TFEX Prediction using Machine Learning Algorithms

Kongpob Siripichaiprom
Kornkitt Poolsup
Suradid Chao

Bachelor of Engineering in Software Engineering

International College

King Mongkut's Institute of Technology Ladkrabang

Academic Year 2017

KMITL-2018-IC-B-003-008

Thesis - Academic Year 2017

Bachelor of Engineer in Software Engineering
International College
King Mongkut's Institute of Technology Ladkrabang

Title: TFEX Prediction using Machine Learning Algorithms

Authors:

1. 57090007 Kongpob Siripichaiprom
2. 57090008 Kornkitt Poolsup
3. 57090035 Suradid Chao

Approved for submission

.....
Asst.Prof.Dr. Chaiwat Nuthong
Advisor
Date: 15/06/2018

Abstract

Predicting TFEX price's direction has always been interesting to both investors and researchers. There are various approaches that can be used to predict TFEX trend. One of the preferred approach is machine learning. Due to the ability to identify stock price's direction from large amounts of data. In this project, long short-term memory (LSTM) is a machine learning algorithm that is used on TFEX price's direction prediction in order to calculate price volatility and momentum for the stock. In order to increase accuracy on prediction, features are needed. There are five features contained in the original dataset, e.g., opened price, closed price, the lowest price, the highest price, and volume. The additional features apart from original dataset features are exponential moving average, elastic volume weighted moving average, on-balance volume, moving average convergence divergence, relative strength index, change in closing price, and percentage price in closing price. An aim is to develop a model that determines TFEX's price of next period compared to today. After all experiments, prediction model for both networks achieve more than 70 percent accuracy using hourly and daily basis as prediction period.

Acknowledgement

This project could not be completed without a very great advices from the advisor, Asst.Prof.Dr. Chaiwat Nuthong. Also this project could not get better result without advisor who has an experience in stock trading and stock prediction, and the one who provide TFEX data, Mr. Kittinu Muayteng. Thank you for knowledge, suggestion, and comments from Dr. Isara Anantavrasilp, Dr. Natthapong Jungteerapanich, and Assoc.Prof.Dr. Veera Boonjing. Including all my colleagues for the support and help.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Thailand Future Exchange (TFEX) is a derivative exchange which is a subsidiary of stock exchange. It is derived from one or more underlying assets which determine its value. Futures are highly leveraged investment, they suit people with limited budget. TFEX is different from the stock market in many ways, one of them is investors can make profit when the stock they hold rises in value. On the contrary, when the stock's value declines, their returns on investment become a loss. To turn their losses into profits, they have to wait until the stock's value rises again and sell these stock. In contrast, investors can invest their money on price's direction of the stock when it is going up or down in the future contract such as TFEX. It indicates the direction of the future graph. TFEX price's direction will be up or down as a result of the actual stock price's direction. This means investors must have confidence in their investment that the future trend will be in the direction they have predicted. This project focuses on TFEX prediction as a tool to assist investors in their decision.

TFEX price's direction can be extremely difficult to predict. It can move up or down due to several factors that may occur at that time such as political situation, business crisis, human's emotion, etc. These factors make the market change dramatically. In this project, the authors expected that price's direction in the past could be used to predict future price's direction.

In the past few years, several stock predictions approaches have been made to gain financial benefits from investment. However, from the work of T. Hellstr said that stock price was unpredictable because it followed the random walk hypothesis [15]. In addition, stock price is affected by multiple known and unknown factors, which make this topic of study to be more challenging. Many approaches has been applied including machine learning algorithm.

Price's prediction using machine learning is chosen by the authors. Machine learning is used to find the patterns embedded in large amount of data.

1.1 Problem Description

TFEX price's direction changes based on SET50, which is the price's direction of the first fiftieth index ranked by the large volume in Stock Exchange of Thailand (SET). This fluctuation in stock price is risky on investment. Although, investors can predict stock price's direction manually, it may take longer time.

To solve this problem, the software that aims to provide a quick and effective guidelines for investment is developed. The software first take in inputs such as opened price, closed price, volumes, highest price, lowest price, date, and times. The software compute these inputs and produces the output in the form of predicted price's direction. The price's direction can be either upward or downward.

1.2 Objectives

In order to indicate the success of the project, several goals have been set. The following list shows all the goals which are needed to be satisfied.

- Predict price's direction of TFEX SET50 futures using Artificial Neural Network (ANN) and Long Short-Term Memory (LSTM)
- Compare performance between ANN and LSTM
- Obtain at least 70% accuracy in prediction of price's direction. This prediction is performed in the chosen time frame using Long Short-Term Memory (LSTM).

1.3 Scope of Work

The proposed system is designed to work under the conditions shown in the following lists.

- Predict the future market price's direction from TFEX dataset with machine learning algorithms. There should be two results: up or down. The prediction periods are available as hourly, daily, and weekly basis.

- Features will be selected from seven features which are:
  - Exponential Moving Average (EMA)
  - Elastic Volume-Weighted Moving Average (eVWMA)
  - Moving Average Convergence Divergence (MACD)
  - On-Balance Volume (OBV)
  - Relative Strength Index (RSI)
  - Change in Closing Price
  - Percentage Change in Closing Price

1.4 Contributions

The expected contribution of this study is the ability to predict TFEX price direction of the next time period. This can be expanded to other stock market in Thailand. In particular, this study applies machine learning algorithms, i.e., ANN and LSTM, to TFEX market and compares performance of each machine learning algorithm. As a result, it is useful for considering machine learning algorithm to predict stock price's direction and comparing the performance of the selected machine learning algorithm with others.

Chapter 2

Literature Review

There are numerous studies on stock market prediction. This section mainly focuses on the related studies on stock prediction using machine learning algorithm.

Das and Padhy [13] used Back Propagation and Support Vector Machine (SVM) to predict the stock prices in the Indian stock market or National Stock Exchange (NSE) of India Limited. They used five input variables: four variables of lagged relative difference in percentage of price (RDP), and one variable of Exponential Moving Average (EMA). The result is SVM yields a higher prediction accuracy comparing to former ANN with Back Propagation technique with a smaller Normalized Mean Square Error (NMSE) and Mean Absolute Error (MAE) and larger Directional Symmetry (DS).

Kumbhare et al. [7] proposed system that used supervised learning algorithm such as SVM and Decision Tree (DT) to predict the stock price in the S&P 500 companies rather than the traditional Artificial Neural Network (ANN). Indicators such as Relative Strength Index (RSI), Exponential Moving Average (EMA), Commodity Channel Index (CCI), Collateralized Mortgage Obligation (CMO), Rate of Change (ROC), Average Directional Index (ADX), and Williams R% (W%R) were used in the system to improve the accuracy of the prediction. In most stock test cases scenario, the SVM outperformed with the accuracy of 87.4% compared to the DT with the accuracy of 81.10%.

Inthachot et al. [16] used ANN, and a combination of hybrid intelligence of ANN and Genetic Algorithm to forecast stock price movement in Thai stock SET50 index. The ANN model comprised of three-layered feed forward model. The model composed of an input layer, a hidden layer, and an

output layer. The input variables used in the study were identical to the study of Senyurt and Subasi [2]. The accuracy of their 5-fold cross validated predictions from ANN model was 56.30%, the hybrid intelligence of Artificial Neural Network (ANN) and Genetic Algorithm (GA) yields the average accuracy as high as 63.60%. It can be seen that the accuracy is higher than the previous ANN model.

Apart from support vector machine, decision tree, and artificial neural network, there are claims made by other researchers that recurrent neural network (RNN) could outperform any other methods mentioned previously. Such neural network has an architecture called Long Short-Term Memory network (LSTM). LSTM is an improved recurrent neural network which can remember thing for longer time. It is claimed that it is appropriate for time series data.

Starting with recurrent neural network, Bernal et al. [1] found that recurrent neural network (RNN) could perform better for time series data. They decided to use Echo State Network (ESN), one of the reservoir network of RNN, to predict on Google stock in S&P500 index. The result is compared it with the Kalman filter, an algorithm which predict the price over time series. They found that ESN test error is less than the Kalman Filter around 79 times.

Hansson [10] used LSTM in his research. For his work, he compared the forecast of stock returns and direction between the conventional time series model (ARMA-GJRGARCH) to the neural network model (LSTM). The prediction was performed on American, Brazilian, and Swedish stock markets (SP500, Bovespa, and OMX). He tested whether the model constructed by machine learning algorithm would yield the same prediction result when exposing to different markets. As a result, the model constructed by time series model ARMA-GJRGARCH attain the maximum prediction result at 52.55%, 50.43%, and 52.14% on Bovespa, SP500, and OMX, respectively. Recurrent neural network model such as LSTM, Deep LSTM, Softmax LSTM, and Softmax deep LSTM yield similar results to Softmax deep LSTM. This LSTM algorithm attained the highest prediction accuracy at 55.30%, 50.87% and 50.00% on OMX, SP500 and Bovespa, respectively. To conclude, different markets were different in terms of factors that drove the market. This suggested that prediction on different market using the same model may yield different results.

Roondiwala et al. [6] also used Long Short-Term Memory network (LSTM) for stock prices prediction. They found that this model got the least root mean squared error (RMSE) when using all parameters including the highest price, the lowest price, opened price, and closed price.

Work of Jia [5] give similar result. He used LSTM to predict Google's daily stock and construct a model training along with ADAM optimizer. The result had low root mean squared error which showed how effectiveness LSTM was to predict the time series data, e.g., stock market.

According to many works, it is expected that LSTM is one of the attractive choice used to predict the sequential data in time series such as TFEX in this project. For this reason, LSTM algorithm is chosen to improve the accuracy and expected to give low value of root mean squared error for TFEX prediction.

Chapter 3

Methodology

To obtain the results for TFEX prediction, it requires knowledge from many disciplines, e.g., future trading, features, and machine learning algorithms. The topics of knowledge are listing as follows.

3.1 Data Collection

There are over thousands tradeable futures worldwide. The chosen futures affect the accuracy of the developed model. This is because each future has different factors contributing to changes in price movement and direction. In this project, Thailand Future Exchange, or in short TFEX, is chosen for prediction.

3.1.1 Thailand Future Exchange (TFEX)

Thailand Future Exchange or TFEX is a subsidiary of Stock Exchange of Thailand (SET), which trades future instead of stock. The organization offers tradeable products including SET50 futures, SET50 options, USD futures, Single Stock Futures, RSS3D futures, Gold futures, and Gold-D. In order to trade future contracts on the selected product, TFEX trader should anticipate the price's direction of the market, and buy the future contract according to their anticipation.

However, the price's direction of TFEX mimics SET50, which has the greatest influence in stock market of Thailand. This means that the price's direction of TFEX can either rise or fall followed the changes in actual stock price.

3.2 Data Preparation

Data is the most important factor for machine learning algorithm. There are three steps for data preparation, which are features selection, data normalization, and data cleaning.

First step, feature selection is required for enhancing accuracy. It is performed and selected using existing features called technical indicator. Next step, data normalization is used for data scaling in order to create appropriate input for machine learning technique. It used features obtained from feature selection step. Last step, clean data for appropriate model's input.

3.2.1 Features Selection

In the collected data, there are only seven attributes, which are opened price, closed price, the highest price, the lowest price, volume, date, and time. This data is insufficient for precise prediction. Additional features such as technical indicator will be used.

Technical indicators are frequently used by traders to reduce noise and simplify the price movements. In order to predict future market's trend with high accuracy, technical indicators are used to enhance accuracy of the model to give better results. In this project, seven technical indicators are selected, which are Exponential Moving Average (EMA), Elastic-Volume Weighted Moving Average (eVWMA), Moving Average Convergence Divergence (MACD), On-Balance Volume (OBV), Relative Strength Index (RSI), Change in Closing Price (CCP), and Percentage Change in Closing Price (PCCP). The first two indicators will be combined and called moving average. These indicators will be discussed in the section below.

