Recurrent Neural Networks in Stock Price Prediction

Charles Liebenberg

Sunday 16th August 2020

Table of Contents

Literature Review **3**

Model Choice: **3**

Recurrent neural networks approach to the financial forecast of Google assets

Forecasting stock prices with long-short term memory neural network based on attention mechanism

NSE Stock Market Prediction Using Deep-Learning Models

Stock price prediction using LSTM, RNN and CNN-sliding window

Summary 5

Data Preparation: **6**

Integration of Principal Component Analysis and Recurrent Neural Network to Forecast the Stock Price of Casablanca Stock Exchange

A Recurrent Neural Network and a Discrete Wavelet Transform to Predict the Saudi Stock Price Trends

Summary 7

Architecture: **8**

Stock Market Price Prediction Using LSTM RNN

Artificial Neural Networks architectures for stock price prediction

Ensembles of Recurrent Neural Networks for Robust Time Series Forecasting

CNN-LSTM Neural Network Model for Quantitative Strategy Analysis in Stock Markets

Designing Stock Market Trading Systems With And Without Soft Computing

Summary 11

Report **12**

Executive Summary 12

Data Collection, Preprocessing and Preparation 12

Neural Network Building (Topology, Evaluation Criteria and Training) 13

Evaluation and Prediction 16

Reference List **18**

Literature Review

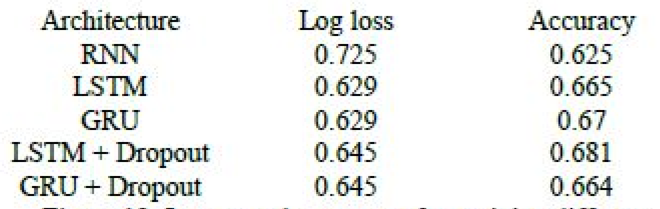
Model Choice:

Recurrent neural networks approach to the financial forecast of Google assets

This paper investigated different types of recurrent neural networks and their ability to forecast price of google assets. It was found that Long-Short Term Memory models performed better than basic recurrent neural networks and Gated Recurrent networks.

The analysis utilised the same basic structure for each recurrent neural network. An input layer was input into two stacked recurrent hidden layers and finally a dense layer with a softmax activation. The models were trained for 100 epochs, and the early stopping callback was found to be useful to prevent overfitting.

The results of the analysis are as follows:



Architecture	Log loss	Accuracy
RNN	0.725	0.625
LSTM	0.629	0.665
GRU	0.629	0.67
LSTM + Dropout	0.645	0.681
GRU + Dropout	0.645	0.664

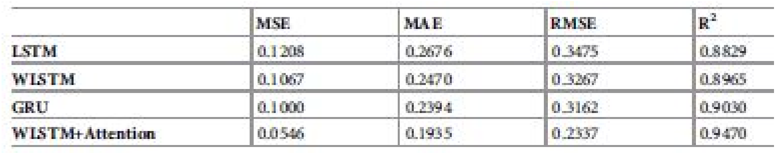
Figure 12: Losses and accuracy after training different architectures

As can be seen, the LSTM and GRU architectures outperform a basic RNN and in turn a regularised LSTM outperforms a regularised GRU.

Forecasting stock prices with long-short term memory neural network based on attention mechanism

This paper found that a Long-Short Term Memory model with attention performed better than conventional LSTM and Gated Recurrent networks.

The analysis utilized open, close, high, low, adjusted close and volume data from Yahoo Finance. It was subsequently normalised for a zero mean and transformed using wavelets. The model was built using a 6 node input layer and connected to an attention-LSTM layer, connected to a dense layer and finally a softmax output layer. 10 nodes were used for the recurrent layer and it was trained for 600 epochs.



	MSE	MAE	RMSE	R^2
LSTM	0.1208	0.2676	0.3475	0.8829
WLSTM	0.1067	0.2470	0.3267	0.8965
GRU	0.1000	0.2394	0.3162	0.9030
WLSTM+Attention	0.0546	0.1935	0.2337	0.9470

The results showed that a wavelet transformed long short term memory model with attention outperformed all other recurrent neural networks as can be seen in the table.

NSE Stock Market Prediction Using Deep-Learning Models

Found that long short term memory models produced meaningfully better results in stock price prediction over convolutional neural networks, multi layer perceptrons and auto regressive moving average models.

