Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 2746845, 13 pages
https://doi.org/10.1155/2020/2746845

Research Article

Deep Learning for Price Movement Prediction Using Convolutional Neural Network and Long Short-Term Memory

Can Yang, Junjie Zhai, and Guihua Tao

School of Computer Science and Engineering, South China University of Technology, Guangzhou, China

Correspondence should be addressed to Junjie Zhai; 201721041369@mail.scut.edu.cn

Received 24 March 2020; Revised 5 June 2020; Accepted 15 June 2020; Published 16 July 2020

Academic Editor: Petr Hájek

The prediction of stock price movement direction is significant in financial studies. In recent years, a number of deep learning models have gradually been applied for stock predictions. This paper presents a deep learning framework to predict price movement direction based on historical information in financial time series. The framework combines a convolutional neural network (CNN) for feature extraction and a long short-term memory (LSTM) network for prediction. We specifically use a three-dimensional CNN for data input in the framework, including the information on time series, technical indicators, and the correlation between stock indices. And in the three-dimensional input tensor, the technical indicators are converted into deterministic trend signals and the stock indices are ranked by Pearson product-moment correlation coefficient (PPMCC). When training, a fully connected network is used to drive the CNN to learn a feature vector, which acts as the input of concatenated LSTM. After both the CNN and the LSTM are trained well, they are finally used for prediction in the testing set. The experimental results demonstrate that the framework outperforms state-of-the-art models in predicting stock price movement direction.

1. Introduction

Financial time series prediction, particularly stock price movement prediction, has been one of the most difficult problems for investors and researchers. Forecasting the direction of stock price movement accurately plays a key role in determining to buy and sell a stock. However, stock price is easily affected by macro- or microeconomics, such as interest rates, exchange rates, and monetary policy, making prediction become a challenging task. Motivated by great profits in stock market investment, researchers and speculators have focused on stock market prediction research for decades. Traditional statistical methods like logistic regression, exponential average, ARIMA, and GARCH were used to predict the stock price movement [1, 2]. However, statistical methods are under an assumption that the time series is generated from a linear process and therefore exhibits a poor performance in nonlinear stock price movement prediction. Accordingly, due to the great success in nonlinear field, machine learning and deep learning methods are gradually applied in forecasting stock price movement. Most of them performed two-stage predictions, which are extracting features and then using them as input to the model to make predictions.

Feature extraction is one of the most important parts in stock prediction process. Better market features always contribute to better predictions. Technical analysis is mostly performed to extract features from the original market data [3]. Machine learning methods such as kNN, ANN, SVM, and RF are often utilized to learn the relationship between the features from the technical analysis and price movement [3, 4]. Moreover, deep learning methods, especially for CNN, which have achieved great success in computer vision and image processing, are also used for feature extraction. A time series to image conversion approach was proposed in [5], in order to help CNN extracting useful features from financial variables. Nevertheless, in the approach, the potential influence from correlated stock markets was ignored. To address this problem, a three-dimensional input tensor construction approach was designed in [6], which is capable of extracting features from correlated stock markets. Inspired by their idea, this paper also employed this three-dimensional input tensor construction approach for feature extraction. Another important part in stock prediction process is selecting or enhancing a model. Recent research studies had revealed that deep learning models are superior to traditional machine learning models in financial market prediction [7–10]. CNN [8], RNN [10], and LSTM [11] were commonly used deep learning models in predicting the stock price movement. In addition, constructing hybrid models is a popular way to enhance the performance of model, such as SVM-ANN model [12], CNN-SVM model [13], and CNN-LSTM model [14–20].

In this study, we proposed a hybrid model consisting of CNN and LSTM to predict the direction of stock price movement. On the one hand, we improved the three-dimensional input tensor for CNN to extract features. There are two differences between our approach and Hoseinzade and Haratizadeh's approach [6]. First, Hoseinzade and Haratizadeh used a diversity of financial variables including stock prices, technical indicators, and stock indices from other markets to construct a three-dimensional input tensor as the input of a specified CNN model. In their input tensor, the influence of transformation of technical indicators and the degree of correlation between other stock markets are ignored, while in our improved three-dimensional tensor, technical indicators were converted into deterministic trend signals following a certain rule and stock markets were ordered according to PPMCC. Another difference lies on that the prediction model used in [6] is a specified CNN, while in our approach, a hybrid model consisting of CNN and LSTM is employed. And the hybrid model is able to combine the advantages of CNN in feature extraction with the advantages of LSTM in time series prediction. On the other hand, we proposed a CNN-LSTM model for stock price movement forecast. Compared with other CNN-LSTM models [14–20], the main difference between them and our proposed hybrid model lies on the CNN-based feature extraction module. Their feature extraction modules mainly aimed at extracting features from one-dimensional or two-dimensional input variables, while ours was aimed at three-dimensional input tensor. Different purposes lead to different structures of feature extraction modules. The final experimental results demonstrated that the improvement on input tensor and the combination of CNN and LSTM can significantly improve the prediction performance of the model.

In brief, the main contributions of this work can be summarized as follows:

(1) We built an improved three-dimensional input tensor for CNN by converting the technical indicators into deterministic trend signals and using PPMCC to order the correlated stock indices.

(2) We designed a CNN-based feature extraction module, which is suitable for extracting features from the three-dimensional input tensor.

(3) Extensive experiments demonstrated that our improvement on the three-dimensional input tensor can significantly improve prediction performance, and our proposed model outperforms several state-of-the-art models in terms of F-measure.

The rest of the paper is organized as follows. The related work is introduced in Section 2. Section 3 proposes our framework and methods. Section 4 provides extensive experiments. Finally, the conclusion is drawn in Section 5.

2. Related Work

In stock market forecast domain, the previous research approaches are usually categorized into two groups. One focuses on achieving better feature extraction from a series of financial variables. The other attempts to improve prediction performance by enhancing the models.

2.1. Feature Extraction. Extracting useful features from a diverse set of financial variables is one of the most important issues in stock price movement prediction. A better prediction performance can be gained by having better input features. Technical analysis can be used to extract market features from the original financial variables. And stock prediction often uses technical analysis to form features used as input for the models. As reported by Shynkevich et al. [3], approximately 20% of stock market prediction models use technical indicators as input features. These models used for extracting market features from technical indicators mainly include machine learning models and deep learning models.

ANN and SVM are commonly used machine learning models for feature extraction in stock market prediction. Thenmozhi and Chand [21] used SVM to extract information transmission features from six global markets over the period from 1999 to 2011 to predict stock returns. Patel et al. [4] focused on investigating the effect of feature extraction on the prediction performance of models. They employed four machine learning models, which are ANN, SVM, RF, and NB, to extract features from ten technical indicators that were converted into deterministic trend signals and then made predictions in Indian stock markets. Their results showed that converting technical indicators into deterministic trend signals is beneficial to feature extraction and hence improving prediction performance.

As a typical deep learning model, CNN had exhibited great ability for feature extraction in computer vision and image processing. Recently, it was gradually applied to extract market features in stock prediction fields. Persio and Honchar [22] used CNN to extract features from a one-dimensional input variable which is obtained from the history of close price. To compensate for the lack of sufficient information in one-dimensional input, researchers attempted to provide more sufficient financial variables for CNN to extract market features. In fact, some researchers directly used the candlestick chart as the input of CNN [23, 24]. Furthermore, instead of directly taking the image as the input of CNN, Sim et al. [25] employed high-frequency data of close price to construct the input image as the input for CNN model. Sezer and Ozbayoglu [5] proposed a time series to image conversion approach, which utilized 15 technical indicators and 15 different intervals of technical indicators to generate a 15×15 input image. However, in the approach, the potential influence from correlated stock markets was ignored. To address this problem, Hoseinzade and Haratizadeh [6] recently proposed an approach to build a three-dimensional input tensor for CNN to extract market features. And the experimental results showed the effectiveness of the three-dimensional input tensor in extracting features and hence contribute to improve the performance of the model in predicting the direction of the stock price movement.

