Predicting Stock Price Movement and Volatility using Multimodal and Multitask Deep Learning

In fulfilment of the course

ECON F366

Laboratory Oriented Project

Submitted by
Amitojdeep Singh
2014B3A70615P

To
Rajan Pandey
Lecturer, Economics and Finance Department
BITS Pilani, Pilani Campus

Acknowledgement

I would like to thank Rajan Pandey Sir who has guided me through this project as my instructor for the Laboratory Oriented Project. Without his constant support & guidance this work would not have been complete.

Index

Abstract

This work implements a Deep Learning Model that utilizes Multimodal & Multitask architecture to utilize information from both price and news data to predict stock price movement and volatility. A multivariate model relying on price data has also been implemented to provide a direct comparison. Prior research work is based on either only news data to make event driven predictions or only the price data to predict using patterns of price movement. The results for this approach are positive and give an insight into utilizing upcoming deep learning architectures for financial prediction tasks. An accuracy of 74.93% for predicting price change direction and MSE of 0.0017 for volatility prediction have been obtained

1. Introduction

Stock Market Predictions have always been an important and popular area of research for Data Scientists as well as Financial Experts. Successful prediction of a stock's future price can lead to significant profits and thus is of great interest to Financial Institutions like Mutual Funds as well as private individuals. Primarily there are three different prediction methodologies:

1. *Fundamental Analysis*: It's major concern is with the company that underlies the stock. Company's past performance and credibility of its accounts is judged using techniques like Ratio Analysis.
2. *Technical Analysis:* Price history of the company is used to identify past trends and numerous patterns and techniques like Moving Average are used for predictions.
3. *Data Mining Technologies:* Artificial Neural Networks are being widely employed for a variety of applications including Stock Price Predictions.

Typically the dataset on which the model is trained is either price dataset, which can be treated as a machine learning based technical analysis and has achieved a state of the art accuracy of around 63%. Recently Event Driven Stock Price Prediction has become prominent due to its higher accuracy in short term predictions using news information. These models have high dependence on frequency of price and news information and require huge computational power. But the financial gains provided by them are core to the earnings of most of the trading firms today and they are widely implemented commercially.

In this work the deep learning model that takes two input sources - price & news data and that gives two outputs - price movement & volatility is proposed. The model trains two LSTMs one on each of the inputs and then concatenates their outputs using learned weights to produce the two outputs. Normalized volatility that lies between 0 - 1 is used to remove the effect of changing average price over the long period. a deep learning model using Open, High, Low and Close prices has also been formulated to observe the advantage of multimodal approach over conventional techniques.

2. Literature Review

In paper titled *Deep Learning for Event-Driven Stock Prediction*[1] *,* Xiao Ding et al. has proposed a deep learning based approach for making event driven stock price prediction. It has used events which were extracted from news text and represented them as vectors using the word2vec model to predict S&P 500 prices better than systems trained on historical data. Textual data has to be represented using constant dimension vector in order to preserve semantics of the words. It must be noted that Lavrenko et al. in *Mining of concurrent text and time series* [2] employed Bag-of-Words based document representation. But, this method is fails to preserve the semantics or word order from the original documents. The embeddings used are known to capture semantics better than bag of words model. For example, the two phrases “Apple sues Samsung” & “Samsung sues Apple” would be inferred with respect to their context in the vector space and will have different implications on stock prices of each of the two companies. Whereas a simple bag of words model would infer them identically leading to mispredictions.

The model employed by Xiao Ding et al. relies upon news data to identify events as well as associated semantics to make predictions. They develop a Deep Learning model with 3 inputs - Long term events, Mid-term events & short term events. It uses convolutional layers and max pooling on top of long & mid term events, whereas they feed the short term events directly as vector embedding to the hidden layer. In market simulation, a greedy strategy was used by their model to yield higher returns than the baseline illustrating the potential of deep learning models for stock price predictions.

Ryo Akita et. al in paper *Deep Learning for Stock Prediction Using Numerical and Textual Information* [3] use Multimodal Deep Learning model using price & news article data. It was able to obtain encouraging results for fifty companies listed on Tokyo Stock Exchange. The model also considered correlation between multiple companies of same industry to further leverage the performance.

3. Data Description

Two datasets used in this work are:

1. Reddit Finance News Dataset: Crawled Historical News Headlines from Reddit Finance News Channel (reddit.com/r/finance). It has top 25 headlines for each date from August 2008 to July 2016, i.e 8 years. This is chosen for it's reliability & low noise because it provides a select number of news which are of very high importance and can affect stock market as a whole directly. Model learns which all types of news headlines can affect stock prices as it can correlate stock market fluctuations to semantics of news items and identify the pattern over the large dataset.

2. DJIA Stock Price Dataset: Dow Jones Industrial Average (DJIA) data for each trading day from August 2008 to July 2016 is used for stock prices. Further, OHLCV (Open, High, Low, Close, Volume) tuple has been used to train the model.

4. Proposed Approach

This section describes the reference model based on CNN and the main model based on multimodal and multitask design using LSTM as it's key component.

4.1 Multivariate model

A model based on Convolutional Neural Network Architecture has been designed as a reference point. The stock price information is fed to the model as OHLCV (Open, High, Low, Close, Volume) tuple. The model then does 1D convolution on it and normalizes the activations across the batch. This is followed by Leaky ReLu activation layer which provides non linearity to the model. Some weights are dropped out at random to prevent overfitting before repeating the above block again. The output of this block is flattened and a dense layer is applied to it. This is followed by batch normalization, Leaky ReLu and another dense layer. The output generated is in the form of activation layer. The trained weights are saved and are used to make predictions as discussed in the results section. The model is shown schematically in the Figure 1.

