Stock-loss Prevention: Mobile Application with CNN-LSTM Model for Predicting Sharp Rises and Falls in Stock Price

Borislav Pavlov, Young Oh Kim, Geo Ryang Park, Min Jae Kim

Sungkyunkwan University, 2066, SEOBU-RO, JANGAN-GU, SUWON-SI, GYEONGGI-DO, KOREA
Capstone project by prof. Hyung Joon Koo
December 12, 2021

Abstract

Agility The application provides two types of agility to the user. First, it is agility related to convenience and mobility. Users can check the stock price information predicted through artificial intelligence anytime, anywhere through their mobile device. The second is to respond quickly to changes in stock prices. In order for users to directly respond to stock price changes, they have to keep checking the stock price trend, and even if they check the stock price change, it is difficult to respond nimbly. However, even if the user does not directly check the stock price, the artificial intelligence is always watching the information on the stock price through the application. In addition, an agile response is possible because artificial intelligence predicts the future stock price one hour later.

Convergence of AI and mobile application AI makes it easier to predict things that are difficult for humans to predict. It finds the tendency of the invisible dataset and determines the weight based on it. It predicts the desired value through gradient descent. This is particularly effective in predicting future values, and is frequently used in the topic of prediction of stock prices selected in this project. In this project, rather than simply predicting stock prices, we aim to find the range of future stock price fluctuations through past stock price trends. It predicts future stock price data one hour later based on about 50 hours of past stock data, and displays how much it has risen compared to the most recent value. The prediction result of AI is delivered to the application, and it is decided whether or not to sound an alarm according to the value set by the user in the application.

XAI(eXplainable Artificial Intelligence) Artificial intelligence can help predict many things that humans cannot predict. However, the inability to verify the internal processes for predictions is not a panacea. For this reason, artificial intelligence is sometimes called an invisible 'black box'. To compensate for these shortcomings, XAI, the existence of explainable artificial intelligence, has recently emerged. The biggest purpose of XAI is to provide users with a visualization of the intermediate learning process in artificial intelligence that skips the intermediate learning stage or prediction process and displays only the input and output. Through XAI, it is possible to give the user justification for the AI prediction results, and at the same time, it also suggests the direction to take in the AI model modification.

1 Introduction

1.1 General Statement

We face a lot of stock information every day. Among them, there is truth, there is luck, there is deceit, and there is sedition. So, a quant that focuses on charts rather than weird information(rumor) has emerged. But no one, including us, can make accurate predictions. Rather than predicting stock prices, we want to create a CNN-based alert program to protect users' wealth. As people's interest in the stock market becomes more active, issues related to stock price manipulation are frequently reported on the news. Also, there are general stock investors suffering from unpredictable forms of abnormal price fluctuations. To prevent this, we planned this project, and we predict that the results completed through this project will be of great help in preventing this phenomenon. Our project aims to predict the surge and plunge in stock prices through AI models. CNN-based AI models learn images of several abnormal stock price graph models through input. The CNN model expects to extract common features from images used in learning and predict theoretically incomprehensible exception situations. In addition, we would like to introduce XAI (explainable artificial intelligence) technology to provide a valid reason for predicting results. Among the XAI techniques, the gradient-CAM technique in which the slope is applied to the image will be used. In addition, for the convenience of users, we will provide text on the results. Since the stock dataset is a time series dataset, we decided to introduce LSTM to improve performance, and decided to use AI fused with CNN and LSTM as in the source [1].

1.2 Motivation

Recently, more and more people are looking for stocks as well as experts and companies. The cycle of generating profits through stocks and investing in companies is providing great vitality to the global economy. As more and more people invest in stocks, there are also many cases where they lose money or get bad results from stocks. It is very difficult to predict stock prices in reality, but we wanted to give a little bit of stability to those who invest in stocks.This is why we have chosen the subject of this project. The purpose of our project is to predict the stock price flow through an AI model, not the exact price in an unstable stock market, and warn the user to prevent damage or also gain a profit. By combining a CNN model for learning stock chart images and an LSTM model suitable for predicting stock prices, predicting the direction of stock price rise and fall, and providing the user with a justification for the prediction as a heat map image of Grad-CAM.

1.3 Objective

This is an application with a function to send alarms for sudden rises and falls of stock prices according to the percentage value set by the user. We use an AI model that combines CNN and LSTM to predict the rise and fall of stock prices, and if the prediction result exceeds the number set by the user, a warning is sent to the user. In addition, XAI is utilized for legitimacy and credibility of the prediction results of AI models. By using the Grad-CAM technique, a heat map image is provided to the user, and it shows the user what part of the stock price graph was focused and learned during the prediction process.

1.4 Brief introduction of techniques

A heat map image using CNN is provided to improve UI/UX to convince users of the result while giving a warning using a rough prediction of the stock price. On the other hand, the performance is supplemented by fusion with LSTM.

1.5 Brief introduction of design and implementation

AI models learned by companies such as S&P 500, AMD, Apple, and TESLA are loaded into the App. However, a correction algorithm was added in consideration of the difference between the predicted value and the actual value.

In the mobile application, the user sets alerts to desired stock items and every 15 minutes, based on the model prediction, the user could be notified with a heatmap image according to the alert.

