrease) from the effect of exchange rate changes on cash and cash equivalent balances held in foreign currencies. |
| **27** | Net Cash Flow / Change in Cash & Cash Equivalents | [Cash Flow Statement] Principal component of the cash flow statement representing the amount of increase (decrease) in cash and cash equivalents. Includes [NCFO]; investing [NCFI] and financing [NCFF] for continuing and discontinued operations; and the effect of exchange rate changes on cash [NCFX]. |
| **28** | Share Based Compensation | [Cash Flow Statement] A component of [NCFO] representing the total amount of noncash; equity-based employee remuneration. This may include the value of stock or unit options; amortization of restricted stock or units; and adjustment for officers' compensation. As noncash; this element is an add back when calculating net cash generated by operating activities using the indirect method. |
| **29** | Depreciation Amortization & Accretion | [Cash Flow Statement] A component of operating cash flow representing the aggregate net amount of depreciation; amortization; and accretion recognized during an accounting period. As a non-cash item; the net amount is added back to net income when calculating cash provided by or used in operations using the indirect method. |
| **30** | Total Assets | [Balance Sheet] Sum of the carrying amounts as of the balance sheet date of all assets that are recognized. Major components are [CashnEq]; [Investments];[Intangibles]; [PPNENet];[TaxAssets] and [Receivables]. |
| **31** | Cash and Equivalents | [Balance Sheet] A component of [Assets] representing the amount of currency on hand as well as demand deposits with banks or financial institutions. |
| **32** | Investments | [Balance Sheet] A component of [Assets] representing the total amount of marketable and non-marketable securties; loans receivable and other invested assets. |
| **33** | Investments Current | [Balance Sheet] The current portion of [Investments]; reported if the company operates a classified balance sheet that segments current and non-current assets. |
| **34** | Investments Non-Current | [Balance Sheet] The non-current portion of [Investments]; reported if the company operates a classified balance sheet that segments current and non-current assets. |
| **35** | Deferred Revenue | [Balance Sheet] A component of [Liabilities] representing the carrying amount of consideration received or receivable on potential earnings that were not recognized as revenue; including sales; license fees; and royalties; but excluding interest income. |
| **36** | Deposit Liabilities | [Balance Sheet] A component of [Liabilities] representing the total of all deposit liabilities held; including foreign and domestic; interest and noninterest bearing. May include demand deposits; saving deposits; Negotiable Order of Withdrawal and time deposits among others. |
| **37** | Property Plant & Equipment Net | [Balance Sheet] A component of [Assets] representing the amount after accumulated depreciation; depletion and amortization of physical assets used in the normal conduct of business to produce goods and services and not intended for resale. |
| **38** | Inventory | [Balance Sheet] A component of [Assets] representing the amount after valuation and reserves of inventory expected to be sold; or consumed within one year or operating cycle; if longer. |
| **39** | Tax Assets | [Balance Sheet] A component of [Assets] representing tax assets and receivables. |
| **40** | Trade and Non-Trade Receivables | [Balance Sheet] A component of [Assets] representing trade and non-trade receivables. |
| **41** | Trade and Non-Trade Payables | [Balance Sheet] A component of [Liabilities] representing trade and non-trade payables. |
| **42** | Goodwill and Intangible Assets | [Balance Sheet] A component of [Assets] representing the carrying amounts of all intangible assets and goodwill as of the balance sheet date; net of accumulated amortization and impairment charges. |
| **43** | Total Liabilities | [Balance Sheet] Sum of the carrying amounts as of the balance sheet date of all liabilities that are recognized. Principal components are [Debt]; [DeferredRev]; [Payables];[Deposits]; and [TaxLiabilities]. |
| **44** | Shareholders Equity | [Balance Sheet] A principal component of the balance sheet; in addition to [Liabilities] and [Assets]; that represents the total of all stockholders' equity (deficit) items; net of receivables from officers; directors; owners; and affiliates of the entity which are attributable to the parent. |
| **45** | Accumulated Retained Earnings (Deficit) | [Balance Sheet] A component of [Equity] representing the cumulative amount of the entities undistributed earnings or deficit. May only be reported annually by certain companies; rather than quarterly. |
| **46** | Accumulated Other Comprehensive Income | [Balance Sheet] A component of [Equity] representing the accumulated change in equity from transactions and other events and circumstances from non-owner sources; net of tax effect; at period end. Includes foreign currency translation items; certain pension adjustments; unrealized gains and losses on certain investments in debt and equity securities. |
| **47** | Current Assets | [Balance Sheet] The current portion of [Assets]; reported if a company operates a classified balance sheet that segments current and non-current assets. |
| **48** | Assets Non-Current | [Balance Sheet] Amount of non-current assets; for companies that operate a classified balance sheet. Calculated as the different between Total Assets [Assets] and Current Assets [AssetsC]. |
