Expert Systems With Applications 113 (2018) 457–480

Contents lists available at ScienceDirect

Expert Systems With Applications

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

ModAugNet: A new forecasting framework for stock market index value with an overfitting prevention LSTM module and a prediction LSTM module

Yujin Baek[a], Ha Young Kim[a,b,*]

[a] Department of Financial Engineering, Ajou University, Worldcupro 206, Yeongtong-gu, Suwon 16499, Republic of Korea
[b] Department of Data Science, Ajou University, Worldcupro 206, Yeongtong-gu, Suwon 16499, Republic of Korea

ARTICLE INFO

Article history:
Received 11 March 2018
Revised 6 July 2018
Accepted 7 July 2018
Available online 9 July 2018

Keywords:
Long short-term memory
Data augmentation
Overfitting
Deep learning
Stock market index

ABSTRACT

Forecasting a financial asset's price is important as one can lower the risk of investment decision- making with accurate forecasts. Recently, the deep neural network is popularly applied in this area of research; however, it is prone to overfitting owing to limited availability of data points for training. We propose a novel data augmentation approach for stock market index forecasting through our ModAugNet framework, which consists of two modules: an overfitting prevention LSTM module and a prediction LSTM module.

The performance of the proposed model is evaluated using two different representative stock market data (S&P500 and Korea Composite Stock Price Index 200 (KOSPI200)). The results confirm the excellent forecasting accuracy of the proposed model. ModAugNet-c yields a lower test error than the comparative model (SingleNet) in which an overfitting prevention LSTM module is absent. The test mean squared error (MSE), mean absolute percentage error (MAPE), and mean absolute error (MAE) for S&P500 decreased to 54.1%, 35.5%, and 32.7%, respectively, of the corresponding S&P500 forecasting errors of SingleNet, while the same for KOSPI200 decreased to 48%, 23.9%, and 32.7%, respectively, of the corresponding KOSPI200 forecasting errors of SingleNet. Furthermore, through the analyses of the trained ModAugNet-c, we found that test performance is entirely dependent on the prediction LSTM module.

The contribution of this study is its applicability in various instances where it is challenging to artificially augment data, such as medical data analysis and financial time-series modeling.

1. Introduction

Forecasting the future price of a financial asset is essential for investors as they can reduce the risk of decision-making by appropriately determining the future movement of an investment asset. It is challenging to know when and how to allocate budgets and thus, technical and quantitative methods have been utilized by investors as an attempt to forecast asset price movement. These approaches consist of discovering a suitable pattern from market data and capturing the optimal moment to make investment decisions.

There has been decades-long controversy surrounding stock market predictability. The earliest study on stock behavior was performed by Bachelier (1900). This work was the first to characterize stock price movement in the random walk manner. Cootner (1964) and Fama (1965) empirically tested the random walk characteristics of price changes. Malkiel and Fama (1970) carried out research based on the efficient market hypothesis. According to the theory, all new information is immediately reflected to the asset price without delay and thus, future asset price movement is independent of past and present information. From their perspective, it is considered impossible to forecast future asset price.

On the other hand, numerous studies have attempted to experimentally disprove efficient market hypothesis and empirical evidences have shown that stock markets are, to some extent, predictable. Traditional approaches for time series prediction use parametric statistical models, such as autoregressive moving average (ARMA), autoregressive integrated moving average (ARIMA), and vector autoregression, to find the best estimates (Box, Jenkins, Reinsel, & Ljung, 2015). Virtanen and Yli-Olli (1987) estimated Finnish stock market index using six explanatory variables, including lagged index and macroeconomic factors through ARIMA-based econometric modeling. Ariyo, Adewumi, and Ayo (2014) proposed

* Corresponding author at: Department of Financial Engineering, Ajou University, Worldcupro 206, Yeongtong-gu, Suwon 16499, Republic of Korea.
E-mail addresses: heyyujin@ajou.ac.kr (Y. Baek), hayoungkim@ajou.ac.kr (H.Y. Kim).

https://doi.org/10.1016/j.eswa.2018.07.019
0957-4174/

ARIMA-based stock price predictive system, which was tested with listed stocks on New York Stock Exchange and Nigeria Stock Exchange. They concluded that the ARIMA model has a particular potential in short-term prediction. Although these econometric models are convenient for describing and evaluating the relationships between variables by statistical inference, they have some limitations for financial time-series analysis. First, they are not able to capture the nonlinear nature of stock prices because they presume a linear form of model structure. In addition, they are assumed to have constant variance, while financial time-series is very noisy and has time-varying volatility (Abu-Mostafa & Atiya, 1996).

There have been many attempts to model nonlinear relationships in financial time-series using machine learning technologies. Artificial neural networks (ANNs) and kernelized support vector machine have been extensively applied due to their capabilities in nonlinear mapping and generalization. Unlike econometric models, ANNs do not have a rigid model structure and a set of assumptions imposed. Given sufficient data, it can be modeled. This statement is supported by the universal approximation theorem, which states that any finite and continuous function could be approximated (Hornik, Stinchcombe, & White, 1989). Wang and Leu (1996) proposed a fusion of ANN and econometric model. Input features were extracted from the ARIMA model and used to train mid-term Taiwanese stock market trend prediction system based on the recurrent neural network (RNN). They showed that using the differenced features extracted from ARIMA (1, 2, 1) produced more accurate predictions than using data without differencing. Hajizadeh, Seifi, Zarandi, and Turksen (2012) proposed two hybrid models to forecast the volatility of S&P500 index return, integrating an ANN with the exponential generalized autoregressive conditional heteroscedasticity model. Their computational results confirmed that both the hybrid models outperformed the single econometric model by yielding lower test errors. Kristjanpoller, Fadic, and Minutolo (2014) proposed the volatility forecasting model in Latin American markets with the fusion of generalized autoregressive conditional heteroscedasticity (GARCH) model and ANNs, and showed that the proposed model outperformed the GARCH model in terms of mean absolute percentage error. Lendasse, de Bodt, Wertz, and Verleysen (2000) proposed a forecasting system for Bel 20 stock market index based on radial basis function neural network. They showed that their system could capture nonlinear relationships in financial time-series data. To forecast the Indian stock market, Patel, Shah, Thakkar, and Kotecha (2015) suggested the hybrid model of ANN, random forest, and support vector regression (SVR). The ANN-SVR combined model resulted in the best overall prediction performance. Wang, Wang, Zhang, and Guo (2012) proposed a hybrid system to predict stock index combining exponential smoothing, ARIMA, and back propagation neural network. Their system was tested on the opening of the Dow Jones Industrial Average Index and the closing of the Shenzhen Integrated Index. Empirical results show that the hybrid system outperformed all individual systems providing more accurate forecasts. Adhikari (2015) used an ensemble method which combines individual forecasts, including ARIMA and ANNs. It is observed that the proposed hybrid model outperformed individual forecasting models. Rather, Agarwal, and Sastry (2015) incorporated RNN in stock returns' prediction system with two linear models, ARMA and exponential smoothing. They found that the improvement of the prediction performance is mainly contributed by RNN.

In addition, machine learning based models have been trained for the purpose of developing trading systems as in the following references; it has been achieved by either forecasting the trading signal, such as expected returns with the regressor or forecasting the movement with the classifier (Atsalakis & Valavanis, 2009; Chen, Leung, & Daouk, 2003; Chiang, Enke, Wu, & Wang, 2016; Kim & Enke, 2016; Vanstone & Finnie, 2009). Chen et al. (2003) used the probabilistic neural network (PNN) to predict Taiwanese market index return. PNN showed enhanced predictive power than the two parametric benchmark models (generalized methods of moments with Kalman filter and random walk forecasting model) and PNN-guided trading strategies obtained higher profits than the other parametric strategies. Vanstone and Finnie (2009) introduced detailed methods to create the stock market trading systems under real-world constraints, such as rules to enter and exit trades, risk control, and money management. They used both the fundamental variables and technical variables to output the trading signal, which captures not only the expected return movement, but also its strength. Chiang et al. (2016) built up the trading decision support system by combining the particle swarm optimization and ANN. The output value from the system indicates the stock index direction based on which trading suggestion to buy or hold the asset is provided. The profit simulation results showed that the proposed system yielded higher returns than the buy-and-hold strategy.

In recent studies, long short-term memory (LSTM) network, which is appropriately structured to learn temporal patterns, is extensively utilized for various tasks of time-series analyses (Lipton, Kale, Elkan, & Wetzel, 2015; Sak, Senior, & Beaufays, 2014; Sundermeyer, Schlüter, & Ney, 2012). LSTM is advantageous over the conventional RNN as it overcomes the problem of vanishing (or exploding) gradients and as it can effectively learn long-term dependencies through memory cells and gates. Thus, many studies on financial time-series modeling are conducted using LSTMs. In previous studies based on ANNs, technical indicators were computed to capture temporal patterns and fed into ANNs as input features (Göçken, Özçalıcı, Boru, & Dosdoğru, 2016; Kim & Han, 2000; Lendasse et al., 2000; Patel et al., 2015; Ticknor, 2013). However, LSTMs do not require hand-crafted features, such as technical indicators. If data are given, suitable data-dependent patterns are automatically detected within data (LeCun, Bengio, & Hinton, 2015). Chen, Zhou, and Dai (2015) proposed an LSTM-based system to predict stock returns and tested it on the Chinese stock market. They used historical price data of the stock and market indexes for sequential learning. Numerical results confirmed a promising predictive power of LSTMs, which result in an improvement of forecasting accuracy. Nelson, Pereira, and de Oliveira (2017) developed an LSTM-based stock price prediction method by which the trading simulation was executed, and its performance has been evaluated in comparison to the baselines, which are multi-layer perceptron, random forest, and a pseudo-random model. Experimental results show that the LSTM-based model provides comparatively low risks in trading strategies. Bao, Yue, and Rao (2017) used LSTM for stock price forecasting and the LSTM-based model showed better performance in terms of predictive accuracy. In addition, different types of sequential data could be fed into networks to enlarge the available information set. Li, Bu, and Wu (2017) extracted investor sentiment from forum posts and fed it into a network with historical market data to predict CSI300 and sentiment. In their experiments, LSTMs outperformed benchmark models of support vector machine and adding sentiment features lead to remarkable improvements in accuracy (from 78.57% to 87.86%), when predicting the next day's open price. Akita, Yoshihara, Matsubara, and Uehara (2016) used textual data from the Nikkei newspaper as input of LSTMs, together with numerical market time-series data to predict 10 individual companies' open price. The trading strategies based on forecasts are simulated. The model trained with the numerical and textual representations extracted by their proposed method helped in making higher profits (1.67 times higher) than the model otherwise trained with only numerical data.

For financial time series analyses using deep neural networks (DNNs), we should be concerned about the problem of overfit-

ting as data are not sufficient (Lee, 2009; Lendasse et al., 2000). In a year, the number of data points that we could collect on a daily basis was only approximately 252. DNNs are promising approaches with enhanced representation power as they learn highly complex nonlinear relationships between variables (LeCun et al., 2015) However, they are prone to the overfitting problem, particularly when the data are insufficient in comparison to the number of model parameters (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). Overfitting deteriorates prediction accuracy, thus regularization techniques, such as $L1$ and $L2$ regularization, dropout, data augmentation, early stopping, reducing network size and reducing learning rate were investigated to avoid an overfitting problem (Goodfellow, Bengio, Courville, & Bengio, 2016; Nowlan & Hinton, 1992; Srivastava et al., 2014). It is easy and common to implement dropout, $L1$ and $L2$ regularization. An early stopping method is used to check the learning curve and stop training before overfitting occurs. This method can prevent overfitting, but it cannot improve generalization performance. Reducing the neural network size can also prevent overfitting, but it does not improve generalization performance because larger and deeper networks can solve more complicated problems. It is necessary to have sufficient data to leverage a deeper and larger network; hence, data augmentation is a typical method to prevent overfitting, while improving generalization accuracy. However, data augmentation for financial time-series is not a simple task, unlike image data augmentation, which can be easily achieved by various transformation techniques (Chatfield, Simonyan, Vedaldi, & Zisserman, 2014; Krizhevsky, Sutskever, & Hinton, 2012; Salamon & Bello, 2017; Perez & Wang, 2017; Zeiler & Fergus, 2014). Transformation-based data augmentation approaches result in generating synthetic data by distorting the original data, which are not desirable for financial time-series. Therefore, data augmentation for financial time-series has received limited academic attention despite its importance in improving the deep learning model's performance and robustness.

Instead, various data representation methods were applied in related studies to avoid the overfitting problem that enlarged input variables could provoke. A variety of input variables could potentially enhance the representation power of DNNs; however, it does not necessarily guarantee the improvement of a model's generalization performance. Sufficient data are in need because the number of model parameters increases as we enlarge the number of input features. To solve this problem, dimension reduction techniques of the principal component analysis (PCA) and autoencoder are mainly used (Bao et al., 2017; Chong, Han, & Park, 2017; Wang & Wang, 2015; Zhong & Enke, 2017). Wang and Wang (2015) proposed the model which combined PCA and a stochastic time effective function neural network (PCA-STNN). They revealed that the PCA-STNN resulted in lower forecast errors than its baselines. Zhong and Enke (2017) proposed the daily stock market return forecasting model using ANNs alongside PCA, fuzzy robust PCA, and kernel-based PCA. From 60 financial and economic factors, data-preprocessing was undertaken with three PCA methods to choose the reduced amount of input variables and the ANN classifier was trained to output the next day's price direction. Trading simulation results showed that the ANNs combined with PCA yielded higher returns than the buy-and-hold strategy. Bao et al. (2017) reduced the input dimensions of LSTM-based DNN regressor from 19 to 25 input variables by applying the stacked autoencoders. They also used a denoising approach of wavelet transformation as data-preprocessing. This network is named WSAEs-LSTM in the paper and the results showed that WSAEs-LSTM outperformed their baseline methods: RNN, LSTM, and WLSTM. However, an important advantage of deep learning is that it learns features from the input data itself. If input data is dimensionally reduced and used as input in neural networks, information loss can occur (Kam & Kim, 2017).

In this paper, we suggest a data augmentation approach to achieve robustness against overfitting in forecasting the stock market index without the need to generate artificial training data. The research is conducted based on our novel forecasting framework for stock market index with a modular LSTM network composed of two modules; an Overfitting Prevention LSTM Module and a Prediction LSTM Module. For convenience, we call this proposed network as ModAugNet. In addition, an overfitting Prevention LSTM Module is referred to as Prevention Module and a Prediction LSTM Module is referred to as Prediction Module. In ModAugNet, the Prediction Module takes only target index as input, while the Prevention Module takes different combinations of relevant stocks each time. As a different instance is created by taking a combination and fed into the Prevention Module each time, our proposed network could be trained by an increased amount of real data points. Based on a unified network of jointly trained Prediction Module and Prevention Module, final prediction is calculated. To investigate whether our framework performs well on different indices, empirical tests were conducted on two representative real-world stock market indices: S&P500 and the Korea Composite Stock Price Index 200 (KOSPI200). KOSPI200 represents the benchmark indicator of the Korean capital market and its derivatives' market is known as top-tier derivatives market globally in terms of trading volume (Ryu, 2015), while S&P500 is globally an important benchmark indicator, which is widely regarded as bellwether for the U.S. economy. The performance of ModAugNet is compared to the comparative model (SingleNet) in which the Prevention Module is absent to examine the effectiveness of the Prevention Module in our proposed network. Furthermore, we analyze the role of the Prevention Module based on the test results.

