Hindawi
Complexity
Volume 2019, Article ID 4324878, 10 pages
https://doi.org/10.1155/2019/4324878

Research Article

Is Deep Learning for Image Recognition Applicable to Stock Market Prediction?

Hyun Sik Sim,[1] **Hae In Kim**,[2] **and Jae Joon Ahn**[2]

[1]*Department of Industrial & Management Engineering, Kyonggi University, Suwon 16227, Republic of Korea*
[2]*Department of Information & Statistics, Yonsei University, Wonju 03722, Republic of Korea*

Correspondence should be addressed to Jae Joon Ahn; ahn2615@yonsei.ac.kr

Received 6 December 2018; Accepted 10 February 2019; Published 19 February 2019

Guest Editor: Thiago C. Silva

Stock market prediction is a challenging issue for investors. In this paper, we propose a stock price prediction model based on convolutional neural network (CNN) to validate the applicability of new learning methods in stock markets. When applying CNN, 9 technical indicators were chosen as predictors of the forecasting model, and the technical indicators were converted to images of the time series graph. For verifying the usefulness of deep learning for image recognition in stock markets, the predictive accuracies of the proposed model were compared to typical artificial neural network (ANN) model and support vector machine (SVM) model. From the experimental results, we can see that CNN can be a desirable choice for building stock prediction models. To examine the performance of the proposed method, an empirical study was performed using the S&P 500 index. This study addresses two critical issues regarding the use of CNN for stock price prediction: how to use CNN and how to optimize them.

1. Introduction

Stock markets have random walk characteristics. Random walk characteristics in stock markets mean that the stock price moves independently at every point in time. Due to the random walk characteristic, stock market prediction using past information is very challenging [1]. In addition, Carpenter et al. [2] insisted that the stock market can be influenced by complex factors, such as business and economic conditions and political and personal issues. There is a high degree of uncertainty in the stock market, which makes it difficult to predict stock price movements [3].

With the globalization and development of information and communication technology (ICT), however, many people are looking toward stock markets for earning excess returns under a convenient investment environment. Therefore, the study of stock market prediction has been a very important issue for investors.

Stock market prediction methods can be categorized into fundamental analysis and technical analysis [4]. Fundamental analysis is a method of analyzing all elements that affect the intrinsic value of a company, and technical analysis is a way of predicting future stock price through graph analysis. When fundamental analysis is applied, some problems may occur. For example, forecasting timeliness can be reduced, subjectivity can be intervened, and the difference between stock price and intrinsic value can be maintained for a long time [5]. Due to the limitation of fundamental analysis, many studies related to stock market prediction using technical analysis have been conducted.

In recent years, many researchers have suggested that artificial neural networks (ANNs) provide an opportunity to achieve profits exceeding the market average by using technical indicators as predictors in stock markets [6–9]. Shin et al. [10] proposed a stock price prediction model based on deep learning techniques using open-high-low-close (OHLC) price and volume and derived technical indicators in the Korean stock market.

However, since many financial market variables are intertwined with each other directly or indirectly, it is difficult to predict future stock price movements by using technical indicators alone, even when applying a typical deep learning model.

In this study, a stock price prediction model based on convolutional neural network (CNN) and technical analysis is proposed to validate the applicability of new learning

methods in stock markets. Unlike typical neural network structures, the CNN, which is most commonly applied to analyze visual imagery, can improve learning performance by convolution and pooling processes [11]. For applying the CNN, various technical indicators, which are used for technical analysis, have been generated as predictors (input variables) of the prediction model, and these technical indicators were converted to images of the time series graph. This study compared the forecasting accuracies of the proposed model and the typical ANN model as well as support vector machine (SVM) model to verify the usefulness of deep learning for image recognition in the stock market.

The remainder of this paper is organized as follows. Section 2 describes the theoretical background for typical ANN, SVM, and CNN. Section 3 introduces the proposed model for stock market prediction in this study. Section 4 demonstrates the empirical results and analysis. Finally, we draw conclusions in Section 5.

2. Background

2.1. Typical ANN. A typical ANN model is a data processing system consisting of layers, connection strengths (weights), a transfer function, and a learning algorithm. The ANN has a structure in which relations between input and output values are learned through iterative weight adjustments. The neural network structure consists of a fully connected layer, in which all neurons are combined with adjacent layers.

The ANN consists of a perceptron, called a neuron, and the overall structure of the general ANN is given in Figure 1(a). The general ANN consists of three layers: the input layer, the hidden layer, and the output layer. In the input layer, the neurons correspond to each input variable. The neurons in the hidden layer and output layer perform the function of calculating the summation of input values and weights in the previous layer.

Figure 1(b) represents the relationship between input and output values in each layer. In Figure 1(b), x_1, x_2, and x_3 represent input signals and have weights of w_1, w_2 and w_3, respectively. The net input function combines the input signal and weight linearly and converts the value through the activation function to output the signal y.

The fully connected layer structure may cause a problem, in which spatial information is lost by ignoring the shape of the data [12]. To increase the representation ability of the data in the ANN model, the number of hidden neurons is increased, or hidden layers are added. However, a vanishing gradient problem occurs when a backpropagation algorithm carries error information from the output layer toward the input layer [13].