20 years of closing price data was gathered and normalised to between 0 and 1. The window size was set to a number of values and trialled for its effect on mean absolute percentage error. It was found that the past 200 days was most effective in predicting the next day's stock price. The architecture used for the LSTM model was 2 recurrent layers followed by a dense layer. The model was trained with an early stopping callback to prevent overfitting

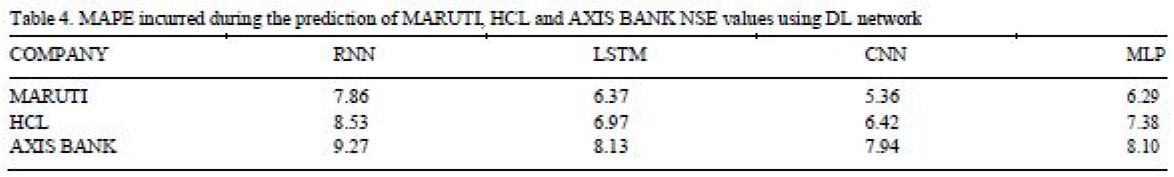


Table 4. MAPE incurred during the prediction of MARUTI, HCL and AXIS BANK NSE values using DL network

COMPANY	RNN	LSTM	CNN	MLP
MARUTI	7.86	6.37	5.36	6.29
HCL	8.53	6.97	6.42	7.38
AXIS BANK	9.27	8.13	7.94	8.10

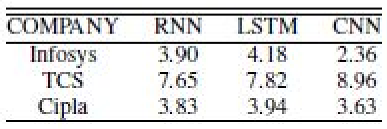
As shown in the graph, the long short term memory outperformed other deep learning architectures.

Stock price prediction using LSTM, RNN and CNN-sliding window

Found that a sliding window convolutional neural network was useful in prediction of stock price in comparison to basic recurrent networks and long short term memory models.

The data was normalised to between 0 and 1. Next, window size was optimised using a set architecture. The same amount of parameters was used in each recurrent hidden layer, and node count was optimised. A weight saving callback was used and the model was optimised using Adam.

TABLE I: ERROR PERCENTAGE



COMPANY	RNN	LSTM	CNN
Infosys	3.90	4.18	2.36
TCS	7.65	7.82	8.96
Cipla	3.83	3.94	3.63

As can be seen in the table, the convolutional neural network proved to have better predictive capabilities for two of the companies. It is hypothesised that the reasoning behind this is that the companies it can predict well are less cyclical in nature and thus when a convolutional network focuses on the current window it is focused on the current direction of stock price. When a company is more cyclical, the recurrent approach of tackling into account previous windows becomes more effective.

Summary

It can be summarized that recurrent neural networks are most effective for time series stock price prediction. Specifically, regularised long short term memory models should result in the lowest prediction error. In certain cases convolutional neural networks may outperform long short term memory models and should be investigated in the name of completeness.

When building time series models, window size should be optimised. It is likely that the optimal window size differs for different datasets owing to the differences found in these reports. Furthermore, weights saving callbacks should be implemented due to the variation in generalisation through each epoch.

Data Preparation:

Integration of Principal Component Analysis and Recurrent Neural Network to Forecast the Stock Price of Casablanca Stock Exchange

Principal components analysis was found to meaningfully decrease external root mean squared error in recurrent neural network stock price prediction.

Principal components analysis was used to decrease dimensionality, using volume, open, high, low closing, simple moving average and exponential moving average as features.

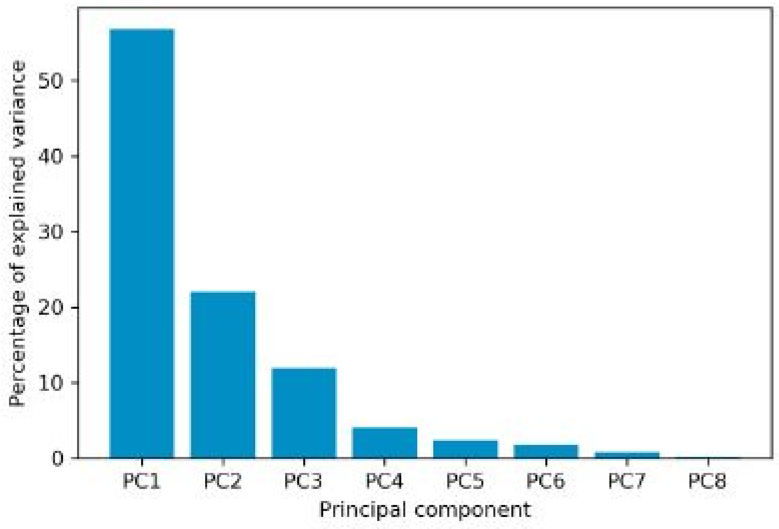


Fig. 2. The percentage of the variance ratio of the 8 features.

It was shown that almost all of the variance could be captured through the use of 6 components, and this was then subsequently implemented. A long short term memory model was used with architecture of an input layer fed to an 18 node LSTM layer and finally a 1 node output. Stochastic gradient descent was used as an optimiser.

Table 5. The mean square error with PCA and without PCA

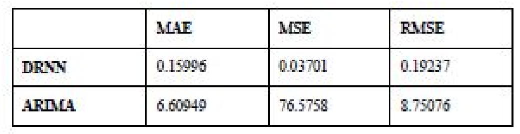
MSE of the Train data(with PCA)	MSE of the Test data (with PCA)	MSE of the Train data(without PCA)	MSE of the Test data (without PCA)
0.004567	**0.00596**	0.001499	**0.011835**

As shown in the table, both train and test metrics benefitted from the implementation of principal components analysis. It was concluded that principal components analysis allowed the neural network to more efficiently use the information in stock price prediction. It was also shown that training time was significantly reduced through the use of principal components analysis.

