Expert Systems with Applications 161 (2020) 113704

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Stock market forecasting with super-high dimensional time-series data using ConvLSTM, trend sampling, and specialized data augmentation

Si Woon Lee [a], Ha Young Kim [b,*]

[a] Department of Artificial Intelligence & Data Science, Ajou University, Worldcupro 206, Yeongtong-gu, Suwon 16499, Republic of Korea
[b] Graduate School of Information, Yonsei University, Yonsei-ro 50, Seodaemun-gu, Seoul 03722, Republic of Korea

ARTICLE INFO

Article history:
Received 30 August 2019
Revised 27 June 2020
Accepted 28 June 2020
Available online 8 July 2020

Keywords:
Stock market index
Deep learning
Overfitting
Mini-batch sampling
Data augmentation
ConvLSTM

ABSTRACT

Forecasting stock market indexes is an important issue for market participants, because even a small improvement in forecast accuracy may lead to better trading decisions than those of other participants. Rising interest in deep learning has led to its application in stock market forecasting. However, it is still challenging to use market-size time-series data to predict composite index prices. In this study, we propose a new stock market forecasting framework, *NuNet*, which can successfully learn high-level features from super-high dimensional time-series data. *NuNet* is an end-to-end integrated neural network framework consisting of two feature extractor modules, a *super-high dimensional market information feature extractor* and a *target index feature extractor*. In addition, we propose a mini-batch sampling technique, *trend sampling*, which probabilistically samples more recent data when training. Furthermore, we propose a novel regularization method, called *column-wise random shuffling*, which is a data augmentation technique that can be applied to convolutional neural networks. The experiments are comprehensively carried out in three aspects for three indexes, namely S&P500, KOSPI200, and FTSE100. The results demonstrate that the proposed model outperforms all baseline models. Specifically, for the S&P500, KOSPI200, and FTSE100, the overall mean squared error of our proposed model *NuNet*(*DA, T*) is 60.79%, 51.29%, and 43.36% lower than that of the baseline model *SingleNet*(*R*), respectively. Moreover, we employ trading simulations with realistic transaction costs. Our proposed model outperforms the buy-and-hold strategy being an average of 2.57 times more profitable in three indexes.

1. Introduction

Forecasting the movement of composite market indexes is a challenging problem from both an academic and practical perspective owing to the complexity of movement. It includes random noise made by various market participants with different points of view, making the movement complex and difficult to predict. In addition, a stock market index is affected by numerous incidents involving individual companies and external incidents, such as political and diplomatic issues. Since the efficient market hypothesis was proposed (Cootner, 1964; Malkiel & Fama, 1970), many other attempts have been made to predict financial market trends and movements (Akita, Yoshihara, Matsubara, & Uehara, 2016; Ballings, Van den Poel, Hespeels, & Gryp, 2015; Bao, Yue, & Rao, 2017; Cavalcante, Brasileiro, Souza, Nobrega, & Oliveira, 2016; Fuertes, Izzeldin, & Kalotychou, 2009). This is because even a little improvement in forecast accuracy may lead to better trading decisions compared to other participants.

Despite considerable research, there are still many challenges in forecasting stock markets. Considering all the data available in a market that helps predict financial markets (market-size data) is too high-dimensional problem, modeling to analyze market-size data is a main challenge. To handle this issue, most research proposes a predictive model based on compressed inputs after feature selection or extraction to reduce the dimension of raw data (Ariyo, Adewumi, & Ayo, 2014; Bao et al., 2017; Barak & Modarres, 2015; Chen, Zhou, & Dai, 2015; Chong, Han, & Park, 2017; Fischer & Krauss, 2018; Nassirtoussi, Aghabozorgi, Wah, & Ngo, 2015). However, such dimension reduction can cause information loss, and it is not easy to extract hand-engineered features from thousands to tens of thousands of dimensions of data such as market-size data. Many recent studies involving high dimensional datasets, in diverse fields including computer vision and natural language processing, use deep neural network (DNN)-based models, which show potential as a large-scale feature extractor for high dimensional data (Devlin, Chang, Lee, & Toutanova, 2018; Gehring, Auli,

* Corresponding author.
E-mail addresses: jwkhlee333@ajou.ac.kr (S.W. Lee), hayoung.kim@yonsei.ac.kr (H.Y. Kim).