3.2.1.1 Moving Average (MA)

Moving average (MA) is a mean of time series data from consecutive time periods. It is called moving because it recomputes as soon as new coming data is available. It is done by dropping the first value in the calculation and adding the latest value. The Moving Average can be represented as shown in Equation (3.1).

$$MA_n = \frac{\Sigma_{i=1}^{n} Close_i}{n} \tag{3.1}$$

where $i \in \mathbb{N}$, n is number of time period of interest and $Close_i$ is the closing price at data number i that count from the first data to data number n.

There are many types of moving average used to predict a future price's direction. In this project, two moving averages are chosen as indicators for TFEX prediction, i.e., exponential moving average and elastic volume-weighted moving average. They will be discussed in the following sections.

Exponential Moving Average (EMA)

Exponential moving average (EMA) is a type of MA which gives more weight to the most recent data. This feature can be written as shown in Equation (3.2).

$$EMA_t = Close_t \times k + EMA_{t-1} \times (1 - k) \tag{3.2}$$

where EMA_t is EMA at current time t, EMA_{t-1} is EMA at previous time, $Close_t$ is the closing price at current time, and k is the exponential smoothing or smoothing factor that can be calculated from Equation (3.3). It determines how much weight is applied on each period.

$$k = \frac{2}{n+1} \tag{3.3}$$

where n is the number of time period using in EMA.

Elastic Volume-Weighted Moving Average (eVWMA)

Elastic volume-weighted moving average (eVWMA) is a moving average of volume over a specific period of time. It can be calculated using an Equation (3.4).

$$eVWMA_t = \frac{(n - Volume_t) \times eVWMA_{t-1} + Volume_t \times Close_t}{n} \tag{3.4}$$

where $eVWMA_t$ is eVWMA at current time t, $eVWMA_{t-1}$ is eVWMA at the previous time period, n is the number of time period, $Volume_t$ is contract volume at the current time, and $Close_t$ is the closing price at current time.

3.2.1.2 Moving Average Convergence Divergence (MACD)

Moving average convergence divergence (MACD) is the feature, which shows relationship between shorter moving average and longer moving average. The result of MACD is obtained by finding the histogram, which shows the relationship of two moving averages. In order to obtain histogram, it must calculate two types of lines, i.e., MACD line and the signal line.

In order to obtain MACD line, there are two lines that must be calculated first, which are shorter moving average and longer moving average. These two lines can be calculated using Equation (3.5) and (3.6), respectively.

$$Shorter\ Moving\ Average = EMA_n \tag{3.5}$$

$$Longer\ Moving\ Average = EMA_m \tag{3.6}$$

where EMA_n and EMA_m is EMA that using n and m as the number of time period and m is greater than n.

The MACD line is a line between these two moving averages. It can be calculated by subtracting longer moving average from shorter moving average as shown in Equation (3.7). The result of calculation is a line shows relationship between these two lines. The relationship between shorter moving average line, longer moving average line, and MACD line is shown in Figure 3.1.

$$MACD\ line = Shorter\ Moving\ Average - Longer\ Moving\ Average \tag{3.7}$$

After MACD line is obtained, the signal line will be calculated by finding EMA of that MACD line, which corresponds to a specific period of time. It creates a transaction signals such as buying and selling, which could be called a 'triggered line'.

Lastly, after both MACD line and the signal line are obtained, the MACD line will be subtracted by the signal line. The result from this calculation will be MACD histogram. The relationship between MACD line, signal line, and MACD histogram will be shown in Figure 3.2.

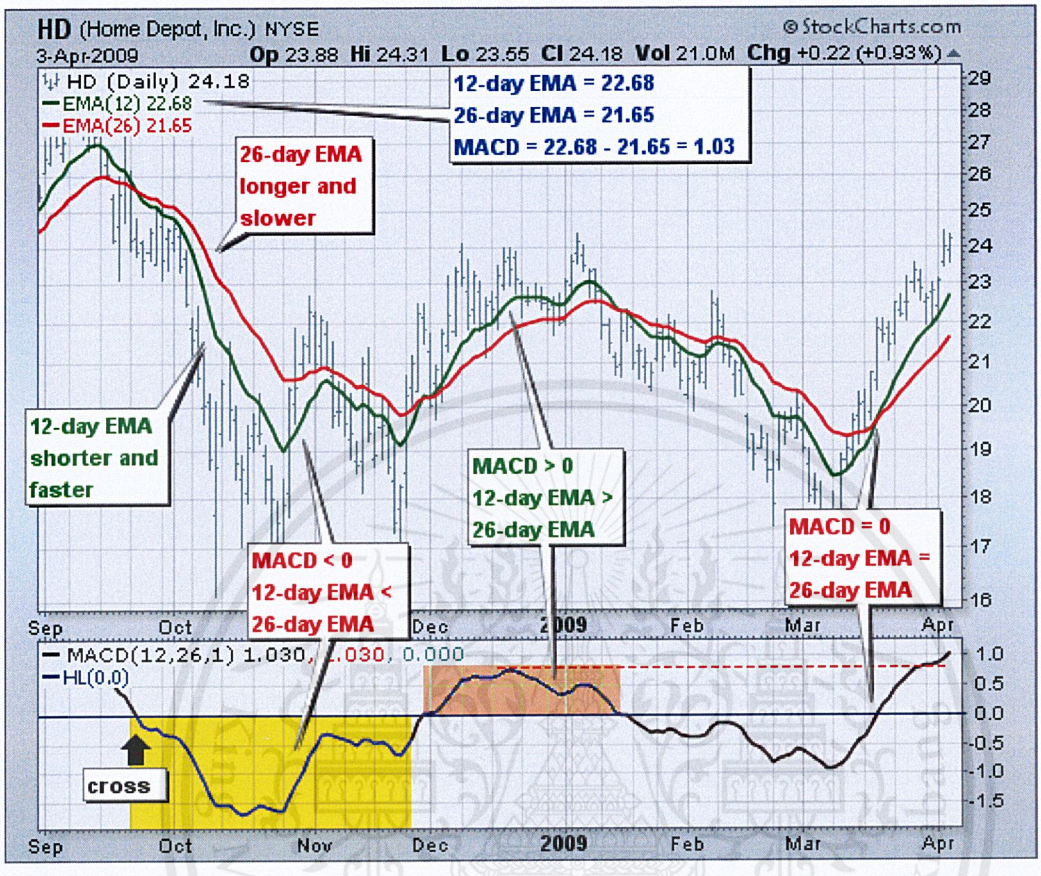

Figure 3.1: This figure shows relationship between shorter moving average line using EMA_{12}, longer moving average line using EMA_{26}, and MACD line. [11]

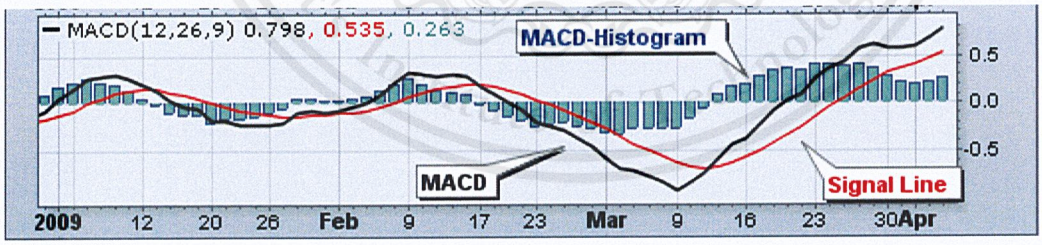

Figure 3.2: This figure shows relationship between MACD line, signal line, and MACD histogram. [11]

3.2.1.3 On-Balance Volume (OBV)

On-balance volume (OBV) is a measurement of buying and selling pressure, which uses to capture positive and negative volume flow. It follows the

Equation (3.8).

$$OBV_t = \begin{cases} Close_t > Close_{t-1}; & OBV_{t-1} + Volume_t \\ Close_t < Close_{t-1}; & OBV_{t-1} - Volume_t \\ Close_t = Close_{t-1}; & OBV_{t-1} \end{cases} \tag{3.8}$$

where OBV_t is OBV at current time t, $Close_t$ and $Close_t - 1$ is the closing price at current time t and previous time, respectively. $Volume_t$ is the volume from each data that is related to the current time t.

3.2.1.4 Relative Strength Index (RSI)

Relative strength index (RSI) calculates the ratio of price movements using gain and loss of each data. The primary usage of RSI is to identify overbought or oversold condition of future trading.

Gain and loss for each data can be calculated by comparing the current closing price with the previous one. If the current closing price exceeds the previous closing price, it will be determined as gain value, otherwise it is determined as loss value. It should be noted that data cannot have gain and loss at the same time. For this reason, in case the calculation result is a gain, the loss will be set to zero automatically, and vice versa.

When all gains and losses are obtained, RSI will use their averages as representation for each variable. For the first calculation of the average gain, Equation (3.9) will be used. Similarly, Equation (3.10) is used for the first calculation of the average loss.

$$Average\ Gain_0 = \frac{\sum_{i=0}^{n-1} Gain_i}{n} \tag{3.9}$$

$$Average\ Loss_0 = \frac{\sum_{i=0}^{n-1} Loss_i}{n} \tag{3.10}$$

where $i \in \mathbb{N}$, n is a specific period of time and $Gain_i$ and $Loss_i$ are gain and loss at data number i that count from the first data to data number n. After these calculations, average gain and loss at time t can be found as Equation (3.11) and (3.12) below.

$$Average\ Gain_t = \frac{Average\ Gain_{t-1} \times (n-1) + Gain_{t-1}}{n} \tag{3.11}$$

$$Average\ Loss_t = \frac{Average\ Loss_{t-1} \times (n-1) + Loss_{t-1}}{n} \tag{3.12}$$

where $Average\ Gain_t$ and $Average\ Loss_t$ are average gain and loss at current time t, $Average\ Gain_{t-1}$ and $Average\ Loss_{t-1}$ are average gain and loss at previous time, $Gain_{t-1}$ and $Loss_{t-1}$ are gain and loss at previous time, and n is a specific period of time.

After average gain and loss are computed, the ratio between these two variables is calculated by dividing average gain by average loss. This ratio is called a Relative Strength value. A Relative Strength value will be used to compute in Equation (3.13) to find a final result of relative strength index.

$$RSI = 100 - \frac{100}{1 + Relative\ Strength} \tag{3.13}$$

which means the value of RSI will only be in the range from 0 to 100.

3.2.1.5 Change in Closing Price (CCP)

Change in closing price (CCP) is the features that shows the trend of changes in closing price. This feature is calculated using the Equation (3.14).

$$CCP_t = Close_t - Close_{t-1} \tag{3.14}$$

where CCP_t is the change in closing price at current time t, $Close_t$ is the closing price at time t, and $Close_{t-1}$ is the closing price from previous time.

3.2.1.6 Percentage Change in Closing Price (PCCP)

Percentage change in closing price (PCCP) is the features that shows the trend of changes in closing price in percentage. This feature is derived from CCP by finding the percentage of changes instead of actual changes in price. It can be calculated using the Equation (3.15).

$$PCCP_t = \frac{CCP_t}{Close_{t-1}} \tag{3.15}$$

where $PCCP_t$ is the percentage change in closing price at current time t, CCP_t is the change in closing price at current time t, and $Close_{t-1}$ is the closing price from previous time.

3.2.2 Data Normalization

Raw data comes in differences in units or scales. These differences will affect the model's prediction accuracy. This means that the accuracy obtains

from prediction will be biased. One of the methods to normalize input data for neural network is Z-Normalization. Z-Normalization is used to rescale data to standard normal curve, in order to have a zero mean and standard deviation of 1. A formula for Z-normalization can be written as shown in Equation (3.16).

$$x'_i = \frac{x_i - \mu}{\sigma} \tag{3.16}$$

where $i \in \mathbb{N}$, x_i is the data of interest at data number i, μ is an average of the data, and σ is the standard deviation of the data, x'_i is the normalized data from x_i.

A normalization has been applied to every data parameters and technical indicators.

3.2.3 Data Cleaning

When added values that are calculated from technical indicators to dataset, some of them might have missing value in the first few data. This happened because some technical indicators required n, which is the number of time period that must be used to calculate the result. In other word, first n value for each n-required indicator could be missing. To solve this problem, the missing value handling method will be used.

