Stock Index Prediction with Machine Learning and Deep Learning Models

Basic Machine Learning Project Report

Hong Thanh Hoai
Vingroup Big Data Institute
v.hoaiht3@vinbigdata.org

Mai Hoang Lan
Vingroup Big Data Institute
v.lanmh1@vinbigdata.org

Nguyen Thi Hong Phuc
Vingroup Big Data Institute
v.phucnth3@vinbigdata.org

Dr Cao Van Loi (Project Advisor)
Vietnam AI Academy
loicv79@gmail.com

Abstract

The volatility of the stock market makes it difficult to predict the stock market with reasonably accuracy. With the rise of artificial intelligence, sophisticated algorithms have shown promising results in stock market prediction. In this project, we implement and compare the performance of several machine learning and deep learning algorithms in predicting the US stock market.

1 Introduction

1.1 Problem Statement

Stock index is commonly used as a performance indicator of a stock market. From an investor's perspective, stock index can be used to predict trends and maximize profits. From a policy maker's perspective, stock index is important in predicting the economy and implementing suitable economic policies.

Stock index is difficult to predict due to its stochastic behavior. The problem is to predict 2 a stock index with reasonably accuracy and performance. The first model is Auto-regressive Integrated Moving Average (ARIMA) which is commonly used in time series forecast. The second model is a 'feature fusion LSTM-CNN' model which combines the insights obtained from LSTM and CNN individually to make more accurately predictions. The model is proposed by Kim & Kim, 2019, and we aim to provide an implementation of their proposed model. The main problem broke down the main problem into 9 sub-problem:

1. Collect data from Yahoo Finance and understand the stochastic nature of stock data.
2. Process the data to remove any empty values in the time series.
3. Transform, normalize the data so it is compatible as the models' input.
4. Implement ARIMA model using Python 3 on Google Colab.
5. Implement LSTM model using Python 3 and Keras library on Google Colab.
6. Implement CNN model using Python 3 and PyTorch on Google Colab.
7. Implement feature fusion LSTM-CNN model using Python 3 on Google Colab.
8. Compare the performance of ARIMA and feature fusion LSTM-CNN models.
9. Propose further work and project extensions.

Data set: Dow Jones stock index daily data (Dow Jones data) from January 2, 2009 to October 16, 2017 is used to train and test the models. In addition, minute-by-minute SPDR S&P 500 ETF Trust data provided by Kim & Kim, 2019 [2] is used to test our models against feature fusion LSTM-CNN model proposed by Kim & Kim, 2019 [2].

1.2 Research Goals

1. Successfully implement ARIMA model for Dow Jones data.
2. Successfully implement LSTM model for Dow Jones data.
3. Successfully implement CNN model for Dow Jones data.
4. Successfully implement feature fusion LSTM-CNN model proposed by Kim & Kim, 2019 [2].
5. Evaluate the models and propose potential improvements to boost the models' accuracy.

2 Auto-regressive Integrated Moving Average (ARIMA)

2.1 Definition

ARIMA is a class of models that present time series based on its own past values (own lags and the lagged forecast errors). ARIMA is suited for non-seasonal time-series.

2.2 Model input

ARIMA model introduces order of differencing into the time series data if the data is non-stationary. Dickey - Fuller test is used to verify data stationarity.

- AR(p) Auto-regression: relationship between a current series and observation over a previous steps. p refers to the use of past values in the regression equation for predicting the current one.
- I(d) Integrated: uses the difference of observations that involves the subtraction of the current values of a series with its previous values d number of times to make the time series stationary.
- MA(q): A moving average components depicts the error of the model as a combination of previous error terms. The order q represents the number of terms to be included in the model.

2.3 Hyper-parameters estimation

- Estimate d
  - Differencing (d) is aimed to make time series stationarity but an over differenced series may harms to the model due to the change of feature of correlation.
  - Auto-correlation function plot (ACF) is used to estimate the value of d. The ACF plot shows the correlation between elements in the time series, whether they are positively correlated or negatively correlated.
  - An acceptable estimation of d is the minimum differencing required to get a near-stationary series which roams around a defined mean and the ACF plot reaches to zeros quickly.
  - In the feature, 1^{st} order differencing is good because 2^{nd} the ACF falls down quickly to negative values.

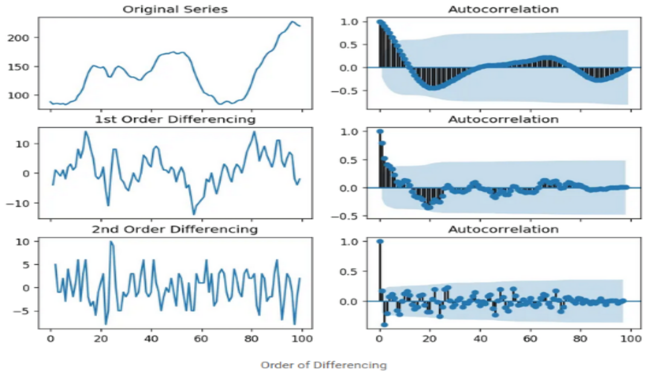


Figure 1: ACF plot with different order of differencing

- Estimate AR (p)
  - Investigating Partial Auto-correction (PACF) for defining p.
  - Set order of AR term (p) equal to as many lags that crosses the significance limit (blue region) in the PACF plot. In the figure, p = 2

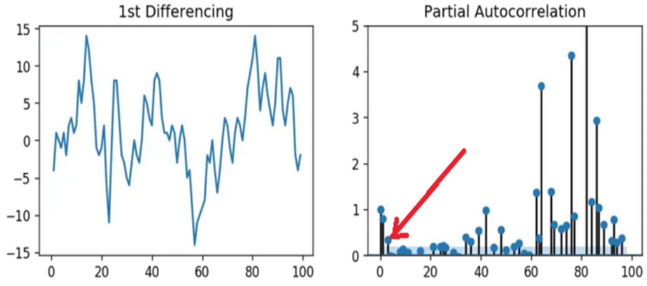


Figure 2: PACF plot for defining p

- Estimate MA (q)
  - Investigating Auto-correlation (ACF) for estimating q
  - ACF represents how many MA terms are required to remove any auto-correlation. In this figure, q = 2 or 3 because there are 3 steps above the blue line.