2.2. Model Enhancement. Combining the model with other techniques is a common way to improve the prediction performance. In [26], the authors used Harmony search and GA to enhance traditional ANN model and then utilized enhanced ANN to make a prediction. And the results showed that the proposed ANN model is found as a dominant model compared with the other models. Besides, Yin and Bai [27] designed an adaptive SVR for stock data at different time scales. Experimental results showed that the improved SVR with dynamic optimization of learning parameters by PSO can achieve a better result than the traditional SVR. However, in recent years, machine learning models are challenged by deep learning models in stock market prediction [28]. By investigating the Chinese stock market, Chen et al. [8] found that the deep learning model outperforms the backpropagation, the extreme learning machine, and RBFNN in stock price prediction. Similarly, Yu and Yan [9] designed a DNN model based on PSR and LSTM to predict stock prices. By predicting multiple stock indices for different periods, they found the proposed DNN model gets a higher prediction accuracy than ARIMA, SVR, and MLP. Furthermore, a similar conclusion can be drawn in [29].

Designing a hybrid model is another popular way to enhance the prediction performance of single-structure model. In [12], a two-stage fusion approach was proposed. SVR in the first stage and the second stage involves different models, including ANN, RF, and SVR. Experiments on Indian stock market demonstrated the effectiveness of the fusion prediction models. Zhou et al. [30] developed a learning architecture by cascading the logistic regression model onto the GBDT for predicting the stock indices. Cao and Wang [13] established a hybrid prediction model, which consists of CNN and SVM, to make stock market predictions. And the results illustrated that the combination of CNN and SVM can significantly improve the model's prediction performance. Long et al. [31] proposed an end-to-end model named MFNN for feature extraction on stock price movement prediction task. In their model, both convolutional and recurrent neurons were integrated to construct the multifilter structure. Experiments on Chinese stock market index CSI300 showed the superiority of MFNN to traditional machine learning models, statistical models, CNN, RNN, and LSTM in terms of the accuracy, profitability, and stability. In fact, a more commonly used hybrid model is the CNN-LSTM model [14–20]. For example, in [14], the authors found that the CNN-LSTM model is superior to LSTM and CNN in stock price movement prediction. In [17], Li et al. added an attention mechanism to the CNN-LSTM model and further improved its scalability and prediction accuracy. Similarly, Zhou et al. [18] developed a generic framework by using LSTM and CNN for adversarial training to predict stock price direction in the high-frequency stock market and achieved significant results.

3. The Proposed Framework

The architecture of our proposed model is illustrated in Figure 1, which is comprised of three major steps, including input data representation, CNN for feature extraction, and LSTM for prediction.

3.1. Data Representation

3.1.1. Data Labelling. In the field of forecasting stock price movement, the price movement direction often was classified into two classes: up and down [6, 32]. Class labels indicate the movement direction of the stock price. In this paper, the labels are computed by using the daily close price of a stock index. Let C_t be the close price for a stock index on day t. The class label for the t-th day is defined as

$$\text{target}_t = \begin{cases} 1, & C_{t+1} > C_t, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

3.1.2. Transformed Deterministic Signals. It is well known that technical indicators are widely used in stock market prediction. In this paper, we employ ten technical indicators and convert them into deterministic trend signals for prediction since Jigar et al. [4] demonstrated that trend deterministic values of technical indicators are better than the native values of technical indicators in stock trend forecasting. Table 1 presents the specific details.

3.1.3. Input Tensor Building. In [32], the authors ordered the features in the two-dimensional input matrix according to the correlation between instances and features before they are presented as input to the CNN. And their results showed that the CNN with a specifically ordered features outperforms CNN that utilizes randomly ordered features. Inspired by their idea, we try to apply this correlation to the three-dimensional input tensors for CNN.

In Figure 2, we show the representation of the three-dimensional input tensor. In this paper, i, j, and k are 10, 10, and 11, respectively. In the proposed framework shown in Figure 1, the input is a three-dimensional tensor, each dimension of which represents the number of technical indicators, the number of trading days, and the number of correlated stock indices, where there are i converted deterministic variables from the technical indicators for each of

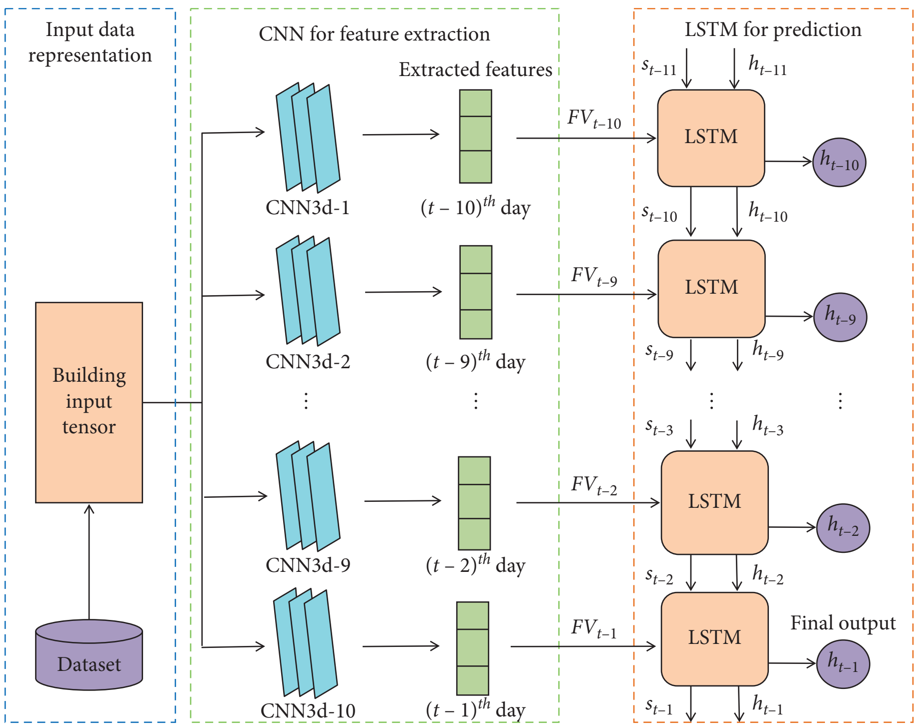

FIGURE 1: The framework of the proposed model for prediction.

TABLE 1: Description of selected technical indicators and deterministic signals.

