Recurrent Neural Networks in Forecasting S&P 500 index.

Samuel Edet
African Institute for Mathematical Sciences

The objective of this research is to predict the movements of the S&P 500 index using variations of the recurrent neural network. The variations considered are the simple recurrent neural network, the long short term memory and the gated recurrent unit. In addition to these networks, we discuss the error correction neural network which takes into account shocks typical of the financial market. In predicting the S&P 500 index, we considered 14 economic variables, 4 levels of hidden neurons of the networks and 5 levels of epoch. From these features, relevant features were selected using experimental design. The selection of an experiment with the right features is chosen based on its accuracy score and its Graphical Processing Unit (GPU) time. The chosen experiments (for each neural network) are used to predict the upward and downward movements of the S&P 500 index. Using the prediction of the S&P 500 index and a proposed strategy, we trade the S&P 500 index for selected periods. The profit generated is compared with the *buy and hold strategy*.

Introduction

The prediction of financial market has long been a subject of research. This is largely due to the profit the market offers. Associated with the possibility of making profit is the possibility of making loss. This makes the financial market a risky business to generate wealth. Some of the factors responsible for the risky nature of the financial market include volatility of interest or currencies responding to government economic policies or change in monetary policies. These result in non-stationarity and non-linearity of the financial time series. By non-linearity, we mean that often time there appears not to be a linear relationship between the factors that influence returns in the market. While by non-stationarity, we mean that financial data may have mean and variance that changes over time. The problem of non-linearity and non-stationarity makes predicting financial instruments a difficult task.

There are different techniques that can be used in forecasting financial instruments (e.g. stock market indices). Broadly speaking, we can classify these techniques into statistical and neural networks. The use of statistical techniques has proven less effective when compared to neural networks, because neural networks are able to capture non-linearities in the market and most importantly, unlike statistical techniques, neural networks do not depend on the underlying assumptions of the independent variables used in the prediction. However, the disadvantage of the neural network is that, there is no formal way of designing the network. The choice of features and the tuning of different parameters in the design of the network is experimental.

Neural networks can be explained using the concept of polynomial curve fitting. Polynomial curve fitting involves fitting a function to a set of data points such that the error function is minimized.

$$y(x) = a_0 + a_1 x + \ldots + a_k x^k \tag{1}$$

Equation 1 can be regarded as a non-linear mapping with input x and output y. The precise form of $y(x)$ is determined by the parameters $a_0, a_1, ..., a_k$. These parameters are similar to the weights in a neural network. Suppose we have N data points in vectors x and y. We denote the data points that make up the input vector x as x^n and that of y as y^n. The polynomial can then be written as a functional mapping in the form $y = y(x^n; \mathbf{a})$, where $\mathbf{a}$ is the vector of the parameters. The curve fitting procedure involves minimizing the square of the error summed over the data points.

$$E = \frac{1}{N}\sum_{n=1}^{N}\left[y(x^n;\mathbf{a}) - y^n\right]^2. \tag{2}$$

Equation 1 represents the case of a uni-variate non-linear function. We can extend this to higher dimensions. By higher dimensions, we mean that the number of input variable, say d is greater than 1. Thus we can consider a higher-order polynomial up to order 3.

$$y = a_0 + \sum_{i_1=1}^{d} a_{i_1} x_{i_1} + \sum_{i_1=1}^{d}\sum_{i_2=1}^{d} a_{i_1 i_2} x_{i_1} x_{i_2} + \sum_{i_1=1}^{d}\sum_{i_2=1}^{d}\sum_{i_3=1}^{d} a_{i_1} a_{i_2} a_{i_3} x_{i_1} x_{i_2} x_{i_3}. \tag{3}$$

An increase in the dimension of the input space will increase the number of parameters. Hence, there will be need for a huge dataset in order to determine these parameters.

We can define a neural network topology in the context of graph theory as a graph (N, C), where N is the set of neurons and C is the set of connections. The relationship between the neurons by means of their connections gives rise to a network. Therefore, we can discuss the topology of the network

based on the framework of the neurons and their connectivity. Most neural networks are layered i.e. they have the input layer, the hidden layer and the output layer. The neural network is referred to as a deep neural network if it has more than one hidden layer. Considering the connectivity of the neurons in the layers, there are different types of interconnection of neurons;

Interlayer connection: The connection of neurons in adjacent layers whose indices differ by one.

Intra-layer connection: The connection of neurons in the same layer. A neuron can also be connected to itself. In this case, the intra-layer connection is said to be a self connection.

