Time Series Analysis for Tesla Stock Prediction Using Machine Learning

Nguyen Tien Dat[1], Vo Quoc Huy[2], Bui Minh Duy[3], Cao My Duyen[4], and Nguyen Thien Kim[5]

[1] University of Information Technology, `22520226@gm.uit.edu.vn`
[2] University of Information Technology, `22520586@gm.uit.edu.vn`
[3] University of Information Technology, `22520311@gm.uit.edu.vn`
[4] University of Information Technology, `22520347@gm.uit.edu.vn`
[5] University of Information Technology, `22520729@gm.uit.edu.vn`

Abstract. Machine learning and deep learning techniques are renowned for playing a pivotal role in Business Analytics by leveraging time-series datasets to provide businesses with insights that assist in decision-making, improve performance, and optimize processes. With the rapid and diverse development of algorithms, there arises a critical need to scrutinize their efficacy within specific domains. This paper addresses this need by utilizing and evaluating six distinct algorithms - ARIMA, SVR, RNN, ResNet, ResNetLSTM - for forecasting the stock price of Tesla Inc. based on historical daily stock price records. The objective of this study is to analyze and predict the trajectory of Tesla's stock prices, considering recent market challenges such as intensified competition in the electric vehicle sector and broader political uncertainties affecting the financial markets. The study conducts a comparative analysis of the predictive capabilities of these algorithms and presents a detailed methodology explaining the selection and application of each model. Based on the analyses, the study highlights the strengths and limitations of each approach. Through systematic evaluation and comparison, the paper aims to enhance the understanding of the suitability and competence of these algorithms for stock price analysis and prediction in the context of Tesla's volatile market environment.

Keywords: Tesla Stock Prediction · Machine Learning · Deep Learning · Time-Series Forecasting · Linear Regression · ARIMA · ResNet · ResNetLSTM · Stock Price Analysis

I MATERIALS

A Dataset

In this study, we scrutinize the historical stock price data for Tesla Inc. (TSLA), sources directly from the Yahoo Finance API. The dataset includes daily information spanning from June 29th, 2010 to May 23th 2025, covering over 3,000 trading days. Each record includes variables that provide important insights and data for the study, specifically as follows:

- Date: The specific day the stock price was recorded.
- Open: Tesla's stock price at the start of the trading day.
- High: The highest price Tesla's stock reached during that day's trading.
- Low: The lowest price Tesla's stock dropped to during that day's trading.
- Close: Tesla's stock price when the market closed for the day.
- Adj Close (Adjusted Close): The closing price, adjusted to account for significant corporate actions like dividends or stock splits.
- Volume: The total number of Tesla shares that changed hands on that particular day.

This data reflects the daily fluctuations in Tesla's stock price and is used to train models that forecast future price trends.

B Descriptive Statistics

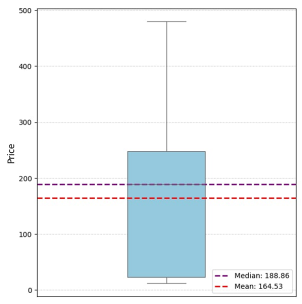

Fig. 1: TSLA stock price's boxplot

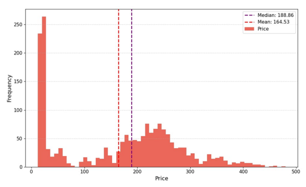

Fig. 2: TSLA stock price's histogram

Table 1: TSLA Descriptive Statistics

Descriptive Statistic	TSLA
Count	1835
Mean	164.525
Std	115.339
Min	11.931
25%	23.432
50%	188.860
75%	248.197
Max	479.860

II METHODOLOGY

A ARIMA

ARIMA stands for Autoregressive Integrated Moving Average. The ARIMA model is using three different concepts: autoregressive (AR) model, moving average (MA) model, and integration, together classified as an ARIMA (p, d, q). It is a quantitative forecasting model over time, where the future value of the predictor variable will depend on the movement trend of that object in the past. The model contains three components/parameters: AR + I + MA. AR is denoted as p, where it shows the weighted linear of sum p values based on ARIMA (p, d, q) terminology. The p-value indicates the number of orders. The formula to denote this AR is shown as:

$$\phi_1 = \phi_1 + \phi_2\delta - \delta x_0 + \phi_3\delta - \delta x_1 + \ldots + \delta e_{t-1} = e_t \tag{1}$$