A Recurrent Neural Network and a Discrete Wavelet Transform to Predict the Saudi Stock Price Trends

This paper found that discrete-wavelet transformed recurrent neural networks outperformed all other models tested.

Open, close, low, high and volume was normalised and then a HAAR discrete wavelet transformation was used to reduce noise. Next the models hyperparameters were optimised, and it was found that three recurrent layers were optimal with batch size= 4, neurons = 4 and epochs =100. The Adam optimiser was used and the following results were found:



	MAE	MSE	RMSE
DRNN	0.15996	0.03701	0.19237
ARIMA	6.60949	76.5758	8.75076

The discrete wavelet transformed recurrent neural network model outperformed the ARIMA model in all metrics. It was concluded that the integration of discrete wavelet transformations and recurrent neural networks could improve performance in predicting stock price.

Summary

It is observed that there is academic precedence for using principal components analysis and discrete wavelet transformations to assist the creation of recurrent neural networks in stock price prediction. Principal components analysis allows for similar variables to be summarised into components and this increases the predictive ability of neural networks. Moreover, it has been shown that the dimensionality of stock price data can be significantly reduced with principal components and that this reduces training time. Discrete wavelet transformations have been shown to allow for noise reduction in time series data. This allows for the neural network to model more signal and less noise.

Architecture:

Stock Market Price Prediction Using LSTM RNN

This paper found that two layer long short term models with relatively high neurons improved stock price prediction accuracy.

Historical data was gathered from Yahoo Finance and was subsequently normalised to between 0 and 1. Adam optimiser was used and the model was trained for 10 epochs. Various architectures were trialled including variations in number of LSTM layers and the associated parameter count.

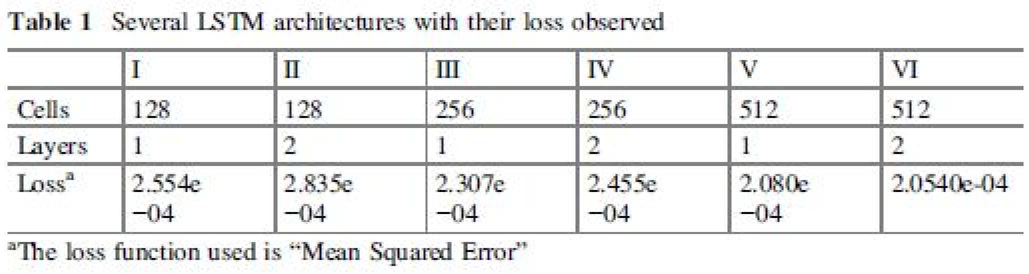


Table 1 Several LSTM architectures with their loss observed

	I	II	III	IV	V	VI
Cells	128	128	256	256	512	512
Layers	1	2	1	2	1	2
Loss[a]	2.554e −04	2.835e −04	2.307e −04	2.455e −04	2.080e −04	2.0540e-04

[a]The loss function used is "Mean Squared Error"

As shown in the table, the results suggested that an architecture with 2 layers and 512 cells was optimal. Moreover, it is observed that prediction accuracy is not heavily improved through hyperparameter optimisation.

Artificial Neural Networks architectures for stock price prediction

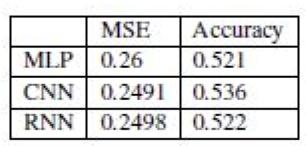
This paper found optimal results for stock price prediction can be found with ensemble learning combining convolutional neural networks and recurrent neural networks.

Data was obtained for the S&P 500 Index. Stock price return was used and normalised for zero mean and unit variance. An 80-20 train-test split was used with 10% of the training set reserved for hyper parameter optimisation (validation). Principal Components Analysis was used to reduce dimensionality from the 16 variables available. A discrete wavelet transformation was used to reduce noise in the dataset.

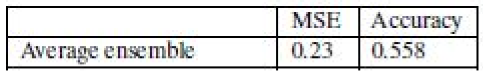
A convolutional neural network was built first with two convolutional layers with number of filters = 64 and pool length = 2. The top of the model contained two dense layers with relu activations and 250 nodes. Two convolutional networks were trained, one normal and one with wavelet decomposed data.

Next a recurrent neural network was trained with two Long Short-Term Memory layers and 1 dense layer. The recurrent layers were found to be optimal with 100 hidden neurons.

These models were then trained separately and ensemble into one model. The results were as follows:



	MSE	Accuracy
MLP	0.26	0.521
CNN	0.2491	0.536
RNN	0.2498	0.522



	MSE	Accuracy
Average ensemble	0.23	0.558

As can be seen, the ensemble approach resulted in a higher peak accuracy than any individual model. It can be concluded that ensemble learning allows for the combination of inferior models to one model with greater prediction capabilities.

