Finding Micro Volatility Trends in Stock Data Using LSTM and DeepMind's WaveNet

Pratikkumar Patel
Department of Computer Science
Boston University
pratikp@bu.edu

Priyank Shah
Department of Computer Science
Boston University
prynk12@bu.edu

Anirvan Maiti
Department of Computer Science
Boston University
anirvan@bu.edu

Abstract

In this paper, we analyze stock market volatility using Long Short-Term Memory (LSTM) and comparing it to DeepMind's WaveNet adapted for time series data to see if techniques used in WaveNet help us find trend in stock market data and if it is in anyway more useful than using LSTM. In addition, we also propose a different way of looking at stock market data that might be helpful in predicting finer ups and downs in intraday price volatility.

1 Introduction

Many have tried forecasting volatile financial time series data with different techniques. But most them have resulted in unpromising results. This has led many to conclude that financial time series data is mostly random volatility combining with macro trends in industry and news around the company.

2 Time Series Data

2.1 Definition

As Wikipedia contributors (2017) says, Time series is a series of data points indexed in time order. Trend and values of data point depends on what has happened in the past. Time series data helps us understand changes around an entity over time. If there is any trend to the data, it makes it easier to understand and predict what will happen in the future. This has business benefits attached to it. We can see it in the examples shown below.

2.2 Examples

- Customer Behavior
- Seasonal Patterns
  - Ticket Sales
  - Polar Ice Caps Data
  - Weather Patterns
- Stock Prices

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

2.3 Problem Definition

We have chosen financial time series data from stock markets as it is a direct indicator of how a business is performing and predicting stock market prices can have major benefits for stock traders. There are many factors which can affect the stock prices. We have listed some of them below.

- Company Performance and technicals
  - This includes quarterly results, profits, price to earning ratio etc. Generally, these are declared at earnings call arranged by companies and made public through many different channels.
- Future of Business/industry
  - This is a very special feature where it behaves like ?Lifting tide raise all boats.? For example, Although a business might not be doing good, but if it?s in a sector that as a whole is growing really fast, prices of the company might go up in the future.
- Momentum
  - Many times, stock markets are carried away by momentum in the market and might sway one way or another just because everyone else is going that way. This kind of trends are easier to predict and many techniques like different moving averages, MACD (Moving average convergence divergence) or RSI (Relative strength index).
- News (Market, Company or Sector, Regulations)
  - This is one of the most unpredictable piece of understanding stock markets. News around a company, sector or regulations can be published anytime. Models that try to predict stock markets must need to incorporate news, market sentiment (stock tweets on twitter) to be accurate. This is currently out of scope for our paper.

Because of such factors, the prediction of future stock prices is really difficult. We are focusing on momentum and short term volatility, to find trends in a stock prices. For this kind of analysis, we have chosen to use Recurrent Neural Networks, Long Short-Term Memory Network (LSTM) and DeepMind's WaveNet. We'll discuss why we've chosen these models and results we've obtained using them in following sections.

3 DataSet

For our dataset, we used the stock prices of "Apple Inc." listed on Nasdaq Stock Exchange. We have downloaded the data from Nasdaq website for time period between Sept-2011 to March-2017.

Downloaded file from the stock exchange included fields listed below.

- Date
- Time
- Last Price
- Volume traded

These text files included all price changes happened in each second. Since, training a neural network on this kind of data is very resource intensive, we need to preprocess and make it smaller so that we can train our models on a normal computing platform with no additional computing resources added. We understand that creating a model, large enough to learn all trends in this data, might be large enough to not be trained fast enough on our laptops. Preprocessed data is our way of compromising between granularity of data and details of predictions we can get from our models.

4 Preprocessing

4.1 Weighted Average using Volume

The data we got from the exchange had tick prices. Which means, it had at least 28,000 records per day. This hugely increased processing time for your models. To reduce processing time, we

decided to reduce the size of data by compromising on granularity. We created 3 separate datasets with different interval sizes.

- 1-MIN : This dataset had weighted averages using volume over 1 minute interval. This reduced the number of records per day to at least 480.
- 5-MIN : In this dataset, we further increased the interval size to 5 minutes. This helped us get the number of prices to 40 per day.
- 15-MIN : Further increasing the interval size to 15 minutes helped us get the number of records per day to 32 per day.

