University of Rajshahi

Department of
Information and Communication Engineering

Stock Market Prediction by Taking Random Values Allying Two Deep Learning Models

Author :
Syed Zaki Reza
Roll : 1710677122

Supervisor :
Prof. Dr. Dipankar Das
Dept. of ICE

University of
Rajshahi

Academic Year : 2020

ICE4242 : Research Project

Abstract

The stock market is a place to lay investments in companies to boost their growth. The stock market can play an important role in a nation's future. A good stock market of a country always produces a decent mindset for entrepreneurs in those countries. But the stock market is a very volatile place. The price fluctuates rapidly in a short moment. There is also some common misconception among small shareholders that big companies always have a good price. The stock price can be changed due to the company's profit or loss at that moment, but it is not only bound to that. The weather forecast, festivals, and international relations of countries also play an important role. However, this project is for general purposes, to predict stock in normal situations. Anyone can use the data to grasp the whole situation of a company for one year. By stock prediction, govt. may also find irregular and suspicious stock fluctuation. To sell and buy stocks only help of stock prediction will be a very risky idea. But to find out some trends, prediction can help. Here, we have used time-series data to predict the next values. Normal deep learning models performs very well by learning complex time-shifted correlations between stepwise trends of a large number of noisy time series, using only the preceding time steps' gradients as inputs. Thus, different models predict different results. Such correlations are present in stock prices, and these models can be used to predict changes in a price's trend based on other stocks' trend gradients of the previous time step. In more narrowly defined terms, this applied part is situated at the intersection of computational finance and financial econometrics. Combining and comparing two or more models can give us a good result. And combining it with random values may increase the fixed trends of a specific model. Thus, an average value and randomness can give us a better insight.

Acknowledgements

Foremost, I wish to express my gratitude toward my advisor, Prof. Dr. Dipankar Das, for helping me to get in the right direction, and for taking on a rather unusual project that combines deep learning for time series analysis with using randomized data for prediction. From my university first year, I always got his help in different fields. Also, his insight into a deeper understanding of technologies helps us to get through the root. I am indebted to him. His guidance, teaching, and engagement with students help everyone to overcome hardship. He is someone who inspires all students around him. All the professors I met, I am indebted to them. The teachings and engagement are something that cannot be explained in words. I extend my gratitude to all my professors who helped me to achieve my goal.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(Syed Zaki Reza)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, project-related motivation and the enthusiasm to work in this field have been described. The project summary as well as the hypothesis to experiment in different aspects has been explained. A brief outline is given to grasp the whole project at a glance.

1.1 Motivation

Stock market prediction is a potentially lucrative and attractive way to generate profit through investing in profitable companies and avoiding risks. While it is very difficult to predict the future values of a company. A vast amount of objectives are responsible for the fluctuation of stock prices. Even so, people seek mathematical possibilities to predict future prices since the late 1960s. A famous book named ‘Beat the Market’ was published by a professor of mathematics called Edward Thorp, at the University of California. After that, a revolution could be seen in finance firms to predict company values through mathematics. Mathematicians were hired as quants to predict stock price. A hedge fund named Renaissance Technologies founded by mathematicians is famous for its systematic trading using quantitative models and generating a phenomenal return. From 1994 through mid-2014, Renaissance's flagship Medallion fund on average generated a 71.8% annual return.

Although mathematical models and machine learning models can predict ample data, this project is focused on predicting the trends, fluctuations, and risky situations of the stock market. Without proper knowledge about the stock market, anyone can lose all their savings in one go. This project is to determine low-risk periods and check irregularities so that, anyone can find a prudent way to invest in the stock market. The stock market is a very risky place to find a proper formula to make a profit. So, the least the risk, the lowest the loss.

1.2 Problem Description

To create an effective deep learning model, a large number of weights are used. But among those weights, some play an immensely significant role. Among some other weights, many play a notable role too. But because of certain weights, a deep learning model can give a different outcome just for a small change in those weights. Thus, a deep learning model depends extravagantly on some scant number of weights. A deep learning model as a whole can be more biased towards a specific result because of certain weights and parameters.

Other than the profit-loss and company valuation principle at a given moment, the stock market fluctuates randomly in a given little amount of time. As linear regressions on time series are a simple measurement of trends, small randomness cannot be predicted by an Artificial Neural Network (ANN) model. Again, the prediction of time series data depends too much on some insignificant amounts of weights. The stock market is also fluctuating because of human emotions. Which can be random in any given situation.

1.4 Relevance

1.3.1 Economy and Stock Market

Development toward a strong economy always relies on productive investments. To achieve a better transition of capital from the investor to entrepreneur, a capable financial system. A stock market is a place to present investors about a company and accumulate required capital. A good investment and entrepreneurial strategy always boost economic growth. Any country heavily dependent on only a single source, such as a bank or government institution for financial resources can turmoil the whole economy if anything goes wrong. One of the major reasons for the financial crisis in 1997-1998 was because of heavy dependency on banking systems to finance domestic investment. A well-balanced diversified financial institution assists an economy to run sustainable growth. A large and sustainable stock market indicates a healthy economy and a flourishing future for a country. This also helps a country to attract domestic and foreign investment thus tending towards further economic growth.

1.3.2 Applied Fields of Deep Learning

Not so long ago, deep learning was only applicable for small amounts of data computation. But recently, due to the progress of the Graphics Proccing Unit (GPU), deep learning has become popular among enthusiastic people. Because of their cheap computation ability, multilayer models can compute a large amount of data and solve very complex problems. Thus, series type data prediction such as Natural Language Processing (NLP), Computer Vision, Voice Processing, Video Processing, Time-Series Forecasting, Robotic Process Automation (RPA), Data Refining, etc. became a norm nowadays. With a vast amount of research, new models are revealing, and models are becoming more efficient.

1.3.3 Time Series Analysis

It is a statistical technique that analyzes series data or trending data changing throughout time intervals. Time Series Analysis is the way to look into the characteristics and extract statistical data of the response variable with respect to time, as the independent variable. A time series is a sequence of various data coupled in a successive order for a certain time period. Time series analysis is used for forecasting or recognizing patterns at a particular time. Time series analysis is a very good tool to analyze the stock market. A gradient-based approach to trend prediction relying on previous series information of correlated variables leads to better results in prediction data. Assimilating time series analysis into deep learning gives us a progressive result.

1.4 Outline of This Project

Chapter 1 acts as an introduction to the topics that form parts of this project. In here stock market prediction, applied machine learning, and time series analysis are discussed. To draw valid conclusions from the results of the experiments some hypothesis is described. Following this initial introduction and overview, the problem that is being solved in this project is also scrutinized.

Chapter 2 narrates the relevant theories and highlights deeper into the existing established mechanisms. Theories related to the economy are also emphasized. Some technologies used in this project are also expressed.

Chapter 3 is all about the details of the methodology and setups for the experiments performed for this project. All the process step by step is explained here. The hypothesis that relates to this project is being implemented here. T chapter subsequently discusses the fundamental problems that can occur performing trend prediction for time series. A description of a special case scenario has been explained.

Chapter 4 gives the insight and validation of this project. The major evaluation process came front to reveal the significance of the project. The risk involved in the crisis time is also focused on in this section.

Chapter 5 is about the findings that this project discloses. Ideas about the implementation of the system and new ways came through the discussions. Some crisis-related findings are also described.

Chapter 6 ends the whole project by summarizing each step. Conclusions are drawn about the findings and hypotheses. Also, how further development has been discussed. Here the potentiality and the capability are illustrated.

Chapter 2

Background Research

In this chapter, the reader will find relevant topics and hypotheses about the project. Also, different topics are discussed here. Some important fields are explained here.

2.1 Stock Market Prediction

2.1.1 The Efficient-Market Hypothesis

It is a hypothesis that states that asset prices reflect all information in a market. So, it is impossible to always get a desirable positive outcome on a risk-adjusted basis, because all prices reflect all available information, which tends toward new information. The EMH hypothesis came forward in 1900, but it become famous by Eugene Fama in 1970. The modern risk-based theories of asset prices and frameworks proceed forward with help of EMH basic logic. According to the EMH hypothesis, the stock price at a certain time is always at a fair exchange value. For that, no stock price can be overvalued or undervalued at that moment and, arguably, this theory is not applicable after a long period. EMH can be classified into three categories. Semi-Strong form EMH states that the current stock price reflects all publicly available information. Weak-Strong form EMH states that current stock value is decided on all the past values, thus technical analysis is useless to profit more than average returns. Strong form EMH states that because all the information lies in the market, other than random chance, it is impossible to earn more than the expected market average. The Random Walk Hypothesis states that because all the information is publicly available, no technical analysis can help to generate more profit than average. But new information can lead to changing stock prices differently. [1] [2]

2.1.1 Time Series-Based Prediction

It is a system where profit is generated more than the average return based on past series of stock information without increasing risk exposure. It assumes future trends to be the same as past trends delineated in historical data. Some important components of time series are level, trend, seasonality, and noise. The level is the constraint area that is the range of fluctuations of the price. The trend is the behavioral changes in time series. Seasonality indicates the repeating pattern of the behavioral cycle over time series. The nose is a collection of some variables that are not explainable by any model. Although time series-based prediction is inconsistent with the random walk hypothesis and all forms of the efficient-market hypothesis, it is practiced in all industries. ANN is very efficient in time series-based prediction because of its arbitrary function mapping, non-linearity, and adaptability. ANN layers can determine the relative arbitrary parameters and establish relationships among them by calculating historical data of stock price. Some ANN methods such as autoencoders and Boltzmann machine works conveniently by feature learning and dimensional reduction. There are also some popular methods AutoRegressive Integrated Moving Average (ARIMA) and Seasonal ARIMA (SARIMA). In AutoRegressive forecasts, predicting the future is based on the linear combination of past values of the variable. And in the Moving Average method of forecasts, predicting the future is based on the linear combination of past forecast errors. Another method TBATS (Trigonometric, Box-Cox transform, ARMA errors, Trend, and Seasonal components) predicts future changes based on exponential smoothing. [3]

2.2 Deep Neural Network

2.2.1 Introduction to Artificial Neural Network

Artificial Neural Networks (ANN) is a computing system that mimics biological Neural Networks that consist of animal brains. It is a collection of connected nodes called artificial neurons. Similar to biological neurons, an artificial neuron can receive a signal, processes it, and can signal other neurons connected to it. This signal in reality is a real number, that activates by a non-linear function of the sum of its inputs. The connection or edges are also adjusted with weights. These weights are the reason that the signal

varies differently in different situations. Also, nodes have a threshold limit. If the signal is stronger than the threshold, only then the signal transmits. It is also called perceptron.