The remainder of this paper is organized as follows. Section 2 describes the LSTMs and the data specifications, while Section 3 explains the workflow of our proposed forecasting framework for the stock market index. Section 4 reveals the experimental results with detailed analyses and Section 5 provides the further discussions. Section 6 concludes and suggests future work.

2. Methodology and materials

2.1. Long short-term memory

The applications of RNNs make it possible to learn temporal patterns from sequential data. The RNNs' memory properties are not present in the previous deep feedforward networks. However, the vanishing gradient issue occurs when practically training standard RNNs, which means that the networks have difficulties in retaining information over a long period of time (Hochreiter, 1998). The LSTM unit was introduced as an alternative method to learn sequential patterns (Hochreiter & Schmidhuber, 1997). LSTMs have gating mechanisms to regulate the flow of information. The amount of incoming information that will be retained is systemically determined at each time step and thus, LSTMs could memorize temporal patterns over a longer time-span.

Fig. 1 represents the structure of an LSTM memory block consisting of memory cell and gates. x_t and h_t correspond to the input and hidden state respectively, at time t, and f_t, i_t, and o_t are the gates which are called forget, input, and output gates, respectively. n_t is the candidate of input to be stored, and the storing amount is later controlled by an input gate. The calculations for each gate, input candidate, cell state, and hidden state are performed using the following formulas:

$$f_t = \sigma\left(A_f x_t + B_f h_{t-1} + b_f\right), \tag{1}$$

$$i_t = \sigma\left(A_i x_t + B_i h_{t-1} + b_i\right), \tag{2}$$

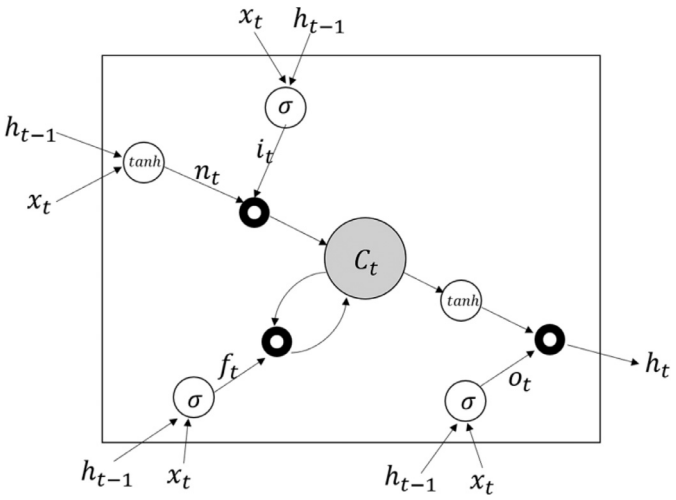

Fig. 1. Structure of an LSTM memory block.

$$o_t = \sigma(A_o x_t + B_o h_{t-1} + b_o), \tag{3}$$

$$n_t = tanh(A_n x_t + B_n h_{t-1} + b_n), \tag{4}$$

$$c_t = c_{t-1} * f_t + n_t * i_t, \tag{5}$$

$$h_t = tanh(c_t) * o_t, \tag{6}$$

where A and B are weight matrices and b are bias vectors. The symbol of $*$ indicates element-wise multiplication. Tanh activation is used for updating the hidden state. Sigmoid activation for gating is marked as σ in Fig. 1 and computed by the sigmoid function $\sigma(z)=(1+e^{-z})^{-1}$, where the output is in the range of [0,1]. If the output value is 0, no information can enter. If the output value increases, more information is allowed, and if it becomes 1, information can be fully reflected. Due to this selective reflection of information, the LSTMs are capable of learning longer temporal patterns.

2.2. Dataset preparation and preprocessing

This study has exploited historical closing price data of two stock market indices (KOSPI200 and S&P500) and 10 stocks from each index from January 4, 2000 to July 27, 2017 on a working day basis. The retrieved companies are listed on the exchange before 2000 and are currently ranked within the top 25 market capitalizations; the components of KOSPI200 are listed on the Korean exchange, while those of S&P500 are listed on the New York Stock Exchange or the NASDAQ stock market. Their prices are all highly correlated to the corresponding stock market index, with the Pearson's correlation coefficient greater than 0.8 (under significance level of 5% and all p-values less than 0.05) and thus, we expected them to be helpful in improving the model performance. The data are all obtainable from Yahoo! Finance.

The challenge is from the fact that the retrieved features differ in magnitude. The stock price of a company is orders of magnitude larger than that of others. During the training phase, the weight adjustment of the network would be overwhelmed by the larger magnitude of input, even if the error comes from the others (Duda, Hart, & Stork, 2012). In order to ensure that larger inputs do not become dominant, we have normalized the original dataset

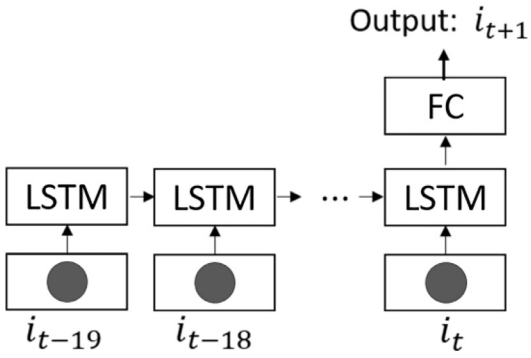

Fig. 2. Architecture of the comparative model (called SingleNet).

by using the following Eq. (7) and obtained the normalized value $\bar{X}$ at each data point:

$$\bar{X} = \frac{X - X_{min}}{X_{max} - X_{min}}, \tag{7}$$

where X is original value at a data point, X_{max} and X_{min} are maximum and minimum value of a specific feature, respectively. Consequently, every feature change could be captured in the range of [0,1].

In this study, price information for the S&P500 between January 4, 2000 and December 31, 2007 (1988 data points) is used as the training dataset, while price information between January 2, 2008 and July 27, 2017 (2,388 data points) is used as the test dataset. For KOSPI200, the same training and test periods are chosen as the S&P500, but the number of data points differs; 1,946 data points for training and 2,354 data points for test. 20% was separated from the training dataset for the purposes of validation.

3. Experiment design

3.1. Model architecture

The movement of the financial time-series comes from the movement of relevant explanatory variables, including the time-series of the response variable itself. In most studies of forecasting financial time-series, previous time-series of target variables are regarded as an essential explanatory variable and used for modeling. The autoregressive time-series of target variables were mainly used as input variables in statistical modeling (Adhikari, 2015; Ariyo et al., 2014; Virtanen & Yli-Olli, 1987; Wang & Leu, 1996; Wang et al., 2012), and the various ANN-based prediction systems are trained with technical indicators, which are computed based on previous values of the target variables (Göçken et al., 2016; Kim & Han, 2000; Lendasse et al., 2000; Patel et al., 2015; Ticknor, 2013).

For stock market index prediction, as target index time-series is a direct and important source of its movement, it can be modeled as shown in Fig. 2. However, the number of data points for training is less than 2,000, which is not sufficient. Training DNNs with limited data may deteriorate the prediction accuracy, resulting in overfitting. In this paper, we overcome the problem of insufficient data points by introducing a modular network called ModAugNet. As shown in Fig. 3, the Prevention Module is attached to take different input features each time, while the Prediction Module always takes the index as input. The Prevention Module is separated to take different input features because we do not expect our network to be overwhelmed by indirect information, which has five times more dimensions than the direct information in this experiment. The dimension of our indirect information (stock prices) is five times greater than the direct information (index); however, di-

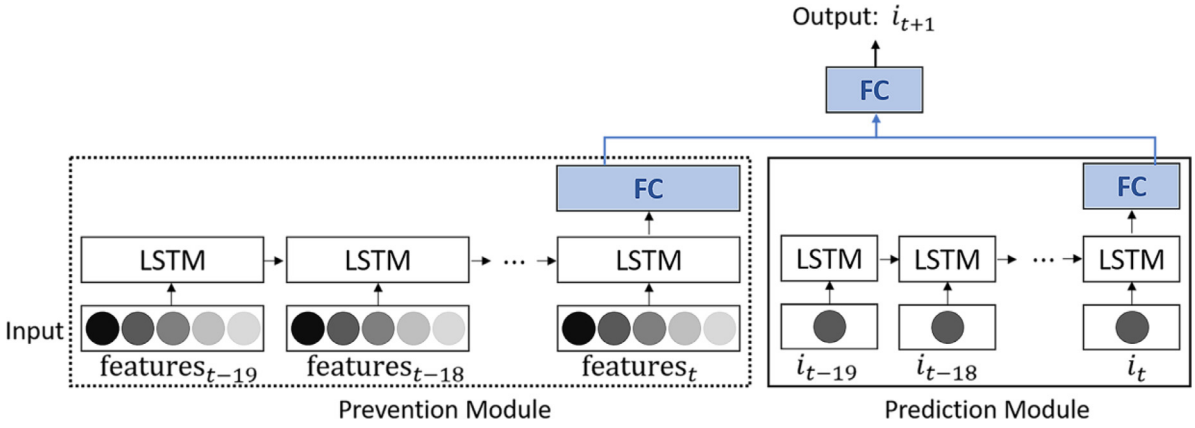

Fig. 3. Architecture of the proposed ModAugNet.

rect information is the most important variable. If we feed six variables without splitting them into two modules, neural networks treat all the input variables equally, which makes it challenging to extract the appropriate features. Thus, we designed two modules, each of which extracts the features from direct and indirect information.

Each module takes the sequence data whose window length is 20. For the Prediction Module, the daily closing price of the stock market index for 20 consecutive trading days, from time $(t-19)$ to time t, is fed into the input nodes when predicting the stock market index at time $(t+1)$. This window is slid along the dataset. When training the network, the 20 first data are fed into the network and the 21st is given as the target value for supervised learning. The parameters of the network are learned to minimize the error between the predicted and target value. Further, moving on one datum, the subsequent 20 data are fed into the network and the 22nd is given as the target value and so on. Subsequent to the completion of the training, we feed the 20 first test data into the trained network. Based on the trained weights, predicted value for the next day's close price is calculated. In Figs. 2 and 3, the daily closing price of stock market index at time t is denoted as i_t. The Prevention Module receives a set of five stocks' closing price from time $(t-19)$ to time t.

In each module, LSTM layers are followed by a fully-connected layer (marked as FC in Figs. 2, and 3). After passing the LSTM layer and consecutive fully-connected layer, the output values from the two modules are concatenated and taken as input for the next fully-connected layer. Finally, the output prediction is calculated. The number of output units in each layer is chosen by trial and error. In our experiment to build the forecasting model for S&P500 (KOSPI200), we chose the number of LSTM output units as 5 (5) and the number of neurons of the corresponding fully-connected layer as 3 (2) in the Prevention Module. For the Prediction Module, we chose the number of LSTM output units as 4 (4) and the number of neurons of the corresponding fully-connected layer in the module as 2 (2). Both modules are concatenated as the output values from the fully-connected layer of each module are fed into the next fully-connected layer that has 2 (3) neurons. Note that the numbers in parentheses indicate the corresponding numbers of output units for predicting KOSPI200. Fully connected layer's neurons in both the Prevention Module and Prediction Module represent the features extracted from the given input training data. Therefore, the number of neurons in each layer indicates the number of features to extract. The advantage of DNNs is that they perform two tasks simultaneously: feature extraction and prediction. Thus, in our network, from one input variable (stock market index) fed into the Prediction Module the features will be learned, and from five input variables (5 stocks) fed into the Prevention Module the features will be learned to minimize the error between the model's output and actual stock market index closing price. Three units from the Prevention Module and two units from the Prediction Module are fed into the next fully connected layer which has two neurons, which means that the five features are used to extract two features to minimize loss.

Note that the number of neurons varies depending on the predicted index. We will explain the reason why we built a separate model for S&P500 and KOSPI200. Recall that deep learning is a data-driven approach. DNNs extract features from the input data to make the output close to the target value. If the input data are different, different features will be extracted. In addition, the number of features may be different; therefore, we need to adjust the number of neurons (which determines the number of features in the layer) of each layer until we get the optimized architecture, which brings us the lowest validation loss. In our experiment, we utilized the stock market information of S&P500 and KOSPI200. There would be common factors between the two markets; however, each market has its own features. It is important to find the optimal hyper-parameters so that we can capture these differences. It would be worthwhile to model all the stock market indexes with an architecture, but for now, finding the optimal architecture for each market index is the most effective method to enhance the prediction accuracy.

3.2. Input data preparation for the Prevention Module

Data augmentation has been mainly suggested as an alternative when training data are limited because training DNNs on a small dataset results in overfitting. Recall that we collected historical stock data from January 4, 2000. Our originally retrieved dataset has at most 2,354 data points. We were concerned about the overfitting problem that the lack of training data could provoke. To prevent overfitting, we propose a data augmentation approach with the LSTM network module using 10 companies' stock that are highly correlated to the stock market index as shown in the second step of Fig. 4. We obtained combinations of 10 companies taken 5 at a time and fed them into the Prevention Module of ModAugNet as seen in the third step of Fig. 4. Considering this combination method, we expected to gain two benefits in the training networks. First, we could use abundant data; the number of data points increased by 252 times, which is $\binom{10}{5}$, and the total number of data points available becomes at least 500,000. Second, our network would learn general patterns from data. Recall that we feed the network any possible five stocks' combination reaching 252 combinations. As the network does not learn from one specific

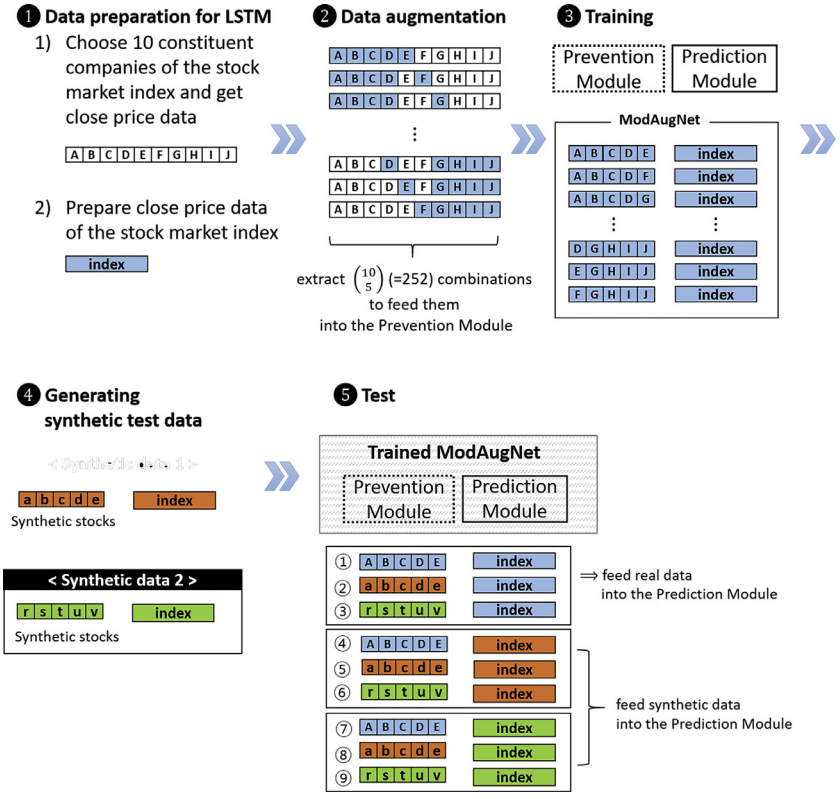

Fig. 4. Process of forecasting framework for stock market index.