1.6 Our achievement with empirical evaluations

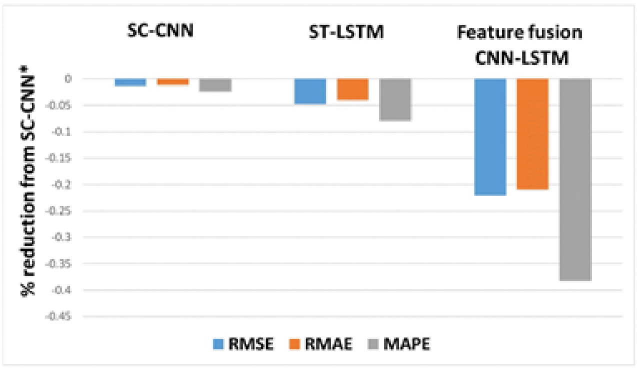
In the final mobile application, the users are able to set alerts, search for stock items in the US market and receive notifications. The notifications are based on the AI model prediction which is triggered every 15 minutes. Moreover, there is authentication and authorization in each service used. Furthermore, all of the user related data such as login information, alerts and notifications received is stored to a database. The images for each notification result are stored on a blob storage provided by Azure.

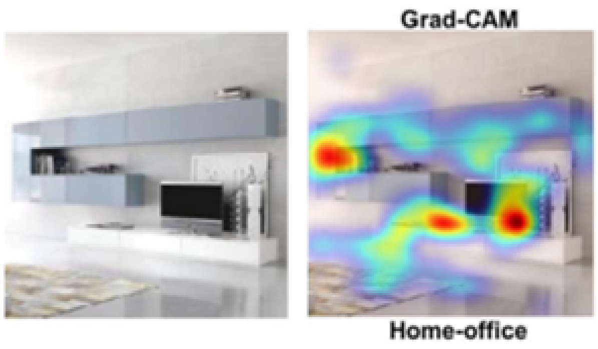
2 Related work

2.1 Relevant techniques of previous work

The image below shows how much the CNN-LSTM fusion model is improved over each model in the source [1].

If we want to predict stock prices from a time series dataset, it is more effective to use LSTM or Transformer algorithm, but the model of [1] is needed because improved UI/UX needs to be provided using CNN.





This image is an image that can be created using Grad-CAM from the source [5]. The reason why AI judged this photo to be home-office is expressed by using CNN's weights.

2.2 Position of our work

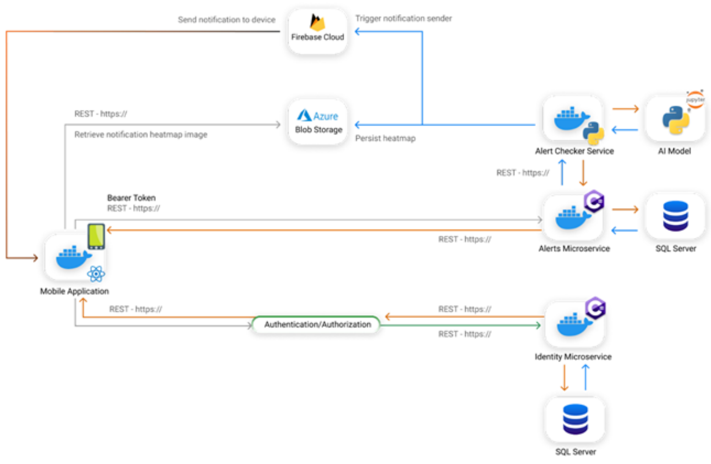
We applied the app and heatmap to the paper [1] that studied the CNN-LSTM combined model academically. And brought it into the realm of the user. Therefore, the unique position of our project is 'Creating an application to notify the rise and fall of stock prices using AI forecasts and heatmap'. There are many applications for stock price prediction in the existing market, but we think it is valuable in that there is no application for preventing losses due to sharp rises and falls in stock prices.

3 Proposed system

3.1 Overall architecture

3.1.1 Overall system architecture

The difference between the architectural design of the solution in the proposal and the final design is clear. The websocket connection was removed, and service for the alerts was added. Furthermore, additional setup was needed for azure blob storage and firebase cloud messaging system in order for the notifications to work. More about the changes and their reasoning is explained in the next chapters.



3.1.2 AI model architecture

The schematic diagram above is the AI model finally used in this project. At the beginning of the project, we tried to use the CNN-LSTM model in [1]. However, even though the model was trained in the same way as described in the paper [1], good results were not obtained, such as a high loss value. Therefore, we decided to build a newly combined CNN-LSTM model. CNN and LSTM applied with resnet's technique were used as main models. In CNN and LSTM, two flat layers that have undergone the flattening process are merged into one