Supra-layer connection: The connection of neurons in distinct layers, where the index of the distinct layers differ by at least 2.

The connections discussed above can either be full or partial. A fully connected network is a network that has all neurons connected, while a partially connected neural network has some of its neurons connected. The connections help with the flow of information. The flow of information in the network is referred to as the *forward propagation* (the flow of information forward). This flow can be symmetric or asymmetric. When the flow of information is such that it is strictly fed forward i.e. direct, the neural network is a feed forward neural network. If the flow of information is such that it is fed forward and fed back (we do not mean back propagated, rather we mean information being loop), the network is a recurrent neural network.

Over time, different neural networks have been used in forecasting the global stock markets. (Shen & Jiang, 2016) proposed a prediction algorithm that exploits the temporal correlation among global stock markets. The algorithm alongside different regression algorithms was used to trace the actual increment in the markets. The results from their model indicated a prediction accuracy of 74.4 %, 76% and 77.6% for NASDAQ, S&P 500 and DJIA indexes respectively. A trading strategy was built based on their proposed model. Results of their simulations show that the trading strategy guaranteed higher profit when compared with the *buy and hold back* strategy (the stock is bought on the first day and all the stocks are sold on the last day). (M. Dixon & Bang, 2016) described the application of the deep neural network (DNN) in predicting financial market movement. The DNN was applied to back-testing a simple trading strategy over 43 different commodities and FX future mid-prices at five minute intervals. The problem with the prediction model of (M. Dixon & Bang, 2016) is that there was no formal argument for the selection of the features used in the model. Feature selection is important because if certain features that are highly correlated with the response variable (the stock to be predicted) are highly correlated with one another, it does not make sense to choose all (we select only one of them), since the neural network will extract the same information from them. Beyond the features having a strong correlation with the response variable, (Shen & Jiang, 2016) recommend that the closing time of the market be put into consideration. For example, in predicting S&P 500 index, if the closing price of the Hong Kong stock index (HSI) has a strong correlation with S&P 500 index, it can be chosen as a feature since the market closes before the opening of S&P 500 index. (Niaki & Hoseinzade, 2013) explored the use of design of experiment in feature selection. After checking the correlations of the features and eliminating them appropriately, (Niaki & Hoseinzade, 2013) placed the features into groups, each group having 2-factor levels of 1 and 0. Factor level 1 indicates the group is relevant to the response variable and factor level 0 indicates the group is not relevant to the response variable.

In this research, we want to forecast the upward and downward movements of the closing price of S&P 500 index using variants of fully connected recurrent neural networks. These networks are: the simple recurrent neural network, the long short term memory, the gated recurrent unit and the error correction neural network. In selecting the features for training, we will apply the feature selection technique of (Niaki & Hoseinzade, 2013). After predicting the movements of the index, the proposed model in each of the networks will be used in building a trading strategy for some trading periods. The profit margin from the best neural network will be compared with the *buy and hold* trading strategy within the same trading period.

Recurrent Neural Networks

Recurrent neural networks have been an important focus of research in neural networks. This is because unlike the traditional artificial neural networks, recurrent neural network is most ideal for predicting sequential problems. Since there is a need for a neural network that takes into account feedback, the recurrent neural network appears most appropriate for a sequential prediction task. For example, in neurolinguistic programming, recurrent neural network is used in learning the distributive representation of words and in the prediction of words given sequence of previous words. Many other applications involve dynamical systems with time sequence of events e.g. robotic engineering. The learning technique employed by recurrent neural networks is that it keeps track of the sequence of events. It does this using a complex activation unit in its layers.

Suppose x_t is a sequence of input data and y_t is a sequence of output data at time step t. At each time steps, there are numbers of neurons. Each of these neurons performs a linear matrix operation with the input data x_t. The result of this operation is fed into an activation function (e.g. tanh, sigmoid, softmax etc.) depending on the type of prediction in consideration. At each time step, the output of the hidden

state h_{t-1} from the previous time step $t-1$ and the input x_t in the current time step is fed into the next hidden state h_t of the current time step t. The resulting linear matrix operation of these parameters is then fed into an activation function to produce a predicted output $\hat{y}_t$.

$$h_t = \sigma(W^{hh}h_{t-1} + W^{hx}x_t), \tag{4}$$

$$\hat{y}_t = \text{softmax}\ (W^{yh}h_t). \tag{5}$$

Equation 4 is used to compute the hidden layer output at each time step and (equation 5) is used to compute the output. W^{hh} is the weight matrix connecting the hidden layers, W^{hx} is the weight matrix connecting the input layer to the hidden layer and W^{yh} is the weight matrix connecting the hidden layer to the output layer. Note that as in the case of a feed forward neural network, the weight matrices are shared parameters i.e. the same weights are used at different time steps. Suppose the dimensions of the input layer, output layer and hidden layer are D_x, D_y and D_h respectively. Then, the dimensions of W^{hh}, W^{hx} and W^{yh} are $D_h \times D_h$, $D_h \times D_x$ and $D_y \times D_h$ respectively.