2.2. SVM. SVM, developed by Vapnik [14], is an artificial intelligence learning method. It is a machine learning technique based on statistical learning theory and structural risk minimization. The purpose is to identify the optimal separating hyperplane to divide two or more classes of data with the learning mechanism by training the input data. SVM is a type of supervised learning to predict and classify items and it is well known as useful machine learning algorithm for classification [15].

Assume that there are n number of data points existing in the eigenspace, $\{(\overline{x}_1, c_1), (\overline{x}_2, c_2), \cdots, (\overline{x}_n, c_n)\}$, the symbol $C_1 \in \{+1, -1\}$ indicates the classification for data point $\overline{x}_1$. These data points serve as the training data for the identification of the optimal separating hyperplane as

$$\overline{\mathrm{w}} \bullet \overline{x} - \alpha = 0 \tag{1}$$

The symbol $\overline{\mathrm{w}}$ denotes the separating margin and a is a constant. There could be multiple solutions to $\overline{\mathrm{w}}$, but the optimal $\overline{\mathrm{w}}$ is the one with the maximum margin. Equation (2) is the solution to the optimization problem:

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2} \|\overline{\mathrm{w}}\|^2 \\ &\text{subject to} \quad c_1 (\overline{\mathrm{w}} \bullet \overline{x} - \alpha) \geq 1, \quad 1 \leq i \leq n \end{aligned} \tag{2}$$

After the network learning obtains the w with the maximum margin, it is then possible to establish the classification $\widehat{C}$ by using (3) on the test data that has yet to be classified.

$$\widehat{C} = \begin{cases} -1, & if\ \overline{\mathrm{w}} \bullet \overline{x} - \alpha \leq -1 \\ +1, & if\ \overline{\mathrm{w}} \bullet \overline{x} - \alpha \geq +1 \end{cases} \tag{3}$$

2.3. CNN. The CNN, as a deep learning technique, is a model that imitates the visual processing of living organisms that recognize patterns or images. The CNN has a structure in which one or more convolutional layers and pooling layers are added to a fully connected layer, which results in an ANN structure.

Figure 2 shows the structure of LeNet-5, which is the most famous CNN algorithm. According to Figure 2, a five-layer CNN was established. LeNet-5 is composed of two convolutional layers for the first two layers and three fully connected layers for the remaining three layers. First, the image of the input layer is filtered through the convolutional layer to extract appropriate features [16].

The convolutional layer is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. Convolution is a mathematical operation that requires two inputs, such as an image matrix and a filter or kernel.

A convolution operation is an elementwise matrix multiplication operation, where one of the matrices is the image and the other is the filter or kernel that turns the image into something else. The output of this is the final convoluted image. If the image is larger than the size of the filter, the filter is moved to various parts of the image to perform the convolution operation. If the convolution operation is performed each time, a new pixel is generated in the output image.

In image processing, there are few sets of filters that are used to perform several tasks. The convolution of an image with different filters (kernels) can perform operations, such as edge detection, blurring, and sharpening, by applying filters.

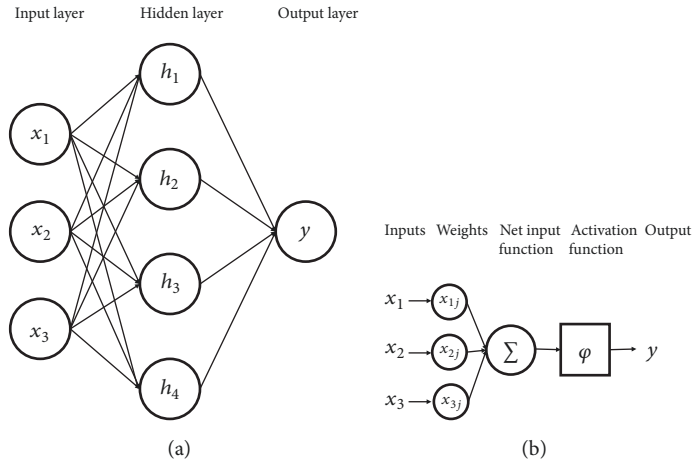

Figure 1: Typical ANN structure. (a) The overall structure of the general ANN. (b) The relationship between the input and output values in each layer.