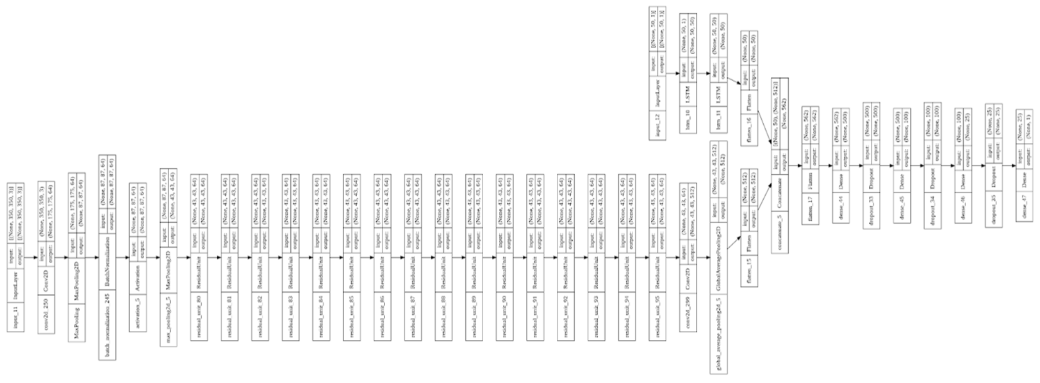


Figure 1: AI architecture call 'Exponential structure' that has the same hyperparameters as this figure with pseudo-Resnet structures. Readers can zoom in on the image to see the details.

layer through the concatenate process, and the merged layer goes through a fully-connected layer and outputs one output.

3.2 Core techniques

CNN is an artificial intelligence model suitable for image learning. In order to utilize CNN, we created an image from a total of 50 hours of stock data. We created an image of a candlestick chart that contains information on close, open, high, and low stock prices so that one image can contain the most information. We have created input data for CNN. However, in the case of CNN, as the learning progresses, the loss of past learning memories and large data are compressed, and prediction errors due to data loss occur. Therefore, a residual layer is added to prevent this.

The data-based image gradually converges through the filter, and it was judged that a 350*350 image converges to a 1*1 result and loses a lot of data. The residual layer plays the role of a fork in the highway in the CNN model. One branch proceeds training in the same way as the CNN structure, but the branch extending to the other side does not perform training and is passed to the next layer with past data. In this way, a new layer from which features are extracted while including past memories can be transferred to the next layer.

Also, we wanted to combine the LSTM model with the CNN model for better prediction results. The idea of combining models was obtained through the paper [1], and a unique combination model was implemented. When using the single model, LSTM performed better than CNN through image learning in stock price prediction, so We thought that combining the two would be of great help in prediction.

3.3 General challenges

One of the main problems was finding a suitable API for retrieving stock information. There are many APIs that can do that, however, most of them are paid or libraries not suitable for a mobile application. Another challenge that the team faced was the implementation of remote push notifications for the mobile application. The notifications development requires multiple components to communicate to each other. All of the notifications are specific for each user and their triggering happens outside the scope of the mobile application.This is the reason firebase cloud messaging system was integrated into the solution.

4 Design

4.1 Design choice

4.1.1 Overall system design choices

The system consists of multiple separate microservices that have single responsibility and are easily scalable. The mobile application was done using React Native library, which is widely used because of its easy setup that provides almost native experience. Moreover, the application can be built for both iOS and Android devices, taking into account that all of the components' design is transformed based on the operating system. The development of the authentication service was done in ASP.NET Core 5, and the reason is that there are many authentication libraries provided by Microsoft that can be easily integrated. Relational database was needed and the choice was SQL Server because it has the best support when integrated in ASP.NET services. Moreover, the Entity Framework provides easy (Code first) database setup, including many utility functions for querying and updating the state of the database. The development of the alerts service was also done in ASP.NET Core 5. The alerts checker service was implemented in python and the reason for that was the integration with the AI Model.

During the mobile application development, a decision was taken to not integrate websocket connection into the system as this is not a required feature and does not contribute to the final goal of the product. The goal of the product is not to provide real time stocks information but to provide alerts when a stock plunge or surge is expected. Assumption is considered that the users will utilize separate applications for monitoring their stocks. This change included a big refactor of the codebase and a new API had to be found for getting stock data, however, the team managed to fulfil the tasks in a short amount of time.

Furthermore, at the start of the project, nobody from the team members had previous experience in developing remote push notifications in mobile applications. This was the reason that the initially proposed system design did not include any systems regarding the notifications flow.

4.1.2 AI model design choices

The argument for using CNN arose from within the team in order to simultaneously implement a function that warns of fluctuations in stock prices and a heat map function that explains the function to the user. However, since stock data is time series data, we judged that CNN alone was insufficient. And since the stock chart is mostly meaningless with a white background, we needed to add another algorithm. We tried to reinforce this by adding simple LSTM layers without complicating the structure as much as possible. Investigating the papers, we found [1] that studied the CNN-LSTM fusion model. Therefore, we chose this structure that can provide a stock price prediction function while providing a heat map.