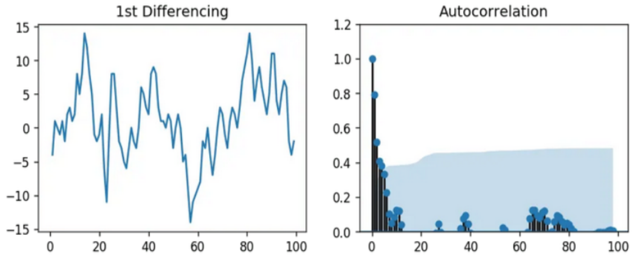


Figure 3: ACF plot for defining q

2.4 Output

Predicted Closing Price of t+5 (days) using the Closing Price of previous 30 days

2.5 Advantage and disadvantage of the ARIMA model

ARIMA forecast are usually more accurate and reliable. ARIMA is running smoothly with one variable only (uni-variate model) and hence cannot exploit leading indicators or explanatory variables. As a matter of fact, the price value have been complicated and commonly related to a set of observable variables.

2.6 Data preparation

As for data preparation, our data set is separated into two sets: a training and a testing sets in which 70% for training, 30% for testing. After the model has finished training, we evaluate the model on the test set. We must ensure that the test set cover a later period in time from the training set, to ensure that the model does not gain from information from future time periods. The model is designed to predict the value from t + 1 (days) to t + 5 (days) using values from time t 30(days) to t(days), as in the case of the stock chart data. To prevent information from the test sets leaking into the training data, training and test data are both scaled in range (0,1).

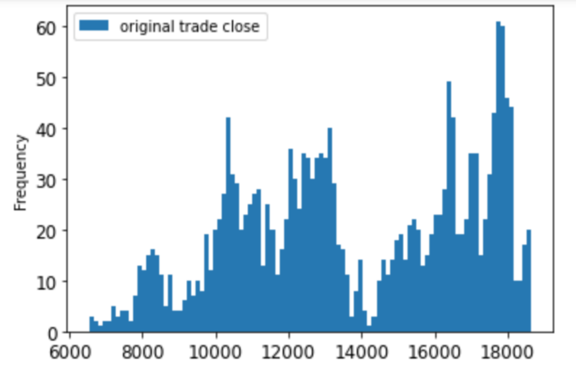


Figure 4: Original Trade Close

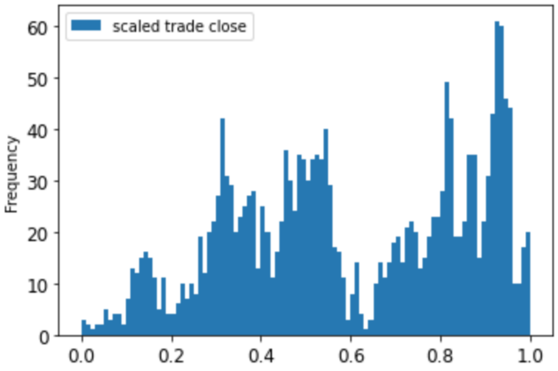


Figure 5: Scaled Trade Close

2.7 Implementation of the ARIMA method

As discussed in previous sections, three parameters are used to help model the major aspects of a times series are: p, d, q. For models with a seasonal component, we use SARIMA but in this case, the pattern would be assumed as a non-seasonal component then we have been applied ARIMA for time series forecasting. Selecting the best parameters for an ARIMA model can be challenging.

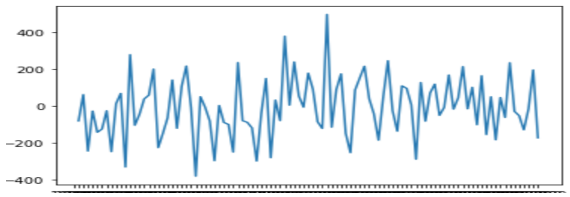


Figure 6: The non-seasonal component

Parameter p: In this model, set order of AR term equal to 1 (p = 1) that crosses the significance limit. By this means, one past values in the regression equation is used for predicting the current one.

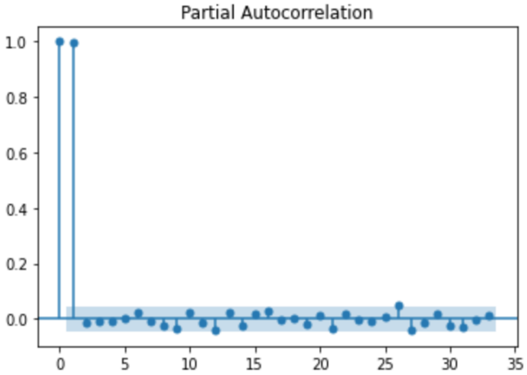
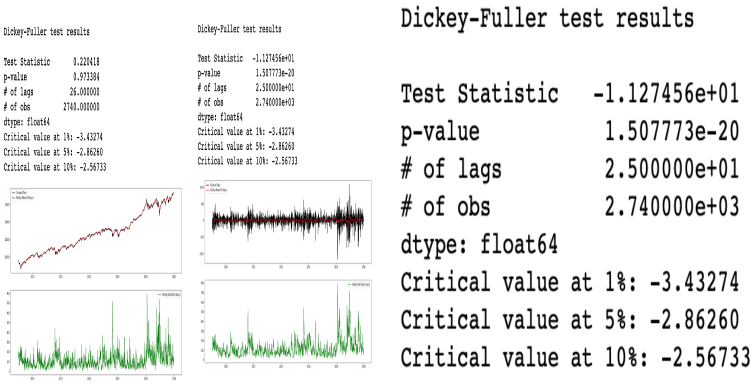


Figure 7: PACF plot of the non-seasonal component

Parameter d: Following the assumption, it would be necessary to check whether the time-series data is stationary and transform to stationary one if it would not be. In our task, differencing the data one time (d=1) have been important to roams around a defined mean.