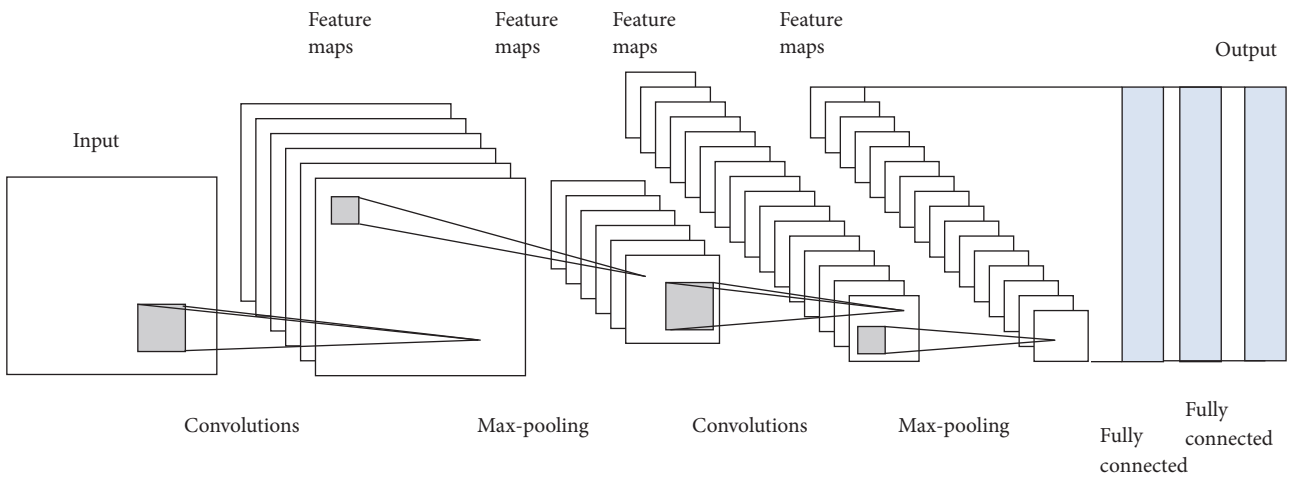

Figure 2: The LeNet-5 structure.

In CNNs, filters are not defined. The value of each filter is learned during the training process [17]. Every filter is spatially small (in terms of width and height) but extends through the full depth of the input volume. During the forward pass, each filter is moved across the width and height of the input volume, and dot products are computed between the entries of the filter and the input at any position. As the filter is moved over the width and height of the input volume, a 2-dimensional feature map that gives the responses of that filter is produced at every spatial position [18].

Intuitively, the network learns filters that activate when they see some type of visual feature, such as an edge of some orientation of the first layer or eventually the entire honeycomb or wheel-like patterns within the higher layers of the network. An entire set of filters is generated in each convolutional layer, and each one produces a separate 2-dimensional feature map.

Figure 3 shows the process of generating a feature map for a convolutional layer. The original image is the one on the left, and the matrix of numbers in the middle is the convolutional matrix or filter. Consider a 4 x 4 matrix, whose image pixel values are 0, 1, 2, and 3, and a 3 x 3 filter matrix, as shown in Figure 3. Then, the convolution of the 4 x 4 image matrix multiplies with the 3 x 3 filter matrix, which results in the feature map, as shown in Figure 3.

The activation functions of every convolutional layer and the first two fully connected layers are shown in (4) (i.e., ReLU (Rectified Linear Unit)). The ReLU function is used to solve the vanishing gradient, which does not reflect the output error of the neural network as it moves away from the output layer in the process of the neural network [19].

$$\max \quad (0, x)$$

$$f(x) = \begin{cases} 0 & for\ x < 0 \\ x & for\ x \geq 0 \end{cases} \tag{4}$$

Generally, the pooling layer is located after the convolutional layer. The pooling layer was introduced for two main reasons [20]. The first was to perform down sampling (i.e., to reduce

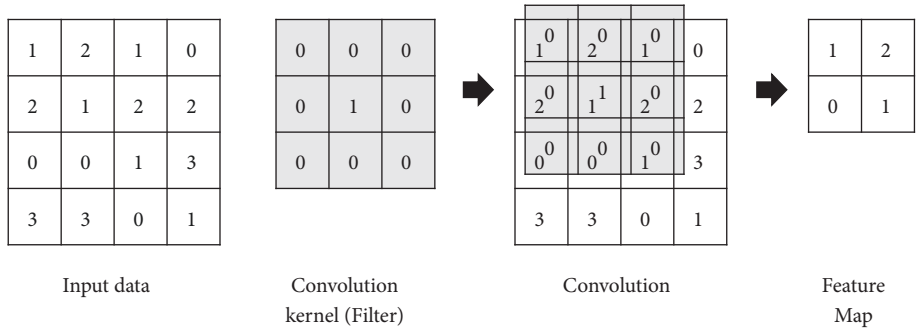

Figure 3: The process of generating the feature map of the convolutional layer.

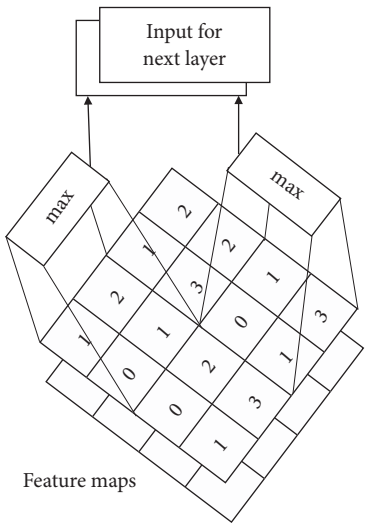

Figure 4: The process in the max pooling layer.

the amount of computation that needs to be done), and the second was to only send the important data to the next layers in the CNN max pooling layers by taking the largest element from the rectified feature map, as shown in Figure 4. The most common form is a pooling layer, with filters of size 2x2, which are applied with a stride of 2 down samples every depth slice in the input by 2 along both the width and height, discarding 75% of the activation. These values are then linked to a fully connected layer, such as an ANN structure, to output the label-specific prediction probabilities

3. CNN Architecture for Building a Stock Price Prediction Model

3.1. Input Image Generation. In this study, historical S&P 500 minute data are used, and these time series data are divided into 30 minute increments for stock price prediction. When learning a prediction model, the closing price and technical indicators are considered as input variables, and target variables are set to values expressed as 1 or 0. If the target has a value of 0, the closing price at time t − 1 is higher than the closing price at time t, as shown in (5). In other words, the stock price prediction model proposed in this study learns the moving pattern of the independent variables for 30 minutes and forecasts the increase or decrease in the stock price after one minute.

$$\text{target} = \begin{cases} 0 & for\ close\ price_t < close\ price_{t-1} \\ 1 & for\ close\ price_t \geq close\ price_{t-1} \end{cases} \quad (5)$$

Table 1 shows the technical indicators used in this study. Nine technical indicators are selected for the prediction model (refer to [21]): simple moving average (SMA), exponential moving average (EMA), rate of change (ROC), moving average convergence divergence (MACD), fast %K, slow %D, upper band, lower band, and %B. Finally, the technical indicators calculated by Table 1 are standardized to have a value between 0 and 1 for converting to images of time series graph.