4.2 ML model & algorithm

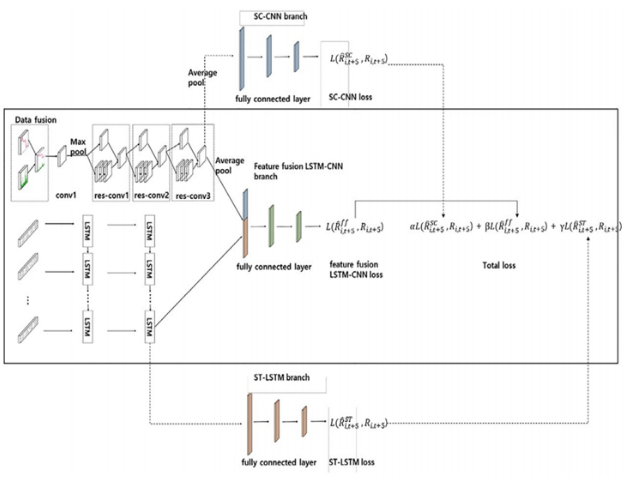
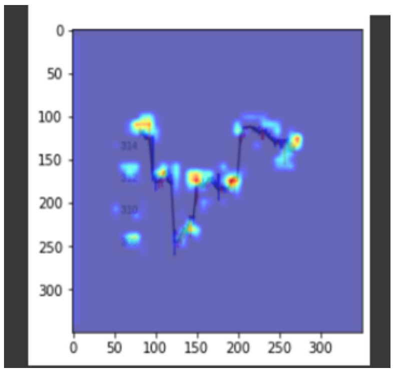

SC-CNN branch
Average pool
fully connected layer
$L(\hat{R}^{SC}_{i,t+5}, R_{i,t+5})$
SC-CNN loss
Data fusion
Max pool
conv1
res-conv1
res-conv2
res-conv3
Average pool
Feature fusion LSTM-CNN branch
LSTM
fully connected layer
$L(\hat{R}^{ff}_{i,t+5}, R_{i,t+5})$
feature fusion LSTM-CNN loss
$\alpha L(\hat{R}^{SC}_{i,t+5},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

Figure 2: This is the architecture of reference[1]

We will compare and contrast the authors of the paper with our model. We put low, high, and closing prices into our CNN and only close prices into LSTM. On the other hand, in [1], trading volume is used. We also tried the Exponential structure, but there was no improvement in performance, so it was simplified. Both codes use log10 for input data. The author of [1] then subtracted the log values and used the relative ratio as input. However, in our code, we did not use the relative ratio by subtracting the input values, but used the logarithmic value as it is. Both codes introduce the Resnet structure to CNN. The author [1] used 3 Resnet Blocks using Resnet jumping 3 convolutional layers and connected them to the FNN. We used 16 Resnet blocks that leap 2 convolutional layers. This was done using Resnet-34 from the source [2]. And by putting one more convolution layer called extractLayer right after these blocks, the convolution layer necessary for the GradCAM function was prepared. It is later used to produce the heatmap image.

The author [1] and we both have two layers in the LSTM. CNN and LSTM join FNN through Flatten and produce predictions. Our output needs to get rid of log10 to get the true predicted value. Since our output has an error with the actual value, we will discuss how to deal with this to produce a predicted value after this part. Adam was used as the loss function. And the learning rate was set to 0.003. Except for the last layer, relu is used as the activation function. The author [1] fixed the epoch, but we used early stopping or selected a value of 30 or less based on the validation RMSE. This will also be discussed after this part.



This photo is a stock chart with heatmap processing. Using the weights learned in extract Layer, the GradCAM function creates the image above.

4.3 Design from UX & UI viewpoints

The mobile application will be available on both iOS and Android. All of the UI components in the application are automatically transformed accordingly to the operating system of the device. The team managed to implement the design of the application exactly as planned and the final designs are illustrated below.

