Time Series Prediction for Stock Price and Opioid Incident Location

by

Kevin Thomas

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved July 2019 by the
Graduate Supervisory Committee:

Arunabha Sen, Chair
Hasan Davulcu
Ayan Banerjee

ARIZONA STATE UNIVERSITY

August 2019

ABSTRACT

Time series forecasting is the prediction of future data after analyzing the past data for temporal trends. This work investigates two fields of time series forecasting in the form of Stock Data Prediction and the Opioid Incident Prediction. In this thesis, the Stock Data Prediction Problem investigates methods which could predict the trends in the NYSE and NASDAQ stock markets for ten different companies, nine of which are part of the Dow Jones Industrial Average (DJIA). A novel deep learning model which uses a Generative Adversarial Network (GAN) is used to predict future data and the results are compared with the existing regression techniques like Linear, Huber, and Ridge regression and neural network models such as Long-Short Term Memory (LSTMs) models.

In this thesis, the Opioid Incident Prediction Problem investigates methods which could predict the location of future opioid overdose incidences using the past opioid overdose incidences data. A similar deep learning model is used to predict the location of the future overdose incidences given the two datasets of the past incidences (Connecticut and Cincinnati Opioid incidence datasets) and compared with the existing neural network models such as Convolution LSTMs, Attention-based Convolution LSTMs, and Encoder-Decoder frameworks. Experimental results on the above-mentioned datasets for both the problems show the superiority of the proposed architectures over the standard statistical models.

DEDICATION

This work is dedicated to my family, thesis advisors, mentors, and friends.

ACKNOWLEDGMENTS

I would like to start by thanking my thesis advisor, Dr. Arunabha Sen for giving me the opportunity to work under him, for his guidance during the entirety of my thesis work which includes the intense technical discussions, the fundamental questions which made me see my problems or solutions from a different perspective and for the unconditional support. I would like to thank my thesis committee members Dr. Hasan Davulcu and Dr. Ayan Banerjee for their consent to be on my thesis committee and taking time out of their busy schedules to provide invaluable insights into my thesis. I am deeply indebted to my mentors Sandipan Choudhuri and Kaustav Basu for helping complete this thesis to fruition ever since I started working on my thesis in August 2018 and for all their suggestions with respect to the machine learning models that need to be used and for their tips in writing this thesis document. Lastly, I am grateful for having an extremely supportive family who supported me when I decided to pursue a thesis and my friends who were there during the tough and turbulent times, especially my roommates Aditya Chayapathy, Arpan Roy, Kausic Gunasekkar and Madhu Venkatesh, along with my friends Bosco Paul and John Santhosh, who all encouraged me and were there for me during the dark and tiring days.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Chapter 1

OVERVIEW

Time series data is a type of data indexed in time order, that is there is always some timestamp associated with every instance of data and if two instances of data are considered, one will always be in the past compared to the other. Time series forecasting is predicting the new values using a model trained on the old values in the time series (Time Series Wikipedia 2019). Time series data can be decomposed into four components which are (Jason Brownlee 2017):

1. Level: The average value in the time series.
2. Trend: The increasing or decreasing value in the time series.
3. Seasonality: The repeating short-term cycle in the time series.
4. Noise: The random variation in the time series.

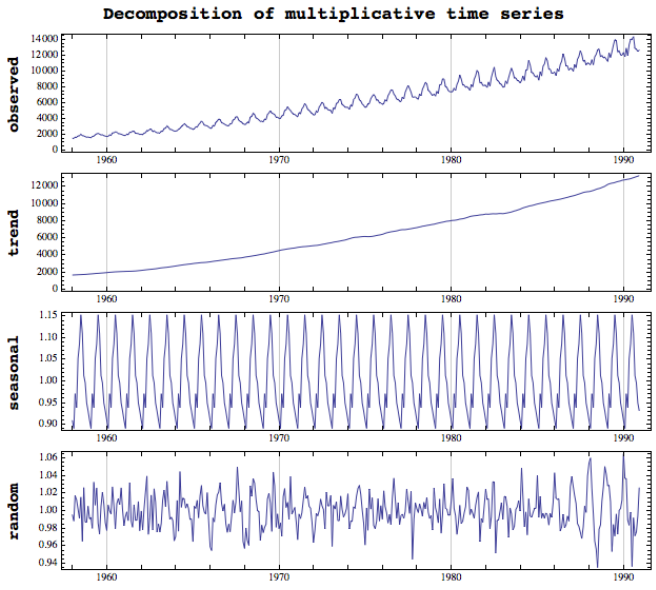

Figure 1. Time Series Decomposition (Mathematica Stack Exchange 2019)

Figure 1 shows the different components of time series data. *Observed* indicates the original observed data, *trend* represents the trend component, *seasonal* represents the seasonal component and *random* represents the noise and level components i.e. the remaining components after the trend and seasonal components are removed from the original time series data. By decomposing every time series into the four components, it becomes easier to analyze and forecast the data.

Time series analysis is the analysis of the relationship between each instance of data in the time series data. Studying this relationship may yield many interesting relations between the different time steps considered. Time Series Forecasting is the use of this analysis to predict the data for future time instances. This amounts to predicting the future and if the predictions are accurate every person with this knowledge will prepare accordingly. For example, analyzing the temperature and humidity of the current day can give us a fair idea of how the weather will be the next day. If people see that the prediction of tomorrow's weather is a thunderstorm then they will remember to bring an umbrella or plan our day accordingly.

The two types of time series forecasting this thesis is dealing with are:

1. Stock Price Prediction
2. Opioid Incident Location Prediction

The first part of the document discusses the Stock Data Prediction Problem. This problem tries to predict trends in the stock prices of ten companies such as Amazon, American Express, etc. found either in the NYSE or the NASDAQ stock market. Since the stock market data is a time series, this indicates that this data should have trend and seasonality

and by decomposing the ever-unpredictable stock market data into those two components then it is possible to predict the future prices. A novel deep learning model which uses a Generative Adversarial Network architecture is used and the results are compared with machine learning models involving Linear Regression, Huber Regression, Ridge Regression, Artificial Neural Networks, etc. Experimental results on the datasets of the ten companies for the proposed model versus the classic machine learning models shows the dominance of the proposed model.

The second part of the document explains the Opioid Incident Prediction problem. This problem is trying to predict the location of the next opioid incidences on the map. The input dataset used was of the Cincinnati and the Connecticut opioid incident dataset. Heat map images were used as input to the Generative Adversarial Network model to predict the next locations of the opioid overdose incidents. The supremacy of the proposed model is evident when comparing the experimental results on both the Cincinnati and the Connecticut datasets for the proposed model and the existing models.

Part I

Stock Price Prediction

Chapter 2

INTRODUCTION

The stock market has been around since the 17th century and it is one of the most mercurial entities in the world. The stock market is an aggregation of buyers and sellers of stock which represents ownership claims to various public companies or businesses (Stock Market Wikipedia 2019). Various people invest their money in companies they consider promising and get shares of the company in exchange. The share prices can either go up or down given time depending on the demand for that share. If people spend an amount more than the current share price to acquire the share, then the share price will increase. This generally implies the company is doing well and more people want to invest in the company as they believe that the share price will increase further. If people are selling their shares at a price lesser than the current share price, then the share price will drop. This generally implies that people do not believe that the company will perform well, and the share price will only fall further. If the share prices increase, the person can make a profit if he sells his shares of the company to someone willing to buy it at a higher price or hold onto the shares and wait for the price to appreciate. If the person is unlucky, the share prices might drop where he can either hold onto the stock expecting the prices to appreciate or sell the stocks expecting the price to depreciate further thereby making a loss but avoiding a bigger by selling the stocks at an even lower price.

The mercurial aspect of the stock market is the unpredictability of the stock price of the companies as these prices are solely decided by how much people are willing to pay to purchase a share of the company. For human beings to invest in a company by buying their stock, the company needs to have a good reputation among the general public and if there

is any change in the way people view the company it can affect the stock price positively or negatively. Let's call the event which brings about the change in the way people view the company and by extension a change in the stock price as trigger events. There are trigger events where we can expect a change in the stock price such as earnings calls, stock splits, mergers, and acquisitions, etc. But we have observed unexpected events trigger catastrophic changes in the stock prices of companies such as when Kylie Jenner tweeted *"Sooo does anyone else not open Snapchat anymore? Or is it just me... ugh this is so sad,"* on February 23rd, 2018 and the market value of Snap Inc. plummeted 6% which equated to a $1.3 billion loss (Kaya Yurieff 2018). In the first case, we can anticipate the occurrences of the trigger events but in the second case, we cannot anticipate the events beforehand.

A sudden change in the stock price can make people millionaires in an instant or can completely annihilate their life savings. Naturally, people will want to invest their hard-earned money into companies which certainly will increase in price and reward them for having faith in that company. Due to the general interest in making a profit in the stock market, researchers have been trying for decades to predict the movements in the stock market but have been largely unsuccessful as there seem to be many factors affecting it which most researchers have not taken into consideration. But with the advent of machine learning, researchers are having a lot more success in their predictions. With each new innovation in the machine learning world, we are getting closer to being able to accurately predict the trends in the stock market.

Chapter 3

RELATED WORK

Since the stock market involves a substantial amount of money flowing around and can make a person rich or poor in a fleeting amount of time, researchers have been trying to predict trends in the stock market using a machine learning since the early 1990s. There are various routes many researchers have chosen but all these routes can be classified under two main types of predictive analysis, which are:

1. Stock Price Prediction: Trying to predict the next time interval's stock price which effectively becomes a regression problem.
2. Stock Direction Prediction: Trying to predict the direction in which the stock price will move in the next time interval i.e. the price increases, decreases or stays the same which effectively becomes a classification problem.

Many scholars have attempted to use different regression techniques. The authors in (Roy et al. 2015) implemented a linear regression model where instead of employing the least square method they used Least Absolute Shrinkage and Selection Operator (LASSO) linear regression on the Goldman Sachs Group, Inc. (GS) on 3686 trading days (from May 4th, 1999 to January 3rd, 2014). The authors in (Henrique et al. 2018) used Support Vector Regression (SVR) on Brazilian, American and Chinese stocks for both daily and up-to-the-minute frequencies. The authors in (Gong et al. 2009) applied Logistic Regression on the three years (2005-2007) worth of stock data of the Shenzhen Development stock A (SDSA) from RESSAT Financial Research Database to predict the trends in next month's stock price according to the current month's stock price. The authors in (Khan et al. 2018) used

Robust Linear Regression on twenty years' worth of NASDAQ stock exchange, New York stock exchange (NYSE), London Stock Exchange (LSE), Karachi Stock Exchange data to predict the stock price.

Ever since online news and social media became popular and the companies started announcing important decisions on their websites or on their Twitter pages, people have tried to incorporate the online articles and social media messages as features in their prediction models. Many researchers have researched the effects of social media and online news articles on the stocks of companies. The authors in (Bollen et al. 2011) analyzed daily Twitter data to fetch the mood of the every message i.e. positive and negative moods measured in terms of 6 dimensions (Calm, Alert, Sure, Vital, Kind and Happy) to validate the effects it has on Dow Jones Industrial Average (DJIA). The authors in (Mao et al. 2012) incorporated tweets as an exogenous input to their linear regression predictive model to predict the S&P 500 closing prices. The authors in (Alostad et al. 2015) collected new articles about the companies in the DJIA to train their directional stock prediction system went on to prove that breaking tweet leads to a disruption in the direction of the stock price.

Since the popularity of artificial neural network skyrocketed, savants have tried to create neural network models to try predicting the trends in different stock markets. The authors in (Gurusen et al. 2011) published a paper about their analysis on how a multi-layered perceptron, a dynamic artificial neural network and a hybrid neural network which uses autoregressive conditional heteroskedasticity (GARCH) to extract new input variables by comparing them across their Mean Square Error (MSE) and Mean Absolute Deviate (MAD) of the NASDAQ stock prices. The introduction of a new recurrent neural network called Long-Short-Term-Memory (LSTM) caused a storm in this field as this particular

neural network could “remember” the past, be it long term or short term. LSTMs were great for predicting time series data as they could pick up obscure features from time series data using their memory and were able to give amazing results in their predictions. Without hesitation, the researchers created models using LSTMs to predict trends in the stock data. The authors in (Roondiwala et al. 2017) created a model to predict NIFTY 50 stock prices using a Sequential model with two LSTM layers and two Dense layers and used Root Mean Square Error (RMSE) as the error metric. The authors in (Tan et al. 2019) proposed a tensor-based event-LSTM which performed on an entire year’s worth of data of the China Securities markets to predict stock data by combining the fundamental features used in stock prediction and the news articles.

In 2014, Ian Goodfellow created the Generative Adversarial Network (GAN). The GAN uses a Generator-Discriminator model to train on the dataset. The inspiration to use a GAN in the model came from (Boris Banushev GAN model 2019), where a GAN with LSTMs as the Generator and Convolutional Neural Networks (CNNs) as the Discriminator on the Goldman Sachs stocks data was applied to a dataset with 112 features.

Chapter 4

DATASETS

The dataset for my research is the US Stock Market dataset to forecast the stock prices of 10 companies from different industry sectors, nine of which belong to the Dow Jones Industrial Average (DJIA) index. The ten companies with their stock tickers are as follows:

1. 3M (MMM)
2. Amazon (AMZN)
3. American Express (AXP)
4. Apple (AAPL)
5. Chevron (CVX)
6. Cisco Systems (CSCO)
7. J.P Morgan Chase (JPM)
8. Procter & Gamble (PG)
9. Verizon (VZ)
10. Walmart (WMT)

The data is fetched the site FirstRateData (FirstRateData 2019), which freely provides stock data of every minute starting from 9:30 am, July 6th, 2004 until 1:15 pm, March 28th, 2019. These constituted over 1.4 million stock datapoints. As training and testing on over 1.4 million data points would have been cumbersome, we considered the last 200,000 data points i.e. the minute data starting from 3:26 pm, September 6th, 2017 to 1:15 pm, March 28th, 2019. For each time period, the data collected by the API has the following features, for the selected time interval:

1. Open: the opening stock price of the minute
2. Close: the closing stock price of the minute
3. High: the highest stock price of the minute
4. Low: the lowest stock price of the minute
5. Volume: the number of shares traded in the minute
6. Number of trades: the number of trades occurred in the minute. One trade can have more than one share traded in it.
7. Weighted Average Price: the weighted average price of the stock in the minute

Since the objective is to forecast minute stock prices, two additional features were considered:

1. Volatility
2. Percentage Change.

Volatility is a metric which captures the stability of the stock price of a particular company. For instance, if the volatility is high, then the stock price is not stable and can be expected to fluctuate substantially. Lastly, the percentage change measures the change in closing and opening prices in that specific time interval. If the change is frequent and increasing, then it can be expected for the stock price to rise as the company is performing well. If the change is frequent and decreasing, then it can be expected for the stock price to fall, as the company is not meeting the mark in the eyes of its investors. Equations 1 and 2 denote how these metrics can be computed with the data obtained from our initial seven main features.

$$Volatility(V) = \frac{(High - Low)}{Low} X100 \qquad (1)$$

$$Percentage\ Change(PC) = \frac{(Close - Open)}{Open} X100 \qquad (2)$$

Using the libraries in Python, 11 new features were considered and those are as follows:

1. Fast Fourier Transforms (FFT) with frequencies 10, 20 and 50. We have considered FFT so that we can capture several long-term and short-term trends by using different frequencies. The higher the frequency the closer it mimics the real stock data. We calculate the FFT using the *fft()* method in *numpy*.
2. The Moving Average over the windows of sizes 480, 1440 and 3360. A window size of 480 indicates 480 minutes i.e. 8 hours or 1 trading day. Likewise, 1140 indicates a 3-day period and 3360 indicates a 7 day period. The moving average helps cut out the noise in the stock price data. Since the average is captured over a window, it captures the trend perfectly and removes the noise at the same time. We calculated the moving average by finding the mean of the result of the *rolling()* method in the *pandas* library on the closing prices column. The window size is passed as a parameter.
3. The Exponential Moving Average with Center of Mass 0.25 and 0.5. The exponential moving average is a type of moving average that places greater weight and significance on the more recent data points and this weight is called the center of mass. The exponential moving average was calculated by taking the mean of the result of the *ewm()* method of the *pandas* dataframe. The center of mass is passed as a parameter.
4. The Moving Average Convergence Divergence (MACD). The MACD is a trend following momentum indicator that shows the relation between two moving averages i.e. the 26-period moving average and the 12-period moving average.