| **49** | Current Liabilities | [Balance Sheet] The current portion of [Liabilities]; reported if the company operates a classified balance sheet that segments current and non-current liabilities. |
| **50** | Liabilities Non-Current | [Balance Sheet] The non-current portion of [Liabilities]; reported if the company operates a classified balance sheet that segments current and non-current liabilities. |
| **51** | Tax Liabilities | [Balance Sheet] A component of [Liabilities] representing outstanding tax liabilities. |
| **52** | Total Debt | [Balance Sheet] A component of [Liabilities] representing the total amount of current and non-current debt owed. Includes secured and unsecured bonds issued; commercial paper; notes payable; credit facilities; lines of credit; capital lease obligations; and convertible notes. |
| **53** | Debt Current | [Balance Sheet] The current portion of [Debt]; reported if the company operates a classified balance sheet that segments current and non-current liabilities. |
| **54** | Debt Non-Current | [Balance Sheet] The non-current portion of [Debt] reported if the company operates a classified balance sheet that segments current and non-current liabilities. |
| **55** | Earnings before Tax | [Metrics] Earnings Before Tax is calculated by adding [TaxExp] back to [NetInc]. |
| **56** | Earning Before Interest & Taxes (EBIT) | [Income Statement] Earnings Before Interest and Tax is calculated by adding [TaxExp] and [IntExp] back to [NetInc]. |
| **57** | Earnings Before Interest Taxes & Depreciation Amortization (EBITDA) | [Metrics] EBITDA is a non-GAAP accounting metric that is widely used when assessing the performance of companies; calculated by adding [DepAmor] back to [EBIT]. |
| **58** | Foreign Currency to USD Exchange Rate | [Metrics] The exchange rate used for the conversion of foreign currency to USD for non-US companies that do not report in USD. |
| **59** | Shareholders Equity (USD) | [Balance Sheet] [Equity] in USD; converted by [FXUSD]. |
| **60** | Earnings per Basic Share (USD) | [Income Statement] [EPS] in USD; converted by [FXUSD]. |
| **61** | Revenues (USD) | [Income Statement] [Revenue] in USD; converted by [FXUSD]. |
| **62** | Net Income Common Stock (USD) | [Income Statement] [NetIncCmn] in USD; converted by [FXUSD]. |
| **63** | Cash and Equivalents (USD) | [Balance Sheet] [CashnEq] in USD; converted by [FXUSD]. |
| **64** | Total Debt (USD) | [Balance Sheet] [Debt] in USD; converted by [FXUSD]. |
| **65** | Earning Before Interest & Taxes (USD) | [Income Statement] [EBIT] in USD; converted by [FXUSD]. |
| **66** | Earnings Before Interest Taxes & Depreciation Amortization (USD) | [Metrics] [EBITDA] in USD; converted by [FXUSD]. |
| **67** | Shares (Basic) | [Entity] The number of shares or other units outstanding of the entity's capital or common stock or other ownership interests; as stated on the cover of related periodic report (10-K/10-Q); after adjustment for stock splits. |
| **68** | Dividends per Basic Common Share | [Income Statement] Aggregate dividends declared during the period for each split-adjusted share of common stock outstanding. Includes spinoffs where identified. |
| **69** | Share Factor | [Entity] Share factor is a multiplicant in the calculation of [MarketCap] and is used to adjust for: American Depository Receipts (ADRs) that represent more or less than 1 underlying share; and; companies which have different earnings share for different share classes (eg Berkshire Hathaway - BRKB). |
| **70** | Market Capitalization | [Metrics] Represents the product of [SharesBas]; [Price] and [ShareFactor]. |
| **71** | Enterprise Value | [Metrics] Enterprise value is a measure of the value of a business as a whole; calculated as [MarketCap] plus [DebtUSD] minus [CashnEqUSD]. |
| **72** | Invested Capital | [Metrics] Invested capital is an input into the calculation of [ROIC]; and is calculated as: [Debt] plus [Assets] minus [Intangibles] minus [CashnEq] minus [LiabilitiesC]. Please note this calculation method is subject to change. |
| **73** | Average Equity | [Metrics] Average equity value for the period used in calculation of [ROE]; derived from [Equity]. |
| **74** | Average Assets | [Metrics] Average asset value for the period used in calculation of [ROE] and [ROA]; derived from [Assets]. |
| **75** | Invested Capital Average | [Metrics] Average invested capital value for the period used in the calculation of [ROIC]; and derived from [InvCap]. Invested capital is an input into the calculation of [ROIC]; and is calculated as: [Debt] plus [Assets] minus [Intangibles] minus [CashnEq] minus [LiabilitiesC]. Please note this calculation method is subject to change. |
| **76** | Tangible Asset Value | [Metrics] The value of tangibles assets calculated as the difference between [Assets] and [Intangibles]. |
| **77** | Return on Average Equity | [Metrics] Return on equity measures a corporation's profitability by calculating the amount of [NetIncCmn] returned as a percentage of [EquityAvg]. |
| **78** | Return on Average Assets | [Metrics] Return on assets measures how profitable a company is [NetIncCmn] relative to its total assets [AssetsAvg]. |
| **79** | Free Cash Flow | [Metrics] Free Cash Flow is a measure of financial performance calculated as [NCFO] minus [CapEx]. |
| **80** | Return on Invested Capital | [Metrics] Return on Invested Capital is ratio estimated by dividing [EBIT] by [InvCapAvg]. [InvCap] is calculated as: [Debt] plus [Assets] minus [Intangibles] minus [CashnEq] minus [LiabilitiesC]. Please note this calculation method is subject to change. |
| **81** | Gross Profit | [Income Statement] Aggregate revenue [Revenue] less cost of revenue [CoR] directly attributable to the revenue generation activity. |