Table 1
Specifications of the training input data fed to each model.

	Prevention Module	Prediction Module
SingleNet	None	Stock market index
ModAugNet-f	Pre-fixed five companies' stock prices	
ModAugNet-c	252 combinations of five companies' stock prices	

combination of five stocks, it is expected to learn from the general perspective.

3.3. Training

For joint training, we used three different loss functions; the mean squared error (MSE), mean absolute percentage error (MAPE) and mean absolute error (MAE). In addition to the widely-known indicators (MSE and MAE), the MAPE, which is also called heteroscedasticity-adjusted MAE, was applied as a non-linear loss measurement (Fuertes, Izzeldin, & Kalotychou, 2009; Kristjanpoller & Minutolo, 2016). These loss functions are defined as follows:

$$\mathrm{MSE} = \frac{1}{N}\sum_{t=1}^{N}(a_t - p_t)^2, \tag{8}$$

$$\mathrm{MAPE} = \frac{1}{N}\sum_{t=1}^{N}\left|1 - \frac{p_t}{a_t}\right|, \tag{9}$$

$$\mathrm{MAE} = \frac{1}{N}\sum_{t=1}^{N}|a_t - p_t|, \tag{10}$$

where a_t and p_t are the actual and predicted values at time t respectively, and N refers to the number of observations. We used the Adam optimizer (Kingma & Ba, 2014) for training our LSTM-based model and $L2$ regularization has also been applied to avoid overfitting during training (Nowlan & Hinton, 1992). All the training processes have been implemented using Keras (Chollet et al., 2015).

We trained three different models as shown in Table 1. First, SingleNet is trained only by stock market index as depicted in Fig. 2. It means that SingleNet is trained without the Prevention Module. Next, ModAugNet is trained by feeding one specific combination of pre-fixed five companies' stock prices into the Prevention Module and stock market index into the Prediction Module. The 5 companies have been selected in descending order of Pearson's correlation coefficient with stock market index among the 10 retrieved companies. This specific combination is one of the $\binom{10}{5}$ (=252) combinations that could be drawn from the 10 retrieved companies. This model is denoted as ModAugNet-f and is trained to examine the role of the Prevention Module in comparison to ModAugNet-c. Finally, ModAugNet is trained by the above mentioned augmented data. One of the 252 combinations was chosen every 200 epochs and fed into the Prevention Module for the duration of the subsequent 200 epochs. The learning rate, batch size, and number of epochs are set at 0.00005, 32, and 200 respectively, and we used the ReLU activation function (Nair & Hinton, 2010) for each layer with the exception of the output layer for which we used the linear activation. This is the network of interest, which is referred to as ModAugNet-c.

3.4. Synthetic test data for the analysis of the trained ModAugNet

We generated synthetic data to investigate how the two modules of trained ModAugNet are working to come up with the final prediction. Two types of synthetic data were generated as in-

<Synthetic data 1>

Stock 1	Stock 2	Stock 3	Stock 4	Stock 5
30.2700*	19.3400*	8.6000*	7.3200*	25.0600*
95.0500**	58.5000**	80.0625**	46.3125**	107.2300**
93.9773	35.6673	42.8646	21.5123	42.5319
86.3682	40.0774	71.9137	32.7380	36.4135
90.8230	37.6242	32.6242	42.2130	50.0072
42.0761	32.7588	49.5146	8.0059	65.7760
54.4551	23.6496	47.7760	30.3630	60.7512
78.3708	22.1363	16.7528	7.4331	42.8505
45.1511	53.7986	29.9641	40.6810	87.0722
80.9823	58.3798	62.5843	25.7652	90.0763
52.9144	21.6238	13.5990	11.8126	75.0358
58.8015	32.4412	79.2704	21.6826	32.8648
80.6418	53.8477	72.9355	27.1330	66.6216
90.1018	41.7594	69.5793	7.8128	91.6469
50.9637	20.0238	32.1205	39.1383	64.3313
55.7780	47.6092	66.8461	30.8644	44.3372
74.5079	29.5871	11.4764	41.3768	57.5522
33.7314	44.9736	12.8056	26.9845	82.0032
88.5760	40.9349	15.5002	23.7163	69.8982
65.4187	53.4469	26.3636	15.0815	79.6973
83.7899	45.5478	16.6434	8.9368	39.2880
80.1729	46.2699	8.7943	46.0460	91.7732

<Synthetic data 2>

Stock 1	Stock 2	Stock 3	Stock 4	Stock 5
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10
10	10	10	10	10

Fig. 5. Example of synthetic test data.

dicated in the fourth step of Fig. 4. As we had two types of synthetic test data and the original test data, we fed one of those into each module at a time as observed in the fifth step of Fig. 4. Thus, we have nine different test input sets, and they are numbered and written in circles in the fifth step of Fig. 4. Detailed analyses will be provided by comparing the test errors obtained from varying test input in Section 4.

Fig. 5 briefly shows what has been generated as noised data. Note that the values are not scaled. It has been captured from the original datasheet before implementation in Keras. These raw values before normalization make it easier to grasp how the synthetic test data look. As mentioned above, we have two types of synthetic data and the original test data. The original test dataset comprises the fixed five companies' stocks which have the highest Pearson's correlation coefficients with stock market index. We referred to this original test dataset for the Prevention Module as 5 stocks. For the Prediction Module, the original test data is referred to as the name of the corresponding stock market index, either S&P500 or KOSPI200. From those five stocks' closing prices, we extracted the minimum and maximum values for each stock as observed in Synthetic Data 1 in Fig. 5. We marked the minimum value of each stock with* and maximum with** in the light-blue shaded area. We filled each data point of a stock with a number randomly chosen from a uniform distribution ranging within the interval (min*, max**). For synthetic index 1, we extracted the minimum and maximum value of the S&P500, and the data points are drawn from a uniform distribution between the corresponding interval. For KOSPI200, we followed the same procedure as we did for S&P500. In the second noised dataset in Fig. 5, all the data points are filled with an arbitrary chosen value of 10. For convenience of analysis, we will denote the first dataset as Synthetic data 1 and the second as Synthetic data 2.

When the synthetic test data is fed into the ModAugNet, the value will be normalized as in Eq. (7). In Fig. 6, an example of the normalized synthetic test data 1 is depicted based on Stock 1 in Fig. 5. As the values in Synthetic data 1 are chosen from the uniform distribution, their graphical patterns will look similar to the pattern of Fig. 6. Every normalized value for Synthetic data 2 is computed to 0 since the data consists of a constant value, which is 10.

Table 2
Test MSEs of ModAugNet-f.

	Prevention Module	Prediction Module	Test MSE
(1)	5 stocks	S&P500	1665.31
(2)	Synthetic data 1		9910.59
(3)	Synthetic data 2		3630.42
(4)	5 stocks	Synthetic data 1	208,459.45
(5)	Synthetic data 1		2,379,50.25
(6)	Synthetic data 2		2,297,98.13
(7)	5 stocks	Synthetic data 2	818,944.20
(8)	Synthetic data 1		7,856,18.40
(9)	Synthetic data 2		9,757,42.48

Table 3
Test MSEs of ModAugNet-c.

	Prevention Module	Prediction Module	Test MSE
(1)	5 stocks	S&P500	342.48
(2)	Synthetic data 1		
(3)	Synthetic data 2		
(4)	5 stocks	Synthetic data 1	389,239.25
(5)	Synthetic data 1		
(6)	Synthetic data 2		
(7)	5 stocks	Synthetic data 2	784,151.16
(8)	Synthetic data 1		
(9)	Synthetic data 2		

4. Experimental results

Two stock market datasets are used to investigate whether our forecasting framework performs appropriately in general. As depicted in the fifth step of Fig. 4, trained ModAugNet is evaluated with nine combinations of test data, including the synthetic data. Nine different combinations are fed into ModAugNet-f and ModAugNet-c, and the corresponding test results are provided in Tables 2–5. The test result of SingleNet, which is trained without the Prevention Module, is also provided to clarify the effectiveness of the Prevention Module. As three different loss functions of MSE,

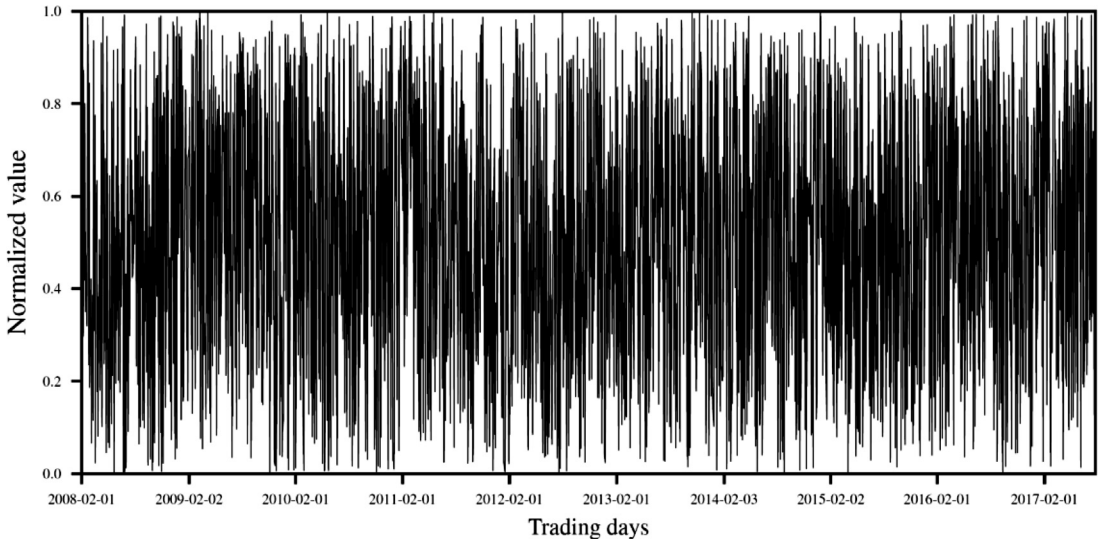

Fig. 6. An example of the normalized synthetic test data 1 (depicted based on Stock 1 in Fig. 5).

Fig. 7. Comparison of the predicted values from SingleNet to the actual values.

Table 4
Test MSEs of ModAugNet-f.

	Prevention Module	Prediction Module	Test MSE
(1)	5 stocks	KOSPI200	14.45
(2)	Synthetic data 1		20.53
(3)	Synthetic data 2		74.67
(4)	5 stocks	Synthetic data 1	3888.90
(5)	Synthetic data 1		4081.95
(6)	Synthetic data 2		4684.85
(7)	5 stocks	Synthetic data 2	31,032.15
(8)	Synthetic data 1		315,10.57
(9)	Synthetic data 2		338,23.00

Table 5
Test MSEs of ModAugNet-c.

	Prevention Module	Prediction Module	Test MSE
(1)	5 stocks	KOSPI200	7.56
(2)	Synthetic data 1		
(3)	Synthetic data 2		
(4)	5 stocks	Synthetic data 1	6124.05
(5)	Synthetic data 1		
(6)	Synthetic data 2		
(7)	5 stocks	Synthetic data 2	33,904.02
(8)	Synthetic data 1		
(9)	Synthetic data 2		

MAE, and MAPE were used, the corresponding out-of-sample forecast errors are summarized in Tables 8 and 9. Detailed analyses will be provided in terms of MSE forecasts as a representative.

4.1. Forecasting S&P500

4.1.1. Test results of SingleNet

The out-of-sample forecast MSE of S&P500 from SingleNet for 2,388 test data points is 746.8 and the predicted values are plotted along with the actual values in Fig. 7.

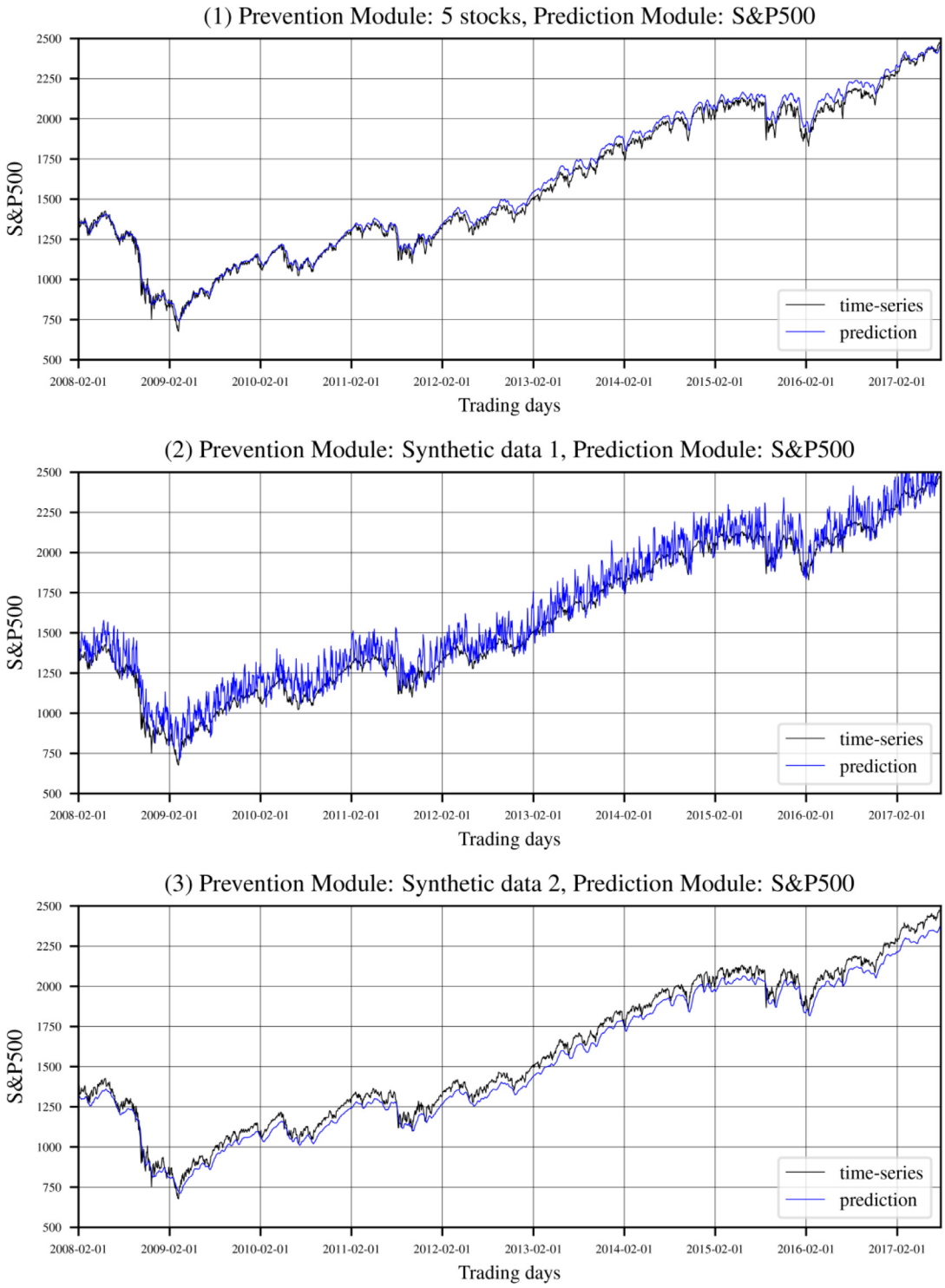

Fig. 8. Comparison of the predicted values from ModAugNet-f to the actual values (S&P500 original test data fed into the Prediction Module).

