Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

AComNN: Attention enhanced Compound Neural Network for financial time-series forecasting with cross-regional features

Zhen Yang [a], Jacky Keung [a], Md Alamgir Kabir [a], Xiao Yu [b,*], Yutian Tang [c], Miao Zhang [a], Shuo Feng [a]

[a] *Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong, China*
[b] *School of Computer Science and Technology, Wuhan University of Technology, Wuhan, China*
[c] *School of Information Science and Technology, ShanghaiTech University, Shanghai, China*

ARTICLE INFO

Article history:
Received 14 December 2020
Accepted 24 June 2021
Available online 1 July 2021

Keywords:
Financial time-series forecasting
Deep learning
Attention mechanism
Hang Seng Index

ABSTRACT

In recent years, many works spring out to adopt the forecast-based approach to support the investment decision in the financial market. Nevertheless, most of them do not consider mining the hidden patterns in the cross-regional financial time-series. However, the fluctuation in financial markets has always been affected by the global economy, instead of a single market. To overcome this issue, this article proposes an Attention enhanced Compound Neural Network (AComNN) that can be applied on features of multiple-sources, including different financial markets and economic entities. The proposed novel approach compounds of Artificial Neural Network (ANN), Long Short-Term Memory (LSTM), and self-attention to progressively capture the time-zone-dependent context behind the financial time-series across regions with multiple filters. Thereby, it provides trading signals for supporting the financial investment decision. The proposed AComNN has been applied on the Hong Kong Hang Seng Index (HSI) trend prediction based on various initial features across regions. The experimental result demonstrates that the AComNN achieves the highest average accuracy for the one-day ahead trend prediction over 60%. Besides, it reveals highly superior competitiveness on the forecasting capability improved by 13.36% on average compared with the baselines. Therefore, we encourage to adopt the proposed method to the practitioners and provide a new thought, considering the analysis of cross-regional features, in the financial time-series forecasting.

1. Introduction

Due to the volatile, nonlinear, complicated, and chaotic characteristics of the financial market, accurately forecasting the trend of the financial time-series has always been challenging [1]. In recent years, a series of well-designed machine-learning-based trading systems emerge for assisting investors or speculators in identifying financially rewarding stocks and exercising their ownership [2,3].

Previous traditional studies mainly adopt some univariate time-series models for the prediction in the financial market, such as AutoRegressive Moving Average model (ARMA) [4], AutoRegressive Integrated Moving Average model (ARIMA) [5–7] and Generalized AutoRegressive Conditional Heteroskedast (GARCH) [8]. However, only considering the influence of its historical behaviors on future movements of the price, the univariate model structure and their simple market pattern assumptions lead to the low financial forecasting capability in the practical application.

Apart from the traditional time-series models, machine learning models also have been adopted in the financial time-series forecasting for years due to their more substantial capability of learning, ease of interpretability, and absence of the presumption, such as Support Vector Machine (SVM) [9,10], Support Vector Regression (SVR) [11], Logistic Regression (LR) [12], Random Forest (RF) [13], eXtreme Gradient Boosting (XGBoost) [14], Decision Tree (DT) [15] as well as a series of ensemble models of stacking [16] and bagging [17].

In recent years, deep learning has been widely applied to various research fields such as pattern recognition, image classification, and autopilot, which obtained great success. Because of their robust fitting and nonlinear mapping capability, researchers also have designed various deep learning models to implement the forecasting in the financial market, such as Long Short-Term Memory (LSTM) [2,18], Convolutional Neural Network (CNN) [19],

* Corresponding author.
E-mail addresses: zhyang8-c@my.cityu.edu.hk (Z. Yang), Jacky.Keung@cityu.edu.hk (J. Keung), makabir4-c@my.cityu.edu.hk (M.A. Kabir), xiaoyu@whut.edu.cn (X. Yu), tangyt1@shanghaitech.edu.cn (Y. Tang), miazhang9-c@my.cityu.edu.hk (M. Zhang), shuofeng5-c@my.cityu.edu.hk (S. Feng).

https://doi.org/10.1016/j.asoc.2021.107649
1568-4946/© 2021 Elsevier B.V. All rights reserved.

Artificial Neural Networks (ANN) [20], Graph Convolutional Neural Network (GCNN) [21] and other hybrid neural networks [22–25].

Nevertheless, two issues suppress the forecasting capability of the above techniques. These are — (1) most of the previous studies of financial market prediction only focus their features on the relationship of inter-markets restricted in one region or even on a single targeting market, obstructing crucial information transmission from the outside market, i.e., information insufficiency. (2) Besides, their models are unable to capture the crucial hidden patterns behind the financial time-series across areas, due to the lack of corresponding adaptation to the multi-regional features, i.e., structural deficiency.

Therefore, in this work, we adopt the Hang Seng Index (HSI) trend prediction task by taking as an example to solve the above two issues. For the information insufficiency, we adopt cross-regional features as the initial input from two perspectives. On the one hand, we collect the technical indicators extracted from the Financial Times Stock Exchange 100 Index (FTSE 100), Standard & Poor's 500 (S&P 500) and HSI in the area of London, New York, and Hong Kong respectively. On the other hand, we collect other highly associated economic indicators such as macro-economic indicators, commodity indicators, and currency exchange indicators obtained from regions of the U.K., the U.S., and China.

For the structural deficiency issue, we propose a novel Attention enhanced Compound Neural Network (AcomNN) for extracting features from multiple sources, which is constructed of the steps of ANN, LSTM, and self-attention in order. The ANN step is responsible for preliminarily extracting semantics from each region and uniform their feature dimensions. The LSTM step horizontally further transfers the refined semantics among regions according to their time-series relation on time zone. Finally, the self-attention step can dynamically focus on the decisive parts of regions for the weights allocation, thereby progressively capturing the time-zone-dependent context behind the financial time-series across regions with multiple filters.

In the experimental stage, we evaluate the AComNN on the HSI prediction with cross-regional features collected from Apr. 2003 to Dec. 2019. The experimental result demonstrates that the highest average accuracy for the one-day ahead HSI trend prediction can up to 60.81%. Compared with the state-of-the-art baselines, our AComNN outperforms them on average over 13%, simultaneously with a relatively very low standard deviation of 0.0355. Additionally, we also implement the trading simulation based on the trading signal provided by the AComNN. The final accumulative return during the simulation can up to 35.04% on average, showing that the performance of the AComNN based investment advisory system is highly competitive and practical in the real world.

The main contributions of our work are as follows:

1. We mitigate the information insufficiency by integrating the cross-regional features of multiple stock markets and economic entities that constitute the raw features.
2. We propose a fine-designed Attention enhanced Compound Neural Network (AcomNN), which can progressively capture the time-series relation characteristics and dynamically allocate attention for cross-regional features in each time zone.
3. We explore the performance of AComNN under different feature smoothing and forecasting windows configuration. Also, we re-implement three state-of-the-art financial forecasting models [16,19,22] to compare with the proposed AComNN, and the experimental results prove that our proposed AComNN achieves an encouraging result among the baselines.

The rest of the paper is organized as follows: Section 2 summarizes the related literature concerning the financial time-series forecasting. Section 3 presents the associated data collection and preparation. Section 4 elaborates on the model construction method. Section 5 demonstrates the experimental design. Section 6 reports the experimental results. Section 7 discusses the whole experiment in forecasting capability and trading simulation. Section 8 discusses the threats to validity of this work. Finally, Section 9 concludes the paper and outlines the future work.

2. Related work

2.1. Feature study in financial markets

Since the price variation in the financial market is highly fluctuating and full of unexpected noises, features studies including determinant factors selection and dimension reduction play a critical role in effectively boosting the accuracy for financial market prediction and mitigating overfitting in training.

In [26], Garefalakis et al. study the determinant factors that influence HSI tendency and conclude that S&P 500 in the U.S. stock exchanges, gold, and crude oil prices play a substantial role in the Hong Kong stock market. Valukonis [27] statistically analyzes several China's stock indices, such as Shanghai Shenzhen CSI 300 index, Shanghai Stock Exchange (SSE) composite index, and ShenZhen Stock Exchange (SZSE) composite index data from 2008 to 2012. And he concludes that the GDP growth and inflation rate are the key factors that affect the Chinese stock market tendency. Weng et al. [28] explore the 23 macro-economic indicators concerning the U.S. market, such as oil price, unemployment rate, trade balance, and monetary supply. Then they adopt three ensemble models and four time-series models to predict four major U.S stock indices including the Dow Jones Industrial Average (DJIA) index, the New York Stock Exchange (NYSE) composite index, the National Association of Securities Dealers Automated Quotation (NASDAQ) composite index, and the S&P 500 index. In addition, technical indicators, as another critical part of market features, have also been adopted in stock prediction. Kara et al. [29] apply ten technical indicators such as momentum, stochastic K%, and RSI in SVM and ANN to predict the movement of the Istanbul Stock Exchange (ISE) national 100 index.

Furthermore, a tremendous amount of feature selection methods are also put forward in financial time-series forecasting because overwhelmed irrelevant features may mislead the prediction models. Tanaka-Yamawaki and Tokuoka [30] adopt evolution computing to implement technical indicators selection on eight stocks from NYSE, thereby to carry on the intra-day forecast. Wang [31] study the HSI and Korea Composite Stock Price Index (KOSPI) moving tends and apply the Principal Component Analysis (PCA) to implement the feature selection and dimension reduction before the classification process. Nti et al. [13] utilize the random forest to operate the feature selection among a series of macro-economic indicators in the Ghana Stock Exchange (GSE), thereby to further predict a 30-day ahead stock-price. Nevertheless, these methods cannot construct a model in an end-to-end manner; instead, models change in feature extraction and objective prediction, causing it difficult to determine their influence on the whole framework.

