Stock Price Forecast Utilizing Long Short Term Memory Artificial Neural Networks

Eugenio Felipe Perez Villarreal [1]

Universidad de Monterrey

Martin Lozano

November 21, 2022

Introduction

Time series prediction is a deeply studied task in multiple scientific fields, and it is an important factor for strategic decision making. We are currently at a time where there is a high availability of data and an ever increasing capability of analysis (Duan et al., 2017). Time series modeling allows for the analysis of data patterns like cyclicality, seasonality, stationarity and data irregularities (J. Vrbka, 2021). One of the main elements of economic forecasting includes the selection of model(s) that befit a given problem, in the case of stocks this includes the tracking of a stock price for a period of time, in which the price is recorded at regular intervals. The reliability of the prediction that a given model yields, often depends on the size of the data, larger data sets create more accurate and reliable predictions. There are multiple methods of obtaining predictions, such as: regression models, time series decomposition models and machine learning models among others. Traditional models focus on linear relationships between variables, this may come in the form of a moving average or an autoregressive component in the case of an autoregressive moving average model (ARMA).

Structure and Findings

In this article, the limitations of traditional methodologies for the prediction of time series are discussed and demonstrated. Afterwards, a case is made as to why neural networks, especially recurrent neural networks pose a solution to the limitations of linear methods of time series prediction. Also, a comprehensive review of how neural networks work, is described and applied on two different stocks, where the architecture of the

network is based on the minimization of the cost function and through the testing and comparison of multiple architectures and their produced results. Consecutively, the stocks are analyzed by their respective performance metrics.

Moving Average

According to (Su et al., 2022) moving average indicators are calculated by averaging the time series data with equal or predefined weights. Generally, time series can be predicted by comparing moving averages at different scales (Su et al., 2022). A single moving average (SMA) can be used to trace the trend of determined period.

Where the moving average can be expressed as $SMA_k = \frac{A_1+A_2+A_3...+A_k}{k}$ where k is the number of periods considered; it is regarded as one of the most popular technical indicators in stock analysis, because they can quickly summarize the overall changing patterns of a time series over a past period. It is common practice to compare multiple moving averages, mixing short term and long term SMA, which can be interpreted as rising

or falling signals depending on their convergence or divergence (Su et al., 2022). The following image corresponds to 3 moving averages with periods of 30, 60 and 200 days.

Where the notation would similarly be

$$SMA_k = \frac{A_1 + A_2 + A_3 ... + A_k}{k}$$

$$SMA_s = \frac{A_1 + A_2 + A_3 ... + A_k}{s}$$

$$SMA_f = \frac{A_1 + A_2 + A_3 ... + A_k}{f}$$

Moving averages can be combined with an autoregressive component, which in term would yield an ARMA or ARIMA model if there is differentiation involved.

ARIMA models

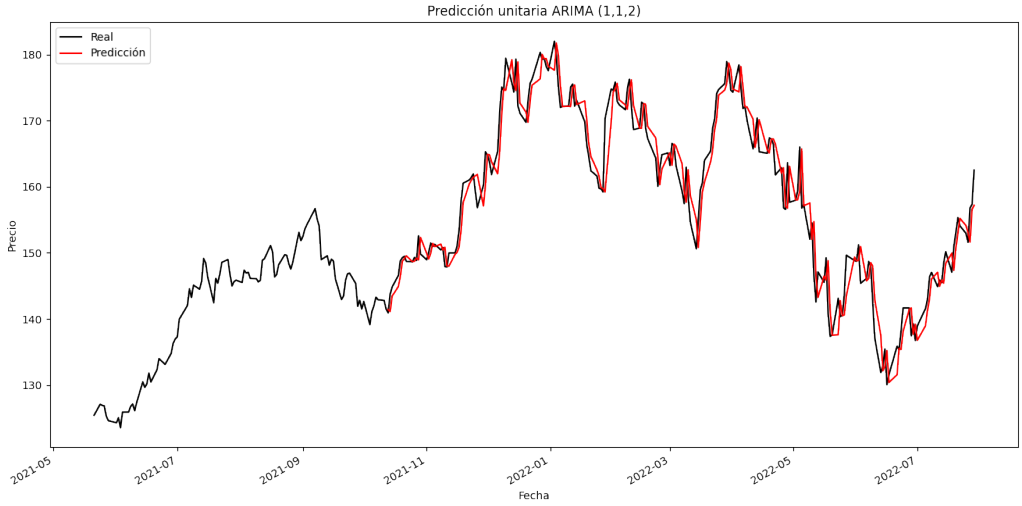
$$X_t = \alpha_1 X_{t-1} + \epsilon_t + \beta_1 \epsilon_{t-1}$$

Where α is the autoregressive component and β is the moving average component. The prediction of time series has used linear methodologies for decades (J. Vrbka, 2021). One important consideration of these methods, is their inability to capture non linear relationships (Wang et al., 2022). In addition, autoregressive integrated moving average (ARIMA) type models, are not able to predict long-term values, since they are dependent on previous values, an alternative solution is to incorporate a regressor or multiple regressors containing current information, this can allow the combination of time series with the regressed variable (J. Vrbka,2021). ARIMA models omit important information relevant to the prediction of time series; especially those with a complex nature, which are especially relevant in the field of economics and finance.