4.1.2. Test results of ModAugNet-f

The representative test MSE for ModAugNet-f is 1665.31 as it is achieved by feeding appropriate test data into each module, as observed in Table 2. The performance is significantly worse than SingleNet, which is supported by the statistical test results in Table 10. As ModAugNet-f has more parameters than SingleNet despite the insufficient data points available, the model performance may be deteriorated in comparison to SingleNet. More specifically, the available training data points for ModAugNet-f is 1988, while 500,976 data points are available for training ModAugNet-c (252 times greater than the available data points for ModAugNet-f).

The fourth column lists the test MSE obtained feeding different test data input into each module. The test MSE increases as noised synthetic test data are fed into the Prevention Module. On the other hand, the noised test input data fed into the Prediction Module dramatically deteriorates the model performance.

Input-varying test errors are visualized in Figs. 8–10. The number above each graph indicates the first column of Table 2 (or test

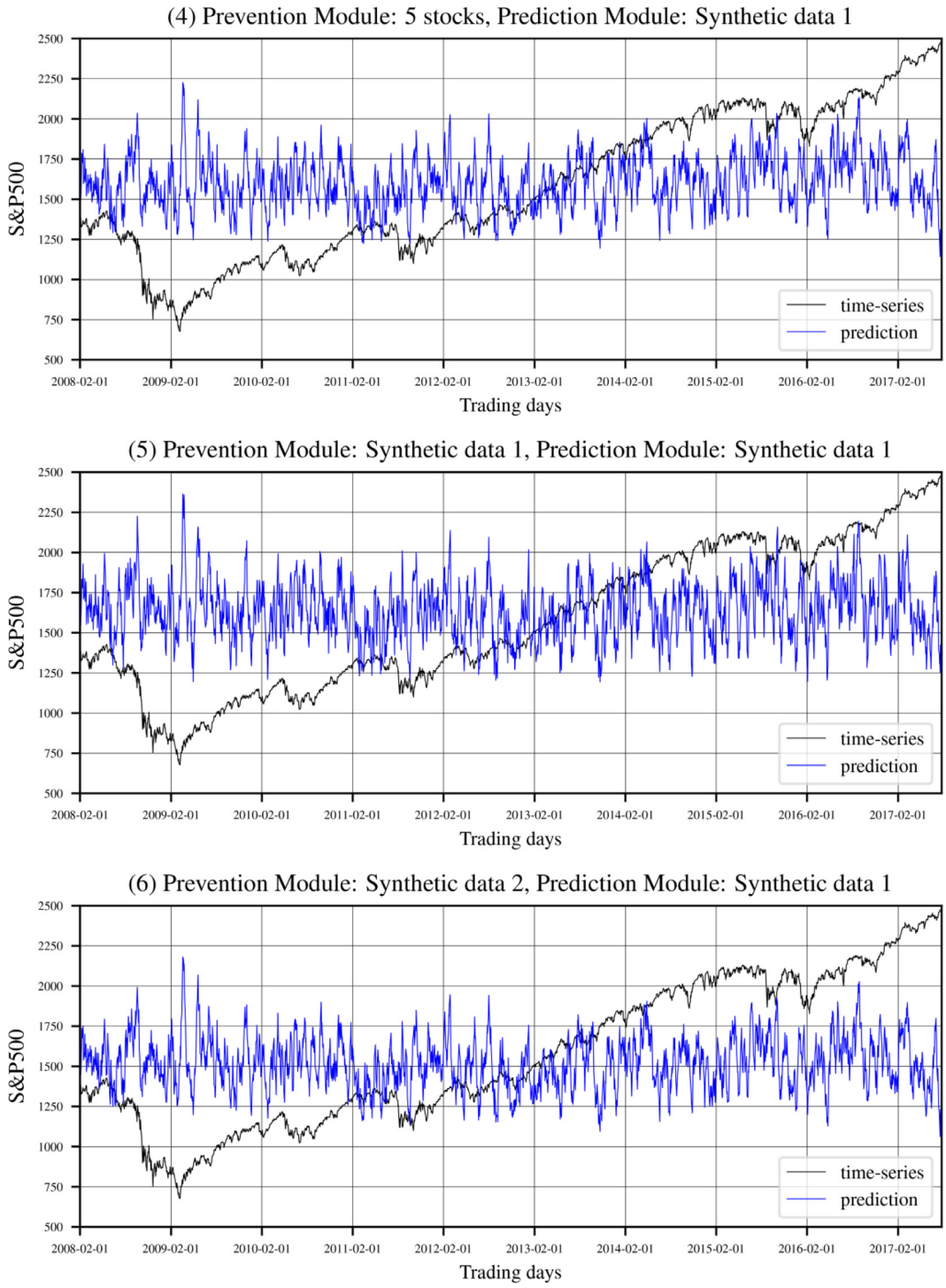

Fig. 9. Comparison of the predicted values from ModAugNet-f to the actual values (Synthetic Data 1 fed into the Prediction Module).

input categories as marked in circles in the fifth step of Fig. 4). Fig. 8 represents the change of predicted values in terms of different input data fed into the Prevention Module. The first graph is the representative test result of ModAugNet-f with appropriate test input data. As noised synthetic data are fed into the Prevention Module, a degradation of the performance is visualized in the second and third graphs of Fig. 8. The overall pattern of the three graphs reflects the input data of the Prediction Module, which is the original S&P500 time-series. However, we confirm that the input data of the Prevention Module is also reflected in the graphs. In particular, the second graph shows the zigzag patterns of Synthetic data 1 as observed in Fig. 6.

Figs. 9 and 10 illustrate the change of predicted values in terms of different input fed into the Prevention Module while Prediction Module always receives the synthetic data. It is shown that the Prediction Module is influential, wherein the shape of its test input is mainly depicted in the output graphs. In Fig. 9, the overall pattern of the three graphs reflects the pattern of Synthetic data 1, which is fed into the Prediction Module in common, while there are marginal differences among the graphs because of the different

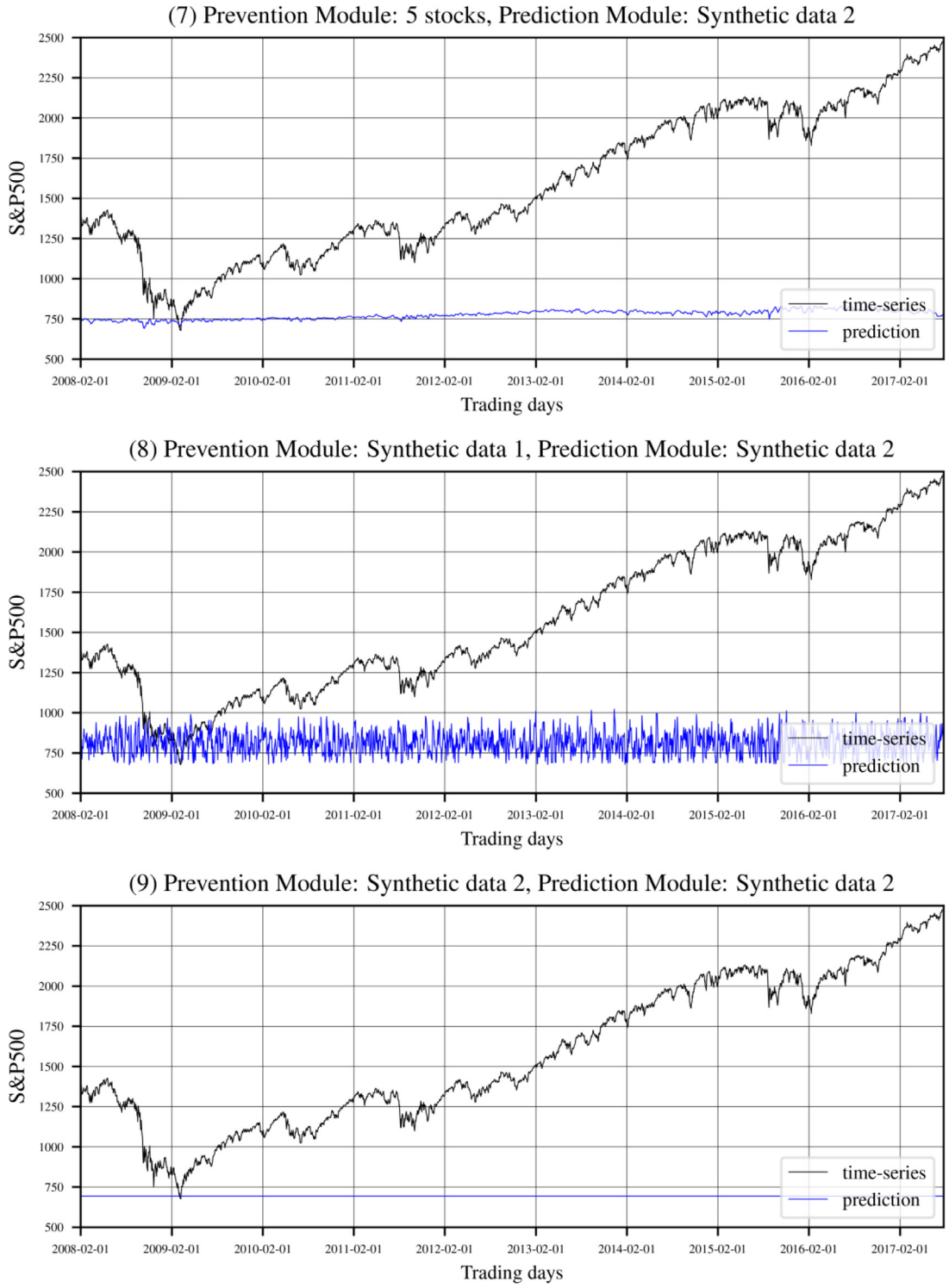

Fig. 10. Comparison of the predicted values from ModAugNet-f to the actual values (Synthetic Data 2 fed into the Prediction Module).

test input data fed into the Prevention Module. In Fig. 10, Synthetic data 2 is consistently provided to the Prediction Module. The third graph in Fig. 10 represents the case in which both the modules take Synthetic data 2, which consists of the same values. The first and second graphs also reflect the shape of the test input of the Prevention Module.

Recall that the first graph in Fig. 8 is a result of the case in which both modules take the original test data; 5 stocks given to the Prevention Module and S&P500 given to the Prediction Module. Despite 5 stocks still given to the Prevention Module, if the test input data of the Prediction Module changes, we could confirm a degradation of performance in the first graph of Fig. 9 and Fig. 10.

4.1.3. Test results of ModAugNet-c

There are two remarkable test results to explain ModAugNet-c. First, the model performance has significantly improved. The representative test MSE for ModAugNet-c is 342.48 (rounded to

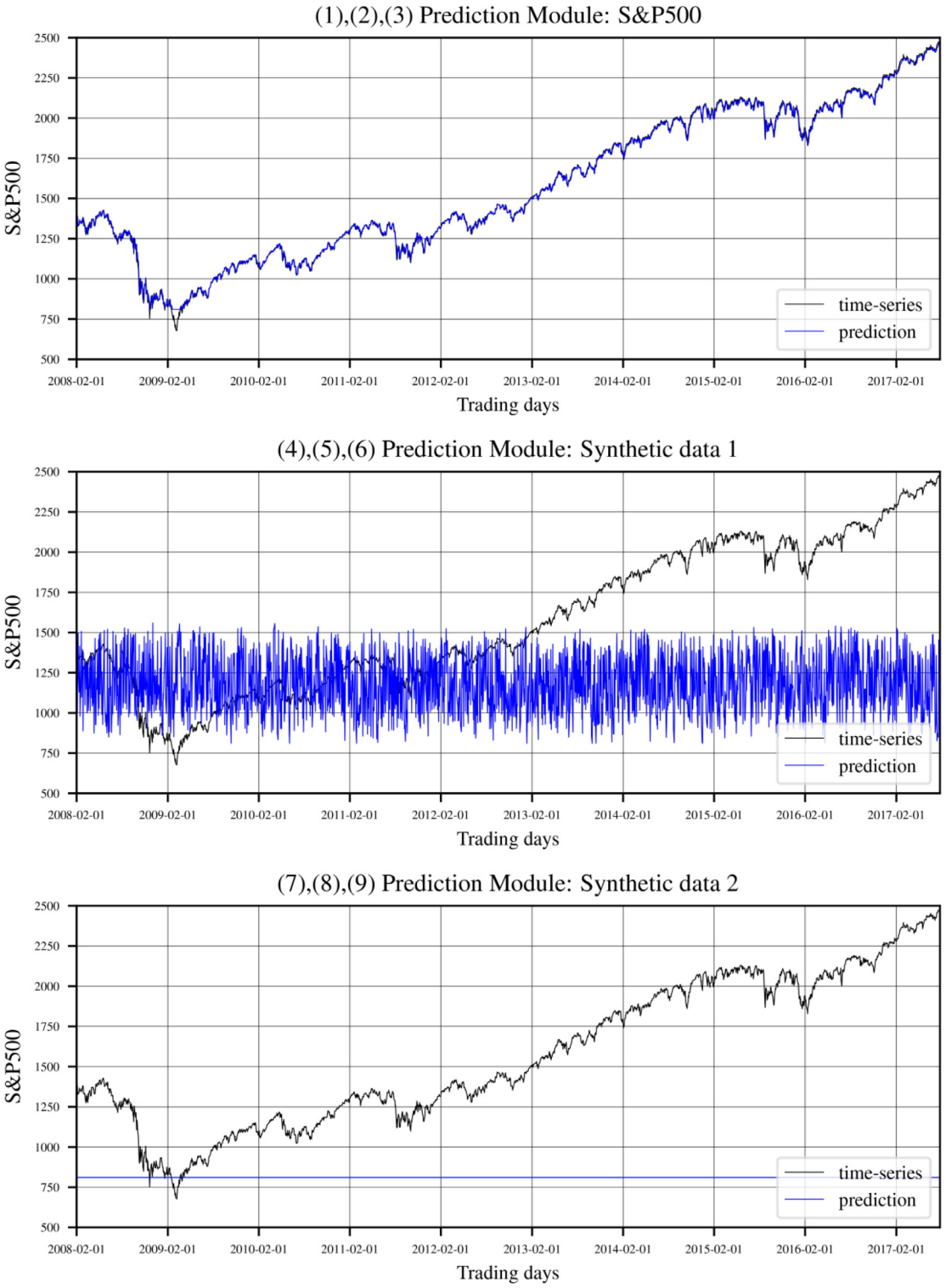

Fig. 11. Comparison of the predicted values from ModAugNet-c to the actual values.

two decimal places), while the test MSE of the SingleNet, where the Prevention Module is absent, is 746.80. Second, the test errors remain the same, regardless of which input data is fed into the Prevention Module. This can be explained through the above-mentioned test method in Section 3.4. to verify the influences of the input features in training. In ModAugNet-c, the test input of the Prevention Module never affects the test performance. It can be interpreted as a disappearance of the Prevention Module when training is completed. Test performance is solely dependent on the test input data of the Prediction Module. Based on the results, we can evaluate that the various training input features of the Prevention Module made no contribution to learning to solve the regression problem, but helped in preventing the problem of overfitting.

Input-varying test errors for ModAugNet-c are visualized in Fig. 11. The numbers above each graph indicate the first column of Table 3 (or test input categories as marked in circles in the

Fig. 12. Comparison of the predicted values from SingleNet to the actual values.

fifth step of Fig. 4). The first graph represents the change of predicted values in terms of different input data fed into the Prevention Module, while 5 stocks were fed into Prediction Module consistently. Inconsistent with the results shown in Table 2, only the test input of the Prediction Module is reflected, regardless of the noised synthetic test input data fed into the Prevention Module in the first graph. The second and third graphs also reflect only the test input data of the Prediction Module.