Traders use the MACD to decide when to buy and sell stocks by comparing it with the 9-period Exponential Moving Average also known as the MACD Signal Line. The MACD is calculated by subtracting the 26-period exponential moving average from the 12-period exponential moving average. The window size is passed as a parameter to the exponential weighted function in *numpy* (ewm() function).

5. The upper and the lower Bollinger bands. The Bollinger bands are a technical tool used to detect if a particular stock is being overbought or oversold. If the price touches the upper band, it indicates that the stock is being overbought thereby triggering a sell signal. If the price touches the lower band, it indicates that the stock is being oversold thereby triggering a buy signal. We first calculated the 20-period moving average and the standard deviation of the 20-period moving average. The upper band was calculated by adding the 20-period moving average with the double of the standard deviation of the 20-period moving average. The lower band was calculated by subtracting double of the standard deviation of the 20-period moving average with the 20-period moving average.

The 20 features that were taken into consideration are as follows:

1. Open
2. Close
3. High
4. Low
5. Volume
6. Number of trades
7. Weighted Average Price

8. Volatility
9. Percentage Change
10. FFT with frequency 10
11. FFT with frequency 20
12. FFT with frequency 50
13. Moving Average of the 1-day-period
14. Moving Average of the 3-day-period
15. Moving Average of the 7-day period
16. MACD
17. Upper Bollinger Band
18. Lower Bollinger Band
19. Exponential Moving Average with Center of Mass as 0.5
20. Exponential Moving Average with Center of Mass as 0.25

I ran my model on the dataset containing these 20 features and the approach and results are mentioned below.

Chapter 5

APPROACH

In this section, the problem of predicting accurate closing prices for the minute data is formalized. A novel deep neural network model has been developed which predicts the closing price for a given time instance in the future (which is denoted by *look_forward*) having observed (or learned from) the past data. In the following section, the procedure for the prediction of the closing price for the *look_forward* time instance with the aforementioned 20 features has been discussed.

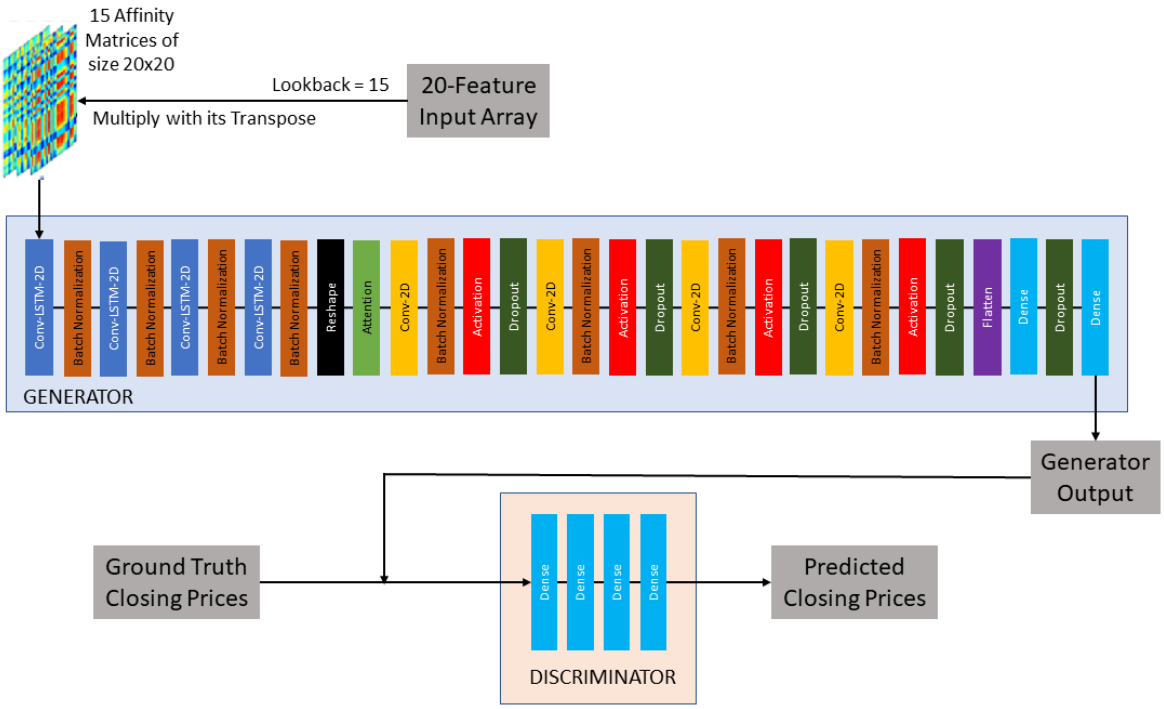

Figure 2: GAN model for Stock Data Prediction

Each company dataset is taken one by one as input. Min-max normalization is performed on the input. This input array is split into a training set and a testing set in the ratio 80:20. The first 80% of the day (in chronological order) is used to train the model with no. of epochs as 2000, a batch size of 4096 and a sample interval of 50.

The 20-feature input matrix is first multiplied with its transpose to create a 20x20 affinity matrix. The use of an affinity is well described by the authors of (Chuxu Zhang et al. 2018) who state that an affinity matrix captures the feature similarities and value scale correlations between two-time series and is robust to input noise at certain time series. This 20x20 affinity is fed as input to the Generator.

Figure 2 represents the structure of the model used to predict the closing stock prices. The model is inherently a Generative Adversarial Network (GAN). This GAN consists of a Generator and a Discriminator which trains each other to produce better results. To explain how a GAN works, consider a situation of cops trying to capture criminal having expertise in dealing with counterfeit notes. Consider the fraudsters to be the Generator and the cops to be Discriminator. The fraudsters create counterfeit money and start circulating them. The cops must learn to distinguish between fake notes and real notes. If the cops are having trouble distinguishing between the real and fake notes, then they must team up with the Bank (Ground Truth) and learn the subtle differences between them. Seeing that the cops have improved in identifying counterfeit notes then the fraudsters must up their game try producing better counterfeit notes, near indistinguishable from authentic currency. This is how a GAN works; the generator and discriminator are in this cat-and-mouse game which works on the principle of adversarial learning. The generator produces values which are then combined with ground truth values and the discriminator must try to discriminate

between the ground truth values and the generator values. If the discriminate cannot discriminate between the values, then the discriminator must reduce the value of the loss function through backpropagation and if it can, then the generator must use backpropagation to reduce its loss function. Both nets are trying to optimize a different and opposing loss function in a zero-sum game. A zero-sum game is the representation of a situation in which each participant's gain or loss of utility is exactly balanced by the losses or gains of the utility of the other participants i.e. the sum of the total gains of the participants and the negative of the total losses equate to zero (Zero-sum Game Wikipedia 2019). As the discriminator changes its behavior, so does the generator, and vice versa. Their losses push against each other (SkyMind AI 2019).

Here are a few terminologies used in the following section regarding the description of the different models:

1. Sigmoid Activation Function:

   The sigmoid activation function is given by the formula:

$$f(x) = \frac{1}{1+ e^{-x}} \qquad (3)$$

2. Long Short Term Memory Units (LSTMs):

   LSTMs are a type of Recurrent Neural Networks (RNN) that keeps track of Long Term memory and Short Term memory, unlike other neural networks that cannot "remember" input details. LSTMs take time and sequence into their account as they have a temporal dimension. This property of the LSTMs help in sentence translation, understanding videos and is effective in understanding time series data.

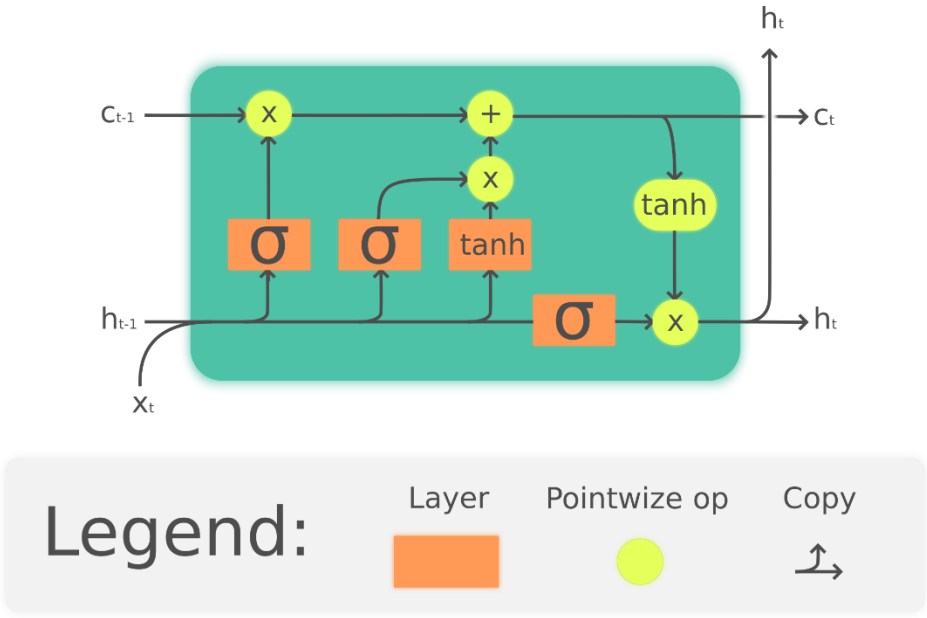

Figure 3: LSTM cell (LSTM cell Image Wikipedia 2019)

An LSTM is composed of a cell (the memory part of the LSTM unit) and three gates, which regulates the flow of information inside the LSTM unit: an input gate, an output gate and a forget gate. The cell keeps track of the dependencies between the elements in the input sequence. The input gate controls the extent to which a new value flows into the cell, the forget gate controls the extent to which a value remains in the cell and the output gate controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit. The activation function of the LSTM gates is the sigmoid function. There are connections into and out of the LSTM gates of which some are recurrent. The weights of these connections, which need to be learned during training, determine how the gates operate (LSTM Wikipedia 2019).

3. Convolutional Neural Networks (CNNs):

   CNNs are a type of neural network that takes an input dataset containing a matrix of values, applies a *kernel* or *filter* on the input matrix to capture the high-level features of the input dataset. This new matrix is then reshaped and fed to a neural network. The advantage of CNNs over other neural network is that is can easily capture the spatial relations between the input features because of the Convolution Phase. CNNs are extremely useful when the inputs are images as they can easily identify spatial relations in the images.

   The input parameters to a Convolutional Neural Networks are as follows:

   a. No. of units
   b. Filter Size
   c. Stride Length
   d. Type of Padding

   Stride is the length by which the filter shifts over the image after every iteration.

   In many cases the user wants the output of the CNN layer to have the same dimensions as the input. In such cases, a padding of zeroes can be added around the output of the CNN layer. This is done by specifying the type of padding during initialization, where *'same'* padding added a padding of zeroes around the output and *'valid'* padding does not add any padding around the output.

4. Convolutional LSTMs (ConvLSTMs):

   ConvLSTMs combines the concepts of Convolutions with LSTMs. The data is first scanned by the kernel or filter and this output matrix is reshaped and fed into LSTM layers. The advantage of using ConvLSTMs is that this neural network model

captures both the spatial (through the convolutions) and the temporal (through the LSTMs) relations between the different features (Xingjian et al. 2015).

5. Dropout:

   A dropout layer is generally added into between other neural network layers to avoid overfitting. Dropout is a regularization method that approximates training many neural networks with different architectures in parallel. During training, some number of layer outputs are randomly ignored or "dropped out". This has the effect of making the layer look-like and be treated like a layer with a different number of nodes and connectivity to the prior layer (Srivastava et al. 2014). This makes the training process noisy, forcing nodes within a layer to probabilistically take on more or less responsible for the inputs, thereby avoiding overfitting (Jason Brownlee 2018). A dropout rate is specified by the user to specify the probability each node has to be dropped out of the network while training.

6. Batch Normalization:

   Batch Normalization is a regularization technique used to improve the speed, performance, and stability of the neural network (Batch Normalization Wikipedia 2019). It forces the model to converge faster by normalizing the distribution of the inputs layers (Ioffe et al. 2015).

7. Attention:

   There may be cases that when predicting outputs using a neural network, some features will be more discriminating than the others. An attention layer will go through the input and output and assign weights to each feature thereby giving each feature a quantified value for the importance it has to the output.

8. *tanh* Activation function:

   The *tanh* activation function is given by the following formula:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \qquad (4)$$

9. Leaky ReLU Activation function:

   Leaky Rectifier Linear Unit (ReLU) Activation function is an activation function with the following formula:

$$f(x) = \begin{cases} \alpha x, & x < 0 \\ x, & x \geq 0 \end{cases} \qquad (5)$$

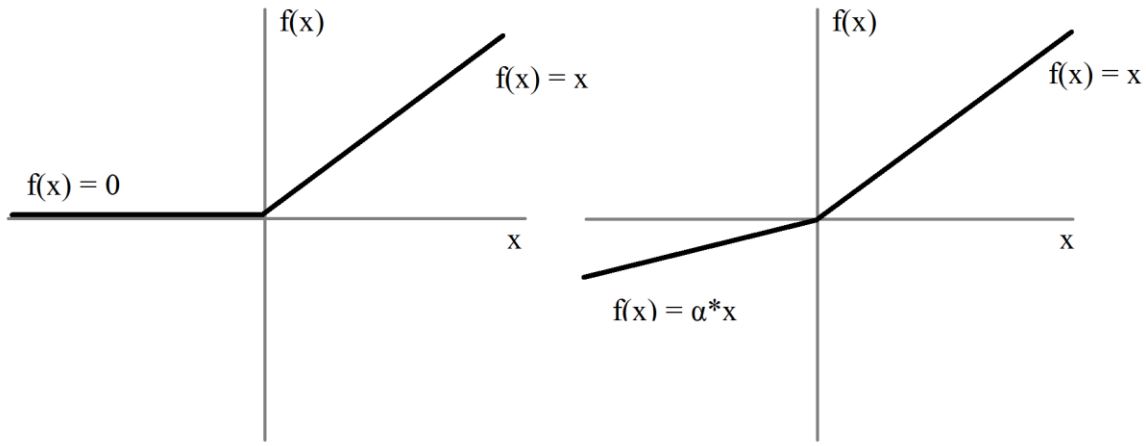

Figure 4: ReLU vs Leaky ReLU functions (Leaky ReLU Image 2018)

The value of α is generally specified by the user.