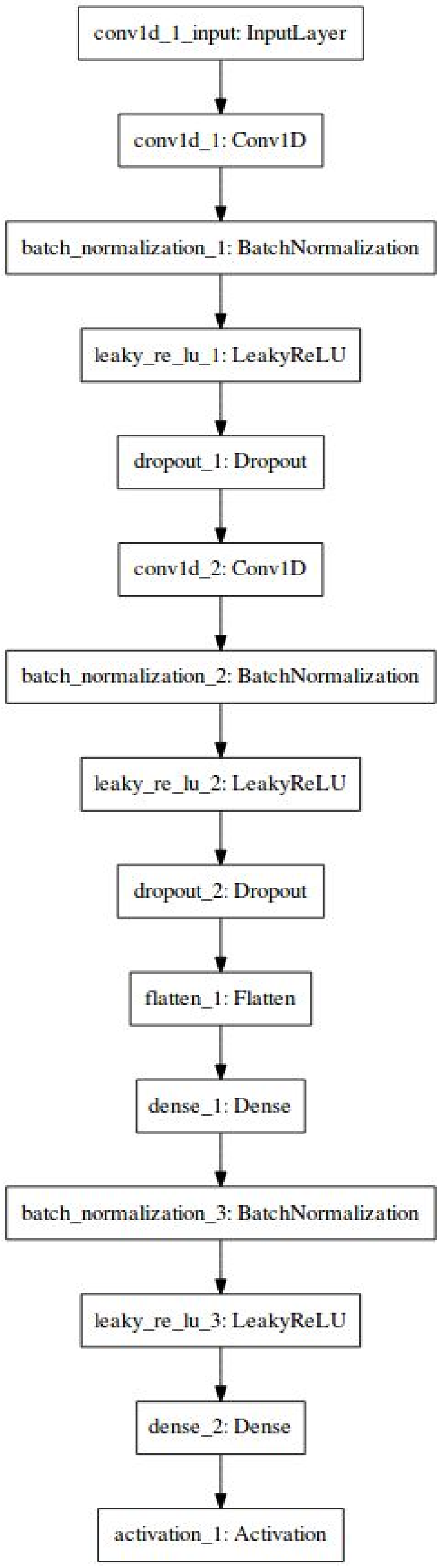
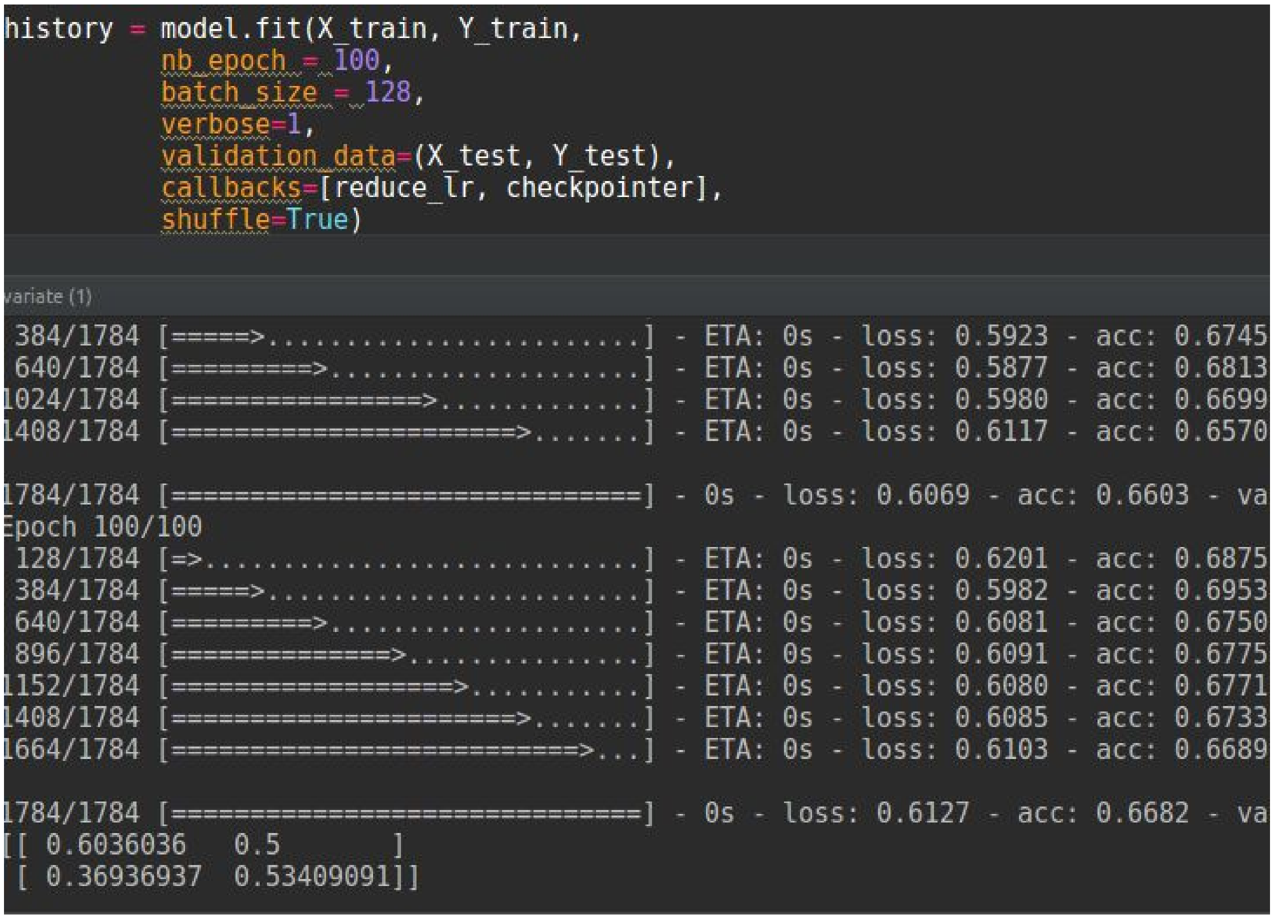