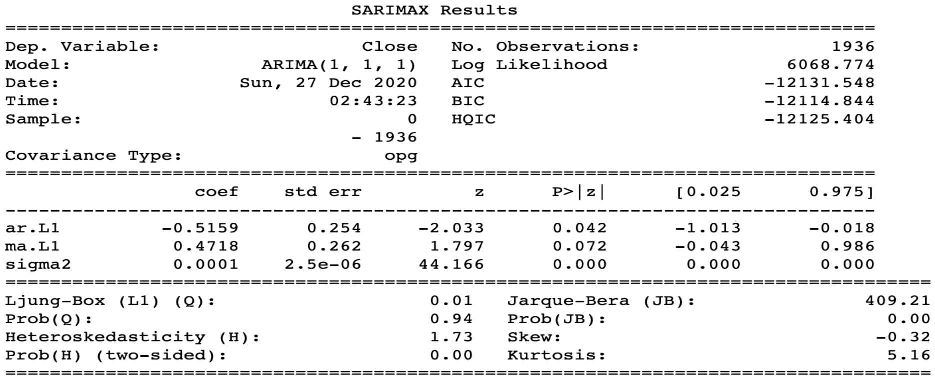
```
Dickey-Fuller test results

Test Statistic        0.220418
p-value               0.973384
# of lags            26.000000
# of obs           2740.000000
dtype: float64
Critical value at 1%: -3.43274
Critical value at 5%: -2.86260
Critical value at 10%: -2.56733
```

```
Dickey-Fuller test results

Test Statistic   -1.127456e+01
p-value           1.507773e-20
# of lags         2.500000e+01
# of obs          2.740000e+03
dtype: float64
Critical value at 1%: -3.43274
Critical value at 5%: -2.86260
Critical value at 10%: -2.56733
```

```
Dickey-Fuller test results

Test Statistic   -1.127456e+01
p-value           1.507773e-20
# of lags         2.500000e+01
# of obs          2.740000e+03
dtype: float64
Critical value at 1%: -3.43274
Critical value at 5%: -2.86260
Critical value at 10%: -2.56733
```

Figure 8: p-value for Non-stationary data and Stationary data

Parameter q: The p-value help us figure out the best parameters. In the figure, q=1 are well-fitted with the training model.

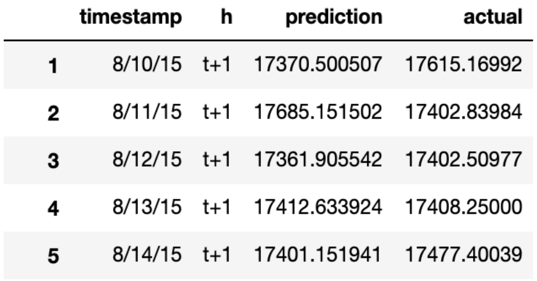


```
                               SARIMAX Results
==============================================================================
Dep. Variable:                  Close   No. Observations:                 1936
Model:                 ARIMA(1, 1, 1)   Log Likelihood                6068.774
Date:                Sun, 27 Dec 2020   AIC                         -12131.548
Time:                        02:43:23   BIC                         -12114.844
Sample:                             0   HQIC                        -12125.404
                               - 1936
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.5159      0.254     -2.033      0.042      -1.013      -0.018
ma.L1          0.4718      0.262      1.797      0.072      -0.043       0.986
sigma2         0.0001    2.5e-06     44.166      0.000       0.000       0.000
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):               409.21
Prob(Q):                              0.94   Prob(JB):                         0.00
Heteroskedasticity (H):               1.73   Skew:                            -0.32
Prob(H) (two-sided):                  0.00   Kurtosis:                         5.16
===================================================================================
```

Figure 9: Summary of ARIMA result

As seen in figure above, the p-value of ma.L1 does not exceed 0.05 (95% confident interval) so the value of q parameter would be equal 1.

2.8 Evaluate the model



	timestamp	h	prediction	actual
1	8/10/15	t+1	17370.500507	17615.16992
2	8/11/15	t+1	17685.151502	17402.83984
3	8/12/15	t+1	17361.905542	17402.50977
4	8/13/15	t+1	17412.633924	17408.25000
5	8/14/15	t+1	17401.151941	17477.40039

Figure 10: Compare predictions to actual Close

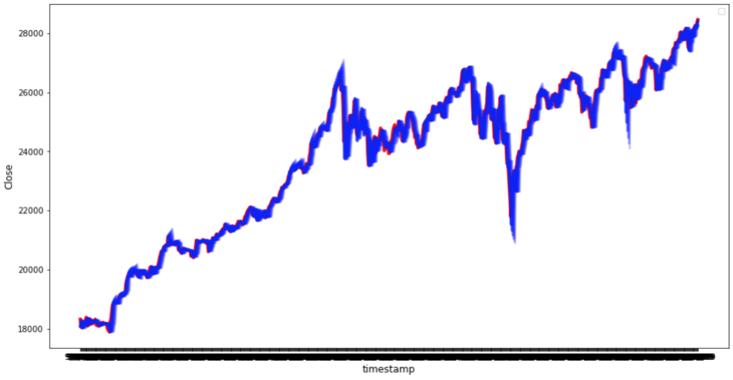


Figure 11: Plot the predictions vs the actual for the next 5 days of the test set from previous 30 days

3 Long Short-term Memory (LSTM)

3.1 Definition

LSTM is an artificial intelligent recurrent neural network (RNN) architecture. RNN allows information to be continuously loop into the network and thus, persist previous information. At every loop in the RNN, some amount of information is ‘drop out’ as an output and some information loops back into the network. The following figure illustrates a simple RNN architecture:

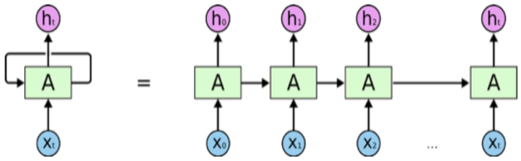


Figure 12: Simple RNN architecture [6]

LSTM is a type of RNN that is capable of learning long-term dependencies. Due to the property, it is powerful in processing, classifying and making predictions for times series data. The following figure shows a single layer LSTM model with hyperbolic tangent $\tanh$ as the activation function.