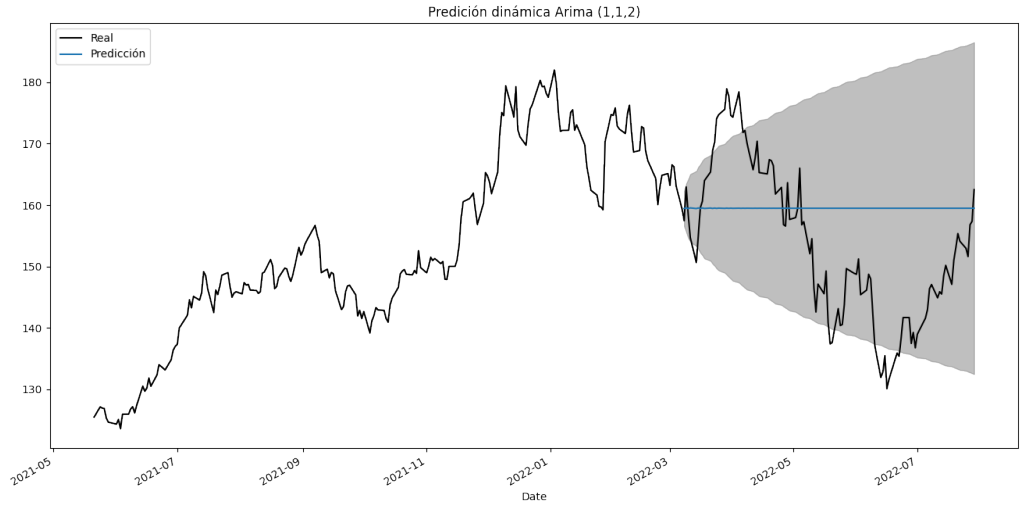
Demonstration

As previously mentioned, one of the main problems is the inability to capture non linear information. Nevertheless, this is only one of the limitations that traditional linear models pose. In the case of ARIMA models, they are not accurate at long term predictions because they require previous values to make a prediction. To illustrate this problem more clearly an ARIMA model was created making step predictions, which only predicts the next value and then reverts back to the original value, making a single step prediction on the next value, so on and so forth, until the end of the time period of the time series.

The following graph is a time series of Apple stock close value, comparing the real value of the stock against an ARIMA model (1,1,2), using one step ahead prediction.



As we can observe this results in an accurate model, where the predicted values are closely related to the real value. Now we present the results of the same model, during the same time period, using a dynamic prediction. Where the shaded area represent the upper and lower confidence intervals in which future values are likely to be inside of.