Figure 1: Model Summary for Multivariate Model

The model is compiled with Nadam optimizer and categorical cross entropy loss function is used. It is trained for 100 epochs and then best weights from those are saved for making predictions. A snapshot of the model training is shown below.



```
history = model.fit(X_train, Y_train,
          nb_epoch = 100,
          batch_size = 128,
          verbose=1,
          validation_data=(X_test, Y_test),
          callbacks=[reduce_lr, checkpointer],
          shuffle=True)

variate (1)
 384/1784 [=====>.......................] - ETA: 0s - loss: 0.5923 - acc: 0.6745
 640/1784 [=========>...................] - ETA: 0s - loss: 0.5877 - acc: 0.6813
1024/1784 [================>............] - ETA: 0s - loss: 0.5980 - acc: 0.6699
1408/1784 [======================>.......] - ETA: 0s - loss: 0.6117 - acc: 0.6570

1784/1784 [==============================] - 0s - loss: 0.6069 - acc: 0.6603 - va
Epoch 100/100
 128/1784 [=>............................] - ETA: 0s - loss: 0.6201 - acc: 0.6875
 384/1784 [=====>.......................] - ETA: 0s - loss: 0.5982 - acc: 0.6953
 640/1784 [=========>...................] - ETA: 0s - loss: 0.6081 - acc: 0.6750
 896/1784 [==============>...............] - ETA: 0s - loss: 0.6091 - acc: 0.6775
1152/1784 [==================>...........] - ETA: 0s - loss: 0.6080 - acc: 0.6771
1408/1784 [======================>.......] - ETA: 0s - loss: 0.6085 - acc: 0.6733
1664/1784 [==========================>...] - ETA: 0s - loss: 0.6103 - acc: 0.6689

1784/1784 [==============================] - 0s - loss: 0.6127 - acc: 0.6682 - va
[[ 0.6036036   0.5       ]
 [ 0.36936937  0.53409091]]
```

Figure 2 : Training the Multivariate CNN model

4.2 Multimodal and Multitask model

This section discusses the process of embedding the news headlines briefly followed by an expository description of the Multitask and Multimodal model used in this work.

4.2.1. Embedding Generation:

Google's word2vec has been used to turn words to embeddings. It is a Vector Space Model (VSM) that maps similar words to nearby points. So, similar words are represented by numbers that are closer to each other and this provides the representation a semantic value to it. This is more powerful than representing words as unique ids with no relation between words with closer ids as helps in training models that leverage the semantics of words to create context. Vocabulary dictionary stores word to index mapping and Inverse Vocabulary dictionary stores index to word mapping. A news item is converted to a list of unique ids of the constituent words. Word2vec model with 100 dimensions has been used to optimize the training speed and capture of semantics. The process is illustrated in the Figure 3.

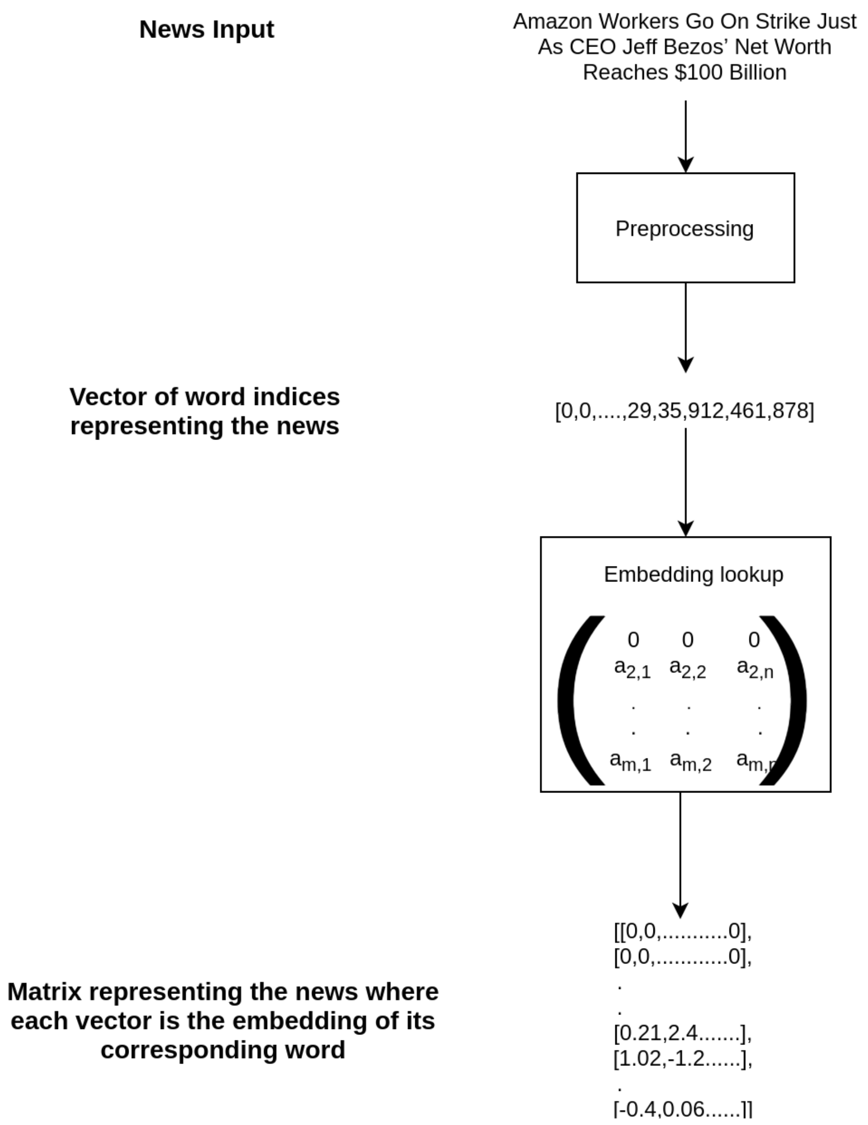

News Input
Amazon Workers Go On Strike Just As CEO Jeff Bezos' Net Worth Reaches $100 Billion
Preprocessing
Vector of word indices representing the news
[0,0,....,29,35,912,461,878]
Embedding lookup
0 0 0
$a_{2,1}$ $a_{2,2}$ $a_{2,n}$
$a_{m,1}$ $a_{m,2}$ $a_{m,n}$
Matrix representing the news where each vector is the embedding of its corresponding word
[[0,0,...........0],
[0,0,............0],
[0.21,2.4.......],
[1.02,-1.2......],
[-0.4,0.06......]]