4.2. Forecasting KOSPI200

4.2.1. Test results of SingleNet

Forecasted MSE of KOSPI200 from SingleNet for 2,354 test data points is 14.53 and the predicted values are plotted along with the actual values in Fig. 12.

4.2.2. Test results of ModAugNet-f

The representative test MSE of KOSPI200 for ModAugNet-f is 14.45, which is close to the forecasted MSE of SingleNet. The test results for KOSPI200 are generally consistent with the results of S&P500. As given in Table 4, the test MSE increased as synthetic test data are fed into the Prevention Module. However, the test MSE dramatically increased when noised synthetic data were input in the Prediction Module.

The test MSE obtained from the nine different test input sets are shown in Figs. 13–15. Note that the number above each graph denotes the first column of Table 4 (or test input categories marked in circles in the fifth step of Fig. 4). Fig. 13 depicts the change of the predicted values as different input data are fed into the Prevention Module, while KOSPI200 is consistently fed into the Prediction Module. The first graph represents the representative test result of ModAugNet-f because each module takes the original test input data; 5 stocks fed into the Prevention Module and KOSPI200 fed into the Prediction Module. As noised test data are fed into the Prevention Module, a degradation of the performance is shown in the graphs of Fig. 13. The overall pattern of the three graphs reflects the input data of the Prediction Module, which is the original KOSPI200 time-series.

Note that we did not feed synthetic data 1 and synthetic data 2 as training data. Thus, it is natural that the test performance deteriorates when these artificial test inputs are fed into the network. ModAugNet-f extracts the features from the Prevention Module as opposed to ModAugNet-c. As the Prevention Module of ModAugNet-f contributes to producing the prediction output, the test input of the Prevention Module matters. Thus, we fed the same test input into the Prediction Module and investigated what happened if we fed the synthetic test input into the Prevention Module. Figs. 14 and 15 illustrate the change of the predicted values as different input data are fed into the Prevention Module, while the Prediction Module always takes synthetic data. In general, we could observe the influences of the Prediction Module in the trained ModAugNet-f as the test input data of the Prediction Module are reflected in the graphs for the most part. We fed Synthetic data 1 into the Prediction Module in common to produce the three graphs in Fig. 14. Note that Synthetic data 1 consists of a number randomly chosen from a uniform distribution as depicted earlier in Fig. 6. The pattern of Synthetic data 1 is highly reflected in the three graphs, although there are marginal differences among the graphs because of the different test input data fed into the Prevention Module. In Fig. 15, Synthetic data 2 is consistently fed into the Prediction Module. In Synthetic data 2, all data points are filled with an arbitrary chosen value of 10. The pattern of Synthetic data 2 is mainly reflected in the three graphs of Fig. 15 despite the marginal differences among the graphs which result from the different test input data fed into the Prevention Module. The third graph in Fig. 15 represents the case where both modules receive Synthetic data 2, which comprises the same constant values. Since both modules take the same test data, constant values are depicted for the network output. The test input data of the Prevention Module are also reflected in the first and second graphs of Fig. 15.

4.2.3. Test results of ModAugNet-c

ModAugNet-c tested with KOSPI200 also showed interesting results, similar to those of forecasting S&P500 with ModAugNet-c. First, the model performance is ameliorated. The representative test MSE of forecasting KOSPI200 with ModAugNet-c is 7.56 (rounded to two decimal places), while the comparative SingleNet yielded the test MSE of 14.45. Second, the test input data fed into the Prevention Module never affect the test performance as the test performance is entirely dependent on the test input data fed into the Prediction Module. The Prevention Module helps to prevent the network from overfitting during training, and then gets deactivated when training is completed.

Input-varying test errors for ModAugNet-c are visualized in Fig. 16. The numbers above each graph indicates the first column of Table 5 (or test input categories as marked in circles in the fifth

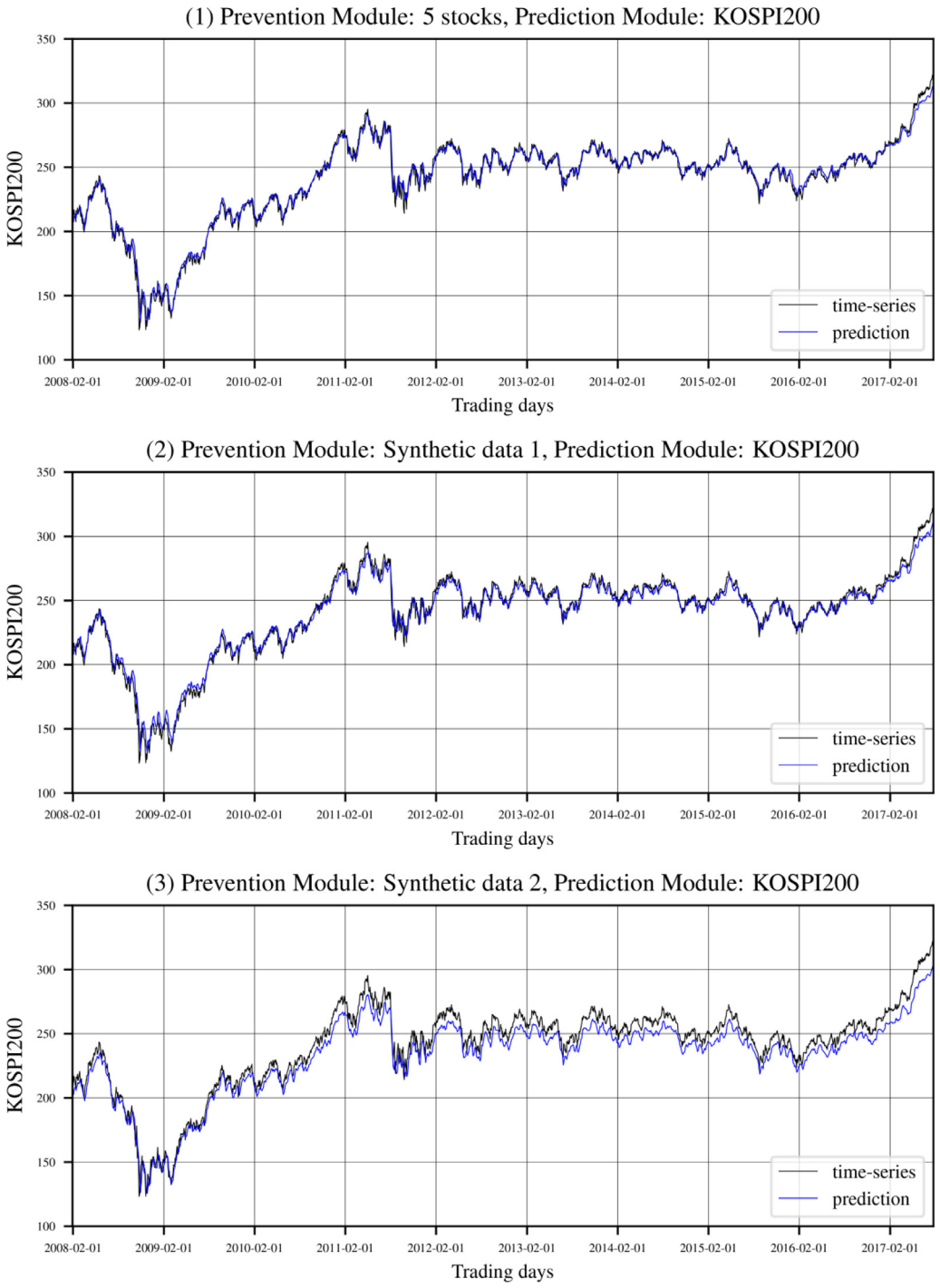

Fig. 13. Comparison of the predicted values from ModAugNet-f to the actual values (KOSPI200 original test data fed into the Prediction Module).

step of Fig. 4). As the test input of the Prevention Module does not affect the model performance, the result is the same graphs when the test input data fed into the Prediction Module remains the same.

4.3. Examining the weight connections of each module

In this section, we provide the detailed weights between the fully connected layers, where the two modules are combined as depicted in the blue-colored FC layers in Fig. 3. To forecast KOSPI200 and S&P500, the number of neurons in each layer is different from each other. By examining the weight connections from each module to the next layer, we can verify the increased influence of the Prediction Module after training with the augmented dataset. The influence of each module is provided by calculating the absolute mean of the weight connections from each module as the magnitude of the weight determines the significance of the node regardless of its sign.

The absolute mean of each module to forecast S&P500 is given in Table 6. First, we can observe that the weight connections from the Prediction Module to the next layer are stronger than the Prevention Module. Second, the magnitude of the connections from the Prediction Module increased with ModAugNet-c, where the 252 times augmented data were used for training. The detailed

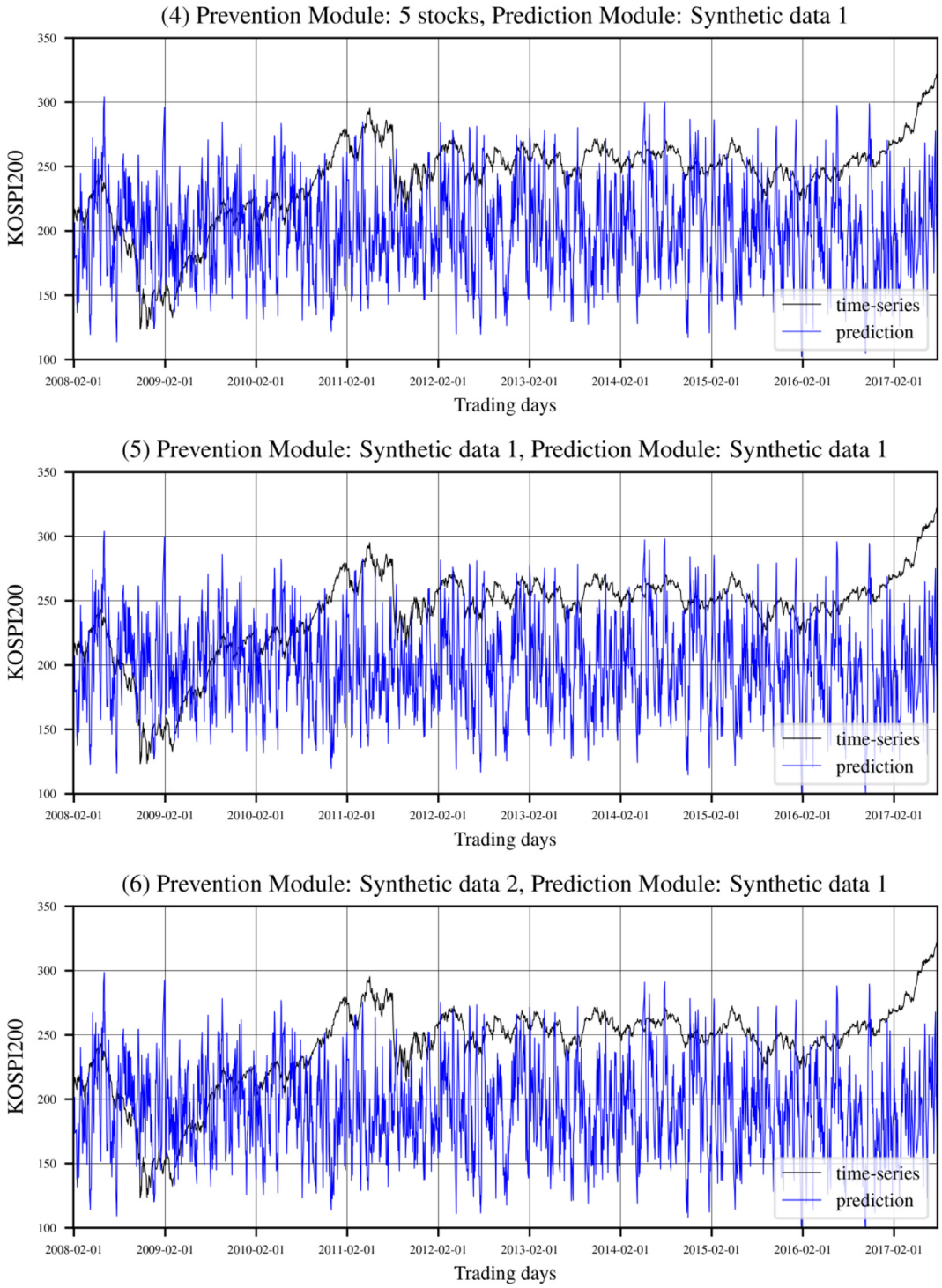

Fig. 14. Comparison of the predicted values from ModAugNet-f to the actual values (Synthetic Data 1 fed into the Prediction Module).

Table 6
Absolute mean of weight connections of each module to forecast S&P500.

	Prevention Module	Prediction Module
ModAugNet-f	0.411	0.806
ModAugNet-c	0.422	0.995

Table 7
Absolute mean of weight connections of each module to forecast KOSPI200.

	Prevention Module	Prediction Module
ModAugNet-f	0.464	0.973
ModAugNet-c	0.371	1.260

weights between the fully-connected layers of ModAugNet-f and ModAugNet-c to forecast S&P500 are illustrated in Figs. 17 and 18, respectively.