The simple recurrent neural network can be trained using a gradient descent algorithm. However, the network is prone to the vanishing and explosion problem when we have a long term dependency (i.e when the time step is large). One way to address this problem is to use the clipping gradient algorithm technique in the case of an explosion or use the orthogonal initialization technique in the case of a vanishing problem. In addition to these, we can improve the dynamics of the network by using complex activation units rather than simple non-linear activation like the sigmoid function. Using complex activation units in the hidden layer results in recurrent networks like the long short term memory (LSTM) and the gated recurrent unit (GRU). The motivation for using a complex activation unit like the long short term memory (LSTM), is to ensure that the network has a persistent memory i.e. it only keeps information relevant to the learning process. This will ensure that it is able to capture long term dependencies. (Hochcreiter & Schmidhuber, 1997) showed that LSTM can learn to bridge time intervals at an excess of 1000 steps even in the case of noisy, incompressible input sequences, without loss of short time lag capabilities. In comparison with real-time recurrent learning, back propagation through time, recurrent cascade correlation, Elman nets, and neural sequence chunking, LSTM leads to many more successful runs, and learns much faster. LSTM also solves complex, artificial long-time-lag tasks that have never been solved by previous recurrent network algorithms. The architecture of an LSTm has an input gate, forget gate, new memory, final memory and output gate.

Similarly, the gated recurrent unit (GRU) proposed in the work of (J. Chung & Bengio, 2014) shares the same architecture as that of an LSTM. It has an input gate, forget gate, new memory, final memory whose functions are same as the LSTM. However, it has no output gate. (J. Chung & Bengio, 2014) describe GRU as follows:

The hidden state is a linear combination of the previous hidden state and the new memory is given as:

$$h_t = (1 - z_t) \circ \tilde{h}_t + z_t \circ h_{t-1}, \tag{6}$$

where z_t is the update gate and $\circ$ is the element-wise multiplication. The update gate is computed as;

$$z_t = \sigma(W_z^{hx}x_t + W_z^{hh}h_{t-1}). \tag{7}$$

The operation of the update gate determines how much of the previous hidden state h_{t-1} should be forwarded to the hidden state h_t. If $z_t \approx 1$, the previous hidden state is copied out to the next hidden state. If $z_t \approx 0$, then only the new memory $\tilde{h}_t$ is forwarded to the hidden state. The computation of the new memory is similar to that of the LSTM.

$$\tilde{h}_t = \tanh(W^{hh}(r_t \circ h_{t-1}) + W^{hx}x_t), \tag{8}$$

where r_t is the reset gate. This gate determines how important the previous hidden state is to the new memory $\tilde{h}_t$.

$$r_t = \sigma(W_r^{hx}x_t + W_r^{hh}h_{t-1}). \tag{9}$$

Another interesting variation of the recurrent neural network is the error correction neural network (ECNN). (H. Zimmermann & Grothmann, 2002) proposed the ECNN. The motivation behind this network is simply due to significant autonomous part in a system that limits our knowledge of the dynamics of the system. For example the presence of noise or shocks in the financial market limits the extraction of adequate information from the system. Under such conditions, it is necessary to use the information from the external shock to guide the dynamics of the network.Therefore, in addition to the external input x_t being fed into the network, the error in the previous time step is also fed in as an input. The mathematical formulation of the dynamics of the system is given as:

$$h_t = \tanh(W^{hh}h_{t-1} + W^{hx}x_t + W\tanh(W^{yh}h_{t-1} - y_{t-1})) \tag{10}$$

$$\hat{y}_t = W^{yh}(h_t). \tag{11}$$

A new parameter W is introduced into the system. W is a weight matrix whose dimension corresponds to the dimension of the error $\hat{y}_i - y_i$. The system optimizes the weight matrices $W, W^{yh}, W^{hx}, W^{hh}$ such that the errors computed in the previous time steps are as minimal as possible.

$$\frac{1}{T}\sum_{t=1}^{T}(\hat{y}_t - y_t)^2 \rightarrow \ \min(W, W^{yh}, W^{hx}, W^{hh}) \tag{12}$$

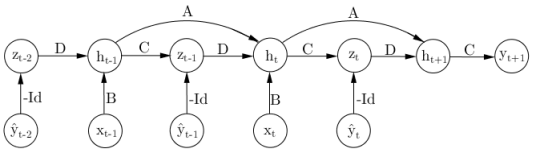

Figure 1. Error Correction Neural Network (ECNN).