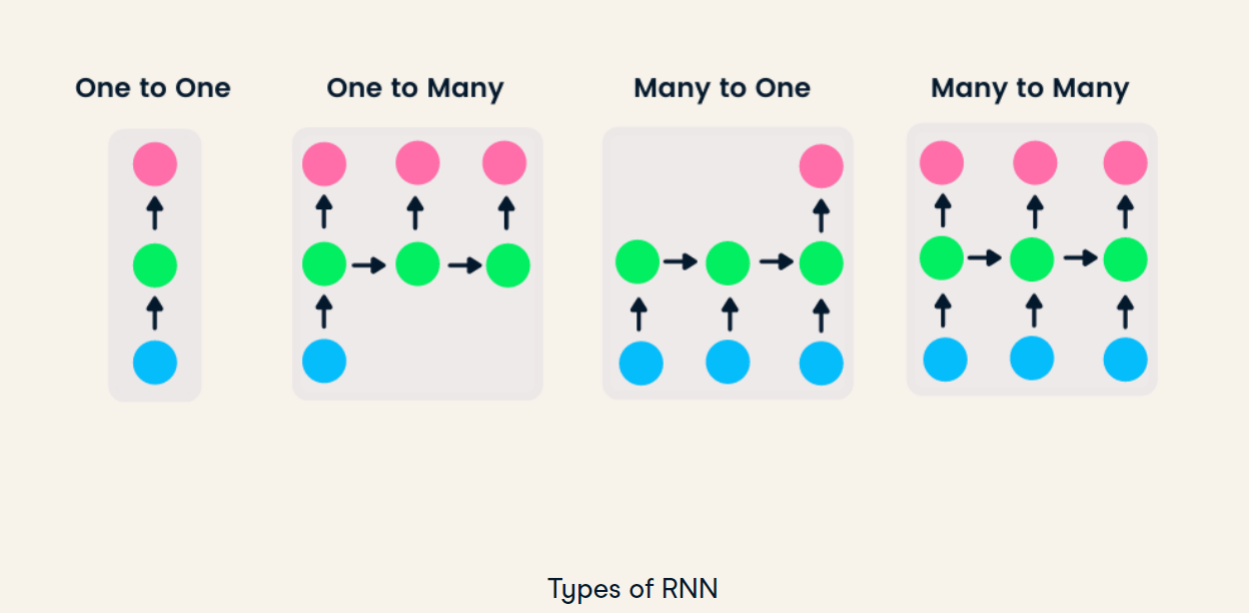


The limitations now become visually evident, where the predicted values have close to null changes across time, making it an unreliable method for predicting future values. Another problem prevalent in traditional methods is that most phenomena that need to be predicted are affected by factors at different levels (J. Vrbka, 2021). As a consequence, additional model elements are added, which take said factors into account, resulting in a broad array of methodologies specifically created to solve a determined problem, this can be observed by the wide array of variations of ARIMA models. In the case of SARIMA a seasonal retrogressive integrated moving average, which has the same elements of the previous model plus a seasonal component. Another instance is the SARIMAX a seasonal autoregressive integrated moving average with exogenous regressors, which on top of the seasonal component S, considers regressors that allow for dynamic predictions. The creation of tailor-made methodologies brings as a consequence; a lack of versatility that characterizes linear methods of prediction. This is where neural networks and machine learning methods come in to place, this is partly because of their nonlinear structure which is appropriate for capturing non linear relationships. It is worth noting, that the ability to capture non linear relationships in the case of neural networks, does not come at the cost of being unable to capture linear relationships, making neural networks a more versatile method of prediction.

Transition Towards Neural Networks

In the face of the limitations imposed by non versatile methods of prediction. New methods, analyzing multiple types of information including time series; neural networks

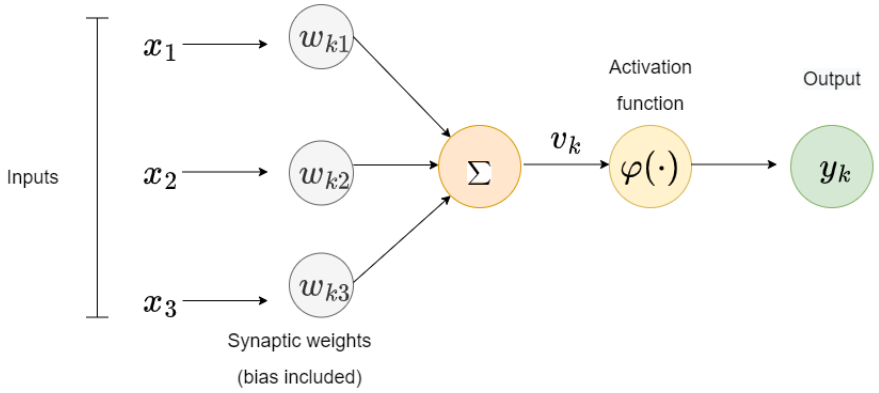
were created. A neural network according to Vochozka (2017), is an algorithm that imitates the manner in which humans learn. Another way of conceptualizing a neural network is as a computer system that takes a given input, processes it and gives an output (Vochozka et al., 2016). There are many types of recurrent neural networks (RNN), the type depends on the relationship an input has to an output. According to (Hyun, S., 2021) an RNN combines variable-length input data with a hidden state that depends on previous time steps to create output data.



Types of RNN

Similarly the human nervous system may be viewed as a three-stage system, which constantly receives information, perceives it, and makes decisions. Typically biological neurons which are structural constituents of the brain, are five to six orders of magnitude slower than those of silicon logic gates, while biological neural events take around a millisecond (10^{-3}) the events that occur in a silicon chip take a nanosecond to occur (10^{-9}), which are present in computers. The brain makes up for the for the slow operation, with an astounding number of neurons, approximately 10 billion with 60 trillion

connections among them (Shepherd and Koch, 1990). In parallel, artificial neural networks are composed of neurons, which receive an input and are multiplied by a weight w_k, added $\sum$ and passed through an activation function $\varphi(\cdot)$ to compute an output.



Where

$$V_k = \sum_{j=1}^{m} w_{kj} x_j + b_k$$

Neural networks may also include a *bias* b_k which increases or lowers the net input of the activation function. Mathematically a neuron can be described as

$$u_k = \sum_{j=1}^{m} w_{kj} x_j$$

And the output

$$y_k = \varphi(u_k + b_k)$$

There are multiple types of activation functions, which account for different outputs or manners in which the summation of weighted values are activated. The sigmoid function is a widely used activation function, also present in LSTM (Long Short Term Memory) RNN, it is a logistic function which can be defined as

$$\varphi(v) = \frac{1}{1 + exp(-av)}$$

One major quality of a sigmoid base activation function is that, it is differentiable, whereas other types of activation function such as threshold functions are not. Also, their output assumes a value from 0 to 1, which is desirable in terms of the standardization of the output it provides.

Feedback

Feedback is said to exist in a dynamic system whenever the output of an element in the system influences part of the input applied to that particular element (Haykin, Simon 1995 p.40). Feedback mechanisms are a vital part of any learning process, and are especially relevant in neural networks known as *recurrent networks*.