Missing value handling could be done by finding mean for each attribute. Then use that to fill in the missing data in each attribute.

3.3 Long Short-Term Memory (LSTM)

To classify the price's direction of TFEX accurately, the efficient and suitable machine learning algorithm is required for prediction since TFEX market itself is a sequential data. The authors expected that anything that happened in the previous time step could affect the price's direction of TFEX in the next time step. The machine learning algorithm, which used in sequential data modeling would be the suitable method for doing price's direction of TFEX prediction.

However, there are several methods, which can be used to predict time series data. One of these methods, a Long Short-Term Memory or LSTM, is widely used in many stock prediction researches. This architecture is an

improved version of recurrent neural network on memory lacking problem due to the problem of long-term dependency.

In order to understand how it works, artificial neural network (ANN), recurrent neural network (RNN), and long short-term memory will be explained in the next section, respectively.

3.3.1 Artificial Neural Network (ANN)

Artificial neural network is a type of machine learning algorithm that mimics biological neural networks. It composed of interconnection of neuron cells forming a network that could receive, process, and transmit complex signal. Similarly, artificial neural network mimicking this structure, it composed of nodes and links together forming a network that could model complex relationship between input and output signals.

In artificial neural network, these nodes are distributed into three distinct layers, i.e., input layer, hidden layer, and output layer. Similar to human brain, the neural network behavior cell receives multiple inputs in various categories as informations and process it. The result will be given after its process has been done. Neural network has three steps of process on its node's structure, which classified as a 'layer'. First, each node receives data from the prior layer's nodes. Then, It performs computation on the received data. After that, the result is passed through the node on the post layer as depicted in Figure 3.3.

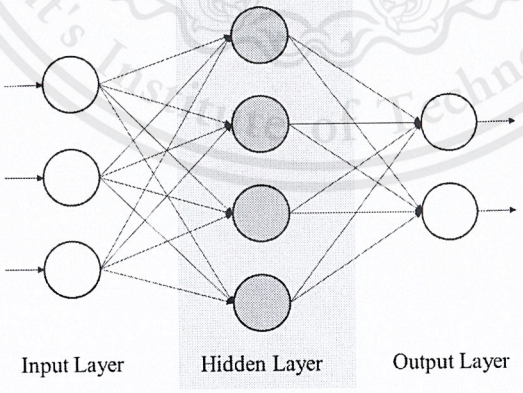

Figure 3.3: Artificial Neural Network (ANN)

Both input and output have only one layer while hidden layer could have more than one layer. Many kinds of informations are received by an input

layer and it is sent to the hidden layer. After that, that information will be used by hidden layers to construct the model by giving weight to the information. The more important of the data input, the more weight it will be given. Finally, output layer will provide an answer after the model is fully developed, which is the thing that information try to lead on.

Each node, apart from input layer, receives previously computed data from nodes in the prior layer. Upon receiving data, the data is combined using summation function. The result is then inputted into another function called activation function. It establishes constraints on the range of data's value in the network as shown in 3.4 where in_i is weight output from node i in the prior layer to node in the current layer as input, *out* is output from the current node, and f is the current node's activation function.

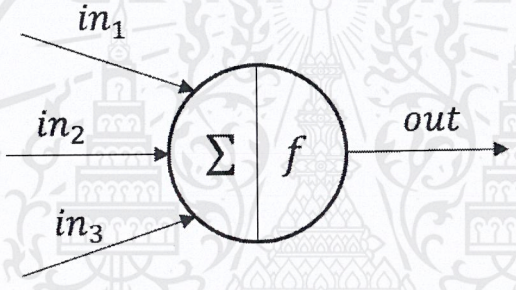

Figure 3.4: Artificial Neuron Activation Function

In order to use model for prediction, the model must be trained. To train the network, back propagation method is used. It is the process for adjusting model's weight to be more balance. It calculates, adjusts, and minimizes the error contribution of each neuron after processing batches of inputs from its previous nodes. It consists of two phases in calculation, which are forward pass and backward pass.

In the forward pass, there are two values that must be consider, i.e., total net input and output. Total net input is summation of the weight on the edge multiplied by its node's value. Output feed the total net input's value to the activation function to get the result. This is represented as Figure 3.4 and can be written as Equation (3.17) and (3.18) for total net input and output, respectively.

$$net = \Sigma(in_i \times node_i) \tag{3.17}$$

$$out = f(net) \tag{3.18}$$

where net is the total net input, in_i is the input weight from node i, $node_i$ is node i's value, out is the output from current node, and f is the activation function.

After a calculation has done, the overall error can be calculated using a square error equation as written in Equation (3.19), called gradient.

$$Error_{total} = \Sigma(\frac{(target - out)^2}{2}) \tag{3.19}$$

where $Error_{total}$ is the overall error, $target$ is the expected value, and out is an actual value of that node, which is calculated from Equation (3.18).

After the forward pass has been done, network will use $Error_{total}$ in order to adjust weight of model in backward pass to minimize an error. It uses derivative function to find the change in error compare to actual weight as written in Equation (3.20).

$$Change\ in\ error = \frac{\partial Error_{total}}{\partial in_i} \tag{3.20}$$

where $Error_{total}$ is overall error, which is calculated from Equation (3.19) and in_i is weight of node i that need to be adjusted.

Unfortunately, this equation can not be done directly. Since $Error_{total}$ is calculated using out and out is calculated using net, which is the only function that has in_i. Instead of directly derivative, this can be done using partial derivative as written in Equation (3.21).

$$\frac{\partial Error_{total}}{\partial in_i} = \frac{\partial Error_{total}}{\partial out}\frac{\partial out}{\partial net}\frac{\partial net}{\partial in_i} \tag{3.21}$$

After the calculation from Equation (3.21) has been completed, this will be used to adjust the value of in_i for next forward pass using the following Equation (3.22).

$$\widehat{in}_i = in_i - \eta\frac{\partial Error_{total}}{\partial in} \tag{3.22}$$

where $\widehat{in}_i$ is a new weight for node i. It is calculated from the old weight in_i of node i subtracted by η that is the learning rate, which optimize loss in the calculation with any number of choices. After this calculation is done, the whole network will repeat this process until the model's output is closer to the target output.

However, The main problem of many neural networks comes from activation function f. Activation function is generally used in every neural network models. It is used to define the node's output and send it to the next layer as an input. This is important for neural network process since most of the data is non-linear. Without activation function, data will feed the linearly input to each node, which is less complex than non-linear. It causes low performance model compare to the neural network model, which has activation function inside.

Activation function limits possible output value from each node into short range to make it easier for computing in next connected node. There are several types of activation functions that can be used in neural network model. In this project, logistic activation function, tan hyperbolic function, and rectified linear unit are choices of activation function for model construction.

Logistic activation function can be called as sigmoid function using σ as a symbol. This function helps rescale the range of output data to start from 0 to 1, which also makes a smooth gradient. It is famous for being used in many neural network structures. Unfortunately, if gradient is getting smaller, each layer and the network will refuses to learn further or learn dramatically slow.

To solve this problem, the Tan hyperbolic function or *tanh* claimed to be more capable with wider range of output from -1 to 1 instead of 0 to 1. The advantage of this range give *tanh* to be more clear when observed the change of its output.

Another choice for activation function is rectified linear unit or *ReLU*. The concept of this function is to decide whether the output is greater than zero or not. If it is, that value will be used as an output, otherwise the value will be zero instead.

3.3.2 Recurrent Neural Network (RNN)

Recurrent Neural Network is a type of neural network. It uses previous computational result as an additional input to the next time step. At each hidden node, there are two outputs, which are concatenated vector for next node and for itself in the next time step. Architecture of recurrent neural network will be shown as Figure 3.5.

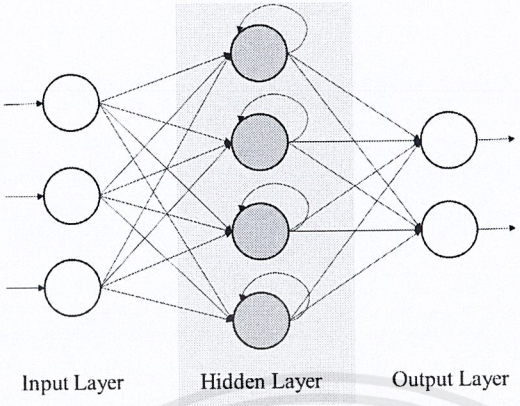

Figure 3.5: Recurrent Neural Network (RNN)

Same as other neural networks, each node's output in RNN can be obtained by formula as written in Equation (3.23).

$$output_t = V \cdot f(U \cdot input_t + W \cdot output_{t-1}) \tag{3.23}$$

where $output_t$ is the output from that node at time step t, $output_{t-1}$ is the result of node computation from previous time step, $input_t$ is the data that fed into node at step t, f is an activation function that try to squash data into very short range, and U, V and W are network parameters that are constant variable through all time step t.

An additional output that feeding itself considered as a memory, but with limitation. This memory is short-term, which means it only remembers what it has in previously computed time step. This problem comes from the vanishing in gradient when neural network try to do backward pass for back propagation in order to adjust model.

The vanishing gradient problem can be seen when there is a huge change in previous layers, but it does not cause big change of the network to the next layer. This problem is occurred when training the network. With help of some activation functions that try to minimizes the value into small range. Some of their output is in range 0 to 1, mostly a sigmoid function. When do the backward pass, specifically in RNN, it must multiplied back with the early node's value and early time step of each node's value to adjust weight of the model. When doing this, the value that is already in floating point multiplied together will give less value output. After long time of training, even big change in parameters has been occurred, the output will be a small value only. This means network model will has a slightly change later on.

Consequently, this property is not sufficient for time series data prediction such as TFEX market. Since the computational results from any time step in the past are not able to retrieve in order to increase prediction accuracy. To solve this problem, more powerful recurrent neural network architectures called Long Short-Term Memory or LSTM is proposed in the following section.

3.3.3 Long Short-Term Memory (LSTM)

Long Short-Term Memory or LSTM is one of the recurrent neural network architectures. It could be used with data that is sequences such as text, sound, or time series. The advantage is LSTM has a memory cell inside each node that keeps track of and stores memory inside the node. LSTM has option to read, to write, to remember, or to forget any memory inside each node through time. This is suitable for TFEX prediction that has long-term dependency information.

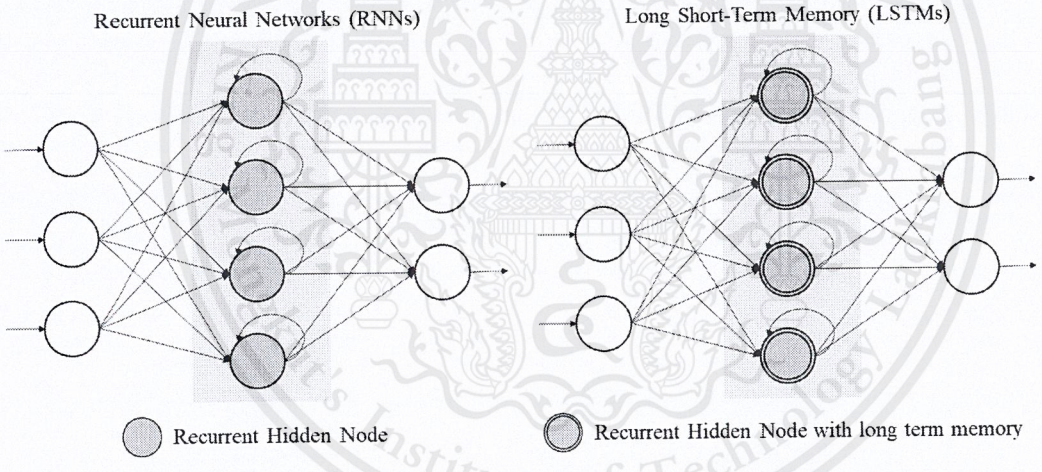

Figure 3.6: Architecture comparison between Recurrent Neural Network or RNN (left) and Long Short-Term Memory or LSTM (right)

The strength of LSTM is it has 'long-term memory' characteristic. It could keep track of each data's state at any time step. State of each data could be either kept or discarded. This is different from RNN that has no memory and the current result relied on previous computation only. LSTM used both previous computational result and its memory to provide output as shown in Figure 3.6 above.