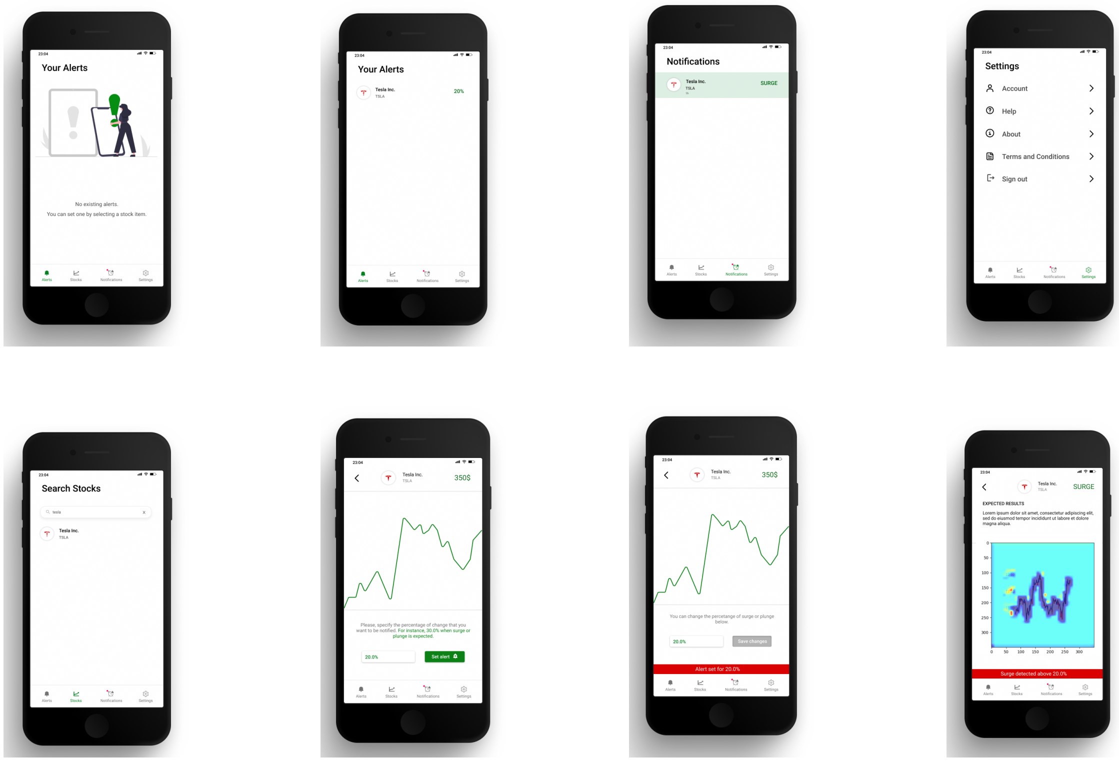


Figure 3: Mobile Application Initial Mockups

5 Implementation

5.1 Description of frontend & backend

The implementation of the mobile application was done using React Native library. The library was developed by Facebook and is widely used in many famous applications such as Instagram, TikTok, etc. Moreover, the library is used with a combination of React as its name states. The state management is implemented by utilizing Redux. Moreover, TypeScript is used instead of JavAscript.

The application has several screens that are only responsible for rendering content according to the current state, all of the logic is kept away from the main components and handled separately. By utilizing redux actions and reducers, each state change is implemented in a separate action that is resolved in its according reducer. The system contains actions and reducers for handling alerts, notifications, authentication, stocks and common state such as loading indications.

The retrieval of stock information is done by using the basic plan of polygon API, which serves the purpose. In order to display the historical data of selected stock items, the “react-native-chart-kit” library is used, which provides intuitive graph visualizations.

Notification handling is done by utilizing firebase libraries. All of the notifications that are received are handled even if the application is in the background. Furthermore, the firebase cloud setup is entirely free. The notifications are currently tested only on android devices using emulators. To be tested in the iOS system, a paid apple developer account is needed because there is no support for iOS emulators.

All of the communication with separate REST APIs is done by utilizing axios. When a user logs in or registers to the system, a token is returned by the authentication API that has expiration time set.This token is attached to the headers of the axios instance. In this way each request later sent will have the token attached. To prevent unexpected logouts when the token expires, the mobile application tries to refresh the token 5 times before it expires by polling the authentication API.

5.2 Dataset for machine learning

To retrieve stock price information, yfinance provided by yahoo was used. Through the yfinance module, the stock price information of a company can be retrieved by time, and a total of 250 days of information are retrieved and saved as a csv file.

The dataset that goes into CNN is a tensorized image. The stock chart was created using [3] with a size of 350x350 in the dataset.

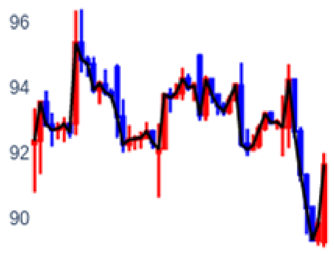
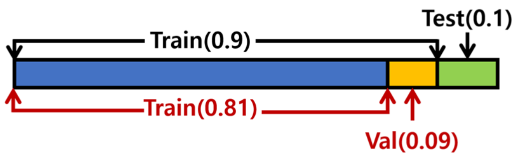


Figure 4: Our images are made by using plotly[3]. The size is 350x350. The images include close prices, low price, high prices. A dataset fetched from Yahoo Finance for the last 250 days of data on an hourly basis was used.

The stock chart image created through plotly is converted into a numpy array after pre-processing. It has a size of 350,350 and has 3 RGB channels. In this process, the saved image file was loaded through PIL Image. The imported image is used as a tensor with the shape of (350,350,3).

In LSTM, closing price data was used as input. In the process, other factors such as volume, high, and low value were used to improve the prediction performance, but there was no significant difference in performance improvement, so only closing price data was used. In order to reduce the error, a method of learning by reducing the number with log10 was used, and the result was obtained by processing the result to the power of 10 after learning by logging the stock price.



The input dataset consisted of 1744 CNNs and LSTMs, respectively. The ratio of train:validation:test was 0.81:0.09:0.1. (After dividing the train and test by 9:1, 10% of the train is used as the validation dataset in the model training process.)

5.3 Open sources taken

The open source we used was mentioned in references. We used parts or concepts of code.

6 Limitations and Discussions

We describe three cases of problems in AI training. The AI part will explain (1) overfitting, (2) underfitting, and (3) approximate and consistent errors between stock price predictions and actual values.

6.1 Discussion of assumptions

Like as paper [1], we also assume that using information processed through different algorithms from the same data will contribute to performance improvement. Because stock data is time series data, the prediction effect occurs in LSTM rather than CNN. It is assumed that our Resnet is more effective than paper [1] because it is a structure from a book that is deeper and has been verified. In fact, when the epoch becomes larger than 30, the amplitude of the prediction graph decreases and if it is overfitted, a converged graph is drawn like 'y = 420'. To solve the (3)'s problem, we will assume one thing. The relative position of the predicted value and the actual value, once determined, persists. That is, if one predicted value is above the actual value, the other values are also above the corresponding actual value. Even empirically, the prediction graph is drawn that way approximately.

6.2 Discussion of limitations

6.2.1 Mobile application limitations

The mobile application limitation is that it is not fully native. React Native library is efficient almost as a native implementation, however, not as much as it. There is still a possibility to add native code to the application, however, it will decrease maintainability of the solution.