All neural networks have a weight parameter, which is an unknown parameter that is found in the hidden layer and its purpose is to make the model fit the training data. In the case of time series, the sum of squared errors can be used as a measure of fit.

$$SS_e = \sum_{k=1}^{K} \sum_{i=1}^{N} (Y_{ij} - f_k(x_i))^2$$

It is often the case that a global minimizer of the error is not the best solution to an optimization problem, as it would likely lead to an overfit solution (Hastie, et al, 2016). Therefore, some other method of regularization must be used, this is achieved by a penalty

term, which helps reduce the error without over fitting. This method usually comes in the form of ***gradient descent***.

Gradient Descent

The process of learning is the constant adjusting of weights and bias in respect to a loss function (Andrychowicz et al., 2016). This is achieved by the training of the network using a subset of the available data; prior to the training of the network a random bias and weights are assigned, and as the training is performed the weights are optimized in respect to the minimization of a cost function, although this adjustment in weights and bias may result only in the obtainment of a local minima, depending architecture of the network. Moreover, said minimization process is done by gradient descent, the "gradient ascent" is the direction of steepest increase given a determined function, in this case a loss function. Naturally, taking the negative gradient results in a change in the synaptic weights, responsible for minimizing a loss function. This is because the gradient of the cost function calculates which changes to which weights matter the most for the minimization process. The complete set of weights can be denoted by θ which consist of.

$$\{\alpha_{0m}, \alpha_m; m = 1, 2, .., M\} \quad M(p+1)$$

$$\{\beta_{0k}, \beta_k; k = 1, 2, .., K\} \quad K(M+1)$$

Backpropagation

Backpropagation can be derived using chain rule for differentiation, this is made by keeping track of local quantities local to each unit.

Using $z_{mi} = \sigma(\alpha_{0m} + \alpha_m^T x_i)$ where $z_i = (z_{1i}, z_{2i}, ..., z_{Mi})$ Where the cost function is

$$z_{mi} = \sigma(\alpha_{0m} + \alpha_m^T x_i)$$

$$z_i = (z_{1i}, z_{2i}, ..., z_{Mi})$$

$$R(\theta) = \sum_{i=i}^{N} R_i = \sum_{k=1}^{K} \sum_{i=1}^{N} (Y_{ij} - f_k(x_i))^2$$

Using derivatives:

$$\frac{\partial R_i}{\partial \beta_{km}} = -2(y_{ik} - f_k(x_i)) g_k'(\beta_k^T z_i) z_{mi},$$

$$\frac{\partial R_i}{\partial \alpha_{ml}} = -\sum_{k=1}^{K} 2(y_{ik} - f_k(x_i)) g_k'(\beta_k^T z_i) \beta_{km} \sigma'(\alpha_m^T x_i) x_{il}$$

Given the expression of the previous derivatives, the gradient descent update at the (r+1)

has the form

$$\beta_{km}^{(r+1)} = \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \beta_{km}^{(r)}}$$

$$\alpha_{ml}^{(r+1)} = \alpha_{ml}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \alpha_{km}^{(r)}}$$

the learning rate is denoted by γ_r

$$\frac{\partial R_i}{\partial \beta_{km}} = \delta_{ki} z_{mi}$$

$$\frac{\partial R_i}{\partial \alpha_{ml}} = s_{mi} x_{il}$$

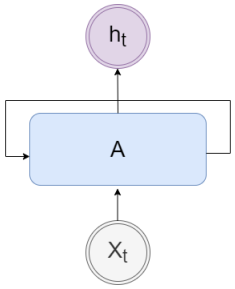
γ_{ki} and s_{mi} are the errors from the current model, these errors satisfy

$$s_{mi} = \sigma'(\alpha_m^T x_i) \sum_{k=1}^{K} \beta_{km} \gamma_{ki}$$

The learning rate r for batch learning is usually taken to be a constant, and can also be optimized by a line search that minimizes the error function at each update. With online learning r should decrease to zero as the iteration r $\rightarrow \infty$.

Recurrent Neural Networks

The process of learning is dependent on previous information, and as information stacks upon more information, *understanding* emerges. Traditional neural networks can't do this, that is have persistence in their learning, as they are incapable of utilizing previous events to inform on later ones. This issue is addressed by creating loops within in it, allowing information to persist and be reused.



Loop in Recurrent Neural Network

Where A looks at a given input x_t and output some value h_t but it loops back

allowing the information to be passed from one previous step, to the next one, this how recurrent neural networks "remember". Furthermore, their usage is mostly suited for modeling complex information, as they can learn to detect patterns, and properly classify or predict based on a given data set. According to (Bas et al., 2016) the geometry and type of neural network determine their composition. Additionally, according to Hastie, et al (2009) neural networks are simply nonlinear statistical models, which is a two stage regression or classification.