https://doi.org/10.1016/j.eswa.2020.113704
0957-4174/

Grangier, Yarats, & Dauphin, 2017; He, Zhang, Ren, & Sun, 2016; Krizhevsky, Sutskever, & Hinton, 2012; Mnih et al., 2015). DNNs were also used to extract features from many variables in financial market, such as technical indicators, multiple company data, and macroeconomic data (Chong et al., 2017; Mo, Wang, & Niu, 2016; Moghaddam et al., 2016; Pradeepkumar & Ravi, 2017). Nevertheless, the number of variables were still constrained, since the model parameters increase as the number of variables increases. This could be solved by having a sufficient dataset. However, the available stock market data is approximately 40 years of time-series, which is 9600 data points as in daily basis. This is insufficient considering that a typical DNN with market-size input data has a large number of parameters that surpasses the number of available data points. This leads to the overfitting problem, resulting in poor generalization performance. Thus, to the best of our knowledge, DNN-based studies have not attempted to use very high dimensional data, such as market-size data, as input without dimension reduction.

The motivation of our research is as follows. First, we believe that a model with minimal pre-processed whole market data would perform better than models using only selected features. More specifically, we hypothesize that utilizing all companies within a stock market would be more beneficial than using a subset of constituents. Approaches that employ hand-crafted features of the market constituents, including financial technical indicators or partial market information, as input data can involve bias of the model constructers' pre-knowledge of market situations. Additionally, variables that can help predict markets may be excluded. Further, creating new features based on econometrics or financial theory is not simple, but, DNNs can learn theoretically unrevealed high-level features related to forecasts from data. That is, if we design an efficient neural network architecture that can handle tens of thousands of dimensional time series without overfitting, then the model could select and learn the necessary information from all the company's information, and these self-learned features would be superior to manually-obtained features. Second, in practice, a model needs to be developed, which can robustly adapt to dynamically changing conditions in the market, such as new companies' listings and changing correlations between market index and individual companies. Previous approaches do not sufficiently reflect such dynamic changes, which may erode forecasting.

Therefore, in this study, we propose a novel DNN framework called *NuNet* that can forecast financial market composite indexes using the numerical time-series data of all market constituents with robustness and extremely simplified pre-processing steps (i.e., without extensive prior knowledge and hand-engineered feature extraction steps) and can be applied easily to diverse markets. Market constituents are publicly-listed companies in the corresponding country's market, including NYSE, NASDAQ, and AMEX for S&P500, and we use six variables: open, high, low, and close price, trading volume (OHLCV), and trading value per company. Our proposed framework is an end-to-end trainable model and is an integrated model of two modules, *target index feature extractor* and *super-high dimensional market information feature extractor* (or simply *market feature extractor*). The *target index feature extractor* learns features from historical time-series of six variables (OHLCV and trading value) of the target index, and the *market feature extractor* learns features from historical time-series of six variables (OHLCV and trading value) of the market constituents (listed companies), and then the features extracted from these two modules are concatenated. Finally, the model predicts the next-day close price of the index based on high-level features learned using these combined features. The proposed model is designed to combine two modules, because the dimensions of the input data used for each module differ greatly. In other words, the dimension of market constituent data is much larger than that of target index data. When the two are used together as input for the model, all variables are treated equally, it is difficult to learn features from the target index data, and target index information may be underestimated. In addition, we propose a data augmentation technique, *random column-wise shuffling,* to prevent the overfitting problem. Furthermore, we propose a mini-batch sampling method, *trend sampling*, to emphasize the recent market trends when the model learns patterns from the training samples, because the market changes dynamically.

For comprehensive experiments, three stock market indexes, the S&P500, KOSPI200, and FTSE100, are used. The indexes are chosen to verify the fairness of the evaluation by measuring the forecasting performance of the markets of various countries, namely, the United States, Korea, and the United Kingdom, respectively. The experiments are designed to empirically show the effectiveness of our proposed method in various cases. First, we investigate the feasibility of our proposed model by comparing its forecasting performance with that of other baseline models. Second, we exploit the efficiency of our proposed *trend sampling* method by comparing the performance between normal random mini-batch sampling. We also investigate the effectiveness of our proposed novel data augmentation technique by analyzing the performance degradation of the model when this technique is not applied. In addition, we train and test our model in various window size of input and number of companies for *market feature extractor* to analyze performance under various conditions. Furthermore, we compare our model performance with other recent studies involving market index prediction to show the superiority of our model. Finally, we employ trading simulation to show our proposed model's potential application to real-time trading in the market.