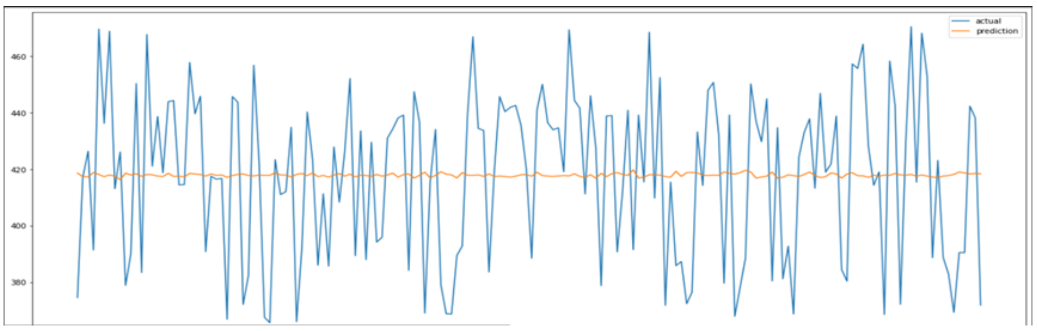
6.2.2 Alerts checker service limitations

Currently, the alerts are retrieved efficiently, however, if the number of alerts in the database is enormous, an optimization needs to be done. The alerts checking process requires communication with many components and the main problem is that the checking of an alert needs to finish completely before the checking of another can start. This means that the notifications have to be sent and persisted in the database, also including generation of notification image to the blob storage, before the service can proceed to another alert. When taking into account that a big amount of people can have alerts set for the same stock item, this could be a bottleneck. Possible solution is to extract the notifications sending and persisting into a separate service. In this way the alerts can be checked in a much quicker way.

6.2.3 AI limitations and constraints

1. Overfitting case

   If the epoch is increased, loss values such as validation RMSE(Root mean square error) seem to be improved on the numerical values. Experimentally, when the epoch starts to exceed 30, the amplitude of the prediction graph gradually decreases. And the prediction curve converges to a straight line.



2. Underfitting case

   If the epoch is very small, such as 10 or less, the loss and RMSE values of validation are inconsistent and large values come out.

   In CNN, the wrong phenomenon of learning the background rather than learning the stock chart is observed. Blue means a low contribution to learning and red means a greater contribution to learning.

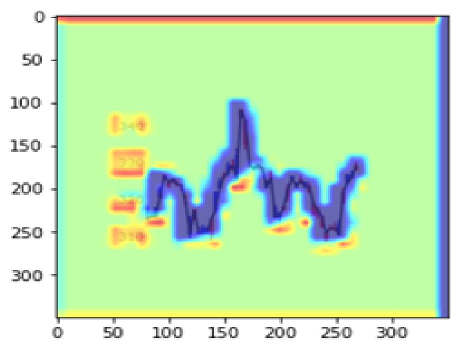
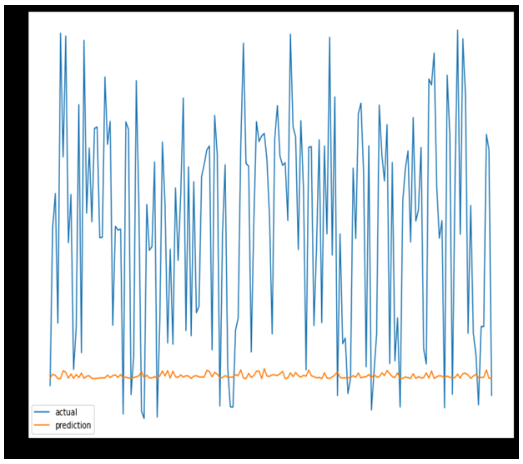


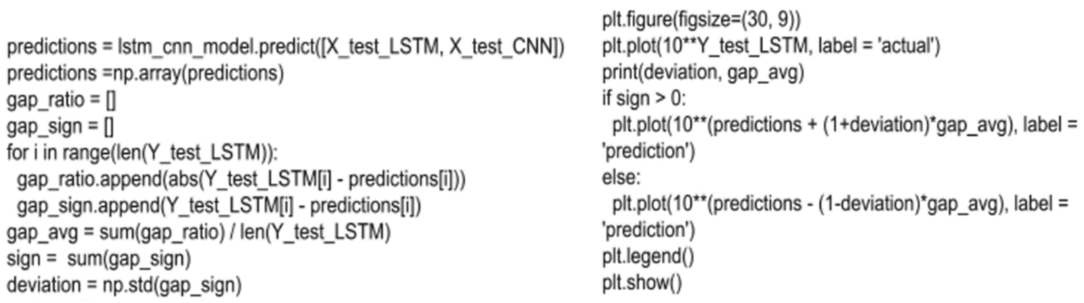
Figure 5: The image is a heatmap image generated when the epoch is small. It seems the CNN doesn't know where to learn.

Therefore, in order to solve the problems of (1) and (2), we need to train the model mainly at epochs between 20 and 30.