2.2. Statistical models in financial time-series

At the early stage, researchers adopt traditional AutoRegressive (AR) models as the prediction engine to deal with financial

Table 1
Each stock market open and close time.

	Open	Close	Time zone
London stock exchange	08:00	16:30	UTC +1
New York stock exchange	13:00	20:00	UTC −4
NASDAQ exchange	13:00	20:00	UTC −4
Hong Kong stock exchange	01:30 (day +1)	8:00 (day +1)	UTC +8

time-series forecasting. Ariyo et al. [6] exploit ARIMA to implement the stock prediction for NYSE and Nigeria Stock Exchange (NSE). Kocak [4] put forward a new high-order fuzzy ARMA(p,q) and apply it to the prediction of the stock market in Turkey. Afterward, other improved traditional AR models are continuously involved in the forecasting of financial markets, such as the AutoRegressive Conditional Heteroskedasticity model (ARCH) by Zumbach et al. [32], GARCH by Lin [8], and VAR by Ülkü et al. [33].

Subsequently, more complex machine learning techniques have steadily become commonly adopted in financial forecasting for their more powerful capabilities to find internal patterns between features and objectives. Wang et al. [17] and Nair et al. [15] adopt tree-based models to forecast the stock indices in China and India. Luo et al. [10] propose an improved PLR-WSVM to overcome four deficiencies that occurred in the previous one. Besides, Jiang et al. [16] exploit the ensemble learning to integrate several tree-based models and simple deep learning models to construct the complex stacking framework to predict the tendency for the U.S market.

In recent years, deep learning-based models as the bionics of human brain structure have been applied in more and more data science fields, including the stock forecasting domain. Rundo [34] propose a deep LSTM model with reinforcement learning layers to forecast the price trend in the high-frequency foreign currency exchange market. Hoseinzade et al. [19] put forward a CNNpred model to analyze the five principal U.S. stock market indices from a three-dimensional perspective. Long et al. [22] propose a Multi-Filters Neural Network (MFNN) for extracting the features by multiple filters, which also achieved some success in the CSI300 index.

However, most of the works we mentioned above either did not exploit the cross-regional features to forecast their objectives or cannot capture the internal patterns from their gathered data. To this end, our motivation is to bridge the gap of the above deficiencies and put forward our own high performing prediction engine in the investment advisory system that can capture the critical patterns between features and labels under the global market.

3. Data collection and preparation

Regionally, our cross-regional features can be divided into three parts, i.e., from the U.K., the U.S., and China. The data comprises both technical indicators and other economic indicators in each region.

Technical indicators are the same in each region, extracted either from FTSE 100, S&P 500, or HSI. The FTSE 100 is composed of 100 constituent stocks in the London stock exchange. The S&P 500 consists of 500 constituent stocks in the NYSE and NASDAQ exchange, while the HSI consists of 50 constituent stocks in the Hong Kong exchange. The open and close time for each of the above stock exchanges are listed in Table 1 [35], which present the time-zone relation between each index.

Other economic indicators are obtained from three major economic entities, i.e., the U.K., the U.S., and China, such as macroeconomic indicators, commodity indicators, and currency indicators, which are different among areas because of their various regional economic characteristics. These indicators can be obtained by days, months, or quarterly.

The whole data are collected from Apr. 2003 to the end of Dec. 2019, a total of 4119 instances, and the detailed information of our collected features have been recorded in Table A.1 in the Appendix section by elaborating on each feature's name, description, type, source, and calculation function. The detailed label, feature generation procedures and their configuration in our experiment are shown in the later parts.

3.1. Label generation

In our experiment, we explore the influence of the forecasting window size (ws) of our models' performance. We set $ws = \{1, 5, 10, 20\}$, where respectively denote one-day, one-week, two-week, and one-month ahead prediction. Since our objective is to forecast the moving trend of HSI, we set a binary variable $y_{t,ws}$ as the label of the instance of the day t, as shown in Eq. (1).

$$y_{t,ws} = \begin{cases} 1 & r_{t,ws} \geq 0 \\ 0 & r_{t,ws} < 0 \end{cases} \tag{1}$$

The $r_{t,ws} = \frac{close_{t+ws}}{close_t} - 1$ is the return from day t to its future ws days, where the $close_t$ represents the close price of HSI at day t while $close_{t+ws}$ represents the close price of day $t + ws$.

3.2. Feature generation

Although lots of literature have proposed various determinant indicators that are highly related to stock index price movement of HSI, S&P 500, DJIA, NASDAQ composite index, and FTSE 100 [13,16,19,26–30], there is no unified criterion on related feature selection. In this work, our raw data are retrieved from regions of the U.K., the U.S., and China, including Open, High, Low, Close price, and Volume (called as OHLCV variables) of FTSE 100, S&P 500, and HSI as well as regional indicators such as macroeconomic indicators, commodity indicators, currency indicators. See Table A.1 in Appendix for details.

After we collect all the indicators in the range of Apr. 2003 to Dec. 2019 from each public source, we mainly implement four steps to pre-process those raw data for the preparation of feeding into our proposed model.

(1) **Smoothing:** One of the most challenging parts of stock prediction is the extreme volatility. Smoothing with the exponential moving average can effectively reduce the unexpected variations and noises in the stock price movement, which is a practical approach to capture a relatively long-term moving trend in the stock market. Some of the previous works [14,16] have adopted the smoothing procedure for their raw data; however, they just set a constant value for smoothing factor (α) without further exploration. Thus, in our work, we uniformly set four candidate $\alpha = \{0.095, 0.3, 0.5, 0.9\}$ to explore the influence of various smoothing effects on our prediction model, where larger α gives more weights to the current time-step. OHLCV variables and all other economic indicators are smoothed by the exponential moving average when they are collected. The exponential smooth can be defined by the following Eq. (2) in a recursive way:

$$\begin{aligned} & S_0 = X_0, \\ & for \quad t > 0, \quad S_t = \alpha * X_t + (1 - \alpha) * S_{t-1}, \end{aligned} \tag{2}$$

where the X_t represents the feature vector in time step t, α is the exponential smoothing factor, and each S_t is recursively calculated by the current X_t and its previous S_{t-1}.

(2) **Calculate technical indicators:** After smoothing each raw feature, we calculate 19 technical indicators based on the OHLCV variables of FTSE 100, S&P 500, and HSI respectively, which can be referred to Table A.1 in Appendix.

(3) **Substitute absolute indicators with their relative rate:** For each indicator, including technical indicators and other economic indicators, we convert their absolute values to relative rates since relative rates have stronger correlations with the labels of the price trend [10,16]. Specifically, we define the relative rate calculation in Eq. (3):

$$\hat{X_t} = X_t / X_{t-1} - 1, \tag{3}$$

where X_t represents the feature vector at the time step t and $\hat{X_t}$ represents the relative rate of the original feature vector at the time step t.

(4) **Data normalization:** Since different features are in different data magnitude, the larger value of features may overwhelm those smaller ones. As such, data normalization has always been critical and commonly used in multivariate machine learning methods. In this work, we adopt the Z-score normalization method for our feature matrix, which can be defined in Eq. (4):

$$\tilde{X} = \frac{X - E[X]}{\sqrt{Var[X]}}, \tag{4}$$

where the $\tilde{X}$ is the normalized feature matrix, X is the original feature matrix, and $E[X]$ and $Var[X]$ are the mean and variance of the X by features.

(5) **Align the date of each indicator:** Since features from different regions or markets have different releasing cycles or different holiday arrangements, causing the fact that some indicators may be released while others may not on the same day, i.e., missing values may appear on some features in a sample. For instance, the Hong Kong exchange will not open on the Chinese lunar new year, while the other three stock exchanges will open on that day, leading the HSI related data to be missing. Furthermore, due to the time-zone difference (as shown in Table 1), when the HSI trend forecasting is implemented, we can only obtain the features (including the technical indicators and other economic indicators) of the previous day for the U.K., and the U.S. region. In this case, to avoid utilizing future data in prediction, we align these two regions' data on day $t-1$ with the HSI of the day t. However, for the regional features in China, we directly align them with HSI by date due to the absence of the time-zone difference. Finally, if missing values still exist in the dataset, we fill those empty positions by their values in previous time steps.

3.3. Back-testing arrangement

In our study, we experiment on each combination of α and ws. To obtain a stable evaluation for the model performance, under each α and ws combination, we configure five groups of continuous historical data for back-testing experiments from July 2017 to December 2019, where the backtest is conducted every six months. Thereby, when applying to the practical trading, we will still re-train the model by every six months with the best α and ws combination to try to follow and reproduce the experimental performance. Table 2(a) presents that for each combination of α and ws, we configure same five groups of back-testing data, while Table 2(b) presents the sample distribution labeled with Increase (+1) or Decrease (0) in each group of back-testing dataset, when the $ws = 1$.

Table 2
Research data statistics.

(a) Five groups of backtests configuration for each combination of ws and α

Group #	Train set	Validate set	Test set
1	04/2003–12/2016	01/2017–06/2017	07/2017–12/2017
2	04/2003–06/2017	07/2017–12/2017	01/2018–06/2018
3	04/2003–12/2017	01/2018–06/2018	07/2018–12/2018
4	04/2003–06/2018	07/2018–12/2018	01/2019–06/2019
5	04/2003–12/2018	01/2019–06/2019	07/2019–12/2019

(b) Samples distribution for $ws = 1$

Group #	Train set		Validate set		Test set	
	Increase	Decrease	Increase	Decrease	Increase	Decrease
1	1767	1616	68	53	74	51
2	1835	1669	74	51	68	53
3	1909	1720	68	53	64	61
4	1977	1773	64	61	70	48
5	2038	1837	70	48	65	61

So, to summarize, we set four candidate values for both the smoothing factor $\alpha = \{0.095, 0.3, 0.5, 0.9\}$ and the forecasting window sizes $ws = \{1, 5, 10, 20\}$. And for each combination of ws and α, we apply the AComNN on five groups of back-testing experiments. Thus totally we generate $4 * 4 * 5 = 80$ datasets to explore the influence of ws and α on our model prediction performance in this work.