The remainder of this paper is organized as follows. Section 2 describes recent advances in forecasting stock markets. Section 3 describes background related to our work. Section 4 describes the overall methodology and practical settings related to the experiments. Section 5 presents and discusses the experimental results. Section 6 summarizes and presents conclusions.

2. Literature review and related work

Financial market prediction research has been conducted in various methods including statistical modeling, traditional machine learning (ML), deep learning (DL), and combinations of these approaches. Statistical approaches have been undertaken using time-series models, such as auto regressive integrated moving average (ARIMA) (Ariyo et al., 2014; Barak, Arjmand, & Ortobelli, 2017; Barak & Modarres, 2015; Fuertes et al., 2009; Gorenc Novak & Velušček, 2016; Guo, Zhang, & Tian, 2018; Ho, Damien, Gu, & Konana, 2017; Huang, 2017; Toczydlowska & Peters, 2018). Ariyo et al. (2014) proposed the ARIMA model to predict the Nokia Stock Index and Zenith Bank Index, showing that it is feasible to use ARIMA for short-term prediction. Huang (2017) used generalized dynamic kernel-based predictors to generate trading signals with features extracted from kernel canonical correlation analysis (KCCA). The study showed an increase in performance compared to conventional feature extractions and pure regression models.

Many traditional ML methods for forecasting the stock market have also been studied (Ballings et al., 2015; Barak et al., 2017; Barak & Modarres, 2015; Chen & Hao, 2017; Gorenc Novak & Velušček, 2016; Kara, Boyacioglu, & Baykan, 2011; Lee, 2009). Specifically, Kara et al. (2011) used artificial neural network (ANN) and SVM for Istanbul market prediction using ten technical indicators as an input. Ballings et al. (2015) compared the ensemble methods of random forest (RF), adaptive boosting (AdaBoost), and kernel factory with the single classifier models of neural net-

works, logistic regression, support vector machines (SVM), and k-nearest neighbor (kNN) for long-term stock prediction. They empirically demonstrated that the RF algorithm performed the best, followed by the SVM. Chen and Hao (2017) proposed a novel approach that combined weighted SVM and kNN to predict the movement of the Chinese stock market in the short, medium, and long terms.

Recently, the DL-based approaches have been actively researched to predict the stock market (Al-Jumeily & Hussain, 2015; Atsalakis & Valavanis, 2009; Chen, Leung, & Daouk, 2003; Chen et al., 2015; Gündüz, Yaslan, & Çataltepe, 2018; Nelson, Pereira, & de Oliveira, 2017; Ticknor, 2013; Wang, Zeng, & Chen, 2015; Zeng, Zeng, Choi, & Wang, 2017; Zhang & Wu, 2009). Akita et al. (2016) used both textual (financial news) and numerical (close price of each company) information to predict the stock price movements of 50 companies listed on the Tokyo Stock Exchange using long-short term memory (LSTM) (Hochreiter & Schmidhuber, 1997). They showed that using multiple company data for individual company predictions leads to better results. Kim and Kim (2019) proposed a feature fusion model, a hybrid framework of long-short term memory–convolutional neural network (LSTM–CNN) that combines features learned from both stock chart images and stock price data. The integrated model outperformed the single models in predicting prices of S&P500 ETF.

Hybrid methods, such as statistical models combined with ML or DL, have improved forecasting performance (Hajizadeh, Seifi, Zarandi, & Turksen, 2012; Kim & Won, 2018; Ince & Trafalis, 2017; Li & Tam, 2017; Patel, Shah, Thakkar, & Kotecha, 2015; Rather, Agarwal, & Sastry, 2015). Patel et al. (2015) proposed a fusion model consisting of ANN, Random Forest (RF), and support vector regression (SVR) to predict the Indian stock market index. Rather et al. (2015) proposed hybrid modeling by combining ARIMA, exponential smoothing, and recurrent neural network (RNN) for Indian stock market prediction. Ince and Trafalis (2017) combined independent component analysis (ICA) with SVM for US market index prediction. ICA was used to select important indicators, where SVM was used to predict based on selected features.