| **82** | Operating Income | [Income Statement] Operating income is a measure of financial performance before the deduction of [IntExp]; [TaxExp] and other Non-Operating items. It is calculated as [GP] minus [OpEx]. |
| **83** | Gross Margin | [Metrics] Gross Margin measures the ratio between a company's [GP] and [Revenue]. |
| **84** | Profit Margin | [Metrics] Measures the ratio between a company's [NetIncCmn] and [Revenue]. |
| **85** | EBITDA Margin | [Metrics] Measures the ratio between a company's [EBITDA] and [Revenue]. |
| **86** | Return on Sales | [Metrics] Return on Sales is a ratio to evaluate a company's operational efficiency; calculated by dividing [EBIT] by [Revenue]. ROS is often a component of DuPont ROE analysis. |
| **87** | Asset Turnover | [Metrics] Asset turnover is a measure of a firms operating efficiency; calculated by dividing [Revenue] by [AssetsAVG]. Often a component of DuPont ROE analysis. |
| **88** | Payout Ratio | [Metrics] The percentage of earnings paid as dividends to common stockholders. Calculated by dividing [DPS] by [EPSUSD]. |
| **89** | Enterprise Value over EBITDA | [Metrics] Measures the ratio between [EV] and [EBITDAUSD]. |
| **90** | Enterprise Value over EBIT | [Metrics] Measures the ratio between [EV] and [EBITUSD]. |
| **91** | Price Earnings (Damodaran Method) | [Metrics] Measures the ratio between [MarketCap] and [NetIncCmnUSD] |
| **92** | Price to Earnings Ratio | [Metrics] An alternative to [PE] representing the ratio between [Price] and [EPSUSD]. |
| **93** | Sales per Share | [Metrics] Sales per Share measures the ratio between [RevenueUSD] and [SharesWA]. |
| **94** | Price to Sales Ratio | [Metrics] An alternative calculation method to [PS]; that measures the ratio between a company's [Price] and it's [SPS]. |
| **95** | Price Sales (Damodaran Method) | [Metrics] Measures the ratio between [MarketCap] and [RevenueUSD]. |
| **96** | Price to Book Value | [Metrics] Measures the ratio between [MarketCap] and [EquityUSD]. |
| **97** | Debt to Equity Ratio | [Metrics] Measures the ratio between [Liabilities] and [Equity]. |
| **98** | Dividend Yield | [Metrics] Dividend Yield measures the ratio between a company's [DPS] and its [Price]. |
| **99** | Current Ratio | [Metrics] The ratio between [AssetsC] and [LiabilitiesC]; for companies that operate a classified balance sheet. |
| **100** | Working Capital | [Metrics] Working capital measures the difference between [AssetsC] and [LiabilitiesC]. |
| **101** | Free Cash Flow per Share | [Metrics] Free Cash Flow per Share is a valuation metric calculated by dividing [FCF] by [SharesWA]. |
| **102** | Book Value per Share | [Metrics] Measures the ratio between [Equity] and [SharesWA]. |
| **103** | Tangible Assets Book Value per Share | [Metrics] Measures the ratio between [Tangibles] and [SharesWA]. |
| **104** | Share Price (Adjusted Close) | [Entity] The price per common share adjusted for stock splits but not adjusted for dividends; used in the computation of [PE1]; [PS1]; [DivYield] and [SPS]. |
| **105** | Ticker Symbol | [Entity] The ticker is a unique identifer for an issuer in the database. Where a ticker contains a . or a - this is removed from the ticker we use. For example BRK.B is BRKB in this dataset. We include the BRK.B ticker in the Related Tickers field of the ticker listing. Where a company is delisted and the ticker is recycled; we utilise that ticker for the currently active company and append a number to the ticker of the delisted company. For example GM represents the current actively traded General Motors entity; and GM1 represents the entity that filed for bankruptcy in 2009. Where we have identified that multiple classes of shares exist for a company; we include the alternative share classes in the Related Tickers field of the ticker listing. For example we provide data for GOOGL; an... |
| **106** | Dimension | [Entity] The dimension field allows you to take different dimensional views of data over time. ARQ: Quarterly; excluding restatements; MRQ: Quarterly; including restatements; ARY: annual; excluding restatements; MRY: annual; including restatements; ART: trailing-twelve-months; excluding restatements; MRT: trailing-twelve-months; including restatements. |
| **107** | Calendar Date | [Entity] The Calendar Date represents the normalized [ReportPeriod]. This provides a common date to query for which is necessary due to irregularity in report periods across companies. For example; if the report period is "2015-09-26"; the calendar date will be "2015-09-30" for quarterly and trailing-twelve-month dimensions (ARQ;MRQ;ART;MRT); and "2015-12-31" for annual dimensions (ARY;MRY). We also employ offsets in order to maximise comparability of the period across companies. For example consider two companies: one with a quarter ending on 2018-07-24; and the other with a quarter ending on 2018-06-28. A naive normalization process would assign these to differing calendar quarters of 2018-09-30 and 2018-06-30 respectively. However, we assign these both to the 2018-06-30 calendar qua... |
| **108** | Date Key | [Entity] The Date Key represents the SEC filing date for AR dimensions (ARQ;ART;ARY); and the [REPORTPERIOD] for MR dimensions (MRQ;MRT;MRY). In addition; this is the observation date used for [Price] based data such as [MarketCap]; [Price] and [PE]. |
| **109** | Report Period | [Entity] The Report Period represents the end date of the fiscal period. |
| **110** | Last Updated Date | [Entity] Last Updated represents the last date that this database entry was updated; which is useful to users when updating their local records. |