Name of indicators	Formulas	Rules for deterministic trend signals
Simple moving average	$SMA_t = C_t + C_{t-1} + \cdots + C_{t-n}/n$	If $C_t > SMA_t$, label "1"; otherwise, label "0"
Weight moving average	$WMA_t = n \cdot C_t + (n-1) \cdot C_{t-1} + \cdots + C_{t-n}/n + (n-1) + \cdots + 1$	If $C_t > WMA_t$, label "1"; otherwise, label "0"
Momentum	$MOM_t = C_t - C_{t-n}$	If $MOM_t > 0$, label "1"; otherwise, label "0"
Stochastic K%	$K_t = C_t - LL_{t-(n-1)}/HH_{t-(n-1)} - LL_{t-(n-1)} \times 100$	If $K_t > K_{t-1}$, label "1"; otherwise, label "0"
Stochastic D%	$D_t = \sum_{i=0}^{n-1} K_{t-i}/n$	If $D_t > D_{t-1}$, label "1"; otherwise, label "0"
Moving average convergence divergence	$MACD_t = MACD_{t-1} + 2/n + 1\,(DIFF_t - MACD_{t-1})$ $DIFF_t = EMA(12)_t - EMA(26)_t$	If $MACD_t > MACD_{t-1}$, label "1"; otherwise, label "0"
Relative strength index	$RSI_t = 100 - 100/1 + (\sum_{i=0}^{n-1} UP_{t-i}/n)/(\sum_{i=0}^{n-1} DW_{t-i}/n)$	If $RSI_t <\ = 30$ or $RSI_t > RSI_{t-1}$, label "1," and if $RSI_t >\ = 70$ or $RSI_t < RSI_{t-1}$, label "0"
William's %R	$WR_t = H_n - C_t/H_n - L_n \times 100$	If $WR_t > WR_{t-1}$, label "1"; otherwise, label "0"
Commodity channel index	$CCI_t = M_t - SM_t/0.015D_t, M_t = H_t + L_t + C_t/3,\ SM_t = \sum_{i=1}^{n} M_{t-i+1}/n,$ $D_t = \sum_{i=1}^{n} \lvert M_{t-i+1} - SM_t \rvert/n$	If $CCI_t < -200$ or $CCI_t > CCI_{t-1}$, label "1," and if $CCI_t > 200$ or $CCI_t < CCI_{t-1}$, label "0"
Accumulation/ distribution oscillator	$AD_t = H_t - C_{t-1}/H_t - L_t$	If $AD_t > AD_{t-1}$, label "1"; otherwise, label "0"

C_t, L_t, and H_t denote the close price, low price, and high price at time t, respectively; LL_t and HH_t represent, respectively, lowest low and highest high in the last t days; UP_t means upward price change while DW_t is the downward price change at time t. EMA refers to the exponential moving average, $EMA(k)_t = EMA(k)_{t-1} + \alpha \times (C_t - EMA(k)_{t-1})$, $\alpha = 2/k + 1$, and k denotes the time period of k–day exponential moving average.

these markets, j days used for prediction, and k correlated market indices.

Different from [6], in our three-dimensional input tensor, the technical indicators are transformed into deterministic trend signals and the stock indices are ranked by PPMCC. Actually, PPMCC is one of the most common measurements of determining linear dependence, which is capable of reflecting the degree of linear correlation between two variables [33, 34]. The calculation formula is as follows:

$$\rho_{p,q} = \frac{\sum_{t=1}^{n}\left(x_{p,t} - \overline{x_p}\right)\left(x_{q,t} - \overline{x_q}\right)}{\sqrt{\sum_{t=1}^{n}\left(x_{p,t} - \overline{x_p}\right)^2 \sum_{t=1}^{n}\left(x_{q,t} - \overline{x_q}\right)^2}}, \qquad (2)$$

where $x_{p,t}$ and $x_{q,t}$ are the values of the p-th and the q-th feature on the t-th day index. $\overline{x_p}$ and $\overline{x_q}$ are the average values of the p-th and q-th feature. n represents the number of data. If $\rho_{p,q} > 0$, there is a positive correlation, and if $\rho_{p,q} < 0$, it is negatively correlated; otherwise, it is linearly independent.

In detail, we take the calculation of PPMCC between S&P 500 and DJIA as an example. $x_{p,t}$ and $x_{q,t}$ in equation (2) are close prices of S&P 500 and DJIA on the t-th day, respectively. $\overline{x_p}$ and $\overline{x_q}$ are the corresponding average of the close price of S&P 500 and DJIA. n is the number of trading days in S&P 500. Following equation (2), we can obtain the PPMCC between S&P 500 and DJIA. The PPMCC between S&P 500 and the other 10 stock indices can also be obtained in a similar way. And the results can be found in Figure 3. Therefore, the order of stock indices in the three-dimensional input tensor is S&P 500, NASDAQ, DJIA, RUSSELL, NYSE, DAX, N225, FTSE, CAC40, HSI, and SSE.

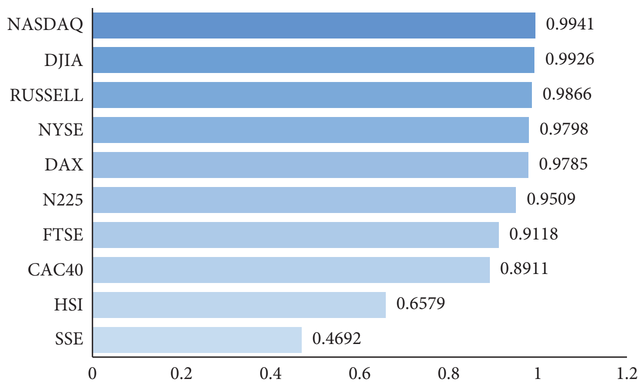

FIGURE 2: Three-dimensional input tensor with a 1 × 1 filter.

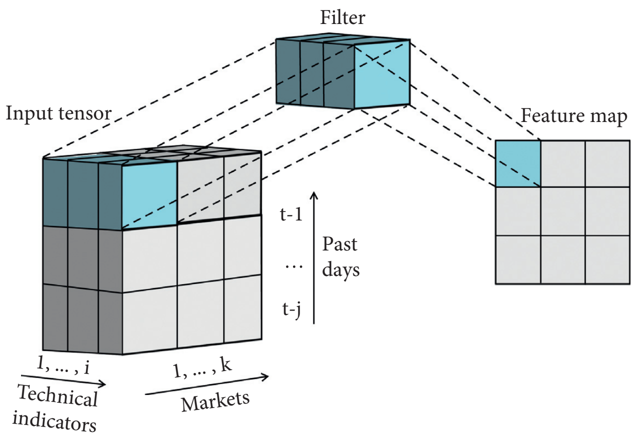

FIGURE 3: PPMCC between SP500 and the other stock indices.

3.2. CNN for Feature Extraction.

In general, the CNN model includes several layers [35], such as the input layer, the convolutional layer, the pooling layer, the fully connected layer, and the output layer. In this paper, we do not employ the pooling layer because Yang et al. [36] claimed in the financial study that if a pooling layer is adopted, the information would probably be lost. Specifically, the convolutional layer is designed for performing convolution operations on the input data. Actually, the convolution operation can be considered as a filter used for the input data. The size of a filter suggests its coverage. Moreover, all the filters share the same weights in the convolution operation, and the weights are updated in training. Similar to [6], Figure 2 exhibits how the 1 × 1 filter works in the three-dimensional input tensor. Next, a fully connected layer is used for linking the flattened layer to the output layer, which is a MLP network that can perform the prediction and classification operations.

Inspired by [32], the authors used a parallel convolutional layer to generate multiple time series representations of different time scales and achieved significant results. And in the proposed CNN feature extraction module, there are 5 layers, including a parallel convolutional layer, a merge layer, two convolutional layers, and a flattened layer, which are shown in the virtual line frame of Figure 4. In the parallel layer, the convolutions for different branches are independent of each other. In the merged layer, all extracted features of parallel layers are concatenated. Then, the concatenated feature will be processed by the remaining two convolutional layers. Finally, the flattened layer obtains the feature vector. Notably, the fully connected and the output layer are only used for training, and in testing, the LSTM network replaces them and is concatenated with the feature vector generated from the flattened layer.

A specific configured in the CNN feature extraction module shows that the input tensor is a matrix of 10 by 11 with a depth of 10. The parallel convolutional layers perform 1 × 1 and 3 × 3 convolutional operations, and the filters both are ten, after which there is one convolutional layer with ten 3 × 5 filters, and in the next convolutional layer, ten 3 × 1 filters are utilized. By the way, in each convolutional layer, the padding method takes "same." Then, a flattened layer is used to generate the feature vector. When training, the flattened layer is concatenated with a fully connected network consisting of two hidden layers: the first layer has 10 neurons and the second layer has 2 neurons. Specifically, the loss function is categorical cross entropy, epochs are 24, and batch size is 32 in our experiments. Finally, the "softmax" activation function is employed in the output layer.