In Table 7, the absolute mean of each module to forecast KOSPI200 is provided. Similar to the case of S&P500, we can observe that the weight connections from the Prediction Module to

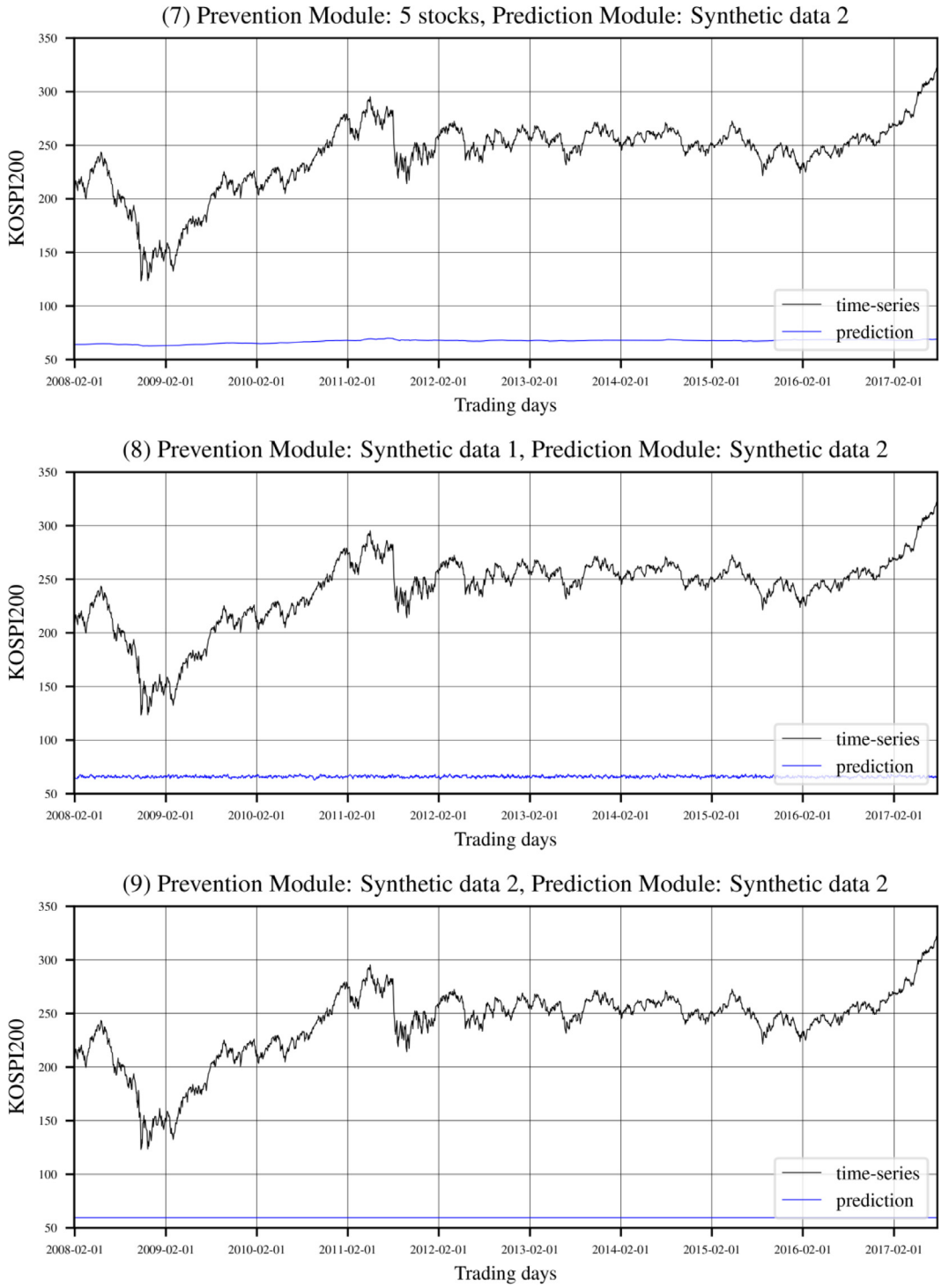

Fig. 15. Comparison of the predicted values from ModAugNet-f to the actual values (Synthetic Data 2 fed into the Prediction Module).

the next layer are stronger than that of the Prevention Module. In addition, the strength of the connections from the Prediction Module intensifies with ModAugNet-c, while the average weight connections from the Prevention Module decrease. The detailed weights between the fully-connected layers of ModAugNet-f and ModAugNet-c to forecast KOSPI200 are illustrated in Figs. 19 and 20, respectively.

In the study, we added a Prevention Module to the model to create a hybrid model to undertake data augmentation within the model. The input of the Prediction Module is the past 20-days' stock index values that are the most closely related to the stock index value of the next day, and as the input of Prevention Module, 5 out of 10 component stocks of the stock market index are selected randomly, and their 20-days' past data is entered into the input of the Prevention Module. In other words, the Prediction Module always takes the same variables as the input variables, but in the Prevention Module, five different variables (stock prices) for each iteration are entered as input variables. In extracting features based on training data in neural networks, if input variables (the number and order of input variables) are not al-

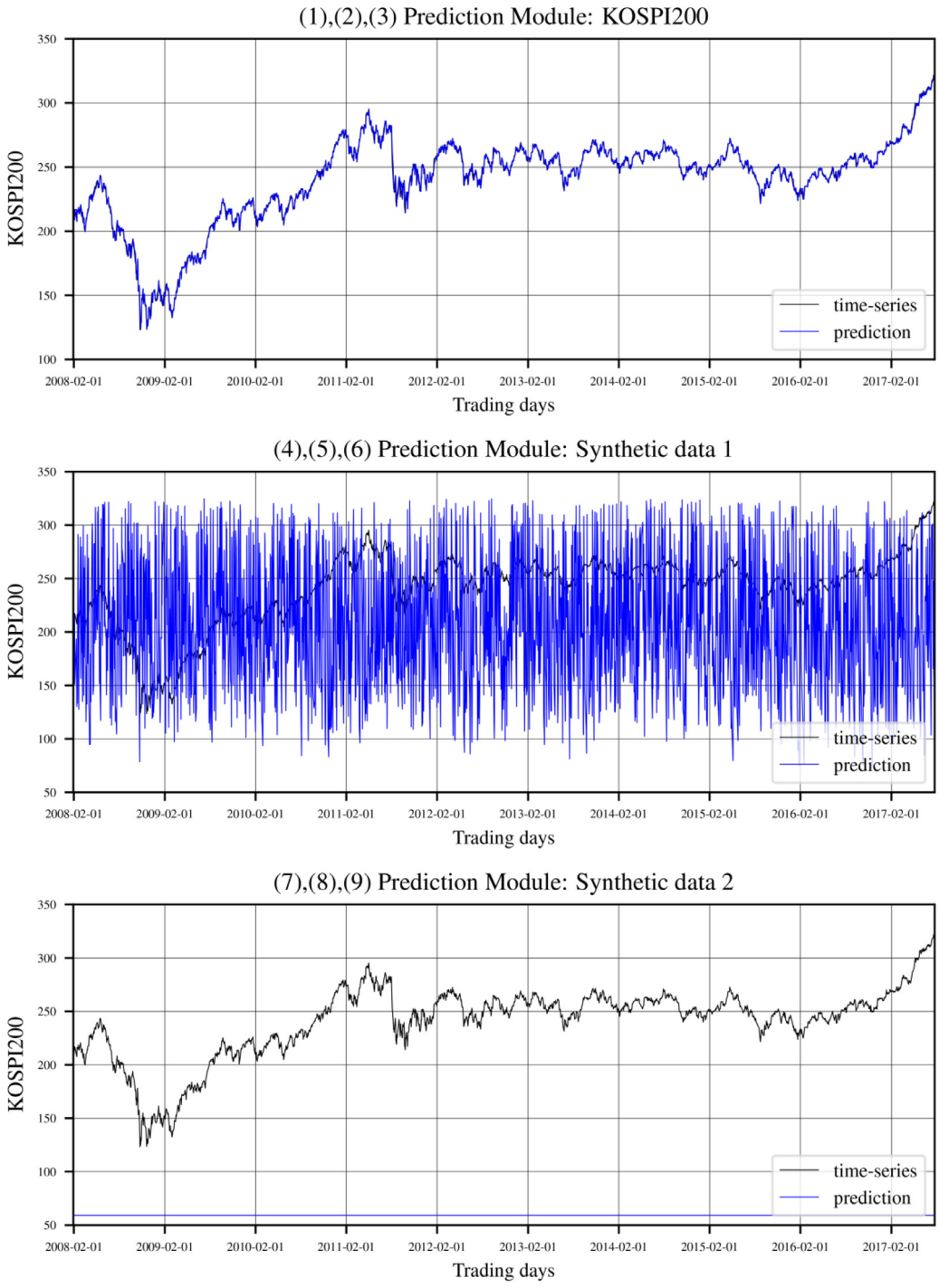

Fig. 16. Comparison of the predicted values from ModAugNet-c to the actual values.

ways the same, and different combinations of variables are given as input variables for each iteration, then it is challenging to learn the features. Therefore, the proposed model, ModAugNet-c, makes the final prediction using only the features learned in the Prediction Module. The features extracted from the Prevention Module do not contribute to the final prediction, but they prevent overfitting by causing data augmentation, thus helping to improve generalization accuracy. In other words, similar to increasing the training data by adding random noise to the original data in the general data augmentation techniques, the Prevention Module plays a role of amplifying the training data by adding noise. That is, five stock prices randomly selected for each iteration act as noise. Therefore, in Figs. 18 and 20, weights associated with the features of the Prevention Module are close to zero. The results of ModAugNet-f also support our argument. ModAugNet-f differs from ModAugNet-c in that five stocks are fixed instead of putting different combinations of stock prices as input variables for each iteration. Therefore, this is an addition of input variables, not data augmentation. In this case, we can learn the features that are useful for the target because the input variables are the same, and the weights associated with the features of the Prevention Module are larger than those of ModAugNet-c, as shown in Figs. 17 and 19. However, the out-of-sample prediction errors of ModAugNet-c are lower than those of ModAugNet-f. This is because the training data is as small as

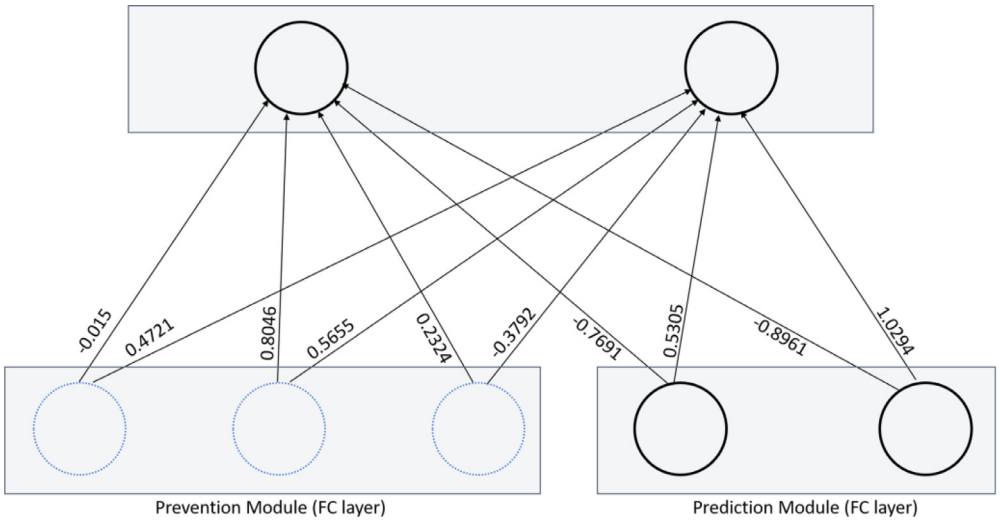

Fig. 17. Weights between the fully-connected layers of ModAugNet-f to forecast S&P500.

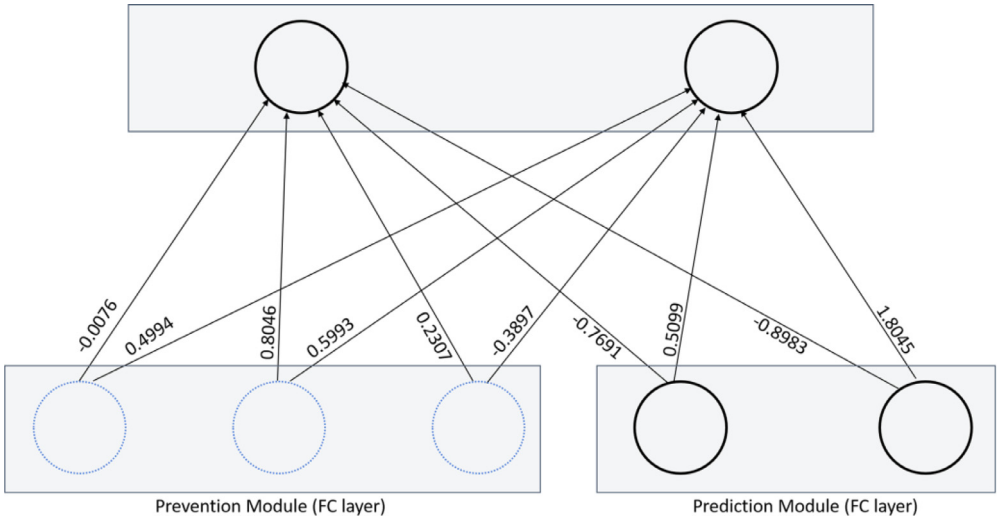

Fig. 18. Weights between the fully-connected layers of ModAugNet-c to forecast S&P500.

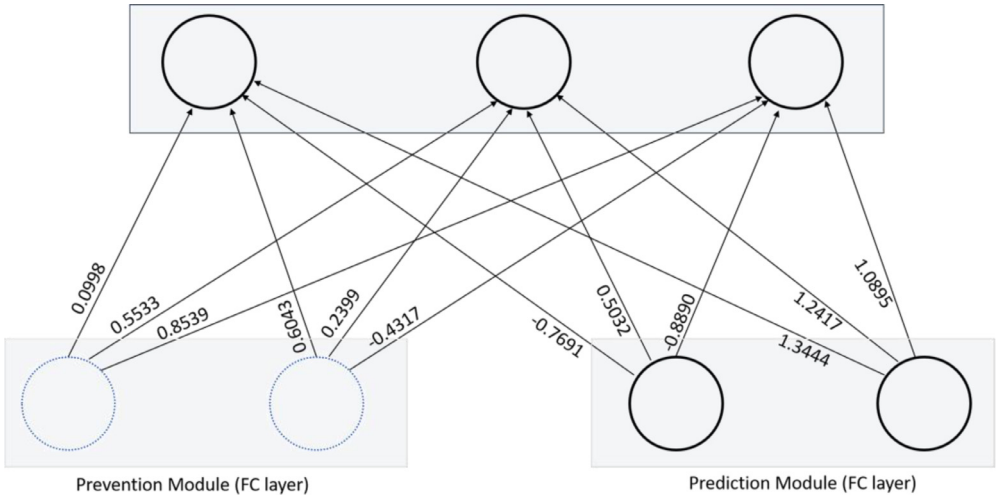

Fig. 19. Weights between the fully-connected layers of ModAugNet-f to forecast KOSPI200.

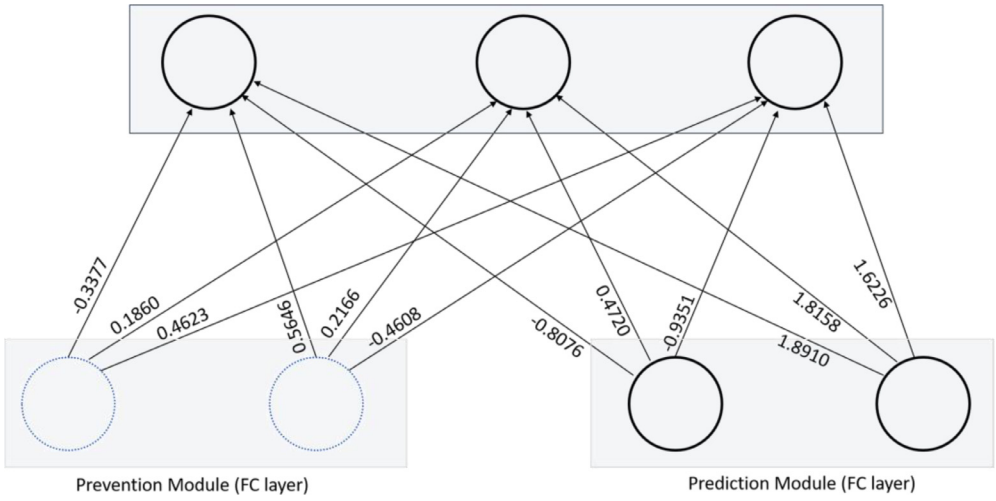

Fig. 20. Weights between the fully-connected layers of ModAugNet-c to forecast KOSPI200.

Table 8
Results of the out-of-sample forecasting performances for S&P500 with 3 different loss functions.

	MSE	MAPE (%)	MAE
DNN	943.38	2.0698	20.701
RNN	831.40	1.9574	19.467
SingleNet	746.80	1.6672	17.918
ModAugNet-f	1665.31	2.0089	33.849
ModAugNet-c	342.48	1.0759	12.058

Table 9
Results of the out-of-sample forecasting performances for KOSPI200 with 3 different loss functions.