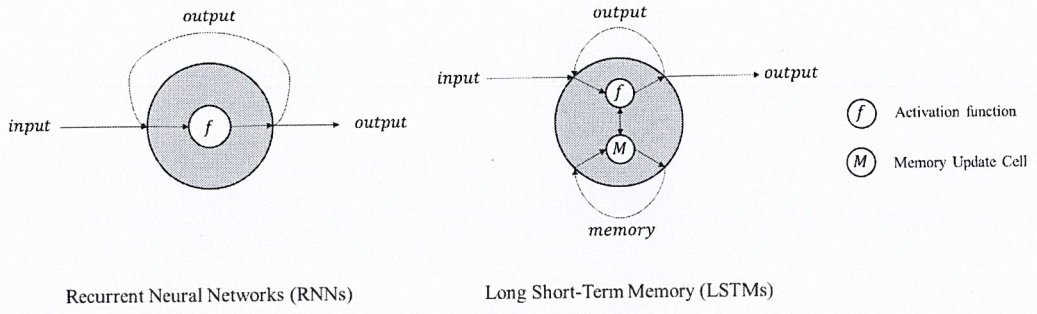

Figure 3.7: Comparison of each node between Recurrent Neural Network or RNN (left) and Long Short-Term Memory or LSTM (right)

Hidden nodes in both networks are called recurrent node. It consists of an activation function, same as artificial neural network node. The difference is that each recurrent node in both networks use its own output as another input to itself in the next time step, along with new input from previous node.

Nevertheless, big difference between recurrent neural network node and long short-term memory node is that only long short-term memory has memory update cell as shown in Figure 3.7. According to recurrent neural network, since it has no memory to keep track of the state of data, the output that loop as input will be vanished through time. This makes RNN's memory useful for only a few steps back. On the other hand, LSTM uses its memory update cell to help each node remember data that may be important for long term use. Memory update cell will interact with the input at every time steps. It has to decide whether to keep or to remove each input by using its memory to consider an action for each data. From there on, both data and memory will be updated and ready to be sent out as an output from that recurrent node. Full LSTM's node architecture is shown in Figure 3.8 below.

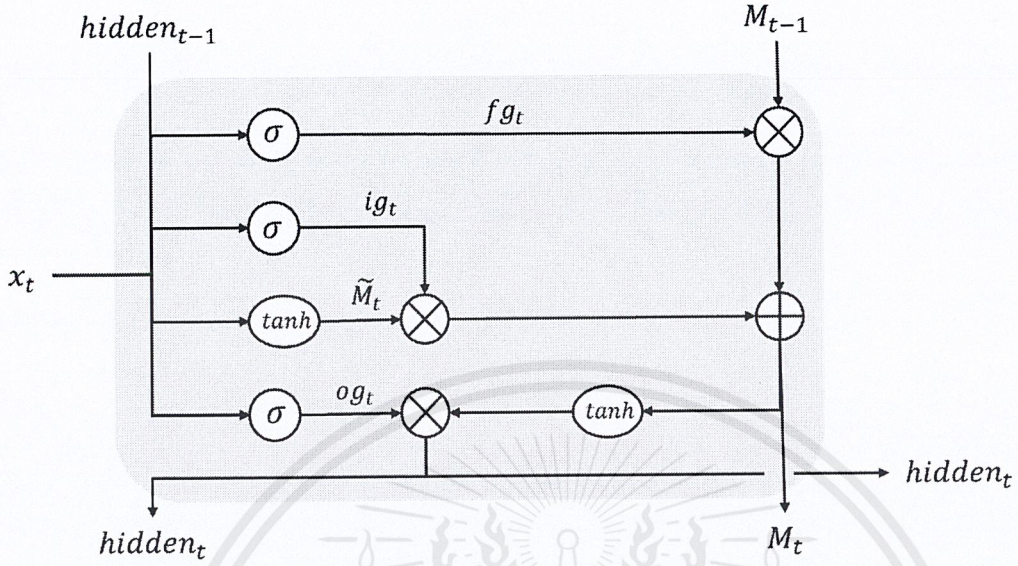

Figure 3.8: Long Short-Term Memory (LSTM) architecture

In LSTM's node architecture, it consists of three gates inside. These gates control the input. They decide what should be the removed, outputted, and kept. First gate is called forget gate, which can be written as Equation (3.24).

$$fg_t = \sigma(W_{fg} \cdot [hidden_{t-1}, x_t] + bias_{fg}) \tag{3.24}$$

where fg_t is the forget gate value at time t, W_{fg} is static weight for forget gate, $hidden_{t-1}$ is the value from previous hidden node, x_t is the data that feed at time t, and $bias_{fg}$ is a static bias for forget gate.

Second gate is the input gate, which is written as Equation (3.25).

$$ig_t = \sigma(W_{ig} \cdot [hidden_{t-1}, x_t] + bias_{ig}) \tag{3.25}$$

where ig_t is the input gate value at time t, W_{ig} is static weight for input gate, $hidden_{t-1}$ is the value from previous hidden node, x_t is the data that feed at time t, and $bias_{ig}$ is a static bias for input gate.

Its equation may look same as the forget gate equation. The different is variables, which are an input gate result ig_t, a static parameter for input gate W_{ig}, and a static bias for input gate $bias_{ig}$. However, this gate also generates a new memory by using an Equation (3.26).

$$\tilde{M}_t = tanh(W_{\tilde{M}} \cdot [hidden_{t-1}, x_t] + bias_{\tilde{M}}) \tag{3.26}$$

where $\tilde{M}$ is a new memory calculated using the same input $hidden_{t-1}$ and x_t as previous gate. But multiplied with static parameter $W_{\tilde{M}}$ that is a static weight of new memory $\tilde{M}$, then add $bias_{\tilde{M}}$ as bias of a new memory cell instead. The only change is that this gate uses $tanh$ activation function instead of σ function.

After both input gate ig_t and new memory $\tilde{M}_t$ are calculated, these two vectors will be multiplied together. The multiplied vector will be concatenated with the forget gate's vector fg_t that is already multiplied with previous memory M_{t-1}. The result is a new memory M_t at current time step t, which will be needed in later time step calculation. This process could be written as Equation (3.27).

$$M_t = fg_t \times M_{t-1} + ig_t \times \tilde{M}_t \tag{3.27}$$

The last gate is called an output gate. It determines the output at the current time step t. Process of this gate is the same as previous gates. The only difference is that when it has been finished its calculation from Equation (3.28), it will be multiplied with the memory M_t that is already computed from previous gate. Then squash it with $tanh$ activation function to get the result h_t for current time step t as written in Equation (3.29).

$$og_t = \sigma(W_{og} \cdot [hidden_{t-1}, x_t] + bias_{og}) \tag{3.28}$$

$$h_t = og_t \times tanh(M_t) \tag{3.29}$$

where W_{og} is the static parameter for output gate, $bias_{og}$ is the static bias of output gate, and og_t is the output gate result for current time step t.

These computations are repeated with number of time steps with the same parameters W_{fg}, W_{ig}, $W_{\tilde{M}}$, and W_{og} across all time step t.

Apart from this architecture, there are several variants on LSTM architecture, which is a bit different. But still based on the same context to make long term memory, which will be useful in the future computation.

Chapter 4

Experimentation

4.1 Experiment Setup

There are two main parts of the program. First is the dataset acquisition, the dataset is represented in .csv format. Second is the program or machine learning implementation. Python is chosen for the implementation. It is used to process the data, train the machine learning model, and measure the performance of each model.

4.2 Development Tools

The system is implemented in Python via PyCharm Version 2017.3. Libraries used are shown in the Table 4.1. The experiments were conducted on personal computer with the following specification: Intel(R) Xeon(R) CPU 3.07 GHz, 24.0 Gb of RAM, and 64-bit Operating System.

This project started with Neuralnet in R, which is the first selected tool for ANN experiment. Since R is the most widely used computer language for

Library	Version	Purpose
TTR	0.23-1	Calculate technical indicators
Tensorflow	1.4	Machine learning library
Keras	2.1.4	High level neural network API written in Python

Table 4.1: Libraries used

Number of hidden nodes	Neuralnet in R		Keras in Python	
	Accuracy	Time Complexity	Accuracy	Time Complexity
10 *nodes*	51.00 %	66.0 mins	56.63 %	0.40 mins
20 *nodes*	53.87 %	139.8 mins	55.13 %	0.41 mins
30 *nodes*	50.53 %	217.8 mins	54.74 %	0.42 mins

Table 4.2: Accuracy and Time complexity comparison between Neuralnet in R and Keras in Python

statistical computation, several libraries are available for data science such as machine learning.

After conducted experiments based on Neuralnet library in R, time complexity seems to be the most important factor that slows down experiments. Therefore, another library is chosen. This library is called Keras. It is a high-level neural network API with focus on fast experimentation. According to work of Aruoba and Fernández-Villaverde [14], Python was faster than R. Moreover, the obtained results of experiment with the same setup shown in Table 4.2 shows that Keras is more efficient than Neuralnet in R. Keras in Python is decided to be used mainly for experimentation from this reason.

4.3 Dataset

The dataset contains 6 columns which are:

- Date: Date in the format of dd/mm/yy
- Opened price: The opened price is the price of the first stock that is traded when the stock market opened on the selected time period.
- Closed price: The closed price is the price of the last stock that is traded before the market closes on the selected time period.
- Lowest price: The lowest price of the stock that is traded on the selected time period.
- Highest price: The highest price of the stock that is traded on the selected time period.
- Volume: The amount of contract that is traded on the selected time period.

The dataset is a 60 minutes dataset consisting of 12,667 data points from February 1st, 2010 to July 6th, 2017.

4.4 Data Preparation

4.4.1 Classifying Data

The data is classified by the following method:

- *Up* if $Close_{t-1} < Close_t$
- *Down* if $Close_{t-1} \geqslant Close_t$

where:

t = The current data point which is sorted by time.

$Close_t$ = The close price of TFEX which could be found in the dataset at current time.

$Close_{t-1}$ = The close price of TFEX which could be found in the dataset at previous time period.

The data point classification starting from the second data point as there is no $ClosePrice_{t-1}$ at the first data point.

4.4.2 Data Calculation

There are 3 steps in data calculation. Firstly, technical indicators calculation. They are calculated by the TTR package using the value in the dataset of each trading period. Secondly, data missing filling method is used to fill the empty field in the dataset to reduce inconsistent data. Lastly, the data will be normalized using Z-normalization.

4.4.3 Measuring Accuracy

The accuracy of the model is measured by comparing the predicted TFEX price's direction with the actual price's direction. The predicted price's direction is obtained from finding the difference between the predicted price and the actual price of the previous time period. If the result is positive, then the predicted price's direction is labeled as upward, otherwise it will be labeled as downward. Accuracy is calculated using ratio of the corrected price's prediction over number of data items in prediction.

4.5 Experiment: Artificial Neural Network and Long Short-Term Memory Accuracy Comparison

Experiment is conducted to compare the prediction accuracy of Artificial Neural Network model and Long Short-Term Memory model in three time frames: hourly, daily, and weekly. In order to obtain and compare the highest prediction accuracy of each model, optimization on its topology must be completed. To achieve this, experimentation on finding the best model's topology or the number of hidden nodes that gives the highest prediction accuracy of both models are conducted.

For experimentation, it is divided into two sub-experiments. The first experiment is conducted to obtain TFEX price's direction prediction accuracy of the model as number of hidden nodes increases for ANN. The second experiment is conducted to obtain TFEX price's direction of prediction accuracy of the model as number of hidden nodes increases for LSTM. The result from both experiments is compared and used to affirm the paper's hypothesis of Long Short-Term Memory yielding higher prediction accuracy on time series data such as TFEX market.

4.5.1 Artificial Neural Network Experiments

4.5.1.1 Artificial Neural Network Methods

This experiment is conducted several iterations and is divided into two distinct phases which are data feeding phase and topology adjusting phase.

