# CS295P Final Report:
# Stock Prediction using Deep Learning

Contributors: Ammunje Karthik Nayak, Himanshu Jain, Nicholas Chung

March 20th, 2021

# Introduction

The stock market is an ever-fluctuating series of prices across thousands of different companies, and accurately predicting these fiscal changes would allow one to exploit to stock market for massive financial gains. While statistical models provide a mathematically motivated framework to forecast stock prices, we decide to add on neural networks to approximate future price trends. Working off the assumption of a market with simplified rules and fees, we predict prices on a random month within a set of non-contiguous years using a combination of stock analysis and deep learning.

# Data Pre-Processing

## Fundamental Data

We extract statistics related to recent stock performance from all the html files. To do this we have leveraged BeautifulSoup and Pandas to scrape the html files. We create a single table, for each month, with columns representing various financial parameters related to a stock and rows representing a company. For simplicity, we have skipped stocks from .to and pink slips.

To overcome inconsistency in data formatting and representation between the html files, we clean the data by normalising feature names, removing special characters and assigning appropriate data types to all features. We leave missing values as they are.

## Technical Data

We work primarily with the streaming data files and use the Pandas library to import file contents into a DataFrame object. In order to feed the data into our LSTM model and reduce the amount of data, we make two major assumptions:

1. Downsampling: averaging prices across each hour of the trading day provides a rough estimate of a stock's trend
2. Similarity across disjoint years: the stock market's movement within the 2001, 2006, and 2011 timeframes are similar enough to be modeled by the neural network

Prior to downsampling, we reassign the volume value of each transaction to represent the actual number of stocks traded in said transaction (as opposed to the total number of stocks traded up to and including the transaction in question). In addition, we keep only the price, volume, open, high, and low prices for each transaction to reduce the size of the data. To reduce the amount of data stored in RAM, we write (or append) the data for each stock to a corresponding file that contains all the data for the stock (one file per stock).

Ultimately, each transaction related to a company (in the resulting .csv file) represents a single hour in the trading day (constrained between 9:00 and 16:00 [7 hours of trading per day]).

# Prediction Architecture

## Fundamental Data

We use various parameters obtained from scraping html files to determine if a stock will perform well. By analysing PEG ratio, current ratio, price to earnings, debt, quarterly earnings etc a score is generated for each stock. For example, PEG ratio should ideally have a value under 1 and a company having PEG between 0-1 is assigned more points than a company with PEG greater than 1. Similarly, we assign points to each stock based on multiple parameters. The overall points accumulated by a stock is it's score and these scores can vary between -5 to 10, where a score of 10 implies that these stocks are the best buy.  Stocks are then sorted according to this score to obtain a list of best to worst stocks.

## Technical Data

### Neural Network

#### Background

Recurrent neural networks (RNNs) are a type of neural network model that allow the output at a certain time instance to depend on not only the input at the same time instance but also the inputs and outputs of previous time inputs. As illustrated in Figure 1, the RNN's defining feature - in a graphical representation of the network - is that it contains directed cycles.

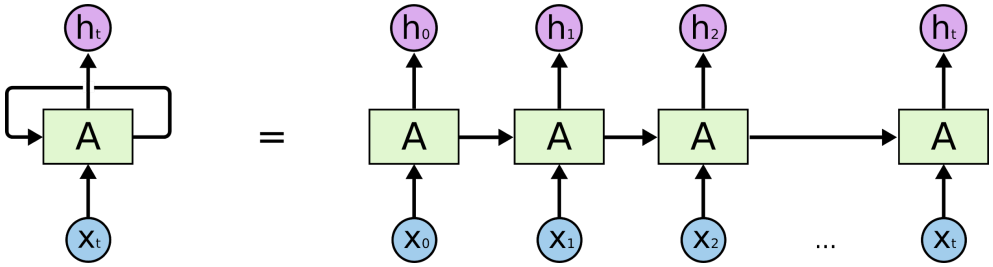


**Figure 1**: A recurrent neural model on the left and its feed-forward (non-cyclic/unrolled) equivalence on the right. In this example, the outputs $h_i$ are dependent on the inputs $x_{j\leq i}$ and outputs $h_{k<i}$. [1]

However, the effect of early inputs on later outputs decreases as the time series extends. One solution is to use an RNN variant called Long Short Term Memory (LSTM). Briefly, by passing not only the outputs but also a cell state through the directed cycle, an LSTM is able to decide what information to retain, forget, and modify. Figure 2 illustrates the LSTM's internal architecture.