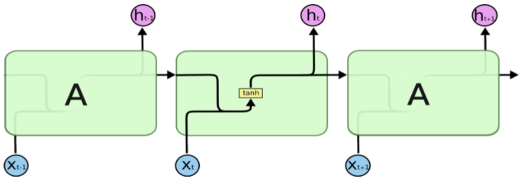


Figure 13: A single layer LSTM model

3.2 Data preparation

To normalize the data, we take the logarithm of the ratio of 2 consecutive daily Close values:

$$\log\left(\frac{Close_t}{Close_{t-1}}\right)$$

In terms of prediction window, we use previous 30 days data to predict the Close index at day $t + 35$.

Combine with data normalization, the following sequence is an input for the LSTM model:

$$\log\left(\frac{Close_t}{Close_{t-1}}\right), \log\left(\frac{Close_{t+1}}{Close_t}\right), \log\left(\frac{Close_{t+2}}{Close_{t+1}}\right), \log\left(\frac{Close_{t+3}}{Close_{t+2}}\right), \log\left(\frac{Close_{t+4}}{Close_{t+3}}\right)\dots$$

The data set downloaded from Yahoo Finance does not contain missing values, so process for missing values is not needed. Data set is divided into 2 sets: training and test sets with ratio 8:2.

3.3 Model input

The model’s input is the sequence of $\log$ shown above.

3.4 Model output

The model’s output is a prediction for $\log\left(\frac{Close_{t+33}}{Close_{t+28}}\right)$

3.5 Implementation of the LSTM model

The LSTM model is implemented on Google Colab using Python 3 and Keras library for deep learning. The implementation code includes 3 main parts. The first part implements data processing and normalization including loading Dow Jones data set with pandas library. Data normalization is done with logarithm function in numpy library.

For the second part, the model’s architecture and hyper-parameters are defined. The LSTM model contains 2 smaller parts: a single layer and 3 fully connected layers. The output from the single layer is then passed through 3 fully connected layers. The important hyper-parameters of the LSTM model includes: drop out rate, number of epoch, regularizer and batch size.

After trial and error optimization, the following hyper-parameters produce the lowest error for the Dow Jones data set: drop out rate for all layers of 0.8, number of epoch equals 20, l2 regularize, 'Adam' optimizer and batch size of 128.

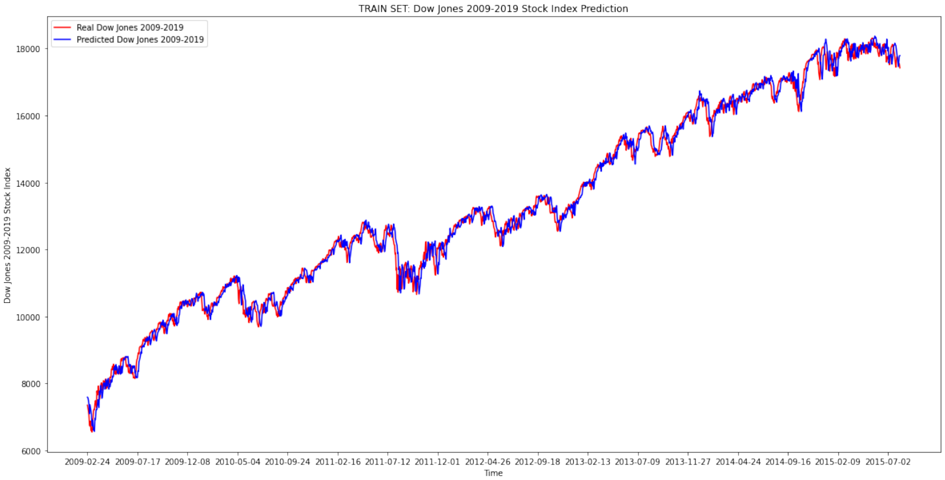


Figure 14: Plot of LSTM model for the training set

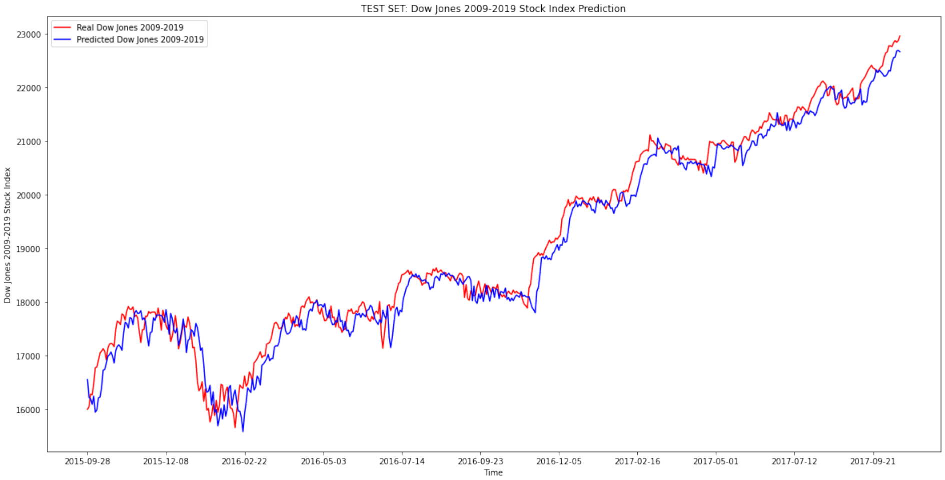


Figure 15: Plot of LSTM model for the test set

3.6 Model Evaluation

The root mean square error (RMSE) of the training, validation and test data are 251, 271, 520 respectively. Due to the stochastic nature of the data, if predicting for shorter time period, the prediction's error will decrease. In terms of run time, the model completely runs under 1 minute which is reasonable for deployment. The following plot shows the LSTM fit for training and testing data set.

If we use data provided by Kim & Kim, 2019 [2], the LSTM model produces slightly better result compare to the implementation done by Kim & Kim, 2019 [2]. The model proposed by Kim & Kim, 2019 [2] has RMSE of 0.119. The model implemented by us has RMSE of 0.111.