Figure 3: Embeddings of a news headline

4.2. Model Design

The model follows a multimodal and multitask design as shown in Figure 4. Inputs from stock price data and news vectors are fed into two separate LSTMs. These are flattened to same number of dimensions and are concatenated to form a single layer of 600 dimensions. This layer is further used as an input to two dense layers, one which predicts volatility and the other price movement. MSE is used to measure volatility loss because it is a regression problem whereas price movement loss is measured using binary cross entropy. Also the dense layers use leaky relu activation function and a dropout of 0.5 to prevent overfitting.

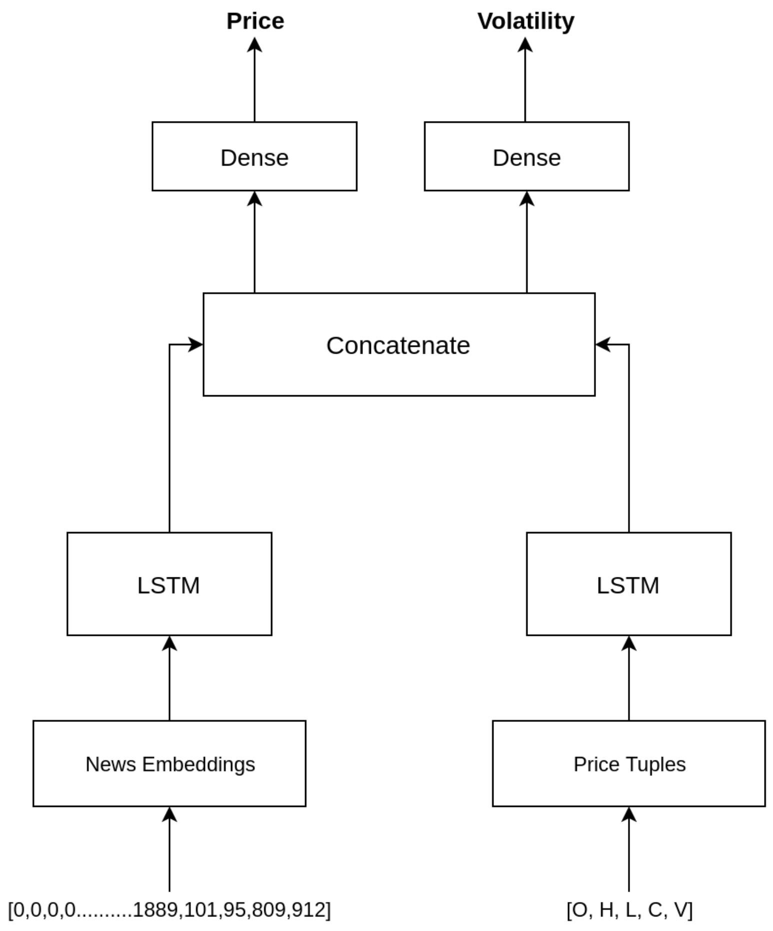


Figure 4: Multimodal and Multitask Model

The summary of model as shown in Figure 5 describes the internals of the model in detail. ts_input is the time series of stock market price and text_input is the news headline data of a given date. The stock price input goes to two LSTM units sequentially before getting flattened for concatenation. The news headlines are also passed through two sequential LSTM block and then the output is flattened. Concatenation unit joins together the activations from price and news LSTM units.