Now, the technical indicators are converted to the images of a time series graph using the input image of the CNN. Finally, 1100 input images in the training period and 275 input images in the test period are generated. Figure 5 shows the example of the input images in the test period when applying only 3 input variables. In Figure 5, the red line, green line, and blue line indicate the closing prices of the S&P 500 index, SMA 20, and EMA 20, respectively.

3.2. CNN Parameter Settings for the Best Prediction Model Architecture. In this study, the LeNet-5 algorithm is used for stock price prediction. The CNN structure of this study is shown in Figure 6. The 64×64×3 input image is filtered in the first convolutional layer by 3×3×3 kernels, with a stride of 1 pixel. Then, max pooling is used in the pooling layer. The main purpose of the pooling operation is to reduce the size of the image as much as possible, taking a 2×2 matrix to minimize pixel loss and obtain the correct characteristic region [22].

The second convolutional layer filters the output of the first convolutional layer using 3×3×3 kernels, with a stride of 1 pixel. After the pooling process is performed once again, flattening, which is a process of converting a two-dimensional array into one long continuous linear vector, is performed. That is, the process of converting a pooled image pixel into a one-dimensional single vector is performed.

In the fully connected layer, the entire connection of 512 neural networks is performed. The number of neurons in both of the first two fully connected layers is 512. Then,

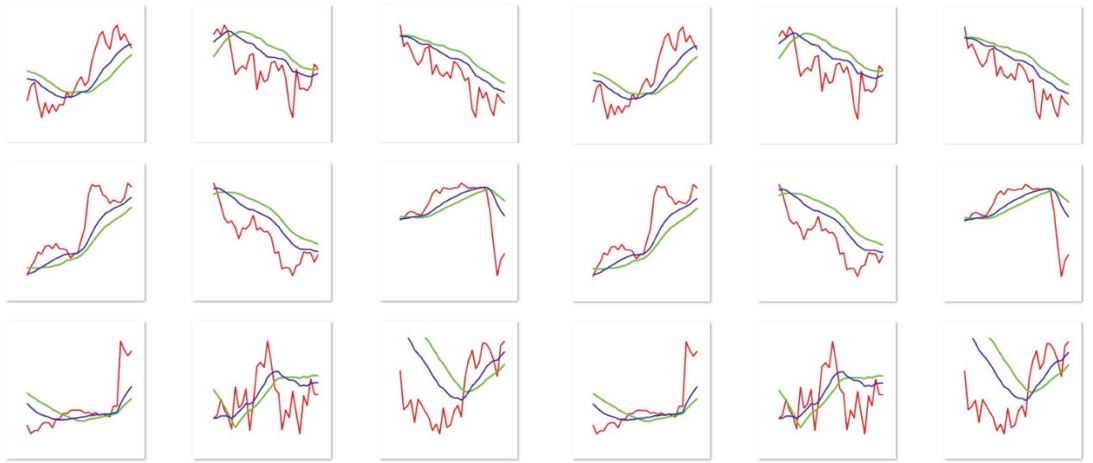

Figure 5: Example of the input image. (a) Generated input image when the closing price increases after 1 minute. (b) Generated input image when the closing price decreases after 1 minute.

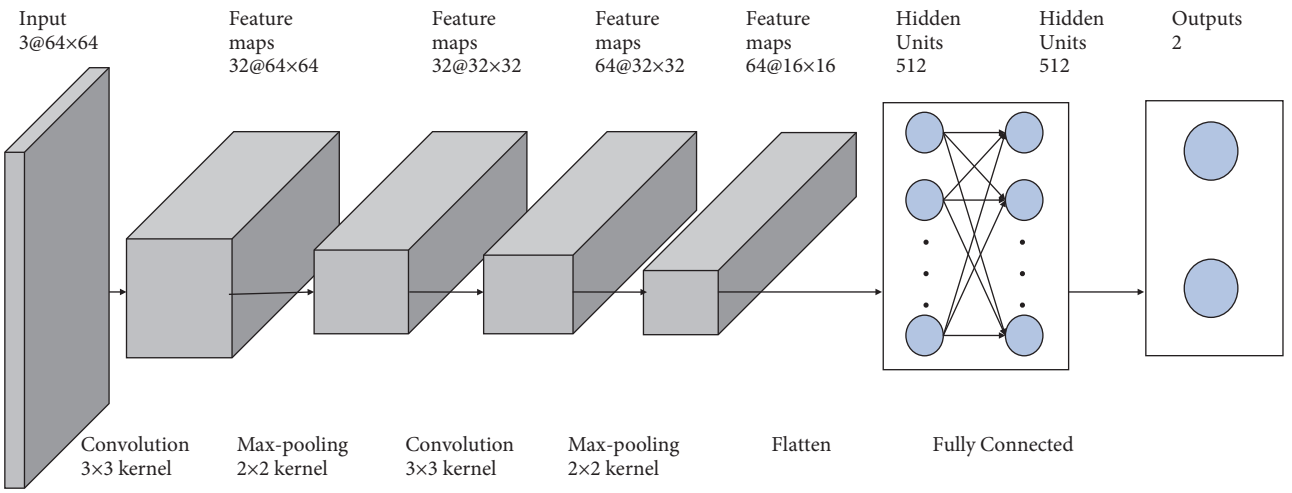

Figure 6: The architecture of the CNN for the prediction model.

because the process is a binary classification, the connection goes through an output layer that contains only one node. The last layer uses the sigmoid activation function.