10. Mean Square Error (MSE) function:

    The MSE is a loss function given by the following formula (Isaac Changhau 2017):

$$L_{MSE} = \frac{1}{n}\sum_{i=0}^{n}(y_{true}^{i} - y_{pred}^{i})^{2} \qquad (6)$$

11. Root Mean Square Error (RMSE) function:

    The RMSE is a loss function which is the square root of the MSE. The formula is as follows (Isaac Changhau 2017):

$$L_{RMSE} = \sqrt{\frac{1}{n}\sum_{i=0}^{n}(y_{true}^{i} - y_{pred}^{i})^2} \qquad (7)$$

12. Mean Absolute Error (MAE) function:

    The MAE is a loss function given by the following formula (Isaac Changhau 2017):

$$L_{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_{true}^{i} - y_{pred}^{i}| \qquad (8)$$

13. Wasserstein Distance function:

    The Wasserstein distance is the adversarial loss function used by our GAN model. It is a distance function defined between probability distributions where if we consider each probability distribution as a pile of dirt then is the minimum "cost" of turning one pile into the other, which is assumed to be the amount of dirt that needs to be moved times the mean distance it has to be moved. Because of the above analogy, this distance is also called the Earth Mover's distance (Wasserstein distance Wikipedia 2019). The formula for Wasserstein loss is as follows, where y_{true} is the ground truth and y_{pred} is the predicted value:

$$L_{Wasserstein} = \frac{1}{n}(y_{true} * y_{pred}) \qquad (9)$$

14. Softmax Activation function:

    The *softmax* activation function is given by the formula:

$$f(x) = \frac{e^x}{\sum_{i=0}^{n} e^x} \qquad (10)$$

The structure of the Generator is as follows in the left to right order, with the input of step *n+1* is the output of step *n*, unless mentioned otherwise:

1. Two-Dimensional Convolutional LSTM Layer with 32 units, a 3x3 filter, a 1x1 stride and *'same'* padding.
2. Batch Normalization Layer
3. Two-Dimensional Convolutional LSTM Layer with 64 units, a 3x3 filter, a 1x1 stride and *'same'* padding.
4. Batch Normalization Layer
5. Two-Dimensional Convolutional LSTM Layer with 128 units, a 3x3 filter, a 1x1 stride and *'same'* padding.
6. Batch Normalization Layer
7. Two-Dimensional Convolutional LSTM Layer with 256 units, a 3x3 filter, a 1x1 stride and *'same'* padding.
8. Batch Normalization Layer
9. The outputs from 2, 4, 6 and 8 are reshaped and concatenated to form a stack on top of each other to form a matrix.
10. An attention layer is created by applying the mean function on the product of a Three-Dimensional Convolution with 1 unit, a 1x1x1 filter, a 1x1x1 stride, *'valid'* padding and *'softmax'* activation function. The input is the output from step 9.
11. Two-Dimensional Convolution Layer with 512 units, a 3x3 filter, a 2x2 stride, and *'valid'* padding.
12. Batch Normalization Layer
13. Leaky ReLU Activation Layer with $\alpha = 0.2$

14. Dropout with dropout rate = 0.3
15. Two-Dimensional Convolution Layer with 256 units, a 3x3 filter, a 2x2 stride, and *'valid'* padding.
16. Batch Normalization Layer
17. Leaky ReLU Activation Layer with $\alpha = 0.2$
18. Dropout with dropout rate = 0.3
19. Two-Dimensional Convolution Layer with 128 units, a 3x3 filter, a 2x2 stride, and *'valid'* padding.
20. Batch Normalization Layer
21. Leaky ReLU Activation Layer with $\alpha = 0.2$
22. Dropout with dropout rate = 0.3
23. Two-Dimensional Convolution Layer with 64 units, a 3x3 filter, a 2x2 stride, and *'valid'* padding.
24. Batch Normalization Layer
25. Leaky ReLU Activation Layer with $\alpha = 0.2$
26. Dropout with dropout rate = 0.3
27. Flatten
28. Dense Layer of 100 units and of Leaky ReLU activation with $\alpha = 0.2$
29. Dense Layer of 1 unit with *tanh* activation.

The outputs from the generator are mixed with the ground truth values and the Discriminator is trained using that. The structure of the Discriminator is as follows:

1. Dense Layer of 1 unit and of Leaky ReLU activation with $\alpha = 0.2$
2. Dense Layer of 1 unit and of Leaky ReLU activation with $\alpha = 0.2$

3. Dense Layer of 1 unit and of Leaky ReLU activation with $\alpha = 0.2$
4. Dense Layer of 1 unit and of *tanh* activation

This is the model structure and the inputs are trained and tested on this structure. The loss function used in the GAN model is a combination of the adversarial loss function called Wasserstein distance function and the Mean Square Error Loss and is given by the formula:

$$L_{GAN} = 0.6 * L_{Wasserstein} + 0.4 * L_{MSE} \qquad (11)$$

A combination of two loss functions is used because the Wasserstein loss calculates the distance between the two probability distributions but not how far apart the predicted values are. Hence MSE is also incorporated in the loss to account for how far apart the predicted values are.

Every hyperparameter in this structure has been selected after running the model in multiple iterations with different sets of hyperparameters. The above final set of hyperparameters is the set that produces the best results.

Chapter 6

EXPERIMENTAL RESULTS

The schematic diagram is given in Figure 2. Root-mean-squared (RMSE) and mean-absolute errors (MAE) are used as evaluation metrics for comparing the above-mentioned techniques with our proposed model. The results are divided into the following predictions:

1. Prediction for the next minute (1-time instance later)
2. Prediction for the third day (1000-time instances later)
3. Prediction for the seventh day (3000-time instances later)

The results of the GAN models were compared against the classic Linear regression technique, the classic Huber regression technique, the classic Ridge regression technique and a classic LSTM model. Here are the explanations for the regression techniques used:

1. Linear Regression:

   Linear regression is a regression technique which models the relationship between a dependent variable and one or more independent variables using a linear approach i.e. one tries to fit a straight line with minimum distance between all the points formed by the dependent and the independent variables. Therefore, one can use the equation of this line to predict new values of the dependent variables given the independent variables. In the case of the Stock Data Prediction Problem, we consider the closing price to be the dependent variables and the other features to be the dependent variable, thereby trying to fit a line to be able to predict future closing price values. The loss function used for Linear Regression is the Mean Square Error.

2. Huber Regression:

   Huber regression is a type of Robust Regression. Robust Regression is a regression technique which is insensitive to outliers and can identify outliers and fit a line across all the points created by the dependent and independent variables with least error. This is done by having by either setting a threshold δ in the loss function where if it crosses the set threshold then the effect on the overall fitting of the line is reduced. Huber regression uses Huber loss as its loss function which is given by the following formula:

$$L_{Huber} = \begin{cases} \frac{1}{2}(y_{true} - y_{pred})^2, & |y_{true} - y_{pred}| \leq \delta \\ \delta|y_{true} - y_{pred}| - \frac{1}{2}\delta^2, & |y_{true} - y_{pred}| > \delta \end{cases} \qquad (12)$$

3. Ridge Regression:

   Ridge Regression is a regression technique that assumes that the data suffers from multicollinearity i.e. existence of near-linear relationships among the independent variables. When multicollinearity occurs, least squares estimates are unbiased, but their variances are large so they may be far from the true value. By adding a degree of bias to the regression estimates, ridge regression reduces the standard errors thereby hoping that the net effect will be to give estimates that are more reliable (NCSS 2019). The loss function is as follows where λ is set by the user:

$$L_{Ridge} = L_{MSE} + \lambda * (slope)^2 \qquad (13)$$

The comparative study results are presented in Table 1 to Table 6. There are 6 tables, 2 for each prediction group. Each prediction group will have a table depicting RMSE values and a table depicting MAE values.

Method	AAPL	AMZN	AXP	CSCO	CVX	JPM	MMM	PG	VZ	WMT
Linear Regression	0.172	1.496	0.052	0.027	0.109	0.064	0.138	0.078	0.036	0.057
Huber Regression	0.419	1.814	0.085	0.056	0.135	0.112	0.189	0.093	0.048	0.073
Ridge Regression	0.172	1.496	0.052	0.027	0.109	0.064	0.138	0.078	0.036	0.057
LSTM	11.032	463.160	1.118	0.899	6.991	3.653	16.188	4.952	3.750	11.552
GAN Model	**0.052**	**3.386**	**0.087**	**0.176**	**0.045**	**0.024**	**0.044**	**0.223**	**0.031**	**0.031**

Table 1: RMSE of all models for all companies for next minute prediction

Method	AAPL	AMZN	AXP	CSCO	CVX	JPM	MMM	PG	VZ	WMT
Linear Regression	0.051	0.601	0.028	0.014	0.035	0.029	0.073	0.020	0.014	0.023
Huber Regression	0.254	0.846	0.058	0.034	0.057	0.075	0.112	0.031	0.025	0.036
Ridge Regression	0.051	0.601	0.028	0.014	0.035	0.029	0.072	0.020	0.014	0.023
LSTM	8.157	364.526	0.731	0.588	5.668	2.821	12.359	3.865	3.050	9.723
GAN Model	**0.032**	**1.613**	**0.060**	**0.042**	**0.026**	**0.020**	**0.030**	**0.155**	**0.021**	**0.024**

Table 2: MAE of all models for all companies for next minute prediction

Method	AAPL	AMZN	AXP	CSCO	CVX	JPM	MMM	PG	VZ	WMT
Linear Regression	3.078	42.140	1.364	0.894	1.696	1.207	3.203	1.180	0.771	1.145
Huber Regression	3.592	39.555	1.258	0.991	1.691	1.367	3.045	1.273	0.885	1.125
Ridge Regression	3.078	42.140	1.364	0.894	1.694	1.207	3.203	1.180	0.771	1.145
LSTM	29.890	445.790	2.306	1.444	6.887	2.085	15.927	5.154	3.757	11.518
GAN Model	**0.513**	**8.614**	**0.250**	**0.191**	**0.272**	**0.188**	**0.507**	**0.247**	**0.127**	**0.176**

Table 3: RMSE of all models for all companies for the third-day prediction

Method	AAPL	AMZN	AXP	CSCO	CVX	JPM	MMM	PG	VZ	WMT
Linear Regression	2.263	28.640	0.965	0.575	1.162	0.886	2.554	0.840	0.583	0.833
Huber Regression	2.440	26.981	0.898	0.641	1.151	1.045	2.435	0.892	0.615	0.798
Ridge Regression	2.263	28.640	0.965	0.575	1.161	0.886	2.555	0.840	0.583	0.833
LSTM	23.798	357.325	1.802	1.112	5.496	1.534	12.257	3.971	3.076	10.117
GAN Model	**0.371**	**5.733**	**0.170**	**0.090**	**0.194**	**0.143**	**0.405**	**0.160**	**0.091**	**0.129**

Table 4: MAE of all models for all companies for the seventh-day prediction

Method	AAPL	AMZN	AXP	CSCO	CVX	JPM	MMM	PG	VZ	WMT
Linear Regression	5.900	61.301	2.295	1.395	2.341	1.854	5.601	1.758	1.255	1.563
Huber Regression	6.728	56.044	2.041	1.203	2.414	1.986	5.278	1.854	1.265	1.687
Ridge Regression	5.900	61.301	2.294	1.400	2.342	1.854	5.577	1.757	1.255	1.562
LSTM	10.757	440.954	11.210	2.048	6.886	12.901	15.728	5.255	3.823	11.540
GAN Model	**1.140**	**12.033**	**0.457**	**0.192**	**0.443**	**0.405**	**0.795**	**0.341**	**0.233**	**0.282**

Table 5: RMSE of all models for all companies for the seventh-day prediction

Method	AAPL	AMZN	AXP	CSCO	CVX	JPM	MMM	PG	VZ	WMT
Linear Regression	4.310	49.886	1.887	1.060	1.763	1.445	4.484	1.282	0.939	1.181
Huber Regression	4.239	45.912	1.672	0.922	1.731	1.515	4.129	1.301	0.905	1.208
Ridge Regression	4.310	49.886	1.887	1.060	1.764	1.445	4.472	1.281	0.940	1.179
LSTM	7.748	353.705	9.985	1.526	5.566	11.463	12.240	4.052	3.109	10.184
GAN Model	**0.895**	**9.473**	**0.361**	**0.148**	**0.356**	**0.312**	**0.615**	**0.244**	**0.176**	**0.212**

Table 6: MAE of all models for all companies for the seventh-day prediction

Figure 5 to Figure 21 show all the closing price vs time graphs containing the ground truth plots in blue and the plot of the predicted values in orange. These figures contain the plots for predictions of the closing price for the next minute, 1000 minutes later and 3000 minutes later from the current minute.