Ensembles of Recurrent Neural Networks for Robust Time Series Forecasting

This article found that combining multiple networks of average performance outperformed smaller ensembles of better networks in time series recurrent neural networks.

Data was gathered and subsequently standardised. Used set architecture and then tested a series of sequence lengths and nodes. It was decided on three LSTM layers separated with a dropout of 0.3. The model was trained with RMSProp and a learning rate equal to 0.001. Batch size was 32.

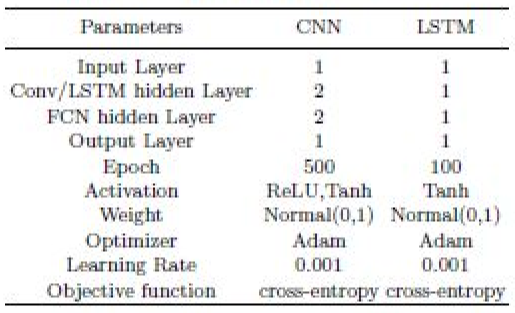
It was found that, when combining recurrent neural networks for ensemble learning, model diversity was essential. Comparably weak predictors with low correlation could be combined for much greater predictive capabilities. It was found that ensembles could be used to forecast results significantly better than baseline estimates.

CNN-LSTM Neural Network Model for Quantitative Strategy Analysis in Stock Markets

This paper found that convolutional neural networks and long short term memory combination neural networks are feasible, robust and highly profitable.

Data was gathered and subsequently normalised with a z-score standardisation. All models were regularised with l2 weight regularization and dropout layers. The model was compiled with the Adam optimiser and cross-entropy was minimised.

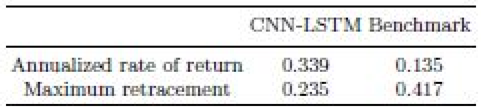
Table 1. The parameters for CNN-LSTM.



Parameters	CNN	LSTM
Input Layer	1	1
Conv/LSTM hidden Layer	2	1
FCN hidden Layer	2	1
Output Layer	1	1
Epoch	500	100
Activation	ReLU,Tanh	Tanh
Weight	Normal(0,1)	Normal(0,1)
Optimizer	Adam	Adam
Learning Rate	0.001	0.001
Objective function	cross-entropy	cross-entropy

The results dictated that relatively small CNN and LSTMs could be combined with the detailed architecture through ensemble learning for good results. When fit to external data, the model resulted in an annualised rate of return of 33.9%. When compared to the baseline result of 13.5%, this is a substantial improvement.

Table 2. The comparison of the results



	CNN-LSTM	Benchmark
Annualized rate of return	0.339	0.135
Maximum retracement	0.235	0.417

Designing Stock Market Trading Systems With And Without Soft Computing

This book references multiple rules of thumbs for choosing artificial neural network architecture from relevant literature. It is suggested that literature suggests the following different approaches:

- Pyramidical topology with number of layers and neurons inferred from this topology.
- Hidden neurons equivalent to 2N+1 with N equal to the number of inputs
- The number of hidden nodes being set to equal the total number of inputs and outputs
- A brute force approach with no intuitive choices

As can be seen, this paper identifies no one industry standard, but it can be summarised that artificial neural networks, as a general rule of thumb, require less complexity than their convolutional and dense counterparts.

Summary

It can be seen that there is substantial difference in model complexity and hidden neurons in the relevant academic literature. In general, the number of layers is quite small, with no paper reviewed implementing a model with more than three layers. There seems to be consensus that ensemble learning allows for improvement in stock price prediction accuracy. Furthermore, it is observed that the Adam optimiser is the standard for stock price prediction and that weight saving callbacks are essential for preserving prediction capabilities.

Report

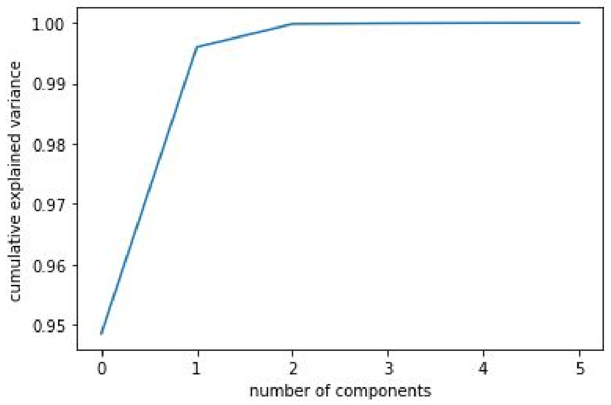
Executive Summary

Stock price prediction is a time-series regression problem extremely relevant to neural networks. This is because of the relative lack of a need for interpretability of the model and the high emphasis on accuracy of predictions. It was found that a Convolutional Neural Network and Long Short-term Memory model could be combined to predict stock price with an external mean squared accuracy of 0.00089 and trainable parameters of roughly 46000.