4 Convolutional Neural Network (CNN)

4.1 Definition

Convolutional Neural Network
Convolutional Neural Network is quite similar to what neural network like Perceptron or Multi-Layer Perceptron does. The difference here is we don't need to flatten the input image (that creates a large number of weights). For instance, if we have the input image with size 112 × 112, we need to take 12544 operations for weights. With CNN, the neurons only have a small set of weights (5 × 5=25, with 5 is kernel size) that were applied a whole bunch of small subsets of the image of the same size. It takes in the 'local' features found in the previous hidden layer rather than pixel images. Moreover, the final output layer will reduce the full image into a single vector of output value.

Convolutional Neural Networks take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. In particular, unlike a regular Neural Network, the layers of a CNN have neurons arranged in 3 dimensions: width, height, depth. In this project, we use images with dimensions 112x112x3 (width, height, depth respectively).

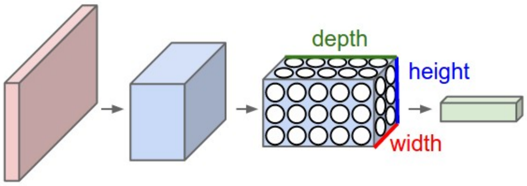


Figure 16: Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels). [4]

Residual block of ResNet
In CNN model, we use three Residual Block (in ResNet structure) that optimizes for data like the stock chart.

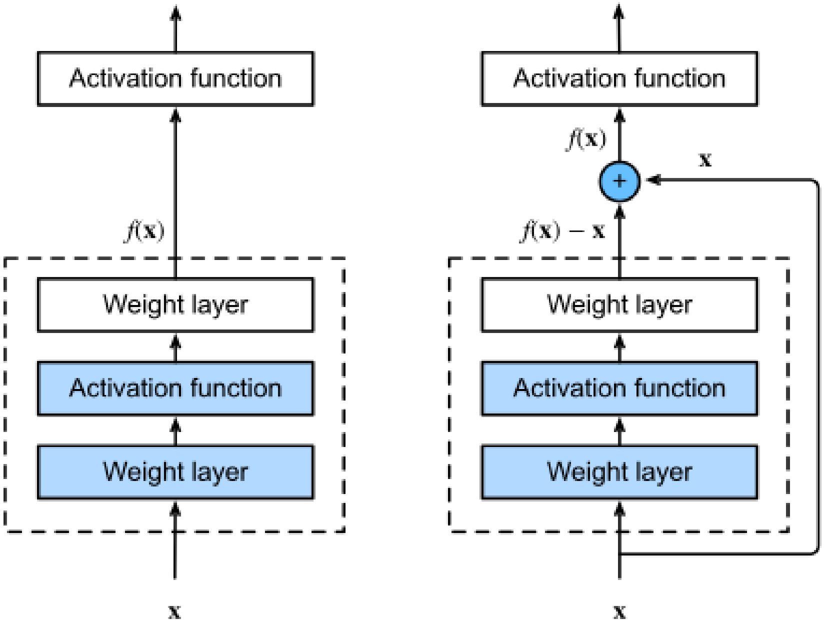


Figure 17: The right figure illustrates the residual block of ResNet, where the solid line carrying the layer input x to the addition operator is called a residual connection (or shortcut connection). With residual blocks, inputs can forward propagate faster through the residual connections across layers. [5]

4.2 Model input

Image size 112x112 with RGB channels (112x112x3) that shows the movement of price in time interval between $t - 30$ and t.

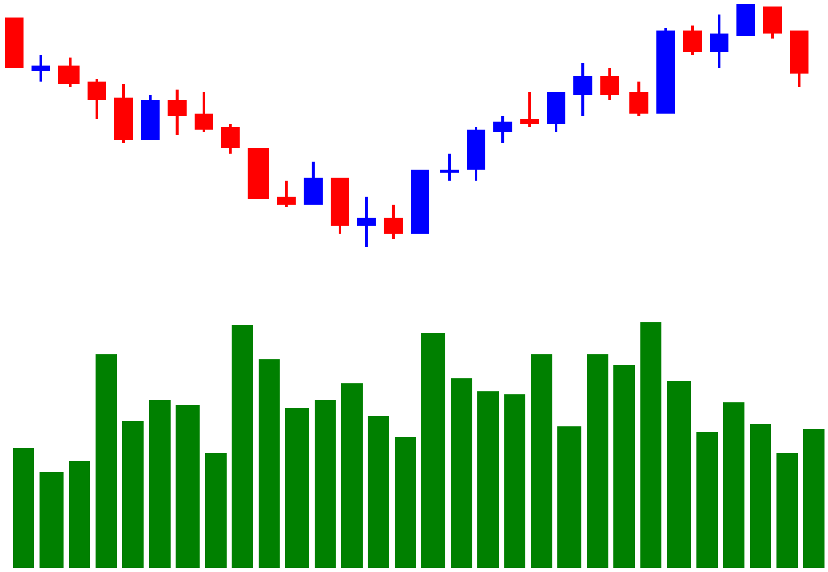


Figure 18: A combination of a candlestick chart and a bar chart that we call a candlebar chart.

4.3 Model output

Predicted Close Price of $t + 5$.

4.4 Purpose in the project

Financial time series data can be used not only as numeric data but also as image data that is transformed in predicting stock prices. Technical analysis uses chart images to predict stock prices by finding patterns in them. Although using an architecture constructed from different representations of the same data will learn duplicate features but will also learn the different characteristics of each architecture, which can improve the prediction accuracy. From this motivation, we want to give CNN a try.

4.5 Advantage and disadvantage of CNN model

The main advantage of CNNs that make them suited to forecasting time series is the ability to use filters to compute dilations between each cell, which in turn allows the neural network to better understand the relationships between the different observations in the time series. However, this just an assumption and it take more time and computational resource for training the model.

4.6 Data Preparation

Convert time series data to image. Each image shows the movement of price in time interval between $t - 30$ and t. Target value is Close Price of $t + 5$.