**Daily Accounting Metrics**

Some Accounting Metrics are available daily as well as monthly and are provided on a separate table.

Below are the metrics provided daily for Apple.

|  | **ticker** | **date** | **lastupdated** | **ev** | **evebit** | **evebitda** | **marketcap** | **pb** | **pe** | **ps** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **None** |  |  |  |  |  |  |  |  |  |  |
| **0** | AAPL | 2019-01-17 | 2019-01-17 | 828187.7 | 11.4 | 9.9 | 739617.7 | 6.9 | 12.4 | 2.8 |
| **1** | AAPL | 2019-01-16 | 2019-01-16 | 823822.0 | 11.3 | 9.8 | 735252.0 | 6.9 | 12.4 | 2.8 |

**Pricing**

Pricing data is available from another Quandl table - Sharadar Equity Prices,SEP . Updated daily, this database provides End-Of-Day (EOD) price data with coverage corresponding to Sharadar's Fundamental dataset, SF1.

Below is an example of the latest pricing information for Apple.

|  | **ticker** | **date** | **open** | **high** | **low** | **close** | **volume** | **dividends** | **closeunadj** | **lastupdated** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **None** |  |  |  |  |  |  |  |  |  |  |
| **0** | AAPL | 2019-01-17 | 154.2 | 157.66 | 153.26 | 155.86 | 29336202.0 | 0.0 | 155.86 | 2019-01-17 |

This data can be a date range, below are Apple prices from the begging of the year

|  | **ticker** | **date** | **open** | **high** | **low** | **close** | **volume** | **dividends** | **closeunadj** | **lastupdated** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **None** |  |  |  |  |  |  |  |  |  |  |
| **0** | AAPL | 2019-01-17 | 154.20 | 157.66 | 153.26 | 155.86 | 29336202.0 | 0.0 | 155.86 | 2019-01-17 |
| **1** | AAPL | 2019-01-16 | 153.08 | 155.88 | 153.00 | 154.94 | 29626386.0 | 0.0 | 154.94 | 2019-01-16 |
| **2** | AAPL | 2019-01-15 | 150.27 | 153.39 | 150.05 | 153.07 | 28227737.0 | 0.0 | 153.07 | 2019-01-15 |
| **3** | AAPL | 2019-01-14 | 150.85 | 151.27 | 149.22 | 150.00 | 31805772.0 | 0.0 | 150.00 | 2019-01-14 |
| **4** | AAPL | 2019-01-11 | 152.88 | 153.70 | 151.51 | 152.29 | 26711076.0 | 0.0 | 152.29 | 2019-01-11 |
| **5** | AAPL | 2019-01-10 | 152.50 | 153.97 | 150.86 | 153.80 | 35065010.0 | 0.0 | 153.80 | 2019-01-10 |
| **6** | AAPL | 2019-01-09 | 151.29 | 154.53 | 149.63 | 153.31 | 44641758.0 | 0.0 | 153.31 | 2019-01-09 |
| **7** | AAPL | 2019-01-08 | 149.56 | 151.82 | 148.52 | 150.75 | 39977341.0 | 0.0 | 150.75 | 2019-01-08 |
| **8** | AAPL | 2019-01-07 | 148.70 | 148.83 | 145.90 | 147.93 | 54308097.0 | 0.0 | 147.93 | 2019-01-07 |
| **9** | AAPL | 2019-01-04 | 144.53 | 148.55 | 143.80 | 148.26 | 56884249.0 | 0.0 | 148.26 | 2019-01-04 |
| **10** | AAPL | 2019-01-03 | 143.98 | 145.72 | 142.00 | 142.19 | 90466553.0 | 0.0 | 142.19 | 2019-01-03 |
| **11** | AAPL | 2019-01-02 | 154.89 | 158.85 | 154.23 | 157.92 | 31738194.0 | 0.0 | 157.92 | 2019-01-02 |

**Pricing Data Explanation**

Below is a table with descriptions and unit types for all the price data types. Generally, 'close' price is used for projects such as this

|  | **table** | **indicator** | **isfilter** | **isprimarykey** | **title** | **description** | **unittype** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **None** |  |  |  |  |  |  |  |
| **0** | SEP | ticker | Y | Y | Ticker Symbol | The ticker is a unique identifer for an issuer in the database. Where a ticker contains a "." or a "-" this is removed from the ticker. For example BRK.B is BRKB. We include the BRK.B ticker in the Related Tickers field. Where a company is delisted and the ticker is recycled; we use that ticker for the currently active company and append a number to the ticker of the delisted company. eg GM is the current actively traded entity; & GM1 is the entity that filed for bankruptcy in 2009. | text |
| **1** | SEP | date | Y | Y | Price Date | The trade date of the price observations. | date (YYYY-MM-DD) |
| **2** | SEP | open | N | N | Open Price - Split Adjusted | The opening share price, adjusted for stock splits and stock dividends. | USD/share |
| **3** | SEP | high | N | N | High Price - Split Adjusted | The high share price, adjusted for stock splits and stock dividends. | USD/share |
| **4** | SEP | low | N | N | Low Price - Split Adjusted | The low share price, adjusted for stock splits and stock dividends. | USD/share |
| **5** | SEP | close | N | N | Close Price - Split Adjusted | The open share closing, adjusted for stock splits and stock dividends. | USD/share |
| **6** | SEP | volume | N | N | Volume - Split Adjusted | The traded volume, adjusted for stock splits and stock dividends. | numeric |
| **7** | SEP | dividends | N | N | Dividends per Share - Split Adjusted | The dividends per share unit, adjusted for stock splits and stock dividends. Includes spinoffs where identified. | USD/share |
| **8** | SEP | closeunadj | N | N | Close Price - Unadjusted | The closing share price, not adjusted for stock splits and stock dividends. | USD/share |
| **9** | SEP | lastupdated | Y | N | Last Updated Date | The last date at which this line item was updated, typically used to filter date to be retrieved for syncing to local records. | date (YYYY-MM-DD) |

**3.2 Python packages – Explanation of the Python packages that will be used to implement RNNs**

**3.2.1 Implementation**

The implementation of the RNNs was carried out using the Python library Keras. Keras runs on top of numerical computation and large-scale machine learning libraries TensorFlow, CNTK or Theano.