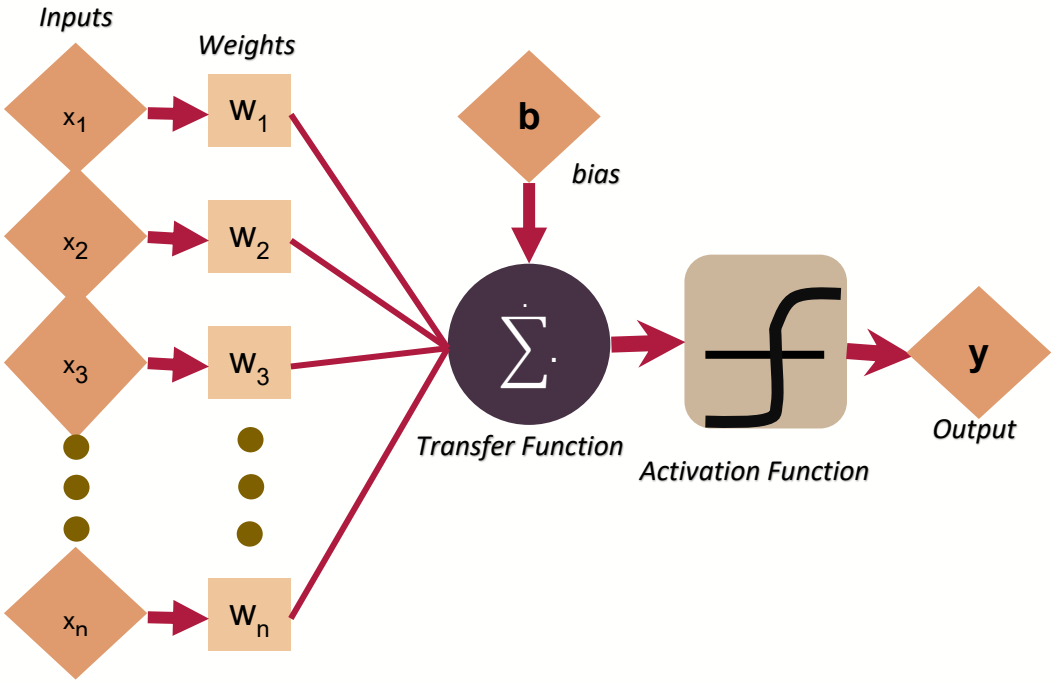

Figure 2.1: An artificial neural node or perceptron.

Equation of a perceptron:

$$y = f(x) = -b + \sum_{i=1}^{m} w_i x_i$$

In deep learning models, the multilayer model is followed. A layer is created with many perceptrons. If all the nodes of a layer are connected with all the nodes of the next layer, it is called a dense layer. There are three types of layers categorized into the input layer, the output layer, and the hidden layer. Normally hidden layers are created as dense layers. There can be as many hidden layers as needed. But increasing the layer does not always improve output results. In reality, it increases the training time. [4]

2.2.2 Convolutional Neural Networks (CNN)

A CNN model can be described with following structure:

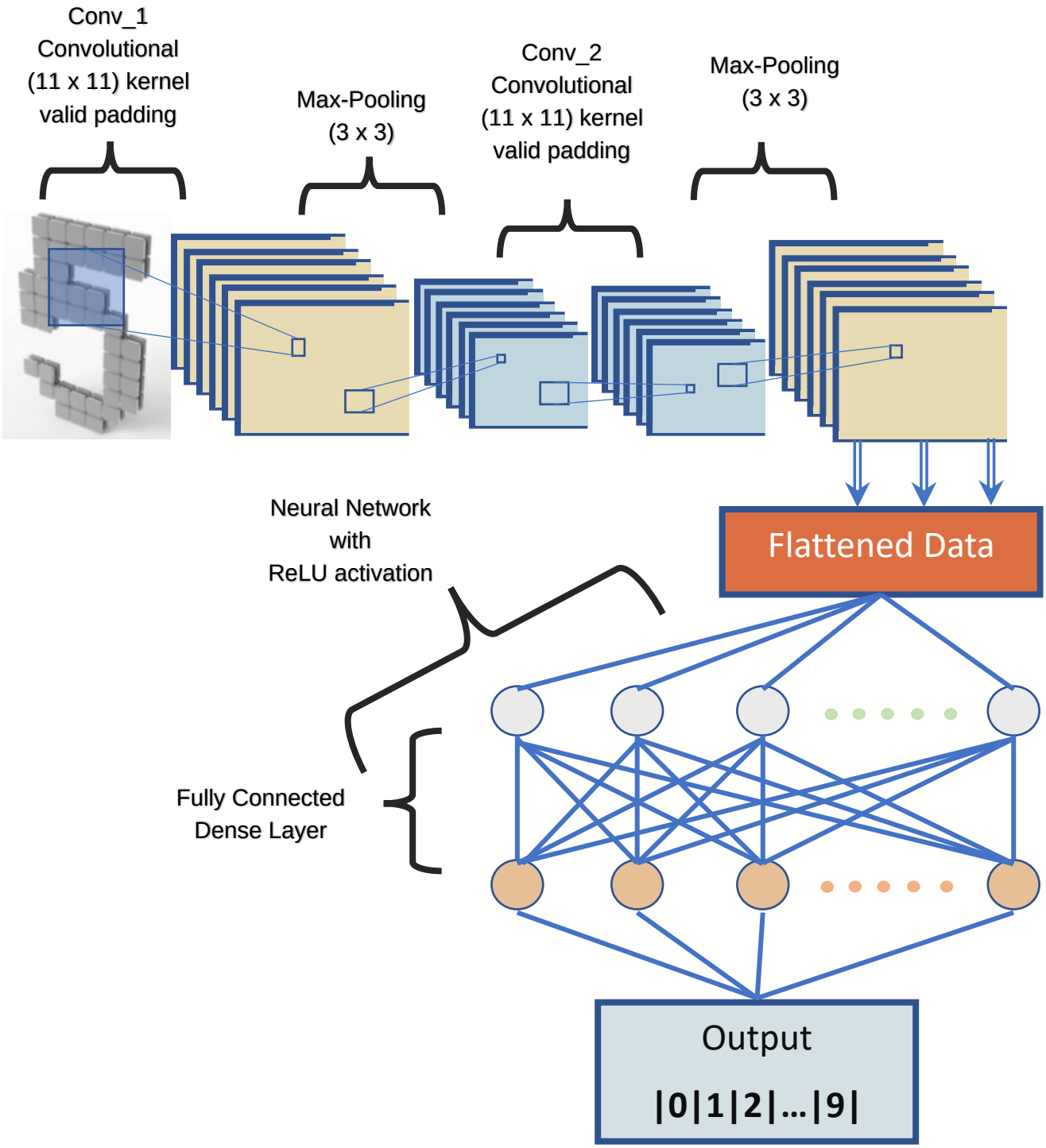

Figure 2.2: A CNN network model.

Convolutional Neural Networks (CNN) is a very popular deep learning tool and is famous for its ability to handle a large amount of data. CNN gives a good performance in pattern recognition. Although it is mostly used in computer vision-related fields, because of its pattern understanding ability and computation of a large amount of data, it can produce a fair result in time series related problems.

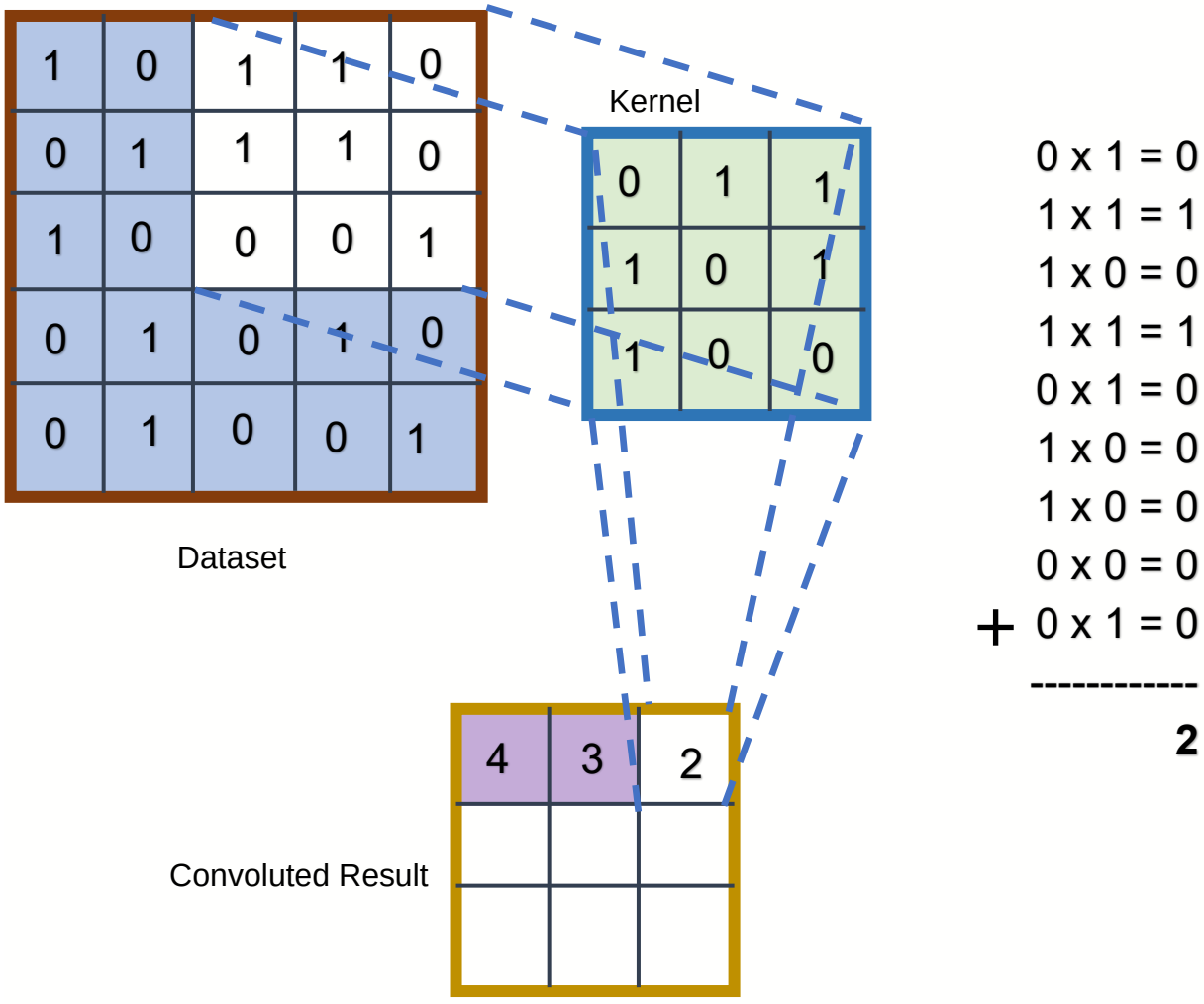

Figure 2.3: A matrix convolution calculation.