4. Attention enhanced Compound Neural Network

In this section, we introduce the construction of our proposed Attention enhanced Compound Neural Network (AComNN). The whole framework includes three main steps: ANN Step, LSTM Step, and Self-Attention Step. Fig. 1 depicts the general framework of the AComNN.

4.1. Before the AComNN construction

As aforementioned in Section 3, our processed features are retrieved from three regions: the U.K., the U.S., and China. In each area, features consist of technical indicators extracted from its corresponding stock market index (i.e., FTSE 100 in the London, S&P 500 in the New York, and HSI in Hong Kong) and other economic indicators with its regional characteristic. Before feeding the features of each region into the model, we arrange them according to the time zone order of the U.K. (UTC +1), the U.S. (UTC −4), then China (UTC +8, day +1) to better refine their time-series relations. Additionally, we partition the technical indicators and other economic indicators in each region to extract the high-frequency (i.e., technical indicators) and low-frequency (i.e., other economic indicators) semantics separately. Up to now, we have ready six-part of features in the order of time zone.

4.2. The artificial Neural Network step

Artificial Neural Network (ANN) was the first kind of network structure inspired by the human brain system that tries to let machines imitate how humans learn. It is composed of a series of fully connected layers customarily used for conveying information via non-linear mappings in the network [36].

In our framework, the prepared features are first fed into the six ANN networks to extract their high dimensional semantics and uniform the feature dimension in each input tunnel. Specifically, features in each layer will first dot product with layers' trainable weights as a linear transformation, which is formally defined in Eq. (5):

$$v_i^j = \sum_k v_k^{j-1} w_{k,i}^{j-1}, \tag{5}$$

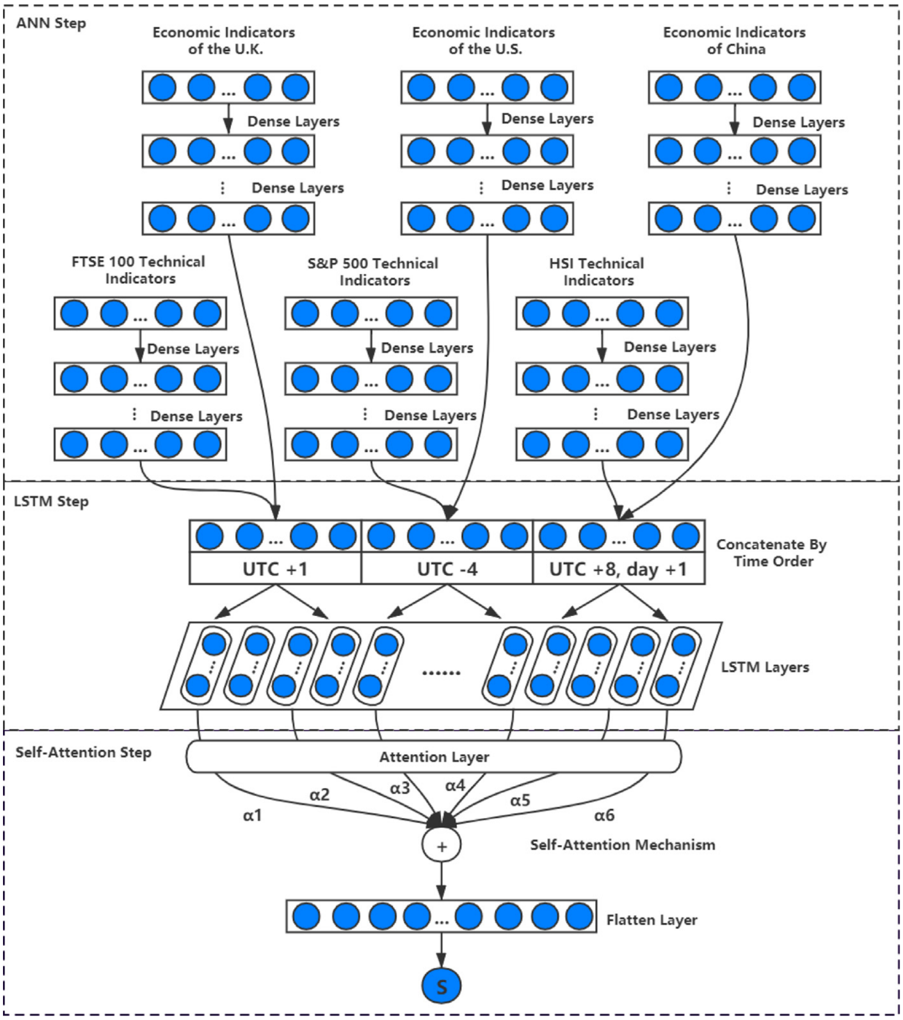

Fig. 1. Attention enhanced Compound Neural Network.

where v_i^j represents the linear transformed result of neural i at the layer j, and $w_{k,i}^{j-1}$ represents the connection weight between neural i of layer j and neural k of layer $j-1$.

In order to speed up the model convergence and mitigate the gradient vanishing and explosion, we adopt batch normalization on the output of the above linear transform [37–39]. Specifically, for each layer, it can be defined in Eqs. (6) and (7):

$$\hat{V}^j = \frac{V^j - E[V^j]}{\sqrt{Var[V^j]}}, \tag{6}$$

$$Y^j = \gamma \hat{V}^j + \beta, \tag{7}$$

where V_j represents a batch of linear transformed features at the layer j, $E[\cdot]$ and $Var[\cdot]$ represent the mean and variance of each batch, respectively, γ and β represent the parameters of batch normalization those to be learned in training, and Y^j represents the output of batch normalization at layer j.

Afterwords, we adopt Rectified Linear Unit (ReLU) as our activation function for ANN Step. More formally, for each layer, it can be defined in Eq. (8):

$$f(y_i^j) = max(0, y_i^j), \tag{8}$$

where y_i^j is the batch normalized features of neural i in layer j.

Through all the above procedures in each layer, the features of six tunnels concatenate together by the same order to be prepared as the input of the LSTM Step.

4.3. The long short-term memory step

LSTM was first proposed in [40], enhanced from Recurrent Neural Network (RNN), mainly applied in sequential or temporal data. Unlike traditional RNNs that have gradient vanishing and explosion problems, which make it impossible to deal with long-term dependencies between features, LSTM provides three gates (i.e., forget gate, input gate, and output gate) to keep the vital information transferred in the network. More formally, their mathematical definitions are shown below in Eqs. (9), (10), (11), (12), (13), and (14):

$$F_t = \sigma(X_t W_{x,f} + H_{t-1} W_{h,f} + b_f), \tag{9}$$

$$I_t = \sigma(X_t W_{x,i} + H_{t-1} W_{h,i} + b_i), \tag{10}$$

$$O_t = \sigma(X_t W_{x,o} + H_{t-1} W_{h,o} + b_o), \tag{11}$$

$$\tilde{C}_t = tanh(X_t W_{x,c} + H_{t-1} W_{h,c} + b_c), \tag{12}$$

$$C_t = F_t \bigodot C_{t-1} + I_t \bigodot \tilde{C}, \tag{13}$$

$$H_t = O_t \bigodot tanh(C_t), \tag{14}$$

where X_t represents the feature vector at time step t; H_{t-1} represents the hidden states of the previous time step; $W_{x,f}$, $W_{h,f}$, and b_f represent the trainable weight vector at forget gate F_t; $W_{x,i}$, $W_{h,i}$, and b_i represent the trainable weight vector at input gate I_t; $W_{x,o}$, $W_{h,o}$, and b_o represent the trainable weight vector at output gate O_t; $W_{x,c}$, $W_{h,c}$, and b_c represent the trainable weight vector for calculating the candidate memory $\tilde{C_t}$; C_t stores the long term memory up to the current time step, which is generated by calculating the weighted sum of both $\tilde{C_t}$ and previous long term memory C_{t-1}; Finally, the current hidden state H_t is obtained by fusing the information from O_t and C_t.

In this step, we adopt the bidirectional LSTM neural network to capture the relation pattern between features of three time zones from six tunnels. The bidirectional spread can enhance the information connection and learn the internal relation pattern from both the forward and backward time dependency [41,42]. Besides, we also adopt batch normalization in each layer of LSTM, which is similar to the ANN Step. At the end of the LSTM Step, we output the hidden states of each time-step for the preparation of the later Self-Attention step.

4.4. The self-attention step

Attention mechanism has been widely applied in many fields such as machine translation [43,44], relation classification [45], and images content description [46], and obtained outstanding achievements for it can automatically and individually set different attention (i.e., weight) to each part of the features.

In this work, the input features are collected from different economic entities and stock markets with multiple categories, and the importance of their refined semantics to each day's prediction varies. Yet, humans cannot easily capture these vital relations in time. Since the self-attention mechanism can dynamically and empirically allocate attention to each part of the regions based on the massive amount of data-driven learning, leading to a more effective manner in the harness of the cross-regional information.