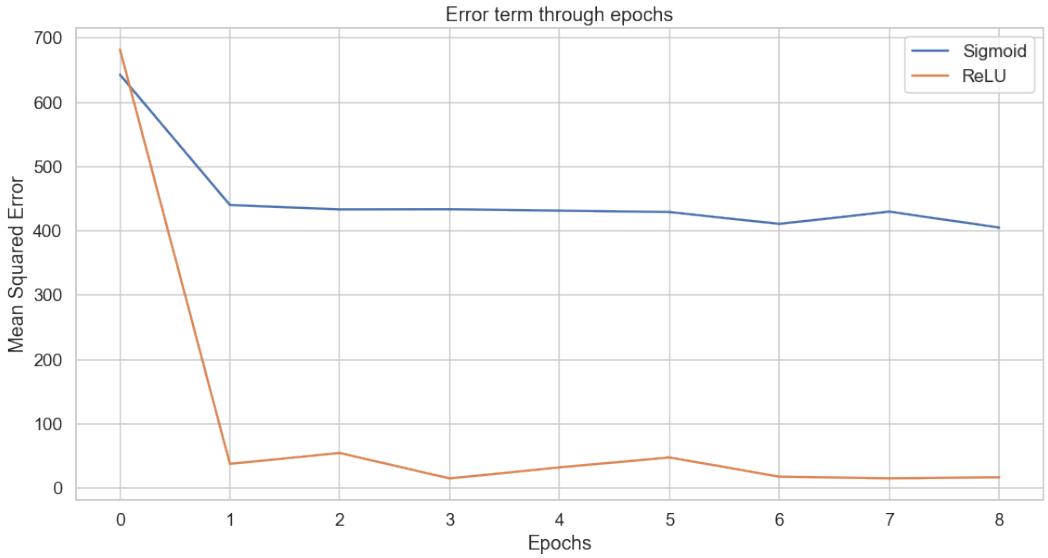
LSTM

Hyun, S (2021) mentions that it has been experimentally proven that LSTM and GRU have higher validation accuracy and prediction accuracy than the standard RNN. His study provides a theoretical and experimental basis for the results that LSTM and GRU have more efficient gradient descent that an standard RNN, which results in a more robust method to the degradation of gradient descent. Moreover, (Sak et al., 2014) mentions that LSTM neural network instantiate a specific architecture able to properly model time sequences that have within them a long dependency rate of their previous values, compared to conventional RNN's. The model was first introduced by Hochreiter and Schmidthuber (1997). This type of networks were specifically created with the purpose of avoiding difficulties with long term dependencies (Sak et al., 2014). Moreover, according to (Satki et al., 2015) the architecture is composed of three main components. The first one, is the input gate, its main function is to filter the selection of inputs and deciding which information is to be retained and which is better to forget; the forget gate can be

represented as: $f_t = \sigma(W_f * \mid h_{t-1}, x_t \mid + b_f)$ where the output given by the sigmoid function is a number between 0 and 1 for each number in the cell, a number 0 would decide to completely forget the information and a number of 1 would mean to keep or "remember" all information.

LSTM architecture

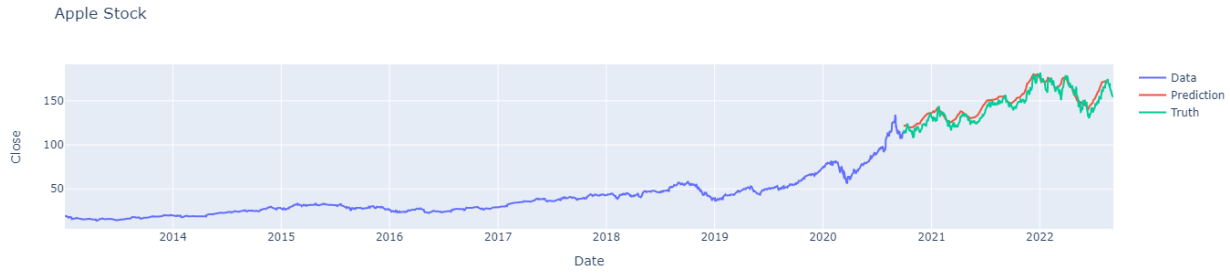
The timeseries used consists of approximately 10 years of information, using the closing price of the stock. Two stocks will be used during the same span of time to have a broader comparison of the efficacy of the network. The train set consists of 80 % of the total observations and the remaining 20 % for the testing of our model. For the construction of the actual model a combination of keras and sklearn libraries were used, the model is compromised by four layers: the first layer takes 15 values as input, the second and third one is compromised of 500 LSTM neurons with a ReLU activation function, finally the fourth layer gives is a single neuron yielding a single output. The justification of the usage of this activation function over the previously mentioned and explained in the investigation, lies in it's capacity to reduce the error term in comparison to one another.



As observed in the previous plot, where the x axis are epochs; one epoch corresponds to the training of the neural network over one cycle, meaning that all the data was used once for the adjustment of synaptic weights and bias term (which isn't present in this model). An epoch is made of one or more batches, which correspond to the number of observations received by the network, in our case the batch size is 15, meaning that each 15 closing values of the stock received by the network an adjustment of the synaptic weights will be produced. Much has been said about choosing an adequate batch size, it is understood that small batch sizes provide regularization for the network, as more frequent adjustments are needed (Smith, 2018). Moreover, at least empirically it makes sense to use a small batch size given that the objective is to generate short term forecasts. Nevertheless, multiple batch sizes were tested and 15 seemed to perform reasonably well and was also a possible number given the GPU memory limitations.