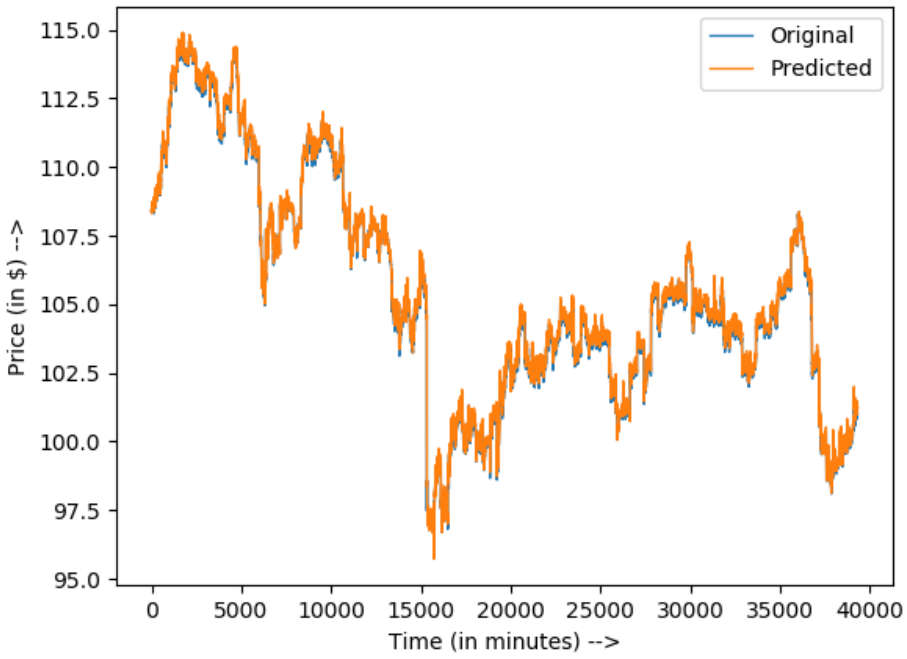

Figure 5: Closing Price Prediction for the next minute for JPM

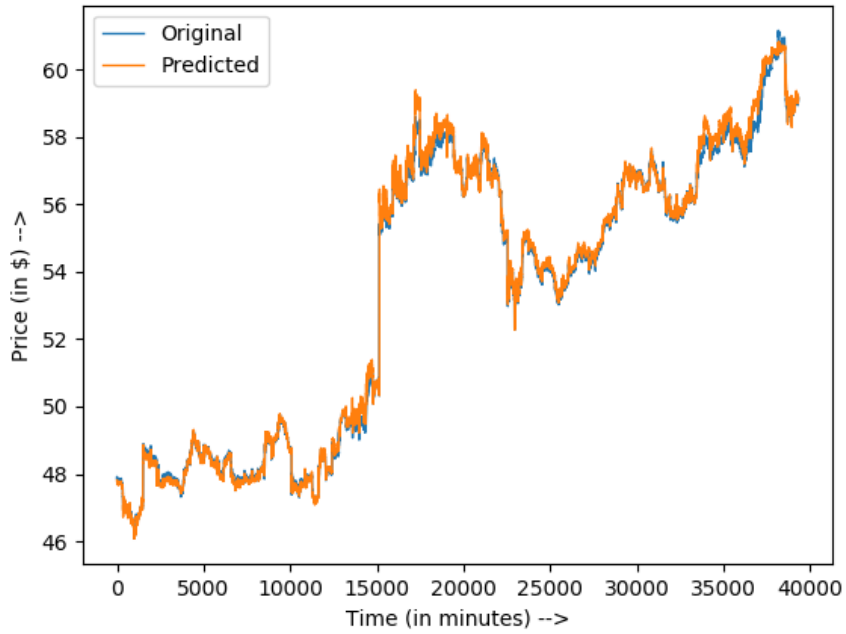

Figure 6: Closing Price Prediction for the next minute for VZ

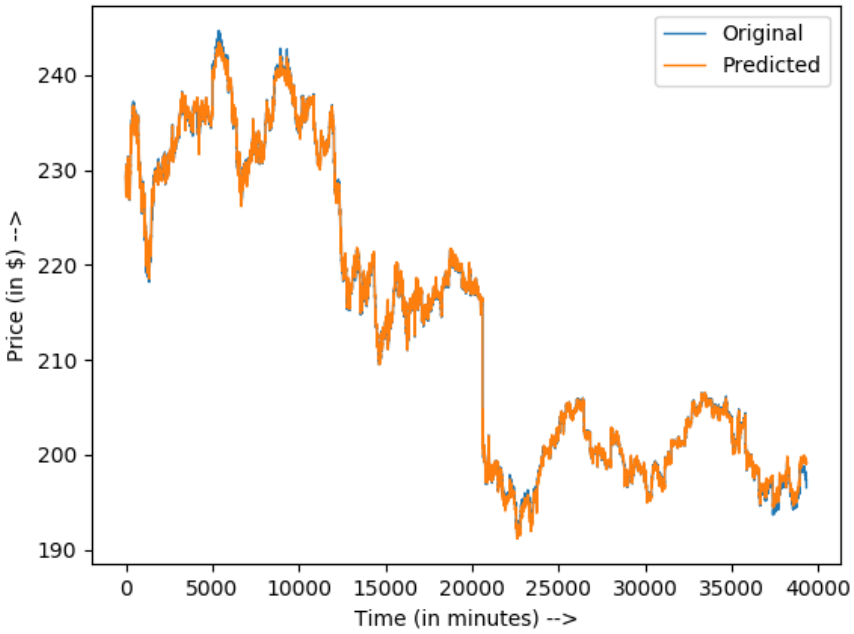

Figure 7: Closing Price Prediction for the next minute for MMM

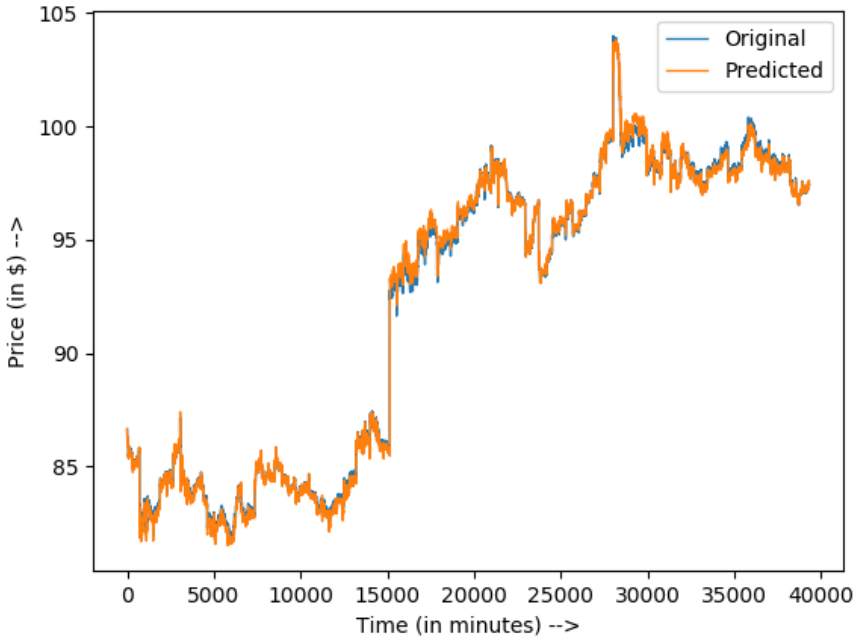

Figure 9: Closing Price Prediction for the next minute for WMT

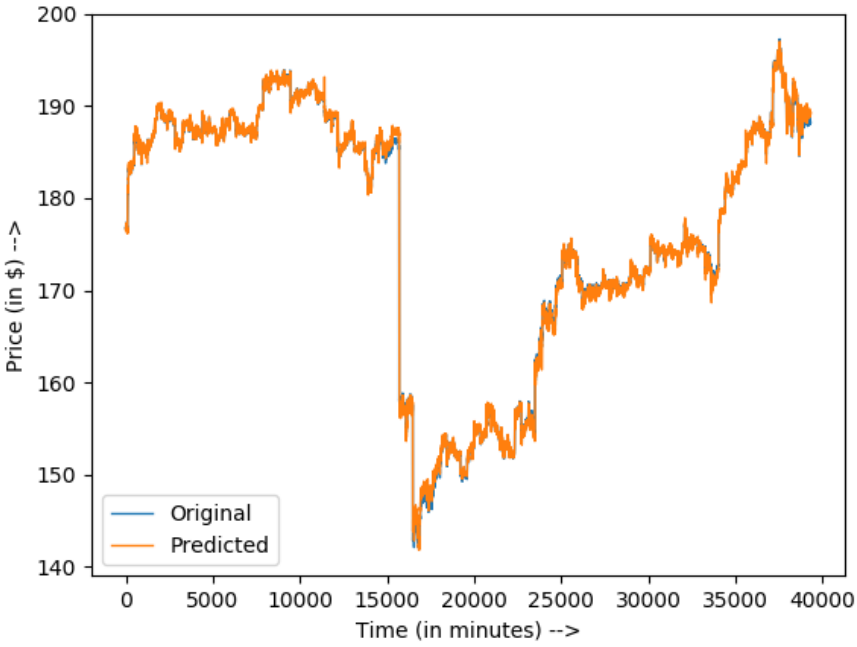

Figure 9: Closing Price Prediction for the next minute for AAPL

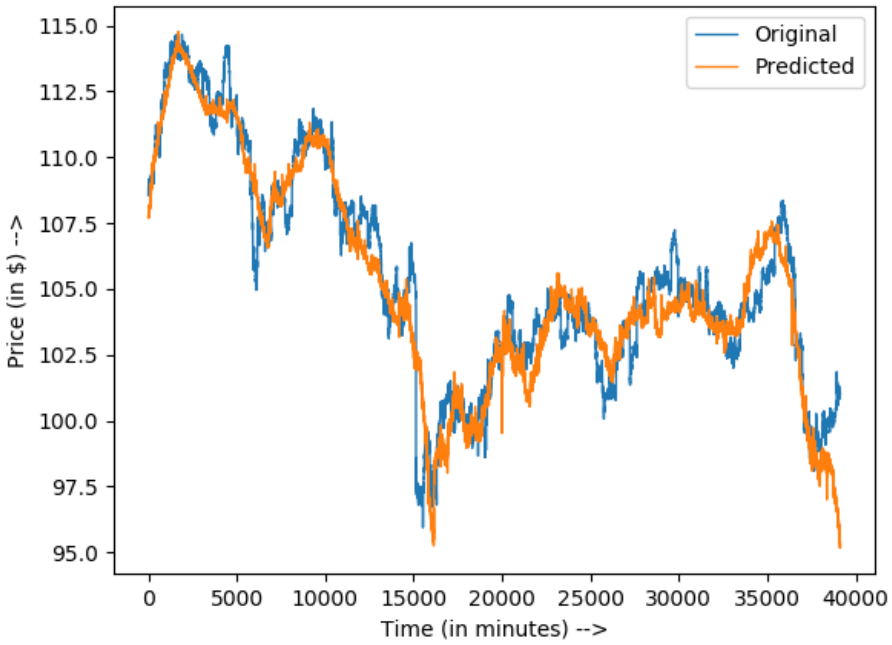

Figure 10: Closing Price Prediction for 1000 minutes later for JPM

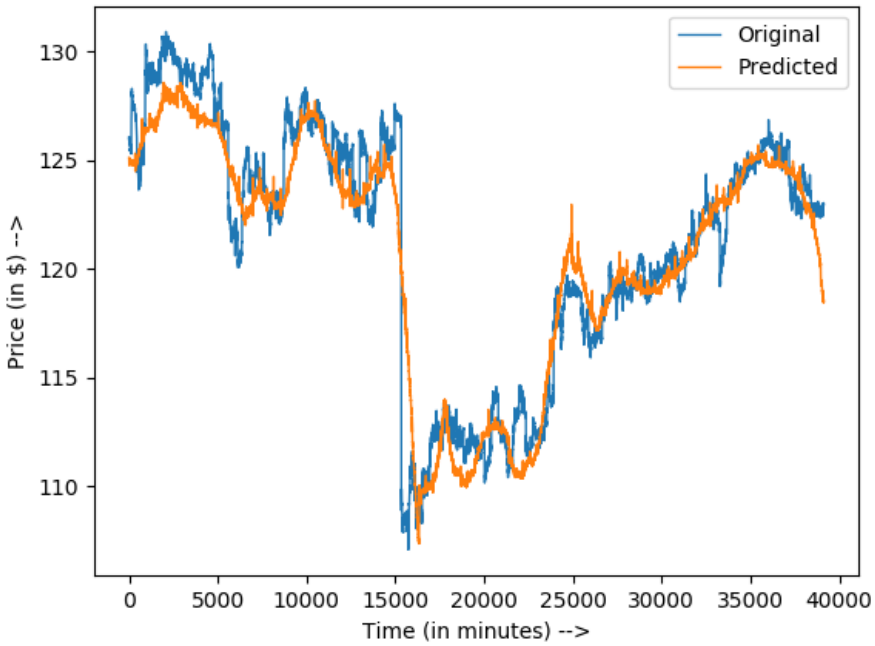

Figure 11: Closing Price Prediction for 1000 minutes later for CVX

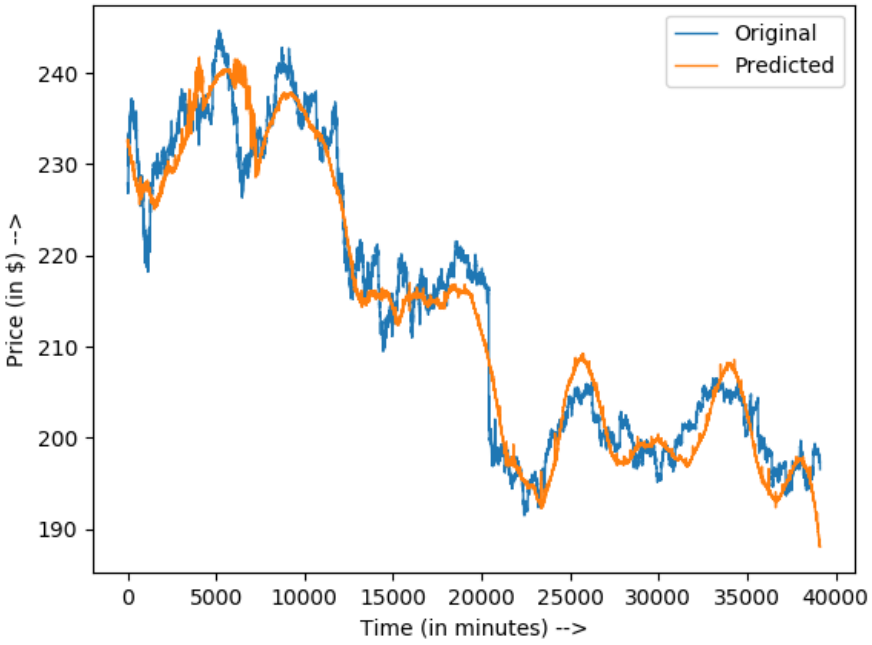

Figure 12: Closing Price Prediction for 1000 minutes later for MMM

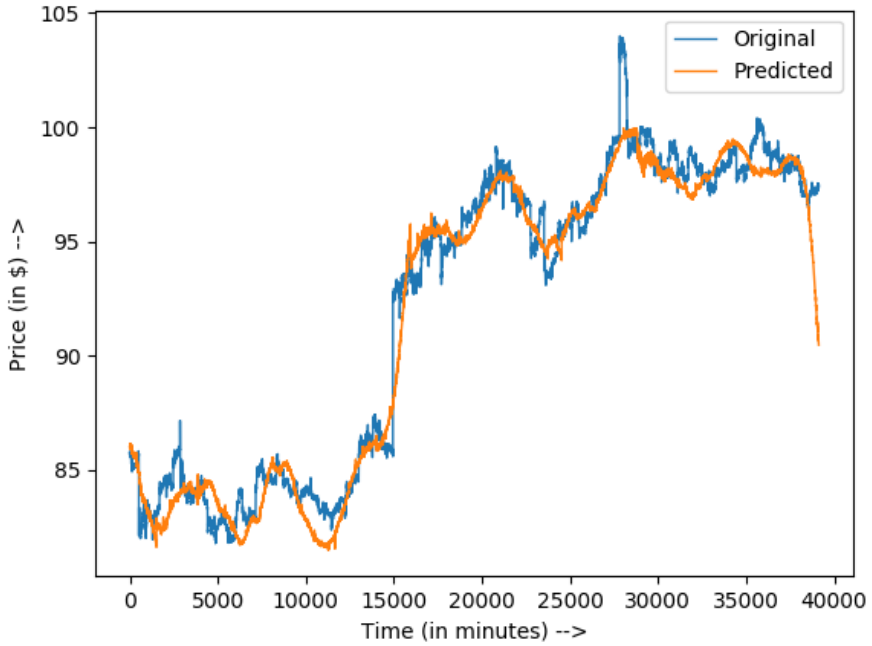

Figure 13: Closing Price Prediction for 1000 minutes later for WMT

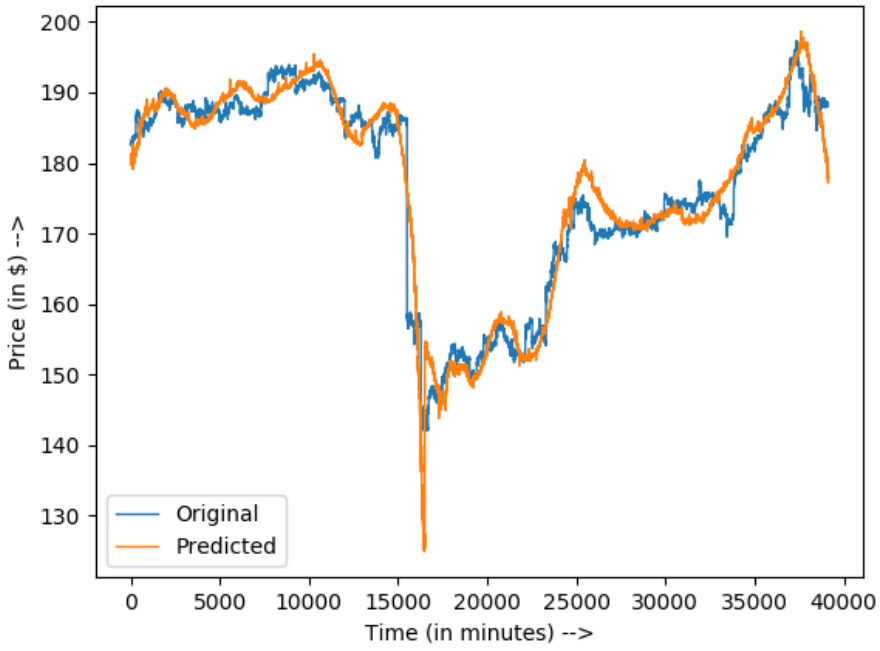

Figure 14: Closing Price Prediction for 1000 minutes later for AAPL

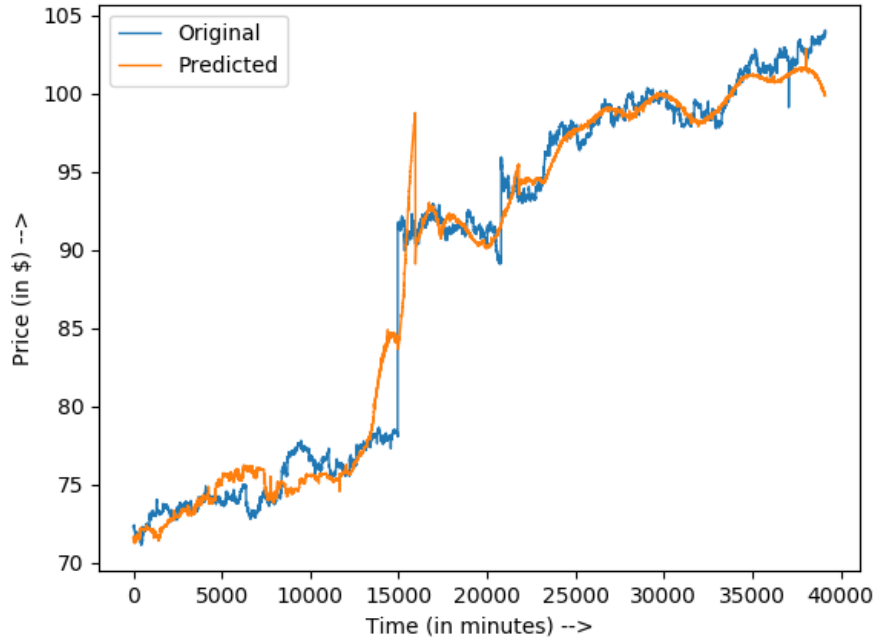

Figure 15: Closing Price Prediction for 1000 minutes later for PG

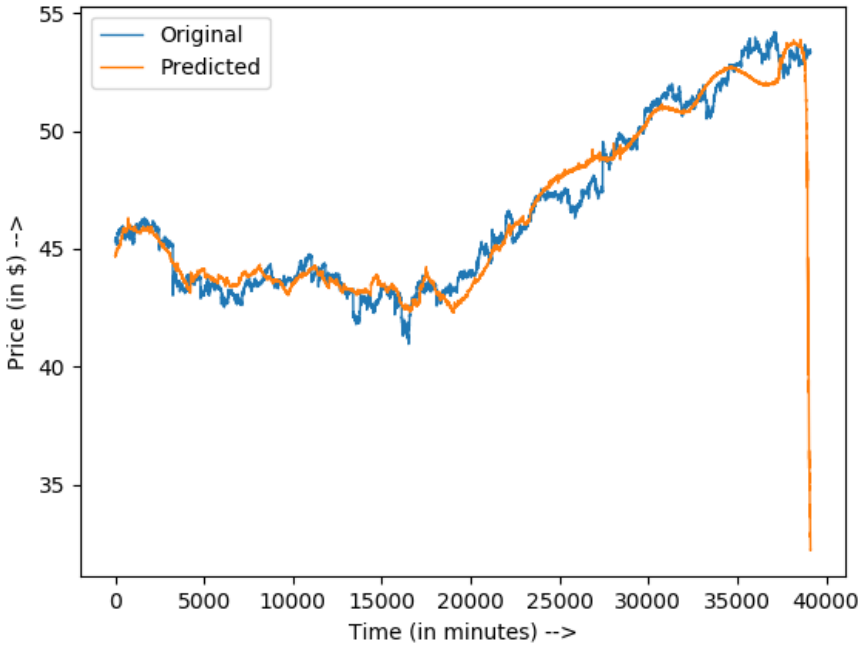

Figure 16: Closing Price Prediction for 1000 minutes later for CSCO

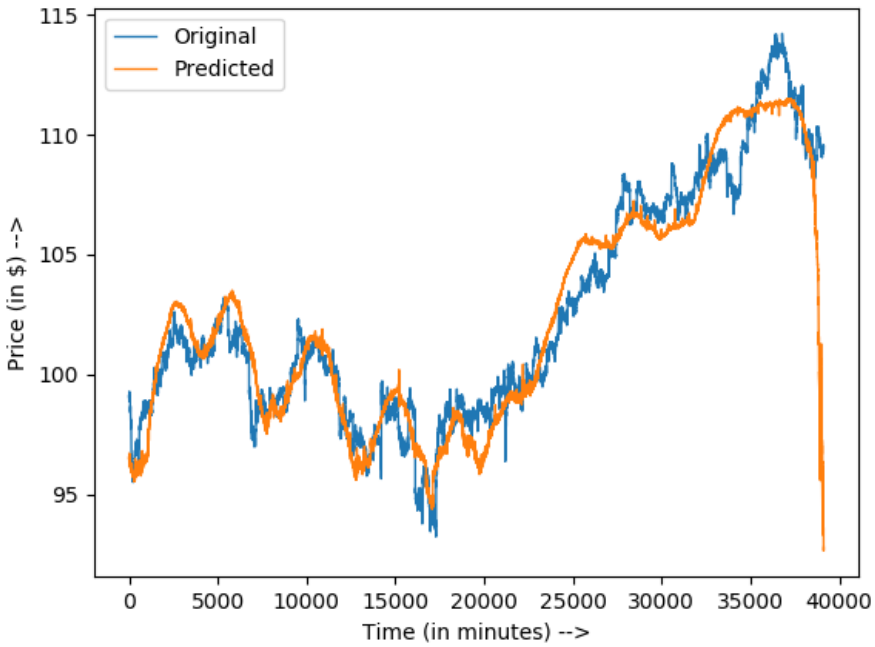

Figure 17: Closing Price Prediction for 1000 minutes later for AMZN

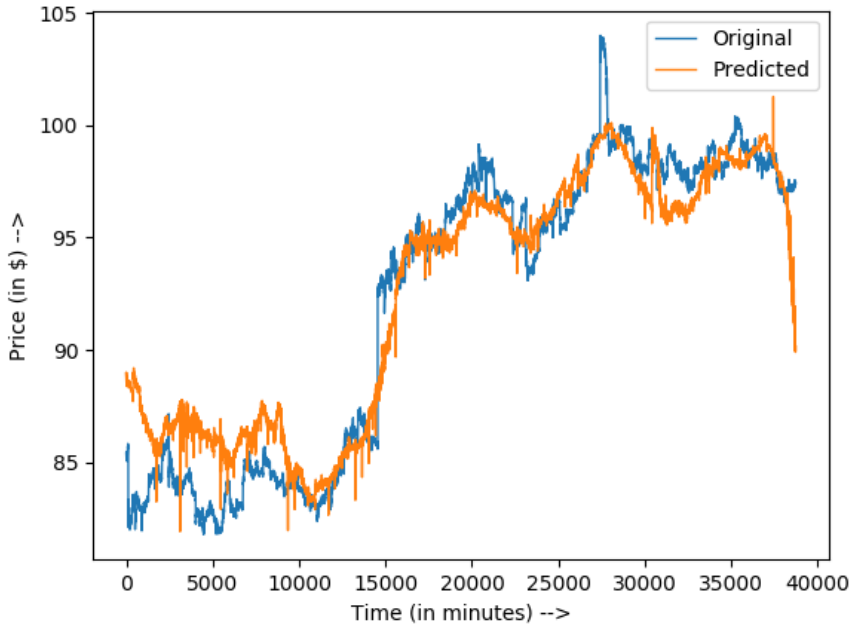

Figure 18: Closing Price Prediction for 3000 minutes later for WMT

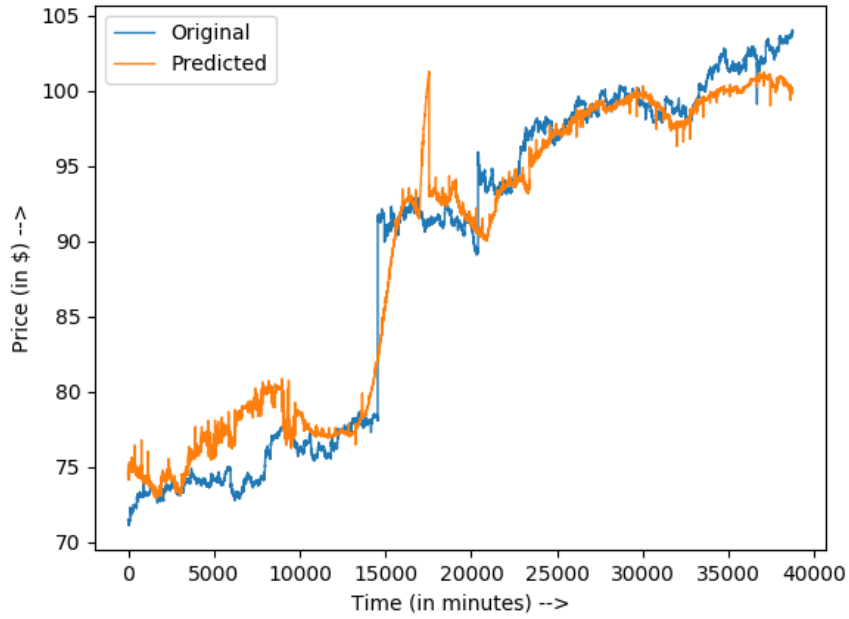

Figure 19: Closing Price Prediction for 3000 minutes later for PG

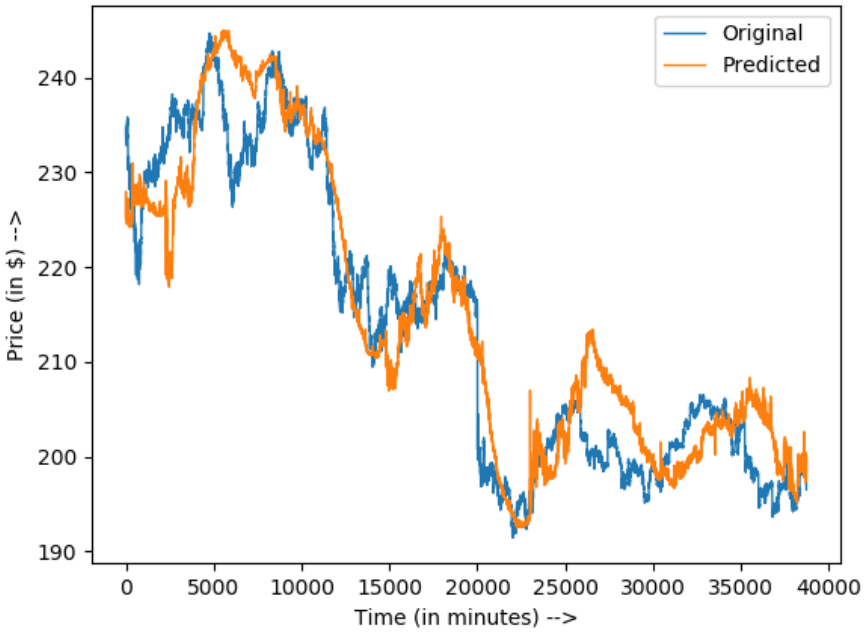

Figure 20: Closing Price Prediction for 3000 minutes later for MMM

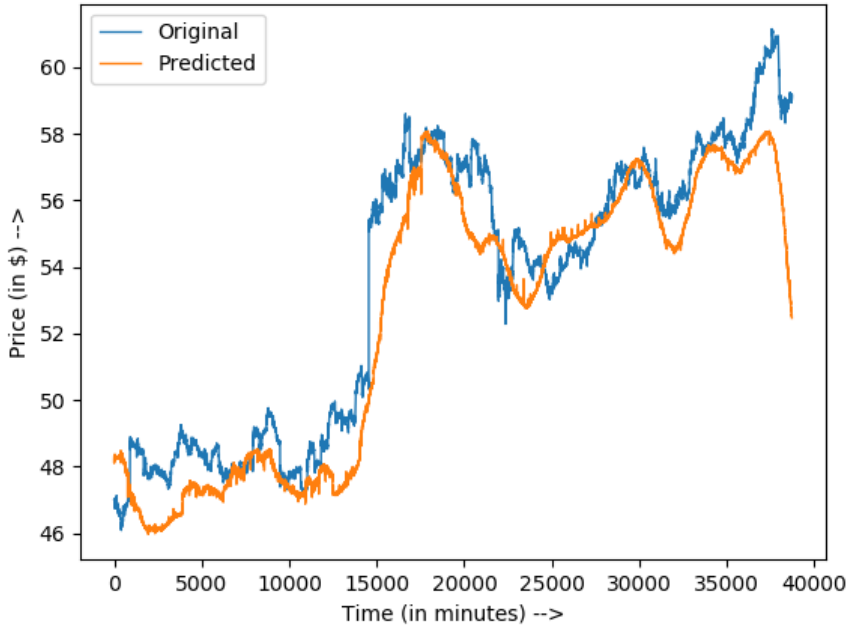

Figure 21: Closing Price Prediction for 3000 minutes later for CVX

Table 7 shows the average model training time and the average time taken to test one sample for the model for Linear regression, Huber regression, Ridge regression, LSTM and the Stock GAN model. The average time in both cases were calculated by measuring the time taken for all ten companies for all the three prediction types (next minute, 1000 minutes later and 3000 minutes later) and averaging them out. The training time was calculated by measuring the time taken for the model to train on the training set. The time taken to test 1 sample was calculated by dividing the time taken to test for the entire test set by the number of samples in the test set.

Method	**Training Time**	**Time for testing 1 sample**
Linear Regression	0.2740 s	1.0548×10^{-7} s
Huber Regression	15.1724 s	1.2273×10^{-7} s
Ridge Regression	0.1445 s	1.0637×10^{-7} s
LSTM	56 min	0.1 s
GAN	**2 hrs 34 min**	**0.3 s**

Table 7: Time taken for Stock Price Prediction Models