3. Approximate and consistent errors between stock price predictions and actual values



As stated below as image, the actual value and the predicted value have a certain difference and are maintained in relative positions. Therefore, a correction algorithm is proposed to solve this problem.



```
predictions = lstm_cnn_model.predict([X_test_LSTM, X_test_CNN])
predictions =np.array(predictions)
gap_ratio = []
gap_sign = []
for i in range(len(Y_test_LSTM)):
  gap_ratio.append(abs(Y_test_LSTM[i] - predictions[i]))
  gap_sign.append(Y_test_LSTM[i] - predictions[i])
gap_avg = sum(gap_ratio) / len(Y_test_LSTM)
sign =  sum(gap_sign)
deviation = np.std(gap_sign)
plt.figure(figsize=(30, 9))
plt.plot(10**Y_test_LSTM, label = 'actual')
print(deviation, gap_avg)
if sign > 0:
  plt.plot(10**(predictions + (1+deviation)*gap_avg), label =
'prediction')
else:
  plt.plot(10**(predictions - (1-deviation)*gap_avg), label =
'prediction')
plt.legend()
plt.show()
```

The values used here are used as exponents of 10 because log 10 is a processed value. The absolute values of the difference between the predicted value and the actual value corresponding to each time point are put in the list. From there, find the arithmetic mean to find gap_avg. Meanwhile, values that have not been processed as absolute values are stored in the gap_sign that is listed. The sign is obtained by adding all the stored values. The relative position of the prediction graph to the actual graph is displayed according to the sign of the sign value. And find the standard deviation from gap_sign. The variance and standard deviation are the number of decimal places. Therefore, standard deviation was used because the standard deviation is larger than the variance to be used for correction of the values. There are two cases here. There are cases where the prediction graph is at the lower side ($sign > 0$) and the case where the prediction graph is at the upper side ($sign \leq 0$).

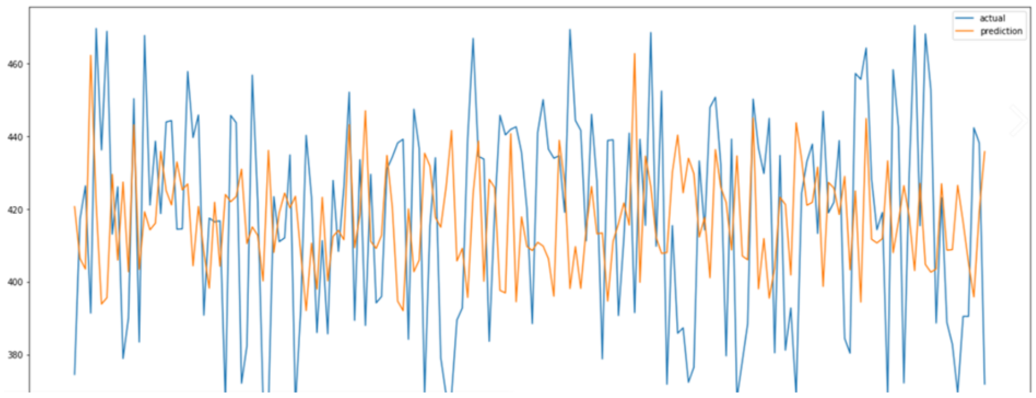
- The lower side ($sign > 0$)

The average difference is multiplied by more to increase the size. Even if we multiply by the same value, the larger number of predicted values increases more and the smaller value increases less, so the amplitude also increases, closer to the actual graph. Multiplying by larger numbers tends to produce better effects. At this time, the amplitude can be slightly increased by multiplying gap_avg by (1 + standard deviation), so it is inserted.

- The upper side ($sign \leq 0$)

This is the case when the prediction graph is above the actual graph. Divide the gap_avg to bring the graph down. However, in this way, the effect of decreasing the amplitude is obtained, which weakens the effect of approaching the real graph. To suppress this, (1-standard deviation) was multiplied by gap_avg to weaken the dividing effect.

So, the result of applying this algorithm is shown in the image below. The standard deviation is not applied to the image below, but the effect of the standard deviation is not large, so the shape does not change significantly. This image suffices to show how the algorithm works.



The position and scale of the waveform became similar to the real graph.

The test dataset was then used to find the mean and standard deviation because of a problem of updating the values. When we actually use the app, it is not good to use the train dataset because there is a risk of insufficient resources due to the large scale. Considering that the data most similar to the future of time series data is the most recent data, it is desirable to use the correction values obtained from the recent data.

7 Evaluation

7.1 Evaluation metrics for ML

Epoch: 26
loss(mse): 0.1524 mape: 11.9957 rmse: 0.3904
val_loss: 0.0050 - val_mape: 2.3413 val_rmse: 0.0706
(Inputs, outputs and hyperparams have already been described in the above.)

7.2 Demonstration of empirical results

The mobile application is easy to use as the team goal does not include any complex user interface functionalities. The goal of the application is providing possibility for the user to set an alert for a desired stock and see any results provided by the prediction. For the purpose, the three main functions are selecting a desired stock, setting alert for the selected stock and viewing of the prediction results.

The selection of the stock items is done by using a search bar in the "Stocks" view of the application. When the name of the desired stock is written, a stock item appears on the screen.