Data Collection, Preprocessing and Preparation

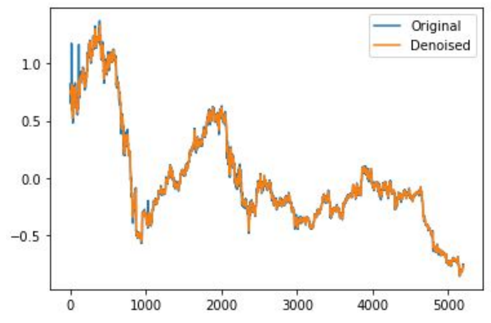
The data was downloaded from finance. Once this data was downloaded, it was normalised. This was done through a min max scaler in which all data was scaled to between 0 and 1. This normalisation was performed to ensure the gradient descent was not dominated by certain variables e.g. volume. Furthermore, a scale between 0 and 1 is intuitie for stock price data as stock price can never be negative. This choice was further corroborated with the literature review.

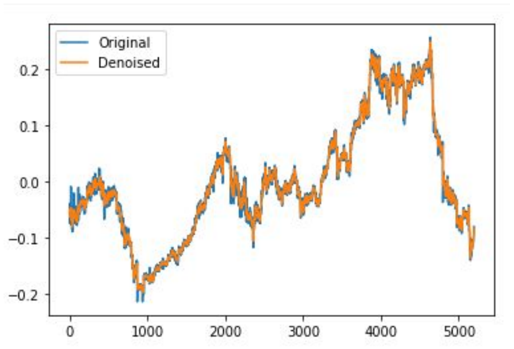
After normalisation, principal components analysis was performed. This allowed for a reduction in dimensionality. It was found that close to 100% of the variation in the data could be explained by two components.



This is intuitively explained by the two categories of information available, price data and volume data. These components summarized these two types of information and effectively reduced the dimensionality and therefore complexity of the analysis.

After principal components analysis was finished, discrete wavelet noise reduction was performed. This is where the data is transformed into a wavelet representation, the data is trimmed according to a variance threshold, and finally the data is reconverted into its original form. The results of this transformation are as follows:

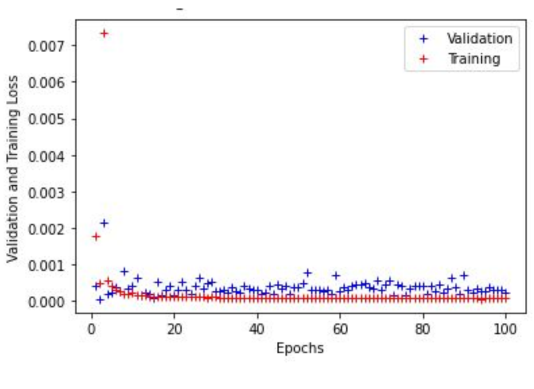




Finally, the data was split into training, testing and validation sets. This was done through the use of keras's TimeSeriesGenerator API. The last 100 days were used for testing, and the training data was split 80-20 into training and validation data. Separate generators were created for both regular data and wavelet transformed data, as it was not yet certain whether these transformations merely reduced noise or actually removed signal from the dataset.

Neural Network Building (Topology, Evaluation Criteria and Training)

First an initial basic long short term model was built for a starting point. Initially the model was structured with two lstm layers and 1 dense layer. This architecture choice was most common in the relevant literature and serves as a good starting point with the option for a decrease or increase in model complexity if the metrics suggest as such.



The model was optimised using the adam optimisation function due to its momentum attributes and popularity in the relevant literature. The loss function was mean squared error as this is the metric that would be evaluated on external data at the end of the analysis for a measure of success. To allow for ease of analysis, the function created in previous analysis was adjusted such that it now plots validation loss against training loss and identifies minimums and at which epoch this occurs.

The model was then trained purposefully to overfit, and did so after about 5 epochs (see diagram). It was observed that there were no late minima and as such the use of an early stopping call back was implemented for all subsequent models. This callback would stop the training of the model if there is no new minimum for validation loss after 10 epochs, and restore the best weights as a result. This would ensure that there was no overtraining/fitting.

The next step in improving the model was addressing its ability to generalise and the rate at which it was overfitting. The first hyperparameter investigated was parameter count. In doing so, the first number of nodes in each lstm layer trialled was the number of days trained on (essentially number of inputs). It was observed that this massively decreased minimum validation loss.

However, after this change the loss graphs suggested that the model may have become too small and begun to underfit. When the model summary was called, it could be seen that the model went from 29000 parameters in the final LSTM layer to 61 parameters in the dense layer. It was hypothesised that this was cause for the model struggling to learn the intricacies of the dataset and thus an increase in dense nodes was trialled.