The deep learning models were trained on an 8-core processor with Tesla V-100 (16GB variant) GPU for 4000 epochs with a batch size of 1024, whereas the regression models were trained on a Dell Inspiron 7559 with 16 GB RAM.

Part II

Opioid Incident Location Prediction

Chapter 7

INTRODUCTION

Opioids are highly addictive drugs often prescribed by doctors to be used as painkillers. According to the statistics provided by the Centers for Disease Control and Prevention (CDC), for the year 2017 around 68% of the more than 70,200 drug overdose deaths in 2017 involved an opioid, and on an average, 130 Americans die every day from an opioid overdose (CDC 2017). This number has increased by 12.5% as compared to the year 2016, which led the US government to declare this epidemic as a public health emergency in October 2017. Blue Cross Blue Shield stated in their 2017 report (BCBS 2017) that 21% of their commercially insured members filled at least one Opioid prescription in 2015. Their data show that members with an Opioid Use Disorder (OUD) diagnosis grew to 493% from 2010 to 2016.

The impact of the opioid epidemic is becoming progressively worse despite all the efforts of the government and the governmental agencies involved. Researchers in the medical and analytical domains are inspecting methods where meticulous analysis of relevant data, may provide some useful insights into the epidemic where relevant data may comprise of prescription patterns of health care professionals such as doctors, dentists, nurses, etc. Predictive analytics of opioid consumption patterns of patients, time and locations of opioid-related incidences, etc. can play an important role in combating the opioid epidemic by providing decision-making tools to stakeholders at various levels ranging from the health care professionals to the policymakers to the first responders. The insights obtained after analysis of such data can be taken into consideration while formulating response at multiple levels. Although a few health insurance companies and data analytics firms have

examined this important issue, analytical research findings from the analysis of publicly available Opioid data are sparse.

Heat maps (hot spots) of opioid incidences are created by the government and NGOs to visualize the impact of the Opioid epidemic. Oftentimes, these maps are created using past data of overdose cases. Opioid incidence heat maps generated using the past opioid incidents data would be beneficial in aiding these aforementioned stakeholders to visualize the profound impact of the epidemic, but such heat maps created using the past data only help in providing retrospective information and may not be useful for preventive action in the current times or the foreseeable future. An example of an opioid incident heat map is shown in Figure 22.