Where p is used to determine the number of orders of past values; t is the time series; ϕ is the coefficient of the AR model; e is the error term with mean zero and variance σ^2.
MA process is denoted by order q in the ARIMA (p, d, q) classification which shows an error value in Equation (4), it also uses the number of orders in the past values, as denoted in equation:

$$y_t = \mu + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \ldots + \theta_q e_{t-q} + e_t \tag{2}$$

Where t is the time series; θ is the slope coefficient; q is the number of orders needed to identify the past values. To identify how many orders are in the calculation of AR, the parameter of q is used; c is the intercept
Integrated or differentiated versions are denoted as d in ARIMA (p, d, q), which is the number of times the time series got different.

$$I(d) = \Delta(y_t) = y_t - y_{t-1} \tag{3}$$

Therefore, The ARIMA(p, d, q) can be represented in the following equation:

$$\begin{aligned} y_t = \mu + \phi_0 + \phi_1\Delta(y_{t-1}) + \ldots + \phi_p\Delta(y_{t-p}) \\ + \theta_1\varepsilon_{t-1} + \ldots + \theta_q\varepsilon_{t-q} + \varepsilon_t \end{aligned} \tag{4}$$

B SVR

SVR extends Support Vector Machines (SVM) to regression tasks. It aims to find a function:

$$f(x) = \langle w, x\rangle + b \tag{5}$$

that predicts target values within a margin ϵ from the true values, while keeping the function as flat as possible. Primal optimization problem:

$$\min_{w,b,\xi,\xi^*} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \tag{6}$$

Subject to:

$$\begin{cases} y_i - \langle w, x_i\rangle - b \le \epsilon + \xi_i \\ \langle w, x_i\rangle + b - y_i \le \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0, \quad i = 1, \ldots, n \end{cases} \tag{7}$$

Where:

- $\|w\|^2$ controls flatness (regularization).
- $C > 0$ balances model flatness and tolerance for errors greater than ϵ.
- ξ_i, ξ_i^* are slack variables for deviations outside the ϵ-tube.

Kernel extension (nonlinear SVR):
By mapping inputs to a higher-dimensional space using a kernel $K(x_i, x_j) = \phi(x_i)^T\phi(x_j)$, the prediction becomes:

$$f(x) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)K(x_i, x) + b \tag{8}$$

C Recurrent Neural Network (RNN)

A Recurrent Neural Network - RNN is a deep neural network designed to process sequential data such as time series, text, audio,... Unlike traditional feed-forward neural networks, RNNs have loops that allow information to persist, making them ideal for tasks where context is crucial. RNNs process sequential data by maintaining a hidden state that is updated at each time step t, according to the formula:

$$h^{(t)} = g_1\left(W_{xh}x^{(t)} + W_{hh}h^{(t-1)} + b_h\right) \tag{9}$$

With the output being:

$$y^{(t)} = g_2\left(W_{hy}h^{(t)} + b_y\right) \tag{10}$$

Where $y^{\langle t\rangle}$ is the output at time step t, W_x is the output weight matrix, and b_x is the output bias term.
Where:

- $x^{(t)}$ is the input vector at time step t.

- $y^{(t)}$ is the output vector at time step t, representing the network's prediction.
- $h^{(t-1)}$ is the hidden state from the previous time step, containing information from prior inputs.
- W_{xh}, W_{hh}, and W_{hy} are weight matrices that transform the inputs and hidden states.
- b_h and b_y are bias vectors.
- g_1 is a nonlinear activation function (e.g., tanh or ReLU) applied to compute the hidden state.
- g_2 is an activation function appropriate for the output task (e.g., softmax for classification).

D Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) architecture that is capable of learning long-term dependencies in sequence data. Traditional RNNs suffer from the vanishing gradient problem, which limits their ability to capture long-term dependencies. LSTM addresses this issue by introducing memory cells and gated mechanisms.

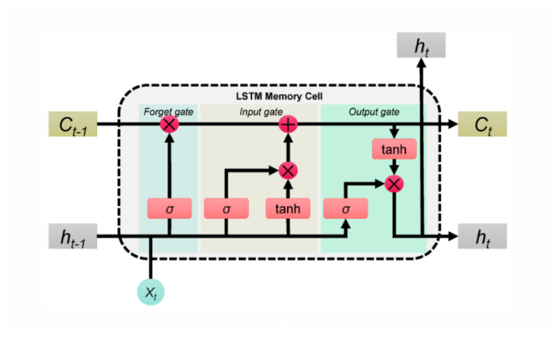

Fig. 3: Long Short-Term Memory

An LSTM cell contains three gates: the input gate, the forget gate, and the output gate. These gates regulate the flow of information and help the network decide what information to keep or discard over time.
The computations inside an LSTM cell at time step t are defined as follows:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{11}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{12}$$

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{13}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{14}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{15}$$

$$h_t = o_t \odot \tanh(c_t) \tag{16}$$

Where:

- x_t is the input at time t
- h_{t-1} is the hidden state from the previous time step
- f_t, i_t, and o_t are the forget, input, and output gates respectively
- c_t is the cell state at time t, and $\tilde{c}_t$ is the candidate value for the cell state
- W_*, U_*, b_* are weight matrices and bias vectors
- σ is the sigmoid activation function
- tanh is the hyperbolic tangent function
- $\odot$ denotes element-wise multiplication

LSTM is particularly well-suited for time-series forecasting tasks such as stock price prediction, as it can capture both short-term fluctuations and long-term trends in sequential data.

E Residual Network (ResNet)

ResNet (or residual network) is a type of deep neural network that is able to perform very well, of with the vanishing gradient as depths grows. It applies residual blocks that have 3 convolutional blocks using different kernel size through shortcut connection that adds initial input to a block output. The ResNet as described in this work consists of three blocks and then is connected to a global average pooling layer and a fully connected layer for the final output.

Residual Block Each residual block uses 1 x 1 convolutions to create three convolutional blocks which are set up as follows:

$$h_1 = \mathrm{ReLU}(\mathrm{BN}(\mathrm{Conv1d}(x))), \tag{17}$$

$$h_2 = \mathrm{ReLU}(\mathrm{BN}(\mathrm{Conv1d}(h_1))), \tag{18}$$

$$h_3 = \mathrm{BN}(\mathrm{Conv1d}(h_2)). \tag{19}$$

After computing $h3$, the value is relayed to a linear unit. Output of the residual block is then calculated by x plus relaying $h3$ while applying the relayed value on a linear unit:

$$y = h_3 + x, \tag{20}$$

$$\hat{h} = \mathrm{ReLU}(y). \tag{21}$$

The convolutional blocks use kernel sizes of 7, 5 and 3. The three residual blocks will have the filter count set to $k_i = \{64, 128, 128\}$.

F ResNetLSTM Model

The architecture of ResNet-LSTM includes ResNet for spatial feature extraction and LSTM for time-series modeling. We adopt ResNet based on ResNet18 to extract features by convolution processes along with residual blocks, which provide a feature tensor as input to the LSTM. The LSTM component incorporates temporal dependencies using 2 layers with 1024 and 256 units each. They utilized dropout to prevent overfitting as well as 2 dense layers for prediction which consisted of 64 and 1 neurons respectively.[1]. In the implemented model, an LSTM layer with 50 units is added to process the transposed ResNet output:

$$x_{\text{lstm}} = x.\text{transpose}(1, 2), \tag{22}$$

Followed by:

$$x_{\text{lstm}}, (h_n, c_n) = \text{LSTM}(x_{\text{lstm}}). \tag{23}$$

The final time step output is selected:

$$x_{\text{final}} = x_{\text{lstm}}[:, -1, :]. \tag{24}$$

The prediction is produced through the application of a fully connected (FC) layer:

$$y = \text{FC}(x_{\text{final}}). \tag{25}$$

III EXPERIMENT RESULT

The evaluation metrics are calculated by measuring errors, such as Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Mean Absolute Scaled Error (MASE), Mean Squared Error (MSE), and Root of Mean Squared Error (RMSE,).