Previous studies faced difficulties with high dimensional input data using DNNs. Larger input dimension often leads to increases in the parameters of the model; it causes overfitting when data are insufficient for training, thus degrading model performance (Duda, Hart, & Stork, 2012). In forecasting financial markets, most research deals with high dimensional inputs by applying dimension reduction techniques. Nassirtoussi et al. (2015) developed a multi-layer dimension reduction technique using news headlines for FOREX market predictions: a feature-selection by abstraction of word-hypernyms, feature-weighting based on the sum of positive and negative sentiment scores, and feature-reduction based on maximum optimization. Zhong and Enke (2017) used principal component analysis (PCA), robust fuzzy PCA, and kernel-based PCA for dimension reduction, and they showed that an ANN combined with diverse PCAs has higher accuracy for S&P500 ETF movements forecast. Bao et al. (2017) proposed a DL framework for stock price forecasting, combining wavelet transforms, stacked auto encoders (SAEs), and LSTM. Wavelet transforms and SAEs were applied for dimension reduction.

3. Background

3.1. Long-short term memory

RNNs are used to learn temporal patterns from sequential data, such as time-series data. Basic RNN has the vanishing gradient problem for long input sequences. LSTM was proposed to solve this problem and learn long-term dependency (Hochreiter & Schmidhuber, 1997). The LSTM model adds gates and cell state in addition to a simple RNN model, allowing retention of longer sequence patterns than simple RNN architecture.

Fig. 1 depicts the structure of an LSTM memory block x_t, where h_t represents input data and hidden state at sequential time t and f_t, i_t, and o_t denote the forget gate, input gate, and output gate, respectively. The function σ is a sigmoid function, which is $\sigma(x) = 1/(1+e^{-x})$. Formulas (1)–(6) describe the details of LSTM operations. $\odot$ denotes an element-wise dot product. W and b are weight matrices and bias vectors, respectively. Forget gate layer f_t determines how much previous information to remember, gathered from h_{t-1} combined with x_t. If f_t equals 0, the model completely forgets previous information; if f_t equals 1, the model completely remembers previous information. Input gate layer i_t determines which information to store in the cell state c_t combined with g_t vector. The output layer determines which information to let out, processed by cell state c_t and filtered input o_t.

$$f_t = \sigma(W_{f\_xh}x_t + W_{f\_hh}h_{t-1} + b_{f\_h}), \tag{1}$$

$$i_t = \sigma(W_{i\_xh}x_t + W_{i\_hh}h_{t-1} + b_{i\_h}), \tag{2}$$

$$o_t = \sigma(W_{o\_xh}x_t + W_{o\_hh}h_{t-1} + b_{o\_h}), \tag{3}$$

$$g_t = \tanh(W_{g\_xh}x_t + W_{g\_hh}h_{t-1} + b_{g\_h}), \tag{4}$$

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

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

3.2. ConvLSTM

The ConvLSTM architecture was recently introduced to formulate precipitation nowcasting as a spatiotemporal sequence forecasting problem in which both the input and the prediction target are spatiotemporal sequences (Xingjian et al., 2015). The ConvLSTM model has been used in the fields of video saliency detection (Villegas, Yang, Hong, Lin, & Lee, 2017), traffic accident prediction (Yuan, Zhou, & Yang, 2018), surveillance event detection (Zhou, Zhu, & Zhao, 2017), and text recognition (Wang et al., 2019). ConvLSTM is known to capture spatiotemporal patterns of large-scale sequential datasets.

The architecture of ConvLSTM is described in Fig. 2 and formulas (7)–(12). The function σ is a sigmoid function. $*$ denotes the convolutional operation and $\odot$ denotes an element-wise dot product. The gates i_t, f_t, and o_t, memory cell C_t, hidden state H, weights W,and bias$b$ are all three-dimensional (3D) tensors. A ConvLSTM

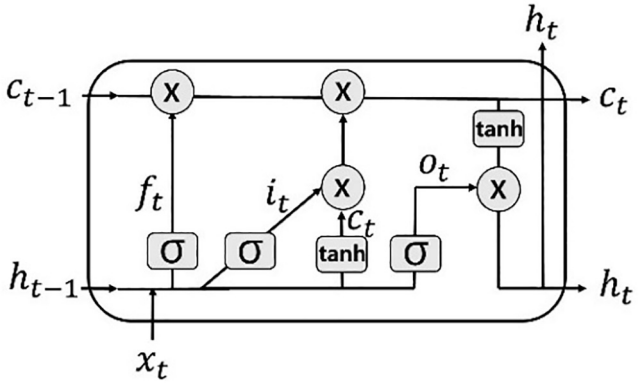

Fig. 1. Structure of an LSTM memory block.