In a CNN network, a set of data is selected from the full data set. Which is called kernel size. The kernel is a fixed dataset for every iteration, generated to clutch the information from the main dataset. Then the kernel is convoluted

piece by piece with the main dataset by degrees of kernel size. On each batch, a fixed kernel takes the convolution result, creates a convolved feature, and passes it to the next layer.
There is an Activation Function Layer that applies an element-wise activation function to the output of the convolution layer. Some common activation functions are ReLU: max(0, x), Sigmoid: $1/(1+e^{-x})$, tanh, Leaky ReLU, etc. In this layer, the output size remains the same as the layer's input size.

After that, the pooling layer is used to compress data by reducing the spatial size of the convolved feature and removing noise and unwanted data. If the pooling value is increased, the compression gets stronger. Thus, CNN is used widely for pattern recognition. [5]

2.2.2 Long Short-Term Memory (LSTM)

Long Short-Term Memory is a sequential deep learning tool derived from the RNN tool. It preserves information and is capable of handling the vanishing gradient problem. A recurrent neural network is also known as RNN is the first algorithm that remembers its input in internal memory. They were introduced by Hochreiter & Schmidhuber in 1997.

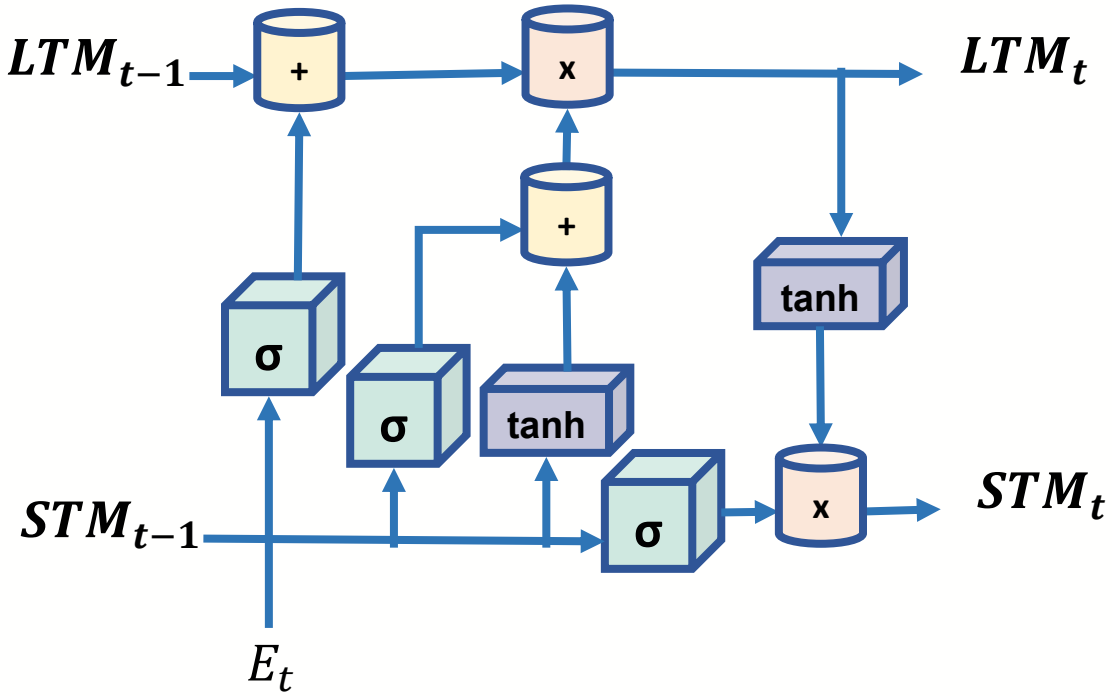

Figure 2.4: LSTM architecture.

LSTM network uses both Long-Term Memory (LTM) and Short-Term Memory (STM) and uses four types of gates for ease of calculation.

Forget Gate: Here data is reduced by removing unwanted and excess data. It takes Previous Long-Term Memory (LTM_{t-1}) as input and decides on which information should be remembered. Previous Short-Term Memory (STM_{t-1}) and Current Event vector (E_t) are combined together [STM_{t-1}, E_t] and multiplied with the weight matrix (W_f) and passed through the Sigmoid activation function with some bias and form Forget Factor (f_t). After that forget Factor (f_t) is multiplied with the Previous Long-Term Memory (LTM_{t-1}) to produce forget gate output.

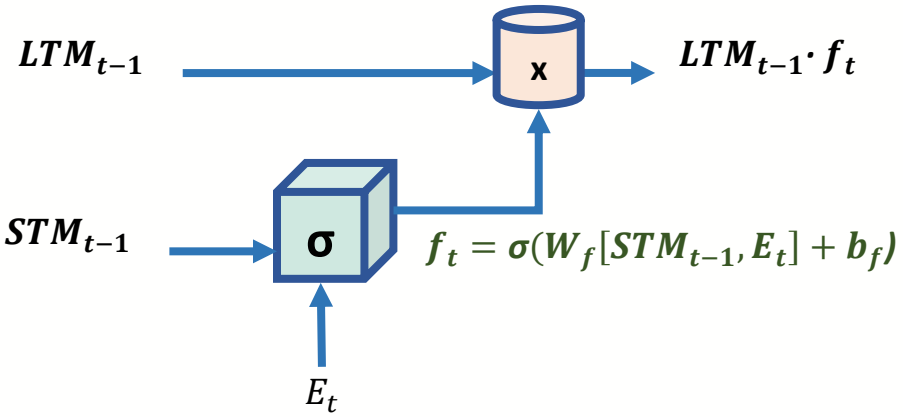

Figure 2.5: Forget gate.

Learn Gate: With current input of a node and STM, necessary information that had recently learned from STM can be applied to the current input. It takes the current input or event (E_t) and previously collected short-term memory (STM_{t-1}) as input. Previous Short-Term Memory (STM_{t-1}) and Current Event vector E_t together creates [STM_{t-1}, E_t] and multiplied with the weight matrix (W_n) with bias passed to tanh function to introduce non-linearity to it, and after that creates a matrix (N_t). For ignoring insignificant information, we calculate one Ignore Factor (i_t). To calculate ignoring factor, Short Term Memory (STM_{t-1}) and Current Event vector (E_t) and multiply with weight matrix (W_i) and passed through Sigmoid activation function with some bias. Learn Matrix (N_t) and Ignore Factor (i_t) is multiplied together makes learn gate result.

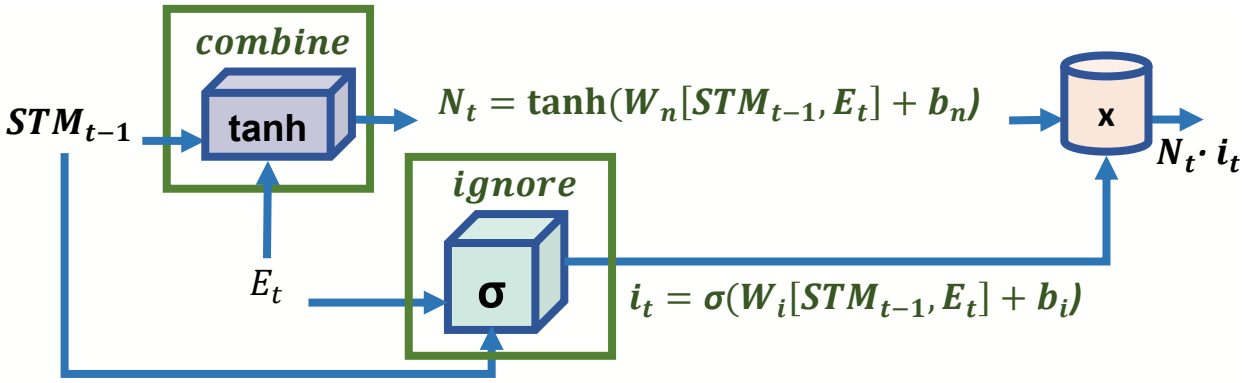

Figure 2.6: Learn gate.

Remember Gate: Here most important data is stored persistently as LTM and filters all STMs that are very important to be remembered. This gate works as LTM and constantly updates with every new output. Previous Short-Term Memory (STM_{t-1}) and Current Event (E_t) are combined to produce output. The output of Forget Gate and Learn Gate are added together to produce an output of Remember Gate which take place as the LTM for the next cell.