3.3. LSTM for Prediction.

In the combination model, the LSTM network, concatenated with the trained CNN, is used for final prediction. Specifically, the feature vector generated

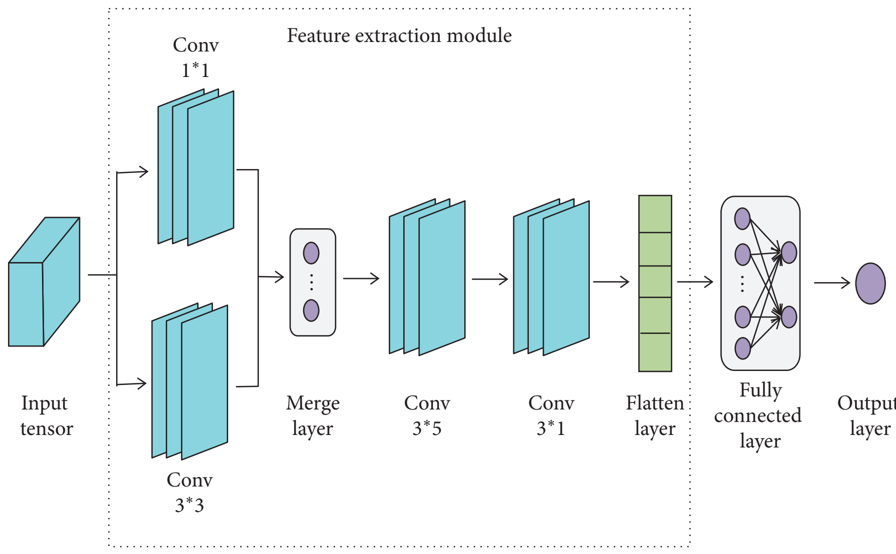

Figure 4: The structure of CNN feature extraction module.

from the flattened layer acts as the input for the LSTM network to make a prediction. The LSTM network is comprised of an input layer, a hidden layer, and an output layer. In detail, the hidden layer, including the memory cells, is the main characteristic of LSTM networks. Each of the memory cells has three gates designed for maintaining and adjusting its cell state s_t: a forget gate (f_t), an input gate (i_t), and an output gate (o_t). Specifically, each of the gates can be considered a filter to fulfill a certain purpose. The forget gate and the input gate define which information to remove from and add to the cell state, respectively. The output gate specifies which information from the cell state will be utilized as output.

Figure 5 illustrates the structure of a memory cell. We formulate the LSTM model to process time series of stock indices, referring to the literature [37]. During a forward pass, h_t denotes an output of LSTM at day t and can be calculated as follows:

$$\begin{cases} f_t = \operatorname{sigmoid}\left(W_{fv}v_t + W_{fh}h_{t-1} + b_f\right), \\ \tilde{s}_t = \tanh\left(W_{\tilde{s}v}v_t + W_{\tilde{s}h}h_{t-1} + b_{\tilde{s}}\right), \\ i_t = \operatorname{sigmoid}\left(W_{iv}v_t + W_{ih}h_{t-1} + b_i\right), \\ s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s}_t, \\ o_t = \operatorname{sigmoid}\left(W_{ov}v_t + W_{oh}h_{t-1} + b_o\right), \\ h_t = o_t \odot \tanh\left(s_t\right), \end{cases} \tag{3}$$

where W_* is the weight matrix and v_t is the input vector at time t. f_t, i_t, and o_t are forgotten, input, and output gates at time t, respectively. $\tilde{s}_t$ and s_t denote the distorted input to the memory cell and the content of the memory cell at time t. In addition, h_t represents the value of the hidden node, and the symbol $\odot$ represents the elementwise production operation. The corresponding details of the back propagation through time are introduced in [38].

In terms of the configuration of the LSTM network, the optimizer adopts the “Adam” optimization algorithm, the loss function is a categorical cross entropy, epochs are 12,

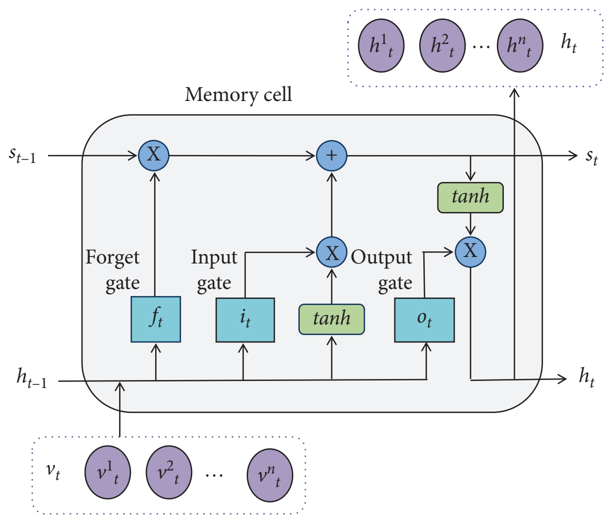

Figure 5: The structure of LSTM memory cell.

and batch size is 64. As for the time steps, the number of hidden neurons, and the dropout rate, we present the levels instead of specified values. Details can be found in Table 2. For each stock index, the determination of these parameters is according to the prediction performance on the validation set. Notably, the proposed model is implemented by Python with a version of 3.5.4. We mainly use such machine learning libraries as “Keras” and “NumPy” for various functionalities.

4. Experiments

In this study, we use 11 influential international stock market indices, including CAC40, DJIA, S&P 500, NASDAQ, DAX, FTSE, NYSE, HSI, N225, SSE, and RUSSELL. The data are from the period of January 4, 2010, to December 29, 2017. All the data are downloaded from Yahoo Finance (https://finance.yahoo.com/).

Table 2: LSTM parameters and their levels in the hybrid model.

Parameters	Levels
Time steps	6, 7, 8, . . ., 14, 15
Number of hidden neurons	50, 100, 150, 200
Dropout rate	0.1, 0.2, 0.3, 0.4

In addition, our experimental scheme for the investigation is based on the proposed deep learning framework, called "CNN3D-DR + LSTM," and the workflow of proposed model can be seen in Figure 6. Besides, the main steps are described as follows:

(1) Data preprocessing: the original dataset is used to generate the labels and the deterministic trend signals, which act as the input within three-dimensional tensors. Then, the stock indices in the three-dimensional tensor are ranked by PPMCC.

(2) Data partitioning: all the labelled data are first divided into 3 parts—the training set for training, the validation set for parameter determination, and the testing set for performance evaluation.

(3) Training: the training dataset is used to train the CNN connecting with a fully connected neural network and then the trained CNN is used to generate a series of feature vectors, which act as the input of the LSTM neural network. Next, we set different parameters and use the obtained feature vectors to train the LSTM network. Then, we use the validation set to evaluate the prediction performance of the hybrid model. The parameters of optimal prediction performance are obtained.

(4) Testing: the testing dataset and the trained CNN are used to compute the feature vectors and then they are put into LSTM with optimal parameters for predicting the direction of stock price movements.

(5) Evaluation: the prediction performance is evaluated by comparing the predicted value with the real ones.

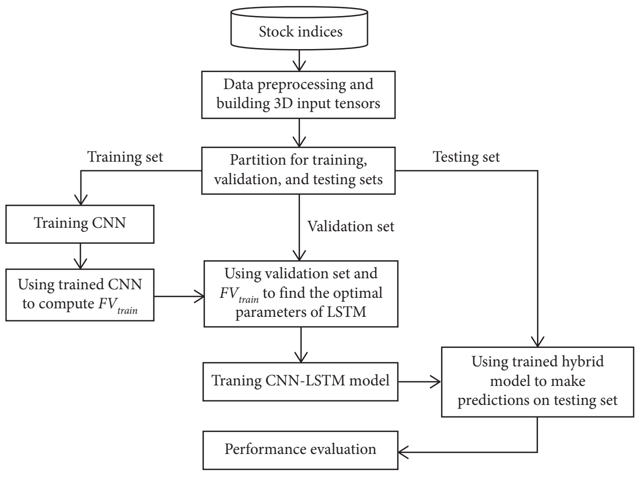
Stock indices
Data preprocessing and building 3D input tensors
Training set
Partition for training, validation, and testing sets
Testing set
Training CNN
Validation set
Using trained CNN to compute FV_{train}
Using validation set and FV_{train} to find the optimal parameters of LSTM
Traning CNN-LSTM model
Using trained hybrid model to make predictions on testing set
Performance evaluation

Figure 6: The workflow of the proposed model.