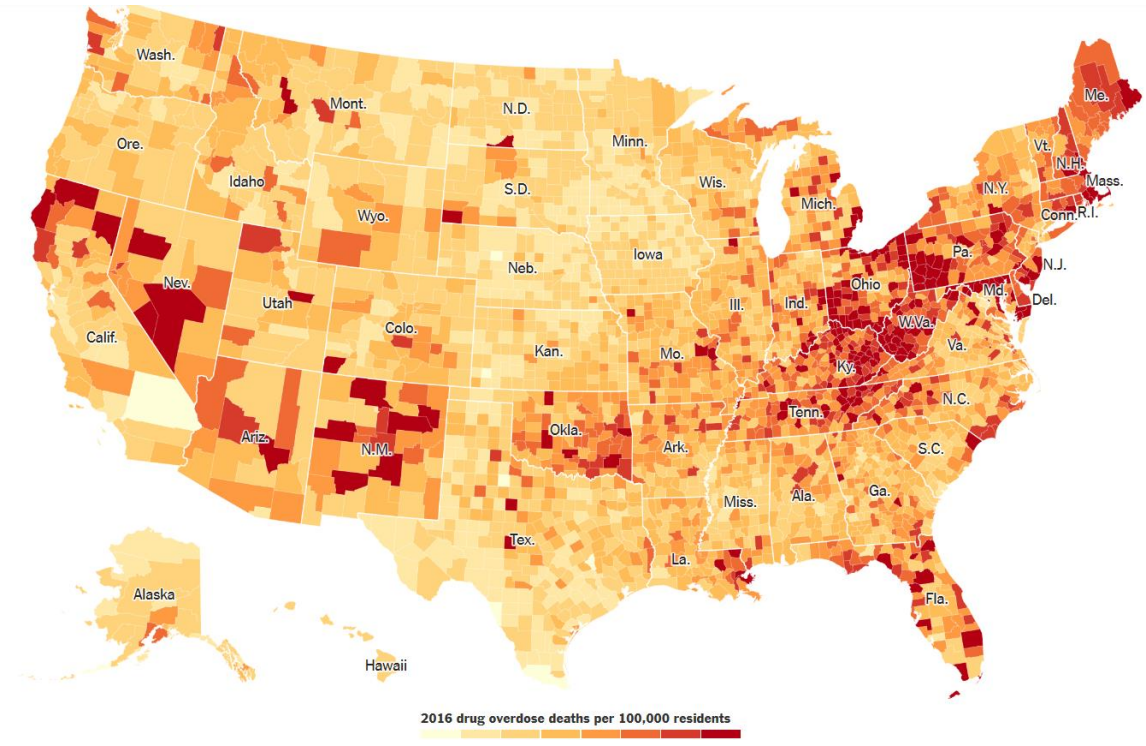

Figure 22: Opioid Overdose Heatmap of the US (Katz and Goodnough 2017)

Policy makers, law enforcement agencies, etc. analyze these heat maps to gain insights into the spread of the epidemic, over a geographical area. Resource allocation decisions, such as the establishment of new Medication Assisted Treatment (MAT) centers, stocking up on Naloxone doses, organizing rehabilitation programs, etc. are often based on the analysis of Opioid incidence hot spots, obtained from previous data. As mentioned before, the post-fact generation of heat maps provides respective authorities with only retrospective information. It may not be as useful for preventive action, in the current or subsequent timeframe. It will be of benefit to these professionals, if they are provided with the analytical tools to predict the heat map for the upcoming timeframe (week, month, year, etc.), by analyzing historical heat maps.

In the second part of this thesis, a novel deep neural network architecture is presented, which learns the subtle spatiotemporal variations in Opioid incidences data and accurately predicts future heat maps. The two datasets considered for evaluating the efficacy of the model are:

1. The Cincinnati Heroin Overdose Dataset
2. The Connecticut Drug-Related Death Dataset.

Chapter 8

RELATED WORK

The medical domain has been meticulously studying the effect of opioids on human beings for some time. The effectiveness and the harms of long-term opioid therapy for chronic pains in adults were studied by the authors of (Chou et al. 2015) with evaluated evidence on it. The authors of (Bohnert et al. 2011) studied the correlation of the maximum prescribed daily opioid dose and dosing schedules with the risk of opioid overdose death amongst patients with cancer, chronic pain, acute pain, and substance use disorder. The authors of (Cicero et al. 2017) performed a systematic literature review, using a qualitative approach to examine the development of an Opioid-use disorder from the point of initial exposure.

In the analytics and machine learning domain, a substantial amount of work has been going on with respect to the opioid problem. The authors of (Mackey et al. 2017) studied illegal sales of prescription opioids on Twitter. The authors of (Rice et al. 2012) developed a model to identify patients at risk for prescription opioid abuse, using drug claims data. The use of machine learning techniques for surveillance of drug overdose was studied by the authors of (Neill et al. 2018). The application of deep neural networks, such as recurrent neural networks, for classifying patients on opioid use was illustrated by the authors of (Che et al. 2017). The authors of (Acion et al. 2017) highlighted the use of machine learning and deep learning for predicting substance use disorder treatment success. Even the data science researchers from IBM Research and experts at Watson Health have started applying data analytics and machine learning techniques to combat the opioid problem (IBM 2017).

The problem of capturing underlying patterns in time sequences has been a long-standing problem in the field of Computer Vision. Recently, generating future patterns have been studied by various research group such as (Isola et al. 2017), (Johnson et al. 2016), (Nyugen et al. 2017) and (Han Zhang et al. 2017) The authors in (Junbo Zhang et al. 2017) have developed spatiotemporal residual networks for crowd flow prediction.

Predictive analysis of opioid incidences involves drawing inferences from a large set of features, many of which are difficult to identify and procure. In order to circumvent this overhead, we propose a methodology to predict the future hot spots (heat maps) by looking at hot spots of the previous months. The future hot spot prediction task requires a deep understanding of the trajectory of the previous incidence locations. We extend the concept presented by (Srivastava et al. 2015) to capture this property by modeling our framework on an encoder-decoder architecture, consisting of time-distributed convolutional layers. We transform the given task into a supervised-learning problem by sequencing monthly opioid-incidence heat maps into fixed-length spatiotemporal representations and utilize them to predict heap maps for the subsequent months.

Chapter 9

DATASETS

The model takes past daily heat maps as input and predicts the heat map for the subsequent day. We have tested the efficacy of our model on two publicly available Opioid incidence datasets:

1. The Cincinnati Heroin Overdose dataset:

   The Cincinnati dataset (*CD*) has been launched by the City of Cincinnati and contains detailed information regarding an Opioid-related incident in Cincinnati, such as location (latitudinal and longitudinal coordinates), time, EMS response type, neighborhood, etc. that require an EMS dispatch (45 features in total). This dataset contains incidences ranging from January 2016 till the present day. As of May 21, 2019, there are 7191 recorded Opioid incidences spanning 1235 days, spread across the neighborhoods of the city.

2. The Connecticut Drug-Related Death dataset:

   The Connecticut Accidental Drug (Opioid) Related Death dataset (*CN*) is very similar to the Cincinnati Dataset. Every row in the Connecticut Dataset denotes an Opioid-related incidence (death in this case) and contains 32 features for each death - sex, race, age, city/county of residence, city/county of death, latitude/longitudinal information, etc. This dataset is not updated as regularly but has 1612 mortality records.

Both datasets offer a plethora of features, but to generate the heat maps from these past data datasets we were primarily interested in three main features:

1. Latitude of Opioid Incident Occurrence
2. Longitude of Opioid Incident Occurrence
3. Date of Opioid Incident Occurrence

With these three features, we were able to generate the past data heat maps for every day in each of the datasets i.e. 1235 heat maps for the Cincinnati Dataset and 1612 heat maps for the Connecticut Dataset.

Chapter 10

APPROACH

In this section[1], the problem of predicting accurate heat maps of the future is formalized. To generate the heat map for a particular region over a specified timeframe, the latitudinal and longitudinal information present in the datasets have been utilized. A model which predicts the heat map for the subsequent day, having observed (or learned from) the past data has been developed. The procedure for the generation of the incidence heat maps for the Cincinnati dataset only, as the procedure is replicated in a similar manner for the Connecticut dataset has been discussed.

For the Cincinnati dataset, we construct CD' ($CD' \subset CD$), by extracting latitudinal and longitudinal information. A tuple $CD'_{\{d,x,y\}}$ contains three entries, where $d \in \{1,..,1235\}$, denotes the day and $\{x, y\}$ represents the latitudinal and longitudinal coordinates respectively. Similarly, for the Connecticut dataset, we generate $CN'_{\{d,x,y\}}$ where $d \in \{1,..,1612\}$ using the same method. For each value of d, a gray-scale image HM_d (heat map for day d) is generated by plotting the latitudinal and longitudinal coordinates $\{x,y\}$. HM_d has intensities ranging between $[0,255]$, with 255 in locations where maximum incidences have occurred and 0 in locations with no incidence data. All plots have been plotted on a predefined scale space, scaled to the Cincinnati land area.

Given a set of heat map images HM_d of n consecutive days, the objective is to predict the heat map for the subsequent day. Due to resource limitations, we could only consider n

[1]The following section's contents were taken from the paper "Predicting Future Opioid Incidences Today" by Sandipan Choudhuri, Kaustav Basu, Kevin Thomas and Arunabha Sen, for which I was the third author. The paper is currently on Arxiv at http://arxiv.org/abs/1906.08891.

(where $2 \leq n \leq 8$) consecutive days, for capturing the spatial and temporal dependencies of Opioid incidences.

The proposed task of predicting future hot spots can be formulated as a supervised learning problem. More formally, a training data-label pair is represented as a stacked volume of the n heat maps $\{HM'_t, HM'_{t+1}, ..., HM'_{t+n-1}\}$ corresponding to images of n consecutive *days* as input, and the HM'^{th}_{t+n} image as the train label, where, $1 \leq t \leq s$ and $s \leq |HM'| - n - 1$. Here, s controls the train-test split and is based on the cardinality of HM'. The value of s is based on the respective dataset and is explained in the subsequent paragraphs. Given missing data i.e. if there are no incidences reported for a day, the corresponding heatmap is a blank image of zero values.

For testing, given a stack $<HM'_t, HM'_{t+1}, ..., HM'_{t+n-1}>$ of n consecutive heat-maps, the model will output the heat map HM'_{t+n} for the $(t+n)^{th}$ day. Here, $(s+1) \leq t \leq |HM'| - n$. As mentioned earlier, to capture the dependency of incidence counts on varying scales of daily information, we worked with large values of n $(2 \leq n \leq 8)$ and found $n = 6$ to be optimal.

A novel deep neural network to solve the above-mentioned problem. The proposed learning task can be modeled under a Generative Adversarial Learning framework that handles spatiotemporal data. Consequently, we construct a model comprising of Attention-Based Stacked Convolutional LSTMs as the generative model G to predict heatmap for the next time-frame. The discriminative model D is based on the CNN architecture and performs convolution operations on the input heatmap in order to estimate the probability whether a sequence comes from the dataset or is produced by G. We use an adversarial loss to train the combined model (G and D). The prime intuition behind using this loss is that it can simulate the operating zones of incidences through historically available indicator data.

However, in practice, minimizing adversarial loss alone cannot guarantee to satisfy the predictions. *G* can generate samples that can confuse *D* without even being close to the actual distribution of future heatmaps. In order to tackle this problem, we propose a prediction error loss that minimized the L1-distance between the actual and generated samples. The model is trained using a joint loss function formed by the combination of adversarial and prediction error losses.

For the Cincinnati Heroin Dataset *CD,* the first 800 heat maps are used for training. The remaining 435 heat maps are used for testing. Similarly, for the Connecticut dataset, the first 1200 heat maps are used for training and the remaining 412 heat maps are used for testing. We trained our algorithm with a set of *n* values *(n = {1,2,3,4,5,6,7,8})* and found *n=7* to yield the minimum mean-squared error on the testing data of both datasets. The testing data heat maps for both datasets start from January 14th, 2017.

Here are a few terminologies used in the following section regarding the description of the different models:

1. Encoder-Decoder Framework:

   The Encoder-Decoder framework is a neural network divided into two parts i.e. the encoder (which is a neural network where the number of units in every hidden layer is lesser than the number of units in the input layer) and a decoder (which is a neural network where the number of units in every hidden layer is greater than the number of units in the input layer). When combined, the encoder-decoder framework aims to efficiently represent the initial input i.e. can map the input into a new input space.

2. Max Pooling:

   Max pooling is a sample-based discretization process which is used to down-sample an input representation, reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned. This is done in part to help over-fitting by providing an abstracted form of the representation. As well, it reduces the computational cost by reducing the number of parameters to learn and provides basic translation invariance to the internal representation. Max pooling is done by applying a max filter to (usually) non-overlapping subregions of the initial representation (Max Pooling 2019).

3. Scalar Exponential Linear Unit (SELU) Activation function:

   The SELU activation function is given by the formula:

$$f(x) = \lambda \begin{cases} \alpha(e^x - 1), & x < 0 \\ x, & x \geq 0 \end{cases} \qquad (14)$$

   The λ and the α are fed as input to the function.

4. Transposed Convolution Neural Network (ConvTranspose):

   Transposed Convolution is used to conduct up-sampling. For example, in a CNN layer, the input is of size 4x4, filter size of 3x3, a stride of 1x1 and no padding; we will get an output of size 2x2. If the need arises to up-sample the 2x2 matrix into the 4x4 matrix, Transposed Convolution is used.

We evaluate the performance of the proposed method on both datasets (Cincinnati and Connecticut), with three other standard machine-learning techniques:

1. Convolutional LSTMs (ConvLSTMs):

   A classic five-layer ConvLSTM model with Global Average Pooling as the last layer.

2. Attention-based ConvLSTMs (Att-ConvLSTMs) (A variant of the model present by (Byoen et al. 2018)):

   The layers used in the Att-ConvLSTM model is as follows:

   a. Two-Dimensional Convolution LSTM Layer with 64 units, a 3x3 filter, a 1x1 stride and *'same'* padding.
   b. Max Pooling with a 2×2 filter
   c. Two-Dimensional Convolution LSTM Layer with 128 units, a 3x3 filter, a 1x1 stride and *'same'* padding.
   d. Max Pooling with a 2×2 filter
   e. Two-Dimensional Convolution LSTM Layer with 256 units, a 3x3 filter, a 1x1 stride and *'same'* padding.

   To highlight the degree of importance that the features from each time-frame exhibit, we weigh each feature map using a *softmax* attention layer. Scaled Exponential Linear Unit (*SELU*) is used as the activation function in each convolutional block.

3. Time-Distributed version of Convolution Encoder-Decoder Framework built upon the UNet++ model (Zhou et al. 2018) (TD Conv-Env-Dec):

   A classic Encoder-Decoder framework which takes the time distributed heat maps as input and produces future heat maps as output.

As these architectures have different input configuration specifications, the input stacks of heat-maps are configured specifically for each model. For Convolutional-LSTMs and TD-Conv-Enc-Dec, the input stack of heat-maps is scaled in the range *[0,1]*. For our approach, we utilize the Wasserstein distance as the adversarial loss function. To ensure stability in the training process, the input stacks are scaled within the range *[-1,1]*.

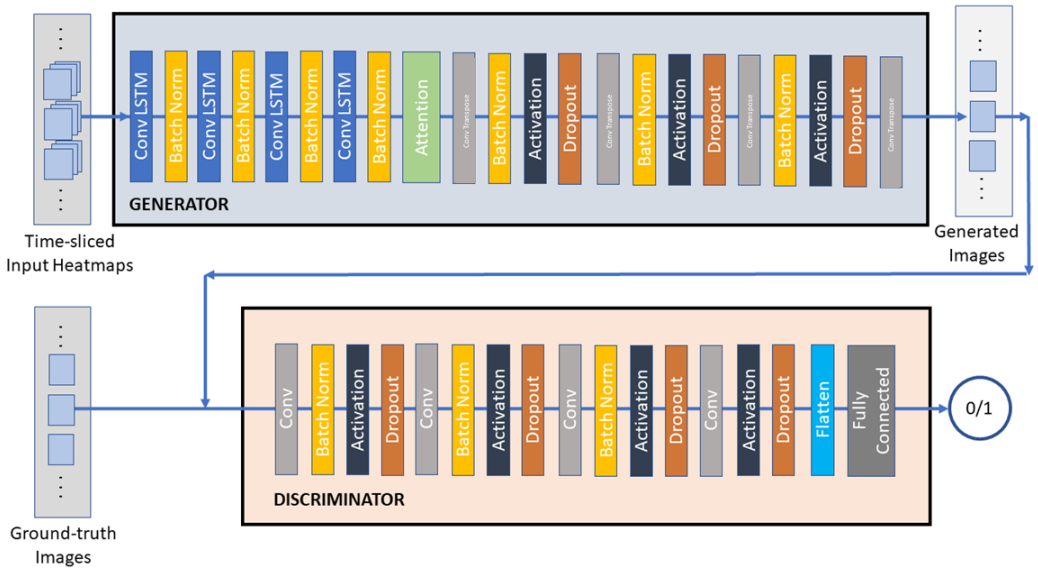

Figure 23: GAN model for Opioid Prediction

The learning task can also be modeled on an encoder-decoder framework TD-Conv-End-Dec, that handles data of different temporal scales. The architecture should be able to bridge the semantic gap between feature maps generated from temporal data. (Ronneberger et al. 2015) and (Drozdzal et al. 2016) systematically investigated the importance of skip connections to capture semantic links between feature maps. We build our model on *UNet++*, an encoder-decoder architecture with nested skip-pathways, proposed by (Zhou et. al.) As our input data is a stack of time-dependent images, we use a

series of nested time-distributed dense convolutional blocks. The nested skip pathways over the time-distributed convolutional layers aid in reducing the semantic gap between feature maps of the encoder and decoder, prior to feature-fusion. This is followed by aggregation of feature-information from *n* different temporal-scales *(n=6)*, where we employ global average-pooling over the flattened output of the last convolutional layer. Feature-reshaping is performed over the output of the global-pooling layer to generate the heat map.