In each section, we used the timestamp to do weighted average of the stock prices using volume. For example, if there are multiple tick prices for time stamp from 9:30 AM to 9:31 AM, we did the weighted average of these data to get the single value saved against 9:30 AM. For 5-MIN data, we did the same for 9:30 to 9:35.

4.2 Stock Split Adjustment

On 9th June 2014, Apple announced stock split for every 1 share everyone got 7 shares in return. This had an impact on the prices starting 9th June. So we had to adjust our stock prices and volume till 6th June, 2014 by dividing stock prices and multiplying volume by 7.

5 Recurrent Neural Network(RNN)

Because of the way traditional neural networks are designed, they cannot predict the future data prices using past data. The only way to do that is to use past data as feature set, which becomes really problematic when you have data going months in past. That is why we need to use the recurrent neural networks.

We have briefly explain Recurrent Neural Networks below. One distinct feature of RNN is that each cell in RNN takes 2 inputs. 1 from the input dataset and another from the output of previous RNN cell. This can be explained as having a loop within the cell itself. Below figure explains an RNN cell where input is taken from dataset and from itself and output is generated.

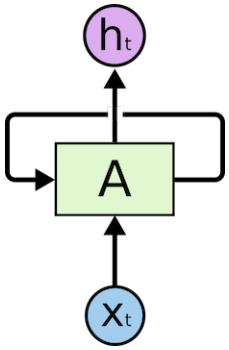

Figure 1: Rolled RNN

If we unroll this cell, it can be viewed as a list of traditional neurons in a neural network, where each cell is connected to the next one.

As shown in figure 2, each node uses the output from the previous node and predict the next node from that. This means, each output is only dependent on just the previous output. It doesn?t look at the entire past of the entity. This is actually a problem when comes to stock prediction because ideally we?d like to look at entire history of the stock before prediction.

For better prediction of future values, we have to use the complete past time series data instead of using just one previous data node. Because of this limitation of RNN, we decided to move on to Long Short-Term Memory Networks (LSTM)

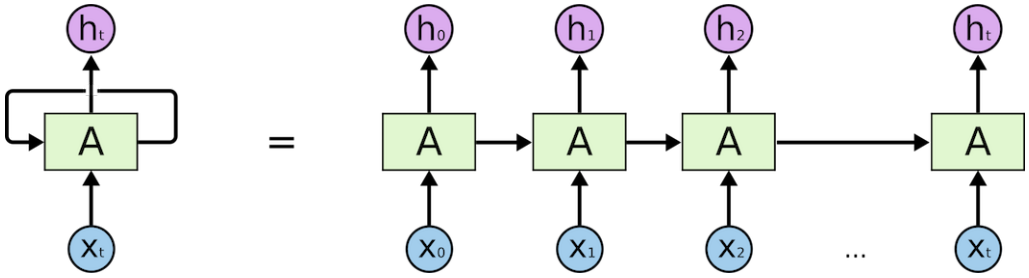

Figure 2: Unrolled RNN

6 Long Short Term Memory Networks (LSTM)

6.1 Definition

Per Olah (2015) Long Short Term Memory networks ? usually just called ?LSTMs? ? are a special kind of RNN, capable of learning long-term dependencies. LSTMs are explicitly designed to avoid the long term dependency problem.

In LSTM, A cell has ability to remove or add information to the cell state, carefully regulated by structures called gates.

Gates are composed out of a sigmoid neural net layer and a point wise multiplication operation. These gates are shown in figure 3.