Features	Description
SPYt-1,SPYt-2,SPYt-3	Previous daily price of S&P 500 index
HIS, FCHI, FTSE, GDAXI, IXIC, DJI	Stock indices
XOM, PG, GE, MSFT, JNJ	SPY asset with strong weight average
25, 50, 75, 100	Number of neurons in the hidden layer
40, 80, 120, 160, 200	Number of epoch

Figure 2. Features for forecasting S&P 500 index.

Figure 1 represents the architecture of ECNN. We observe the new variable $z_{t-\tau}$, which is the measure of error in the previous hidden state i.e. $z_{t-\tau} = y_{t-\tau} - \hat{y}_{t-\tau}$. The aim of this network is to take into consideration the error as an input and optimize the system such that $z_{t-\tau} \to 0$. The network has two inputs, the first is the external input x_t that directly influences the hidden state h_t and the second is the internal error z_{t-1} that affects state h_t. Another variable that can be seen is $-Id$, which is the negative identity matrix. This weight does not change in the learning process.

According to (H. Zimmermann & Grothmann, 2002), in time series forecasting, we often are faced with the difficulty of trend following behaviour of models. Trend following models are models that underestimate upward trends. This model failure can be identified by a sequel of non-alternating model errors $z_\tau = (y_\tau - \hat{y}_\tau)$. Therefore, to reduce the trend following tendencies, we enforce the alternating errors z_τ. This can be achieved by adding a penalty term to the error function of the error correction neural network.

$$\frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_t)^2 + \lambda(z_t - z_{t-1})^2 \to \min(A, B, C, D) \qquad (13)$$

The additional penalty term $\lambda(z_t - z_{t-1})^2$ is used to minimize the residual auto-covariance in order to avoid trend following behaviour. (H. Zimmermann & Grothmann, 2002) observe that predictions are not sensitive to the choice of λ.

Data Compilation and Implementation

The dataset in this research contains the daily closing price of the S&P 500 index and 14 financial independent variables considered to possibly have an impact on the closing price of the S&P 500 index. These independent variables are called the features or predictors from which we will use the DOE technique to choose the features that are most relevant to the prediction. The dataset has 2110 daily closing price for the predictors and the S&P 500 index from 1st January, 2009 to 1st April, 2017.

From Figure 2, SPYt-1,SPYt-2, SPYt-3 represents the closing price of day $t-1, t-2, t-3$ respectively. For the indices, we have that: HIS is the Hang Seng index, FCHI is the CAC 40 index in France, FTSE is the financial times stock exchange of 100 companies in the UK, GDAXI is the German DAX index, IXIC represents the NASDAQ composite index and DJI is the Dow Jones Industrial average index both in the US. The ticker symbols XOM, PG, GE, MSFT and JNJ in Table 2 represent the stock of Exxon Mobil Corporation, Procter & Gamber Co, General Electric company, Microsoft Corporation and Johnson & Johnson. We consider the returns of SPYt-1,SPYt-2, SPYt-3 and the returns of the previous days of the stock indices and the stock of the companies. In calculating the returns (log returns), we use the formulae:

$$\text{Return} = \ln\left(\frac{x_t}{x_{t-1}}\right). \qquad (14)$$

The other features in Figure 2 are the number of neurons in the hidden layer and the number of epochs (an epoch is simply a full training cycle on a training set).

Implementation

Using the Graphic Processing Unit (GPU) access on the Amazon Web Service (AWS) platform, for efficient computation, the codes are written using the Keras library, which is an open source neural network library written in Python. The procedure for implementation is as follow: import the daily closing price of the financial indicators from Yahoo finance; pre-process the datasets; find the correlation of all the features and selcet only one feature for any two features whose correlation is greater than 0.5. After selecting the features that are relevant for prediction, we find the difference in the daily log returns and where the difference is positive, this is classified as 1 (upward movement) and where the difference is negative, this is classified as 0 (downward movement). This process is performed on the whole dataset before splitting it to train set (80% of the dataset) and test set(20% of the dataset). The train set is used to learn the structure of the dataset. After training, we test the neural network on the test dataset. The quality of prediction is measured based on the following metrics; misclassified sample, accuracy, confusion matrix and classification report. After prediction, we propose a trading strategy and use the propose strategy alongside the prediction made by the neural networks to simulate the profit made for 1 week, 1 month, 3 months, 6 months, 9 months and 1 year trading periods. These profits are compared with what is obtainable using the buy and hold strategy.