ReLU activation function is better at reducing the MSE (mean squared error), therefore it is used as the default activation function for consecutive networks. It can be noticed that the largest decrease in error function in both activation functions is present in the first epoch, afterwards a slower gradual decrease can be noticed as the network is feed more epochs. Another thing worth noting, is that the usage of more epochs does not correspond to a further minimization of the cost function, this means that if 30 or 40 epochs were used, the cost function would remain virtually the same, but it could result in over fitting, as a consequence only 15 epochs were used.

Once the structure of the network has been decided, a comparison can be made between the historical data and the prediction generated by the model, the following graph presents predictions that are done parallel to the comparison and training of the historical data of the stock. These predictions are not dynamic, meaning that the model is not producing prediction over prediction, but instead relies on the adjustment of weights based on the comparison of historical data.



A close relationship between the predicted values and the true values is observed, this means that the training of the model was successful, or in other words, that the adjustment of the synaptic weights effectively reduced the error term. Which is reflected by

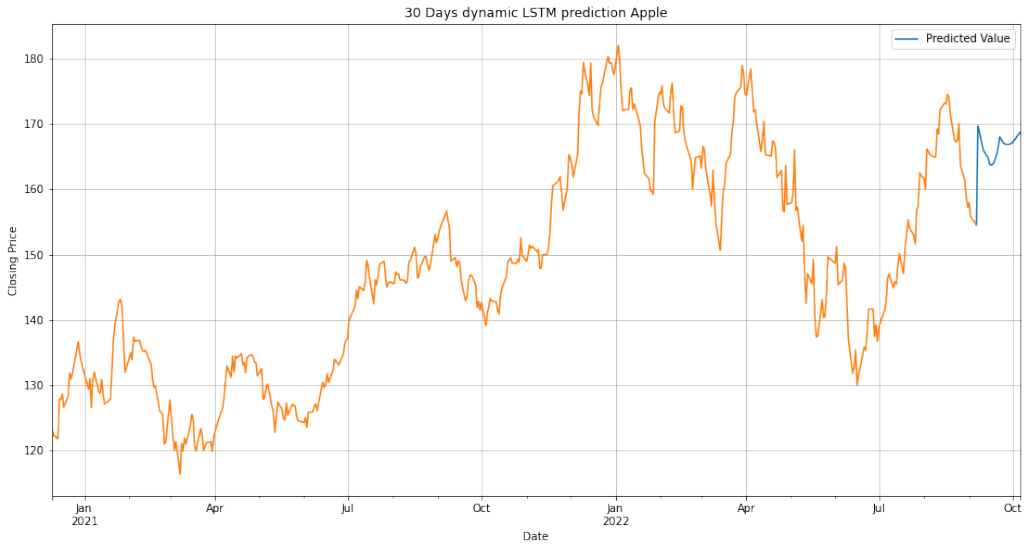
a low distance between the predicted value and the true value of the time series. However, as good as the network is in predicting values that have already taken place in time, how good is it at dynamically forecasting future values that haven't happen yet? after all, this was one of the critics made to traditional methods regarding the prediction of time series, specifically ARIMA models and their incapacity to predict long term.

Dynamic Forecast

In order for dynamic forecast to be possible it's necessary to tweak the code behind the production of predicted values of the model, this change comes in the form of dynamically assigning the input as the last n values produced by the model and iterating over then until a determined range of values is achieved, for this, specific functions are provided that can take the trained model as an extension to produce dynamic forecasts.

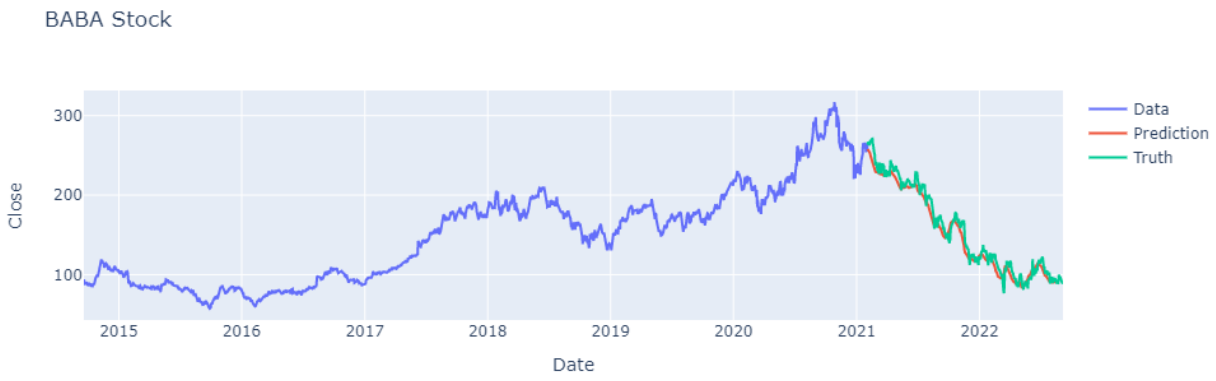
```
close_data = close_data.reshape((-1))

def predict(num_prediction, model):
    prediction_list = close_data[-look_back:]

    for _ in range(num_prediction):
        x = prediction_list[-look_back:]
        x = x.reshape((1, look_back, 1))
        out = model.predict(x)[0][0]
        prediction_list = np.append(prediction_list, out)
    prediction_list = prediction_list[look_back-1:]
    return prediction_list

def predict_dates(num_prediction):
    last_date = dataset_nd['Date'].values[-1]
    prediction_dates = pd.date_range(last_date, periods=num_prediction+1).tolist()
    return prediction_dates

num_prediction = 30
forecast = predict(num_prediction, model)
forecast_dates = predict_dates(num_prediction)
```