This did not result in an increase in the models ability to learn the signal, and instead actually increased minimum validation loss. It was then thought that maybe an increase in the number of parameters in the LSTM hidden layers would reduce the underfitting. A double in the number of nodes was trialled. It was found that similarly to an increase in dense nodes, an increase in LSTM hidden layers resulted in a larger minimum validation loss. The final increase in complexity trialled was the addition of another LSTM layer. This resulted in no change in minimum validation loss and generalisation. Therefore, in the name of maintaining the simplest model for any given accuracy, the original architecture was maintained.

The next thing that was investigated was generalisation techniques. The higher the accuracy before external and internal measures start deviating indicates a better generalisation. From the previous analysis, it was found that dropout and weight regularisation helped greatly. Both l1 and l2 regularisation was trialled. Neither of these regularisation techniques were found to be useful in increasing validation accuracy before overfitting. Furthermore, a larger minimum validation loss was reached when they were implemented.

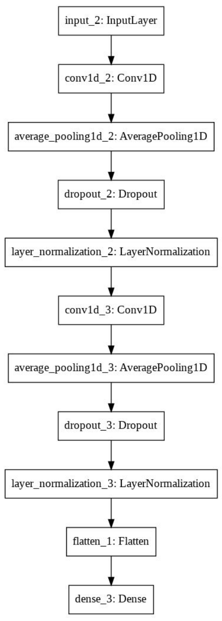
The next regularisation technique trialled was dropout. Upon research, it was found that the keras implementation of long short term memory models have an inbuilt dropout function. This was trialled, and it was found that it did not increase the models ability to generalise. Furthermore, the regular dropout hidden layer was trialled and this also did not seem to help with generalisation.

The next step in improving the models ability to generalise was to test normalisation layers. The use of batch normalisation with LSTM layers is incorrect. This is because weights are shared in a recurrent neural network, and thus the activation response for each recurrent look could have a different scale. Therefore the normalisation technique named layer normalisation, which normalises after each time step, can be used instead. It was found that this technique also proved not useful.

Finally, the use of an attention layer was investigated. A resource was found defining a layer that implemented attention into keras's API. A long short term memory model has the constraint that all inputs are forced to be a set length. An attention layer allows the model to selectively pay attention to the inputs instead. This was thought to allow the model to focus on certain periods of importance. However, it was found that the addition of this attention layer did not actually improve performance. This could perhaps be due to the relative importance of all data, and therefore no part of it should be selectively focused.

It was therefore concluded that the basic structure developed based on informed choices from the literature review was ideal.

The literature review suggested that a convolutional neural network is useful in stock price prediction. Therefore, this was investigated with keras's 1D convolutional neural network implementation

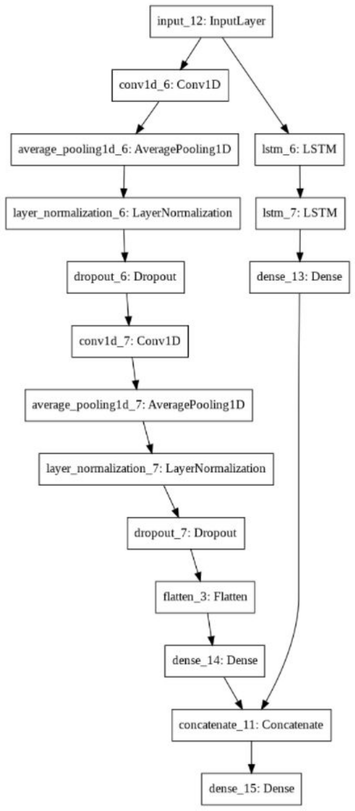

input_2: InputLayer
conv1d_2: Conv1D
average_pooling1d_2: AveragePooling1D
dropout_2: Dropout
layer_normalization_2: LayerNormalization
conv1d_3: Conv1D
average_pooling1d_3: AveragePooling1D
dropout_3: Dropout
layer_normalization_3: LayerNormalization
flatten_1: Flatten
dense_3: Dense

The same process was undertaken as all previous models. A basic model was built and purposefully overfit. It was then trimmed and reduced in complexity following the rule of thumb of the lowest complexity model for a given accuracy.

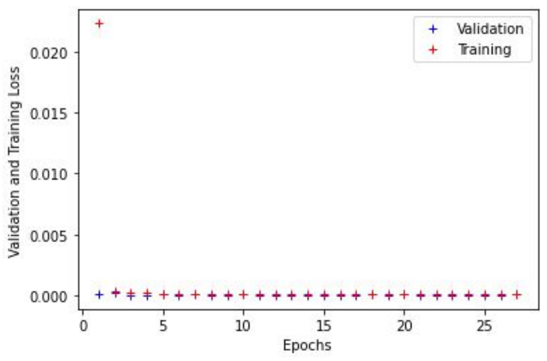
The model was compiled and fit the same way as the previously built long short term model. The regularisation techniques dropout, layer normalisation and parameter reduction were all investigated in the same manner and the ideal structure was found to be: two convolutional layers separated with 50% dropout and layer normalisation followed by a dense layer.