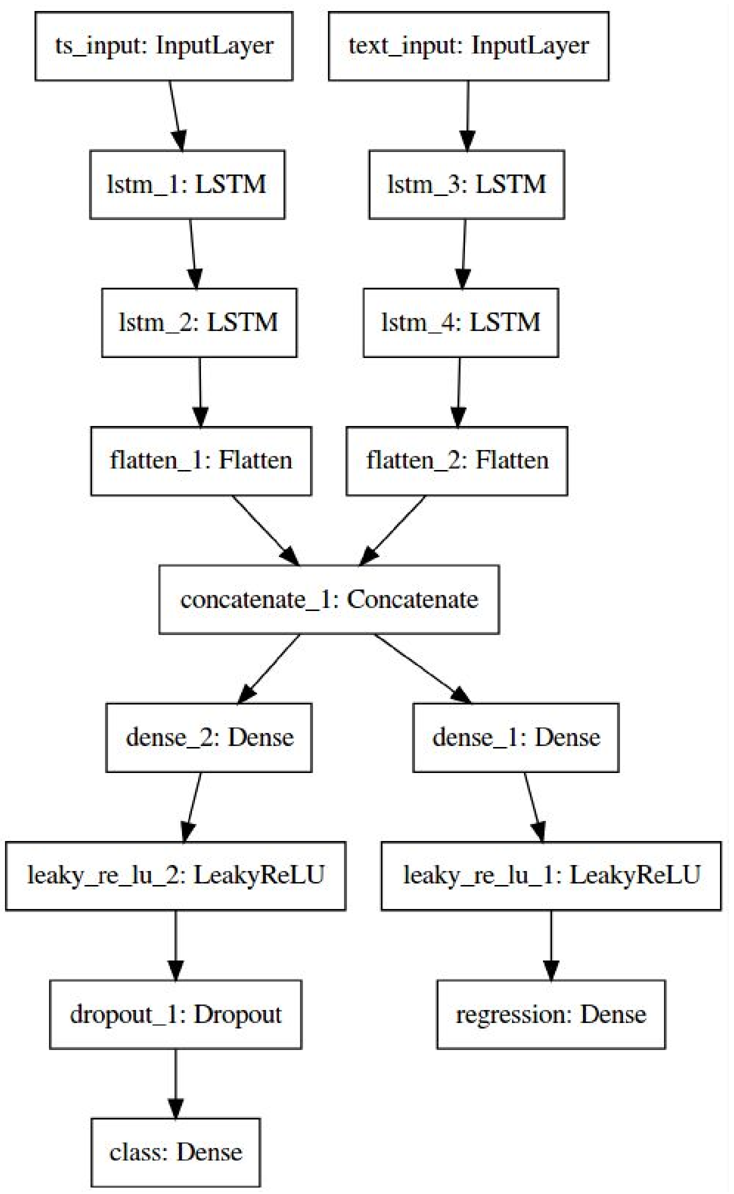
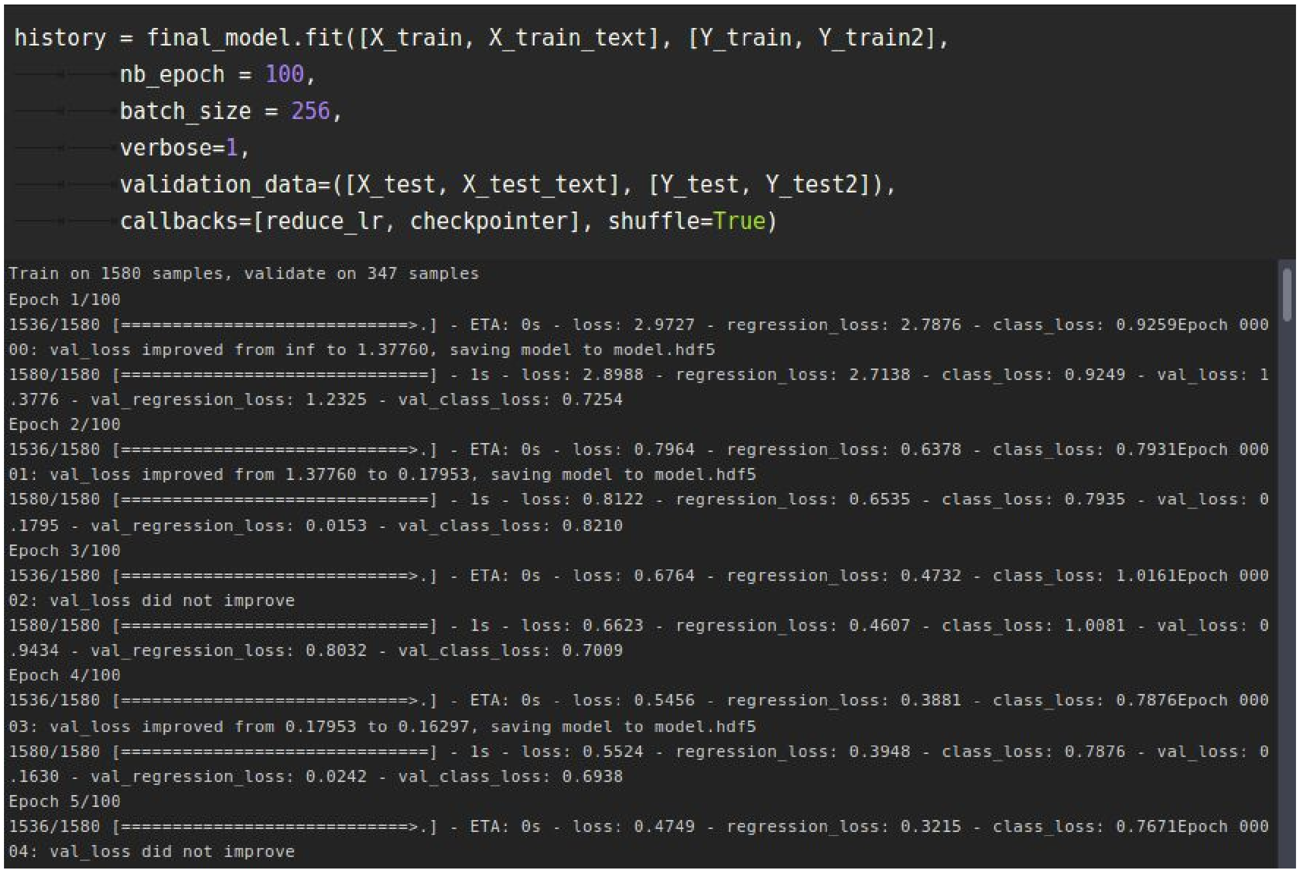


Figure 5: Model Summary for Multimodal and Multitask model

The concatenated layer goes as input to one Dense layer followed by a Leaky ReLu activation layer and a dropout layer. This gives the predicted direction of movement of stock price movement. Here 0 indicates a prediction of downward price movement and 1 indicates a predicted upwards price movement.

This concatenated layer is also fed to another dense layer with a Leaky ReLu activation which predicts volatility of the price. It must be noted that this volatility is a normalized continuous value where 0 means no volatility and 1 denotes the volatility of the date in the dataset with most volatility. For other days the volatility predicted is the fraction of maximum volatility.