Adaptive Optimization Methods. Stochastic gradient descent (SGD) has been widely used when training CNN models. Despite its simplicity, SGD performs well empirically across a variety of applications but also has strong theoretical foundations [23].

Training neural networks is equivalent to solving the nonconvex optimization problem in

$$\min_{w \in \mathbb{R}^n} \quad f(w) \tag{6}$$

where f represents a loss function. The iterations of SGD can be described in

$$w_k = w_{k-1} - \alpha_{k-1} \widehat{\nabla} f(w_{k-1}) \tag{7}$$

where w_k denotes the k^{th} iteration, α_k represents a (tuned) step size sequence (also called the learning rate), and $\widehat{\nabla} f(w_k)$ denotes the stochastic gradient computed at w_k.

The Adam optimization algorithm is an algorithm that can be used instead of the classical SGD procedure to update network weights iteratively based on training data. The Adam algorithm is popular in the field of deep learning because it achieves good results quickly [24]. The updated Adam equation can be represented in

$$w_k = w_{k-1} - \alpha_{k-1} \frac{\sqrt{1-\beta_2^k}}{1-\beta_1^k} \frac{m_{k-1}}{\sqrt{v_{k-1}} + \epsilon} \tag{8}$$

where

$$\begin{aligned} m_{k-1} &= \beta_1 m_{k-2} + (1-\beta_1) \widehat{\nabla} f(w_{k-1}) \\ v_{k-1} &= \beta_2 v_{k-2} + (1-\beta_2) \widehat{\nabla} f(w_{k-1})^2 \end{aligned} \tag{9}$$

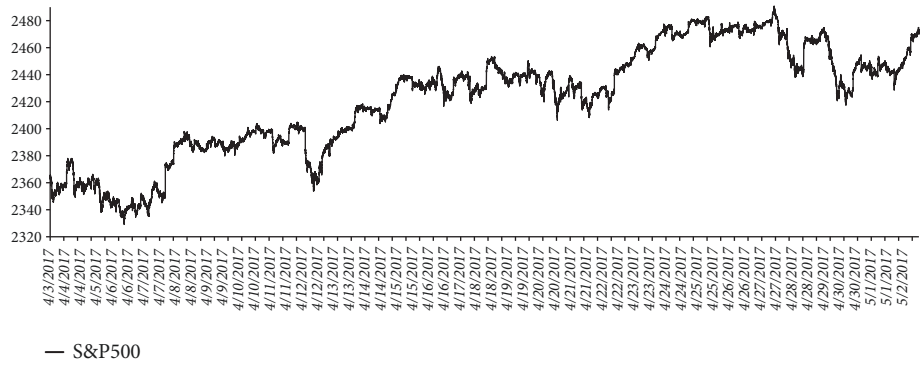

Figure 7: Minute closing prices of the S&P 500 index.

$\beta \in [0, 1)$ represents a momentum parameter, and v_0 is initialized to 0.

Dropout. The dropout method introduced by Hinton et al. [25] is known as a very effective way to reduce overfitting when applying neural networks with many hidden layers. This method consists of setting the output of each hidden neuron in the chosen layer to zero with some probability (usually 50%). In this paper, the dropout method was applied after the pooling operations.

Loss Function. The ANN uses the loss function as an indicator to determine the optimal weight parameter through learning [26]. In this study, the mean square error (MSE) and cross entropy error (CEE) were adopted to comprise the objective function (loss function). Equations (10) and (11) show the MSE measure and CEE measure, respectively. y_k represents the output of the neural networks, and t_k represents the target value in (10) and (11).

$$MSE = \frac{1}{n}\sum_{k}^{n}\left(y_k - t_k\right)^2 \tag{10}$$

When calculating the MSE, the neurons in all output layers are entered. This loss function is most commonly used because it is simple to calculate. Basically, the difference between the output of the model and the target distance is used as an error. The advantage of squaring the distance difference is that the difference between data with small distance differences and the large data error becomes larger, which has the advantage of being able to know exactly where the error is located.

$$CEE = -\sum_{k}^{n} t_k \log y_k \tag{11}$$

The CEE only counts the neuron corresponding to the target, which results in a larger penalty as it moves farther from the target.

Epoch and Batch Sizes. An epoch consists of one full training cycle for the data. An epoch is an iteration over the entire training data and target data provided. The epochs are equal to 2500 in this study. The batch size is a term used in machine learning and refers to the number of training examples utilized in one iteration. The batch size is 1 [27] in this study.

Steps per epoch indicate the number of batch iterations before a training epoch is considered finished. These steps represent the total number of steps (i.e., batches of samples) before declaring one epoch finished and starting the next epoch.