	MSE	MAPE (%)	MAE
DNN	15.06	1.8691	2.295
RNN	14.66	1.3694	2.291
SingleNet	14.53	1.3241	2.229
ModAugNet-f	14.45	1.2397	2.832
ModAugNet-c	7.56	1.0077	1.975

1900, thus preventing overfitting rather than adding input variables, which is more helpful in improving generalization performance.

4.4. Results of the out-of-sample forecasting performances with different loss functions

We adopted the three different loss functions of the MSE, MAPE, and MAE as mentioned in Section 3.3, and the corresponding results are summarized in Tables 8 and 9. Furthermore, we trained the two widely-known architectures (DNN and RNN) as additional baseline methods to examine whether our proposed ModAugNet-c outperforms the conventional ANNs in terms of forecasting errors. For training DNN and RNN, we chose an architecture with two hidden fully-connected layers each of which has 5 and 3 neurons. The learning rate, batch size, and number of epochs are set at 0.0001, 32, and 200 by default, and we used the ReLU activation function for the hidden layers. In Tables 8 and 9, five models to predict the next day's stock market index closing price are compared in terms of the three error measures. Note that the MAPE values are given in the percentage form which is multiplied by 100 to the original value so that we can state the differences between the errors rather than using the decimal form. ModAugNet-c provides the smallest errors among all the error measures. These results are backed-up by the following statistical tests in Section 4.5.

In Table 8, we confirmed the test performance improvement of the proposed model of ModAugNet-c in S&P500 forecasting. We observed that the test MSE of ModAugNet-c decreased to 63.7%, 58.8%, and 54.1% for the test MSE of the DNN, RNN, and SingleNet, respectively. The test MAPE of ModAugNet-c decreased to 48%, 45%, and 35.5% for the test MAPE of the DNN, RNN, and SingleNet, respectively. The test MAE of ModAugNet-c decreased to 41.8%, 38.1%, and 32.7% for the test MAE of the DNN, RNN, and SingleNet, respectively.

In Table 9, we could also observe the test performance improvement of the proposed model of ModAugNet-c in KOSPI200 forecasting. Based on the results, the test MSE of ModAugNet-c decreased to 49.8%, 48.4%, and 48% for the test MSE of the DNN, RNN, and SingleNet, respectively. The test MAPE of ModAugNet-c decreased to 46.1%, 26.4%, and 23.9% for the test MAPE of the DNN, RNN, and SingleNet, respectively. The test MAE of ModAugNet-c decreased to 13.9%, 13.8%, and 11.4% for the test MAE of the DNN, RNN, and SingleNet, respectively.

4.5. Statistical tests

Diebold–Mariano (DM) test (Diebold & Mariano, 2002) and Wilcoxon Signed rank (WS) tests were carried out to verify the equivalence of the prediction accuracy between the two models, which is to show whether there are statistically significant differences between the two models' out-of-sample forecasts. First, we will explain the DM test. The forecast errors are denoted by $e_{i,t}$, which is defined as $e_{i,t} = p_{i,t} - a_t$, for $i = 1, 2$, where a_t refers to the actual time-series and $p_{i,t}$ refers to the forecasts of a_t. Further, the null hypothesis of the equal level of prediction accuracy is $\mathbb{E}(d_t) = 0$, where $d_t \equiv \varphi(e_{1,t}) - \varphi(e_{2,t})$, and d_t refers to a loss differential with any given loss function $\varphi(\bullet)$. The DM statistics is calculated as follows:

$$DM = \frac{\bar{d}}{\sqrt{2\pi \dot{f}_d(0)/T}}, \tag{11}$$

where $\bar{d} = \frac{1}{T}\sum_{t=1}^{T}(d_t)$ and $\dot{f}_d(0)$ are consistent estimates of $f_d(0)$, which is the spectral density of the loss differential at the frequency of 0.

Second, we will explain the WS test. The observed loss differentials can be tested according to the null hypothesis of zero-

Table 10
DM and WS tests between the pairs of forecasting models of S&P500: the values below the diagonal indicate the *p*-values for the WS test and the values above the diagonal indicate the *p*-values for the DM test.

		DNN	RNN	SingleNet	ModAugNet-f	ModAugNet-c
DNN	MSE		0.01*	0.00*	0.00*	0.00*
	MAPE		0.00*	0.00*	0.14	0.00*
	MAE		0.01*	0.00*	0.00*	0.00*
RNN	MSE	0.00*		0.00*	0.00*	0.00*
	MAPE	0.00*		0.00*	0.35	0.00*
	MAE	0.00*		0.00*	0.00*	0.00*
SingleNet	MSE	0.00*	0.04*		0.00*	0.00*
	MAPE	0.00*	0.19		0.00*	0.00*
	MAE	0.00*	0.00*		0.00*	0.00*
ModAugNet-f	MSE	0.00*	0.00*	0.00*		0.00*
	MAPE	0.00*	0.00*	0.00*		0.00*
	MAE	0.00*	0.00*	0.00*		0.00*
ModAugNet-c	MSE	0.00*	0.00*	0.00*	0.00*	
	MAPE	0.00*	0.00*	0.00*	0.00*	
	MAE	0.00*	0.00*	0.00*	0.00*	

Table 11
DM and WS tests between the pairs of forecasting models of KOSPI200: the values below the diagonal indicate the *p*-values for the WS test and the values above the diagonal indicate the *p*-values for the DM test.

		DNN	RNN	SingleNet	ModAugNet-f	ModAugNet-c
DNN	MSE		0.26	0.41	0.38	0.00*
	MAPE		0.00*	0.00*	0.00*	0.00*
	MAE		0.77	0.01*	0.00*	0.00*
RNN	MSE	0.00*		0.73	0.65	0.00*
	MAPE	0.00*		0.00*	0.00*	0.00*
	MAE	0.00*		0.00*	0.00*	0.00*
SingleNet	MSE	0.00*	0.00*		0.78	0.00*
	MAPE	0.00*	0.00*		0.00*	0.00*
	MAE	0.00*	0.00*		0.00*	0.00*
ModAugNet-f	MSE	0.00*	0.00*	0.00*		0.00*
	MAPE	0.00*	0.00*	0.00*		0.00*
	MAE	0.00*	0.00*	0.00*		0.00*
ModAugNet-c	MSE	0.00*	0.00*	0.00*	0.00*	
	MAPE	0.00*	0.00*	0.00*	0.00*	
	MAE	0.00*	0.00*	0.00*	0.00*	

median loss differential, if the exact finite-sample tests are available; median$(d_t)=0$. If the sampled loss differential has a symmetrical distribution, then the null hypothesis is consistent with the above DM test, indicating an equal level of accuracy. The WS statistics is defined as follows:

$$\mathrm{WS} = \sum_{t=1}^{T} I_+(d_t)\mathrm{rank}(|d_t|), \tag{12}$$

where $I_+(d_t) = \begin{cases} 1, & d_t > 0 \\ 0, & otherwise \end{cases}$

The DM and WS test results are shown in Tables 10 and 11. The values below the diagonal in Tables 10 and 11 indicate the *p*-values for the WS test, while the values above the diagonal indicate the *p*-values for the DM test. Note that the null hypotheses of the DM and WS tests assume an equal level of prediction accuracy between the two models. We reject the null hypothesis under the *p*-values less than 0.05 and we can conclude that the prediction accuracy of the two comparative models is significantly different. Otherwise, we cannot reject the null hypothesis at significance levels of 5% or lower if the *p*-value is greater than 0.05. In Table 10 and Table 11, we marked the *p*-values less than 0.05 with*. In both the stock markets, we confirm that the outperformance of our proposed ModAugNet-c is statistically significant in all the error measures.

Table 12
Comparison of the test errors of forecasting S&P500 in recession and recovery periods.

	Total (2,388)	Recession (277)	Recovery (2,111)
MSE	342.48	1045.30	250.26
MAPE (%)	1.0759	2.6780	0.8657
MAE	12.058	20.454	10.956

5. Discussions

5.1. Impact of the financial crisis on the performance of ModAugNet-c

In our experiments, as mentioned in Section 2.2, price information between January 4, 2000 and December 31, 2007 is used as the training dataset, while price information between January 2, 2008 and July 27, 2017 is used as the test dataset. As our train period is before the financial crisis hit in 2008 and the test period contains a strong bear market until March 10, 2009, we investigated the effect of a financial crisis on the test performance of the proposed ModAugNet-c.

To begin with, we split the total test period into two subperiods as observed in Tables 12 and 13; Recession as the period between January 2, 2008 and March 9, 2009; and Recovery as the period between March 10, 2009 and July 27, 2017. The corresponding number of trading days in each period is marked in parenthe-

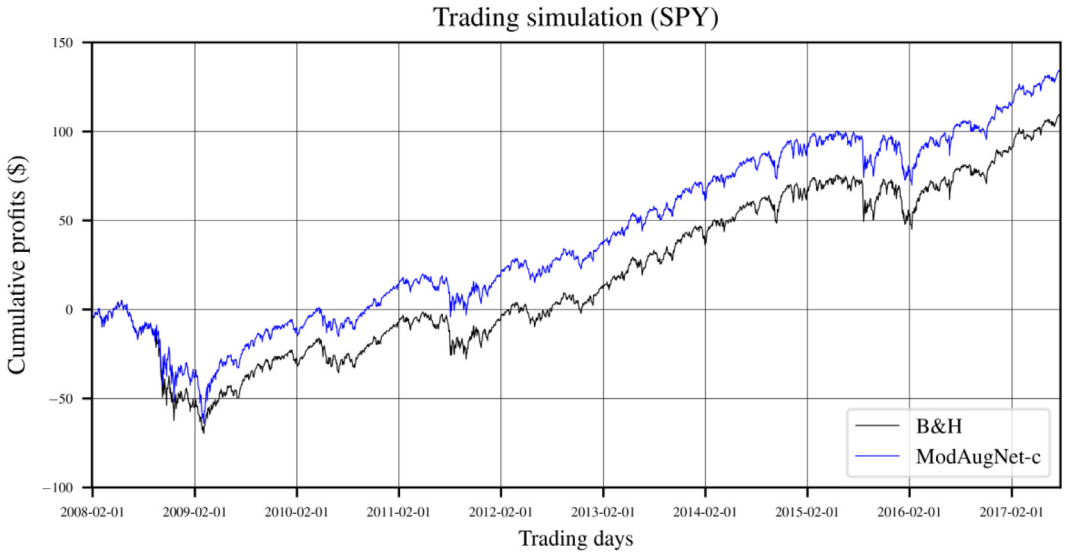

Fig. 21. Cumulative profits for a unit investment in SPY for the test period.

Table 13
Comparison of the test errors of forecasting KOSPI200 in recession and recovery periods.

	Total (2,354)	Recession (274)	Recovery (2,080)
MSE	7.56	16.81	6.34
MAPE (%)	1.0077	2.0573	0.8689
MAE	1.975	3.078	1.829

ses of Tables 12 and 13. Based on the test results obtained, we can observe a degradation of the test performance during Recession in all the cases. The main reason of this degradation is the unprecedented patterns of price movement during Recession. Recall that deep learning is a data-driven approach. The network learns the patterns from the training data, thus it could not accurately infer when the totally unseen test data are provided and thus, the generalization accuracy would decrease. In our experiment setting, the test period contains the highly volatile movement of the stock market. It is notable that S&P500 hit a fresh 12-year low on March 9, 2009. Recently, an online-learning algorithm was applied to catch-up with the time-varying market information, which means that the model will be trained with the test data once there is first an inference with the test data (Hendricks, 2017; Koyano & Ikeda, 2017; Soulas & Shasha, 2013).

5.2. Performance comparison between ModAugNet-c and the other initiatives

Although it is difficult to compare the model performances when the input variables and training periods vary, we suggested the comparison of model performance between ModAugNet-c and the other two initiatives mentioned in the introduction to show the competitiveness of our model in prediction. According to the paper of Bao et al. (2017), we computed the annual test MAPE to predict the next day's S&P500 closing price and summarized the figures in decimal form in line with the results of WSAEs-LSTM(*) in Table 14 (MAPE values were shown in percentage form except for Table 14). Note that the figures of WSAEs-LSTM (*) and Table 15 are taken from Bao et al. (2017). In Table 14, we could observe that ModAugNet-c yielded lower test MAPE in comparison to WSAEs-LSTM.

Wang and Wang (2015) proposed PCA-based stock market index forecasting model (PCA-STNN). The S&P500 forecasting test errors for the period from September 22, 2011 to August 31, 2012 are computed and summarized in Table 16. Note that the root mean squared error (RMSE) is calculated based on the MSE score. They suggested MAPE in percentage form and we considered what they wrote as the test errors of PCA-STNN from the paper. Although the input variables, training periods, and the model architectures are different, our model showed lower test errors among all the error measures.

5.3. Trading simulation

We performed a simple trading simulation to validate how our system would be used. The trading was performed based on the output of ModAugNet-c and the results were compared with the buy and hold (B&H) strategy.

Table 17 indicates our trading strategy. First, we calculated the expected rate of return on the asset for the day t as follows,

$$r(t) = \frac{\widetilde{Close}(t) - Open(t)}{Open(t)}, \tag{13}$$

where $Open(t)$ is an open price of the day t. $\widetilde{Close}(t)$ is the predicted close price which is the output of the ModAugNet-c, while $Close(t)$ is the realized close price of the day t. Then, we set the threshold level $\kappa(\kappa \geq 0)$ based on which the trading decisions are made as presented in the second column of Table 17. If $r(t)$ is greater than κ, transactions of purchasing a unit of the asset at the open price and selling the unit at the close price are made. If $r(t)$ is less than $-\kappa$, then, the reverse occurs. Transactions do not occur otherwise. The daily profit and loss corresponding to each condition was calculated as indicated in the last column of Table 17.

The SPDR S&P500 trust ETF (SPY) and the Samsung KODEX200 ETF (KODEX) were chosen as trading assets. We acquired the price information of each asset from the Thomson Reuters terminals. The back-test trading simulation was performed over the period from February 1, 2008 to July 26, 2017. We applied the threshold level of 1.35% for both index trading. For both SPY and KODEX, the trading result based on the ModAugNet-c output was better than the trading result with the B&H strategy. For SPY, the return from the B&H strategy was 78.74%, while the return was 96.55% when traded based on the ModAugNet-c output. For KODEX, the return from the B&H strategy was 52.70%, while the return was 63.85% when traded based on the ModAugNet-c output. We set an

Table 14
Comparison of the test performance in terms of MAPE in decimal form: the figures of WSAEs-LSTM are takean from the original paper of Bao et al. (2017).