Finally, the wavelet transformed data generators were trialled. These generators were used in training the previously found best models. It was found that peak validation minimum loss had increased in both models upon this implementation. This is a reflection that the wavelet noise reduction removed signal rather than noise and thus this data was not used again.

Separately, the long short term model outperformed the convolutional network. Peak minimum validation loss for the LSTM and CNN were 0.8*10^-6 and 0.0006 respectively. However, the literature review suggested that two inferior models could be combined through ensemble learning to create one model with a greater performance.


input_12: InputLayer
conv1d_6: Conv1D
average_pooling1d_6: AveragePooling1D
lstm_6: LSTM
layer_normalization_6: LayerNormalization
lstm_7: LSTM
dropout_6: Dropout
dense_13: Dense
conv1d_7: Conv1D
average_pooling1d_7: AveragePooling1D
layer_normalization_7: LayerNormalization
dropout_7: Dropout
flatten_3: Flatten
dense_14: Dense
concatenate_11: Concatenate
dense_15: Dense

Keras's functional API was used to create two sets of layers representing the previously established ideal structure for the two kinds of networks. To combine these two models, the Concatenate layer was used. This allowed for the combination of the two outputs from these models despite their different shapes. After these outputs are combined, they are input into another Dense layer and finally fit to the data.

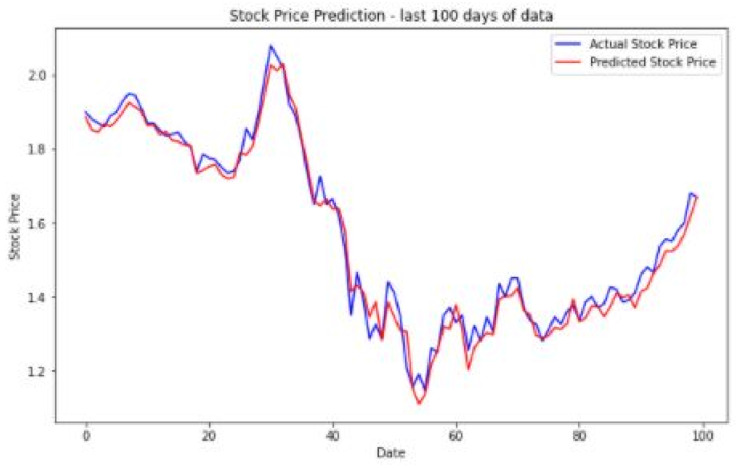


This ensemble approach resulted in a new minimum in validation loss. Furthermore, the model seemed to have improved in generalisation as can be seen by the convergence of training loss and validation loss.

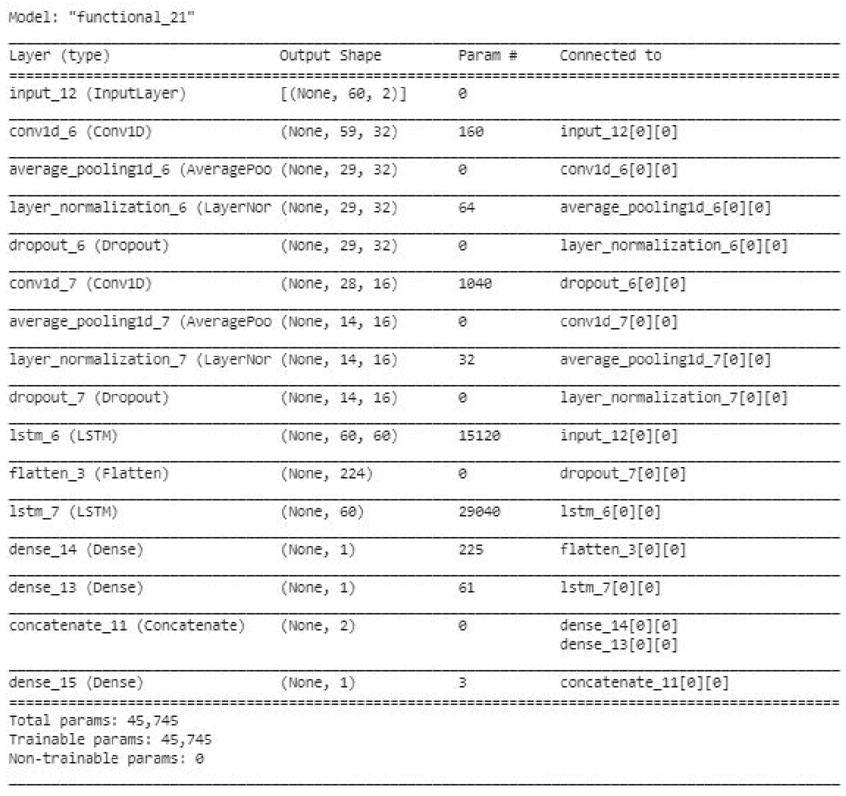
Evaluation and Prediction

The final merged model had a test mean squared error of 0.0008 with parameters of roughly 46000. The naive baseline for tomorrow's stock price prediction is today's stock price, and this approach results in a mean squared error of roughly 0.025. When compared to this, the ensemble model is much more effective.