Additionally, it is worth pointing out that, while the model is trained and the synaptic weights are locally optimized for the prediction of values; uncertainty grows as the model forecasts dynamically into the future. Therefore, only 30 values are produced, further tests would be needed to measure the reliability of the range of dynamic forecasts. Lastly, with the training of the model now finalized and the modifications on the code now allowing for a dynamic forecast, it's time to see the results.

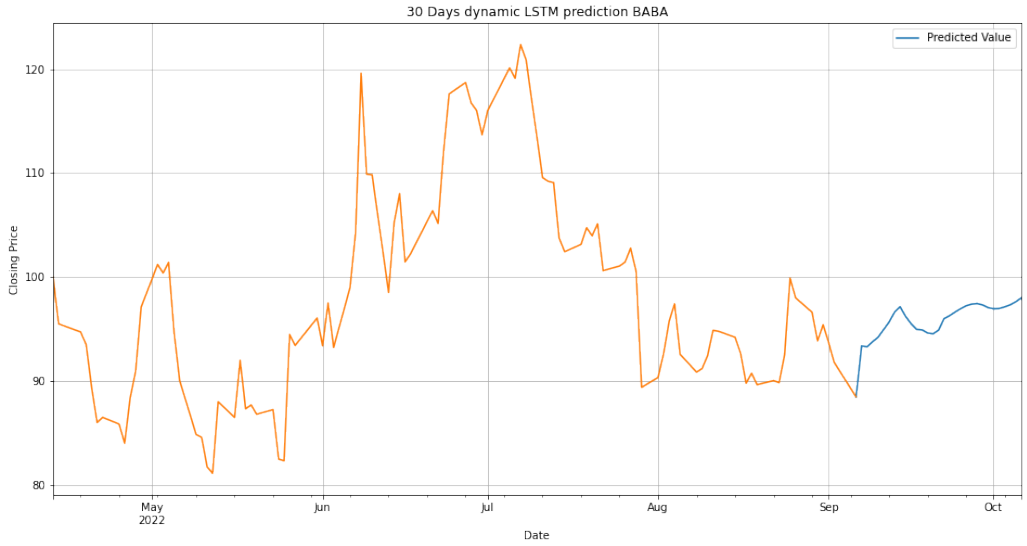


It is observed from the graph that forecast values behave very differently from the straight line that was previously produced by the ARIMA model, in this case the result is not linear. Even though, the testing of the accuracy of the forecast produced by neural networks is beyond the scope of this investigation, a key point that is found, is that neural networks are able to dynamically forecast in a non linear fashion, which regardless of the accuracy is in itself an accomplishment. However, for the sake of completeness another stock (BABA) will be used to try and replicate and compare performance metrics. The same architecture of the network is used with BABA and the following result is obtained in

terms of training and testing of the model.



Much like in the case of APPLE an 80% split was used for the training of the model and the remaining 20% was used as the test set, a close relationship between the predicted value and the true value is observed, meaning that the training of the model was successful in minimizing the cost function.



After the training is done, a dynamic prediction is produced over 30 days which analogous to APPLE looks to be non linear.

Metrics

The metrics chosen for the performance of the metrics, were those concerned with difference between a predicted value and an observed value, these are present in multiple others investigations, such as: (Patel et al., 2015) and (Chong et al., 2017).

	MSE_BABA	MSE_APPL	Mean_Absolute_Error_BABA	Mean_Absolute_Error_APPL	RMSE_BABA	RMSE_APPL
0	4259.02	12708.160160	45.36	26.496243	65.26	112.730476
1	789.29	34.920044	20.04	3.814047	28.09	5.909318
2	298.71	23.183617	13.00	3.225243	17.28	4.814937
3	247.07	33.627213	12.00	3.307042	15.71	5.798898
4	177.61	39.326229	10.00	3.990536	13.32	6.271063
5	230.49	12.632185	11.23	2.462352	15.18	3.554178
6	175.50	15.160715	9.84	2.470460	13.24	3.893676
7	190.04	18.499514	10.03	2.985900	13.78	4.301106
8	147.81	19.598803	8.83	3.062657	12.15	4.427053
9	152.59	13.652797	9.26	2.467645	12.35	3.694969
10	195.02	9.181006	9.37	2.019402	13.96	3.030018
11	506.31	7.519006	16.21	1.703154	22.50	2.742081
12	271.53	8.831658	12.08	2.014722	16.47	2.971811
13	194.70	14.166310	10.18	2.668170	13.95	3.763816
14	117.82	9.553762	8.06	1.999157	10.85	3.090916