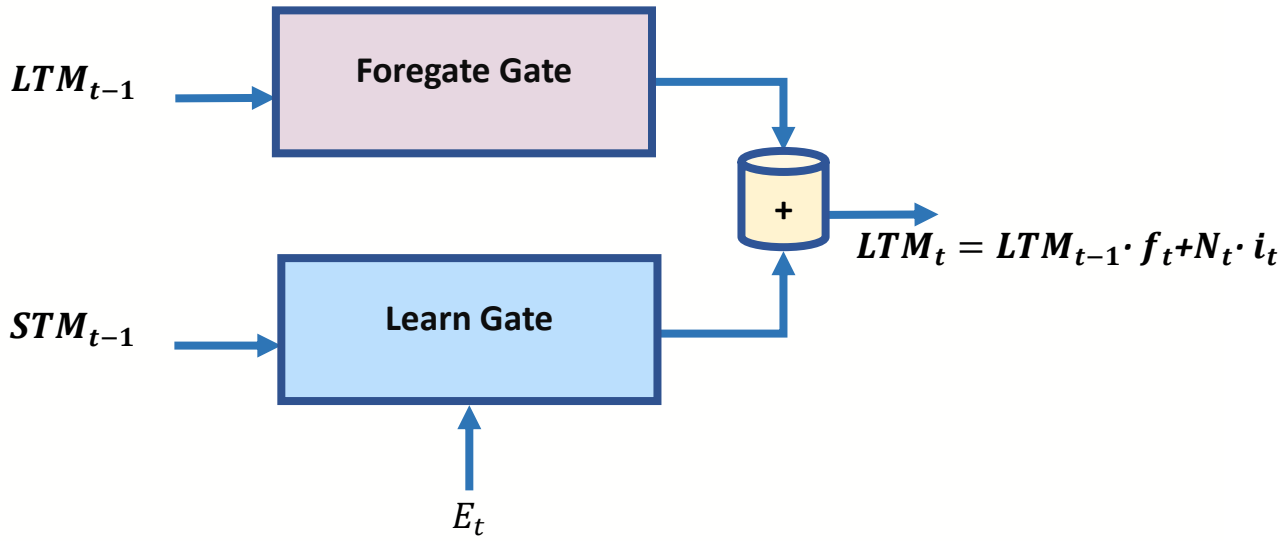

Figure 2.7: Remember gate.

Use Gate: This gate works with LTM, STM, and current input to predict the output of the current event and works as an updated STM. Here is important

information from Previous Long-Term Memory and Previous Short-Term Memory mixed to create STM for the next cell and produce output for the current event. Previous Long-Term Memory (LTM_{t-1}) moves through the Tangent activation function with some bias for producing U_t. Previous Short-Term Memory (STM_{t-1}) and Current Event (E_t) are combined together and proceed through the Sigmoid activation function with some bias to produce V_t. Output U_t and V_t are then multiplied together to produce the output of the use gate. And thus, STM is updated for the next cell.

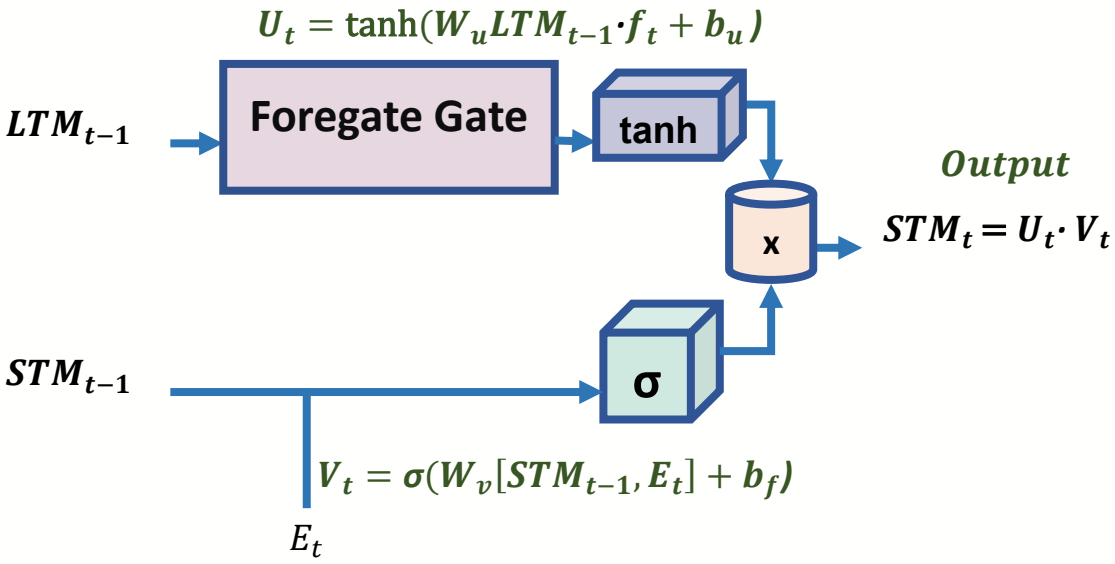

Figure 2.8: Use gate.

For LSTM's long-term dependencies. it is widely used in Natural Language Processing, Voice Recognition, Image OCR Models, etc. It is also good for time series analysis for its stored long-term memory. LSTM does not have a problem where weights become so small that it underfits the model. But sometimes, for large weights, it overfits the model. [6]

Chapter 3

Methodology and Experiments

In this chapter, the setup for the experiment is explained from top to bottom. All the steps are described in a detailed manner. The implementation of the system is shown here. The experiment outcomes are also been demonstrated.

3.1 Tools Used for Experiments

3.1.1 Data Collection

For the stock market dataset, we chose only stock price data from Dhaka Stock Exchange's top ten performed companies. Each geographical stock exchange varies from one another. For availability from our project location and needs for advice from professionals, we chose the place that will give us the most resources. To collect data, we used the internet. And collected data for the including the date, the opening price, the highest price, the lowest price, the closing price, and the volume of stock. To work on the project, we chose only one company at a time. For the regular performing period, we took data from 1999 to 2018. We used one-year data from 2018 for testing. And for crisis period performance, we collected data from 1999 to 2021. And to test our models in the crisis period, we chose data for the entire COVID-19 pandemic period, from 2020 to 2021. [7]

3.1.2 Computing Platform

We used Google Colaboratory or “Colab” for our experiments. We choose Colab because of its swiftness, robustness, and interactive programming environment. Since 2017, Colab is free for public use. Training deep neural networks requires a good performance of GPU is needed. But with the Colab facility, high performing Tensor Procession Unit (TPU) is also free to use. This gives us great results compared to any other hardware. And we can do

multiple experiments at the same time using Colab without any extra resources. We used another free feature of Google to store data in the cloud. We used Google Colab's previously installed libraries for all the experiments. We also used Colab so that we can share our experiment with anyone in real-time.

3.1.3 Experimental Setup

As we have said, we used Colab for doing all the experiments. We needed a small setup for our experiment. To set up the environment, a Google account was required. We created a Google account with an IEEE student account. We allocated a specific part in the drive space. Hence, we installed Colab in the account and created a notebook. We did not set setup runtime by using GPU or TPU. Standard runtime was used for all the experiments. For our project, we did not need any other dependency to install in our notebook. We used all the libraries from the Colab server.

3.1.4 Libraries

Some libraries helped a lot for the completion of our project. A small description is given bellow:

Tensorflow: It is an end-to-end free open-source platform for machine learning and artificial intelligence. It can be used for many purposes alongside data-related works such as training and inference of deep neural networks. Tensorflow's name came from the mathematical object name Tensor. A tensor is a mathematical object to describes a multilinear relationship with other mathematical objects in a vector space. Tensorflow works very well with tensors.

Pandas: Pandas is free software written especially for Python programming language. It is mainly used for data manipulation and analysis. The name came to form two words, 'panel' and 'data'. Data reshaping, indexing, labeling, integrating, etc. can be easily done by Pandas.

NumPy: The full form of NumPy stands for Numerical Python. It is an open-source software created for large multidimensional arrays and matrixes.

NumPy can be used for any type of numerical data with an N-dimensional array. It also works well with matrices.

Keras: It is an open-source software library for the Python interface to solve artificial neural network problems. It is used for building neural network blocks such as layers, activation functions, optimizers, etc. Simple, consistent, and extensible API. Keras is popular for its minimal structure, running capability on both CPU and GPU, and for high scalability of computation.

Statsmodels: It is a Python module providing classes and functions for the estimation of many different statistical models. This open-source API is also used for testing data. This library tests result by comparing existing statistical packages to ensure their correctness.

Sklearn: This library gives us tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction, feature extraction, parameter tuning in Python. It gives us algorithms for both supervised and unsupervised learning. It is also used for data normalization.

Random: For machine learning implementation, we need random numbers to train models. This library creates pseudo-random numbers to create any type of numerical data. It generates numbers in uniform, Gaussian, lognormal, negative exponential, gamma, and beta distributions.

Matplotlib: It is a library that focuses on data visualization. It plots all kinds of data such as histograms, scatter plots, 3D plots, polar plots, line plots, image plots, contour plots, etc.

3.2 Proposed System

The stock market is always unpredictable. So, using only one deep learning model cannot perform best by only using previous data. To predict the future values from two popular models, LSTM & CNN models are better options for prediction. An LSTM network stores data from the previous layer. Predicting future data requires knowledge about previous price shifting. Similarly, the

CNN network performs proficiently in pattern recognition. Thus, the CNN model has the ability to predict the price trend easily.