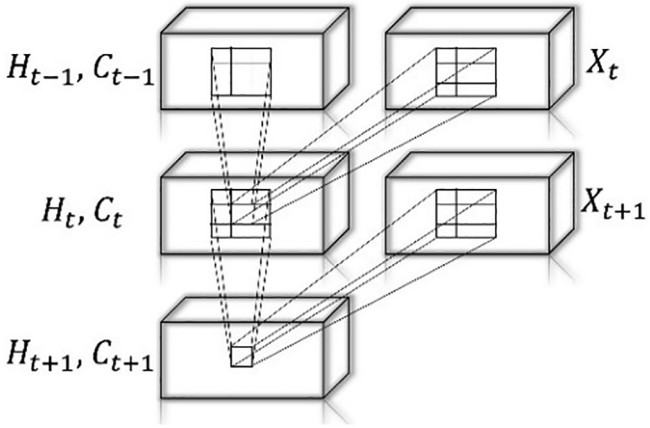

Fig. 2. ConvLSTM architecture.

layer is a recurrent layer, just like the LSTM, but internal matrix multiplications are exchanged with convolution operations. The data flow through the ConvLSTM cells keeps the input dimension 3D instead of being only a one-dimensional (1D) vector. As a result, a ConvLSTM layer uses weight sharing in the same way as CNN does and takes input data as sequential data, which enables the model to process time-series data, like RNN does.

$$i_t = \sigma(W_{i\_xi} * X_t + W_{i\_hi} * H_{t-1} + W_{i\_ci} \odot C_{t-1} + b_i), \tag{7}$$

$$f_t = \sigma(W_{f\_xf} * X_t + W_{f\_hf} * H_{t-1} + W_{f\_cf} \odot C_{t-1} + b_f), \tag{8}$$

$$o_t = \sigma(W_{o\_xo} * X_t + W_{o\_ho} * H_{t-1} + W_{o\_co} \odot C_t + b_o), \tag{9}$$

$$g_t = \tanh(W_{g\_xc} * X_t + W_{g\_hc} * H_{t-1} + b_c), \tag{10}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot g_t, \tag{11}$$

$$H_t = o_t \odot \tanh(C_t), \tag{12}$$

4. Methodology

4.1. Dataset and preprocessing

Table 1 shows the data periods used for our work. This study uses 4000 daily time steps in total, data points before September 13, 2018. For all experiments, the training, validation, and test dataset ratios are 0.7, 0.1, and 0.2, respectively, meaning the time steps are 2800, 400, and 800, respectively. The dataset is divided by aspect of time, a general method to split time series data, which prevents contamination of the dataset. The validation data is used for model selections and finding optimal hyper-parameters such as learning rate and mini-batch size. The test data provides the generalization performance of the model. Three benchmark composite stock price indexes (KOSPI200, S&P500, and FTSE100) and all stocks listed on the largest open market of the corresponding country (Korea, the United States, and the United Kingdom) are used. For S&P500 forecasting, the S&P500 index itself and all stock data listed on the NYSE, NASDAQ, and AMEX are used. Likewise, for KOSPI200, the index itself and all company data listed on the KOSPI and KOSDAQ market are used. Similarly, all stock data listed on the London Stock Exchange and the index itself are used for FTSE100 index prediction.

Table 1
Overall data periods used for each index.