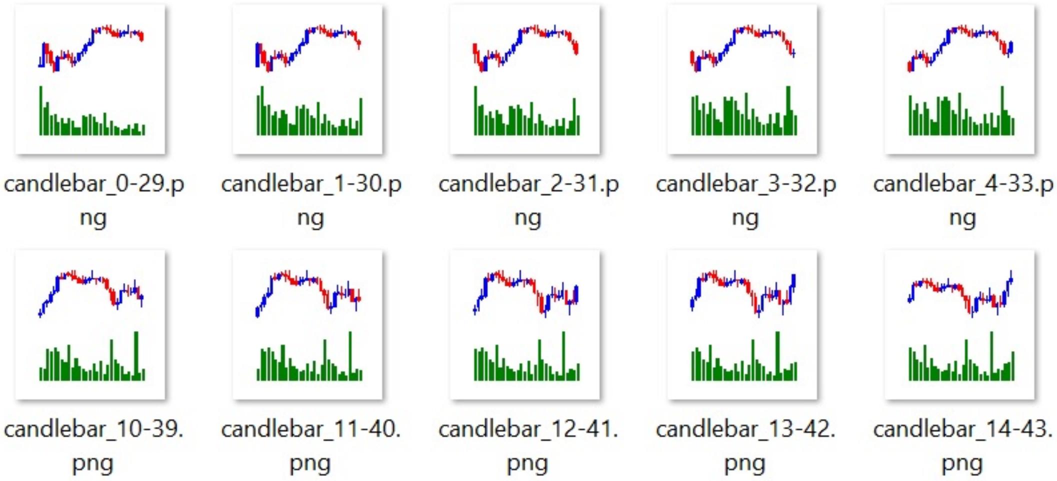


Figure 19: Image data for CNN model.

4.7 Implementation of the CNN model

Layer name	Output size	Kernel size	Pad	Stride	Dropout
conv	32x56x56	7x7	3	2	
max_pool	32x28x28	3x3	2		
res_conv1	128x28x28	1x1	0	1	
		3x3	1	1	
		1x1	0	1	
res_conv2	256x14x14	1x1	0	2	
		3x3	1	1	
		1x1	0	1	
res_conv3	512x7x7	1x1	0	2	
		3x3	1	1	
		1x1	0	1	
average_pool	512x1x1	7x7	0		
fc1	500				0.5
fc2	100				0.5
fc3	25				0.5

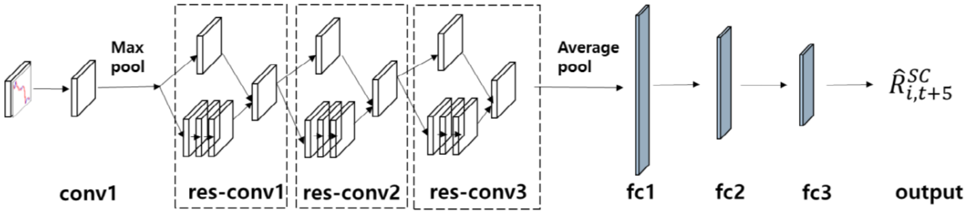


Figure 20: CNN architecture. [2]