Table 3: Confusion matrix for two-class classification.

Actual/predicted	Positive	Negative
Positive	tp	fn
Negative	fp	tn

Accuracy is an important evaluation indicator. However, it may not be suitable for an unbalanced dataset [32]. For a full assessment of the prediction performance, we also take precision, recall, and F-measure into consideration. In order to evaluate the prediction performance for each class, the precision, recall, and F-measure take the mean of values for positive and negative classes. By the way, the mean of the F-measure values for positive and negative classes is also called macroaverage F-measure [6, 32]. Furthermore, we use the ROC curve that is created by plotting the TPR against the FPR at different possible thresholds to visualize the performance of the proposed models. And the AUC (area under the ROC curve) is taken as an overall performance measure because it is independent of the cutoff value. The higher the AUC value is, the better prediction performance that the model achieves.

4.1. Evaluation Methodology. The evaluation scheme is based on the confusion matrix for two-class classification shown in Table 3; here, tp, fp, fn, and tn denote true positive, false positive, false negative, and true negative counts, respectively. Precision, recall, accuracy, and F-measure are commonly used indicators to evaluate the prediction performance, and the corresponding formula is as follows:

$$\begin{cases} \text{precision} = \dfrac{\text{tp}}{\text{tp} + \text{fp}}, \\ \text{recall} = \dfrac{\text{tp}}{\text{tp} + \text{fn}}, \\ \text{accuracy} = \dfrac{\text{tp} + \text{tn}}{\text{tp} + \text{fp} + \text{tn} + \text{fn}}, \\ F - \text{measure} = \dfrac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}. \end{cases} \tag{4}$$

4.2. Experiments on S&P 500. In this section, we conduct extensive experiments on the S&P 500 to investigate the effectiveness of the proposed model. For the simplicity of description, we let "D" represent the fact that the technical indicators have been converted into deterministic trend signals, and we let "R" represent the fact that the stock indices in the three-dimensional tensor have been ranked by PPMCC. Specifically, we design several models for comparative experiments as follows:

(1) CNN3D: we take a three-dimensional input tensor as the input data for a CNN model to make a prediction. In the input tensor, the value of a technical indicator is normalized and not converted to deterministic trend signals.

(2) CNN3D-DR: in this model, the difference from CNN3D is the fact that the technical indicators in the

input tensor are transformed into deterministic trend signals and the stock indices in the tensor are ranked by PPMCC.

(3) LSTM-D: to test the LSTM network, the deterministic trend signals transformed from technical indicators are utilized as the input to make predictions.

(4) CNN3D + LSTM: regardless of the deterministic trend signals, the CNN is used to extract features from the three-dimensional input tensor, while the LSTM network is employed for making predictions. The input tensor used here is the same as in CNN3D.

(5) CNN3D-D + LSTM: in the three-dimensional input tensor, the technical indicators are transformed into deterministic trend signals but the stock indices are not ranked. And the CNN is used to extract features, while the LSTM network is used to make predictions.

(6) CNN3D-DR + LSTM: in contrast to CNN3D + LSTM, in the input tensor, the technical indicators are transformed into deterministic trend signals and the stock indices are ranked by PPMCC.

First of all, we divide the dataset into three parts: training set, validation set, and testing set. The validation set is used to determine the optimal parameters in LSTM network. Here, we define γ as the ratio of training set and validation set to the testing set. For example, $\gamma = 80/20$ means that the ratio of training set and validation set is 80% of dataset, while the testing set is 20%. For simplicity, we set the ratio between training set and validation set as 4 : 1. Table 4 shows the macroaverage F-measure of CNN3D-DR + LSTM with different parameters on validation set in S&P 500 when $\gamma = 80/20$. And we can find the optimal time steps in LSTM is 6, the optimal number of hidden neurons is 100, and the optimal dropout rate is 0.3.

Then, we design a group of experiments with different sizes of the training set and the testing set to detect the suitability and robustness of the proposed framework. We conduct experiments on S&P 500 and exhibit the average prediction results of the experiments with γ. γ can be set to 60/40, 65/35, 70/30, 75/35, and 80/20, and Table 5 presents the corresponding optimal parameters. Furthermore, we show the average performance of different models with different γ in Table 6. For a clearer visualization, we illustrate the results in Figure 7.

To compare the results of CNN3D and CNN3D-DR, we find that CNN3D-DR can provide better average performance. In particular, in the comparison between the CNN3D + LSTM and the CNN3D-DR + LSTM, the CNN3D-DR + LSTM shows significant superiority compared to the CNN3D + LSTM, which demonstrates that the improvement of three-dimensional input tensor can significantly improve the prediction accuracy. Furthermore, neither CNN3D-DR nor LSTM-D defeats CNN3D-DR + LSTM, indicating that the hybrid model is effective in improving prediction performance. In brief, the CNN3D-DR + LSTM outperforms the others in the given situation and demonstrates that the improvement of three-dimensional input tensor and the combination of CNN and LSTM can improve the prediction performance. To better evaluate the performance of stock price movement direction prediction, we illustrate the ROC curves of different experiment groups where γ takes 80/20 in Figure 8, from which we can find that the area under the ROC curve (AUC) of the proposed model is larger than the others.

4.3. Comparison with Other Models. In addition, we conduct a group of experiments to evaluate the performance of the proposed model compared with several state-of-the-art models. We apply all the models in predicting stock price movement direction on five different stock indices, which are S&P 500, DJIA, NASDAQ, NYSE, and RUSSELL, respectively. In the comparison with other models, we divide the dataset into 2 parts: the first 80% of the data is used for training, while the remaining 20% acts as the testing data. Accordingly, the performance of these models is compared in terms of the average macroaverage F-measure.

In terms of the proposed hybrid model, Table 7 shows its optimal parameters on different stock indices. Besides, in other models, the same parameter settings reported in the original paper are used. The details of other models are described as follows:

(1) PCA + ANN [39]: first, the initial data are mapped to a new feature space by using PCA. Then, we use the resulting representation of the data to train a three-layered ANN for stock price direction prediction. In the hidden layer, the number of neurons is set to 10 and a tangent sigmoid function is used. And a logistic sigmoid transfer function is used in the output layer.

(2) SVM [4]: ten technical indicators are represented as trend deterministic data and are then fed into SVM to predict stock price index movement. For each stock, the optimal parameters of SVM are obtained from several given parameter levels. By the way, the selected ten technical indicators are same as this paper.

(3) CNN-cor [32]: the feature set is extracted from different technical indicators, price and temporal information, and then ordered by the correlations between instances and features. Finally, the ordered features are used to build a two-dimensional input matrix for the specified CNN to predict the direction of stock price movement.

(4) CNNpred [6]: a diverse set of financial variables, including technical indicators, stock indices, commodities, future contracts, etc., is used to construct three-dimensional input tensors. Then, the input tensors are fed into a specified CNN model to make predictions.

(5) CNN + LSTM: we implement a common CNN-LSTM model for comparison. In the model, ten technical indicators are used to construct a two-dimensional input data for CNN to extract features. The ten technical indicators are the same as those in [4] and the parameters of CNN are the same as this

Table 4: The macroaverage F-measure of CNN3D-DR + LSTM with different parameters on validation set in S&P 500 when $\gamma = 80/20$.