4. Empirical Studies

4.1. Experimental Settings. In this study, the empirical analysis covers a 1-month period. The dataset consists of minute data of the S&P 500 index from 10:30 pm on April 3, 2017, to 2:15 pm on May 2, 2017. The entire dataset covers 41,250 minutes. Figure 7 shows a time series graph of the S&P 500 closing price during the analysis period.

Among the entire dataset, 33,000 minutes are allocated for the training data (80% of the entire data), and 8,250 minutes are allocated for the testing data (20% of the entire data). When the time series data are converted into an image every 30 minutes, the training data consist of 1,100 input images, and the testing data consist of 275 input images.

For experimenting with the CNN algorithm, the technical indicators used for forecasting the stock price in [21] are employed as input variables here.

To evaluate the forecasting accuracy, the following three measurements are employed: hit ratio, sensitivity, and specificity (see (12)–(14)).

$$hit\ ratio = \frac{n_{0,0} + n_{1,1}}{n} \tag{12}$$

$$sensitivity = \frac{n_{0,0}}{n_{0,0} + n_{0,1}} \tag{13}$$

$$specificity = \frac{n_{1,1}}{n_{1,0} + n_{1,1}} \tag{14}$$

In (12)-(14), $n_{0,0}$ and $n_{0,1}$ represent the number of predicted values of 0 and the number of predicted values of 1 when the actual value is 0, respectively. Additionally, $n_{1,0}$ and $n_{1,1}$ represent the number of predicted values of 0 and the number of predicted values of 1 when the actual value is 1, respectively. The hit ratio is a metric or measure of the prediction model performance when the target variable is binary. While

Table 1: Technical indicators used for the proposed prediction model.

Technical indicators	Formula (n=20)	Explanation
SMA	$\frac{\sum(Price, n)}{n}$	n = Time Period
EMA	$Close(i) \bullet P + (EMA\,(i-1) \bullet (1-P)$	$Close(i)$ = The closing price at time i $EMA(i-1)$ = Exponentially moving average of the closing price at time *i-1* P = the percentage using the price value
MACD	$FastMA - Slow\ MA$	Fast MA is the moving average (5) Slow MA is the moving average (20)
ROC	$100 \bullet \frac{Close - Close\ n\ ago}{Close\ n\ ago}$	
Fast %K	$100 \bullet \frac{Close - Low}{High - Low}$	
Slow %D	$SMA(Slow\ K\%, Dma)$	$Slow\ K\%$ = SMA (Fast %K, KMA) KMA = Period of moving average used to smooth the slow %K values
Upper Band	$Middle\ Band + (y \bullet n - standard\ deviation)$	$Middle\ Band$ = n-period moving average
Lower Band	$Middle\ Band - (y \bullet n - standard\ deviation)$	y = factor applied to the standard deviation value
%B	$\frac{Price - Lower\ Band}{Upper\ Band - Lower\ Band}$	

Table 2: Input variables for each CNN model.

	Input variables
CNN1	Closing price
CNN2	Closing price, SMA, EMA
CNN3	Closing price, SMA, EMA, ROC, MACD
CNN4	Closing Price, SMA, EMA, ROC, MACD, Fast %K, Slow %D, Upper Band, Lower Band

the hit ratio is simply a measure of discrepancies between the predicted value and actual observations, sensitivity and specificity measure the conditional discrepancies depending on actual observations.

4.2. Experimental Results. In this study, TensorFlow was used for the experiment. TensorFlow is a famous deep learning development framework in which grammar is developed in the form of a Python library. To verify the usefulness of the technical indicators as an input variable, four CNN models are constructed with different technical indicators. The CNN models are created by applying 0, 2, 4, and 9 technical indicators. In this study, these models are called CNN1, CNN2, CNN3, and CNN4, respectively. Table 2 presents the input variables applied to these four models.

Table 3 shows the accuracies of the four models. To determine the adaptive optimization method, all CNN parameters (except for the adaptive optimization method) are applied equally to each model. Here, the dropout probability, batch size, and epoch are fixed at 0.5, 1, and 2500, respectively. Additionally, the steps per epoch in the training and testing data were set to 250 and 50, respectively, and the loss function was the CEE.

As shown in Table 3, when the SGD optimizer is used for the adaptive optimization method, CNNs achieve a high level of predictive performance. CNN1, which is the prediction model without technical indicators, has the highest hit ratio among the four models. Therefore, technical indicators cannot affect the positive impact of the CNN on stock price forecasting. However, a large difference between the sensitivity and specificity of CNN1 indicates that an overfitting problem occurs due to considering only one input variable.

Table 4 shows the accuracies of the four models with SGD optimizers using different loss functions. From Table 4, we know that the use of the MSE as a loss function increases the predictability rather than the use of the CEE.

The accuracies of the four models with the SGD optimizer and MSE loss function using different dropout probabilities are given in Table 5. CNN1 has the highest hit ratio (0.85) when the dropout probability is 0. The results in Table 5 show that an increase in the dropout probability does not contribute to the predictive performance of the CNN, which is interesting because dropout options are widely known to play an important role in deep learning architecture construction. In the case of this experiment, however, since the learning image of CNN models is simpler than the character recognition or text recognition generally applied to CNNs, it is considered that the dropout option has a negative effect.

Table 6 shows the accuracies of the four CNN models with different steps per epoch when applying the SGD optimizer,

Table 3: Accuracy comparison for CNNs with different optimizers during the test period.