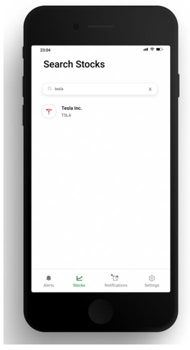
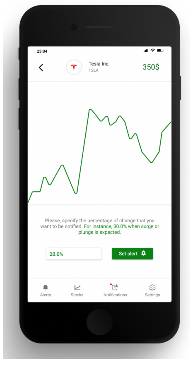


Figure 6: Mobile Application Search Stocks Screen

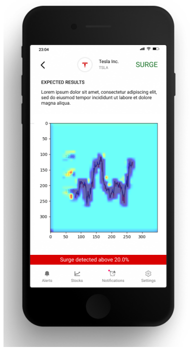
When the user clicks on the desired stock item, the view changes with more information regarding the chosen stock. The selected stock is visible on the top of the screen. Moreover, on the screen appears a graph that shows the prices of the selected stock item for the last 10 days. Furthermore, on the bottom of the screen, an input field is provided and button responsible for setting an alert of the selected stock.



Please, specify the percentage of change that you want to be notified. For instance, 30.0% when surge or plunge is expected.

Figure 7: Mobile Application Selected Stock Screen

Finally, after notification is received by the AI model prediction, the users can easily see the results by selecting the new notification either from the "Notifications" screen or directly from the notification item received on the device. By selecting the notification, the result from the prediction will be shown on the screen. The result includes an explanation of whether plunge or surge is expected for that stock item, the percentage of the expected surge or plunge and a heat map image of the expected stock graph. On the heat map image the users can clearly see what points of the expected graph are in a surge or plunge.



EXPECTED RESULTS

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Figure 8: Mobile Application Notification Result

In conclusion, the application is easy to use. There are no unnecessary functionalities, the screens are clear and have single responsibility.

7.3 Meeting the objective of the project

The developer software solution fulfils the initial requirements and the goal that was set in the beginning. The users are able to search for stock items in the US market, set alerts for them and receive notifications according to the AI model predictions. Moreover, the final user interface design is according to the initially discussed one in the project plan.

8 Conclusion

For the AI model, which is the main implementation detail, two types of prediction models were used, CNN and LSTM. A linear regression model was used to predict the price change after 1 hour using 50 hours of historical data. In the actual evaluation process, input data was generated by fetching data every 15 minutes instead of 1 hour. This is to prevent the alarm cycle from being too long to 1 hour. The reason for using the CNN model that uses the stock price chart image as an input is to create a heatmap applying the Grad-CAM technique through the gradient and weight values stored in the CNN model. Also, in the case of LSTM, since stock price data is composed of time series data and stock price flows are arranged, it is predicted that the learning rate will be high. Therefore, the two models were used by fusion in a way that creates one fully-connected layer through flattening and concatenate processes in the intermediate stage between CNN and LSTM.

The idea of the CNN-LSTM combination model was conceived in the existing paper in [1], and the model was initially planned to use the model presented in the paper in [1]. We succeeded in learning by implementing the model in the paper under the same conditions, but failed to obtain a low loss value or a high prediction rate as described in the paper. In addition, it was difficult to apply to Grad-CAM because the process of processing the dataset was difficult. Therefore, we decided to build our own CNN-LSTM coupling model. In addition, the CNN model is applied with a residual method to reduce the loss rate of the learning process, which is a disadvantage of CNN. Forecasting results are generally not good, but the trend part was well predicted. Prediction graph with a similar waveform was successfully obtained by correcting the height of the graph.

The reason why our self-made model in this project had a low predictive rate was considered as follows:

[1] Difficulty in forecasting due to uncertain fluctuations in stock prices
[2] Difference in layer depth between CNN model and LSTM model
[3] Fast overfitting due to sequential input dataset
[4] Low learning rate due to small epoch value
[5] 1 hour data is used in the train process, but 15 minute data is used in evaluate

Because stock price prediction itself is a very unknown field and the tendency of stock price fluctuations is very different for each company, reason 1 could not be dealt with further in this project. The number of layers of CNN is 34 by adding residual layers, whereas the number of layers of LSTM is composed of 2 layers. Reason 2 was written as the cause of the error, considering that the difference in learning rate may have occurred due to the difference in these layers. Ironically, as the number of epochs increases in the model training process, the loss (error) value continues to decrease, but overfitting occurs severely, so that nothing is displayed in the Grad-CAM heatmap or a phenomenon such as learning a meaningless background continues. Therefore, we had to look at the loss and compromise of model training appropriately to obtain a clear heatmap, and the epoch at that time was determined to be about 30. If this method is solved, We think that it will be possible to obtain a more meaningful result by reducing the loss value and obtaining a clear heatmap graph. Reason 5 is considered to be the biggest cause of error. In the actual learning, data of one hour period was used, but in the evaluation process, data of a period of 15 minutes was used, so there was a difference in the shape of the candlestick chart graph.

9 References

[1] Kim T, Kim HY (2019) 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
[2] 오렐리앙 제롱 저, 박해선 옮김., (2020). 핸즈온 머신러닝 2판. 한빛미디어. resnet-34
[3] Candlestick Charts in Python. Plotly Python Open Source Graphing Library. Retrieved December, 3, 2021. From [https://plotly.com/python/]
[4] Github code corresponding to the source article in [1]. https://github.com/luanft/lstm-cnn-model
[5] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2019). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. International Journal of Computer Vision, 128(2), 336–359. https://doi.org/10.1007/s11263-019-01228-7