Gathering the data was done by accessing Quandl APIs, for price data and accounting metric data. Price data needed to be manipulated using the Pandas library into firstly monthly price from a daily pricing dataset then changed into monthly returns. Returns being the change (positive or negative) in the monthly share price.

Factor data (accounting metrics) was also returned from a Quandl API. As not all factor data is available on a monthly basis quarterly data was maintained until new data became available. Returns data is then added to the factor data. After running a correlation matrix and factors with a correlation of > 0.95 are dropped to reduce model complexity.

As Keras only accepts data in the form of arrays the Pandas dataframe is amended to a Numpy array. As the factor data is composed of different scales these values are normalised so that they take small values on a similar scale. The SKlearn module MinMax scalar is used to pass a feature range between 0 and 1 for the data.

3.2.2 Training the RNN models

A neural network implementation in Keras is based around four objects. Layers (combined into a network or model), input data and corresponding targets, loss function that defines the feedback signal used for learning and the optimiser which determines how learning proceeds. A layer is a data-processing module that takes as input one or more tensors and outputs the same. Some layers are stateless but more often that have a state: the layer’s weights, one or several tensors learned with stochastic gradient descent, which together contain the network’s knowledge.

Different layers are appropriate for different tensor formats and different types of data processing. For instance, simple vector data, stored in 2D tensors of shape (samples, features), is often processed by densely connected layers, also called fully connected or dense layers (the Dense class in Keras). Sequence data, stored in 3D tensors of shape (samples, timesteps, features), is typically processed by recurrent layers such as GRU or LSTM layers. Image data, stored in 4D tensors, is usually processed by 2D convolution layers (Conv2D).

The project dataset is a timeseries dataset therefore a 3D tensor shape is required.

After the architecture is defined there are still choices to be made about the loss function and the optimiser. As this is fundamentally a regression the chosen loss function used can be mean absolute error (mae) or mean squared error (mse). Mae being the absolute value of the difference between the predictions and the targets and mse being the square of the difference. The default in the Keras documentation[[5]](#footnote-5) is mse and this was used at initiation of the model.

The dataset will be data for one stock ‘A’ for the time period 2009-13-31 to 2019-01-01, a matrix (36, 72) in shape. Train data will be the first 32 instances of the data set. Initially a very small network was used consisting of one hidden layer with 32 units, units being the subunits of the layer. The design of neural networks is often described as more of an art than a science[[6]](#footnote-6), a process of trial and error but also adopting know best practices, was the approach taken in the design stage. Small networks are often seen as a way of mitigating overfitting. The network ends with a single unit and no activation, a linear layer. This will single layer will be maintained for all architectures used, as this is a regression problem and we are trying to predict a single continuous value. Applying an activation function would constrain the range the output could take. The optimiser is for the initiation of the model is ‘rmsprop’, the recommend optimiser for RNNs in the Keras documentation[[7]](#footnote-7). The implementation also uses the mean absolute error as the metric to be monitored by the model. To validate the network K-fold cross validation is used splitting the data into K partitions, instantiating the K identical models and training each one on K-1 partitions while evaluation is based on the remaining partition. Taking the average of the scores returns a single measurement. The epochs, the length of time the model is trained for, is set to 100 to restrain processor requirements. Further parameters that will be adjusted are the number and size of the hidden layers. After the parameters have been amended a model is trained on all the training data and its performance evaluated on the test data.

A simple machine learning model will be used as a baseline for comparison with more complicated models. This is created with a small densely connect layer, a simple feed forward network for comparison with more complicated and computationally expensive models such as RNNs.

The next step in the implementation process was to move from just a densely connected neural networks to recurring neural networks, adding memory to the network. RNNs process sequences by iterating through the sequence elements and maintaining a state containing information relative to what it has seen so far. Keras has a SimpleRNN layer which processes batches of sequences and takes the inputs of shape (batch size, timesteps, input features), rather than just (timesteps, input features) the 3D rather than the 2D mentioned above.

A SimpleRNN layer was used in the next, with changes to the numbers of layers and the units the parameters that could be measured. As is captured in the main literature on Keras simple RNNs are generally too simplistic to be of real use[[8]](#footnote-8). SimpleRNN has a major issue: although it should theoretically be able to retain at time*t* information about inputs seen many timesteps before, in practice, such long-term dependencies are impossible to learn. This is due to the vanishing gradient problem.

GRU and LSTM layers are designed to solve this problem, and these are applied next. Keras has an implementations for both. Again, the parameters the parameters for these can be amended. Tuning these parameters is captured in testing results.

**4 TESTING AND RESULTS**

**4.1 Comparison with accuracy metrics**

Evaluation of all models initial will be done with 4-fold cross validation with 100 epochs with mean absolute error (mae) as the validation metric, the mean for these folds is calculated. The mae is then calculated on the entire test data set providing a final test mae score.

To determine a baseline for testing, a simple small densely connected network was created.