	Year 1	Year 2	Year 3	Year 4	Year 5	Year 6	Average
ModAugNet-c	0.009	0.008	0.006	0.005	0.007	0.007	0.007
WSAEs-LSTM(*)	0.012	0.014	0.010	0.008	0.011	0.010	0.011

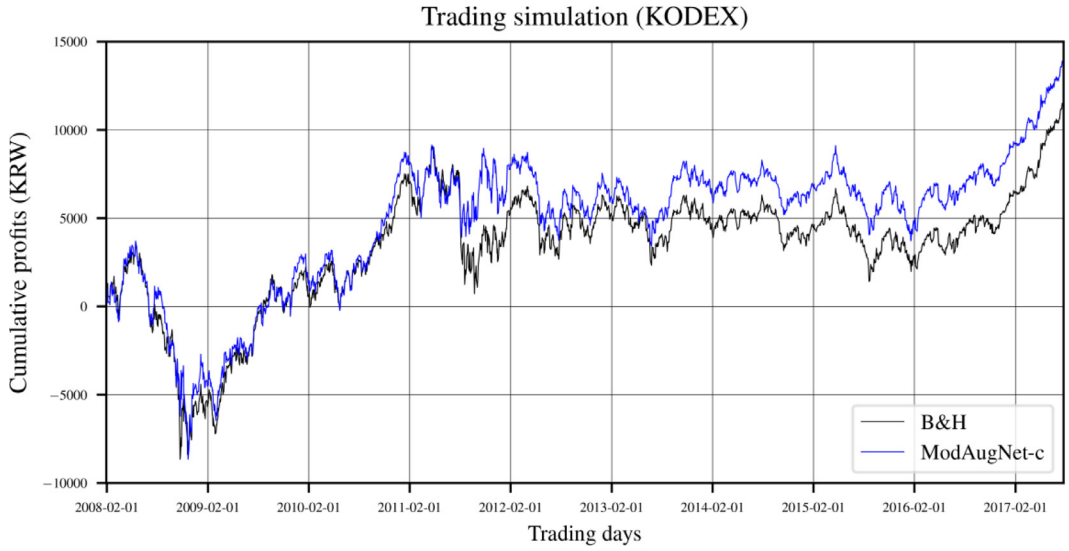

Fig. 22. Cumulative profits for a unit investment in KODEX for the test period.

Table 15
Time interval of the six prediction years: from Bao et al. (2017).

Year	Time interval
Year 1	October 1, 2010 – September 30, 2011
Year 2	October 1, 2011 – September 30, 2012
Year 3	October 1, 2012 – September 30, 2013
Year 4	October 1, 2013 – September 30, 2014
Year 5	October 1, 2014 – September 30, 2015
Year 6	October 1, 2015 – September 30, 2016

Table 16
Comparison of the test performance in terms of MAE, RMSE, and MAPE (in the percentage form): the figures of PCA-STNN are taken from the original paper of Wang and Wang (2015).

	ModAugNet-c	PCA-STNN(**)
MAE	10.8262	15.5181
RMSE	14.6061	19.2467
MAPE (%)	0.8568	1.1872

Table 17
Trading strategy using the predicted output of ModAugNet-c.

Conditions	Transactions	Daily profit/loss
$r(t) \geq \kappa$	Buy → Sell	$Close(t) - Open(t)$
$r(t) < -\kappa$	Sell → Buy	$Open(t) - Close(t)$
$-\kappa \leq r(t) < \kappa$	Hold	0

amount of initial investment as a unit of SPY(KODEX). Daily cumulative profit was calculated based on daily capital gain and daily profit/loss from the transactions presented in Table 17. The cumulative profits for both indices were shown in Figs. 21 and 22.

We validated that our system could be the starting point for developing the DNN based trading system by showing positive model-based profits. However, in our setting, we only considered the price information of the index and its components as input variables to make the network's output closer to the next day's close price. It is necessary to include trading signals as input variables since they provide more relevant information. Also, we did not include real-world constraints such as transaction cost and short-sale constraints. Since it is important to consider such conditions when making trading decisions, the trading systems should be elaborated under real-life financial environments.

6. Conclusions

In recent years, deep learning-based methods have been adopted to solve financial related problems because of the advantage that machine learning techniques, particularly deep learning based methods, have such as fewer assumptions, and as they can solve complex problems based on data. As deep neural networks are a data-driven approach, unlike traditional machine learning algorithms, performance improves as data increases (L'heureux, Grolinger, Elyamany, & Capretz, 2017). If there is sufficient data, deep neural networks are more likely to improve accuracy as they are larger and deeper, while large networks cannot be used due to overfitting if data is insufficient. Therefore, in the financial time-series analyses using DNNs, it is important to overcome the shortage of limited data available to avoid overfitting. One of the frequently recommended alternatives is data augmentation; however, it is not desirable to create a time-series by transforming the original data and to use it for forecasting as they are artificial. In this paper, we propose a novel augmentation approach for financial time-series forecasting to make the model robust to overfitting without the need to generate an artificial time-series to augment available training data.

ModAugNet-c was implemented in three steps; (1) build a modular network which could separately take the main information (fed into the Prediction Module) and relevant information (fed into the Prevention Module), (2) select 10 relevant features as input

candidates, and (3) choose one of the combinations of 10 companies, taking 5 at a time, every 200 epochs and feed them into the Prevention Module for the duration of the subsequent 200 epochs, while the stock market index data are consistently fed into the Prediction Module.

The proposed modular forecasting framework for financial time-series, ModAugNet-c, is then tested by applying it to S&P500 and KOSPI200 data. We observed two remarkable results. First, it is shown that our approach is effective in improving the model performance after being trained by an augmented dataset: the test MSE, MAPE, and MAE for S&P500 decreased to 54.1%, 35.5%, and 32.7% respectively, of the corresponding S&P500 forecasting errors of SingleNet. In addition, the test MSE, MAPE, and MAE for KOSPI200 decreased to 48%, 23.9%, and 32.7%, respectively, of the corresponding KOSPI200 forecasting errors of SingleNet. Second, we found that the prediction is made only through the Prediction Module, while the Prevention Module is no longer involved when training is completed. This is investigated by feeding noised test data into each module. During training, input data fed into this Prevention Module worked as injected noise, which could make the network concentrate on learning important patterns from the Prediction Module.

In the paper, we leveraged the data augmentation approach by introducing modular network architecture to use only a combination of existing data, thus we did not consider artificial training data by distorting the original data as suggested in image data augmentation (Chatfield et al., 2014; Krizhevsky et al., 2012; Salamon & Bello, 2017; Perez & Wang, 2017; Zeiler & Fergus, 2014). The proposed model, ModAugNet-c, is a new framework that combines a Prediction Module participating in actual prediction and a Prevention Module serving as an internal data augmentation tool. Therefore, stock market index prediction is an example, and by modifying the definitions of input variables and output of each module, the same framework as the proposed method can be applied to various deep learning based financial problems, including financial market movement classification, volatility prediction, portfolio optimization, and option pricing. Furthermore, the proposed framework can not only be extended in the financial field, but also in the medical field.

This study considered only financial quantified time-series; however, various studies revealed that stock market movement can be modeled using different types of information, such as investors' sentiment, news events, and macroeconomic factors (Akita et al., 2016; Li et al., 2017). In addition, the proposed network could be trained for the purposes of trading by adding supplementary classifier or regressor, so that the model can additionally output the direction or the expected return of an asset. We used only the closing price of the stock market index and relevant stocks; however, the use of technical indicators will help in making profitable models. Thus, our future task involves effectively dealing with these multimodal data by designing an optimized multi-modular trading system.

CRediT author statement

Yujin Baek: Writing – Original Draft, Methodology, Formal Analysis, Software, Data Curation, Investigation, Visualization

Ha Young Kim: Conceptualization, Methodology, Formal Analysis, Investigation, Writing – Review & Editing, Supervision, Project Administration, Resources, Funding Acquisition.

Financial support

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (NRF-2017R1C1B5018038). The funders had no role in the study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Conflict of interest

None declared.

References

Abu-Mostafa, Y. S., & Atiya, A. F. (1996). Introduction to financial forecasting. *Applied Intelligence, 6*(3), 205–213.
Adhikari, R. (2015). A mutual association based nonlinear ensemble mechanism for time series forecasting. *Applied Intelligence, 43*(2), 233–250.
Akita, R., Yoshihara, A., Matsubara, T., & Uehara, K. (2016). Deep learning for stock prediction using numerical and textual information. In *Proceedings of the IEEE/ACIS fifteenth international conference on computer and information science (ICIS)* (pp. 1–6). IEEE.
Ariyo, A. A., Adewumi, A. O., & Ayo, C. K. (2014). Stock price prediction using the ARIMA model. In *Proceedings of the UKSim-AMSS sixteenth international conference on computer modelling and simulation (UKSim)* (pp. 106–112). IEEE.
Atsalakis, G. S., & Valavanis, K. P. (2009). Forecasting stock market short-term trends using a neuro-fuzzy based methodology. *Expert Systems with Applications, 36*(7), 10696–10707.
Bachelier, L. (1900). *Théorie de la spéculation*. Gauthier-Villars.
Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS ONE, 12*(7), e0180944.
Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.
Chatfield, K., Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Return of the devil in the details: Delving deep into convolutional nets. arXiv:1405.3531.
Chen, A. S., Leung, M. T., & Daouk, H. (2003). Application of neural networks to an emerging financial market: Forecasting and trading the Taiwan Stock Index. *Computers & Operations Research, 30*(6), 901–923.
Chen, K., Zhou, Y., & Dai, F. (2015). A LSTM-based method for stock returns prediction: A case study of China stock market. In *Proceedings of the IEEE international conference on big data (Big Data)* (pp. 2823–2824). IEEE.
Chiang, W. C., Enke, D., Wu, T., & Wang, R. (2016). An adaptive stock index trading decision support system. *Expert Systems with Applications, 59*, 195–207.
Chollet, F., et al.. *Keras* https://github.com/keras-team/keras.
Chong, E., Han, C., & Park, F. C. (2017). Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications, 83*, 187–205.
Cootner, P. H. (1964). *The random character of stock market prices*. Cambridge, MA: MIT Press.
Diebold, F. X., & Mariano, R. S. (2002). Comparing predictive accuracy. *Journal of Business & Economic Statistics, 20*(1), 134–144.
Duda, R. O., Hart, P. E., & Stork, D. G. (2012). *Pattern classification*. John Wiley & Sons.
Fama, E. F. (1965). The behavior of stock-market prices. *The Journal of Business, 38*(1), 34–105.
Fuertes, A. M., Izzeldin, M., & Kalotychou, E. (2009). On forecasting daily stock volatility: The role of intraday information and market conditions. *International Journal of Forecasting, 25*(2), 259–281.
Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning*: 1. Cambridge: MIT press.
Göçken, M., Özçalıcı, M., Boru, A., & Dosdoğru, A. T. (2016). Integrating metaheuristics and artificial neural networks for improved stock price prediction. *Expert Systems with Applications, 44*, 320–331.
Hajizadeh, E., Seifi, A., Zarandi, M. F., & Turksen, I. B. (2012). A hybrid modeling approach for forecasting the volatility of S&P 500 index return. *Expert Systems with Applications, 39*(1), 431–436.
Hendricks, D. (2017). Using real-time cluster configurations of streaming asynchronous features as online state descriptors in financial markets. *Pattern Recognition Letters, 97*, 21–28.
Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6*(02), 107–116.
Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation, 9*(8), 1735–1780.
Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks, 2*(5), 359–366.
Kam, H. J., & Kim, H. Y. (2017). Learning representations for the early detection of sepsis with deep neural networks. *Computers in Biology and Medicine, 89*, 248.
Kim, K. J., & Han, I. (2000). Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. *Expert systems with Applications, 19*(2), 125–132.
Kim, Y., & Enke, D. (2016). Developing a rule change trading system for the futures market using rough set analysis. *Expert Systems with Applications, 59*, 165–173.
Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv:1412.6980.

Koyano, S., & Ikeda, K. (2017). Online portfolio selection based on the posts of winners and losers in stock microblogs. In *Proceedings of the IEEE symposium series on computational intelligence (SSCI)* (pp. 1–4). IEEE.

Kristjanpoller, W., Fadic, A., & Minutolo, M. C. (2014). Volatility forecast using hybrid neural network models. *Expert Systems with Applications, 41*(5), 2437–2442.

Kristjanpoller, W., & Minutolo, M. C. (2016). Forecasting volatility of oil price using an artificial neural network-GARCH model. *Expert Systems with Applications, 65*, 233–241.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings of the advances in neural information processing systems* (pp. 1097–1105).

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436.

Lee, M. C. (2009). Using support vector machine with a hybrid feature selection method to the stock trend prediction. *Expert Systems with Applications, 36*(8), 10896–10904.

Lendasse, A., de Bodt, E., Wertz, V., & Verleysen, M. (2000). Non-linear financial time series forecasting-Application to the Bel 20 stock market index. *European Journal of Economic and Social Systems, 14*(1), 81–91.

L'heureux, A., Grolinger, K., Elyamany, H. F., & Capretz, M. A. (2017). Machine learning with big data: Challenges and approaches. *IEEE Access, 5*, 7776–7797.

Li, J., Bu, H., & Wu, J. (2017). Sentiment-aware stock market prediction: A deep learning method. In *Proceedings of the international conference on service systems and service management (ICSSSM)* (pp. 1–6). IEEE.

Lipton, Z. C., Kale, D. C., Elkan, C., & Wetzel, R. (2015). Learning to diagnose with LSTM recurrent neural networks. arXiv:1511.03677.

Malkiel, B. G., & Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The Journal of Finance, 25*(2), 383–417.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the twenty seventh international conference on machine learning (ICML-10)* (pp. 807–814).

Nelson, D. M., Pereira, A. C., & de Oliveira, R. A. (2017). Stock market's price movement prediction with LSTM neural networks. In *Proceedings of the international joint conference on neural networks (IJCNN)* (pp. 1419–1426). IEEE.

Nowlan, S. J., & Hinton, G. E. (1992). Simplifying neural networks by soft weight-sharing. *Neural Computation, 4*(4), 473–493.

Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock market index using fusion of machine learning techniques. *Expert Systems with Applications, 42*(4), 2162–2172.

Perez, L., & Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. arXiv preprint arXiv:1712.04621.

Rather, A. M., Agarwal, A., & Sastry, V. N. (2015). Recurrent neural network and a hybrid model for prediction of stock returns. *Expert Systems with Applications, 42*(6), 3234–3241.

Ryu, D. (2015). The information content of trades: An analysis of KOSPI 200 index derivatives. *Journal of Futures Markets, 35*(3), 201–221.

Sak, H., Senior, A., & Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Proceedings of the fifteenth annual conference of the international speech communication association*.

Salamon, J., & Bello, J. P. (2017). Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters, 24*(3), 279–283.

Soulas, E., & Shasha, D. (2013). Online machine learning algorithms for currency exchange prediction. *Online machine learning algorithms for currency exchange prediction*: 31. Computer Science Department in New York University Technical Report.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research, 15*(1), 1929–1958.

Sundermeyer, M., Schlüter, R., & Ney, H. (2012). LSTM neural networks for language modeling. In *Proceedings of the thirteenth annual conference of the international speech communication association*.

Ticknor, J. L. (2013). A Bayesian regularized artificial neural network for stock market forecasting. *Expert Systems with Applications, 40*(14), 5501–5506.

Vanstone, B., & Finnie, G. (2009). An empirical methodology for developing stock-market trading systems using artificial neural networks. *Expert Systems with Applications, 36*(3), 6668–6680.

Virtanen, I., & Yli-Olli, P. (1987). Forecasting stock market prices in a thin security market. *Omega, 15*(2), 145–155.

Wang, J., & Wang, J. (2015). Forecasting stock market indexes using principle component analysis and stochastic time effective neural networks. *Neurocomputing, 156*, 68–78.

Wang, J. H., & Leu, J. Y. (1996). Stock market trend prediction using ARIMA-based neural networks. In *Proceedings of the IEEE international conference on neural networks: 4* (pp. 2160–2165). IEEE.

Wang, J. J., Wang, J. Z., Zhang, Z. G., & Guo, S. P. (2012). Stock index forecasting based on a hybrid model. *Omega, 40*(6), 758–766.

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Proceedings of the European conference on computer vision* (pp. 818–833). Springer.

Zhong, X., & Enke, D. (2017). Forecasting daily stock market return using dimensionality reduction. *Expert Systems with Applications, 67*, 126–139.