In the data feeding phase, the initial data parameters in the obtained dataset along with five technical indicators are fed into ANN to construct the machine learning model. The five technical indicators used are *Exponential Moving Average (EMA), On-Balance Volume (OBV), Moving Average Convergence Divergence (MACD), Relative Strength Index (RSI), and Exponential Volume-Weighted Moving Average (eVWMA).*

In the topology adjusting phase, number of hidden nodes is set, recorded, and changed to find the trend of prediction accuracy as number of hidden nodes increases. The process of finding the best topology for ANN model is further broken down into two steps which are the coarse estimation step and the fine estimation step.

In the coarse estimation step, number of hidden nodes is set and increased from 10 to 100 hidden nodes and incremented by 10. At the end of each iteration the prediction accuracy and its respective number of hidden nodes is recorded. Its goal is to find the approximate number of hidden nodes that give the highest prediction accuracy. At the end of this step, the number of hidden nodes that give the highest prediction accuracy for three time frames which are hourly, daily, and weekly are obtained as *hiddenNodesHourly*, *hiddenNodesDaily*, and *hiddenNodesWeekly* respectively. These number of hidden nodes are passed to the next step which is the fine estimation step.

In the fine estimation step, the number of hidden nodes for three time frames resulted from previous step is further experimented to find the precise number of hidden nodes that give the highest prediction accuracy for the model. This time the number of hidden nodes is set and increased from *hiddenNodesHourly - 5, hiddenNodesDaily - 5, hiddenNodesWeekly* to *hiddenNodesHourly + 5, hiddenNodesDaily + 5, hiddenNodesWeekly + 5* respectively, incremented by 1. At the end of each iteration, the prediction accuracy and its respective number of hidden nodes is also recorded. After that, three numbers of hidden nodes that give the highest prediction accuracy for the model in three time frames are obtained.

4.5.1.2 Artificial Neural Network Results

The results of the experiment using Artificial Neural Network or ANN are shown in Table 4.3. The experiment shows the trend of prediction accuracy of the coarse estimation on hourly time frames.

Number of hidden nodes (node)	ANN accuracy (%)	Time complexity (sec)
10	79.82	1.58
20	79.46	1.54
30	80.06	1.53
40	82.23	1.40
50	80.45	1.67
60	81.64	1.53
70	79.98	1.81
80	80.85	1.70
90	80.65	1.44
100	81.99	2.00

Table 4.3: Artificial Neural Network accuracy on coarse estimate hourly prediction

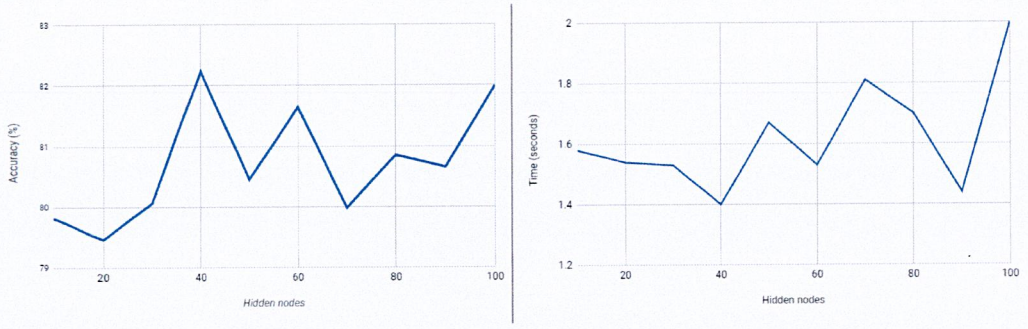

Figure 4.1: ANN coarse estimate hourly prediction accuracy (left) and running time in seconds (right) on 60-minute dataset

From the Table 4.3 and figure 4.1, the trend of prediction accuracy lies around 80% with the highest at 82.23% at 40 hidden nodes.

Number of hidden nodes (node)	ANN accuracy (%)	Time complexity (sec)
10	73.57	1.17
20	73.00	1.13
30	76.17	1.11
40	76.13	1.14
50	77.43	1.20
60	76.80	1.21
70	77.75	1.16
80	76.44	1.25
90	77.79	0.98
100	76.29	1.30

Table 4.4: Artificial Neural Network accuracy on coarse estimate daily prediction

The ANN experiments on daily time frames with hidden node starting from 10 to 100 nodes show that 90 nodes model yield the highest accuracy as shown in Table 4.4. This leads to a conclusion that the number of hidden nodes in the range of 85 to 95 yields high accuracy. This number of hidden nodes is passed and further experimented in the fine estimation step as shown in table 4.7. Also looking at the line graph of the experiment shows in figure 4.2 shows that the prediction accuracy increases as the number of hidden nodes increases. The rising trend stop at around 50 hidden nodes before

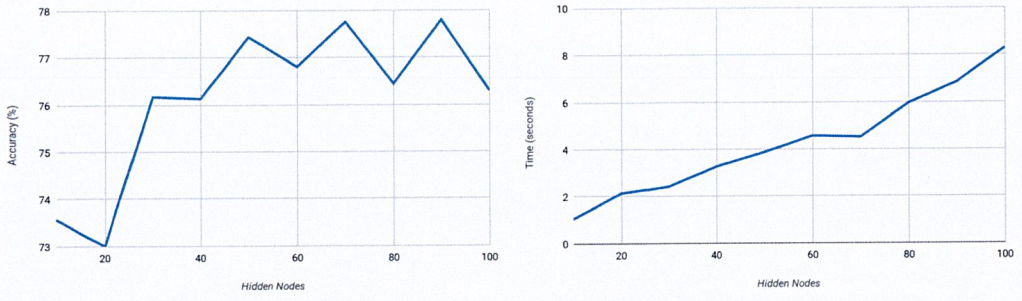

Figure 4.2: ANN coarse estimate daily prediction accuracy (left) and running time in minutes (right) on 60-minute dataset

swinging up and down within 76 to 78 % from 50 hidden nodes onwards. The observation suggests that the number of hidden nodes greater than 50 yields high prediction accuracy in the range of 76 to 78 % and looking at the swinging trend give some evident that the number of hidden nodes greater than 100 may give higher prediction accuracy but not significantly.

Number of hidden nodes (node)	ANN accuracy (%)	Time complexity (sec)
10	72.22	1.47
20	72.30	1.28
30	72.42	0.95
40	71.74	1.24
50	71.38	1.15
60	72.18	1.83
70	72.34	1.03
80	72.30	1.84
90	71.58	2.52
100	72.06	2.05

Table 4.5: Artificial Neural Network accuracy on coarse estimate weekly prediction

From the experimentation of coarse estimation on weekly time frame as shown in 4.5, the prediction accuracy lies within 71 to 72 % range from hidden nodes starting from 10 to 100 hidden nodes. The highest prediction accuracy is 72.42% at 30 hidden nodes.

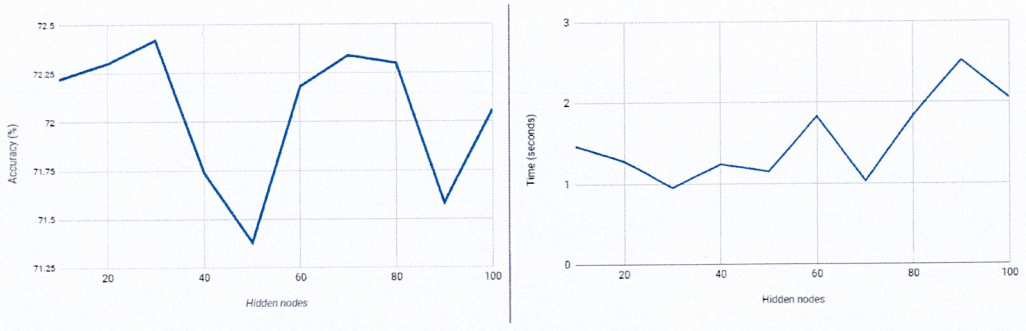

Figure 4.3: ANN coarse estimate weekly prediction accuracy (left) and running time in seconds (right) on 60-minute dataset

Looking at the line graph for the ANN's coarse estimation on weekly time frame in figure 4.3, there is an abrupt drop of prediction accuracy at 50 hidden nodes following by a recovery. The overall trend suggests that the prediction accuracy will continue to lies within 71 to 72 percentage range.

Number of hidden nodes (node)	ANN accuracy (%)	Time Complexity (sec)
35	78.99	1.37
36	80.81	1.47
37	80.41	1.57
38	80.02	1.76
39	81.04	1.73
40	77.58	1.56
41	81.60	1.07
42	80.06	1.89
43	82.50	1.79
44	81.36	1.78
45	81.44	1.51

Table 4.6: Artificial Neural Network on fine estimate hourly prediction

The experimentation on fine estimation on hourly time frame is shown in table 4.6 and 4.4. The number of hidden nodes that yield high prediction accuracy obtained from the coarse estimation on hourly time frame is 40 hidden nodes. The experiment is conducted on hidden nodes from the range of 35 to 45 nodes as shown by the table. The table shows that the prediction accuracy lies within 78 to 82 percentage range. The highest prediction accuracy is 82.50% at 43 hidden nodes.

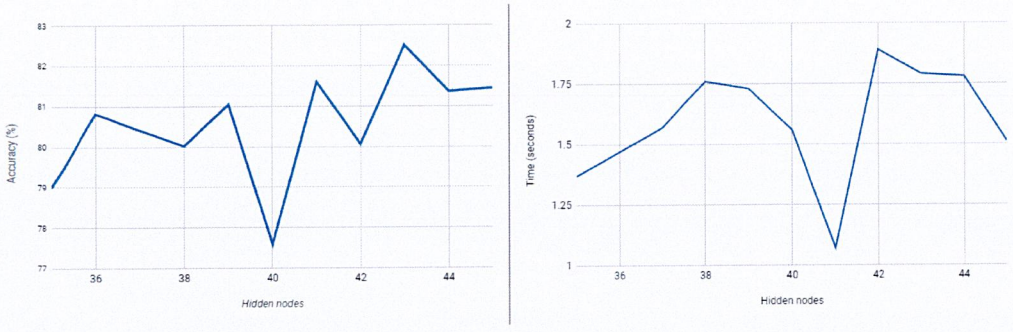

Figure 4.4: ANN fine estimate hourly prediction accuracy (left) and running time in seconds (right) on 60-minute dataset

Number of hidden nodes (node)	ANN accuracy (%)	Time Complexity (sec)
85	77.75	1.41
86	77.75	1.67
87	77.67	1.95
88	77.71	1.70
89	78.15	1.81
90	77.59	2.50
91	78.10	2.17
92	77.36	2.00
93	77.55	1.64
94	77.32	1.84
95	78.50	1.92

Table 4.7: Artificial Neural Network on fine estimate daily prediction

Experimentation on ANN's fine estimation on daily time frame is shown in the table 4.7 and figure 4.5. The result number of hidden nodes that give high accuracy obtained from the coarse estimation on daily time frame is 90 nodes, therefore the range of hidden nodes experimented is from 85 to 95 nodes. The table shows that the prediction accuracy lies within 77 to 78 percent range with the highest prediction accuracy of 78.50% at 95 hidden nodes.