4.8 Evaluate the model

4.8.1 Training set

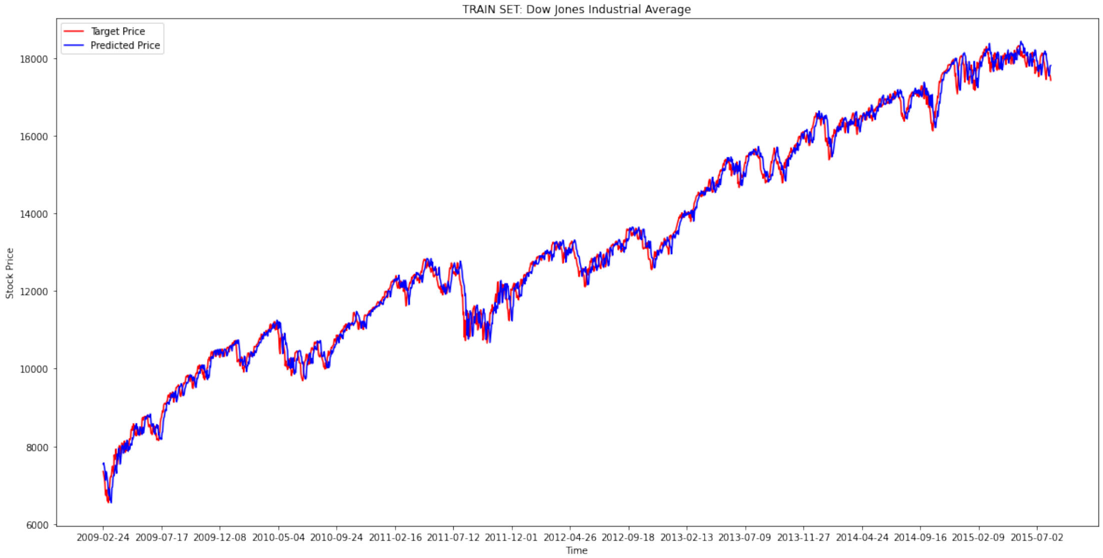


Figure 21: $RMSE = 250.6945 \quad MAE = 191.6990$

4.8.2 Testing set

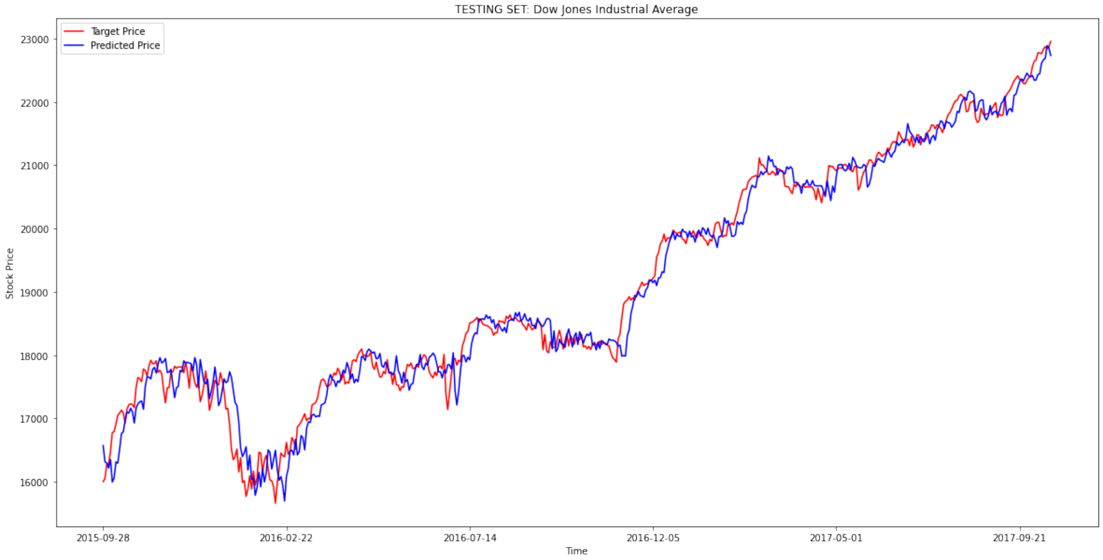


Figure 22: $RMSE = 270.6161 \;\; MAE = 201.3691$

5 Feature fusion LSTM-CNN model

5.1 Definition

We take the fusion of image and temporal features, which come from concatenating each feature, and each separate attribute such as image and temporal features. It is composed of three steps. First, CNN is trained to reduce CNN loss. This process involves learning the graphical features of the chart image. Second, LSTM is trained to reduce LSTM loss. In this process, the temporal feature is learned from the stock time series. Finally, feature fusion LSTM-CNN is trained to reduce feature fusion LSTM-CNN loss. In this process, feature fusion LSTM-CNN shares the parameters that are contained in convolutional layer in CNN and LSTM layer in LSTM.

5.2 Model input

Time series and candlebar chart for time interval between $t - 30$ and t of price.

5.3 Model ouput

Predicted Close Price of $t + 5$.

5.4 Advantage and disadvantage of LSTM-CNN model

We assume that the fusion of the different features comprising the extracted stock chart images and stock time-series data from the same data can improve the training model. In practice, it gives a little better result than each separate model.

5.5 Data Preparation

We have two data streams for each type of data - time series and image chart. The preparation is similar to LSTM and CNN model.

5.6 Implementation of the LSTM-CNN model

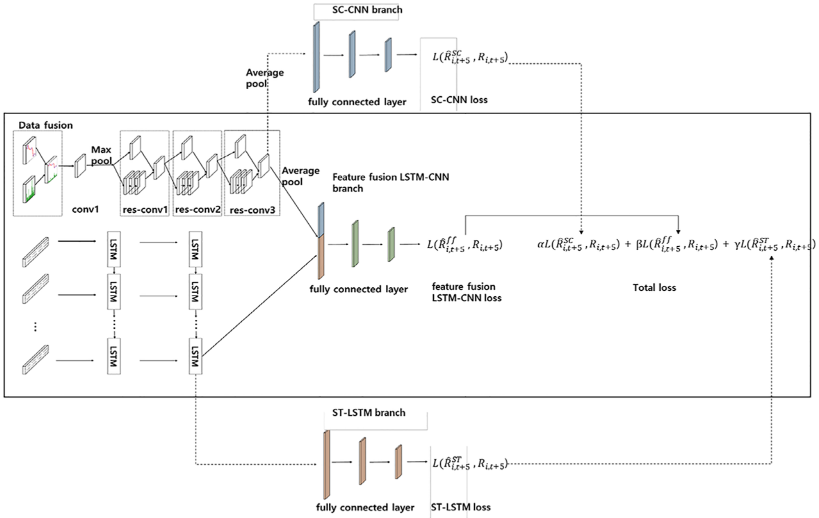

SC-CNN branch
Average pool
fully connected layer
$L(\hat{R}^{SC}_{i,t+5}, R_{i,t+5})$
SC-CNN loss
Data fusion
Max pool
conv1
res-conv1
res-conv2
res-conv3
Average pool
Feature fusion LSTM-CNN branch
LSTM
fully connected layer
$L(\hat{R}^{ff}_{i,t+5}, R_{i,t+5})$
feature fusion LSTM-CNN loss
$\alpha L(\hat{R}^{SC}_{i,t+5}, R_{i,t+5}) + \beta L(\hat{R}^{ff}_{i,t+5}, R_{i,t+5}) + \gamma L(\hat{R}^{ST}_{i,t+5}, R_{i,t+5})$
Total loss
ST-LSTM branch
fully connected layer
$L(\hat{R}^{ST}_{i,t+5}, R_{i,t+5})$
ST-LSTM loss

Figure 23: LSTM-CNN architecture. [2]

We applied the joint-training method. The loss functions of the models are defined as follows:

$$
\begin{gathered}
L\left(\hat{R}^{SC}_{i,t+5}, R_{i,t+5}\right) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{R}^{sc}_{i,t+5} - R_{i,t+5}\right)^2} \\
L\left(\hat{R}^{FF}_{i,t+5}, R_{i,t+5}\right) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{R}^{ff}_{i,t+5} - R_{i,t+5}\right)^2} \\
L\left(\hat{R}^{ST}_{i,t+5}, R_{i,t+5}\right) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{R}^{st}_{i,t+5} - R_{i,t+5}\right)^2} \\
TotalRMSE = \alpha * L\left(\hat{R}^{SC}_{i,t+5}, R_{i,t+5}\right) + \beta * L\left(\hat{R}^{FF}_{i,t+5}, R_{i,t+5}\right) + \gamma * L\left(\hat{R}^{ST}_{i,t+5}, R_{i,t+5}\right)
\end{gathered}
$$

where N is the number of data points, $\hat{R}$ is a predicted return value, and R is a target return value. $L\left(\hat{R}^{SC}_{i,t+5}, R_{i,t+5}\right)$ is the loss function of the CNN model, $L\left(\hat{R}^{FF}_{i,t+5}, R_{i,t+5}\right)$ is the loss function of the feature fusion LSTM-CNN model, and $L\left(\hat{R}^{ST}_{i,t+5}, R_{i,t+5}\right)$ is the loss function of the ST-LSTM model. The values of α, β, and γ are the weights of the total RMSE. These parameters indicate the degree of reflection of each model loss. We set $\alpha = 0.2, \beta = 1, \gamma = 0.2$.