In our AComNN, we follow the steps in [45] and adopt the self-attention mechanism after the features are extracted from the LSTM Step. Therefore, the self-attention module can allocate weights to the hidden states in each time-step dynamically. The following Eqs. (15), (16), and (17) illustrate each step of self-attention mechanism we applied:

$$M = tanh(H), \tag{15}$$

$$\alpha = softmax(w^T M), \tag{16}$$

$$r = H\alpha^T, \tag{17}$$

where $H = [h_1, h_2, h_3, h_4, h_5, h_6]$ represents the hidden states output from the LSTM Step, composed of six time-steps, w is a trainable parameter vector, and the softmax layer is used for generating the corresponding weights for each time-step's hidden states. Finally, the weighted hidden states in each time-step will be added together to form the output of the Self-Attention Step, as shown in Eq. (17). Afterward, we flatten the tensor from the Self-Attention Step and adopt a single neuron with a sigmoid activation function to output a scalar in the range of (0,1) to denote the HSI price upward ($\geq$ 0.5) or downward ($<$ 0.5). The loss function we adopt is the binary cross-entropy loss which is defined in Eq. (18):

$$Loss = -\frac{1}{n}\sum_i (y_i log\hat{y}_i + (1-y_i)log(1-\hat{y}_i)), \tag{18}$$

where $\hat{y}_i$ represents the predicted result of ith instance in a mini-batch while y_i represents its corresponding actual label. Besides, we adopt Adam [47] as our network optimizer for its efficiency in computation and convergence as well as its excellent performance in solving gradient with high noise.

4.5. Mitigating overfitting

Overfitting has always been a critical problem in the training stage of deep learning. It behaves as the phenomenon of low training loss while high test loss. In our experiments, we mainly adopt two kinds of approaches for our AComNN to mitigate the overfitting, i.e., stochastic dropout and regularization.

(1) Stochastic dropout has been a useful trick and widely applied in the deep learning field to prevent overfitting [48, 49]. It makes specific amounts of neurons stop working with a certain probability p ($p = 0.5$ in this work) in the training stage. As such, the model can be more generalized, for it does not rely too much on certain local features, thereby mitigating the overfitting [50].

(2) Regularization as another overfitting prevention trick is also adopted in the each layer of our model, which includes $L1$ and $L2$ regularization. $L1$ regularization prevents the overfitting by sparsing the weight matrix while the L2 regularization adopts weights decay to mitigate overfitting [51].

5. Experiment design

5.1. Evaluation metrics

As we mentioned in Section 3.3, we conduct five consecutive back-testing experiments under each combination of α and ws. To obtain a stable evaluation for the AComNN performance (including the forecasting capability and stability) under each combination of α and ws, we define Average Accuracy (Avg. Acc.) and Standard Deviation (Std. Dev.) respectively as our evaluation metrics. The Avg. Acc and Std. Dev. are defined in Eqs. (20) and (21). The $Accuracy_i$ represents the accuracy in the ith back-testing experiment, which can be calculated by Eq. (19). The TP_i, TN_i, FP_i, and FN_i represent the number of True Positive, True Negative, False Positive and False Negative samples in the ith back-testing experiment of a particular group.

$$Accuracy_i = \frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i} \tag{19}$$

$$Avg.Acc. = 1/5\sum_{i=1}^{5} Accuracy_i \tag{20}$$

$$Std.Dev. = \sqrt{\frac{\sum_{i=1}^{5}(Accuracy_i - Avg.Acc.)}{5}} \tag{21}$$

5.2. Experimental procedure

Fig. 2 presents the general experiment flow graph. The first part is for the AComNN experiment. It can be seen from the figure that we adopt four forecasting window sizes $ws = \{1, 5, 10, 20\}$, four smoothing factors $\alpha = \{0.095, 0.3, 0.5, 0.9\}$, and implement five consecutive backtests for each of above two variables' combination. For each back-testing experiment, we implement a model training, model selection and best model deployment procedure. Afterward, for each group of five back-testing experiments, we summarize their Avg. Acc. and Std. Dev. and compare with other groups' results on the test set. Finally, the ws and α combination with the highest Avg. Acc. will be selected as the best configuration for further evaluation and application in part 2 and part 3. The second part is for baseline comparison. We re-implement three state-of-the-art models with their best configuration recorded in their papers to make comparisons with our proposed AcomNN of best configuration on Avg. Acc and Std.Dev, i.e., forecasting capability comparison. Additionally, we

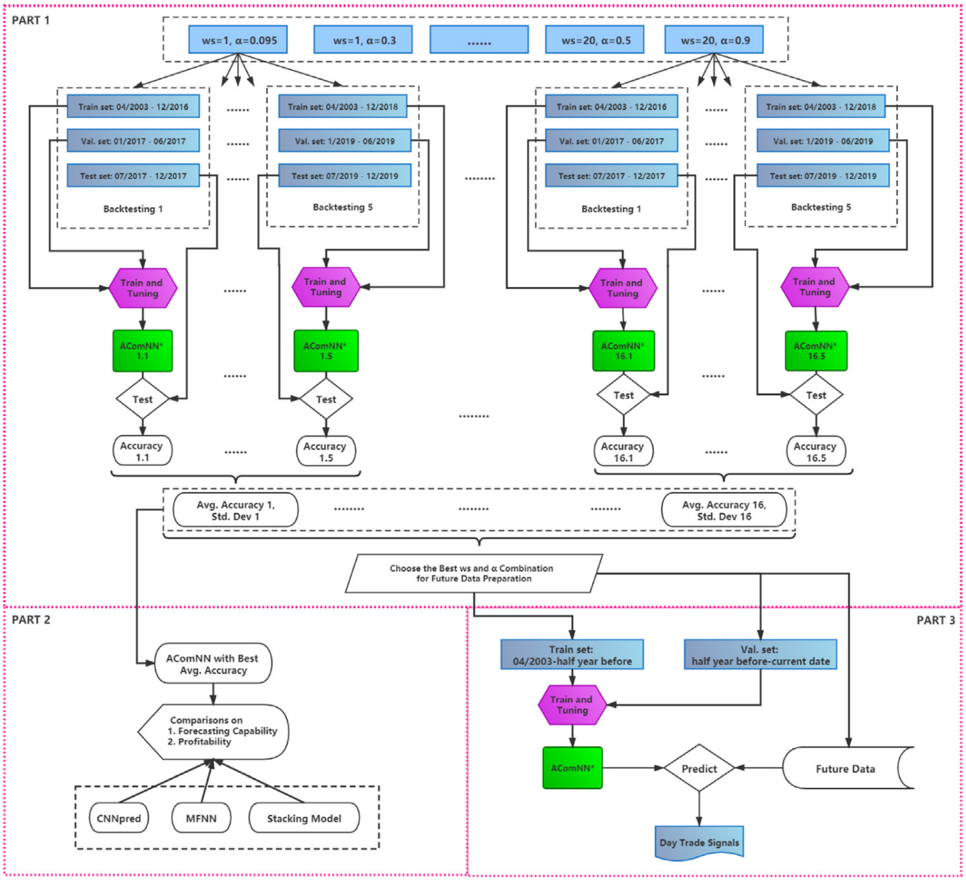
PART 1
ws=1, α=0.095
ws=1, α=0.3
......
ws=20, α=0.5
ws=20, α=0.9
Train set: 04/2003 - 12/2016
Val. set: 01/2017 - 06/2017
Test set: 07/2017 - 12/2017
Backtesting 1
Train set: 04/2003 - 12/2018
Val. set: 1/2019 - 06/2019
Test set: 07/2019 - 12/2019
Backtesting 5
Train set: 04/2003 - 12/2016
Val. set: 01/2017 - 06/2017
Test set: 07/2017 - 12/2017
Backtesting 1
Train set: 04/2003 - 12/2018
Val. set: 1/2019 - 06/2019
Test set: 07/2019 - 12/2019
Backtesting 5
Train and Tuning
AComNN* 1.1
Test
Accuracy 1.1
AComNN* 1.5
Accuracy 1.5
AComNN* 16.1
Accuracy 16.1
AComNN* 16.5
Accuracy 16.5
Avg. Accuracy 1, Std. Dev 1
Avg. Accuracy 16, Std. Dev 16
Choose the Best ws and α Combination for Future Data Preparation
PART 2
AComNN with Best Avg. Accuracy
Comparisons on
1. Forecasting Capability
2. Profitability
CNNpred
MFNN
Stacking Model
PART 3
Train set: 04/2003-half year before
Val. set: half year before-current date
Train and Tuning
AComNN*
Predict
Future Data
Day Trade Signals

Fig. 2. Experimental procedure.

also adopt the trading simulation for each of the models and measure their profitability on a series of investment evaluation metrics, i.e., profitability comparison. The third part demonstrates the future prediction-based trading strategy, which illustrates the arrangement of the future model training and prediction strategy following the same steps in back-testing experiments.

5.3. AComNN training methodology

In this subsection, we illustrate the training methodology of AComNN, which is composed of the training algorithm and model tuning.

For each time of the backtest, we tune the parameters to find the model of the parameter combination that obtains the highest accuracy on the validation set. When there are several models with the same highest accuracy, we choose the one with the lowest validation loss among them. Table 3 presents the parameters to be tuned in our model and their candidate values in the experiment.

And for each training process with a fixed model parameter combination, we follow the Algorithm 1 as our standard training procedure. We adopt Adam as our optimizer, which has been mentioned before. The initial learning rate ρ is set to 0.01, and the batch size N_{batch} is set to 256. Since Adam itself can decay the learning rate automatically, we do not implement an extra callback function for learning rate decay. During the training stage, we train 500 epochs for each backtest. Besides, during iteration, we only keep the model weights that obtain the highest accuracy on the validation set, as another method to mitigate overfitting.

Table 3
Parameters tuned in the AComNN.

(a) ANN step related parameters

Depth #	Units #	Kernel regularizer	Bias regularizer
1	32	1e−2	1e−3
2	64	1e−3	1e−4
3	128	/	/
4	256	/	/

(b) LSTM step related parameters

Depth #	Units #	Kernel regularizer	Bias regularizer
1	32	1e−4	1e−5
2	64	1e−5	1e−6
3	128	/	/
4	256	/	/

5.4. Baselines

In this section, we briefly introduce the baseline models to be compared with the proposed AComNN.