Number of hidden neurons	Dropout rate	Time steps in LSTM									
		$n = 6$	$n = 7$	$n = 8$	$n = 9$	$n = 10$	$n = 11$	$n = 12$	$n = 13$	$n = 14$	$n = 15$
$N = 50$	0.1	0.6289	0.6107	0.6213	0.5865	0.5583	0.5912	0.5708	0.5546	0.5368	0.5767
	0.2	0.6057	0.6055	0.5928	0.5759	0.5565	0.6254	0.5720	0.5888	0.5787	0.5492
	0.3	0.6177	0.6070	0.5801	0.6235	0.5943	0.6059	0.5640	0.5622	0.5445	0.5445
	0.4	0.6147	0.6207	0.5822	0.6070	0.5604	0.6131	0.5593	0.5599	0.5559	0.5582
$N = 100$	0.1	0.6095	0.6033	0.6156	0.5980	0.5883	0.6223	0.5739	0.5572	0.5467	0.5380
	0.2	0.5879	0.6069	0.5792	0.5874	0.5988	0.6442	0.5753	0.5821	0.5783	0.5505
	0.3	**0.6506**	0.6190	0.5980	0.5950	0.5540	0.6217	0.5723	0.5993	0.5545	0.5555
	0.4	0.5998	0.6107	0.5996	0.5950	0.5681	0.6028	0.5681	0.5991	0.5405	0.5593
$N = 150$	0.1	0.6095	0.6295	0.5950	0.5794	0.5644	0.5884	0.5811	0.5555	0.5559	0.5613
	0.2	0.6170	0.6010	0.5876	0.6021	0.5876	0.6258	0.5940	0.5458	0.5915	0.5622
	0.3	0.6057	0.6093	0.5876	0.6137	0.5816	0.6187	0.5786	0.5678	0.5368	0.5626
	0.4	0.5863	0.6123	0.5749	0.6026	0.5597	0.6203	0.5487	0.5686	0.5661	0.5603
$N = 200$	0.1	0.6043	0.6198	0.6040	0.5809	0.5712	0.6284	0.5375	0.5661	0.5456	0.5245
	0.2	0.5981	0.6174	0.5869	0.6033	0.5621	0.5919	0.5708	0.5858	0.5467	0.5670
	0.3	0.6035	0.6093	0.6047	0.5988	0.5943	0.6150	0.5895	0.5970	0.5268	0.5513
	0.4	0.6078	0.6204	0.5869	0.5853	0.5742	0.6269	0.5357	0.5670	0.5473	0.5783

Table 5: Optimal parameters of CNN3D-DR + LSTM for different γ in S&P 500.

γ	Time steps in LSTM	Number of hidden neurons	Dropout rate
60/40	8	100	0.1
65/35	9	50	0.1
70/30	6	50	0.3
75/25	6	100	0.4
80/20	6	100	0.3

Table 6: The average performance of the experiments with different γ.

Metrics	γ	CNN3D	CNN3D-DR	LSTM-D	CNN3D + LSTM	CNN3D-D + LSTM	CNN3D-DR + LSTM
Accuracy	60/40	0.5082	0.4957	0.5202	0.4993	0.5693	0.5833
	65/35	0.5095	0.5009	0.5348	0.5130	0.5605	0.5728
	70/30	0.4798	0.4952	0.5339	0.5079	0.5991	0.6160
	75/25	0.4856	0.5200	0.5229	0.5150	0.5968	0.6316
	80/20	0.4878	0.5180	0.5175	0.5010	0.5653	0.5916
	Average	0.4942	0.5060	0.5258	0.5072	0.5782	0.5991
Precision	60/40	0.5019	0.4954	0.5124	0.4991	0.5671	0.5818
	65/35	0.5071	0.4985	0.5317	0.5029	0.5529	0.5650
	70/30	0.4904	0.4927	0.5150	0.5004	0.5929	0.6107
	75/25	0.5045	0.5165	0.5006	0.5222	0.5921	0.6281
	80/20	0.4949	0.5183	0.5089	0.5009	0.5620	0.5894
	Average	0.4998	0.5043	0.5137	0.5051	0.5734	0.5950
Recall	60/40	0.5017	0.4964	0.5107	0.4992	0.5635	0.5779
	65/35	0.5062	0.4975	0.5276	0.5027	0.5513	0.5621
	70/30	0.4938	0.4937	0.5130	0.5002	0.5910	0.6094
	75/25	0.5034	0.5155	0.5003	0.5218	0.5897	0.6249
	80/20	0.4951	0.5184	0.5087	0.5008	0.5596	0.5860
	Average	0.5001	0.5043	0.5121	0.5049	0.5710	0.5921
F-measure	60/40	0.4954	0.4951	0.4962	0.4986	0.5602	0.5754
	65/35	0.4989	0.4980	0.5154	0.5000	0.5507	0.5609
	70/30	0.4632	0.4923	0.5043	0.4989	0.5912	0.6096
	75/25	0.4791	0.5157	0.4911	0.5141	0.5894	0.6250
	80/20	0.4875	0.5163	0.5072	0.4995	0.5577	0.5845
	Average	0.4848	0.5035	0.5028	0.5022	0.5698	0.5911