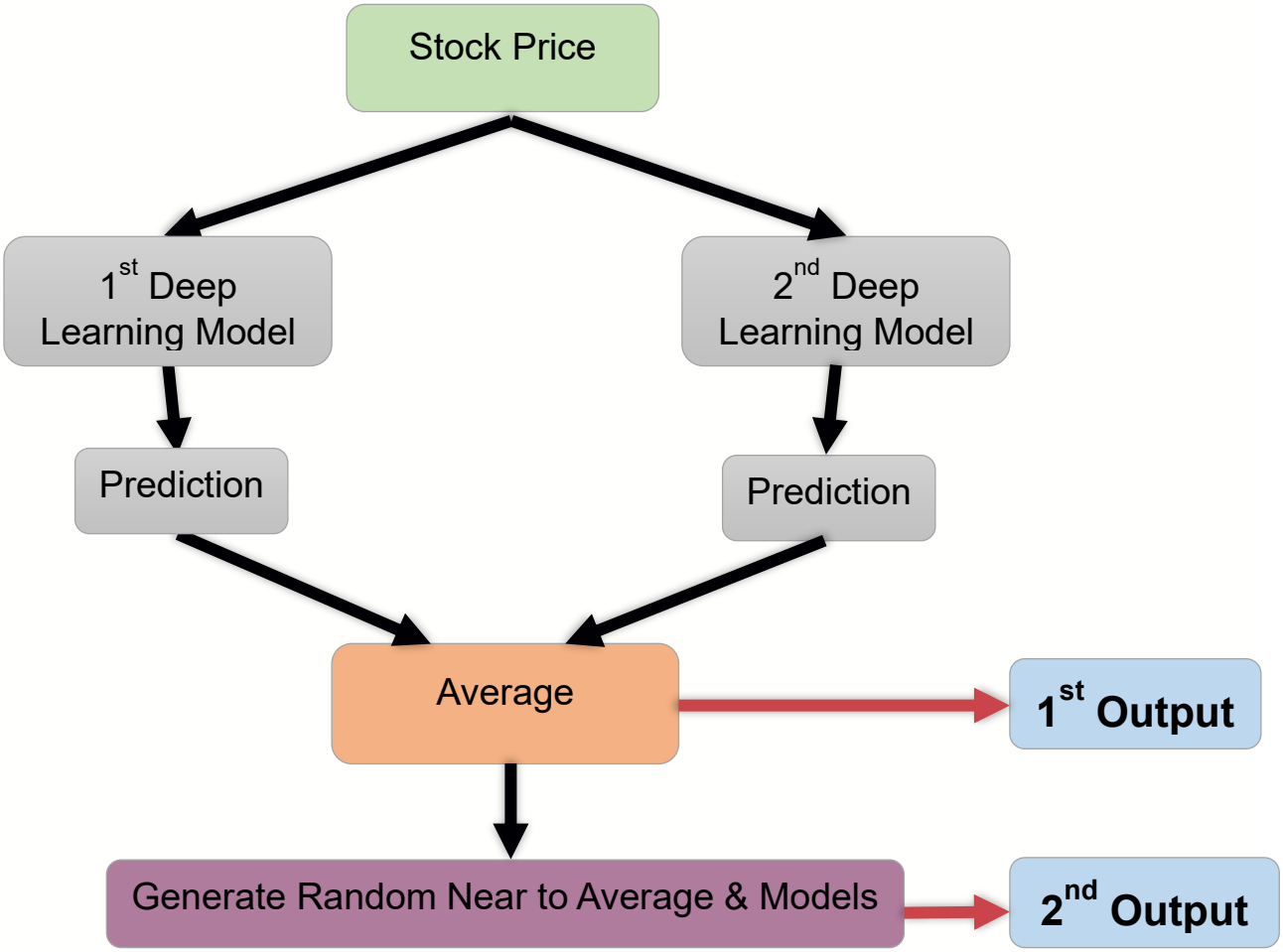

Figure 3.1: Proposed system architecture.

Both CNN and LSTM have the potential to predict stock prices accurately. But CNN requires a lot of data to predict future value. So, predicting new companies stock prices cannot be done using CNN. Also, LSTM does have Exploding Gradient problem, where weights become too large that it overfits the model. We use the average value between the predicted value from LSTM & CNN, so that accuracy becomes high.

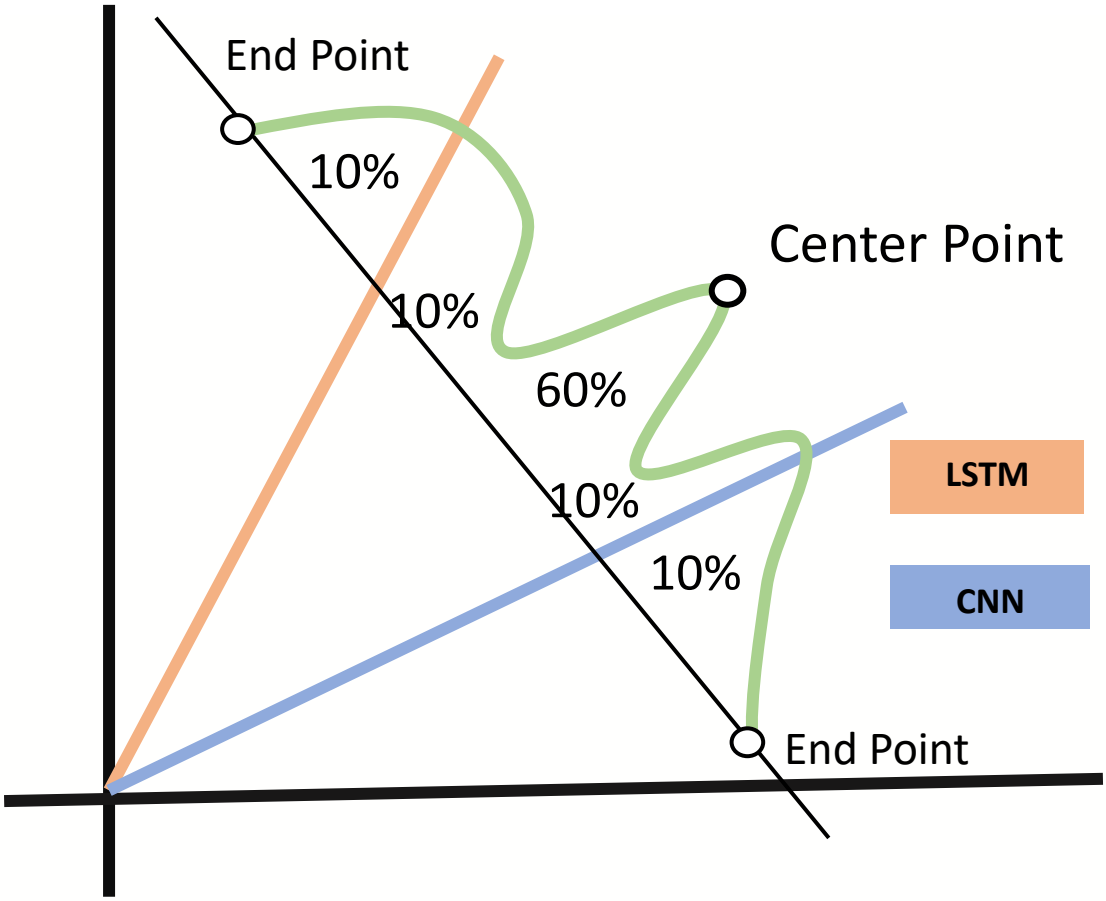

Figure 3.2: Choosing random values between two models.

Also, stock price varies randomly. So, if randomness occurs within the predicted range, the biasness of the LSTM & CNN will be canceled. The random value cannot be an accurate predicted price, but the random value will be closed to the actual future price.

3.3 Training the Deep Learning Models

For data prediction in both models, we followed some steps.

Input: Historical stock price
Output: Prediction of stock price and trends

Step 1: Dataset collection and reading previous data
Step 2: Taking open price for training in 2D array
Step 3: Feature scaling the data so that the data values vary from 0 to 1
Step 4: Feature selection with 90 timestamp and 1 output node
Step 5: Training the deep learning model using LSTM or CNN
Step 6: Compiling with Adam optimizer and loss as Mean Log Squared Error
Step 7: Taking open price for prediction
Step 8: Making the prediction
Step 9: Visualizing the result using plotting technique

3.3.1 CNN Model Data Prediction

The summery of the network is as the following:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv1d (Conv1D)              (None, 80, 64)            768

 max_pooling1d (MaxPooling1D)  (None, 80, 64)           0

 conv1d_1 (Conv1D)            (None, 70, 128)           90240

 max_pooling1d_1 (MaxPooling1D)  (None, 70, 128)        0

 conv1d_2 (Conv1D)            (None, 60, 128)           180352

 max_pooling1d_2 (MaxPooling1D)  (None, 60, 128)        0

 conv1d_3 (Conv1D)            (None, 50, 128)           180352

 flatten (Flatten)            (None, 6400)              0

 dense (Dense)                (None, 128)               819328

 module_wrapper (ModuleWrapper)  (None, 1)              129

=================================================================
Total params: 1,271,169
Trainable params: 1,271,169
Non-trainable params: 0
_________________________________________________________________
```

We used, 1D CNN network, with

kernel size=11,
time duration=90 days,
activation function=ReLU,

In compiler, we used,
optimizer='adam',
loss='mean_squared_logarithmic_error',
metrics='kullback_leibler_divergence',

In model fitting session,
batch size=30,
epochs=200,

Everything else as default parameters.

Plotting output of the result

Figure 3.3: Real price vs CNN network predicted price.

3.3.2 LSTM Model Data Prediction

The summery of the network is as the following:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 90, 50)            10400
_________________________________________________________________
dropout (Dropout)            (None, 90, 50)            0
_________________________________________________________________
lstm_1 (LSTM)                (None, 90, 100)           60400
_________________________________________________________________
dropout_1 (Dropout)          (None, 90, 100)           0
_________________________________________________________________
lstm_2 (LSTM)                (None, 90, 100)           80400
_________________________________________________________________
dropout_2 (Dropout)          (None, 90, 100)           0
_________________________________________________________________
lstm_3 (LSTM)                (None, 90, 100)           80400
_________________________________________________________________
dropout_3 (Dropout)          (None, 90, 100)           0
_________________________________________________________________
lstm_4 (LSTM)                (None, 100)               80400
_________________________________________________________________
dropout_4 (Dropout)          (None, 100)               0
_________________________________________________________________
dense (Dense)                (None, 1)                 101
=================================================================
Total params: 312,101
Trainable params: 312,101
Non-trainable params: 0
_________________________________________________________________
```

We used, 1D CNN network, with
Dropout=0.2,
Time duration=90 days,

In compiler, we used,
optimizer='adam',
loss='mean_squared_logarithmic_error',
metrics='kullback_leibler_divergence',

In model fitting session,
batch size=30,

epochs=200,

Everything else as default parameters.