- MFNN: It is a hybrid deep learning method extracting features of previous a range of time with multiple filters of one-layer CNN, multi-layer CNN, and LSTM [22].
- CNNpred: It formulates the input features with technical indicators of five stock market indices in the U.S and related macroeconomic indicators to construct a 3D cube and adopt a 3D CNN to extract features [19].
- Stacking Model: It exploits the stacking method to integrate four tree-based classifiers and four RNN models to construct

Algorithm 1 AComNN Training Process

```
Require: Prepared training samples $X = x_1, \dots, x_n$
Initialization: Initialize parameters $\rho = 0.01$, $N_{batch} = 256$, $epoch = 0$, $accuracy_{best} = -\infty$, $loss_{acc\_best} = \infty$ and $weights_{best} = null$
while $epoch < 500$ do
    Stochastically retrieve $N_{batch}$ data from $X$ with replacement
    Fit the retrieved data with AComNN
    Calculate the $accuracy_{tmp}$ on validation set
    Calculate the $loss_{acc\_tmp}$ on validation set
    if $accuracy_{tmp} > accuracy_{best}$ then
        $weights_{best} \leftarrow$ AComNN.weights
        $accuracy_{best} \leftarrow accuracy_{tmp}$
        $loss_{acc\_best} \leftarrow loss_{acc\_tmp}$
    end if
    $epoch++$
end while
return $weight_{best}$, $accuracy_{best}$, $loss_{acc\_best}$
```

the ensemble model for feature extracting. It then adopts a Lasso logistic regression as the meta classifier to output the final forecasting results [16].

5.5. Experimental devices and tools

The experiments are conducted on a Ubuntu GPU server with four GTX1080ti GPUs of 11 GB memory for each. Our proposed AComNN is constructed by Tensorflow 2.3, which is a powerful deep learning framework.

6. Experimental results

6.1. AComNN performance results

For this part, we discuss the performance of our AComNN with different forecasting windows and smoothing factors. After we finish the first part of the experimental procedure in Fig. 2, we present Table 4 to demonstrate the best model's accuracy on the test set in each back-testing experiment, and we also list the Avg. Acc. and Std. Dev. among five backtests on the right side of the table. It is obvious that the AComNN predicting for the datasets with $ws = 1$ and $\alpha = 0.5$ obtains the highest Avg. Acc. which is 0.6081 simultaneously with a very low Std. Dev. of 0.0355 while the predicting for dataset with $ws = 1$ and $\alpha = 0.095$ obtains the lowest Std. Dev. of 0.0267 with the fourth-highest Avg. Acc. of 0.5759. The highest accuracy among all backtests is 0.8720 with $ws = 20$, $\alpha = 0.9$ during 07/-12/2017 while the lowest accuracy of 0.3051 lies in the $ws = 10$, $\alpha = 0.5$, 01/-06/2019.

Besides, we found that when $ws = 1$, as we mentioned before, the highest Avg. Acc. lies in the $\alpha = 0.5$, which may due to fact that data smoothed excessively is not conducive to short-term forecasts. However, for $ws = \{5, 10\}$, the highest Avg. Acc. is always obtained when $\alpha = 0.095$, and basically, the Avg. Acc. decreases with the enlarging of α, showing that their prediction accuracy is improved under the relatively longer-term trend captured by a smaller α. Furthermore, we also notice that setting the $\alpha = 0.9$ obtains the highest Avg. Acc. in the twenty-day ahead prediction, and the Avg. Acc. shows a decreasing tendency with the shrinking of α. A potential explanation is that the twenty-day ahead prediction is implemented in the extreme absence of the latest necessary information; with the smoothing effect increment, the available original information becomes exceedingly less and even distorted, leading the forecasting capability decrease. Concluded above, the exponential smoothing indeed can remove the unexpected noises and improve the prediction accuracy yet the prediction with different ws needs different specific smoothing factors.

In Fig. 3, each box represents the accuracy of the five consecutive backtests under a forecasting window and smoothing factor combination. It is evident that from the one-day ahead prediction to the twenty-day ahead prediction, the range of the forecasting accuracy gradually becomes large, which means with the extension of the window size, the prediction accuracy becomes more and more unstable. Because the most necessary information it needs has not been published when forecasting with a relatively large window size. Fig. 4(b) reflects the same conclusion by the line chart, the Std. Dev. becomes steadily larger with the enlarging of the ws. On the other hand, The Fig. 4(a) presents that the one-day ahead prediction always obtains the highest Avg. Acc. comparing with the prediction for other ws under the same α, since it has the chance to utilize all the latest information in the forecast. And there is a general decreasing tendency in Avg. Acc. with the expansion of the forecasting window.

Concluded from the above analysis, one-day ahead prediction obtains the highest Avg. Acc. under every smoothing factor for it has all the updated information on the current time. Combined with the smoothing factor of $\alpha = 0.5$, we obtain the best Avg. Acc. of 0.6081 among all other counterparts and also hold a relatively lower Std. Dev. of 0.0355, which means it also very stable in the back-testing. Although configured with $\alpha = 0.9$, $ws = 20$ can obtain the highest accuracy over 87%; it cannot stably get such high accuracy in other backtests. To this end, we set the forecasting window size $ws = 1$ and smoothing factor $\alpha = 0.5$ as the best configuration to pre-process our future data and make a comparison with the baselines in the later section.

6.2. Comparison result

For this section, because the authors of those baseline models did not publish their source code, we try our best to re-implement them according to their interpretation in papers [16,19,22], and make comparisons with our proposed AComNN. Similarly, we evaluate the three baseline models on the same five consecutive backtests from 07/2017 to 12/2019 and try to tune their model parameters to fit the training data. Afterwards, we calculate their average accuracy and standard deviation based on above experiments to assess their performance on the one-day ahead HSI trend prediction task.

Table 5 illustrates the comparison result between the AComNN and the other three baselines. It is clear that the AComNN outperforms the other three baselines under almost all backtests. Additionally, the AComNN obtains the highest Avg. Acc. of 0.6081 with improvements to the CNNpred, Stacking model and MFNN by 10.53%, 14.55%, and 15.01%, respectively. Furthermore, the AcomNN also obtains a very small Std. Dev. of 0.0355. Compared with the CNNpred and MFNN, the AComNN reduces the Std. Dev. by 15.19% and 6.27%, respectively; however, the Stacking model achieves the lowest Std. Dev., showing that in spite of its predictive accuracy is not very high but it has stable performance on the HSI moving trend forecast.

In addition, we also conduct the T-test to evaluate the statistical significance of difference between each model. Table 6 presents the t and p-value of each pair of models based on the one-tailed T-test, where the null hypothesis is the performance between each pair of the models are the same, while the alternative hypothesis is the row models outperform the column models. It is obvious that the p-values between the AComNN and each of the baseline are always much smaller than 0.05, statistically

Table 4
AComNN experimental results with different ws and α.

(a) forecasting window size $ws = 1$

α	07/-12/2017	01/-06/2018	07/-12/2018	01/-06/2019	07/-12/2019	Avg. Acc.	Std. Dev.
0.095	0.6000	0.5785	0.5280	0.6017	0.5714	0.5759	**0.0267**
0.3	0.6320	0.5702	0.5920	0.5932	0.5238	0.5823	0.0354
0.5	0.6240	0.6033	0.5440	0.6186	0.6508	**0.6081**	0.0355
0.9	0.6400	0.5620	0.5840	0.6102	0.5793	0.5951	0.0272

(b) forecasting window size $ws = 5$

α	07/-12/2017	01/-06/2018	07/-12/2018	01/-06/2019	07/-12/2019	Avg. Acc.	Std. Dev.
0.095	0.6640	0.4876	0.5600	0.5508	0.5714	**0.5668**	0.0566
0.3	0.5760	0.4463	0.52800	0.5847	0.5238	0.5318	**0.0493**
0.5	0.6480	0.5785	0.4800	0.4237	0.5635	0.5387	0.0785
0.9	0.6240	0.5289	0.5280	0.4830	0.5317	0.5391	0.0461

(c) forecasting window size $ws = 10$

α	07/-12/2017	01/-06/2018	07/-12/2018	01/-06/2019	07/-12/2019	Avg. Acc.	Std. Dev.
0.095	0.6880	0.4793	0.5600	0.4068	0.5794	**0.5427**	0.0951
0.3	0.6400	0.4711	0.5440	0.5000	0.5556	0.5421	**0.0576**
0.5	0.6640	0.5124	0.5040	**0.3051**	0.5635	0.5098	0.1171
0.9	0.6720	0.4876	0.3600	0.3136	0.5397	0.4746	0.1284

(d) forecasting window size $ws = 20$

α	07/-12/2017	01/-06/2018	07/-12/2018	01/-06/2019	07/-12/2019	Avg. Acc.	Std. Dev.
0.095	0.7440	0.4628	0.6240	0.3475	0.4524	0.5261	**0.1403**
0.3	0.7520	0.3884	0.5520	0.3305	0.5397	0.5125	0.1471
0.5	0.8560	0.5537	0.3200	0.5000	0.4921	0.5444	0.1745
0.9	**0.8720**	0.3719	0.6320	0.3390	0.5714	**0.5573**	0.1933

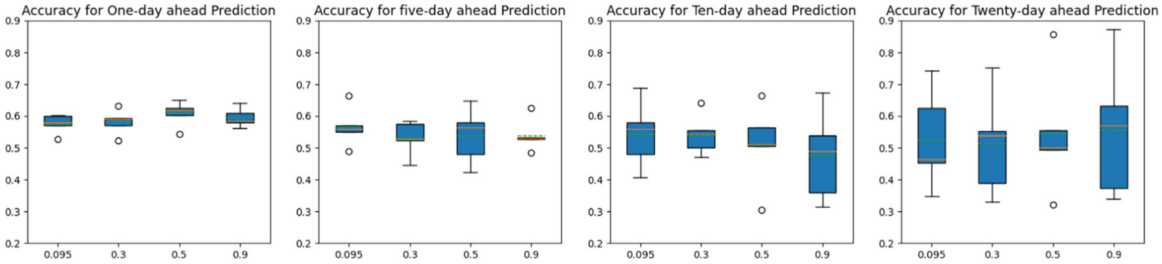

Fig. 3. AComNN overall experimental results.