	Optimizer	Hit ratio	Specificity	Sensitivity
CNN1	Adam	0.63	0.9545	0.3587
	SGD	0.65	0.9596	0.3810
CNN2	Adam	0.52	0.5822	0.4800
	SGD	0.56	0.5248	0.6144
CNN3	Adam	0.58	0.6460	0.4408
	SGD	0.60	0.6429	0.4840
CNN4	Adam	0.56	0.54	0.5810
	SGD	0.58	0.5939	0.5868

Table 4: Accuracy comparison for CNNs with different loss functions during the test period.

	Loss function	Hit ratio	Specificity	Sensitivity
CNN1	MSE	0.66	0.6611	0.6508
	CEE	0.65	0.9596	0.3810
CNN2	MSE	0.67	0.6825	0.6209
	CEE	0.56	0.5248	0.6144
CNN3	MSE	0.62	0.6151	0.6302
	CEE	0.60	0.6429	0.4840
CNN4	MSE	0.62	0.6114	0.6393
	CEE	0.58	0.5939	0.5868

Table 5: Accuracy comparison for CNNs with different dropout probabilities during the test period.

	Dropout probability	Hit ratio	Specificity	Sensitivity
CNN1	0	0.85	0.9593	0.6971
	0.25	0.67	0.6904	0.6507
	0.5	0.66	0.6611	0.6508
CNN2	0	0.62	0.6679	0.5878
	0.25	0.68	0.6992	0.6744
	0.5	0.67	0.6825	0.6209
CNN3	0	0.64	0.9559	0.2487
	0.25	0.64	0.9091	0.3012
	0.5	0.62	0.6151	0.6302
CNN4	0	0.66	0.6548	0.6872
	0.25	0.62	0.6040	0.6375
	0.5	0.62	0.6114	0.6393

Table 6: Accuracy comparison for CNNs with different steps per epoch during the test period.

	Steps per epoch (train / test)	Hit ratio	Specificity	Sensitivity
CNN1	400 / 100	0.68	0.7221	0.6413
	800 / 200	0.54	0.5324	0.5434
CNN2	400 / 100	0.53	0.6155	0.46
	800 / 200	0.52	0.6257	0.4118
CNN3	400 / 100	0.61	0.9304	0.31
	800 / 200	0.52	0.4824	0.5238
CNN4	400 / 100	0.54	0.6220	0.4734
	800 / 200	0.54	0.7112	0.3450

Table 7: Predictive accuracies of ANNs and SVMs.

	Hit ratio	Specificity	Sensitivity
ANN1	0.4872	0.6866	0.2979
ANN2	0.5602	0.5674	0.5522
ANN3	0.5653	0.6561	0.4801
ANN4	0.5573	0.6269	0.4626
SVM1	0.48	0.8881	0.0922
SVM2	0.4655	0.8582	0.0922
SVM3	0.5018	0.4851	0.5177
SVM4	0.5455	0.5149	0.5745

Table 8: Optimized parameters for CNNs.

Parameter	Considered value (option)	Selected Value (option)
Adaptive optimization method	Adam, SGD	SGD
Dropout probability	0 ~ 0.5	0
Loss function	MSE, CEE	MSE
Steps per epoch (train / test)	200 / 50 ~ 800 / 200	200 / 50

MSE loss function and dropout probability of 0. Based on the results of Table 6, we can realize that an increase in steps per epoch causes an overfitting problem and results in a decrease in accuracy. As a result, it is not effective in increasing the number of steps for stock price prediction based on a CNN using technical indicators.

To verify the performance of CNN models, ANN and SVM models are generated and their accuracies are evaluated. The same input variables for CNNs in Table 2 are applied to ANNs and SVMs. Before exploring the ANN and SVM for stock price prediction, small preliminary experiments were performed to obtain proper parameter settings for the successful implementation of the ANN and SVM. As a result, the number of hidden layers, the number of hidden units, and the activation function of ANN are set to be 1, 3, and sigmoid, respectively. And SVM uses polynomial kernel to make a nonlinear classification interface.

Based on the results show in Table 7, when the ANN and SVM are applied, technical indicators are shown to be input variables positively affecting the stock price prediction, as opposed to when the CNN is applied. Nevertheless, the predictive performances of the ANN and SVM are lower than that of the CNN (refer to Table 5 when dropout probability is 0). Therefore, CNNs using input images can be a useful method for stock price prediction. In practice, CNN models are good at detecting patterns in images such as lines. CNNs can detect relationships among images that humans cannot find easily; the structure of neural networks can help detect complicated relationships among features. For example, in CNN, color images are composed of RGB channels, and the features of input for each channel can be extracted. This allows CNN to extract features better than when it uses a vectorized input such as ANN [28].

5. Concluding Remarks

In this study, we attempted to check the applicability of the CNN for stock market prediction. Previously, many researchers have suggested that ANNs offer a chance to achieve profits in financial markets. Therefore, this study determined the predictive performances of the CNN and ANN to validate the usefulness of the CNN. In addition, SVM, well known for useful classification algorithm, was employed to verify the usefulness of the CNN.

To design the CNN architecture, this study focused on two points. First, the CNN parameters were optimized. For this, the experiments were performed over the parameter range given in Table 8, and the best experiments were obtained. Second, technical indicators, which are well known as efficient input variables in stock price forecasting, were verified to play a role as a suitable input image for CNNs when technical indicators are converted into images.