Performing an experimental design, our features are referred to as factors. These factors are experimental and quantitative. They are experimental because we can set the levels and they are quantitative because we can assign a specified level of a quantitative factor. We will assign 1 and 0 to be the levels associated with the factors. 1 indicates that the factor is relevant to the prediction and 0 indicates otherwise. We have the company stocks and stock index as factors with 2 levels, the number of neurons has 4 levels (25, 50, 75, 100) and the number of epoch has 5 levels (40, 80, 120, 160, 200). Therefore, we are dealing with $2^{14} \times 4 \times 5$ factorial design. This implies we have 327680 experiments to do. It is impractical for us to do this number of experiments. Therefore in building a practical design, we will compute the pairwise correlation of all the factors. This is because, if the factors are strongly correlated, then the network will extract the same information from them, so it does not make sense to keep all the factors (one of them is sufficient). By strongly correlated, we mean that the absolute value of their correlation is at least 0.5.

From Figure 3 below, we first consider those factors that have a strong correlation with SPY. Those factors include; SPYt-1, GE, MSFT, PG, JNJ, DJI, IXIC. Since GE, MSFT, PG and JNJ belong to the same group and each pair of them are strongly correlated, we will select only JNJ (since JNJ is most correlated with SPY). Also since DJI and IXIC are in the same group and are strongly correlated, we will select the index that is more correlated with SPY. Hence we select IXIC.

Therefore, the number of factors have been reduced to 3 factors (SPYt-1, JNJ and IXIC) each having 2 levels, 4-levels of the number of neurons and 5-levels of the number of epochs. Now, we have a $2^3 \times 4 \times 5$ design i.e.. 160 experiments. However, we cannot have SPYt-1, JNJ and IXIC all have level 0. Therefore the experiments reduces to 140 experiments. We will not be doing the complete factorial design, we will use the fractional factorial design and consider $\frac{1}{7}$ of the complete factorial design i.e. 20 experiments. In the design, SPYt-1, JNJ and IXIC will all have the level 1, while the levels of the hidden neuron and epoch will vary. We will be performing 20 experiments, each for the simple recurrent neural network, long-short term memory and gated recurrent unit.

Figure 3. Correlation of features.

	SPY	SPYt-1	SPYt-2	SPYt-3	FCHI
SPY	1.000000	-0.496888	0.006337	-0.003752	0.005153
SPYt-1	-0.496888	1.000000	-0.496892	0.006336	-0.004540
SPYt-2	0.006337	-0.496892	1.000000	-0.496901	0.010580
SPYt-3	-0.003752	0.006336	-0.496901	1.000000	-0.037684
FCHI	0.005153	-0.004540	0.010580	-0.037684	1.000000
FTSE	-0.108714	0.004658	-0.024392	-0.036033	0.378694
GDAXI	-0.112388	-0.003895	0.006566	-0.011610	0.344643
XOM	-0.415252	0.012621	-0.014121	0.008859	0.113898
GE	-0.483895	0.002849	0.002000	-0.008233	0.094181
MSFT	-0.468465	0.002053	-0.005670	0.000472	0.065550
PG	-0.482705	0.002723	-0.006275	0.006816	0.057984
JNJ	-0.485034	0.004802	-0.001488	-0.000721	0.035067
DJI	-0.492941	0.006909	-0.005361	-0.000986	0.096512
IXIC	-0.494649	0.006205	-0.005072	-0.001746	0.074634

	FTSE	GDAXI	XOM	GE	MSFT
SPY	-0.108714	-0.112388	-0.415252	-0.483895	-0.468465
SPYt-1	0.004658	-0.003895	0.012621	0.002849	0.002053
SPYt-2	-0.024392	0.006566	-0.014121	0.002000	-0.005670
SPYt-3	-0.036033	-0.011610	0.008859	-0.008233	0.000472
FCHI	0.378694	0.344643	0.113898	0.094181	0.065550
FTSE	1.000000	0.186405	0.260738	0.239626	0.225608
GDAXI	0.186405	1.000000	0.158445	0.255542	0.229085
XOM	0.260738	0.158445	1.000000	0.843109	0.728928
GE	0.239626	0.255542	0.843109	1.000000	0.926529
MSFT	0.225608	0.229085	0.728928	0.926529	1.000000
PG	0.235010	0.221170	0.843379	0.943994	0.910331
JNJ	0.221451	0.195103	0.815062	0.945740	0.953022
DJI	0.265053	0.261944	0.862835	0.970866	0.929466
IXIC	0.237530	0.260123	0.818048	0.967740	0.951511