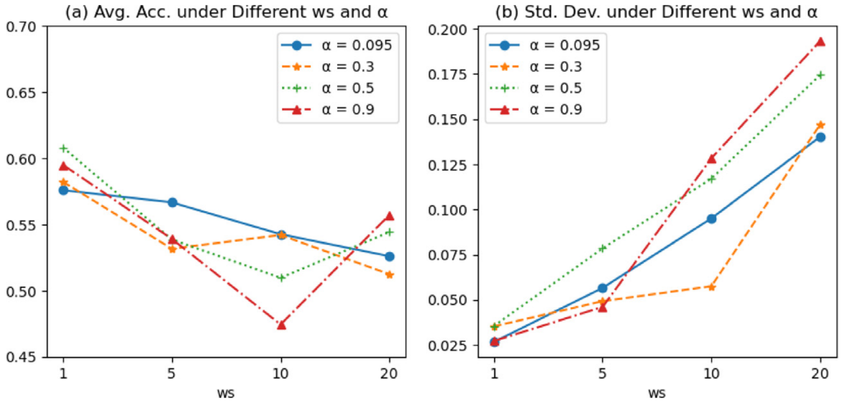

Fig. 4. The average accuracy and the standard deviation comparison among different ws and α.

Table 5
Comparison with baselines.

	Model accuracy on test set in each backtest					Avg.Acc.	Std.Dev.	Improvements on	
	07/-12/17	01/-06/18	07/-12/18	01/-06/19	07/-12/19			Avg.Acc.	Std.Dev.
AComNN	**0.6240**	**0.6033**	0.5440	**0.6186**	**0.6508**	**0.6081**	0.0355	/	/
CNNpred	0.5920	0.5620	0.4880	0.5932	0.5159	0.5502	0.0419	10.53%	**−15.19%**
Stacking	0.4803	0.5242	**0.5659**	0.5455	0.5385	0.5309	**0.0286**	14.55%	24.12%
MFNN	0.5760	0.5619	0.4960	0.5339	0.4762	0.5288	0.0379	**15.01%**	−6.27%
Average								13.36%	-0.89%

Table 6
T-test for statistical significance of difference between each models.

	AComNN		CNNpred		Stacking		MFNN	
	t	*p*-value	*t*	*p*-value	*t*	*p*-value	*t*	*p*-value
AComNN	/		2.1080	**0.0340**	3.3849	**0.0047**	3.0526	**0.0079**
CNNpred			/		0.7620	0.2340	0.7580	0.2351
Stacking					/		0.0875	0.4662
MFNN							/	

Table 7
HSI trading simulation result.

(a) 07/2017–12/2017				
	Return	Annual return	Sharpe ratio	Max. Drawdown
AComNN	**0.3304**	**0.7381**	**4.7278**	**−0.0376**
CNNpred	0.1835	0.3857	2.6717	−0.0593
Stacking	0.0048	0.0092	0.0726	−0.0670
MFNN	0.0591	0.1177	0.9159	−0.1074
(b) 01/2018–06/2018				
	Return	Annual return	Sharpe ratio	Max. Drawdown
AComNN	**0.4413**	**1.0773**	**3.8943**	**−0.0485**
CNNpred	0.1221	0.2591	1.1867	−0.0834
Stacking	−0.0510	−0.0972	−0.5542	−0.2054
MFNN	0.0558	0.1148	0.5460	−0.1021
(c) 07/2018–12/2018				
	Return	Annual return	Sharpe ratio	Max. Drawdown
AComNN	**0.1330**	**0.2735**	**1.2397**	**−0.0562**
CNNpred	−0.1196	−0.2187	−1.2855	−0.1499
Stacking	0.0728	0.1411	0.6877	−0.1368
MFNN	−0.0735	−0.1375	−0.7735	−0.1362
(d) 01/2019–06/2019				
	Return	Annual return	Sharpe ratio	Max. Drawdown
AComNN	**0.3945**	**0.9774**	**4.6725**	**−0.0385**
CNNpred	0.1536	0.3403	1.8958	−0.1126
Stacking	0.0251	0.0509	0.3214	−0.1010
MFNN	0.0702	0.1493	0.8953	−0.1027
(e) 07/2019–12/2019				
	Return	Annual return	Sharpe ratio	Max. Drawdown
AComNN	**0.4530**	**1.0617**	**4.9307**	**−0.0481**
CNNpred	−0.0085	−0.0163	−0.1139	−0.1245
Stacking	−0.0069	−0.0128	−0.0823	−0.1320
MFNN	−0.0659	−0.1227	−0.8807	−0.1715
(f) Average				
	Return	Annual return	Sharpe ratio	Max. Drawdown
AComNN	**0.3504**	**0.8256**	**3.8930**	**−0.0458**
CNNpred	0.0662	0.1500	0.8709	−0.1059
Stacking	0.0090	0.0182	0.0890	−0.1284
MFNN	0.0092	0.0243	0.1406	−0.1240

proving that the improvement of AComNN is statistically significant and the null hypothesis is rejected. However, for the rest of baselines, they have no statistical significance of difference with each other.

6.3. Trading simulation

To measure the profitability, we further implement a trading simulation on HSI and adopt a series of assessment metrics [22], including the Return, Annual Return, Sharpe Ratio, and Maximum Drawdown in the financial market to evaluate our proposed AComNN and other three baseline models. The Annual Return is calculated based on the Return (i.e., the trading profits during the simulation period). The Sharpe Ratio estimates the excess profits obtained by the investor for each additional unit of risk. And the Maximum Drawdown refers to the maximum of the drawdown over the history for evaluating the maximum possible loss during investment.

For the trading strategy, we set the initial investment capital to 1 and neglect the commission charge in Hong Kong Exchange with the action of buy and short selling so that to implement the day trading according to the increase or decrease signal provided by each investment advisory model for trading support. Specifically, when the predicted stock price rises, we fully invest in the index at the close price of the day t and sell all its shares at the close price of day $t+1$. On the other hand, when the predicted stock price falls down, we perform a short position opposite the process above to continue obtaining the profits.

Since randomly selecting one period to implement the trading simulation may cause unexpected evaluation bias, we conduct a trading simulation on each of the backtests and obtain their average result to further comparison. At each of the backtest, we reset the investment capital to 1 and follow the trading operation mention above to implement a separate trading simulation. Table 7 presents the performance of AComNN and other baseline models in the five trading simulations.

7. Discussion

The above experiments illustrate the predictive and profit capability of our proposed AComNN and the comparison with other baseline models.

For comparing the forecasting capability, our proposed AComNN outperforms the other three baselines because we consider the global major stock markets and economic entities for cross-regional feature extraction. With each of the stock markets opens and closes, their influence transfer from western hemisphere to the eastern hemisphere and finally affect the HSI trend in the next day. Besides, the other economic indicators worldwide also influence the financial market cyclically [26–28]. Thus, we mitigate the information insufficiency. Furthermore, our proposed AComNN is fine-designed for the above crucial time-series information transfer and refine, with a self-attention mechanism at the end of the model for dynamically allocating weights to each part of cross-regional features.

Nevertheless, for the other three baseline models, the MFNN adopts the historical data of the previous 120 time-steps to form each instance. For each sample, it only considers the features of Open, Close, High, Low, Volume, and Amount of a single stock index for the moving trend prediction, lacking the interaction with other macroeconomic indicators and technical indicators. As for the CNNpred, although it collects various features of multiple stock indices and even includes macro-economic indicators, its features are restricted in one region, lacking the assessment of the global market. Furthermore, it only adopts one kind of network; thus, it cannot understand the features based on multiple perspectives. Besides, both of the above models do not adopt effective data pre-processing, causing the difficulty in crucial information capture. For the last model, Stacking is a good approach to ensemble different models' classification ability. Based on the simultaneous prediction by multiple models, its prediction indeed is the stablest. However, the base learners are relatively too similar, which is a series of tree-based models and another series of variants of RNN, leading those base learners are unable to capture and discriminate heterogeneous patterns in the same task.

For the comparison of profitability, we adopt the trading simulation on each of the backtests and average their simulation results to comprehensively present their performance. Table 7 demonstrates the AComNN can obtain a high profit in each backtest and the average Annual Return can up to 82.56%, which is much higher than the profits of other baseline models. Besides, in each backtest, the AComNN also keep the lowest Max. Drawdown (−0.0458 on average) and highest Sharpe Ratio (3.8930 on average), proving that the AComNN that synthesizes cross-regional features has strong anti-risk ability compared with baselines. As for the other three baselines, although their prediction accuracy are not low, like the MFNN can obtain an forecasting accuracy of 56.20% on 01/-06/18, they still cannot get a high profit on average. This phenomenon reflects that the baselines are unable to provide the correct prediction on the days with large price fluctuation, causing their huge loss on those days and further leads low profit overall.