Plotting output of the result

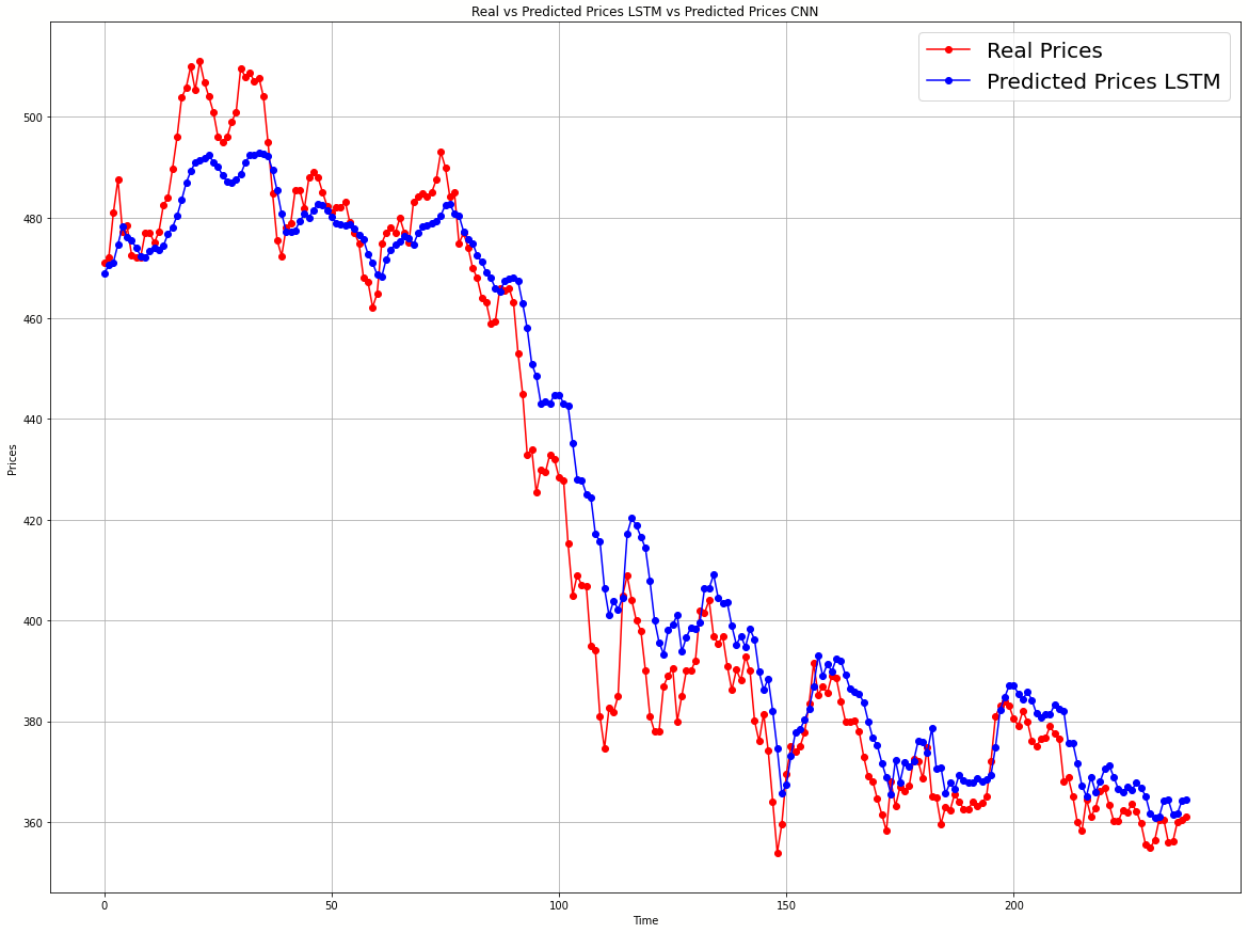

Figure 3.4: Real price vs LSTM network predicted price.

3.4 Further experiments

For further data prediction we got the average value of both models, and created random value beside them.

3.4.1 Average Between CNN and LSTM Prediction

To take average value, we took both predicted values that came from CNN & LSTM experiments. We took mean of the two value.

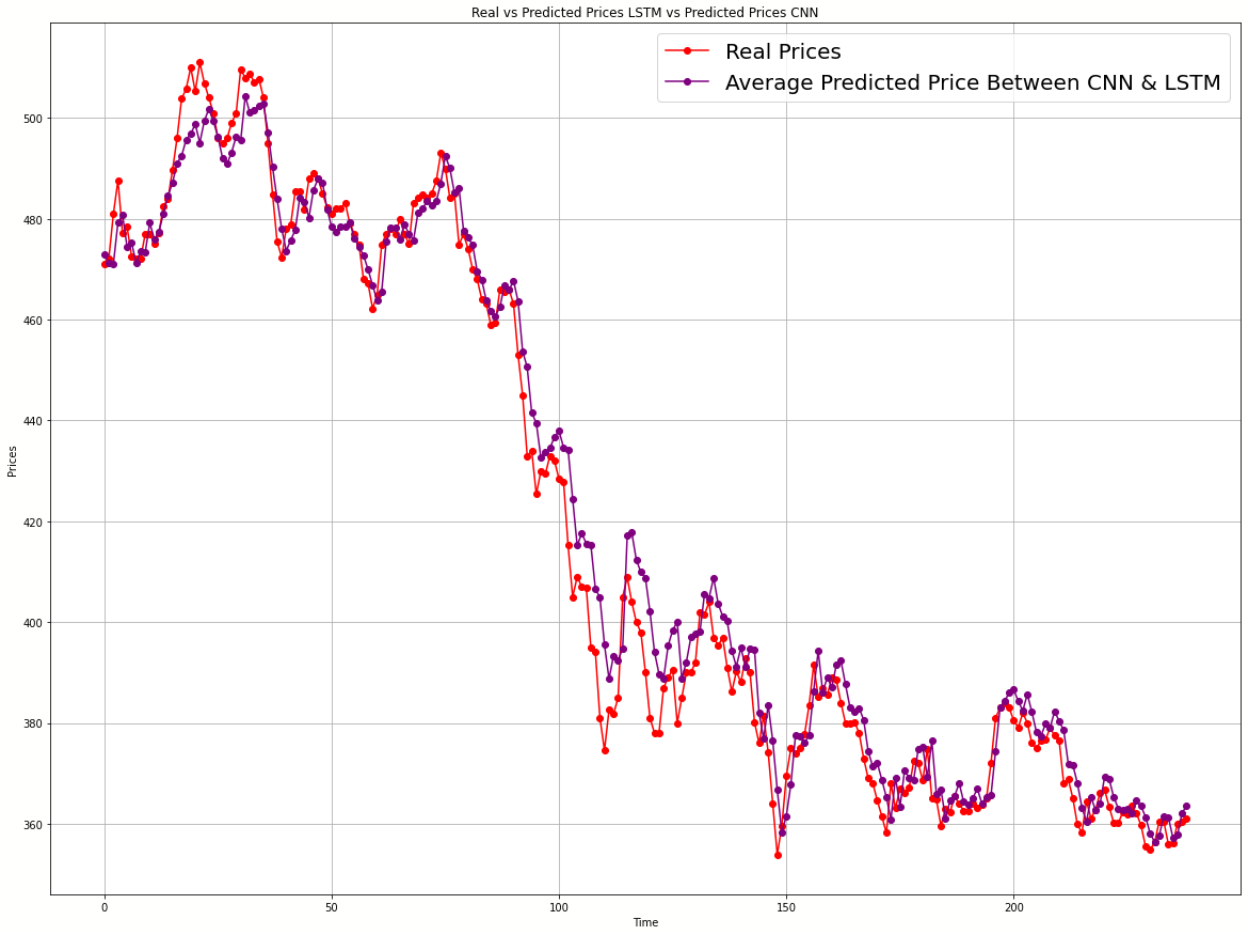

Figure 3.5: Real price vs average price between CNN & LSTM prediction.

3.4.2 Random Value Between CNN and LSTM Prediction

To take random values between prices of LSTM and CNN, we created a special probability of random values.

Firstly, we took the subtraction value of CNN & LSTM predicted price.

Secondly, we multiplied the subtracted value by 0.25 and added it with the average value of CNN & LSTM to find one location. We also subtracted the

0.25 multiplied value from the average value of CNN & LSTM to determine the location of another point.

$$location\ shifter = abs(LSTM - CNN) * 0.25$$

Thirdly, we generated random values between two locations.

$$random\ value = random\ [\ average\ of\ LSTM\ \&\ CNN - location\ shifter,\ average\ of\ LSTM\ \&\ CNN + location\ shifter]$$

Fourthly, we created a similar situation near CNN & LSTM values.

Fifthly, we created a function, so that the value near average will be 60%. And 20% random values will be generated near LSTM & CNN each.