Testing results for the Dense layer:

|  |  |
| --- | --- |
| **Model Layer** | Dense |
| **Number of Layers** | 1 |
| **Number of Units** | 32 |
| **Optimiser** | rmsprop |
| **Loss** | mse |
| **Metrics** | mae |
| **Validation** |  |
| **Fold** |  |
| **1** | 0.343737632 |
| **2** | 0.134157419 |
| **3** | 0.175569564 |
| **4** | 0.177201182 |
| **Validation Mean** | 0.207666449 |
| **Test Score** | **0.108901992** |

Testing results for the Simple RNN layer:

|  |  |
| --- | --- |
| **Model Layer** | Simple RNN |
| **Number of Layers** | 1 |
| **Number of Units** | 32 |
| **Optimiser** | rmsprop |
| **Loss** | mse |
| **Metrics** | mae |
| **Validation** |  |
| **Fold** |  |
| **1** | 0.287335515 |
| **2** | 0.236223578 |
| **3** | 0.156718925 |
| **4** | 0.206536397 |
| **Validation Mean** | 0.221703604 |
| **Test Score** | **0.302181363** |

Testing results for the GRU layer:

|  |  |
| --- | --- |
| **Model Layer** | GRU |
| **Number of Layers** | 1 |
| **Number of Units** | 32 |
| **Optimiser** | rmsprop |
| **Loss** | mse |
| **Metrics** | mae |
| **Validation** |  |
| **Fold** |  |
| **1** | 0.284213245 |
| **2** | 0.147503138 |
| **3** | 0.100005403 |
| **4** | 0.151451439 |
| **Validation Mean** | 0.170793306 |
| **Test Score** | **0.102192424** |

Testing results for the LSTM layer:

|  |  |
| --- | --- |
| **Model Layer** | LSTM |
| **Number of Layers** | 1 |
| **Number of Units** | 32 |
| **Optimiser** | rmsprop |
| **Loss** | mse |
| **Metrics** | mae |
| **Validation** |  |
| **Fold** |  |
| **1** | 0.282652259 |
| **2** | 0.131771699 |
| **3** | 0.098661974 |
| **4** | 0.125670373 |
| **Validation Mean** | 0.159689076 |
| **Test Score** | **0.096809559** |

Test score results of the different layers.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | MAE Score | Change from Baseline | % Change from Baseline |
| LSTM (1) | 0.09680956 | -0.01209243 | -11.1 |
| GRU (1) | 0.10219242 | -0.00670957 | -6.2 |
| Baseline | 0.10890199 | 0 | 0.0 |
| Simple RNN (1) | 0.30218136 | 0.19327937 | 177.5 |

For further analysis major parameters that are configurable within the architectures are number of layers and number of units. Changing the layers to two (excluding the output layer) had the following effects on the test scores, retaining all other model metrics. (number of layers)

|  |  |  |  |
| --- | --- | --- | --- |
| Model | MAE Score | Change from Baseline | % Change from Baseline |
| LSTM (1) | 0.09680956 | -0.01209243 | -11.1 |
| GRU (1) | 0.10219242 | -0.00670957 | -6.2 |
| Simple RNN (2) | 0.1022668 | -0.00663519 | -6.1 |
| LSTM (2) | 0.10668621 | -0.00221578 | -2.0 |
| Baseline | 0.10890199 | 0 | 0.0 |
| GRU (2) | 0.13110155 | 0.02219956 | 20.4 |
| Simple RNN (1) | 0.30218136 | 0.19327937 | 177.5 |

Maintaining the single layer on the GRU and LSTM model the number of units can be adjusted. The results of amending the layers are shown below.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | MAE Score | Change from Baseline | % Change from Baseline |
| LSTM (32) | 0.096809559 | -0.012092431 | -11.10395818 |
| GRU (32) | 0.102192424 | -0.006709566 | -6.161104827 |
| Baseline | 0.10890199 | 0 | 0 |
| GRU (33) | 0.133968711 | 0.025066721 | 23.01768857 |
| LSTM (36) | 0.138162658 | 0.029260668 | 26.86880893 |
| GRU (30) | 0.152388826 | 0.043486836 | 39.93208573 |
| GRU (31) | 0.156469032 | 0.047567042 | 43.67876305 |
| GRU (42) | 0.157975078 | 0.049073088 | 45.06169963 |
| LSTM (25) | 0.158706248 | 0.049804258 | 45.73310167 |
| LSTM (42) | 0.159598216 | 0.050696226 | 46.55215797 |
| LSTM (33) | 0.161811277 | 0.052909287 | 48.58431629 |
| GRU (25) | 0.16665864 | 0.05775665 | 53.03544031 |
| LSTM (31) | 0.173270687 | 0.064368697 | 59.10699838 |
| GRU (36) | 0.176900193 | 0.067998203 | 62.43981684 |
| LSTM (30) | 0.177981168 | 0.069079178 | 63.4324295 |

With the major parameters reviewed testing now looked to amending the loss function. The other loss functions available for regression problems are mean squared log error (msle) and mean absolute error (mae). Msle has the effect of relaxing the punishing effect of large differences in large predicted values. When predicting a large value, you may not want to punish a model as heavily as mean squared error. Mae is a loss function that is generally more robust to outliers, large or small values far from the mean value.  When using these for GRU and LSTM single layer models with 32 units the results for the MAE were as below.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | MAE Score | Change from Baseline | % Change from Baseline |
| LSTM (mse) | 0.096809559 | -0.012092431 | -11.10395818 |
| GRU (mse) | 0.102192424 | -0.006709566 | -6.161104827 |
| Baseline | 0.10890199 | 0 | 0 |
| GRU (mae) | 0.134331852 | 0.025429862 | 23.35114556 |
| LSTM (mae) | 0.13441053 | 0.02550854 | 23.42339229 |
| GRU (msle) | 0.155538872 | 0.046636882 | 42.82463698 |
| LSTM (msle) | 0.441106677 | 0.332204687 | 305.0492347 |