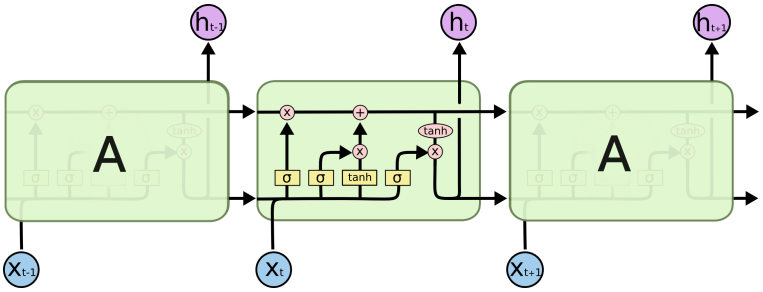


**Figure 2**: A diagram of an unrolled LSTM model indicating what functions/operations are performed. [1]

## Data and Model Setup

To train the model, we import the .csv files created when processing the streaming files. For each company's stock, we create our input data by creating multiple series of the hour-by-hour transactions. Specifically, each data point in our input data is a contiguous series of 70 transactions from one of the three years, in which each transaction contains information about the company stock's current, high, low, open, and closing price. In addition, in order to train the model not on the absolute values of prices but rather on the relative values, we normalized each of the five aforementioned prices using the min-max method. Using this method, the range of the values is transformed into a [0,1] range.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
batch_normalization (BatchNo (None, 70, 5)             20
_________________________________________________________________
lstm (LSTM)                  (None, 70, 70)            21280
_________________________________________________________________
dropout (Dropout)            (None, 70, 70)            0
_________________________________________________________________
lstm_1 (LSTM)                (None, 70, 70)            39480
_________________________________________________________________
dropout_1 (Dropout)          (None, 70, 70)            0
_________________________________________________________________
lstm_2 (LSTM)                (None, 70, 70)            39480
_________________________________________________________________
dropout_2 (Dropout)          (None, 70, 70)            0
_________________________________________________________________
lstm_3 (LSTM)                (None, 70)                39480
_________________________________________________________________
dropout_3 (Dropout)          (None, 70)                0
_________________________________________________________________
dense (Dense)                (None, 1)                 71
=================================================================