8. Threats to validity

Some of our re-implementation to the original baseline paper for fair comparison may cause threats to the validity, we list below:

For the MFNN, referring to its Section 3.1 in [22], it labels samples with −1, 0, and 1 as Decrease, No change (when the return fluctuation does not exceed a certain threshold) and Increase, in which they account for 10%, 80%, and 10% respectively as the best configuration. In order to avoid class imbalance problem, MFNN stochastically deletes instances in the class of No change until the number of samples in each category becomes the same. However, MFNN is a model proposed for predicting minute-level trading; thus, after dropping 70% of the whole data, it still has enough data for training. While in this experiment for day-level trading, after we go through the same process, the train set decreases to several hundred, causing the MFNN's performance to even worse. Thus to implement a fair comparison, we keep all the samples and only label Increase and Decrease to re-implement the MFNN as a binary classifier like AComNN, CNNpred, and the Stacking model. However, although we change the MFNN to a binary classifier, its highest accuracy reaches 57.60%, which also achieves and exceeds the best performance recorded in his paper (55.50%).

For the CNNpred, using five U.S. stock indices with U.S. macro-economic indicators for the U.S. stock market prediction, in order to change the prediction target to HSI for a fair comparison, we adopt SSE composite index, SZSE composite index, CSI 300, Hang Seng China Enterprises Index (HSCEI), and HSI to substitute the S&P 500, NASDAQ composite index, DJIA, NYSE composite index, and RUSSELL index that adopted in its paper. Additionally, we exploit the economic indicators in the China region to replace the U.S. economic indicators mentioned in its original paper. Besides, we adopt Avg. Acc. and Std. Dev. as the evaluation metrics instead of the F-measure which is adopted in the CNNpred paper. Because in our trading strategy, the prediction for increase and decrease are both important. We exploit the increase signals to buy stocks and decrease signals to implement the short position. However, in the CNNpred paper, it only make the decision by the increase signal in trading. For the prediction capability, the CNNpred performs almost the same in both our experiment and its own paper, but it does not show the same high profitability in our experiment, which may be caused by different dataset and the targeting stock market.

For the Stacking model, the original paper adopts some technical indicators and macro-economic indicators obtained from the U.S. market to implement the prediction for the several stock indices in the U.S. Similarly, we substitute the prediction target to HSI and adopt the economic indicators in the China region as its features to try to implement the fair comparison. The difference of the performance between our re-implemented model and the original model may due to the different prediction objectives and dataset.

9. Conclusion and future work

In this paper, we propose a novel Attention enhanced Compound Neural Network (AComNN) as the prediction engine for the financial trading system to fully exploit the time-series inter-relation of features across regions. Further, we explore its performance with four different forecasting windows and another four different smoothing factors. In addition, we verify its robustness and performance by five consecutive back-testing experiments under each window size and smoothing factor combination.

The experimental results demonstrate that our proposed AComNN can obtain an average accuracy of up to 60.81%. Compared with the state-of-the-art baselines, the proposed AComNN outperforms them on average over 13.36%, simultaneously with a very low standard deviation of 0.0355. Furthermore, we implement five trading simulations based on the five consecutive backtests. The AComNN also shows a more powerful profitability than the other baselines, which manifests that the AComNN based investment advisory system has high practicality and competitive performance in the real world.

For future work, we will extract some cross-regional sentimental features from social media as an additional information source to assist the financial prediction, and apply the methodology of this paper to explore more other stock indices. Thereby, the forecasting capability and profitability of our investment advisory system can be further strengthened in multiple financial markets.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported in part by the General Research Fund of the Research Grants Council of Hong Kong (No. 11208017) and the research funds of City University of Hong Kong (7005028, 7005217), and the Research Support Fund by Intel (9220097), and funding supports from other industry partners (9678149, 9440227, 9440180, 9220103 and 9229029).

Appendix

The Table A.1 describes the indicators we adopt in our experiment, including their names, descriptions, types and sources or calculation functions.

Table A.1
Description of used features.

#	Feature	Description	Type	Source
(a) Technical Indicators of HSI/S&P 500/FTSE 100				
1	Open	Open Price	Primitive	Yahoo Finance
2	High	High Price	Primitive	Yahoo Finance
3	Low	Low Price	Primitive	Yahoo Finance
4	Close	Close Price	Primitive	Yahoo Finance
5	Volume	Trading Volume	Primitive	Yahoo Finance
6	RSI	Relative Strength Index	Technical Indicator	TA-Lib
7	STOCHF %K	Stochastic Oscillator Fast %K	Technical Indicator	TA-Lib
8	STOCHF %D	Stochastic Oscillator Fast %D	Technical Indicator	TA-Lib
9	STOCH %D	Stochastic Oscillator Slow %D	Technical Indicator	TA-Lib
10	WILLR	Williams %R	Technical Indicator	TA-Lib
11	MACD	Moving Average Convergence/Divergence	Technical Indicator	TA-Lib
12	ROC_{Close}	Rate of Change for Close Price	Technical Indicator	TA-Lib
13	CCI	Commodity Channel Index	Technical Indicator	TA-Lib
14	OBV	On Balance Volume	Technical Indicator	TA-Lib
15	P{1, ..., 5}CCR	Previous n days Close Price Change rate	Technical Indicator	$\frac{Close_{today}}{Close_n} - 1$, $n = 1, \ldots, 5$
(b) Economic Indicators for China				
16	CHEPUI	Economic Policy Uncertainty Index for China	Academic Indicator	FRED
17	3MHIBOR	3-Month Hong Kong Interbank Offered Rate	Interest Rate Indicator	Tushare
18	GDPYGR	GDP Year-on-year Growth Rate	National Economy	Tushare
19	PIYGR	Primary Industry Year-on-year Growth Rate	National Economy	Tushare
20	SIYGR	Secondary Industry Year-on-year Growth Rate	National Economy	Tushare
21	TIYGR	Tertiary Industry Year-on-year Growth Rate	National Economy	Tushare
22	CPIYGR	CPI Year-on-year Growth Rate	Price Index	Tushare
23	CPIMGR	CPI Month-on-month Growth Rate	Price Index	Tushare
24	M0YGR	M0 Year-on-year Growth Rate	Currency Supply	Tushare
25	M0MGR	M0 Month-on-month Growth Rate	Currency Supply	Tushare
26	M1YGR	M1 Year-on-year Growth Rate	Currency Supply	Tushare
27	M1MGR	M1 Month-on-month Growth Rate	Currency Supply	Tushare
28	M2YGR	M2 Year-on-year Growth Rate	Currency Supply	Tushare
29	M2MGR	M2 Month-on-month Growth Rate	Currency Supply	Tushare
(c) Economic Indicators for the U.K.				
30	3MLIBOR	3-Month London Interbank Offered Rate	Interest Rate Indicator	FRED
31	GOLDAM	Gold Fixing Price 10:30 A.M. (London time) in London Bullion Market, based in U.S. Dollars	Commodity Indicator	FRED
32	GOLDPM	Gold fixing price 3:00 P.M. (London time) in London Bullion Market, based in U.S. Dollars	Commodity Indicator	FRED
33	UKEPUI	Economic Policy Uncertainty Index for the U.K.	Academic Indicator	FRED
(d) Economic Indicators for the U.S.				
34	DGS10	10-year Treasury Constant Maturity Rate	Interest Rate Indicator	FRED
35	DGS1	1-year Treasury Constant Maturity Rate	Interest Rate Indicator	FRED
36	EFFR	Effective Federal Funds Rate	Interest Rate Indicator	FRED
37	DAAA	Moody's Seasoned Aaa Corporate Bond Yield	Interest Rate Indicator	FRED
38	DBAA	Moody's Seasoned Baa Corporate Bond Yield	Interest Rate Indicator	FRED
39	TEDRATE	TED Spread	Interest Rate Indicator	FRED
40	T10YIE	10-year Breakeven Inflation Rate	Interest Rate Indicator	FRED
41	T5YIFR	5-Year, 5-Year Forward Inflation Expectation Rate	Interest Rate Indicator	FRED
42	DTWEXBGS	Trade Weighted U.S. Dollar Index	Exchange Rate	FRED
43	DEXUSEU	U.S./Euro Foreign Exchange Rate	Exchange Rate	FRED
44	DEXUSCH	U.S./China Foreign Exchange Rate	Exchange Rate	FRED
45	DEXUSJP	U.S./Japan Foreign Exchange Rate	Exchange Rate	FRED
46	DCOILWTICO	Crude Oil Prices: West Texas Intermediate (WTI)	Commodity Indicator	FRED
47	VIXCLS	CBOE Volatility Index: VIX	Financial Indicator	FRED
48	USEPUI	Economic Policy Uncertainty Index for the U.S.	Academic Indicator	FRED

Yahoo finance (https://finance.yahoo.com/) is a comprehensive financial website.
Ta-Lib (https://mrjbq7.github.io/ta-lib/) is a tool for performing technical analysis of financial market data.
Tushare (https://tushare.pro/) is an open source community for financial market data.
FRED (https://fred.stlouisfed.org/) is a financial database maintained by the Federal Reserve Bank of St. Louis.

References

[1] Y.S. Abu-Mostafa, A.F. Atiya, Introduction to financial forecasting, Appl. Intell. 6 (3) (1996) 205–213.

[2] M. Kraus, S. Feuerriegel, Decision support from financial disclosures with deep neural networks and transfer learning, Decis. Support Syst. 104 (2017) 38–48.

[3] R.C. Brasileiro, V.L. Souza, A.L. Oliveira, Automatic trading method based on piecewise aggregate approximation and multi-swarm of improved self-adaptive particle swarm optimization with validation, Decis. Support Syst. 104 (2017) 79–91.

[4] C. Kocak, ARMA (P, q) type high order fuzzy time series forecast method based on fuzzy logic relations, Appl. Soft Comput. 58 (2017) 92–103.

[5] A.A. Adebiyi, A.O. Adewumi, C.K. Ayo, Comparison of ARIMA and artificial neural networks models for stock price prediction, J. Appl. Math. 2014 (2014).