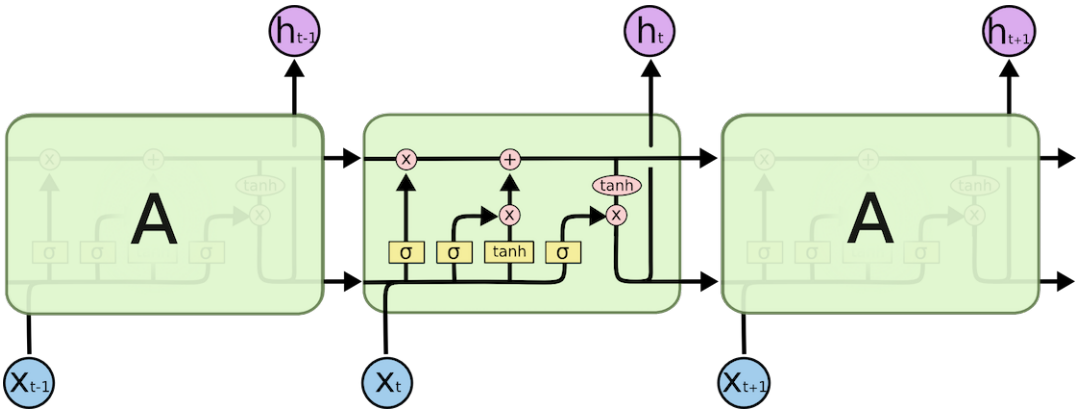

Figure 3: LSTM

Before we explain our models, one important parameter here that needs to be explained in look back. This parameter defines how many data points in the past does the network look back. This is used for generating training data that is suitable for direct input to the LSTM network. We have created 3 different models using LSTM. Starting with model 1 with 1 LSTM layer, we kept increasing number of LSTM layers stacked on top of each other to 3 layers in Model 3.

We also performed experiments with different look back parameter values. Results we got can be explained intuitively as well by looking at results from all experiments.

6.2 Our LSTM Models

We have followed guidelines from Siraj (2017) to implement our models. We have used Keras which was using TensorFlow in the backend. This has made our life easier in terms of implementing different LSTM models and tinker around before finalized our models. It also helped us get visual output of our model 3 that explains how LSTM and dense layers are stacked and what kind of inputs they are taking.

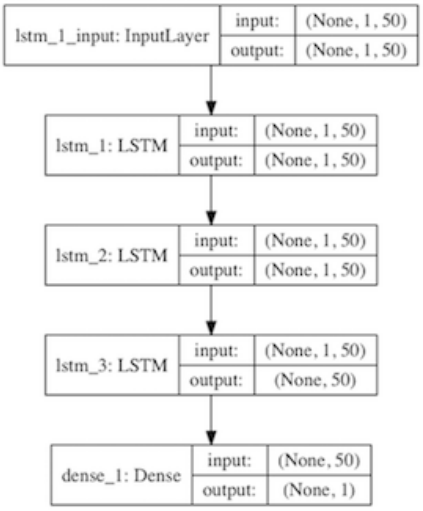

Figure 4: LSTM

Table 1: LSTM Training and Testing Results

Parameters	Model 1 (1 LSTM Layer)	Model 2 (2 LSTM Layers)	Model 3 (3 LSTM Layers)
Look Back : 50 Epochs : 30	Train Score: 0.53 RMSE Test Score: 0.63 RMSE Test Score: 14.91 RMSE	Train Score: 0.46 RMSE Test Score: 0.45 RMSE Test Score: 28.98 RMSE	Train Score: 0.45 RMSE Test Score: 0.56 RMSE Test Score: 57.02 RMSE
Look Back : 100 Epochs : 30	Train Score: 0.48 RMSE Test Score: 0.45 RMSE Test Score: 28.39 RMSE	Train Score: 0.41 RMSE Test Score: 0.92 RMSE Test Score: 15.30 RMSE	Train Score: 0.54 RMSE Test Score: 0.67 RMSE Test Score: 25.38 RMSE
Look Back : 50 Epochs : 50	Train Score: 0.39 RMSE Test Score: 0.36 RMSE Test Score: 31.22 RMSE	Train Score: 0.40 RMSE Test Score: 0.43 RMSE Test Score: 29.86 RMSE	Train Score: 0.35 RMSE Test Score: 0.39 RMSE Test Score: 56.66 RMSE

6.3 Discussion

Looking at the results, we can see that as we go from a simple 1 layer LSTM model 1 to model 3 with multiple LSTM layers, it starts to learn more short term volatility and Test Scores starts decreasing. But important thing to notice here is Model 1 is universally doing better. This can be explained intuitively by the fact that overall, there is not much to learn when it comes to intraday prices. It is difficult to find a trend. But improvement in Model 3 suggests that our model has started learning about these micro volatility little better than previous 2 models.

Also, one more observation we can make is based on epochs. As we increase epochs to 50, it reduces the train score and test scores. This suggests that there is a scope to learn and models can still be upgraded to reduce error rate in prediction.