Dataset	Training period	Validation period	Test period
S&P500	Oct. 24, 2002–Jul. 13 2015	Jul. 14, 2015–Feb. 10 2017	Feb. 13, 2017–Sep. 13 2018
KOSPI200	Jul. 22, 2002–Jun. 15 2015	Jun. 16, 2015–Jan. 24 2017	Jan. 25, 2017–Sep. 13 2018
FTSE100	Nov. 14, 2002–Jul. 16 2015	Jul. 17, 2015–Feb. 13 2017	Feb. 14, 2017– Sep. 13 2018

Training a DNN-based model is difficult when the scale of variable differs. Thus, before training, we normalize the data using Min-Max scaling, a common normalization technique. Formula (13) shows the Min-Max scaling of the dataset. $\widehat{X}$ is data after Min-Max scaling, X is data before Min-Max scaling, and X_{max} and X_{min} are the maximum (max) and minimum (min) value of data X, respectively. For all stock values, Min-Max scaling is undertaken variable-wise (e.g., closing price to closing price, opening price to opening price) except trading value. The values X_{max} and X_{min} are the global min and max values of the training dataset. For example, X_{max} and X_{min} values for the S&P500 index are extracted by computing the global min and max of the training period, Oct. 24, 2002 to Jul. 13, 2015. The values X_{max} and X_{min} are used to normalize the training, validation, and test dataset. However, trading value scaling is undertaken time step-wise for all companies, reflecting the weight of the trading value of each company in specific time step.

$$\widehat{X} = \frac{X - X_{min}}{X_{max} - X_{min}}, \widehat{X} \in [0, 1] \tag{13}$$

For a company with historical data of less than 4000 time steps, we employ a zero-padding technique, which makes all data available for 4000 time steps. The zero-padding technique we employ is a simple preprocessing procedure. If a company's stock data have length of t (t < 4000) time steps, then we append length of (4000 – t) zero-matrix to the variable data, which makes a length of 4000 time steps. For data with more than 4000 time steps, we cut off the time steps over 4000, which gives a data length of 4000 time steps. The reason of appending 'zeros' is to provide coherent information to the model that the corresponding company has missing values during that time period. Zero-padding informs the model that the company data is empty, because the company is not listed at that time.

4.2. The proposed model: NuNet

Our stock market index forecasting model, called *NuNet*, is designed to learn sequential patterns of the target index and market constituents. Fig. 3 shows the overall framework of our proposed model. The model is composed of two modules, *market feature extractor* (*super-high dimensional market information feature extractor*) and *target index feature extractor*. As mentioned in the Introduction, our model is separated into two sub-modules owing to the large scale of input data. A single-module architecture is avoided, because it may be difficult to learn features from the target index data, which is the most important data for solving the problem among the market-size data, due to dimension size difference. As shown in Fig. 3, the *market feature extractor* takes last T time steps of all company data at time t as an input, which is $(AC_{t-T+1}, AC_{t-T+2}, AC_{t-T+3}, \cdots, AC_t)$, where AC_t is data of all companies listed in the market at time t. The data is initially fed into two CNN layers, aimed to extract common features from each company variables without consideration of the time steps. The convolutional operation is done variable-wise for each company at each time steps, and the model learns to extract features in units of six variables (OHLCV and trading value). Max-pooling layer extracts core features by applying max-operation to the neighboring two companies' feature maps at a time with sliding window size of two company; thus, it removes some redundant features and takes