The model is compiled using Mean Square Error loss for volatility and binary cross entropy loss for the predicted price movement direction. A snapshot of the training process is as shown in Figure 6. The model was trained for 100 epochs and the weights with the least total loss were used for making predictions.



```
history = final_model.fit([X_train, X_train_text], [Y_train, Y_train2],
        nb_epoch = 100,
        batch_size = 256,
        verbose=1,
        validation_data=([X_test, X_test_text], [Y_test, Y_test2]),
        callbacks=[reduce_lr, checkpointer], shuffle=True)
```

```
Train on 1580 samples, validate on 347 samples
Epoch 1/100
1536/1580 [============================>.] - ETA: 0s - loss: 2.9727 - regression_loss: 2.7876 - class_loss: 0.9259Epoch 000
00: val_loss improved from inf to 1.37760, saving model to model.hdf5
1580/1580 [==============================] - 1s - loss: 2.8988 - regression_loss: 2.7138 - class_loss: 0.9249 - val_loss: 1
.3776 - val_regression_loss: 1.2325 - val_class_loss: 0.7254
Epoch 2/100
1536/1580 [============================>.] - ETA: 0s - loss: 0.7964 - regression_loss: 0.6378 - class_loss: 0.7931Epoch 000
01: val_loss improved from 1.37760 to 0.17953, saving model to model.hdf5
1580/1580 [==============================] - 1s - loss: 0.8122 - regression_loss: 0.6535 - class_loss: 0.7935 - val_loss: 0
.1795 - val_regression_loss: 0.0153 - val_class_loss: 0.8210
Epoch 3/100
1536/1580 [============================>.] - ETA: 0s - loss: 0.6764 - regression_loss: 0.4732 - class_loss: 1.0161Epoch 000
02: val_loss did not improve
1580/1580 [==============================] - 1s - loss: 0.6623 - regression_loss: 0.4607 - class_loss: 1.0081 - val_loss: 0
.9434 - val_regression_loss: 0.8032 - val_class_loss: 0.7009
Epoch 4/100
1536/1580 [============================>.] - ETA: 0s - loss: 0.5456 - regression_loss: 0.3881 - class_loss: 0.7876Epoch 000
03: val_loss improved from 0.17953 to 0.16297, saving model to model.hdf5
1580/1580 [==============================] - 1s - loss: 0.5524 - regression_loss: 0.3948 - class_loss: 0.7876 - val_loss: 0
.1630 - val_regression_loss: 0.0242 - val_class_loss: 0.6938
Epoch 5/100
1536/1580 [============================>.] - ETA: 0s - loss: 0.4749 - regression_loss: 0.3215 - class_loss: 0.7671Epoch 000
04: val_loss did not improve
```

Figure 6: Training the Multimodal and Multitask Model

5. Analysis and Results

This section presents the results of both the models and a comparison studying the advantage of multimodal and multitask model over multivariate convolutional model.

5.1. Multivariate CNN Model

The model was trained for 100 epochs and the weights with least loss l were used for making the final predictions. Model loss measures the difference between true and predicted values of the price using categorical cross entropy. The price movement is a 0-1 classification problem and hence categorical cross entropy, which reduces to binary cross entropy in this case is used a measure of loss. The results of minimizing the loss over the course of 100 epochs are as shown in Figure 7.

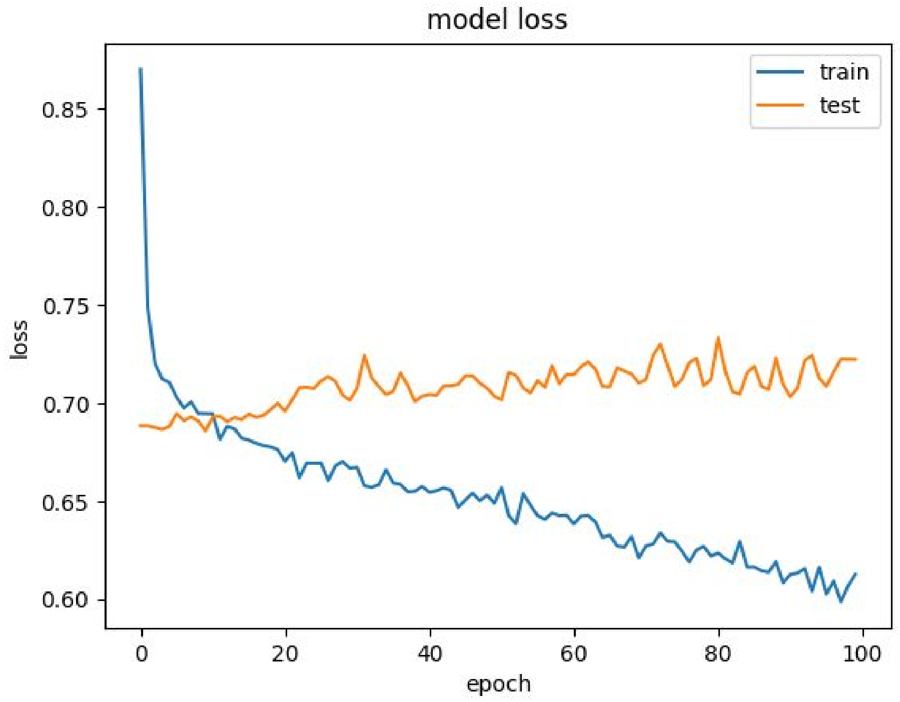


Figure 7: Training and Test loss of Multivariate CNN model

Initially the training loss is very high and test loss is moderate. After around 15 epoch test loss achieves a minimum value and the model is well fitted at this stage because the test and training loss are similar. After around 20 epochs over fitting sets in and training loss continues to fall with test loss gradually climbing. This is indicative of low informativeness of the price information to predict price movement direction.