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n} \left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100$$

$$\text{MASE} = \frac{1}{n}\sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{\frac{1}{m-1}\sum_{j=1}^{m-1} |y_{j+1}^{\text{train}} - y_j^{\text{train}}|}$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

where y_i is the actual price, $\hat{y}_i$ is the predicted price, n is the number of predictions, and y^{train} is the training data.

A ARIMA

Input Layer: Univariate time series data — Close price of TSLA stock. Model Order: ARIMA(3,1,2) — determined through auto arima optimization. Parameters:

- p = 3 — autoregressive terms
- d = 1 — differencing order
- q = 2 — moving average terms

Data Preprocessing:

- Logarithmic transformation
- First-order differencing for stationarity
- ADF test for stationarity verification

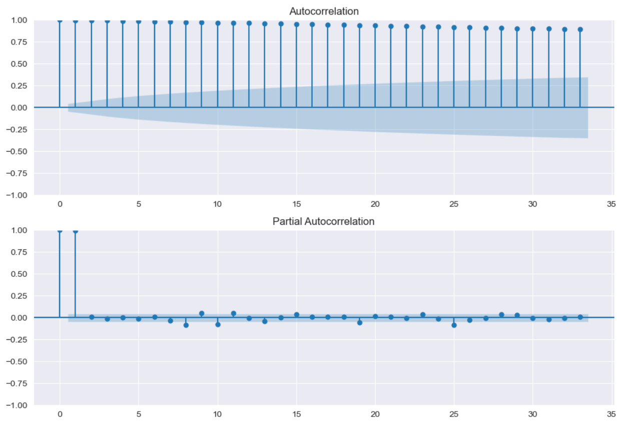

Fig. 4: Autocorrelation

Training Process:

- Train-test split: 80-20 ratio
- Auto ARIMA for parameter optimization
- AIC criterion for model selection

Model Evaluation: Mean Squared Error (MSE) for regression performance.

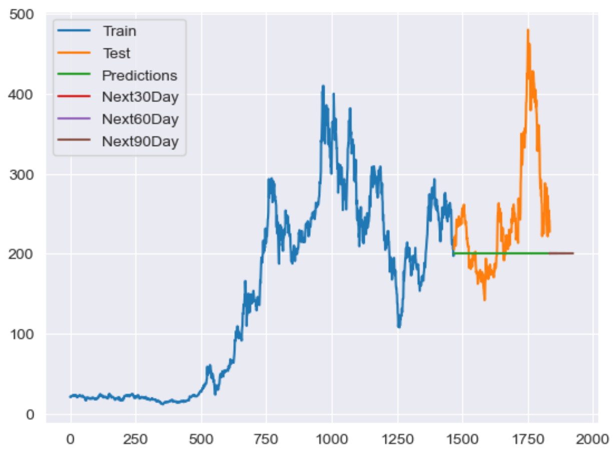

Fig. 5: ARIMA TSLA 8-2

B LSTM

Input Layer: Input shape: $(100, 1)$ — univariate time series with time steps = 100. LSTM Layers:

- First LSTM Layer: 50 units with return sequences=True
- Second LSTM Layer: 50 units with return sequences=True
- Third LSTM Layer: 50 units

Dense Layer (Output Layer): 1 neuron — linear activation for regression output. Data Preprocessing:

- MinMaxScaler: feature range=(0,1)
- Train-test split: 80-20 ratio
- Sequence creation with time step=100

Training Parameters:

- Loss function: Mean Squared Error (MSE)
- Optimizer: Adam
- Epochs: 100
- Batch size: 64
- Validation data: 20% of dataset

Model Evaluation: RMSE, MAPE, and MAE metrics for performance assessment.