	PG	JNJ	DJI	IXIC
SPY	-0.482705	-0.485034	-0.492941	-0.494649
SPYt-1	0.002723	0.004802	0.006909	0.006205
SPYt-2	-0.006275	-0.001488	-0.005361	-0.005072
SPYt-3	0.006816	-0.000721	-0.000986	-0.001746
FCHI	0.057984	0.035067	0.096512	0.074634
FTSE	0.235010	0.221451	0.265053	0.237530
GDAXI	0.221170	0.195103	0.261944	0.260123
XOM	0.843379	0.815062	0.862835	0.818048
GE	0.943994	0.945740	0.970866	0.967740
MSFT	0.910331	0.953022	0.929466	0.951511
PG	1.000000	0.956926	0.966192	0.951937
JNJ	0.956926	1.000000	0.952219	0.960124
DJI	0.966192	0.952219	1.000000	0.989273
IXIC	0.951937	0.960124	0.989273	1.000000

Results

From Figure 5 , we can observe that for the simple recurrent neural network, experiments 11 and 20 yield the best accuracy score, their accuracy score being 75%. However, from Figure 9 , comparing the GPU time for both experiments, the GPU time for experiment 11 is 58.6347 seconds and that of experiment 20 is 373.436 (approximately six times more than experiment 11). Based on this information, the neural model that will be used for prediction will be experiment 11. Hence, the simple recurrent neural network model will have the following features; SPYt-1, JNJ, IXIC, 75 hidden neurons and 40 epochs. For the long short term memory, experiments 2 and 6 had the best accuracy score of 74%. However, comparing the GPU time for both experiments, we have that the GPU time for experiments 2 and 6 are 880.36 seconds and 455.511 respectively. Based on the GPU time, we will consider experiment 2 as long short term memory neural network model. This model has the following features; SPYt-1, JNJ, IXIC, 25 neurons and 80 epochs. Similarly, for the gated recurrent neural network model, we will choose experiment 6 (accuracy score of 74% and GPU time of 409.454). The features of this model are; SPYt-1, JNJ, IXIC, 50 neurons and 40 epoch. We observe that on average, experiment 6 performs best in the three different neural networks with a mean accuracy score of 74%.

Figure 4. Design of experiment.

Experiments	SPYt-1	JNJ	IXIC	Neurons	Epochs
1	1	1	1	25	40
2	1	1	1	25	80
3	1	1	1	25	120
4	1	1	1	25	160
5	1	1	1	25	200
6	1	1	1	50	40
7	1	1	1	50	80
8	1	1	1	50	120
9	1	1	1	50	160
10	1	1	1	50	200
11	1	1	1	75	40
12	1	1	1	75	80
13	1	1	1	75	120
14	1	1	1	75	160
15	1	1	1	75	200
16	1	1	1	100	40
17	1	1	1	100	80
18	1	1	1	100	120
19	1	1	1	100	160
20	1	1	1	100	200

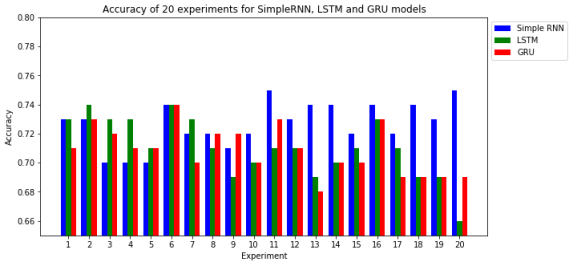

Figure 5. Accuracy of 20 experiments for SimpleRNN, LSTM and GRU.

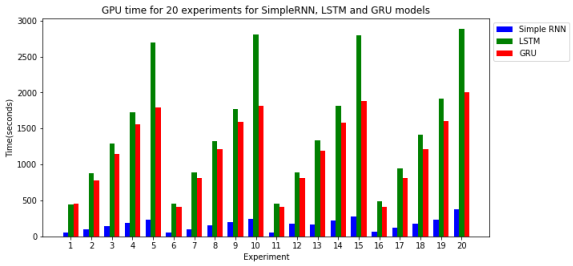

Figure 6. GPU time of 20 experiments for SimpleRNN, LSTM and GRU.