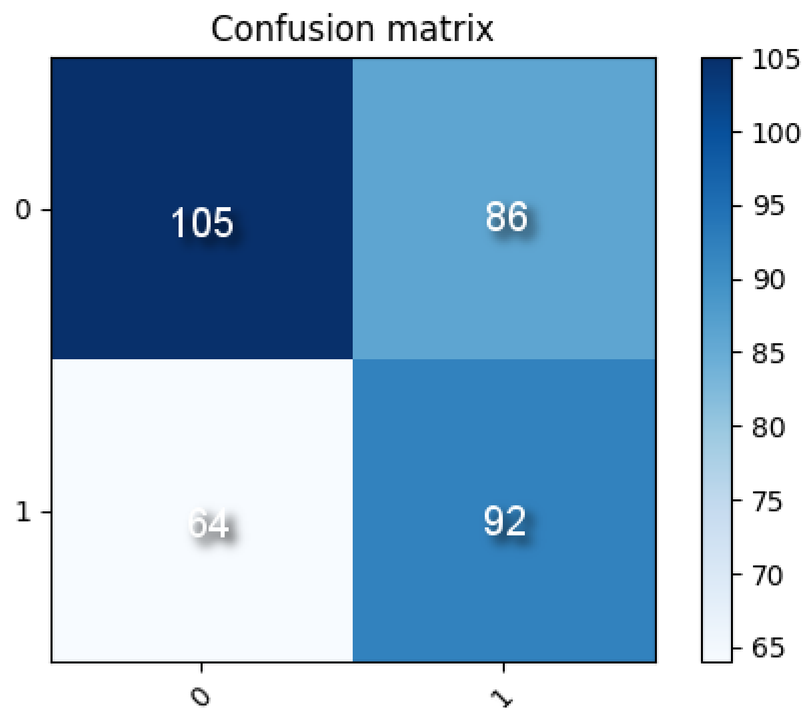


Figure 8: Confusion Matrix for Multivariate CNN

The model achieves an **accuracy of 56.77%** on the test set of randomly chosen 347 trading days of the given dataset. Confusion matrix in Figure 8 shows the relationship between true and predicted price movement for each of the 347 trading days of the test set. X axis has the true labels and Y axis is the predicted labels. In case of 178 days of positive price movement, the model predicts correctly for 92 cases and predicts a downward trend for the remaining 86. While for the 169 days with downward price movement, the model predicts correctly for 105 cases but predicts increase in price for 64 days. The model is distinctly better at predicting negative price movements as compared to positive one's.

5.2 Multimodal and Multitask Model

The model was trained for 100 epochs and the weights with lowest loss amongst all were used to make the final predictions. Model loss measures the difference between true and predicted values of the price and volatility. The price movement is a 0-1 classification problem and hence binary cross entropy is used a measure of loss. Whereas volatility is a continuous measure and thus mean squared error is used as loss function. The results of minimizing combined loss are as shown in Figure 9. Initially both test and training loss are quite high as the model has random weights and they decrease rapidly as epochs go on. After around 20 epochs the test and training loss converge and settle indicating a well-fitted model.

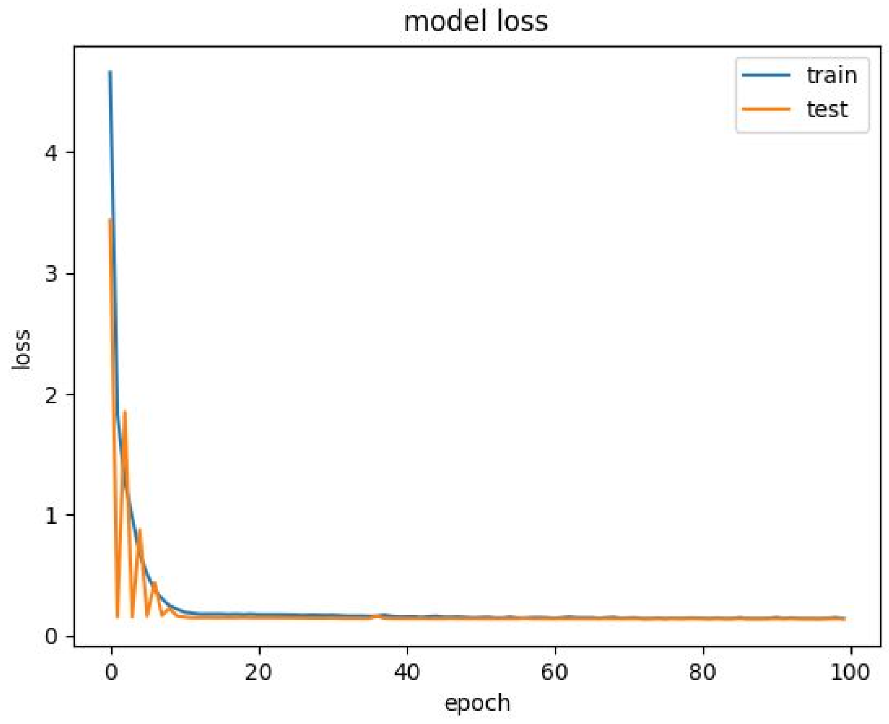


Figure 9: Change in training and test loss over the epochs

The best weights from the above training are used to make predictions over the test set obtained using a random set of 10% of the total trading days. The metrics observed for volatility are:

MSE = 0.0017

MAE = 0.0708

MAPE = 3.1578

Figure 10 gives a plot comparing predicted and actual volatility for the test set. It is clear that the model is able to capture most of the dependencies and big jumps quite well giving a good estimate of market risk on any given day.