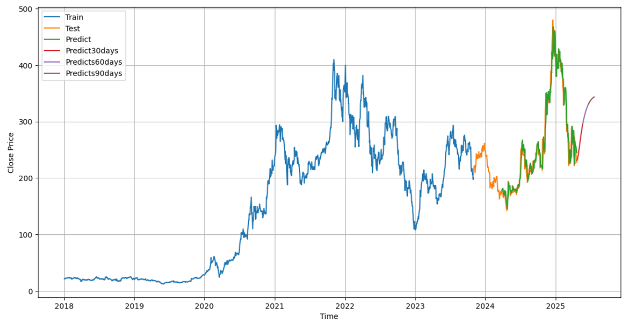

Fig. 6: LSTM TSLA 8-2

C SVR

- SVR with Linear Kernel: `kernel= linear`, $C = 100$
- SVR with RBF Kernel:
  - Configuration 1: $C = 100$, $\epsilon = 0.0001$, $\gamma = 10$
  - Configuration 2: $C = 1{,}500{,}000$, $\epsilon = 10^{-7}$, $\gamma = 100$
- SVR with Polynomial Kernel: `kernel= poly` , $C = 100$, $\epsilon = 0.0001$, degree $= 3$

Table 2: RMSE results of SVR models with different kernels

Model	RMSE
SVR Linear	0.6342
SVR RBF (C=100, ϵ=0.0001, γ=10)	0.2061
SVR RBF (C=1,500,000, ϵ=1e-7, γ=100)	**0.1306**
SVR Polynomial	0.7265

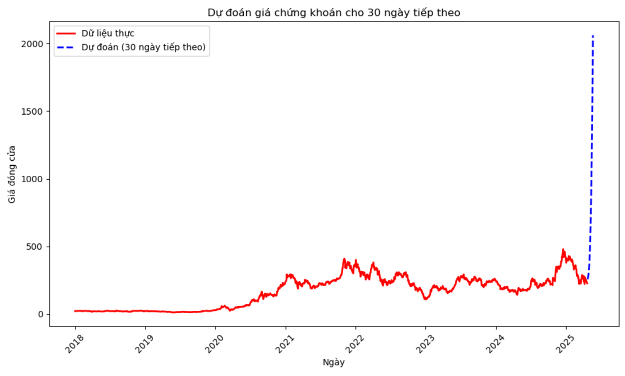

Fig. 7: SVR TSLA 8-2

D RNN

Input Layer: Input shape: $(100, 1)$ — univariate time series with time_steps = 100.
SimpleRNN Layer: 50 units — basic RNN for sequential data processing.
Dense Layer (Output Layer): 1 neuron — linear activation for regression output.
Loss function: Mean Squared Error (MSE) — suitable for regression tasks.
Optimizer: Adam — adaptive learning rate.
Epochs: 24 — full passes over the training data.
Batch size: 64 — samples per gradient update.
Verbose: 1 — show training progress.

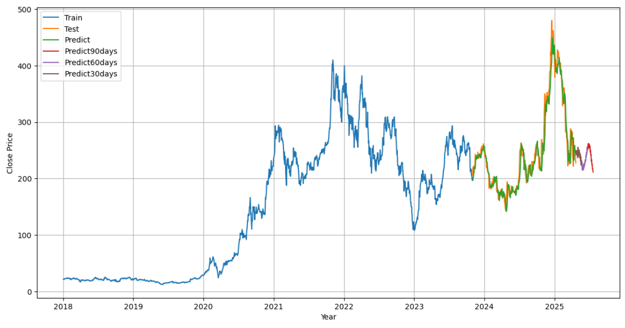

Fig. 8: RNN TSLA 8-2

E ResNet

The design layout of the ResNet model is as follows:

$$\begin{aligned}\text{ResNet} = [&\text{ResBlock1}, \text{ResBlock2},\\ &\text{ResBlock3}, \text{GlobalAvgPooling}, \text{FC}].\end{aligned} \tag{26}$$

The model receives input data and computes it through the three residual blocks, then bounds global average pooling to spatial dimensionality reduction, then ultimately produces the output through a fully connected layer detached from the pooling operation.