Look back parameter has an interesting impact on the error rate. Although it increase immediate error rate for all models, it has major impact on the 2nd test score (3rd line in each cell). This test score is calculated over prediction that was done using our own prediction. We tried to predict the entire future of Apple stock using our model and calculated RMSE score. It was obvious why RMSE scores are so high when it comes to entire future. But increasing look back parameter has reduced this error quite a lot, suggesting that as we look far in the past, it improves ability to predict the future.

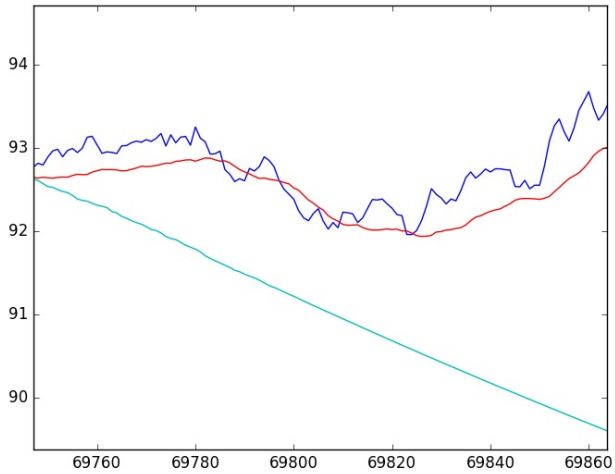

Figure 5: LSTM Predictions

6.4 LSTM Conclusion

As we discussed above, LSTM does learn some of the trends and tries to reduce prediction rate. But we are still not satisfied because our look back parameter reduces our ability to see past a day or two. In our case, we were using 5-MIN data, which means, our model was looking 2 days in the past and predicting the future. Although it learned over the entire history of Apple stock, since it just had past 2 days data to look at, it was never going to be too accurate to be useful. This led us to look for another way we can look in the history of stock prices before predicting the next value.

7 WaveNet

To solve the problem of short term history look back, we looked at Deepmind?s WaveNet project. This is a generative model for Raw Audio. Here is how they explained the necessity for WaveNet.

> Researchers usually avoid modelling raw audio because it ticks so quickly: typically 16,000 samples per second or more, with important structure at many time-scales. Building a completely autoregressive model, in which the prediction for every one of those samples is influenced by all previous ones (in statistics-speak, each predictive distribution is conditioned on all previous observations), is clearly a challenging task.
> Oord et al. (2016)

We have the same problem with Stock data because that also ticks so quickly. In stock prediction as well, every one of the samples are influenced by all the previous ones. This led us to hypothesize that using WaveNet like model might help us better predict micro volatility trends in stock data. Below is an image that explains how convolutional layers are used in WaveNet to stack up summarized layers on top of base RNN and restacking them again helps give better results.

7.1 Our Model

We have used opensourced code of WaveNet from Babuschkin (2017) to implement our version of WaveNet to train on stock data. We have used below model for stock data.

Dilations : [1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512]
Residual channels : 64
Dilation channels : 32

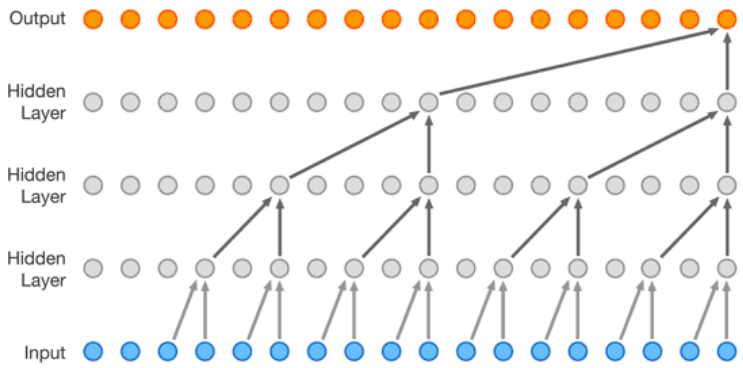

Figure 6: WaveNet Convolutions

Table 2: WaveNet Testing Results

No of Predictions	Dialations : 512 Filter width : 2 Steps : 3000
50	Test Score: 1.29 RMSE
100	Test Score: 1.63 RMSE

quantization channels: 1000

Here, dilations explains how convolutional layers are set up. Residual and Dilation Channels hyper parameters are used for configuring the model in such a way that it makes it easier for model to converge. Quantization channels is something that defines the range of input and output values that will be used for training and generated after model is trained.