Our empirical experiments demonstrate the potential usefulness of the CNN by showing that it could improve the predictive performance more than the ANN. In this sense, the CNN appears to be a desirable choice for building stock prediction models. In addition, technical indicators were input variables that did not positively affect the stock price prediction when the CNN was implemented for the prediction model. This result is because technical indicators cannot be good input variables, as they are similar to the moving pattern of the closing price. Therefore, building a stock price prediction model with better performance can be expected if other factors that move opposite the stock price, such as gold price and interest rate, are considered as input variables for the CNN. As a result of this study, it is difficult to predict technical indicators of stock market by general data mining classification technique. Therefore, CNN, which is a deep learning method that analyzes time series data into graphs, can be a useful for stock price prediction.

Data Availability

The data used in this study can be accessed via https://www.kesci.com/home/dataset/5bbdc2513631bc00109c29a4/files.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] E. F. Fama, "Random walks in stock market prices," *Financial Analysts Journal*, vol. 51, no. 1, pp. 75–80, 1995.

[2] G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen, "Artmap: a neural network architecture for incremental learning supervised learning of analog multidimensional maps," *IEEE Transactions in Neural Networks*, vol. 3, no. 5, pp. 698–713, 1992.

[3] Y. Kara, M. Acar Boyacioglu, and Ö. K. Baykan, "Predicting direction of stock price index movement using artificial neural networks and support vector machines: the sample of the Istanbul stock exchange," *Expert Systems with Applications*, vol. 38, no. 5, pp. 5311–5319, 2011.

[4] B. G. Malkiel, *A Random Walk Down Wall Street*, W. W. Norton & Company, New York, NY, USA, 1999.

[5] J. L. Bettman, S. J. Sault, and E. L. Schult, "Fundamental and technical analysis: substitutes or complements?" *Accounting & Finance*, vol. 49, no. 1, pp. 21–36, 2009.

[6] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market prediction," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018.

[7] J. Lee, "A stock trading system based on supervised learning of highly volatile stock price patterns," *Journal of Korean Institute of Information Scientists and Engineers*, vol. 19, no. 1, pp. 23–29, 2013.

[8] C.-M. Hsu, "A hybrid procedure for stock price prediction by integrating self-organizing map and genetic programming," *Expert Systems with Applications*, vol. 38, no. 11, pp. 14026–14036, 2011.

[9] Y. K. Kwon, S. S. Choi, and B. R. Moon, "Stock prediction based on financial correlation," in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pp. 2061–2066, ACM, Wash, D.C., USA, June 2005.

[10] D. Shin, K. Choi, and C. Kim, "Deep learning model for prediction rate improvement of stock price using RNN and LSTM," *The Journal of Korean Institute of Information Technology*, vol. 15, no. 10, pp. 9–16, 2017.

[11] Y. J. Song and J. W. Lee, "A design and implementation of deep learning model for stock prediction using tensorflow," *KIISE Transactions on Computing Practices*, vol. 23, no. 11, pp. 799–801, 2017.

[12] R. J. Schalkoff, *Artificial Neural Networks*, vol. 1, McGraw-Hill, New York, NY, USA, 1997.

[13] Y. Bengio, P. Simard, and P. Frasconi, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[14] V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer, New York, NY, USA, 1995.

[15] C. L. Jan, "An effective financial statements fraud detection model for the sustainable development of financial markets: evidence from Taiwan," *Sustainability*, vol. 10, no. 2, p. 513, 2018.

[16] Y. LeCun, B. Boser, J. S. Denker et al., "Gradient-based learning applied to document recognition," in *Shape, Contour and Grouping in Computer Vision*, pp. 319–345, Springer, Berlin, Germany, 1999.

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Object recognition with gradient-based learning," in *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, 1998.

[18] S. Sarraf and G. Tofighi, "Classification of alzheimer's disease using fmri data and deep learning convolutional neural networks," *Computer Vision and Pattern Recognition*, Article ID 1603.08631, 2016, https://arxiv.org/abs/1603.08631.

[19] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, vol. 15, pp. 315–323, 2011.

[20] D. Cires, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in *Proceedings of The Twenty-Second International Joint Conference on Artificial Intelligence*, pp. 1237–1242, 2011.

[21] S. B. Achelis, *Technical Analysis from A to Z*, McGraw Hill, New York, NY, USA, 2001.

[22] M. Abadi, A. Chu, and l. Goodfellow, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2016.

[23] H. Robbins and S. Monro, "A stochastic approximation method," *Annals of Mathematical Statistics*, pp. 102–109, 1985.

[24] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," *Machine Learning*, Article ID 1412.6980, 2014, https://arxiv.org/abs/1412.6980.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS '12)*, pp. 1097–1105, Lake Tahoe, Nev, USA, December 2012.

[26] J. B. Hampshire and A. H. Waibel, "Novel objective function for improved phoneme recognition using time-delay neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 1, no. 2, pp. 216–228, 1990.

[27] D. R. Wilson and T. R. Martinez, "The general inefficiency of batch training for gradient descent learning," *Neural Networks*, vol. 16, no. 10, pp. 1429–1451, 2003.

[28] Y. Tsai, J. Chen, and J. Wang, "Predict forex trend via convolutional neural networks," *Journal of Intelligent Systems*, Article ID 1801.03018, 2018.