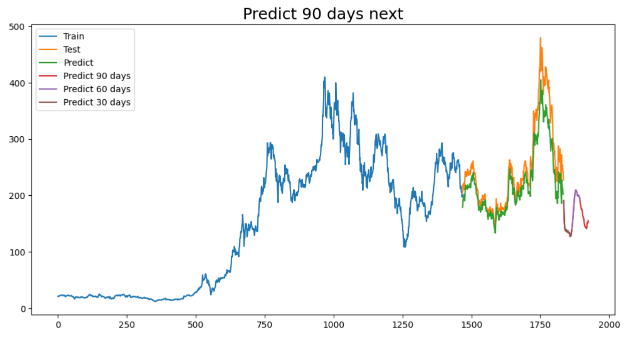

Fig. 9: ResNet TSLA 8-2

F ResNetLSTM

The total architecture is:

$$\text{ResNetLSTM} = [\text{ResBlock1}, \text{ResBlock2}, \text{ResBlock3}, \text{LSTM}, \text{FC}]. \tag{27}$$

Model training was done using the Adam optimizer with a learning rate set at 0.001, mean squared error (MSE) loss, a batch size of 64, and 100 epochs.

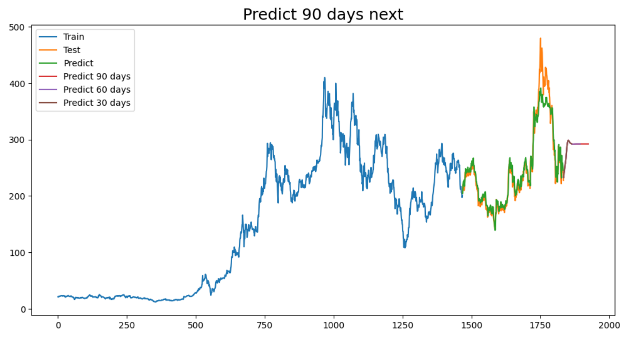

Fig. 10: ResNetLSTM TSLA 8-2

The results are summarized in Table 3.

Table 3: Forecasting performance of ResNet and ResNetLSTM on the test set.

Model	MAE	MAPE (%)	MASE	MSE	RMSE
ResNet	24.8383	8.7462	2651.4954	1253.6210	35.4065
ResNetLSTM	23.3139	7.7512	2488.7656	1254.4014	35.4175

G Evaluation Results

Table 4: Comparison of forecasting performance across models

Model	MAE	RMSE
ARIMA	61.109	88.889
RNN	9.620	13.800
LSTM	263.064	275.176
GRU	251.373	261.875
ResNet	25.420	32.440
ResNet-LSTM	12.553	19.304

IV CONCLUSION

This study conducted a comparative analysis of various machine learning and deep learning models for forecasting Tesla's stock prices using historical time-series data. The evaluation metrics, specifically MAE and RMSE, clearly demonstrate the variability in performance across different models. Traditional statistical models like ARIMA showed the weakest performance, with the highest MAE (61.109) and RMSE (88.889), indicating limited capability in capturing the nonlinear patterns in stock data.
Among deep learning models, ResNet-LSTM achieved the lowest error metrics (MAE: 12.553, RMSE: 19.304), suggesting its superior ability to model both spatial and temporal dependencies. The standalone RNN model also performed well (MAE: 9.620, RMSE: 13.800), surpassing the more complex LSTM and GRU models, which were affected by higher error values. Notably, LSTM and GRU underperformed, with RMSE values above 260, highlighting their limitations in this context.

Overall, the hybrid ResNet-LSTM model emerged as the most effective approach for Tesla stock prediction, balancing both residual feature extraction and sequence learning.

References

1. Kaggle code for lsmt tesla stock forecasting
2. Stock Price Prediction of Tesla, Apple using LSTM, ResearchGate, 2023.
3. Stock Price Prediction, Apple using ResNetLSTM, ResearchGate, 2023.
4. Duong Phi Long, “Lecture Notes” University of Information Technology. Unpublished lecture material