Another parameter is the optimiser. Below are all the optimisers available within Keras.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | MAE Score | Change from Baseline | % Change from Baseline |
| LSTM RMSprop | 0.096809559 | -0.012092431 | -11.10395818 |
| GRU RMSprop | 0.102192424 | -0.006709566 | -6.161104827 |
| Baseline | 0.10890199 | 0 | 0 |
| LSTM SGD | 0.115189731 | 0.006287741 | 5.773761454 |
| LSTM Nadam | 0.116197966 | 0.007295976 | 6.69958009 |
| LSTM Adagrad | 0.131034791 | 0.022132801 | 20.32359691 |
| GRU Adamax | 0.137720406 | 0.028818416 | 26.4627084 |
| LSTM Adadelta | 0.144968882 | 0.036066892 | 33.11867139 |
| LSTM Adamax | 0.157864362 | 0.048962372 | 44.96003425 |
| GRU Adagrad | 0.159108102 | 0.050206112 | 46.10210734 |
| GRU Nadam | 0.165420949 | 0.056518959 | 51.89892256 |
| LSTM Adam | 0.173822135 | 0.064920145 | 59.61336862 |
| GRU Adadelta | 0.188970447 | 0.080068457 | 73.52340998 |
| GRU SGD | 0.207090199 | 0.098188209 | 90.16199703 |
| GRU Adam | 0.213645548 | 0.104743558 | 96.18149092 |

**5 ANALYSIS AND DISCUSSION**

5.1.1 Analysis of all comparisons

Testing was baselined with a simple single dense layer machine learning feed forward model with 32 units achieving a MAE score of 0.10890199. After introducing comparisons with RNNs of the same structure; single layer with 32 units the baseline model score ranked third out of the four models. Only the SimpleRNN did not beat the baseline. SimpleRNN suffering from the vanishing gradient problem (the network is unable to propagate useful information from the output end of the model back to the layers near the input end of the model) seems ill suited to analysing time series data at this stage. GRU and LSTM showed improvement on baseline. LSTM significantly, in percentage terms an 11.1% improvement, GRU a 6.2% improvement.

Adding another layer did significantly improve the SimpleRNN model. However, adding layer complexity to the GRU and LSTM did not improve scores

GRU slipped dramatically by 20.4% in relation to the baseline and adding a layer to LSTM meant the it only slightly outperformed the baseline.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | MAE Score | Change from Baseline | % Change from Baseline |
| Simple RNN (2) | 0.1022668 | -0.00663519 | -6.1 |
| LSTM (2) | 0.10668621 | -0.00221578 | -2.0 |
| Baseline | 0.10890199 | 0 | 0.0 |
| GRU (2) | 0.13110155 | 0.02219956 | 20.4 |

Although adding a layer to LSTM was still an improvement on the baseline score it did not better the one-layer GRU or LSTM models.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | MAE Score | Change from Baseline | % Change from Baseline |
| LSTM (1) | 0.09680956 | -0.01209243 | -11.1 |
| GRU (1) | 0.10219242 | -0.00670957 | -6.2 |
| Simple RNN (2) | 0.1022668 | -0.00663519 | -6.1 |
| LSTM (2) | 0.10668621 | -0.00221578 | -2.0 |
| Baseline | 0.10890199 | 0 | 0.0 |
| GRU (2) | 0.13110155 | 0.02219956 | 20.4 |
| Simple RNN (1) | 0.30218136 | 0.19327937 | 177.5 |

For the highest ranked LSTM and GRU models adding layers did not improve accuracy so the single layer was retained when adjusting the number of units in the layer. (number of units)

Adding or subtracting units also had only detrimental effects on the model, in testing all returned a score worse than the baseline and significantly so. Units of 32 do appear to be the optimal number for both GRU and LSTM layers.

Changing the loss function to alternatives for regression mean absolute error and mean squared log error similarly only had negative effects on the score, with none of the alternatives better than baseline. The recommended loss function still performed much better.

Testing with different optimisers again did not improve on the RMSprop optimiser.

5.1.2 Summary of results/ preferred model

Testing with various parameters was conclusive in that a LSTM model with a single layer, 32 units, an MSE loss function and a RMSprop optimiser is clearly the model with the lowest MAE score of all models beating the baseline with an 11.1% improvement on it. The GRU with the same parameters did significantly beat the baseline a 6.2% improvement. In testing the only other models to beat the baseline were a Simple RNN with a two-layer, 32 unit, MSE loss function and a RMSprop optimiser which had a 6.1% improvement on baseline, and a LSTM two-layer, 32 unit, MSE loss function and RMSprop optimiser with a 2% improvement.

So, it appears that the only parameter that has a significant effect on score is adding a second layer, but this again does not better the single layer LSTM and GRU models.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | MAE Score | Change from Baseline | % Change from Baseline |
| LSTM (1) | 0.09680956 | -0.01209243 | -11.1 |
| GRU (1) | 0.10219242 | -0.00670957 | -6.2 |
| Simple RNN (2) | 0.1022668 | -0.00663519 | -6.1 |
| LSTM (2) | 0.10668621 | -0.00221578 | -2.0 |
| Baseline | 0.10890199 | 0 | 0.0 |

**6 CONCLUSION**

Making generalisations on RNN and their effectiveness on predicting stock returns from accounting metrics depends entirely on the match between the structure of the data and the assumptions of the network architecture.

Processing sequences by iterating through the sequence elements and maintaining a *state* containing information relative to what it has seen so far has in this project improved prediction from a simple feed forward machine learning model. It does appear the case that when predicting time series data, a network containing a form of ‘memory’ performs best at this task than one that doesn’t. It also appears that a simpler RNN, with a single layer works better than a equivalent with the added complexity of stacked layers. Also apparent is that for this model 32 units is as little or as much that should be used in the layer.