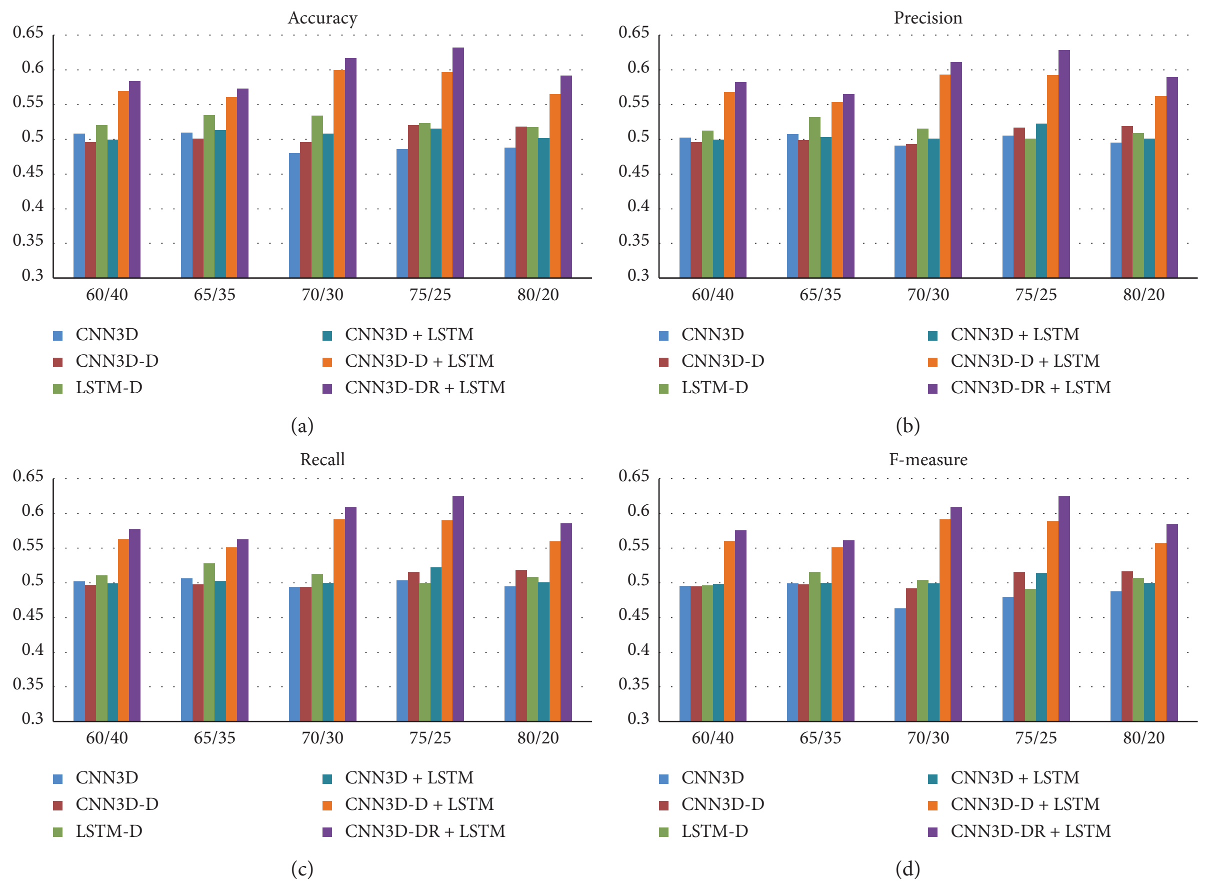

FIGURE 7: The average performance of the experiments with different γ. (a) Accuracy. (b) Precision. (c) Recall. (d) F-measure.

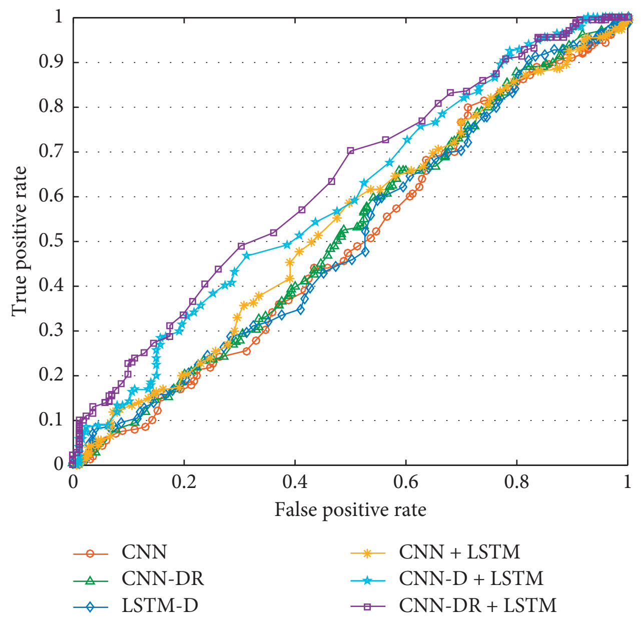

FIGURE 8: ROC curves of different models when γ = 80/20.

Table 7: Optimal parameters of the proposed model in different stock indices when γ takes 80/20.

Stock index	Time steps in LSTM	Number of hidden neurons	Dropout rate
S&P 500	6	100	0.3
DJIA	10	50	0.1
NASDAQ	9	200	0.3
NYSE	6	50	0.1
RUSSELL	10	200	0.1

Table 8: Average macroaverage F-measure of different models.

Market model	PCA + ANN	SVM	CNN-cor	CNNpred	CNN + LSTM	CNN3D-DR + LSTM
S&P 500	0.4469	0.4963	0.3928	0.4837	0.5165	0.5864
DJIA	0.4150	0.4595	0.3900	0.4979	0.5036	0.6215
NASDAQ	0.4199	0.4284	0.3796	0.4931	0.5081	0.5860
NYSE	0.4070	0.4299	0.3906	0.4751	0.5007	0.5910
RUSSELL	0.4525	0.5102	0.3924	0.4846	0.5241	0.5987
Average	0.4283	0.4649	0.3891	0.4869	0.5106	0.5967

Table 9: Best macroaverage F-measure of different models.

Market model	PCA + ANN	SVM	CNN-cor	CNNpred	CNN + LSTM	CNN3D-DR + LSTM
S&P 500	0.5627	0.5208	0.5723	0.5532	0.5345	0.6061
DJIA	0.5518	0.5594	0.5253	0.5612	0.5280	0.6418
NASDAQ	0.5487	0.4776	0.5498	0.5576	0.5423	0.6005
NYSE	0.5251	0.5012	0.5376	0.5592	0.5214	0.6053
RUSSELL	0.5665	0.5115	0.5602	0.5787	0.5476	0.6181
Average	0.5510	0.5141	0.5491	0.5620	0.5348	0.6144

paper. Besides, LSTM is utilized for price direction forecasting. The time steps, number of hidden neuron, and dropout rate are 10, 50, and 0.1, respectively.

Table 8 shows the average results. In addition, we also show the best performance of the models in Table 9. The experimental results on different stock indices demonstrate that the proposed model is superior to the other common models, including ANN, SVM, CNN, and CNN + LSTM.

5. Conclusion

This paper presented a combined deep learning framework with CNN and LSTM neural networks to predict the stock price movement direction. First, we improved the three-dimensional input tensor by transforming the technical indicators into deterministic trend signals and ranking the correlated stock indices according to PPMCC. Then, we designed a CNN-based module for feature extraction. Finally, we employed a LSTM network for stock price movement direction prediction.

Extensive experiments demonstrated that the deterministic trend signals and the ranked stock indices in the three-dimensional input tensor play a significant role in improving the prediction performance. Moreover, the result of comparing with several state-of-the-art models showed the superiority of the proposed model in predicting direction of the stock price movement.

In future work, it would probably be a core challenge to design better learning models via intelligently extracting more valuable features to further improve the prediction performance.

Abbreviations

ANN:	Artificial neural network
ARIMA:	Autoregressive integrated moving average
CAC40:	CAC40 index
CNN:	Convolutional neural network
DAX:	DAX performance index
DJIA:	Dow Jones industrial average
DNN:	Deep neural network
FPR:	False positive rate
FTSE:	FTSE 100 index
GA:	Genetic algorithm
GARCH:	Generalized autoregressive conditional heteroscedasticity
GBDT:	Gradient boosted decision tree
HSI:	Hang Seng index
kNN:	k-nearest neighbor
LSTM:	Long short-term memory
MFNN:	Multifilter neural network
MLP:	Multilayer perception
N225:	Nikkei 225 index
NASDAQ:	NASDAQ composite index
NB:	Naive Bayes
NYSE:	New York stock exchange index

PPMCC:	Pearson product-moment correlation coefficient
PSO:	Particle swarm optimization
PSR:	Phase-space reconstruction
RBFNN:	Radial basis function neural network
RF:	Random forest
RNN:	Recurrent neural network
ROC:	Receiver operating characteristic
RUSSELL:	RUSSELL 2000 index
S&P 500:	S&P 500 index
SSE:	SSE composite index
SVM:	Support vector machine
SVR:	Support vector regression
TPR:	True positive rate.

Data Availability

The data used to support the findings of this study can be downloaded from Yahoo Finance (https://finance.yahoo.com/).

Conflicts of Interest

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

References

[1] J. Sun, K. Xiao, C. Liu, W. Zhou, and H. Xiong, "Exploiting intra-day patterns for market shock prediction: a machine learning approach," *Expert Systems With Applications*, vol. 127, pp. 272–281, 2019.

[2] Z. Lin, "Modelling and forecasting the stock market volatility of sse composite index using garch models," *Future Generation Computer Systems*, vol. 79, pp. 960–972, 2018.

[3] Y. Shynkevich, T. M. McGinnity, S. A. Coleman, A. Belatreche, and Y. Li, "Forecasting price movements using technical indicators: investigating the impact of varying input window length," *Neurocomputing*, vol. 264, pp. 71–88, 2017.

[4] J. Patel, S. Shah, P. Thakkar, and K. Kotecha, "Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques," *Expert Systems with Applications*, vol. 42, no. 1, pp. 259–268, 2015.