For the proposed model, *`RMSProp'* is used as the model optimizer instead of the standard gradient descent optimizer because *RMSProp* optimizer allows us to have a higher learning rate to converge to the global minima faster and making sure we do not overshoot the global minima because it restricts the oscillations in the vertical direction (Rohith Gandhi 2018). Learning rate set to 0.00005. Our model is trained for 1500 epochs with a batch size of 8.

The structure of the Generator is as follows:

1. Two-Dimensional Convolutional LSTM layer with 16 units, a 3x3 filter, a 1x1 stride and *'same'* padding.
2. Batch Normalization Layer
3. Two-Dimensional Convolutional LSTM layer with 32 units, a 3x3 filter, a 1x1 stride and *'same'* padding.
4. Batch Normalization Layer
5. Two-Dimensional Convolutional LSTM layer with 64 units, a 3x3 filter, a 1x1 stride and *'same'* padding.
6. Batch Normalization Layer

7. Two-Dimensional Convolutional LSTM layer with 128 units, a 3x3 filter, a 1x1 stride and *'same'* padding.
8. Batch Normalization Layer
9. Attention Layer
10. Two-Dimensional Transposed Convolutional Neural Network layer with 128 units, a 3x3 filter, a 1x1 stride and *'same'* padding.
11. Batch Normalization Layer
12. Leaky ReLU Activation Layer with $\alpha = 0.2$
13. Dropout with dropout rate = 0.3
14. Two-Dimensional Transposed Convolutional Neural Network layer with 64 units, a 3x3 filter, a 1x1 stride and *'same'* padding.
15. Batch Normalization Layer
16. Leaky ReLU Activation Layer with $\alpha = 0.2$
17. Dropout with dropout rate = 0.3
18. Two-Dimensional Transposed Convolutional Neural Network layer with 32 units, a 3x3 filter, a 1x1 stride and *'same'* padding.
19. Batch Normalization Layer
20. Leaky ReLU Activation Layer with $\alpha = 0.2$
21. Dropout with dropout rate = 0.3
22. Two-Dimensional Transposed Convolutional Neural Network layer with 16 units, a 3x3 filter, a 1x1 stride and *'same'* padding.

The structure of the Discriminator is as follows:

1. Two-Dimensional Convolutional Neural Network layer with 16 units, a 3x3 filter, a 1x1 stride and *'same'* padding.
2. Batch Normalization Layer
3. Leaky ReLU Activation Layer with $\alpha = 0.2$
4. Dropout with dropout rate = 0.25
5. Two-Dimensional Convolutional Neural Network layer with 16 units, a 3x3 filter, a 1x1 stride and *'same'* padding.
6. Batch Normalization Layer
7. Leaky ReLU Activation Layer with $\alpha = 0.2$
8. Dropout with dropout rate = 0.25
9. Two-Dimensional Convolutional Neural Network layer with 16 units, a 3x3 filter, a 1x1 stride and *'same'* padding.
10. Batch Normalization Layer
11. Leaky ReLU Activation Layer with $\alpha = 0.2$
12. Dropout with dropout rate = 0.25
13. Two-Dimensional Convolutional Neural Network layer with 16 units, a 3x3 filter, a 1x1 stride and *'same'* padding.
14. Batch Normalization Layer
15. Leaky ReLU Activation Layer with $\alpha = 0.2$
16. Dropout with dropout rate = 0.25
17. Flatter Layer
18. Dense

This is the model structure and the inputs are trained and tested on this structure. The loss function used in the GAN model is a combination of the adversarial loss function called Wasserstein distance function and the Mean Square Error Loss and is given by the formula:

$$L_{GAN} = 0.6 * L_{Wasserstein} + 0.4 * L_{MSE} \qquad (15)$$

A combination of two loss functions is used because the Wasserstein loss calculates the distance between the two probability distributions but not how far apart the predicted values are. Hence MSE is also incorporated in the loss to account for how far apart the predicted values are.

Every hyperparameter in this structure has been selected after running the model in multiple iterations with different sets of hyperparameters. The above final set of hyperparameters is the set that produces the best results.

Chapter 11

EXPERIMENTAL RESULTS

The schematic diagram is given in Figure 21. Mean-squared (MSE) and mean-absolute errors (MAE) are used as evaluation metrics for comparing the above-mentioned techniques with our proposed model. The results are presented in Table 8 and Table 9. Twenty-five daily Opioid incidence predictions for the Cincinnati and Connecticut datasets are illustrated in Figure 24 to Figure 28. The left column indicates the ground truth images for the respective daily incidences. The right column images are the predicted heat maps for the corresponding days. Greater the intensity, greater the likelihood of Opioid incidences occurring in that geographical area.

Method	**MSE**	**MAE**
ConvLSTM	0.0628	0.0083
Att-ConvLSTM	0.04256	0.0062
TD-Conv-End-Dec	0.0562	0.0075
GAN model	**0.03371**	**0.0057**

Table 8: Evaluation Results of the following methods on Cincinnati dataset

Method	**MSE**	**MAE**
ConvLSTM	0.0583	0.0081
Att-ConvLSTM	0.0415	0.0067
TD-Conv-End-Dec	0.0517	0.0073
GAN model	**0.0309**	**0.0052**