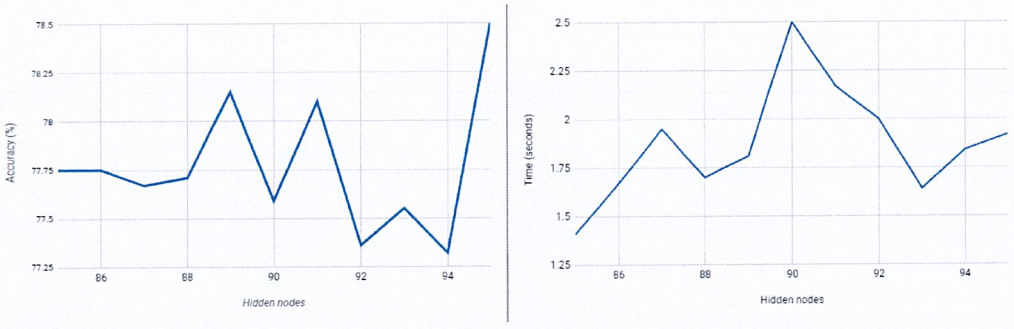

Figure 4.5: ANN fine estimate daily prediction accuracy (left) and running time in seconds (right) on 60-minute dataset

Number of hidden nodes (node)	ANN accuracy (%)	Time Complexity (sec)
25	62.13	1.03
26	71.66	1.90
27	71.74	1.54
28	72.78	1.57
29	71.62	1.83
30	71.18	1.26
31	71.06	1.70
32	71.98	1.72
33	71.78	1.33
34	71.94	1.44
35	71.30	1.28

Table 4.8: Artificial Neural Network on fine estimate weekly prediction

The experimentation on ANN's fine estimation on weekly time frame is shown via table 4.8. The number of hidden nodes that result in high prediction accuracy obtained from the ANN's coarse estimation on weekly time frame is 30 hidden nodes, therefore the hidden nodes experiment range is 25 to 35 hidden nodes. The table shows that the prediction accuracy lies within 62 to 72 percent range. The highest prediction accuracy is 72.78% at 28 hidden nodes.

The line graph shown by figure 4.6 shows that the prediction accuracy rises right from the beginning of experimented range, in this case is 26 hidden nodes. The rise is then follows by stable trend where the prediction accuracy

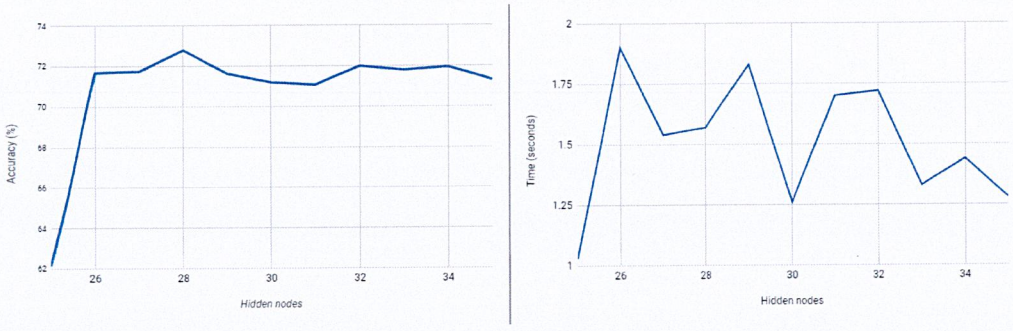

Figure 4.6: ANN fine estimate weekly prediction accuracy (left) and running time in seconds (right) on 60-minute dataset

lies within 71 to 72 percentage range.

4.5.2 Long Short-Term Memory Experiments

4.5.2.1 Long Short-Term Memory Methods

LSTM's experiment procedures is the same as that of ANN's experiment that is the experiment is divided into two phases which are the data feeding phase and the model adjusting phase.

In the LSTM's data feeding phase, same process that is done with ANN's data feeding phase is repeated here. That is five same technical indicators are calculated and fed into LSTM to construct machine learning model.

In the LSTM's topology adjusting phase, the same process done in ANN's topology adjusting phase is also repeated. That is in this phase the experiment is further broken down into coarse estimation step and fine estimation step. The resulted number of hidden nodes obtained from coarse estimation step is further experimented in the fine estimation step to find the number of hidden nodes that yield the highest prediction accuracy. In the coarse estimation step, the hidden nodes tested range is from 10 to 100 hidden nodes. In the fine estimation step, the result number of hidden nodes on three time frames which are hourly, daily, and weekly is used to further determine the exact number of hidden nodes that yield the highest prediction accuracy. The number of hidden nodes' range experimented in this step is *hiddenNodesHourly - 5, hiddenNodesDaily - 5, hiddenNodesWeekly* and *hiddenNodesHourly + 5, hiddenNodesDaily + 5, hiddenNodesWeekly + 5* repectively.

4.5.2.2 Long Short-Term Memory Results

The results of the Long Short-Term Memory experiments on coarse estimation on hourly time frames are shown in Table 4.9.

The LSTM's experimentation on coarse estimation on hourly time frame is shown in the Table 4.9 and figure 4.7. It shows that there is a significant different of approximately 20% between the highest prediction accuracy and the lowest prediction accuracy obtained by the LSTMS' models. The highest prediction accuracy obtained is 72.04% at 10 hidden nodes. Also the prediction accuracy drops as number of nodes increases.

Number of hidden nodes (node)	LSTM accuracy (%)	Time complexity (sec)
10	72.04	4.24
20	63.07	4.24
30	62.09	4.40
40	64.65	4.03
50	56.56	5.96
60	57.35	5.86
70	55.96	5.34
80	56.87	5.41
90	60.47	8.00
100	53.99	9.23

Table 4.9: Long Short-Term Memory on coarse estimate hourly prediction

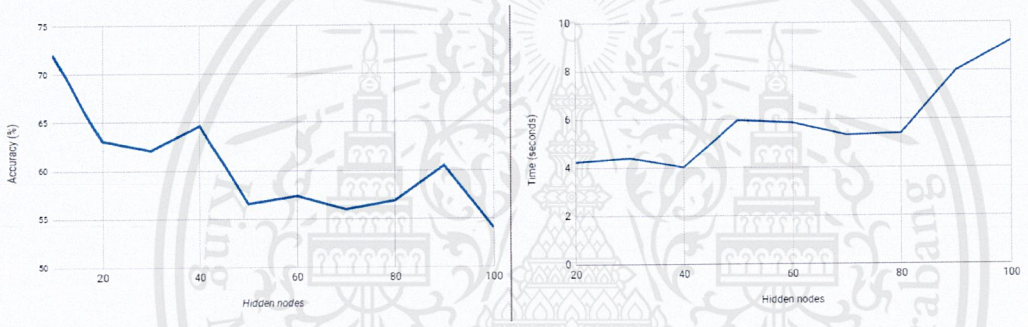

Figure 4.7: LSTM coarse estimate hourly prediction accuracy (left) and running time in seconds (right) on 60-minute dataset

Number of hidden nodes (node)	LSTM accuracy (%)	Time complexity (sec)
10	74.61	3.99
20	68.40	3.81
30	66.55	4.31
40	63.15	4.50
50	62.52	5.26
60	62.56	7.73
70	60.15	7.27
80	62.20	4.31
90	60.98	4.56
100	62.95	8.90

Table 4.10: Long Short-Term Memory on coarse estimate daily prediction

The experimentation on LSTM's coarse estimation on daily time frame is shown in the Table 4.10 and figure 4.8. The experiment shows that the range of model's prediction accuracy lies between 56 to 66 percent. The highest prediction accuracy obtained is 74.61% at 10 hidden nodes. Also the prediction accuracy drops as number of nodes increases.

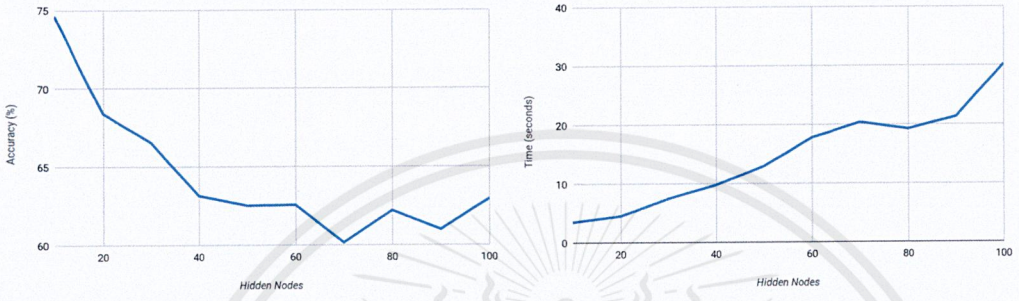

Figure 4.8: LSTM coarse estimate daily prediction accuracy (left) and running time in seconds (right) on 60-minute dataset

Number of hidden nodes (node)	LSTM accuracy (%)	Time complexity (sec)
10	66.05	3.95
20	58.41	3.90
30	57.09	3.90
40	59.77	6.55
50	60.89	4.34
60	58.73	4.87
70	55.60	5.12
80	56.16	4.32
90	56.00	9.57
100	56.00	5.18

Table 4.11: Long Short-Term Memory on coarse estimate weekly prediction

The LSTM's coarse estimation on weekly time frame is shown on the Table 4.11 and figure 4.9. The experiments show that the model's prediction accuracy range lies between 55 to 56 percent. The highest prediction accuracy is 66.05% with 10 hidden nodes. Also the prediction accuracy drops as number of nodes increases.

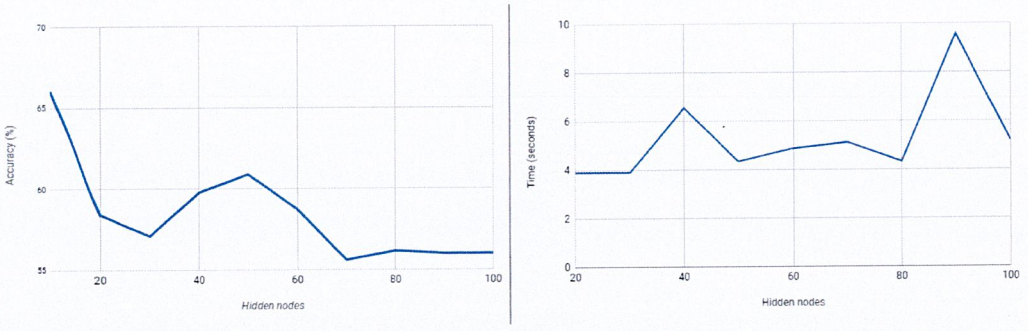

Figure 4.9: LSTM coarse estimate weekly prediction accuracy (left) and running time in seconds (right) on 60-minute dataset

Number of hidden nodes (node)	LSTM accuracy (%)	Time Complexity (sec)
5	73.22	4.13
6	70.10	4.39
7	73.54	4.61
8	72.55	4.36
9	69.63	4.31
10	68.40	4.05
11	66.11	3.96
12	68.52	4.64
13	67.81	3.89
14	67.50	4.11
15	64.42	4.83

Table 4.12: Long Short-Term Memory on fine estimate hourly prediction

From the observation, 10 hidden nodes yield highest accuracy in the LSTM's fine estimation on hourly time frame. This lead to the conclusion that the number of nodes in range 5 to 15 could yield high accuracy. The results of this experiment in Table 4.12 and figure 4.10 shows that the prediction accuracy reaches the highest at 7 hidden nodes with 73.54%. Also the prediction accuracy drops as number of nodes increases.

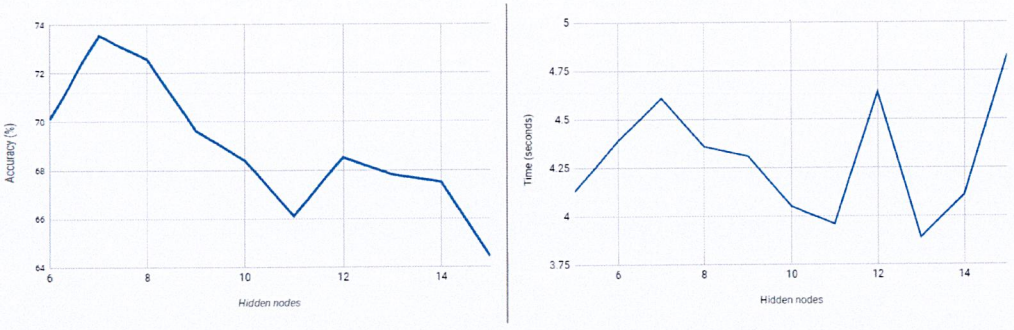

Figure 4.10: LSTM fine estimate hourly prediction accuracy (left) and running time in seconds (right) on 60-minute dataset

Number of hidden nodes (node)	LSTM accuracy (%)	Time Complexity (sec)
5	71.41	3.94
6	72.29	4.10
7	72.91	4.37
8	66.39	4.14
9	68.67	3.67
10	69.46	4.42
11	69.39	4.58
12	67.97	4.06
13	66.27	3.85
14	67.96	4.91
15	62.30	4.07

Table 4.13: Long Short-Term Memory on fine estimate daily prediction

The experimentation on LSTM's fine estimation on daily time frame is shown in the table 4.13 and figure 4.11. The number of hidden nodes obtained from the LSTM's coarse estimation on daily time frame is 10 hidden nodes, therefore the experiment range of hidden nodes is 5 to 15 hidden nodes. The experiments show that the highest prediction accuracy is 72.91% at 7 hidden nodes. Also the prediction accuracy drops as number of nodes increases.