Simple RNN model				
Classification Report				
Binary class	Precision	Recall	F1-score	Support
0	0.80	0.73	0.76	55
1	0.69	0.77	0.73	44
Misclassified Sample = 25				
Accuracy = 75%				

Simple RNN model		
Confusion matrix		
Binary class	0	1
0	40	15
1	10	34

LSTM model				
Classification Report				
Binary class	Precision	Recall	F1-score	Support
0	0.80	0.71	0.75	55
1	0.68	0.77	0.72	44
Misclassified Sample = 26				
Accuracy = 74%				

LSTM model		
Confusion matrix		
Binary class	0	1
0	39	16
1	10	34

GRU model				
Classification Report				
Binary class	Precision	Recall	F1-score	Support
0	0.81	0.69	0.75	55
1	0.67	0.79	0.73	44
Misclassified Sample = 26				
Accuracy = 74%				

GRU model		
Confusion matrix		
Binary class	0	1
0	39	16
1	10	34

Figure 7. Classification Report and Confusion matrix for SimpleRNN, LSTM and GRU.

Table captures the classification report and confusion matrix of the chosen experiments. The classification report gives information about the precision, recall, accuracy and F1-score. The precision metric is the ratio of correctly predicted class to the total predicted class, recall measures the sensitivity of the class, F1-score is the weighted average of precision and recall, and support is simply the total observations we have. Let $i = 0, 1$ be the binary class, t_0, t_1, f_0 and f_1 be correctly classified 0, correctly classified 1, misclassified 0 and misclassified 1 respectively. We denote precision, recall and F1-score of a class as p_i, r_i and f_i respectively. Therefore the indicators in Table 7 can be calculated as follows:

$$p_i = \frac{t_i}{t_i + f_i}, \quad r_i = \frac{t_i}{t_i + f_{\sim i}}, \quad f_i = \frac{2p_i r_i}{p_i + r_i}. \tag{15}$$

$$\text{Accuracy} = \frac{t_i + t_{\sim i}}{t_i + t_{\sim i} + f_i + f_{\sim i}}. \tag{16}$$

(Note: $\sim i$ denotes not i. If $i = 0$, then $\sim i = 1$ vis-a-vis).

In predicting the movement of the S&P 500 index for the next 99 days, the selected model for the simple recurrent neural network model predicted 74 trends correctly. Out of the 55 downward trends, only 15 were mis-classified as upward trends. Also, out of the 44 upward trends, 10 were mis-classified as downward trends. The long-short term memory model and the gated recurrent unit model both forecast 73 trends correctly, they mis-classified 16 downward trends as upward trends and 10 upward trends as downward trends.

Trading model

In the previous section, we have discussed how the neural networks can be evaluated using the number of correctly predicted movements. However, investors are more concerned with the profit they stand to earn. Hence, we will use the prediction models in the previous section to estimate the profit an investor stand to gain investing in S&P 500 index over some trading periods (1 week, 1 month, 3 months, 6 months, 9 months and 1 year). We will compare the returns of our training model with the *buy and hold strategy*.

In the trading strategy, we will assume that:

- The number of days in the training periods 1 week, 1 month, 3 months, 6 months, 9 months and 1 year are

5 days, 20 days, 60 days, 120 days, 180 days and 240 days respectively.

- The investor has an initial investment capital of $\$100,000$.
- Trading is made using the closing price on the same day.
- Transaction cost is not considered in the computation.

The trading strategy to be implemented is as follows:

- When the output is 1 i.e. the model forecasts an increase in S&P 500, if the capital is in the form of cash, all the cash will be used in buying stocks. If the capital is in form of stocks, no trade will be performed.
- When the output is 0 i.e. the model forecasts a decrease in S&P 500, if the capital is in form of cash, no trade will be performed. If the capital is in form of stocks, all the stocks will be liquidated (sell all the stocks).
- At the end of the trading period, if the capital is in form of stocks, we will liquidate the stock using the closing price of the last trading day. Whatever we have in cash is the return after the trading period. The profit is the difference between the return and the initial investment.
- We will compare the profit from our trading strategy with the *buy and hold strategy*. In the *buy and hold strategy*, the investor simply uses the initial investment capital to buy stocks on the first trading day and liquidate all the stocks at the end of the trading period.