[5] O. B. Sezer and A. M. Ozbayoglu, "Algorithmic financial trading with deep convolutional neural networks: time series to image conversion approach," *Applied Soft Computing*, vol. 70, pp. 525–538, 2018.

[6] E. Hoseinzade and S. Haratizadeh, "Cnnpred: cnn-based stock market prediction using a diverse set of variables," *Expert Systems with Applications*, vol. 129, pp. 273–285, 2019.

[7] Y. Chen, W. Lin, and J. Z. Wang, "A dual-attention-based stock price trend prediction model with dual features," *IEEE Access*, vol. 7, pp. 148047–148058, 2019.

[8] L. Chen, Z. Qiao, M. Wang, C. Wang, R. Du, and H. E. Stanley, "Which artificial intelligence algorithm better predicts the Chinese stock market?" *IEEE Access*, vol. 6, pp. 48625–48633, 2018.

[9] P. Yu and X. Yan, "Stock price prediction based on deep neural networks," *Neural Computing and Applications*, vol. 132, pp. 1–20, 2019.

[10] H. M, G. E. A., V. K. Menon, and S.K. P., "Nse stock market prediction using deep-learning models," *Procedia Computer Science*, vol. 132, pp. 1351–1362, 2018.

[11] S. Borovkova and I. Tsiamas, "An ensemble of lstm neural networks for high-frequency stock market classification," *Journal of Forecasting*, vol. 38, no. 6, pp. 600–619, 2019.

[12] J. Patel, S. Shah, P. Thakkar, and K. Kotecha, "Predicting stock market index using fusion of machine learning techniques," *Expert Systems with Applications*, vol. 42, no. 4, pp. 2162–2172, 2015.

[13] J. Cao and J. Wang, "Stock price forecasting model based on modified convolution neural network and financial time series analysis," *International Journal of Communication Systems*, vol. 32, no. 12, p. e3987, 2019.

[14] S. Jain, R. Gupta, and A. A. Moghe, "Stock price prediction on daily stock data using deep neural networks," in *Proceedings of the 2018 International Conference on Advanced Computation and Telecommunication (ICACAT)*, IEEE, New York, NY, USA, pp. 1–13, 2018.

[15] J. Eapen, D. Bein, and A. Verma, "Novel deep learning model with cnn and bi-directional lstm for improved stock market index prediction," in *Proceedings of the 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, IEEE, New York, NY, USA, pp. 264–270, 2019.

[16] X. Zhan, Y. Li, R. Li, X. Gu, O. Habimana, and H. Wang, "Stock price prediction using time convolution long short-term memory network," in *Proceedings of the International Conference on Knowledge Science, Engineering and Management*, Springer, Berlin, Germany, pp. 461–468, 2018.

[17] C. Li, X. Zhang, M. Qaosar, S. Ahmed, K. M. R. Alam, and Y. Morimoto, "Multi-factor based stock price prediction using hybrid neural networks with attention mechanism," in *Proceedings of the 2019 IEEE International Conference on Dependable, Autonomic and Secure Computing, International Conference on Pervasive Intelligence and Computing, International Conference on Cloud and Big Data Computing, International Conference on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, IEEE, Berlin, Germany, pp. 961–966, 2019.

[18] X. Zhou, Z. Pan, G. Hu, S. Tang, and C. Zhao, "Stock market prediction on high-frequency data using generative adversarial nets," *Mathematical Problems in Engineering*, vol. 34, 2018.

[19] J. Liu, Y. Chen, K. Liu, and J. Zhao, "Attention-based event relevance model for stock price movement prediction," in *Proceedings of the China Conference on Knowledge Graph and Semantic Computing*, Springer, Berlin, Germany, pp. 37–49, 2017.

[20] P. Oncharoen and P. Vateekul, "Deep learning using risk-reward function for stock market prediction," in *Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence*, pp. 556–561, Berlin, Germany, 2018.

[21] M. Thenmozhi and G. Sarath Chand, "Forecasting stock returns based on information transmission across global markets using support vector machines," *Neural Computing and Applications*, vol. 27, no. 4, pp. 805–824, 2016.

[22] L. D. Persio and O. Honchar, "Artificial neural networks architectures for stock price prediction: comparisons and applications," *International Journal of Circuits, Systems and Signal Processing*, vol. 10, pp. 403–413, 2016.

[23] S.-J. Guo, F.-C. Hsu, and C.-C. Hung, "Deep candlestick predictor: a framework toward forecasting the price movement from candlestick charts," in *Proceedings of the 2018 9th International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*, IEEE, Berlin, Germany, pp. 219–226, 2018.

[24] K. Jearanaitanakij and B. Passaya, "Predicting short trend of stocks by using convolutional neural network and candlestick patterns," in *Proceedings of the 2019 4th International Conference on Information Technology (InCIT)*, IEEE, Berlin, Germany, pp. 159–162, 2019.

[25] H. S. Sim, H. I. Kim, and J. J. Ahn, "Is deep learning for image recognition applicable to stock market prediction?" *Complexity*, vol. 10, 2019.

[26] M. Göçken, M. Özçalıcı, A. Boru, and A. T. Dosdoğru, "Integrating metaheuristics and artificial neural networks for improved stock price prediction," *Expert Systems with Applications*, vol. 44, pp. 320–331, 2016.

[27] Y. Guo, S. Han, C. Shen, Y. Li, X. Yin, and Y. Bai, "An adaptive svr for high-frequency stock price forecasting," *IEEE Access*, vol. 6, pp. 11397–11404, 2018.

[28] R. Singh and S. Srivastava, "Stock prediction using deep learning," *Multimedia Tools and Applications*, vol. 76, no. 18, pp. 18569–18584, 2017.

[29] Q. Wang, W. Xu, and H. Zheng, "Combining the wisdom of crowds and technical analysis for financial market prediction using deep random subspace ensembles," *Neurocomputing*, vol. 299, pp. 51–61, 2018.

[30] F. Zhou, Q. Zhang, D. Sornette, and L. Jiang, "Cascading logistic regression onto gradient boosted decision trees for forecasting and trading stock indices," *Applied Soft Computing*, vol. 84, p. 105747, 2019.

[31] W. Long, Z. Lu, and L. Cui, "Deep learning-based feature engineering for stock price movement prediction," *Knowledge-Based Systems*, vol. 164, pp. 163–173, 2019.

[32] H. Gunduz, Y. Yaslan, and Z. Cataltepe, "Intraday prediction of borsa istanbul using convolutional neural networks and feature correlations," *Knowledge-Based Systems*, vol. 137, pp. 138–148, 2017.

[33] M.-T. Puth, M. Neuhäuser, and G. D. Ruxton, "Effective use of Pearson's product-moment correlation coefficient," *Animal Behaviour*, vol. 93, pp. 183–189, 2014.

[34] J. Guo and X. Li, "Prediction of index trend based on lstm model for extracting image similarity feature," in *Proceedings of the 2019 International Conference on Artificial Intelligence and Computer Science*, pp. 335–340, New York, NY, USA, 2019.

[35] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," *The Handbook of Brain Theory and Neural Networks*, vol. 3361, no. 10, p. 1995, 1995.

[36] H. Yang, Y. Zhu, and Q. Huang, "A multi-indicator feature selection for cnn-driven stock index prediction," in *Proceedings of the International Conference on Neural Information Processing*, Springer, Berlin, Germany, pp. 35–46, 2018.

[37] C. Yang, S. Ren, Y. Liu, H. Cao, Q. Yuan, and G. Han, "Personalized channel recommendation deep learning from a switch sequence," *IEEE Access*, vol. 6, pp. 50824–50838, 2018.

[38] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: a search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[39] X. Zhong and D. Enke, "Forecasting daily stock market return using dimensionality reduction," *Expert Systems with Applications*, vol. 67, pp. 126–139, 2017.