Total params: 139,811
Trainable params: 139,801
Non-trainable params: 10
```

While a single-layer LSTM model may be able to capture time-dependent patterns in the stock prices, stacking multiple layers of LSTMs allow the complete model to learn more complex patterns. [2] In our model, we use 4 sets of LSTM-Dropout layers to both increase the complexity of the model while preventing overfitting (via the Dropout layers). To compute the final price at the next time step, the model is finalized with a single Dense layer that outputs a single value.

## Selection from Prediction

With the trained model, we are able to make predictions one time-step into the future, which represents a single hour of a business day. We use the 1/e optimal stopping rule in combination with the output predictions to determine the best time step to sell each stock at. Once we've determined the time step, we then convert the time step back into a date and time (using the original date and time information imported from the .csv file) and search the streaming files for the most recent valid transaction time.

Stocks to buy are selected from the top 100 stocks outputted from the fundamental data prediction.

# Results

After training on 1000 stocks, we notice several things. First, the model does not converge on several company stock prices for a particular year, regardless of the number of epochs that is used. Nevertheless, for a large majority of stocks, the validation and test mean-squared error (MSE) are low and reflect the compropable prediction that the model makes on the unseen future prices.

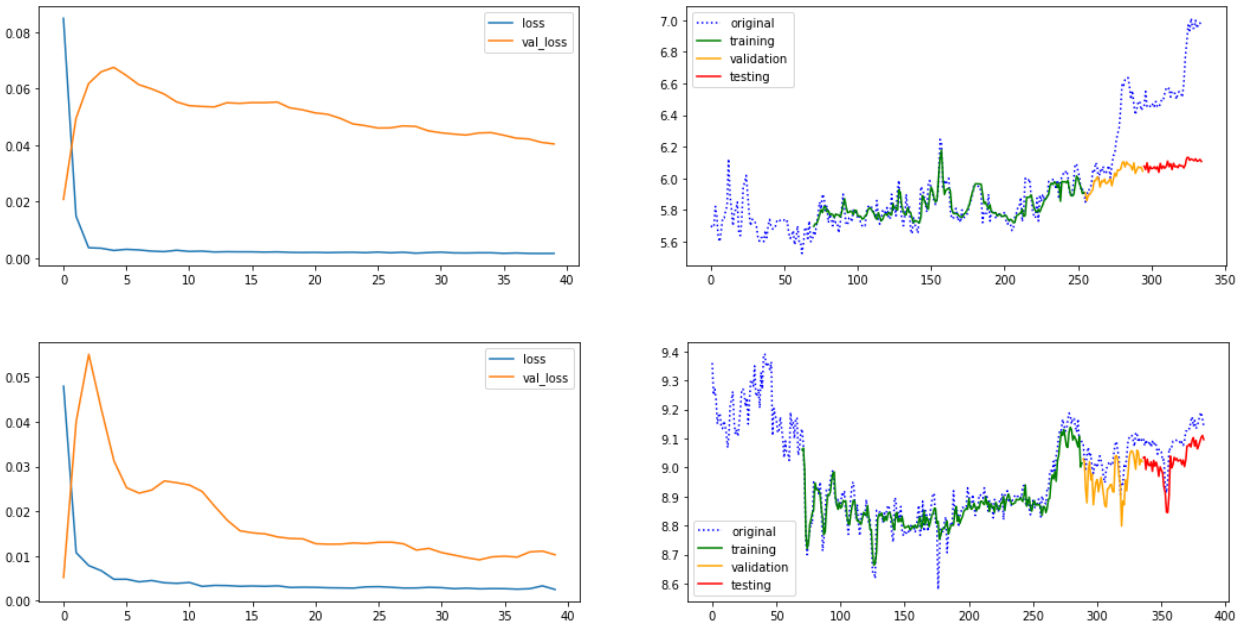


**Figure 3:** Each row/pair of graphs represent the loss and actual prediction made by training and using the model. The top row is an example of the model incorrectly predicting the stock (large offset from the actual data with mean-squared error > 4%), while the bottom row indicates the model tracking the movement of prices fairly well (mean-squared error < 1.5%).

While the MSE may appears as a useful metric for determining model confidence, we determined that there is very little correlation between validation MSE and testing MSE.