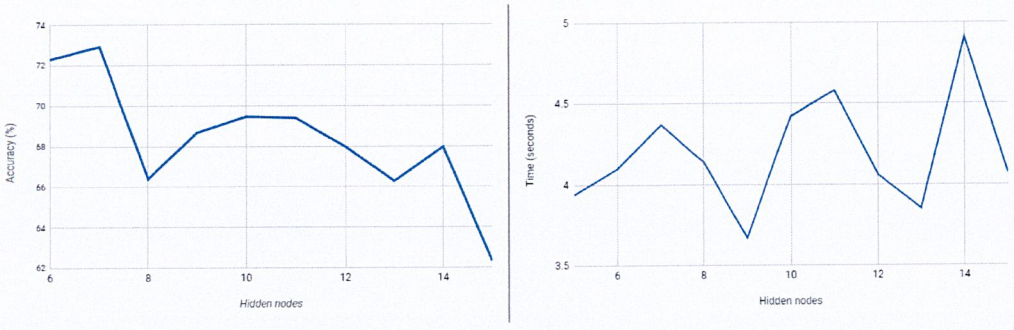

Figure 4.11: LSTM fine estimate daily prediction accuracy (left) and running time in seconds (right) on 60-minute dataset

Number of hidden nodes (node)	LSTM accuracy (%)	Time Complexity (sec)
5	69.98	3.94
6	66.21	3.90
7	65.81	4.01
8	64.85	4.17
9	62.17	4.12
10	63.45	4.11
11	65.05	3.79
12	60.53	3.78
13	63.49	3.82
14	63.09	4.50
15	61.93	3.89

Table 4.14: Long Short-Term Memory on fine estimate weekly prediction

The experimentation on LSTM's fine estimation on weekly time frame is shown in the Table 4.14 and figure 4.12. The obtained number of hidden nodes from the LSTM's coarse estimation on weekly time frame is 10, therefore the experiment range is 5 to 15 hidden nodes. The experiments show that the highest prediction accuracy is 69.98% at 5 hidden nodes. Also the prediction accuracy drops as number of nodes increases.

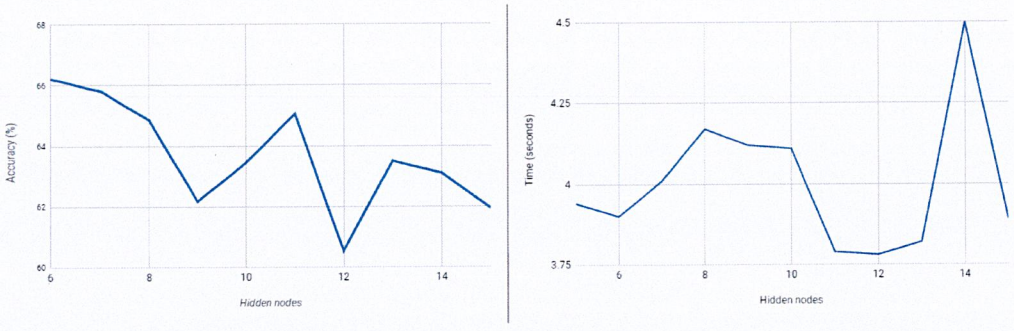

Figure 4.12: LSTM fine estimate weekly prediction accuracy (left) and running time in seconds (right) on 60-minute dataset

4.6 Observation

After experimentation on ANN's and LSTM's machine learning model, their performance has been measured and compared in the following comparison.

Number of hidden nodes	ANN accuracy (%)	LSTM Accuracy (%)
10	79.81	72.04
20	79.46	63.07
30	80.06	62.09
40	82.23	64.65
50	80.45	55.56
60	81.64	57.35
70	79.98	55.96
80	80.85	56.87
90	80.65	60.47
100	81.99	53.99

Table 4.15: Comparison between Artificial Neural Network and Long Short-Term Memory accuracy on coarse estimate hourly prediction

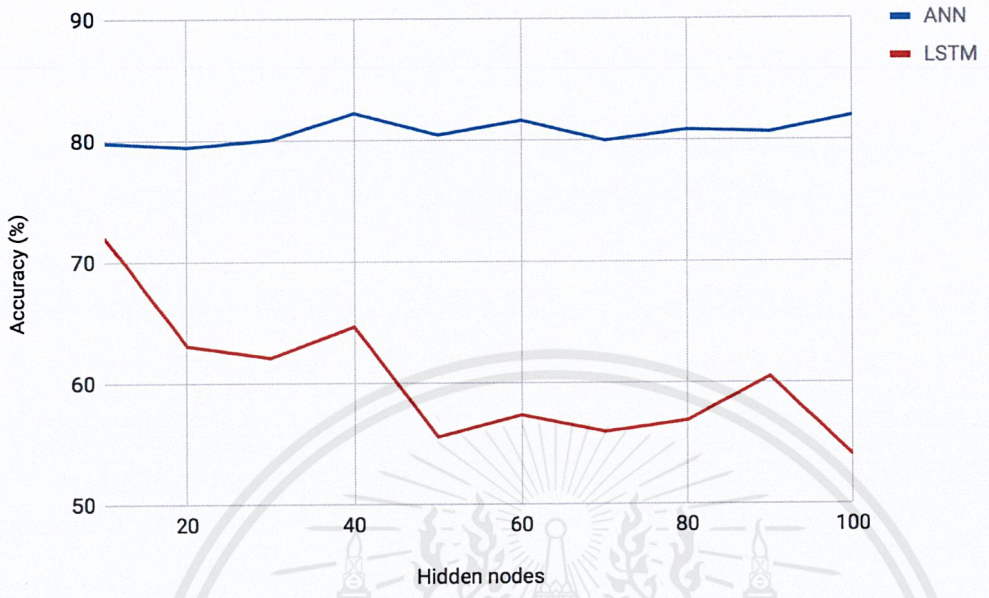

Figure 4.13: Accuracy comparison between Artificial Neural Network and Long Short-Term Memory from 10 to 100 nodes on coarse estimate daily prediction

The comparison between ANN's and LSTM's experiment on coarse estimation on hourly time frame is shown in the Table 4.15. It shows that ANN's prediction accuracy is higher than LSTM's prediction accuracy in all number of hidden nodes and it lies around 80% without a drop. While LSTM's prediction accuracy reached its peak using only 10 hidden nodes and decreases as the number of hidden nodes increases. This can be shown in figure 4.13.

Number of hidden nodes	ANN accuracy (%)	LSTM Accuracy (%)
10	73.57	74.61
20	73.00	68.40
30	76.17	66.55
40	76.13	63.15
50	77.43	62.52
60	76.80	62.56
70	77.75	60.15
80	76.44	62.20
90	77.79	60.98
100	76.29	62.95

Table 4.16: Comparison between Artificial Neural Network and Long Short-Term Memory accuracy on coarse estimate daily prediction

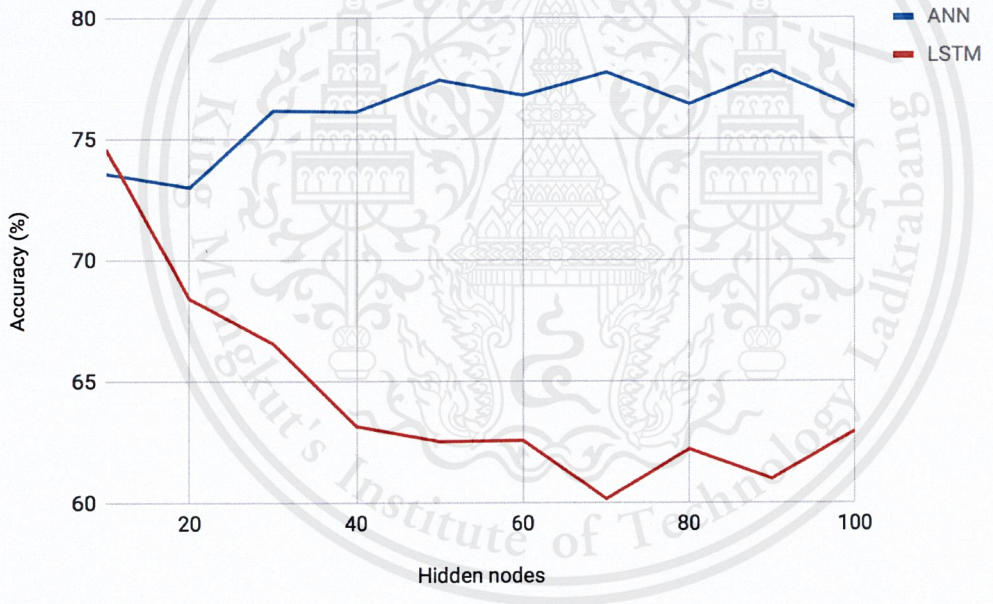

Figure 4.14: Accuracy comparison between Artificial Neural Network and Long Short-Term Memory from 10 to 100 nodes on coarse estimate daily prediction

Similarly the daily time frame coarse experimentation shown in Table 4.16, shows that ANN's prediction accuracy is higher than LSTM's prediction accuracy except on 10 hidden nodes in which LSTM's prediction accuracy is slightly higher than ANN's prediction accuracy. As shown in the figure 4.14,

ANN's prediction accuracy lies on the same range from 76 to 78 % with a little swing, while LSTM's prediction accuracy is constantly dropping as the number of hidden nodes increases.

Number of hidden nodes	ANN accuracy (%)	LSTM Accuracy (%)
10	72.22	66.05
20	72.30	58.41
30	72.42	57.09
40	71.74	59.77
50	71.38	60.89
60	72.18	58.73
70	72.34	55.60
80	72.30	56.16
90	71.58	56.00
100	72.06	56.00

Table 4.17: Comparison between Artificial Neural Network and Long Short-Term Memory accuracy on coarse estimate weekly prediction

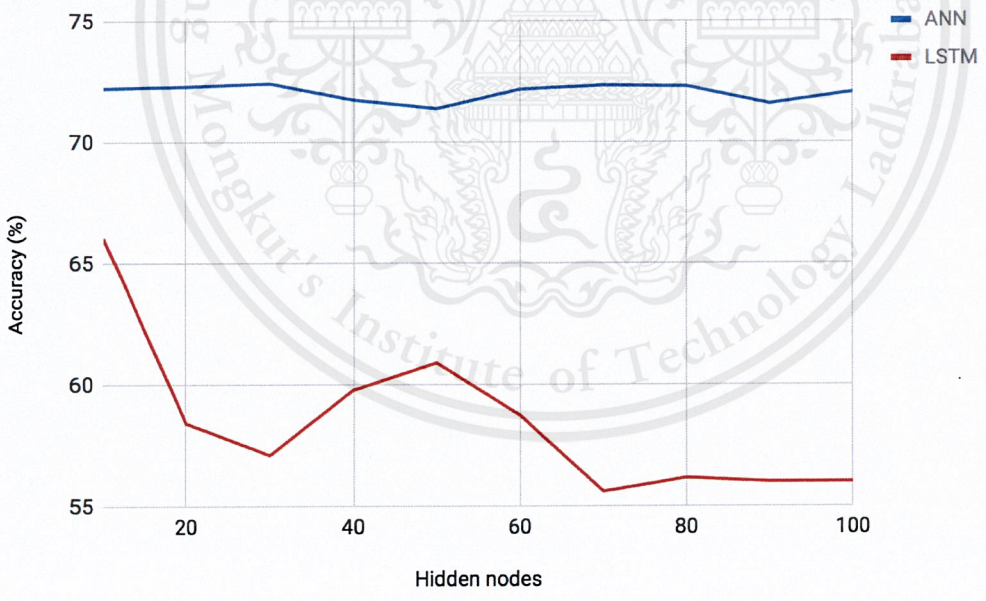

Figure 4.15: Accuracy comparison between Artificial Neural Network and Long Short-Term Memory from 10 to 100 nodes on coarse estimate weekly prediction

The result of experimentation of coarse estimation on weekly time frame for both machine learning model is similar to the result of hourly and daily time frame as shown in Table 4.17. That is ANN's prediction accuracy is higher than LSTM's prediction accuracy which is around 72% accuracy. On the other hand, LSTM's prediction accuracy gradually dropped after 10 hidden nodes and stopped around 55% accuracy as figure 4.15.