7.2 WaveNet Testing Results

In figure 6 and 7, we can see that WaveNet model is able to predict micro trends in intraday stock prices with good accuracy. In these figures, blue line indicates ground truth and green line indicates predicted values. One of the problem with our model is, it takes a very long time (20 secs/prediction) to predict values for a day (includes predicting up to 50 values). Also, since scaling of data has been done differently for LSTM and WaveNet. That gives us different numbers when it comes to Test Errors in RMSE.

8 Future Work

As noted, current implementation doesn't allow us to compare test error RMSE number directly between LSTM and WaveNet. What we'd like to do is, train both model on same preprocessed and scaled data so that both models can be compared on level playing ground.

Another planned work includes performing frequency filters on tick data to analyze high frequency data using WaveNet and low frequency (major trends) using moving average models and combine those results into predicting daily chart for a stock. We are confident that these models are able to learn intrinsic patterns in stock volatility and will be able to predict them with high accuracy when upgrades are implemented.

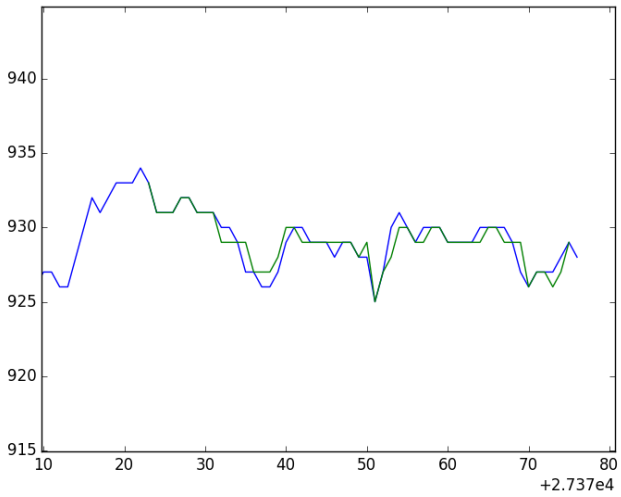

Figure 7: WaveNet Results Mapped on Ground Truth - 50 Predictions (1 Day)

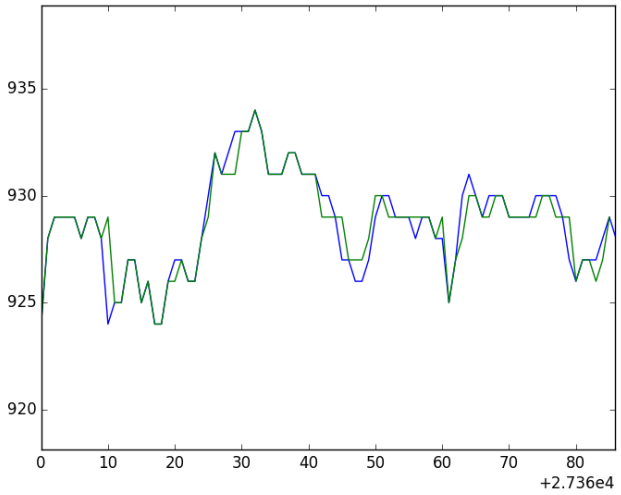

Figure 8: WaveNet Results Mapped on Ground Truth - 100 Predictions (2 Day)

References

Wikipedia contributors
Time series
Wikipedia, The Free Encyclopedia, May 2017
https://en.wikipedia.org/w/index.php?title=Time_series&oldid=773611204.

Christopher Olah
Understanding LSTM Networks
http://colah.github.io/, August 27, 2015
http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu
WAVENET: A GENERATIVE MODEL FOR RAW AUDIO
https://deepmind.com/, Sep, 2016

https://deepmind.com/blog/wavenet-generative-model-raw-audio/
https://arxiv.org/pdf/1609.03499.pdf.

Siraj
How to Predict Stock Prices Easily
https://github.com/llSourcell/, Mar, 2017
https://github.com/llSourcell/How-to-Predict-Stock-Prices-Easily-Demo.

Igor Babuschkin
A TensorFlow implementation of DeepMind's WaveNet paper
https://github.com/ibab/, Mar, 2017
https://github.com/ibab/tensorflow-wavenet.