5.7 Evaluate the model

5.7.1 Training set

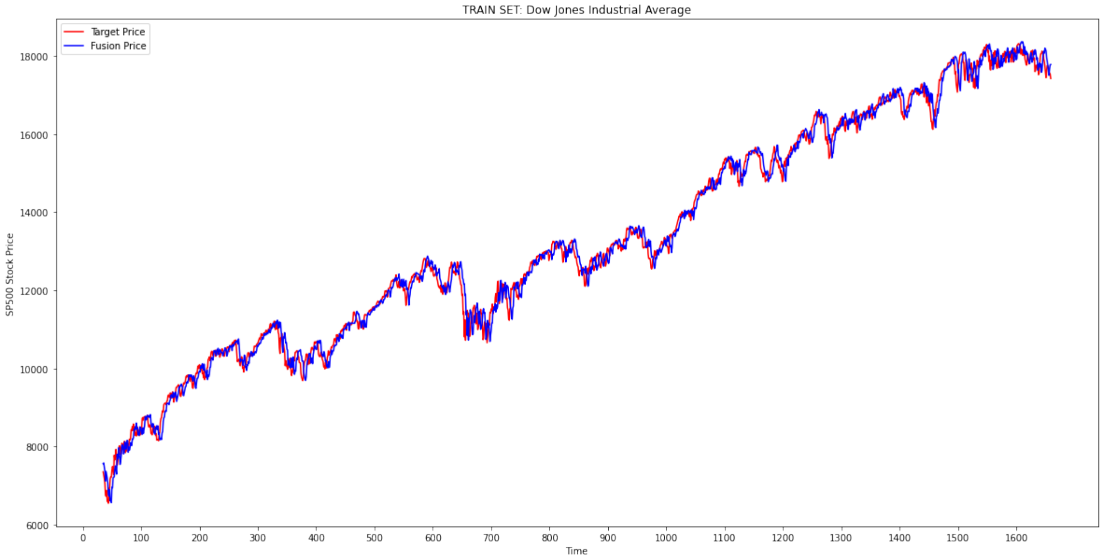


Figure 24: $\mathit{RMSE} = 251.2169 \;\; \mathit{MAE} = 191.4447$

5.7.2 Test set

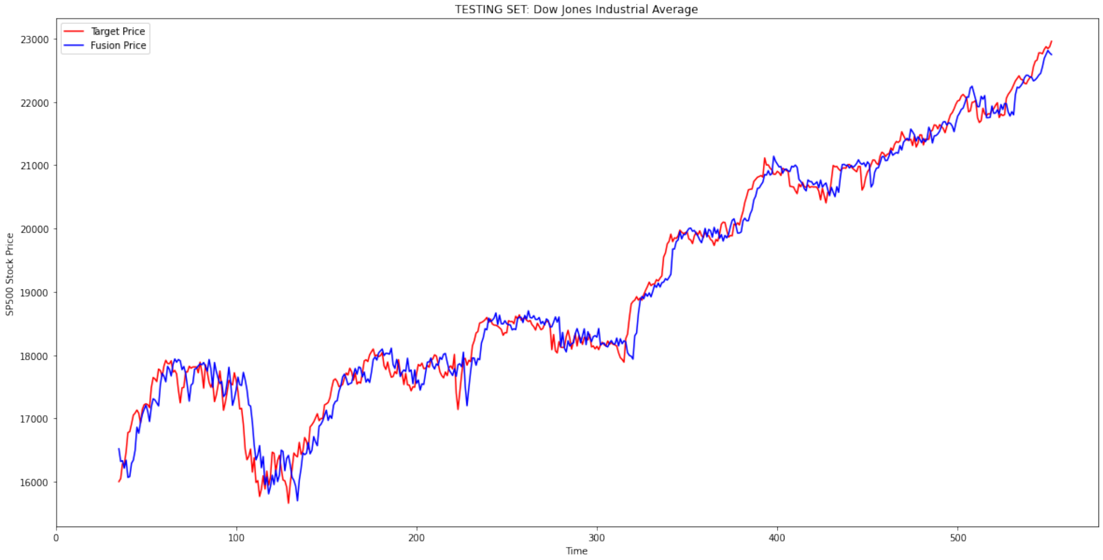


Figure 25: $\mathit{RMSE} = 267.5648 \;\; \mathit{MAE} = 198.2916$

6 Model Comparison

Table 1: Result on Dow Jones Industrial Average (DJI) (Test Set)

	RMSE	**MAE**
ARIMA	274.9319	182.2872
LSTM	296.7456	228.3258
CNN	270.6161	201.3691
LSTM-CNN	269.0664	198.8619

7 Future Work

There are several limitations to the proposed method that came up during experimentation. First and foremost, it appears the CNN model did not work well as we expected. We hope that CNN architecture will extract some latent features that LSTM cannot, but it came to the model gives an approximate result as the LSTM does.

In the future, we will try some recent models like deep transformer models for time series forecasting. Along with the stock's historical trading data, we will use NLP techniques to do sentiment analysis on news that may affect the movement of stock. Besides, using Generative Adversarial Network (GAN) with LSTM, CNN architecture, and Deep Reinforcement Learning (DRL) for deciding when and how to change the GAN's hyper-parameters is very promising.

Acknowledgement

We would like to thank Dr Cao Van Loi from Vietnam AI Academy for his insights and mentorship. We would like to also thank Vingroup Big Data Institute for providing a valuable AI Engineer Training Program.

References

[1] H.Q.Thang. *Vietnam Stock Index Trend Prediction using Gaussian Process Regression and Autoregressive Moving Average Model*. Research and Development on Information and Communication Technology, HUST, 2018.

[2] Kim T, Kim HY. *Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data*. PLoS ONE 14(2): e0212320. `https://doi.org/10.1371/journal.pone.0212320`, 2019.

[3] Hao Y, Gao Q. *Predicting the Trend of Stock Market Index Using the Hybrid Neural Network Based on Multiple Time Scale Machine Learning*. MDPI Appl. Sci. 2020, 10(11), 3961. `https://doi.org/10.3390/app10113961`, 2020.

[4] CS231n. *Convolutional Neural Networks (CNNs / ConvNets)*. `https://cs231n.github.io/convolutional-networks/`.

[5] Aston Zhang, Zachary C. Lipton. *Dive into Deep Learning*.

[6] *Understanding LSTM Network*. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`, 2015.