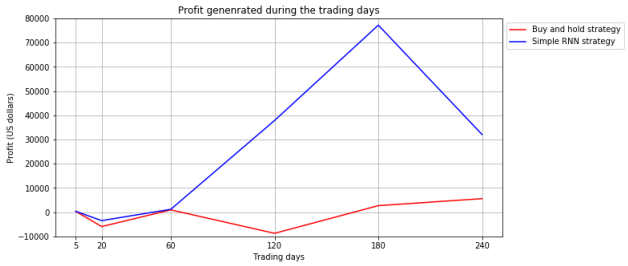

Figure 8. Profit of the trading strategies over some trading periods.

Figure 8 shows the simulation of the profit generated for the trading periods (1 week, 1 month, 3 months, 6 months, 9 months and 1 year), using the *buy and hold strategy* and the proposed strategy based on the Simple RNN forecast. We observe from the graph that the Simple RNN model strategy outperform the buy and hold strategy. The Simple RNN strategy only made a loss during the 1 month (20 days) trading period, while the *buy and hold strategy* made a loss for 1 month (20 days) and 6 months (120 days) trading periods. From Table 9, the loss made using the Simple RNN strategy was $\$3,489.18$ which is smaller than the loss of $\$5,952.63$ made using the *buy and hold strategy*. In addition to this, we observe that the Simple RNN trading strategy loss margin is smaller compared to its profit margin. Also, while the average profit of the Simple RNN strategy over the six trading periods is $\$24,200.61$, that of the *buy and hold strategy* results in a loss of $\$857$. The peak profit of the Simple RNN strategy is $\$77,224.58$, which was generated in the 9 months trading period. This profit is approximately 77.2% of the initial investment capital. Similarly, the peak profit of the *buy and hold strategy* is $\$5,582.98$, which was generated in the 1 year trading period. This profit is approximately 5.6% of the initial investment capital.

Trading periods	SimpleRNN profit ($)	Buy and hold profit ($)
1 week (5 days)	345.63	340.84
1 month (20 days)	-3,489.18	-5,952.63
3 months (60 days)	1,188.78	943.63
6 months (120 days)	37,873.18	-8,742.67
9 months (180 days)	77,224.58	2,685.85
1 year (240 days)	32,060.69	5,582.98

Figure 9. Trading Profit of Simple RNN and Buy and hold strategies.

Conclusion

In this research, we investigated variants of the recurrent neural network. We discussed the simple recurrent neural network whose hidden layers are made up of simple activation functions e.g. the sigmoid function. In addition to this, we considered recurrent neural networks with complex activation functions, which include: the long short term memory and gated recurrent unit networks. Since these networks may not take into account immediate shocks typical of financial market trend. We discussed the error correction neural network (ECNN). In applying these networks (i.e Simple RNN, LSTM, GRU) to forecast the movement of S&P 500 index, we used the concept of experimental design to choose the features that are most appropriate for prediction. In each case of the three neural networks to be used for prediction, we performed 20 experiments to determine which of the experiments give the best accuracy score. For the three neural networks, the common features were the SPYt-1, JNJ, IXIC closing price. Our findings showed that, it was sufficient to use 75 hidden neurons and 40 epochs for the Simple RNN model, 25 neurons and 80 epochs for the LSTM model and 50 neurons and 40 epochs for the GRU model. The three selected experiments for these models were able to predict the movement of S&P 500 index with an accuracy of 75%, 74% and 74% respectively.

In addition to the accuracy of forecasts of these models, we have been able to show that using the proposed strategy with the Simple RNN forecast yields better profit than the buy and hold strategy. Specifically, we observe that the Simple RNN model strategy seldom makes loss and its profit margins far outweigh its loss. In the simulation of the profit of both strategies, we have not taken into account transaction or frictional cost (cost of buying and selling etc). Frictional cost will definitely influence the feasibility of using the Simple RNN strategy in practice. It is difficult to take into account this cost, since the cost varies.

References

Hochcreiter, S., & Schmidhuber, J. (1997). Long-short term memory. *Journal of Neural Computation*, *9*, 1735-1780.

H. Zimmermann, R. N., & Grothmann, R. (2002). Modelling and forecasting financial data. *Springer US*.

J. Chung, K. C., C. Gulcehre, & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR abs/1412.3555*.

M. Dixon, D. K., & Bang, J. (2016). Classification-based financial markets prediction using deep neural networks. *Available from* `https://papers.ssrn.com/sol3/papers.cfm?abstract_id` = 2756331.

Niaki, S., & Hoseinzade, S. (2013). Forecasting s&p 500 index using artificial neural networks and design of experiment. Journal of Industrial Engineering International.

Shen, S., & Jiang, H. (2016). Stock market forecasting using machine learning algorithm. CS229 Machine Learning Autumn.