Table 9: Evaluation Results of the following methods on Connecticut dataset

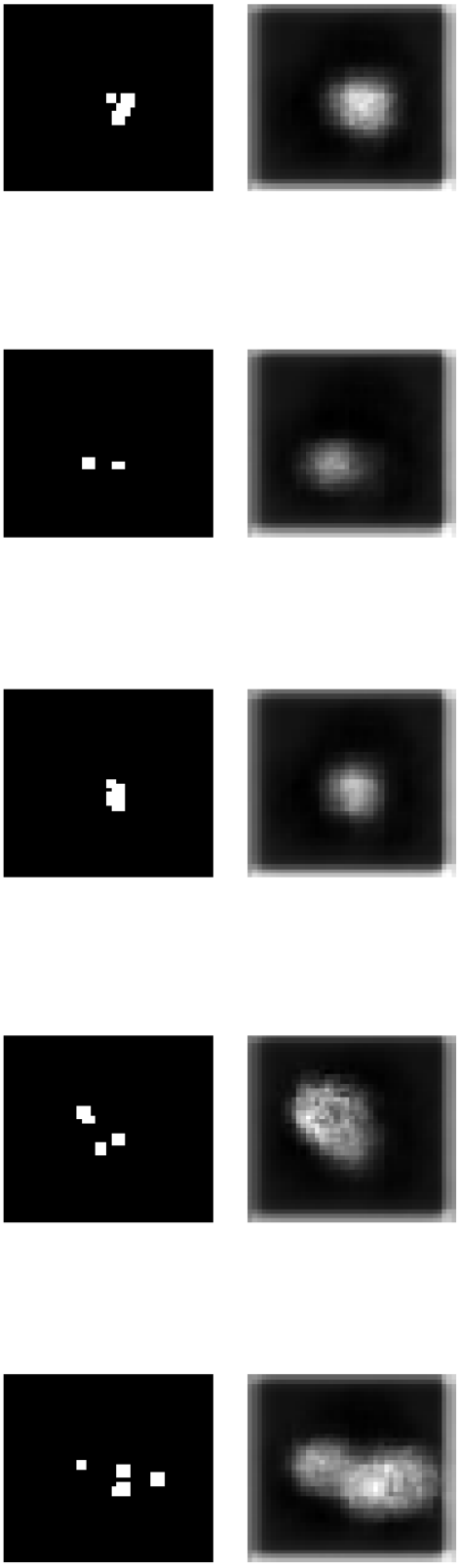

Figure 24: Opioid Output Set 1

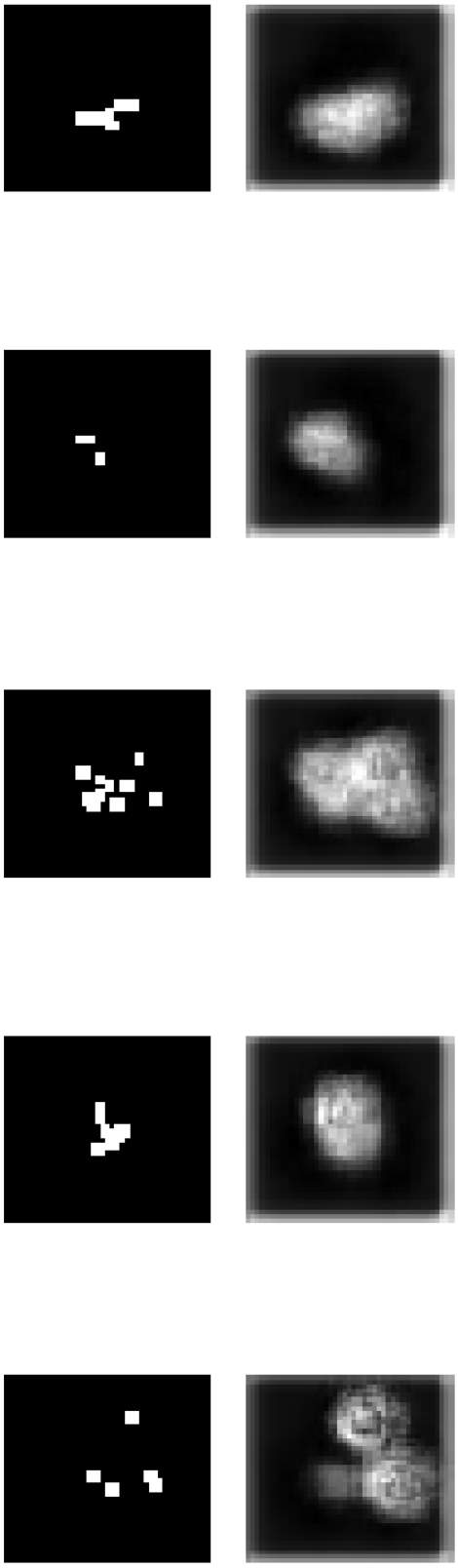

Figure 25: Opioid Output Set 2

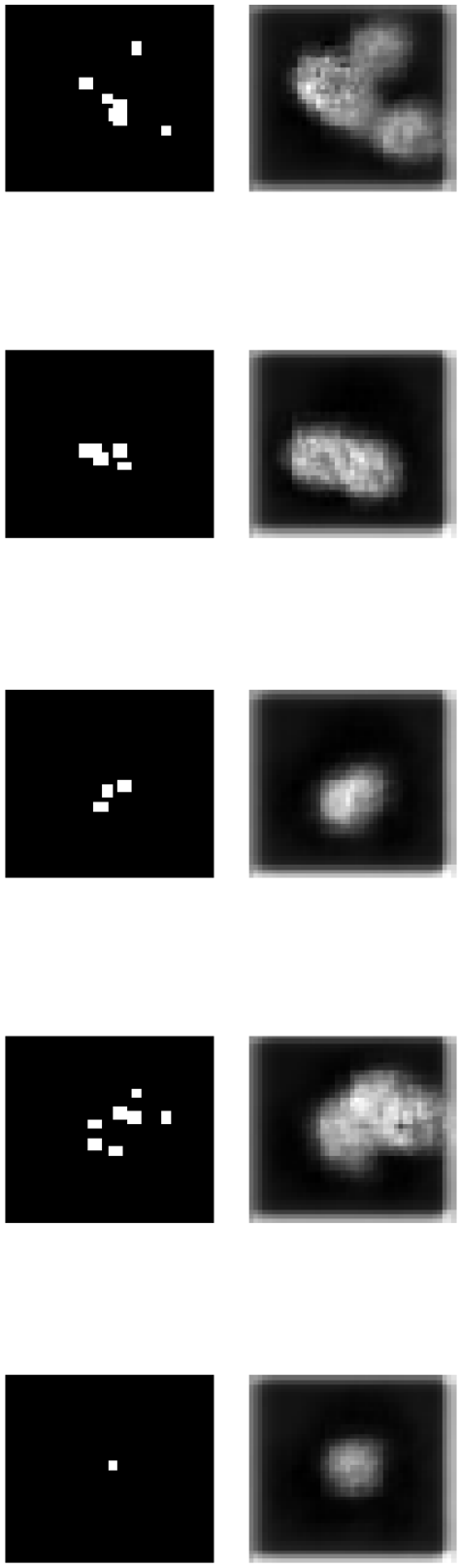

Figure 26: Opioid Output Set 3

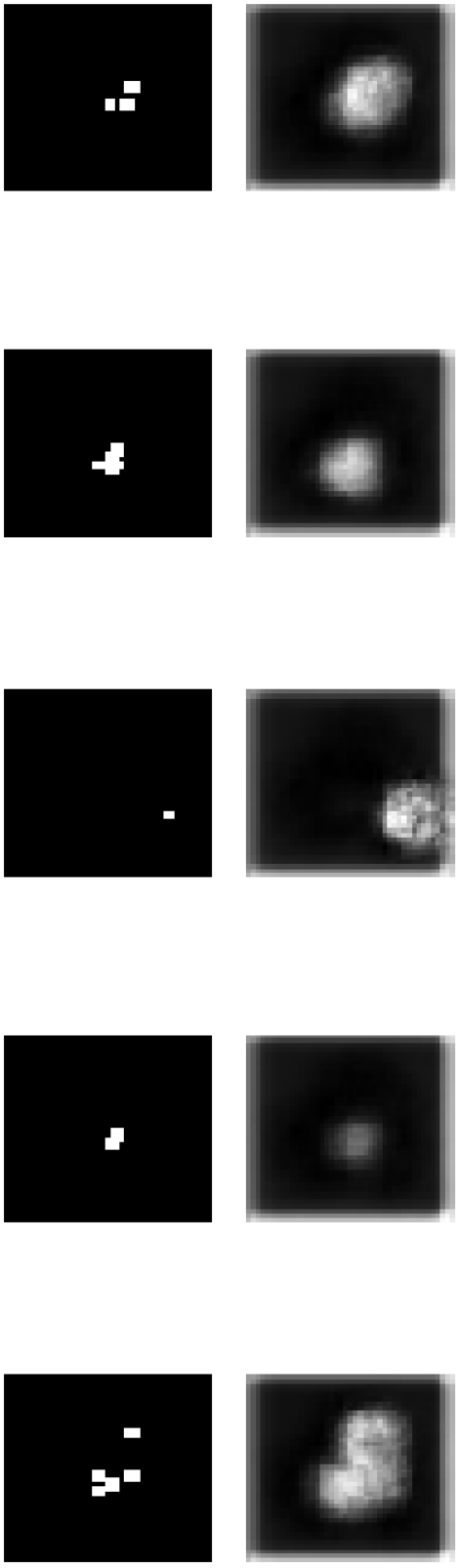

Figure 27: Opioid Output Set 4

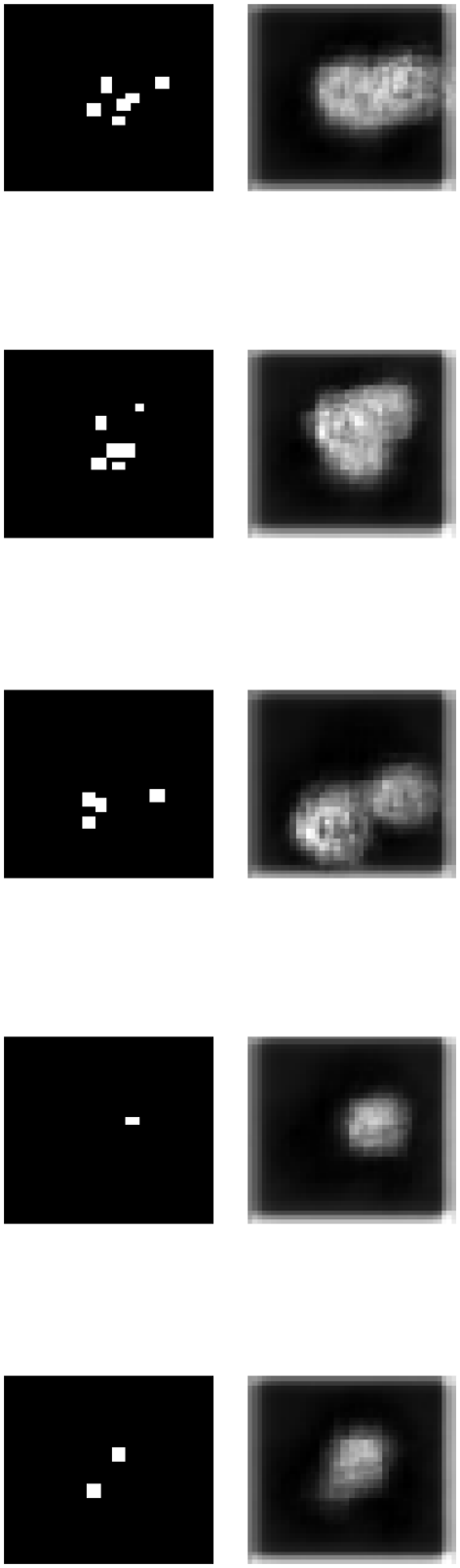

Figure 28: Opioid Output Set 5

Table 10 and Table 11 shows the average model training time and the average time taken to test one sample for the model for ConvLSTM, Att-ConvLSTM, TD-Conv-End-Dec and the Opioid GAN models for both the Cincinnati and Connecticut datasets. The training time was calculated by measuring the time taken for the model to train on the training set. The time taken to test one sample was calculated by dividing the time taken to test for the entire test set by the number of samples in the test set.

Method	**Training Time**	**Time for testing 1 sample**
ConvLSTM	4 hrs 32 min	0.3 s
Att-ConvLSTM	5 hrs 13 min	0.4 s
TD-Conv-End-Dec	6 hrs 58 min	0.7 s
GAN model	**8 hrs 3 min**	**0.7 s**

Table 10: Time taken for Overdose Location Prediction Models for Cincinnati Dataset

Method	**Training Time**	**Time for testing 1 sample**
ConvLSTM	5 hrs 48 min	0.3 s
Att-ConvLSTM	6 hrs 27 min	0.4 s
TD-Conv-End-Dec	8 hrs 34 min	0.7 s
GAN model	**9 hrs 6 min**	**0.7 s**

Table 11: Time taken for Overdose Location Prediction Models for Connecticut Dataset

All the models were trained on an 8-core processor with Tesla V-100 (16GB variant) GPU for 4000 epochs with a batch size of 1024.

Chapter 12

THESIS CONCLUSION

In this thesis document, I investigated time series analysis and time series forecasting using novel deep learning methods in two cases, which are as follows:

1. Stock Data Prediction
2. Opioid Incident Prediction

In the first case of Stock Data Prediction, a state-of-the-art deep learning architecture using a Generative Adversarial Network (GAN) has been presented for the prediction of the closing price of the stock. The Discriminator train the Generator using the input data with 20 features. The performance of the model was compared to the performance of three regression models i.e. Huber regression, Linear regression, and Ridge regression; as well as an LSTM model. The performance metrics used for comparison were Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). It is evident from the results that for the next minute prediction, the GAN model results were comparable to the regression models and beats the LSTM models; but for the third-day prediction and the seventh-day prediction, the GAN model trumps all the regression models and the LSTM models by quite a margin. With respect to train and test time, the regression techniques are faster than the deep learning models due to the relative simplicity of regression over deep learning. If the deep learning techniques are compared, then the GAN model takes a longer time to train and test than the LSTM model but the testing times are relatively close compared to the training time. The GAN model does take a longer time to train and test but does give better results in most cases.

In the second case of Opioid Incident Prediction, a novel deep learning architecture using a GAN has been presented for the generation of future heat maps, by analyzing past Opioid incidences, for the Cincinnati and Connecticut datasets. The input heat maps were generated using the latitude and longitude coordinates in the dataset and were associated with the attached date. The performance of the model was compared to existing works using Convolutional LSTMs (ConvLSTMs), Attention-based ConvLSTMs and Encoder-Decoder frameworks. Even with a small dataset, it has been observed that the predictions provided by the model are accurate and are better than the existing models. The performance metrics used were Mean Square Error (MSE) and Mean Absolute Error (MAE). With respect to the training times, the GAN model takes the longest time to train compared to the other three models. With respect to the testing times, the ConvLSTM and Att-ConvLSTM models are relatively faster but the times of the TD-Conv-End-Dec model is comparable to the GAN model. The GAN model does take a longer time to train (and test in some cases) but does give better results.

REFERENCES

Acion, Laura, Diana Kelmansky, Mark van der Laan, Ethan Sahker, DeShauna Jones, and Stephan Arndt. "Use of a machine learning framework to predict substance use disorder treatment success." PloS one 12, no. 4 (2017): e0175383.

Alostad, Hana, and Hasan Davulcu. "Directional prediction of stock prices using breaking news on Twitter." In 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), vol. 1, pp. 523-530. IEEE, 2015.

Batch Normalization Wikipedia 2019. https://en.wikipedia.org/wiki/Batch_normalization

BCBS 2017. America's Opioid Epidemic and its Effect on the Nation's Commercially-Insured Population. https://www.bcbs.com/the-healthof-america/reports/americas-opioid-epidemic-and-its-effect-on-the-nationscommercially-insured

Bohnert, Amy SB, Marcia Valenstein, Matthew J. Bair, Dara Ganoczy, John F. McCarthy, Mark A. Ilgen, and Frederic C. Blow. "Association between opioid prescribing patterns and opioid overdose-related deaths." Jama 305, no. 13 (2011): 1315-1321.

Bollen, Johan, Huina Mao, and Xiaojun Zeng. "Twitter mood predicts the stock market." Journal of computational science 2, no. 1 (2011): 1-8.

Boris Banushev GAN model 2019. https://github.com/borisbanushev/stockpredictionai

Byeon, Wonmin, Qin Wang, Rupesh Kumar Srivastava, and Petros Koumoutsakos. "Contextvp: Fully context-aware video prediction." In Proceedings of the European Conference on Computer Vision (ECCV), pp. 753-769. 2018.

CDC 2017. https://www.cdc.gov/drugoverdose/epidemic/index.html

Che, Zhengping, Jennifer St Sauver, Hongfang Liu, and Yan Liu. "Deep Learning Solutions for Classifying Patients on Opioid Use." In AMIA Annual Symposium Proceedings, vol. 2017, p. 525. American Medical Informatics Association, 2017.

Chou, Roger, Judith A. Turner, Emily B. Devine, Ryan N. Hansen, Sean D. Sullivan, Ian Blazina, Tracy Dana, Christina Bougatsos, and Richard A. Deyo. "The effectiveness and risks of long-term opioid therapy for chronic pain: a systematic review for a National Institutes of Health Pathways to Prevention Workshop." Annals of internal medicine 162, no. 4 (2015): 276-286.

Cicero, Theodore J., and Matthew S. Ellis. "The prescription opioid epidemic: a review of qualitative studies on the progression from initial use to abuse." Dialogues in clinical neuroscience 19, no. 3 (2017): 259.

Cincinnati. 2019. Cincinnati Heroin Overdoses. https://insights.cincinnatioh.gov/stories/s/Heroin/dm3s-ep3u/

Connecticut. 2019. Accidental Drug Related Deaths. https://data.ct.gov/Healthand-Human-Services/Accidental-Drug-Related-Deaths-2012-2018/ecj5-r2i9

D Wei. 2017. Combating the Opioid Epidemic with Machine Learning. https://www.ibm.com/blogs/research/2017/08/combating-the-opioidepidemic-with-machine-learning/

Drozdzal, Michal, Eugene Vorontsov, Gabriel Chartrand, Samuel Kadoury, and Chris Pal. "The importance of skip connections in biomedical image segmentation." In Deep Learning and Data Labeling for Medical Applications, pp. 179-187. Springer, Cham, 2016.

FirstRateData 2019. http://firstratedata.com/

Gong, Jibing, and Shengtao Sun. "A new approach of stock price prediction based on logistic regression model." In 2009 International Conference on New Trends in Information and Service Science, pp. 1366-1371. IEEE, 2009.

Guresen, Erkam, Gulgun Kayakutlu, and Tugrul U. Daim. "Using artificial neural network models in stock market index prediction." *Expert Systems with Applications* 38, no. 8 (2011): 10389-10397.

Henrique, Bruno Miranda, Vinicius Amorim Sobreiro, and Herbert Kimura. "Stock price prediction using support vector regression on daily and up to the minute prices." The Journal of finance and data science 4, no. 3 (2018): 183-201.

IBM 2017. IBM Opioid Github. https://github.com/IBM/predict-opioidprescribers

Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015).

Isaac Changhau 2017. https://isaacchanghau.github.io/post/loss_functions/

Isola, Phillip, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. "Image-to-image translation with conditional adversarial networks." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1125-1134. 2017.

Jason Brownlee 2017. https://machinelearningmastery.com/decompose-time-series-data-trend-seasonality/

Jason Brownlee 2018. https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/

Johnson, Justin, Alexandre Alahi, and Li Fei-Fei. "Perceptual losses for real-time style transfer and super-resolution." In European conference on computer vision, pp. 694-711. Springer, Cham, 2016.

Katz and Goodnough 2017. https://www.nytimes.com/interactive/2017/12/22/upshot/opioid-deaths-are-spreading-rapidly-into-black-america.html

Kaya Yurieff 2018. https://money.cnn.com/2018/02/22/technology/snapchat-update-kylie-jenner/index.html

Khan, Umair, Farhan Aadil, Mustansar Ghazanfar, Salabat Khan, Noura Metawa, Khan Muhammad, Irfan Mehmood, and Yunyoung Nam. "A Robust Regression-Based Stock Exchange Forecasting and Determination of Correlation between Stock Markets." Sustainability 10, no. 10 (2018): 3702.

Leaky ReLU Image 2018. https://blog.paperspace.com/vanishing-gradients-activation-function/

LSTM Wikipedia 2019. https://en.wikipedia.org/wiki/Long_short-term_memory

LSTM cell Image Wikipedia 2019. https://en.wikipedia.org/wiki/Long_short-term_memory#/media/File:The_LSTM_cell.png

Mackey, Tim K., Janani Kalyanam, Takeo Katsuki, and Gert Lanckriet. "Twitter-based detection of illegal online sale of prescription opioid." American journal of public health 107, no. 12 (2017): 1910-1915.

Mao, Yuexin, Wei Wei, Bing Wang, and Benyuan Liu. "Correlating S&P 500 stocks with Twitter data." In Proceedings of the first ACM international workshop on hot topics on interdisciplinary social networks research, pp. 69-72. ACM, 2012.

Mathematica Stack Exchange 2019. https://mathematica.stackexchange.com/questions/16723/time-series-decomposition-in-mathematica

Max Pooling 2019. https://computersciencewiki.org/index.php/Max-pooling_/_Pooling

NCSS 2019. https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Ridge_Regression.pdf

Neill, Daniel B., and William Herlands. "Machine learning for drug overdose surveillance." Journal of Technology in Human Services 36, no. 1 (2018): 8-14.

Nguyen, Anh, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. "Plug & play generative networks: Conditional iterative generation of images in latent space." In

Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4467-4477. 2017.

Rice, J. Bradford, Alan G. White, Howard G. Birnbaum, Matt Schiller, David A. Brown, and Carl L. Roland. "A model to identify patients at risk for prescription opioid abuse, dependence, and misuse." Pain Medicine 13, no. 9 (2012): 1162-1173.

Rohith Gandhi 2018. https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." In International Conference on Medical image computing and computer-assisted intervention, pp. 234-241. Springer, Cham, 2015.

Roondiwala, Murtaza, Harshal Patel, and Shraddha Varma. "Predicting stock prices using LSTM." International Journal of Science and Research (IJSR) 6, no. 4 (2017): 1754-1756.

Roy, Sanjiban Sekhar, Dishant Mittal, Avik Basu, and Ajith Abraham. "Stock market forecasting using LASSO linear regression model." In Afro-European Conference for Industrial Advancement, pp. 371-381. Springer, Cham, 2015.

Srivastava, Nitish, Elman Mansimov, and Ruslan Salakhudinov. "Unsupervised learning of video representations using lstms." In International conference on machine learning, pp. 843-852. 2015.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting." The journal of machine learning research 15, no. 1 (2014): 1929-1958.

Stock Market Wikipedia 2019. https://en.wikipedia.org/wiki/Stock_market

SkyMind AI 2019. https://skymind.ai/wiki/generative-adversarial-network-gan

Tan, Jinghua, Jun Wang, Denisa Rinprasertmeechai, Rong Xing, and Qing Li. "A Tensor-based eLSTM Model to Predict Stock Price using Financial News." In Proceedings of the 52nd Hawaii International Conference on System Sciences. 2019.

Time Series Wikipedia 2019. https://en.wikipedia.org/wiki/Time_series

Wasserstein Distance Wikipedia 2019. https://en.wikipedia.org/wiki/Wasserstein_metric

Xingjian, S. H. I., Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. "Convolutional LSTM network: A machine learning approach for precipitation nowcasting." In Advances in neural information processing systems, pp. 802-810. 2015.

Zero-sum Game Wikipedia 2019. https://en.wikipedia.org/wiki/Zero-sum_game

Zhang, Chuxu, Dongjin Song, Yuncong Chen, Xinyang Feng, Cristian Lumezanu, Wei Cheng, Jingchao Ni, Bo Zong, Haifeng Chen, and Nitesh V. Chawla. "A Deep Neural Network for Unsupervised Anomaly Detection and Diagnosis in Multivariate Time Series Data." arXiv preprint arXiv:1811.08055 (2018).

Zhang, Han, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N. Metaxas. "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks." In Proceedings of the IEEE International Conference on Computer Vision, pp. 5907-5915. 2017.

Zhang, Junbo, Yu Zheng, and Dekang Qi. "Deep spatio-temporal residual networks for citywide crowd flows prediction." In Thirty-First AAAI Conference on Artificial Intelligence. 2017.

Zhou, Zongwei, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. "Unet++: A nested u-net architecture for medical image segmentation." In Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support, pp. 3-11. Springer, Cham, 2018.

BIOGRAPHICAL SKETCH

Kevin Thomas was born in Kozhikode, India on August 12th, 1993. He received his formal education from Vidyashilp Academy, Bangalore, India. His higher secondary education was completed at Sindhi High School, Bangalore, India. In 2011, Kevin joined R.V. College of Engineering, Bangalore, India for a Bachelor's Degree in Computer Science Engineering. Upon graduating in 2015, he joined as a Product Engineer for Sprinklr Inc. in the Bangalore office, working in the Technical Services team. In 2017, Kevin left Sprinklr Inc. and joined Arizona State University for Masters in Computer Science where he decided to pursue his Master's thesis.