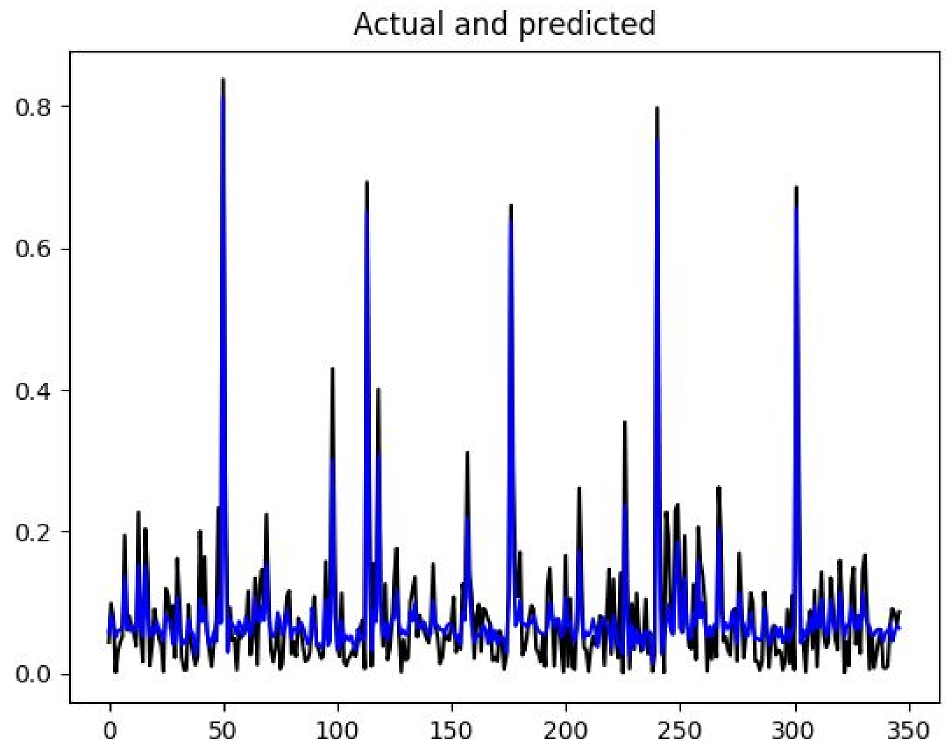


Figure 10: Actual volatility in black and predicted volatility in blue

Model achieves an **accuracy of 74.93%** on this test set which is better than any technique that is based solely on price data. It is in line with sophisticated models that use only news data despite its simplistic design and treatment of stock market as a whole rather than taking individual stocks.

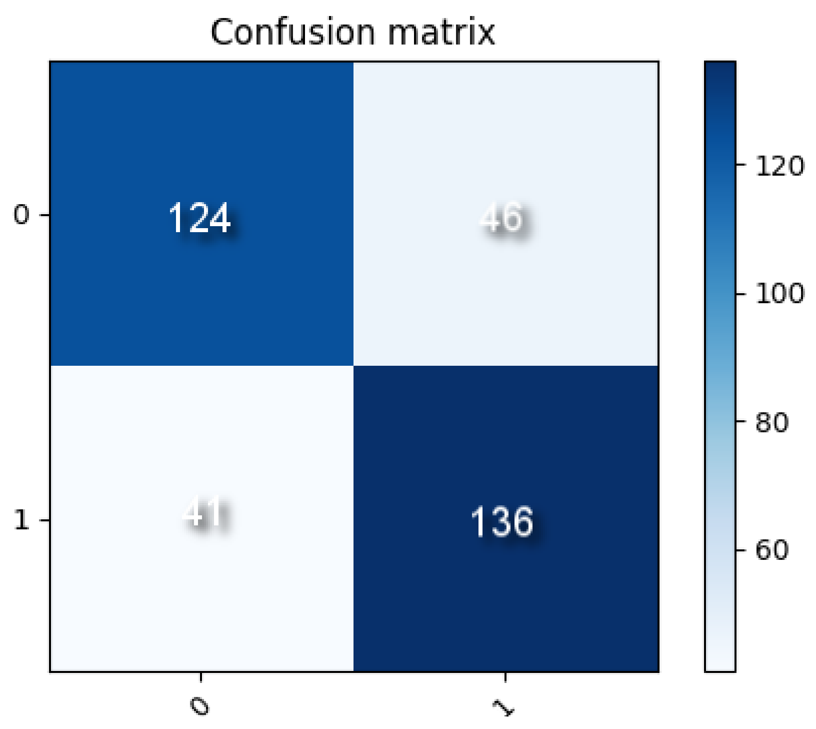


Figure 11: Confusion Matrix with true labels on X axis and predicted labels on Y axis
0 - price decrease and 1- price increase

Confusion matrix shown in Figure 11 shows the relationship between true and predicted price movement for each of the 347 trading days of the test set. In case of 182 days of positive price movement, the model predicts correctly for 136 cases and predicts a downward trend for the remaining 46. While for the 165 days with downward price movement, the model predicts correctly for 124 cases but predicts increase in price for 41 days.

The model is slightly better at predicting positive price movements as compared to negative one's but has a very balanced confusion matrix overall. This indicates the robustness and predictive power of the model.

6. Conclusion

Multimodal and Multitask Deep Learning holds great potential for stock price predictions and has been shown to achieve better results than techniques relying solely on price data by supplementing it with news data. An accuracy of 74.93% for predicting price change direction and MSE of 0.0017 for volatility prediction have been obtained beating the 56.77% for a comparable model that relies on just price data. The model was found to be predicting major volatility trends quite well. Also the confusion matrix for the price movement direction is quite well balanced testifying the robustness and efficacy of the model.

It is worth noting that the model uses news information for stock market as a whole instead of targeted news to make trading decisions for individual stocks which restricts its performance and at the same time makes the implementation simplistic. Also the high frequency price data and live news feed can be used to make a real time High Frequency Trading (HFT) system using the architecture of proposed model.

To sum it up, this work provides an exposure into the application of Multimodal and Multitask learning to financial time series data. The positive results obtained validate the credibility of model for more complex tasks. In future work trading strategies using a target set of companies can be applied to observe the model's performance for a real portfolio.

References

1. Ding, Xiao, et al. "Deep Learning for Event-Driven Stock Prediction." *Ijcai*. 2015.
2. Lavrenko, Victor, et al. "Mining of concurrent text and time series." *KDD-2000 Workshop on Text Mining*. Vol. 2000. 2000.
3. Akita, Ryo, et al. "Deep learning for stock prediction using numerical and textual information." *Computer and Information Science (ICIS), 2016 IEEE/ACIS 15th International Conference on*. IEEE, 2016.