There are three RNNs available with Keras; Simple RNN, GRU and LSTM. Both GRU and LSTM have the concept of memory built in to address the vanishing gradient issue. LSTM vastly outperformed GRU in testing.

The key difference between a GRU and an LSTM is that a GRU has two gates (reset and update gates) whereas an LSTM has three gates (namely input, output and forget gates.)  The input gate regulating how much of the new cell state to keep, the forget gate regulates how much of the existing memory to forget, and the output gate regulates how much of the cell state should be exposed to the next layers of the network. The GRU operates using a reset gate and an update gate. The reset gate sits between the previous activation and the next candidate activation to forget previous state, and the update gate decides how much of the candidate activation to use in updating the cell state. The increased regulation of memory, LSTMs control the exposure of memory content, while GRUs expose the entire cell state to other units in the network of the LSTM has a dramatic impact on predicting stock returns using accounting metric data. It would be good to expand the universe of stock metric data to see if LSTMs were still viable with a dramatic increase in data, GRUs essentially use the same principals as LSTMs but are less computationally expensive. A trade-off between computational expensiveness and representational power may swing in GRUs favour with a dramatic increase in data.

**References**

**Abe M. and Nakayama H.,** “Deep Learning for Forecasting Stock Returns in the Cross-Section”,

eprint arXiv:1801.01777  (2018)

**Balkin and Ord** , “Automatic neural network modeling for univariate time series”, International Journal of Forecasting, 16 509–515, (2000)

**Bengio, Y. et al**., “Learning Long-Term Dependencies with Gradient Descent is Diﬃcult” IEEE Transactions on Neural Networks 5(2):157–16 (1994)

Chollet, Francois., “Deep Learning with Python”, Manning 2018

**Cho, et al.,** “Learning phrase representations using RNN encoder-decoder for statistical machine translation.” arXiv preprint arXiv:1406.1078, (2014)

**Cybenko G.,** “Approximation by Superpositions of a Sigmoidal Function”, Math. Control Signals Systems (1989) 2:303-314 (1989)

**Elton, E. and Gruber, M.,** “Modern Portfolio Theory and Investment Analysis”, John Wiley (1991)

**Fama E. and French K.,** “The Cross-Section of Expected Stock Returns”, The Journal of Finance, Vol. 47, No. 2. (Jun. 1992), pp. 427-465.

**Goodfellow et al,** “Deep Learning”, <https://www.deeplearningbook.org/contents/mlp.html>, (2016)

**Heaton et al,** “Deep Learning for Finance: Deep Portfolio”, Applied Stochastic Models in Business and Industry, Vol 33, issue (2017)

**Hertz et al.,** “Introduction to the theory of neural computation”, volume 1 of Santa Fe Institute studies in the sciences of complexity. Addison Wesley Pub. Co. (1991)

[**Hochreiter S., and Schmidhuber J.,** “LONG SHORT-TERM MEMORY” Neural Computation 9(8):1735-1780, (1997)](http://www.bioinf.jku.at/publications/older/2604.pdf)**Levin, A.,** “Stock selection via nonlinear multi-factor models”, Advances in Neural Information Processing Systems. pp. 966–972 (1996)

**McCulloch, W., and Pitts, W.,** “A logical calculus of the ideas immanent in nervous activity”.  Bulletin of Mathematical Biophysics, 5:115–133. (1943)

**Nielsen M.,** <http://neuralnetworksanddeeplearning.com/index.html> (2018)

**Olson D. and Mossman C.,** “Neural network forecasts of Canadian stock returns using accounting ratios “, International Journal of Forecasting 19, 453–465 (2003)

**Qi, M.,** “Predicting US recessions with leading indicators via neural network models”, International Journal of Forecasting 17(3) (Jul-Sep), 383-401. (2001)

**Rosenblatt F.,** “Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms”, Spartan Books (1962)

**Ross et al,** “The arbitrage theory of capital asset pricing”. Journal of Economic Theory 13(3), (1976)

**Rumelhart et al.,** “Learning representations by back-propagating errors". *Nature*. **323** (6088): 533–536. (1986))

**Sang Il Lee and Seong Joon Yoo.,** “Threshold-based portfolio: the role of the threshold and its applications”, The Journal of Supercomputing (2018)

**Zhang et al.,**“Multi Factor Stock Selection Model Based on LSTM”, International Journal of Economics and Finance; Vol. 10, No. 8; (2018)

1. The Capital Asset Pricing Model (CAPM) describes the relationship between systematic risk and expected return for assets, particularly stocks https://www.investopedia.com/terms/c/capm.asp [↑](#footnote-ref-1)
2. The risk-free rate of return is the theoretical rate of return of an investment with zero risk. The risk-free rate represents the interest an investor would expect from an absolutely risk-free investment over a specified period of time https://www.investopedia.com/terms/r/risk-freerate.asp [↑](#footnote-ref-2)
3. The price-to-sales ratio is a valuation ratio that compares a company’s stock price to its revenues. The price-to-sales ratio is an indicator of the value placed on each dollar of a company’s sales or revenues.  [↑](#footnote-ref-3)
4. A capitalization-weighted stock market index designed to replicate the performance of top 300 stocks traded in the Shanghai and Shenzhen stock exchanges. [↑](#footnote-ref-4)
5. <https://keras.io/losses/> [↑](#footnote-ref-5)
6. <https://livebook.manning.com/#!/book/deep-learning-with-python/chapter-4> [↑](#footnote-ref-6)
7. <https://keras.io/optimizers/> [↑](#footnote-ref-7)
8. <https://livebook.manning.com/#!/book/deep-learning-with-python/chapter-4> [↑](#footnote-ref-8)