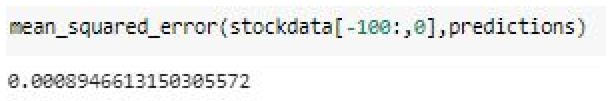
The following graph shows the model and its predictions on the last 100 days of data. As can be seen, the model predictions are slightly lagging the actual stock prices. This is a reflection of the model reacting to data as opposed to predicting, however, the model does maintain some predictive capacity as shown by its beating of the baseline.



Furthermore, it is not known if this model beats the baseline by a great enough margin to overcome transaction costs. This could render all of the gains of the model obsolete and result in no real profit.



```
Model: "functional_21"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_12 (InputLayer)           [(None, 60, 2)]      0
__________________________________________________________________________________________________
conv1d_6 (Conv1D)               (None, 59, 32)       160         input_12[0][0]
__________________________________________________________________________________________________
average_pooling1d_6 (AveragePoo (None, 29, 32)       0           conv1d_6[0][0]
__________________________________________________________________________________________________
layer_normalization_6 (LayerNor (None, 29, 32)       64          average_pooling1d_6[0][0]
__________________________________________________________________________________________________
dropout_6 (Dropout)             (None, 29, 32)       0           layer_normalization_6[0][0]
__________________________________________________________________________________________________
conv1d_7 (Conv1D)               (None, 28, 16)       1040        dropout_6[0][0]
__________________________________________________________________________________________________
average_pooling1d_7 (AveragePoo (None, 14, 16)       0           conv1d_7[0][0]
__________________________________________________________________________________________________
layer_normalization_7 (LayerNor (None, 14, 16)       32          average_pooling1d_7[0][0]
__________________________________________________________________________________________________
dropout_7 (Dropout)             (None, 14, 16)       0           layer_normalization_7[0][0]
__________________________________________________________________________________________________
lstm_6 (LSTM)                   (None, 60, 60)       15120       input_12[0][0]
__________________________________________________________________________________________________
flatten_3 (Flatten)             (None, 224)          0           dropout_7[0][0]
__________________________________________________________________________________________________
lstm_7 (LSTM)                   (None, 60)           29040       lstm_6[0][0]
__________________________________________________________________________________________________
dense_14 (Dense)                (None, 1)            225         flatten_3[0][0]
__________________________________________________________________________________________________
dense_13 (Dense)                (None, 1)            61          lstm_7[0][0]
__________________________________________________________________________________________________
concatenate_11 (Concatenate)    (None, 2)            0           dense_14[0][0]
                                                                 dense_13[0][0]
__________________________________________________________________________________________________
dense_15 (Dense)                (None, 1)            3           concatenate_11[0][0]
==================================================================================================
Total params: 45,745
Trainable params: 45,745
Non-trainable params: 0
__________________________________________________________________________________________________
```



```
mean_squared_error(stockdata[-100:,0],predictions)
```

```
0.0008946613150305572
```

Reference List

Vanstone, B & Hahn, T (2011) Designing Stock Market Trading Systems With And Without Soft Computing.

Persio, L.D., & Honchar, O. (2017). Recurrent Neural Networks Approach to the Financial Forecast of Google Assets.

Qiu J, Wang B, Zhou C (2020) Forecasting stock prices with long-short term memory neural network based on attention mechanism.

Hiransha M, Gopalakrishnan E.A., Vijay Krishna Menon, Soman K.P. (2018), NSE Stock Market Prediction Using Deep-Learning Models

S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon and K. P. Soman (2017), "Stock price prediction using LSTM, RNN and CNN-sliding window model," 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)

Zahra Berradi, Mohamed Lazaar, Integration of Principal Component Analysis and Recurrent Neural Network to Forecast the Stock Price of Casablanca Stock Exchange (2019), Procedia Computer Science, Volume 148

Jarrah, Mu'tasem & Salim, Naomie. (2019). A Recurrent Neural Network and a Discrete Wavelet Transform to Predict the Saudi Stock Price Trends. International Journal of Advanced Computer Science and Applications.

Pawar, Kriti & Jalem, Raj & Tiwari, Vivek. (2019). Stock Market Price Prediction Using LSTM RNN: Proceedings of ICETEAS 2018.

Di Persio, Luca & Honchar, Oleksandr. (2016). Artificial neural networks architectures for stock price prediction: Comparisons and applications.

Krstanovic, Sascha & Paulheim, Heiko. (2017). Ensembles of Recurrent Neural Networks for Robust Time Series Forecasting. 34-46.

Liu, Shuanglong & Zhang, Chao & Ma, Jinwen. (2017). CNN-LSTM Neural Network Model for Quantitative Strategy Analysis in Stock Markets.