According to all coarse estimation experiment above (i.e., hourly, daily, weekly), ANN accuracy graph is gradually drops down as time frame is augmented. Surprisingly, LSTM does not follows ANN's accuracy trend but its accuracy peaked at daily time frame.

From all experiments, Artificial Neural Network highest accuracy is around 40 hidden nodes with 82.23% as shown in Table 4.3 with hourly time frame. This surpassed the project objective on 70% accuracy using Long Short-Term Memory. However, precise number of hidden nodes is still required. After conduct another Artificial Neural Network experiment on specific number of hidden nodes between 35 to 45 nodes, the model with 43 hidden nodes give the highest accuracy result of 82.50% as shown in Table 4.6.

The accuracy of Long Short-Term Memory is peaked at 10 hidden nodes with 74.61% which has great gap over the second one as shown in Table 4.10 with daily time frame. Further experimentation is conducted, starting from 5 to 15 hidden nodes. The model with seven hidden nodes obtained the highest accuracy of 72.91% as shown in Table 4.13.

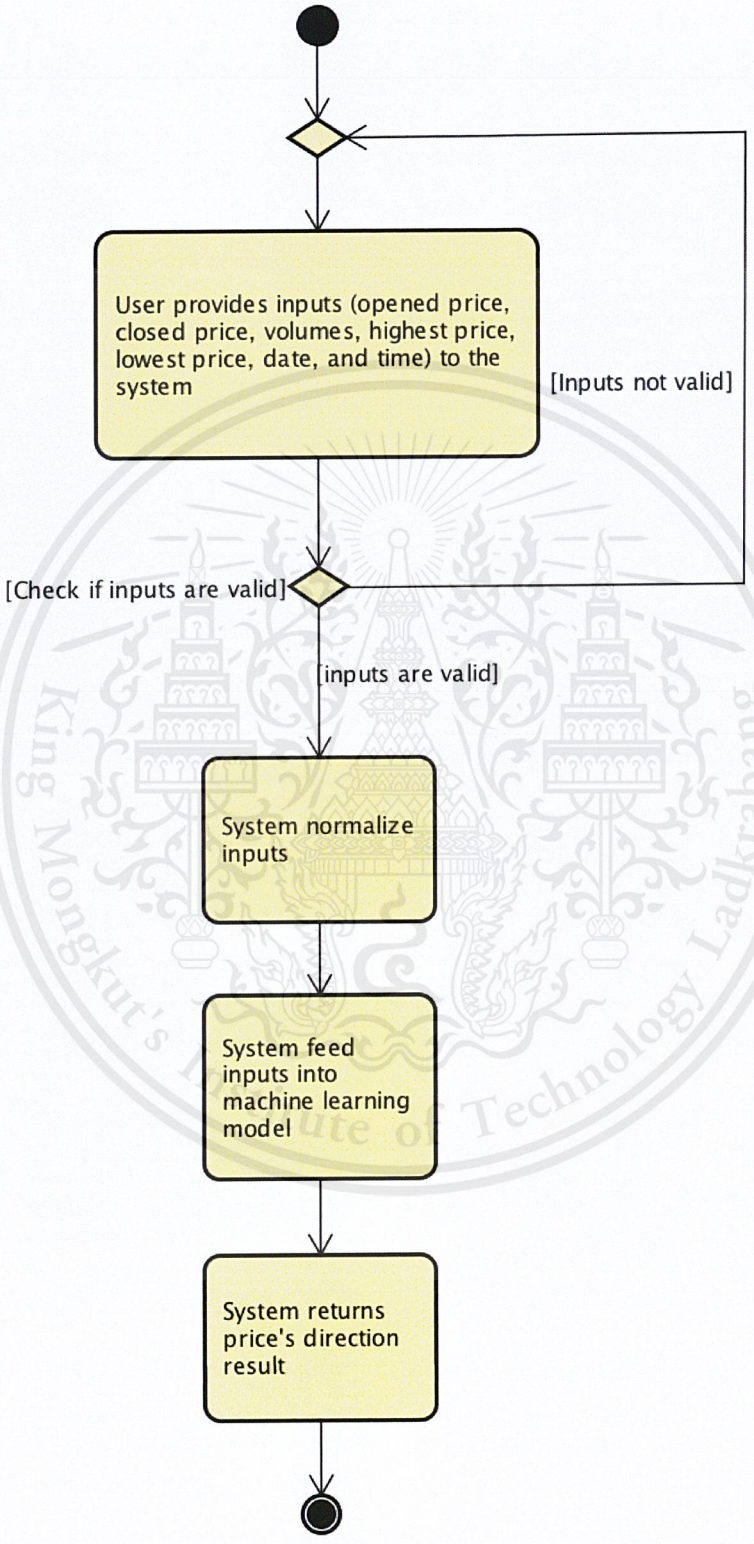

Figure 4.16: Activity diagram show operation flows of the application

Chapter 5

Conclusion and Discussion

The problem of predicting the movement of stock market prices is that there are many factors that affected the direction of stock. However, the accuracy of the prediction is important to determine the best trading path. The results of the forecast can affect both the investors and the future of that stock on how they invest on the assets. There are many techniques that are used to analyze the stock price's direction, machine learning is one of the most popular techniques. Therefore, this study focuses on predicting TFEX price's direction using Long Short-Term Memory (LSTM) based on daily data from year 2001 to 2017.

The selected features, that are extracted from the historical TFEX price's direction as input variables, are used to generate the prediction rules. The output of the models is the price's direction of the TFEX market in either upward or downward direction on the next period of the given timeframe. These experiments use Artificial Neural Networks or ANN as the based line for performance comparison with LSTM. The results show that the highest prediction accuracy ANN could achieved for hourly timeframe is 82.5% using 43 hidden nodes while the best LSTM could achieved for hourly is 73.54% accuracy using 7 hidden nodes on hourly prediction. For the daily timeframe ANN once again out performed LSTM by achieving the prediction accuracy of 78.5% using 95 hidden nodes while LSTM only achieved prediction accuracy of 72.29% using 7 hidden nodes. Lastly, the weekly timeframe, ANN also out performed LSTM with the prediction accuracy of 72.78% using 28 hidden nodes. LSTM on the other hand achieved the prediction accuracy of 69.98% using 5 hidden nodes. Looking at the results of the experiments show that ANN is better at modelling complex relationship between input features such as price, volumes, and technical indicators and the price's direction in all timeframes as compare to LSTM. However looking at LSTM

experiments results alone show that despite of the timeframe getting larger LSTM perform best at daily timeframe. This result suggests that LSTM may be more suitable for modelling relationship between input and output when timeframe is neither too little nor too large. In addition, the experiment also shows that ANN is better than LSTM but it does not mean that ANN will be better in long-term prediction since it cannot cope with the vanishing and exploding gradient problem as LSTM does. The ANN accuracy trend is better at higher number of hidden nodes while LSTM better at lower number of hidden nodes even though it takes a longer time than ANN did. Considering the training time, ANN uses around 1.5 seconds to train while the LSTM model uses around 4 seconds.

In theory, increasing nodes in LSTM should not lower the accuracy by this much, in fact, it should have risen a bit. The reason behind the decline of accuracy trend of LSTM as timeframe getting bigger may come from the dataset used for training is overfitting. Overfitting data occur when the data, in supervised learning, has biasedly labeled on some result which in fact should be balanced.

In future studies, more appropriate input variables will be used to improve the performance of the models. Also, a new approach for TFEX price's direction prediction might be constructed using several techniques along with the hidden layer modification in order to obtain better result.

References

[1] A. Bernal, R. Pidaparthi, and S. Fok. *Financial Market Time Series Prediction with Recurrent Neural Networks* [Online]. Available: http://cs229.stanford.edu/proj2012/BernalFokPidaparthi-FinancialMarketTimeSeriesPredictionwithRecurrentNeural.pdf

[2] A. Subasi and G. Senyurt. *Stock Market Movement Direction Prediction Using Tree Algorithms* [Online]. Available: http://eprints.ibu.edu.ba/1187/1/41.%20Stock%20market%20movement%20direction%20prediction%20using%20tree%20algorithms.pdf

[3] A.A. Assaf, E. Alnagi, and Q.A. Al-radaideh. *Predicting Stock Prices Using Data Mining Techniques* [Online]. Available: http://acit2k.org/ACIT/2013Proceedings/163.pdf

[4] G. Ericson, J. Martens, J. Takaki, R. Astala, and T. Petersen. *Normalize Data* [Online]. Available: https://msdn.microsoft.com/en-us/library/azure/dn905838.aspx

[5] H. Jia. (2016, August 28) *Investigation Into The Effectiveness Of Long Short Term Memory Networks For Stock Price Prediction* [Online]. Available: https://arxiv.org/pdf/1603.07893.pdf

[6] H. Patel, M. Roondiwala, and S. Varma. *Predicting Stock Pices Using LSTM* [Online]. Available: https://www.ijsr.net/archive/v6i4/ART20172755.pdf

[7] H. Raichandani, P. Kumbhare, and R. Makhija. *Stock Market Prediction* [Online]. Available: http://www.cse.scu.edu/~mwang2/projects/Predict_stockMarket_16w.pdf

[8] K. Agarwal. *Advantages Of Trading Futures Over Stocks (APPL)* [Online]. Available: http://www.

investopedia.com/articles/active-trading/032515/advantages-trading-futures-over-stocks.asp

[9] K.J. Kim. *Financial Time Series Forecasting Using Support Vector Machines* [Online]. Available: http://lcsd2.svms.org/finance/Kim2003.pdf

[10] M. Hansson. *On stock return prediction with LSTM networks* [Online]. Available: http://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=8911069&fileOId=8911070

[11] *MACD (Moving Average Convergence/Divergence Oscillator* [Online]. Available: http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators:moving_average_convergence_divergence_macd

[12] S. Madge. *Predicting Stock Price Direction using Support Vector Machines* [Online]. Available: https://www.cs.princeton.edu/sites/default/files/uploads/saahil_madge.pdf

[13] S. Padhy and S.P. Das. *Support Vector Machines for Prediction of Futures Prices in Indian Stock Market* [Online]. Available: https://pdfs.semanticscholar.org/5dd7/05f0f2e2bdc64472244f29001f8a2d32fc38.pdf?_ga=2.256039435.1279440698.1528559798-152142410.1528559798

[14] S.B. Aruoba and J. Fernández-Villaverde. *A Comparison of Programming Languages in Economics. A Comparison of Programming Languages in Economics* [Online]. Available: https://www.sas.upenn.edu/~jesusfv/comparison_languages.pdf

[15] T. Hellström. *A Random Walk through the Stock Market* [Online]. Available: http://www.e-m-h.org/Hell98.pdf

[16] V. Boonjing, M. Intrachot, and S. Intakosum. *Predicting SET50 Index Trend Using Artificial Neural Network And Support Vector Machine* [Online]. 9101, pp.404-414. Available: http://dl.acm.org/citation.cfm?id=2986376

[17] V. Boonjing, M. Intrachot, and S. Intakosum. *Artificial Neural Network and Genetic Algorithm Hybrid Intelligence for Predicting Thai Stock Price Index Trend* [Online]. Available: https://www.hindawi.com/journals/cin/2016/3045254/

[18] V. Joaquin. *OpenML: exploring machine learning better, together* [Online]. Available: `https://www.openml.org/`