From the previous table there are multiple things that can be noticed, the first one and more obvious one is that the training is more efficient at reducing the error term in the case of APPL rather than in the case of BABA, even though they both were trained using the same architecture. Another thing worth noting, is that the mean squared and the root mean squared error of APPLE is higher at the start of training relative to BABA, this makes sense as the root mean squared error is linearly connected with the mean squared error and because there is a significant difference in the magnitude of the price of both stocks, where in the case of Apple is around 180 USD, Alibaba is around 80 USD.

Therefore, larger deviations make sense considering differences in magnitude. Also, we can see that the error term does not linearly decay throughout the passing of training epochs and looks to be increasingly less efficient at reducing the error term as the number of epochs increase, which raises questions for future investigations as to the appropriate method to deciding the optimal number of epochs in an LSTM neural network.

Conclusions

The limitations of traditional methods were made clear by their incapacity to predict dynamically into the future, fact which was exemplified by the autoregressive integrated model (ARIMA) which could appropriately model observed behavior, provided the necessary adjustments were made to the time series (differentiation). However, the ARIMA model was unable to reliably create a forecast with unseen values. Also, the advantages of LSTM ANN were also made clear, which similar to the ARIMA model also had the capacity to model current behavior but in addition was capable of forecasting into the future. Nevertheless more research is needed for the optimization of ANN architecture. Also, more investigation and further testing of stocks is required to determine how accurate the ANN forecast is, and how uncertain it becomes as the number of forecast observations increase.

There is interesting research being made which combines the usage of LSTM RNN and ARIMA models for the forecasting of time series specifically COVID cases around the world (Rabelo and Soares, 2022). In which the logic behind its usage is that because ARIMA models are able to capture linear information, the residuals produced by the model will inevitably contain non linear information, these residuals are feed into a neural

network which are better at explaining non linear behavior to produce a more *accurate* forecast. It's important to understand that while RNN's are a more complex method of explaining non linear behavior they are not necessarily better in all cases, decisions over a determined methodology should be based on goodness of fit and not personal preference over a determined methodology.

References

Araújo Morais, L. R., da Silva Gomes, G. S. (2022). Forecasting daily Covid-19 cases in the world with a hybrid ARIMA and neural network model. Applied Soft Computing Journal, 126. https://doi.udemproxy.elogim.com/10.1016/j.asoc.2022.109315

Choe, D.-E., Kim, H.-C., Kim, M.-H. (2021). Sequence-based modeling of deep learning with LSTM and GRU networks for structural damage detection of floating offshore wind turbine blades. *Renewable Energy*, 174, 218–235. https://doi.udemproxy.elogim.com/10.1016/j.renene.2021.04.025

Chong, E., Han, C., Park, F. C. (2017). Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications*, 83, 187–205. https://doi.org/10.1016/j. eswa.2017.04.030. http://www.sciencedirect.com/science/article/pii/ S0957417417302750.

Jiang, W. (2021). Applications of deep learning in stock market prediction: Recent progress. Expert Systems With Applications, 184. https://doi.udemproxy.elogim.com/10.1016/j.eswa.2021.115537

Patel, J., Shah, S., Thakkar, P., Kotecha, K. (2015). Predicting stock market index using fusion of machine learning techniques. Expert Systems with Applications, 42, 2162–2172. https://doi.org/10.1016/j.eswa.2014.10.031. http://www.

sciencedirect.com/science/article/pii/S0957417414006551.

Rowland, Z., and J. Vrbka. 2016. Optimization of a company's property structure aiming at maximization of its profit using neural networks with the example of a set of construction companies. Mathematical Modeling in Economics 3–4 (7): 36–41.

Smith, L. N. (2018). A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. arXiv preprint arXiv:1803.09820.

Vrbka, J (2017). Using Artificial Neural Networks for Timeseries Smoothing and Forecasting. *Springer* https://doi.org/10.1007/978-3-030-75649-9

Wang, X.Y., and M. Han. 2015. Improved extreme learning machine for multivariate time series online sequential prediction. *Engineering Applications of Artificial Intelligence* 40: 28–36.

3Blue1Brown. (2017, October 5). What is a neural network? [Video]. YouTube. https://www.youtube.com/watch?v=aircAruvnKk

3Blue1Brown. (2017, October 16). Gradient descent, how neural networks learn [Video]. YouTube. https://www.youtube.com/watch?v=aircAruvnKk

3Blue1Brown. (2017, November 3). What is backpropagation really doing? [Video]. YouTube. https://www.youtube.com/watch?v=aircAruvnKk

3Blue1Brown. (2017, November 3). Backpropagation calculus [Video]. YouTube. https://www.youtube.com/watch?v=aircAruvnKk