[6] A.A. Ariyo, A.O. Adewumi, C.K. Ayo, Stock price prediction using the ARIMA model, in: 2014 UKSim-AMSS 16th International Conference on Computer

Modelling and Simulation, IEEE, 2014, pp. 106–112.

[7] J.E. Jarrett, E. Kyper, ARIMA Modeling with intervention to forecast and analyze chinese stock prices, Int. J. Eng. Bus. Manag. 3 (3) (2011) 53–58.

[8] Z. Lin, Modelling and forecasting the stock market volatility of SSE composite index using GARCH models, Future Gener. Comput. Syst. 79 (2018) 960–972.

[9] Y. Lin, H. Guo, J. Hu, An SVM-based approach for stock market trend prediction, in: The 2013 International Joint Conference on Neural Networks (IJCNN), IEEE, 2013, pp. 1–7.

[10] L. Luo, S. You, Y. Xu, H. Peng, Improving the integration of piece wise linear representation and weighted support vector machine for stock trading signal prediction, Appl. Soft Comput. 56 (2017) 199–216.

[11] B.M. Henrique, V.A. Sobreiro, H. Kimura, Stock price prediction using support vector regression on daily and up to the minute prices, J. Financ. Data Sci. 4 (3) (2018) 183–201, http://dx.doi.org/10.1016/j.jfds.2018.04.003, URL http://www.sciencedirect.com/science/article/pii/S2405918818300060.

[12] A. Dutta, G. Bandopadhyay, S. Sengupta, Prediction of stock performance in indian stock market using logistic regression, Int. J. Bus. Inf. 7 (1) (2012).

[13] K.O. Nti, A. Adekoya, B. Weyori, Random forest based feature selection of macroeconomic variables for stock market prediction, Am. J. Appl. Sci. 16 (7) (2019) 200–212.

[14] S. Basak, S. Kar, S. Saha, L. Khaidem, S.R. Dey, Predicting the direction of stock market prices using tree-based classifiers, North Amer. J. Econ. Financ. 47 (2019) 552–567.

[15] B. Nair, V. Mohandas, N.R. Sakthivel, A decision tree- rough set hybrid system for stock market trend prediction, Int. J. Comput. Appl. 6 (9) (2010) 1–6.

[16] M. Jiang, J. Liu, L. Zhang, C. Liu, An improved stacking framework for stock index prediction by leveraging tree-based ensemble models and deep learning algorithms, Physica A 541 (2020) 122272, http://dx.doi.org/10.1016/j.physa.2019.122272, URL http://www.sciencedirect.com/science/article/pii/S0378437119313093.

[17] H. Wang, Y. Jiang, H. Wang, Stock return prediction based on bagging-decision tree, in: 2009 IEEE International Conference on Grey Systems and Intelligent Services (GSIS 2009), 2009, pp. 1575–1580.

[18] D.Q. Nelson, A.C.M. Pereira, R.A. de Oliveira, Stock market's price movement prediction with LSTM neural networks, in: 2017 International Joint Conference on Neural Networks (IJCNN), 2017, pp. 1419–1426.

[19] E. Hoseinzade, S. Haratizadeh, Cnnpred: CNN-based stock market prediction using a diverse set of variables, Expert Syst. Appl. 129 (2019) 273–285, http://dx.doi.org/10.1016/j.eswa.2019.03.029, URL http://www.sciencedirect.com/science/article/pii/S0957417419301915.

[20] O.B. Sezer, M. Ozbayoglu, E. Dogdu, A deep neural-network based stock trading system based on evolutionary optimized technical analysis parameters, Procedia Comput. Sci. 114 (2017) 473–480, http://dx.doi.org/10.1016/j.procs.2017.09.031, URL http://www.sciencedirect.com/science/article/pii/S1877050917318252 Complex Adaptive Systems Conference with Theme: Engineering Cyber Physical Systems, CAS October 30 – November 1, 2017, Chicago, Illinois, USA.

[21] F. Feng, X. He, X. Wang, C. Luo, Y. Liu, T.-S. Chua, Temporal relational ranking for stock prediction, ACM Trans. Inf. Syst. 37 (2) (2019) http://dx.doi.org/10.1145/3309547.

[22] W. Long, Z. Lu, L. Cui, Deep learning-based feature engineering for stock price movement prediction, Knowl.-Based Syst. 164 (2019) 163–173, http://dx.doi.org/10.1016/j.knosys.2018.10.034, URL http://www.sciencedirect.com/science/article/pii/S0950705118305264.

[23] J. Eapen, D. Bein, A. Verma, Novel deep learning model with CNN and bi-directional LSTM for improved stock market index prediction, in: 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), 2019, pp. 0264–0270.

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

[25] R. Zhao, Y. Deng, M. Dredze, A. Verma, D. Rosenberg, A. Stent, Visual attention model for cross-sectional stock return prediction and end-to-end multimodal market representation learning, 2019, arXiv:1809.03684.

[26] A. Garefalakis, A. Dimitras, D. Koemtzopoulos, K. Spinthiropoulos, Determinant factors of Hong Kong stock market, Int. Res. J. Financ. Econ. 62 (2011) 50–60.

[27] M. Valukonis, China's stock market trends and their determinants analysis using market indices., Econ. Manag. 18 (4) (2013) 651–660, URL http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=95278267&site=ehost-live.

[28] B. Weng, W. Martinez, Y.-T. Tsai, C. Li, L. Lu, J.R. Barth, F.M. Megahed, Macroeconomic indicators alone can predict the monthly closing price of major U.S. indices: Insights from artificial intelligence, time-series analysis and hybrid models, Appl. Soft Comput. 71 (2018) 685–697, http://dx.doi.org/10.1016/j.asoc.2018.07.024, URL http://www.sciencedirect.com/science/article/pii/S1568494618304125.

[29] Y. Kara, M. Acar Boyacioglu, 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 Syst. Appl. 38 (5) (2011) 5311–5319, http://dx.doi.org/10.1016/j.eswa.2010.10.027, URL http://www.sciencedirect.com/science/article/pii/S0957417410011711.

[30] M. Tanaka-Yamawaki, S. Tokuoka, Adaptive use of technical indicators for the prediction of intra-day stock prices, Physica A 383 (1) (2007) 125–133, http://dx.doi.org/10.1016/j.physa.2007.04.126, URL http://www.sciencedirect.com/science/article/pii/S0378437107005067 Econophysics Colloquium 2006 and Third Bonzenfreies Colloquium.

[31] Y. Wang, Stock price direction prediction by directly using prices data: an empirical study on the KOSPI and HSI, 2013, arxiv e-prints arxiv:1309.7119 arxiv:1309.7119.

[32] G. Zumbach, L. Fernández, Option pricing with realistic ARCH processes, Quant. Finance 14 (1) (2014) 143–170.

[33] N. Ülkü, D. Kuruppuarachchi, O. Kuzmicheva, Stock market's response to real output shocks in eastern European frontier markets: A varwal model, Emerg. Mark. Rev 33 (2017) 140–154.

[34] F. Rundo, Deep LSTM with reinforcement learning layer for financial trend prediction in FX high frequency trading systems, Appl. Sci. 9 (20) (2019) http://dx.doi.org/10.3390/app9204460, URL https://www.mdpi.com/2076-3417/9/20/4460.

[35] Wikipedia, List of stock exchange trading hours - wikipedia, 2020, https://en.wikipedia.org/wiki/List_of_stock_exchange_trading_hours (Accessed on 10/17/2020).

[36] W.S. Sarle, Neural networks and statistical models, 1994.

[37] S. Hochreiter, The vanishing gradient problem during learning recurrent neural nets and problem solutions, Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 06 (02) (1998) 107–116, http://dx.doi.org/10.1142/S0218488598000094, arXiv:https://doi.org/10.1142/S0218488598000094.

[38] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Trans. Neural Netw. 5 (2) (1994) 157–166.

[39] S. Santurkar, D. Tsipras, A. Ilyas, A. Madry, How does batch normalization help optimization?, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 31, Curran Associates, Inc., 2018, pp. 2483–2493, URL http://papers.nips.cc/paper/7515-how-does-batch-normalization-help-optimization.pdf.

[40] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780, http://dx.doi.org/10.1162/neco.1997.9.8.1735, arXiv:https://doi.org/10.1162/neco.1997.9.8.1735.

[41] A. Graves, J. Schmidhuber, Framewise phoneme classification with bidirectional LSTM and other neural network architectures, Neural Netw. 18 (5) (2005) 602–610, http://dx.doi.org/10.1016/j.neunet.2005.06.042, URL http://www.sciencedirect.com/science/article/pii/S0893608005001206 IJCNN 2005.

[42] A. Graves, A. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, in: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 6645–6649.

[43] M.-T. Luong, H. Pham, C.D. Manning, Effective approaches to attention-based neural machine translation, 2015, arXiv:1508.04025.

[44] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L.u. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 30, Curran Associates, Inc., 2017, pp. 5998–6008, URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

[45] P. Zhou, W. Shi, J. Tian, Z. Qi, B. Li, H. Hao, B. Xu, Attention-based bidirectional long short-term memory networks for relation classification, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Association for Computational Linguistics, 2016, pp. 207–212.

[46] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, Y. Bengio, Show, attend and tell: Neural image caption generation with visual attention, in: F. Bach, D. Blei (Eds.), in: Proceedings of Machine Learning Research, vol. 37, PMLR, Lille, France, 2015, pp. 2048–2057, URL http://proceedings.mlr.press/v37/xuc15.html.

[47] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2017, arXiv:1412.6980.

[48] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: F. Pereira, C.J.C. Burges, L. Bottou, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems, Vol. 25, Curran Associates, Inc., 2012, pp. 1097–1105, URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[49] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, 15 (1), 2014, pp. 1929–1958.

[50] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R.R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, 2012, arXiv:1207.0580.

[51] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, The MIT Press, 2016.