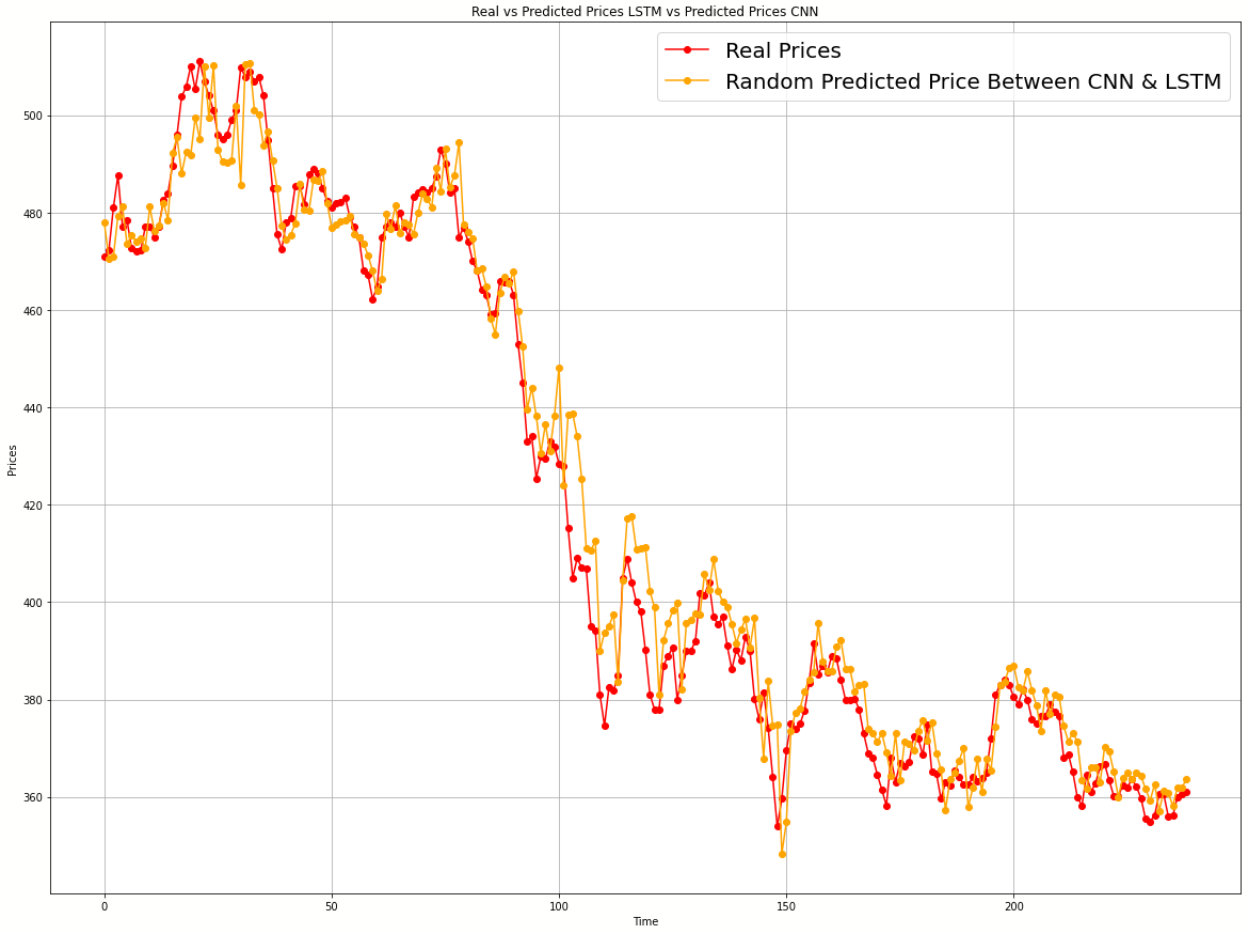

Figure 3.6: Real price vs random price between CNN & LSTM prediction.

3.5 Performance During a Financial Crisis

We took prediction data during a crisis period time such as the COVID-19 pandemic time. All the experimenters were the same, but the prediction data was tested only for the crisis period and was trained with normal situation data.

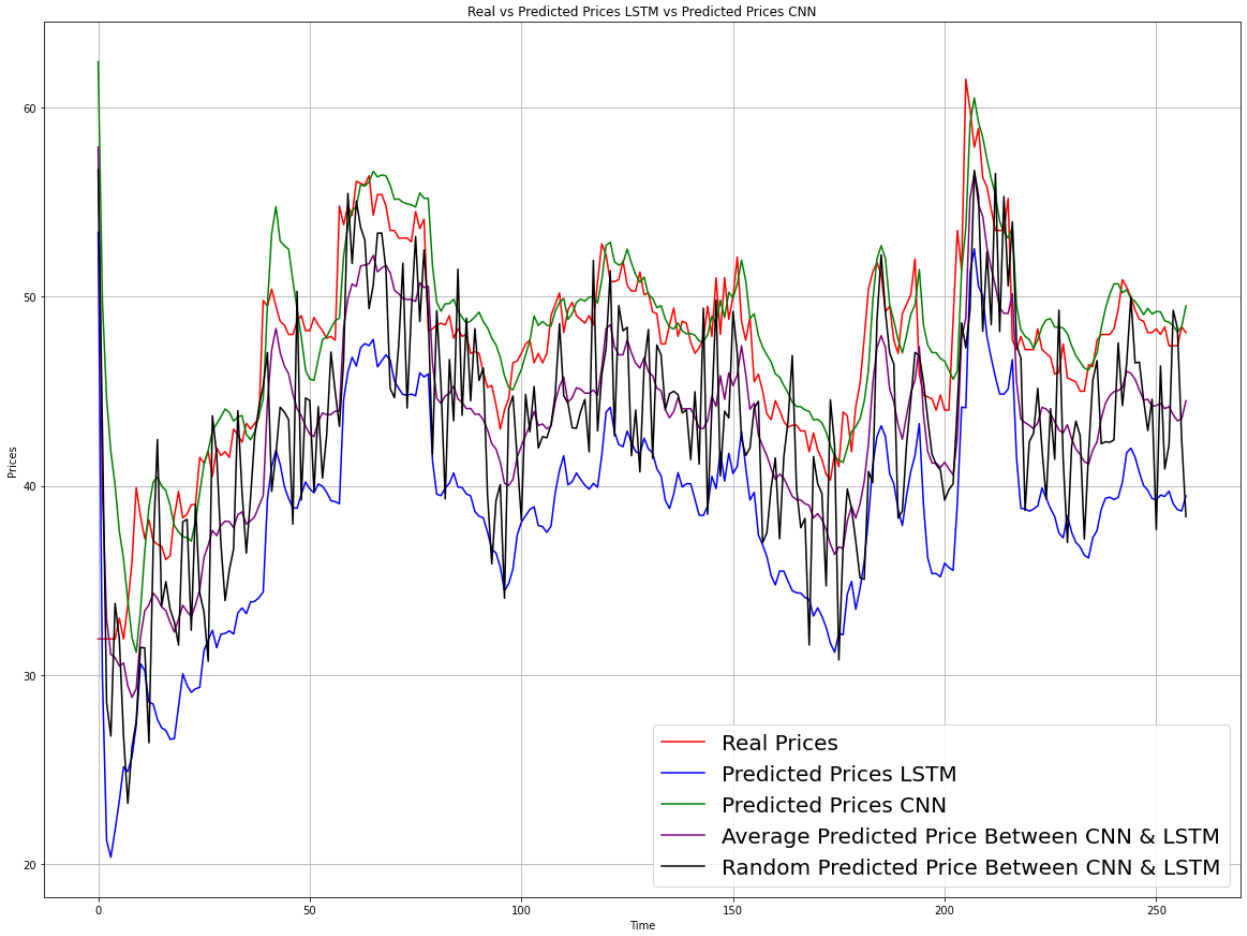

Figure 3.7: Real price vs LSTM prediction vs CNN prediction vs average price between CNN & LSTM prediction vs Random price between CNN & LSTM prediction in crisis (COVID-19 pandemic) situation.

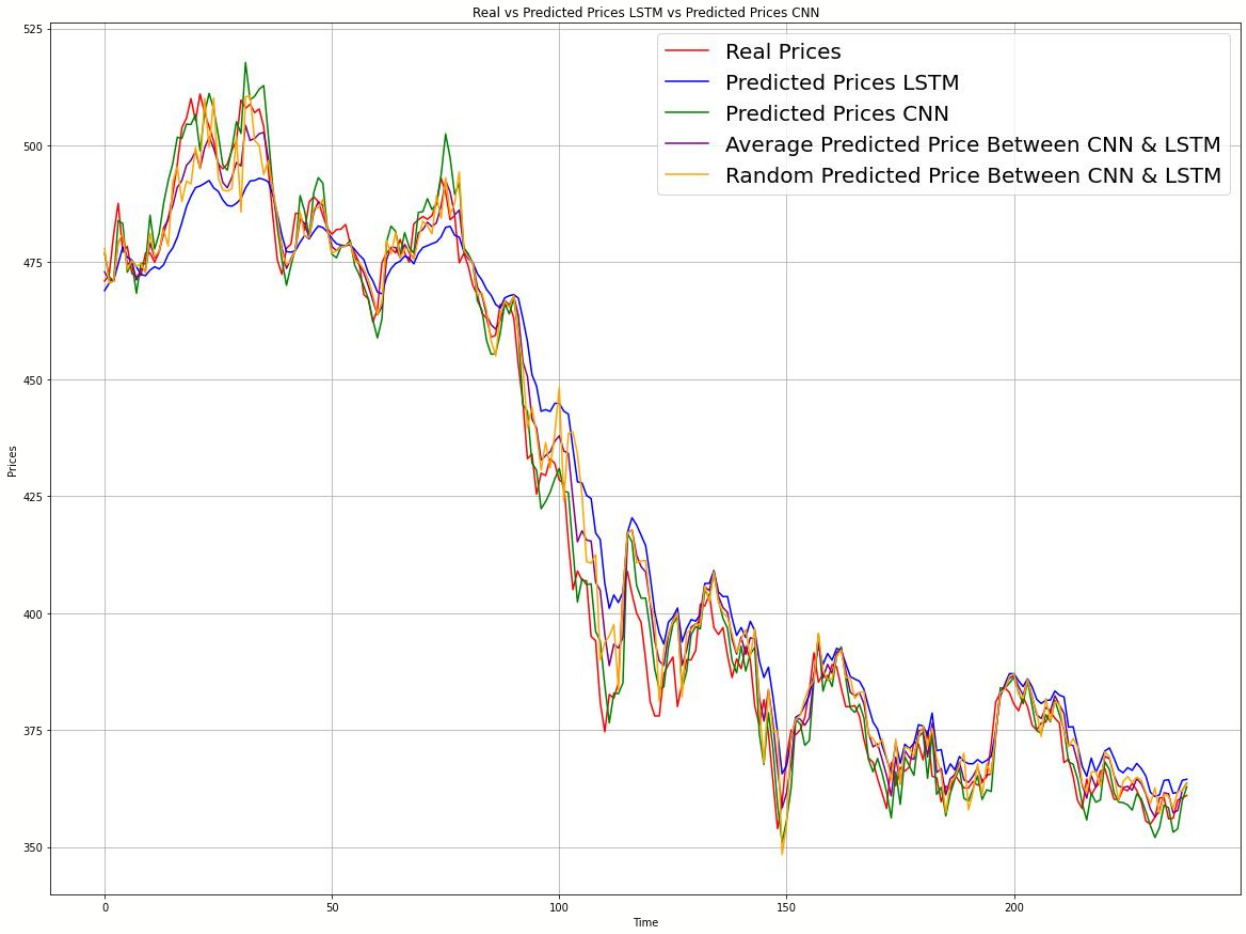

Figure 3.8: Real price vs LSTM prediction vs CNN prediction vs average price between CNN & LSTM prediction vs Random price between CNN & LSTM prediction in regular situation.

Chapter 4

Experimental Results

In this chapter, project evaluation is done. project-related validation is done in this chapter. Here the result of the experiments is compared with one another.

4.1 Results of The Experiments

Here we are showing some statistical results of our findings.

4.1.1 Coefficient of Determination (R2) Value

Table 4.1: Measurements of R^2 values.

Experiment no.	CNN Model	LSTM Model	Random between CNN & LSTM value	Average between CNN & LSTM value
1.	0.9872	0.9592	0.9769	0.9826
2.	0.9883	0.9636	0.9815	0.9817

4.1.2 Mean Absolute Error

Table 4.2: Measurements of Mean Absolute Error.