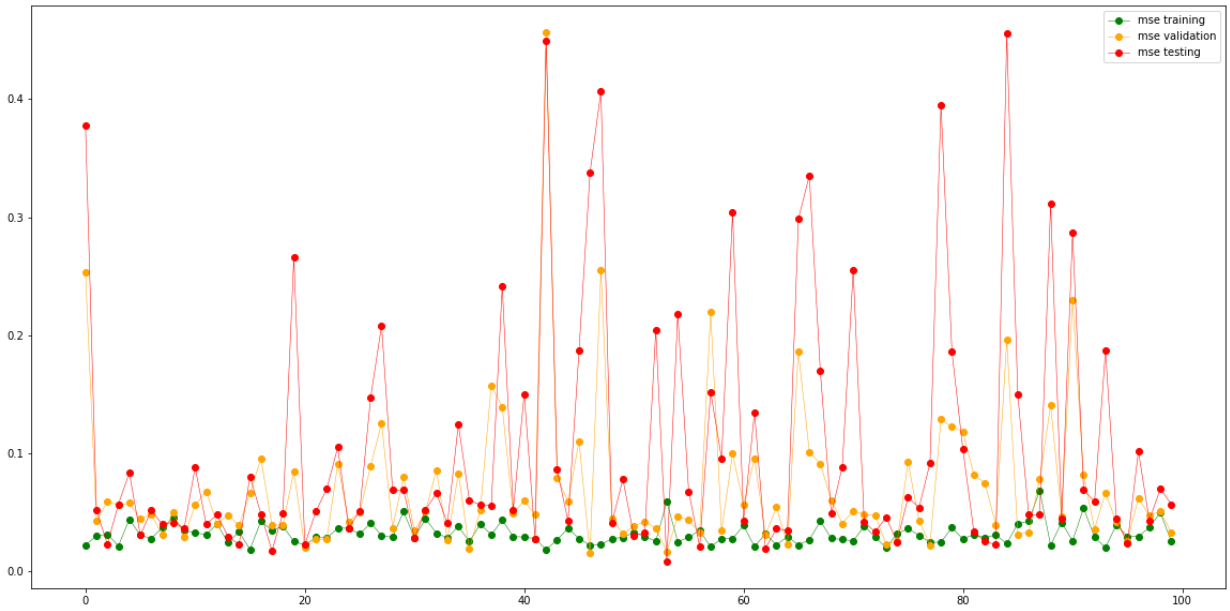


**Figure 4:** A graph of a subset of the MSEs recorded for each company stock's training, validation, and testing predictions. Primarily, it illustrates that the testing MSE is relatively large compared to the training and validation MSE; however, from empirical observations of the graphs (as in Figure 3), MSE < 1.5% appeared to provide a fairly good prediction.

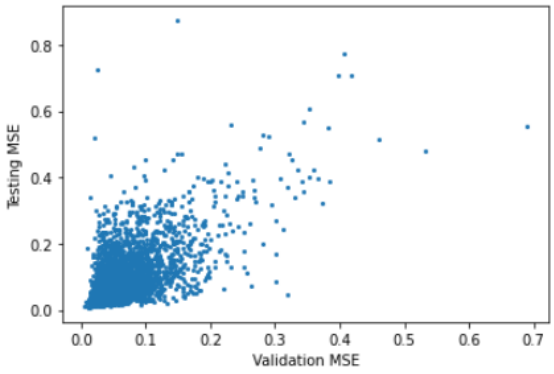


**Figure 5:** A plot of Testing vs Validation MSE to illustrate the low correlation between the two.

Using the combination of fundamental data stock selection and neural network prediction, we are able to achieve an average of 5% profit (~$5000) over 10 iterations of a portfolio size of 100.

# Conclusion and Improvements

Ultimately, the stock prediction model proved to be a challenging task. Many iterations are required to tune the neural network model, and it is difficult to balance the amount of time we had with the amount of data we needed to train on. In addition, the data inconsistency and sparsity in the fundamental data required a more generalized approach to predicting; even when

we determined that the company stock data performed well in one month, it was ambiguous whether the stock would continue improving or worsen the following month. If we had more time and data, more improvements to the neural network model (potentially reducing the number of layers) and fundamental data analysis would improve our performance in selecting stocks. In addition, allowing the optimal stopping to prioritize sells on the same day to minimize transaction fees would be a great improvement to our prediction model.

# References


[1] Olah, Christopher. “Understanding LSTMs.” (2015) https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[2] Goldberg, Yoav. "A Primer on Neural Network Models for Natural Language Processing." J. Artif. Intell. Res.(JAIR) 57 (2016): 345-420. https://scholar.google.com/scholar?cluster=3704132192758179278&hl=en&as_sdt=0,5.

[3] Bruss, Thomas F. “On the 1/e-strategy for the best-choice problem under no information.” In arXiv:2004.13749v2. (Rev 2020): https://arxiv.org/abs/2004.13749v2.