Experiment no.	CNN Model	LSTM Model	Random between CNN & LSTM value	Average between CNN & LSTM value
1.	4.3783	6.8616	5.3125	5.1863
2.	4.6897	8.2954	5.9063	5.2550

4.1.3 Mean Square Error

Table 4.3: Measurements of Mean Square Error.

Experiment no.	CNN Model	LSTM Model	Random between CNN & LSTM value	Average between CNN & LSTM value
1.	35.7099	114.0878	64.4751	48.7156
2.	32.5964	101.8041	51.7497	50.9805

4.1.4 Root Mean Square Error

Table 4.4: Measurements of Root Mean Square Error.

Experiment no.	CNN Model	LSTM Model	Random between CNN & LSTM value	Average between CNN & LSTM value
1.	5.7093	10.0898	7.19372	7.1400
2.	5.9757	10.6811	8.0296	6.9796

4.1.3 Standard Error of the Estimate

Table 4.5: Measurements of Standard Error of the Estimate.

Experiment no.	CNN Model	LSTM Model	Random between CNN & LSTM value	Average between CNN & LSTM value
1.	6.0009	10.7261	8.0634	7.0090
2.	5.7333	10.1322	7.2240	7.1701

4.1.3 Average Dispersion

Table 4.6: Measurements of Average Dispersion.

Experiment no.	CNN Model	LSTM Model	Random between CNN & LSTM value	Average between CNN & LSTM value
1.	4.6897	8.2954	5.9063	5.2550
2.	3.7035	6.7285	5.4396	4.5935

4.2 Results in a Crisis Scenario

Here we are showing some statistical results of our findings.

4.2.1 Coefficient of Determination (R2) Value

Table 4.7: Measurements of R^2 values on crisis situation.

Experiment no.	CNN Model	LSTM Model	Random between CNN & LSTM value	Average between CNN & LSTM value
1.	0.6542	-12.7487	-4.1646	-2.0513
2.	0.5523	-1.7733	-0.1452	0.2646

4.2.2 Mean Absolute Error

Table 4.8: Measurements of Mean Absolute Error on crisis situation.

Experiment no.	CNN Model	LSTM Model	Random between CNN & LSTM value	Average between CNN & LSTM value
1.	2.1378	8.3977	4.4900	3.8473
2.	1.9964	19.1965	10.1404	8.8986

4.2.3 Mean Square Error

Table 4.9: Measurements of Mean Square Error on crisis situation.

Experiment no.	CNN Model	LSTM Model	Random between CNN & LSTM value	Average between CNN & LSTM value
1.	9.3459	371.6934	139.6258	82.4921
2.	12.1028	74.9776	30.9611	19.8812

4.2.4 Root Mean Square Error

Table 4.10: Measurements of Root Mean Square Error on crisis situation.

Experiment no.	CNN Model	LSTM Model	Random between CNN & LSTM value	Average between CNN & LSTM value
1.	3.4789	8.6589	5.5642	4.4588
2.	3.0571	19.2793	11.8163	9.0825

4.2.3 Standard Error of the Estimate

Table 4.11: Measurements of Standard Error of the Estimate on crisis situation.

Experiment no.	CNN Model	LSTM Model	Random between CNN & LSTM value	Average between CNN & LSTM value
1.	3.4924	8.6927	5.5859	4.4762
2.	3.0690	19.3545	11.8624	9.1179

4.2.3 Average Dispersion

Table 4.12: Measurements of Average Dispersion on crisis situation.

Experiment no.	CNN Model	LSTM Model	Random between CNN & LSTM value	Average between CNN & LSTM value
1.	1.9964	19.1965	10.1404	8.8986
2.	2.1378	8.3977	4.4900	3.8473

4.3 Evaluation of ANN Models and Predicted Data

For evaluation process, we used build in evaluating function, and the result we get as follows:

In CNN Model:
test loss = 8.0702e-05,
kullback_leibler_divergence [Matrix] = 1.7760e-04,

In LSTM Model:
test loss = 2.4661e-04,
kullback_leibler_divergence [Matrix] = -0.0116,
From Statsmodels API library, we got, for LSTM Model,

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.974
Model:                            OLS   Adj. R-squared:                  0.974
Method:                 Least Squares   F-statistic:                     8936.
Date:                Fri, 08 Apr 2022   Prob (F-statistic):          3.71e-190
Time:                        19:43:11   Log-Likelihood:                -850.67
No. Observations:                 239   AIC:                             1705.
Df Residuals:                     237   BIC:                             1712.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
```

const	-20.2604	4.689	-4.321	0.000	-29.498	-23
x1	1.0610	0.011	94.531	0.000	1.039	1.083

Omnibus:	25.947	Durbin-Watson:	0.421
Prob(Omnibus):	0.000	Jarque-Bera (JB):	34.300
Skew:	-0.727	Prob(JB):	3.56e-08
Kurtosis:	4.154	Cond. No.	3.55e+03

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.55e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

For CNN Model,

OLS Regression Results

Dep. Variable:	y	R-squared:	0.990
Model:	OLS	Adj. R-squared:	0.990
Method:	Least Squares	F-statistic:	2.434e+04
Date:	Fri, 08 Apr 2022	Prob (F-statistic):	7.00e-241
Time:	19:43:11	Log-Likelihood:	-732.89
No. Observations:	239	AIC:	1470.
Df Residuals:	237	BIC:	1477.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	.975]
const	7.0322	2.668	2.636	0.009	1.776	12.288
x1	0.9839	0.006	156.015	0.000	0.972	0.996

Omnibus:	6.246	Durbin-Watson:	1.569
Prob(Omnibus):	0.044	Jarque-Bera (JB):	9.041
Skew:	0.112	Prob(JB):	0.0109
Kurtosis:	3.926	Cond. No.	3.35e+03

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.35e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

Chapter 5

Discussion

In this chapter, project discussion, we discussed about project-related problems and opportunities. Opinions on experiment result is included in here.

5.1 Findings of The Experiments

From all the findings of the experiments, there is clear evidence for time-shifted correlations between the price behavior of Dhaka Stock Exchange stocks price in contradiction to both the random walk hypothesis and the efficient-market hypothesis. By using deep-layered neural networks for trend prediction in intercorrelated time series. The one-day prediction of data tells that the accuracy of our experiment is very close to the actual value. Where the root mean square (RMSE) value is nearly 7.30. And the R^2 value is nearly 97%. Thus, as long as the training data is enough, time-series price prediction can be done. This shows that even if the price cannot be predicted accurately, the trend that occurs in the stock market can be predicted with deep learning models. But in the time of crisis or special cases, the stock price cannot be predicted accurately. In conclusion, the results of this project regarding the stock market future price prediction are positive under conscientious observance of statistical validation measures.

5.2 Findings of The Crisis

In a crisis period, the outcomes of the experiments tell us that even if the training data set is huge, crisis situation price cannot be predicted using this model. However, both LSTM and CNN models predicted results that occur with more than 60% in R^2. This shows that, even in a risky situation, such as a crisis event, knowing about data trending is somehow possible. But, with enough dataset of normal regular historical data, Prediction will never be accurate. We saw the root mean squared (RMS) value to be more than 9.80. It can be just a chance that our model sometimes predicts crisis situations fairly well. But even then, the difference between the CNN value and LSTM value gives us a large distance.

Chapter 6

Conclusion

In this chapter, at the end, a summery and future planning are discussed. Also, how the model can perform better is also discussed.

6.1 Summary of The Findings and Experiments

Whether it is a normal period or a crisis event, some trends in the stock market are always predictable. But, as the long distant future prediction is calculated, we got very incorrect data. In our experiment, we solely tried to predict the next day's price and the trends of the market situation. From that, if there is no external issue occurring in the market, prediction can be done quite accurately in any regular situation. Although it contradicts the hypothesis of efficient market hypothesis and random walk hypothesis, some predictions can be done by time-series data. The more the data is used for the training, the better the result gets. In this project, it was also amazing to learn that, within LSTM & CNN model, there is no one model which beats another model in prediction. Both models predict future values quite similarly in all situations. However, we saw that the average of the two models gives us even more astonishing results. So, allying two deep learning models actually creates a higher chance for the prediction to be accurate. However, we also saw, that random price generation performs better than LSTM or CNN model, but it never performs better than the average value. So, in the end, we can say, predicting stock price with time-series analysis with help of two or more deep learning models can give us fairly better results.

6.2 Future Plan for Further Research

For future work, we want the prediction of distant future values to be accurate as possible. We saw that even with enough dataset, we cannot predict 100% accurately, but the trending of stock fluctuations is always possible. So, by collecting more information about the market, the trend prediction can be more accurate. We also say that random prediction actually performed lower than average values, so in future works, more deep learning models may give more desirable results. We also worked with one company dataset at a time. There is the possibility that one company's price fluctuations can give an impact on another company. So, we can work on one inner connection of data.

Reference

[1] Eugene F. Fama, Efficient Capital Markets: A Review of Theory and Empirical Work, The Journal of Finance, Vol. 25, No. 2, Papers and Proceedings of the Twenty-Eighth Annual Meeting of the American Finance Association New York, N.Y. December, 28-30, 1969 (May, 1970), pp. 383-417 (35 pages)

[2] Benjamin Möws, “Deep Learning for Stock Market Prediction: Exploiting Time-Shifted Correlations of Stock Price Gradients”.

[3] Xu Lu and Tianzhong Zhao, September. Research on Time Series Data Prediction based on Clustering Algorithm - A Case Study of Yuebao. In 2017 AIP Conference Proceedings 1864.

[4] Arti Rana; Arvind Singh Rawat; Anchit Bijalwan; Himanshu Bahuguna, "Application of Multi Layer (Perceptron) Artificial Neural Network in the Diagnosis System: A Systematic Review", 2018 International Conference on Research in Intelligent and Computing in Engineering (RICE)

[5] Manav Mandal, " Introduction to Convolutional Neural Networks (CNN)," 2021 May. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/

[6] Christopher Olah, "Understanding LSTM Networks," 2015 August. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs

[7] DSE's EOD Data, "Previous Year Data from 1999-2020 (DMY Format)," [Online]. Available: https://dsecsv.simdif.com/previous-year.html