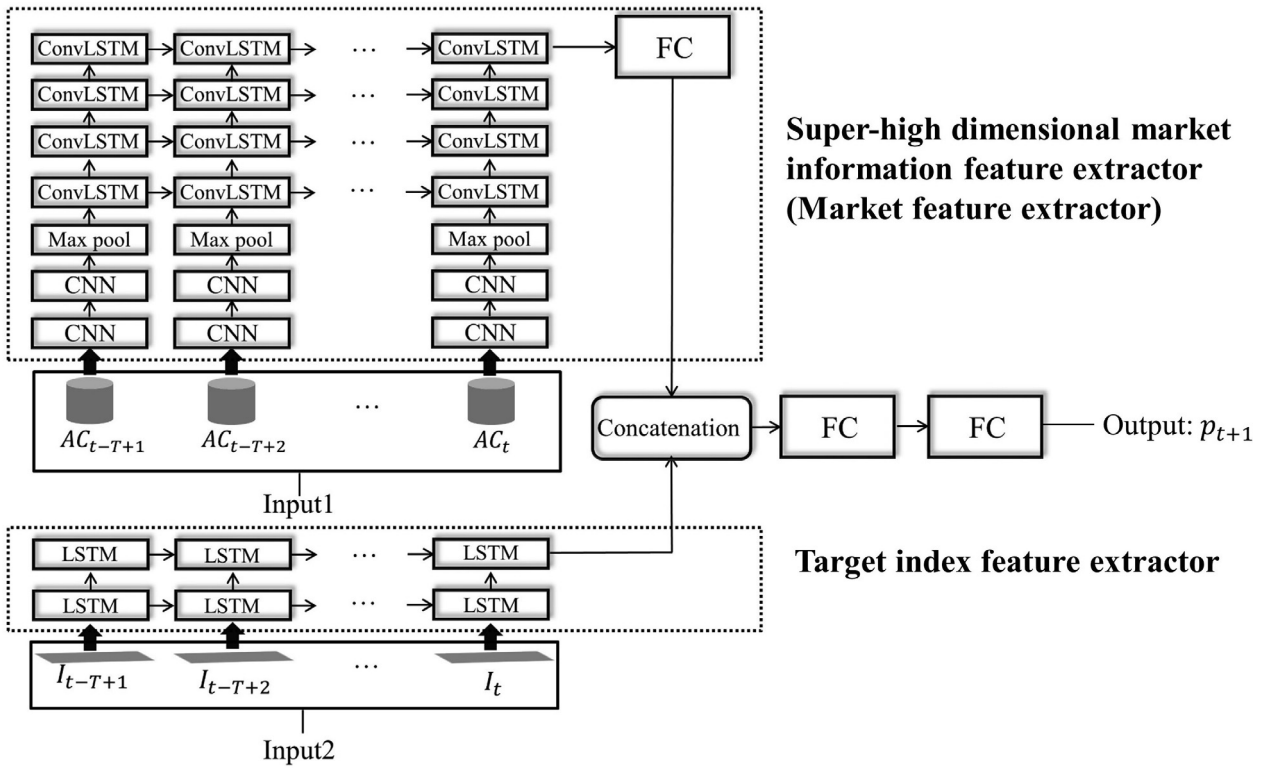

Fig. 3. NuNet model framework.

key features to the next layer. ConvLSTM has four layers, and each layer aims to learn spatiotemporal features of market-size company data through inter-company relationships. The time dependent convolutional operation is done to the fixed number (filter size) of company features at each time steps with fixed sliding window size (stride), summarizing and extracting inter-company feature relationships in aspect of time. The final output of the ConvLSTM is connected to a fully connected (FC) layer to learn high-level features and final representations extracted from market-size data. On the other hand, the last T time steps of the target index data at time t, $(I_{t-T+1}, I_{t-T+2}, I_{t-T+3}, \cdots, I_t)$, where I_t denotes target index data at time t, feed into the *target index feature extractor* module. This module consists of two layers of stacked LSTM to learn temporal patterns of the target index. Finally, the outputs of each module are concatenated for feature fusion and connected to fully connected layers to learn combined features to predict the closing price of target index at time $t + 1$, which is p_{t+1}. The total steps are end-to-end trainable.

4.3. Proposed data augmentation technique: Column-wise random shuffling

Most ML methods may not have insufficient data problems; nevertheless, those models are not suited for learning patterns from high dimensional datasets (Duda et al., 2012). To reduce the input dimension, diverse feature selection and feature extraction methods are usually applied before training. We could apply such methods to DL, but DL learns features directly from the data, and dimension reduction of input may cause information loss, resulting into poor performance. Therefore, more data is required when the input data has very high dimensionality, since it increases the parameters in DNNs. The small amount of training data in DNNs leads to remembering the training data (in other words, learning patterns from noise or outliers), resulting in overfitting. By monitoring whether it is overfitting through a learning curve of training and validation data, it is possible to prevent overfitting with various regularization methods, such as early stopping, dropout, L1, L2 regularization, data augmentation, and reducing network size (Chollet, 2018). However, early stopping can cause underfitting. Reducing the neural network architecture is considered a last method, because the representation capacity of the neural network is also reduced. As in our study, when training data points are significantly smaller than the number of parameters, one of the best ways to prevent overfitting is data augmentation.

Therefore, we propose a novel data augmentation approach that does not corrupt the original input dataset and is specialized for financial time-series input datasets. Our proposed data augmentation method, called *column-wise random shuffling,* randomly shuffles the input company's order per mini-batch sampling. Fig. 4 visualizes an example of *column-wise random shuffling* in CNN and Max-pooling layer of *NuNet* when using four companies (C), sequence length (T) of 5, three CNN filters (N), and six variables (V). The actual CNN layer has two layers; however, the figure assumed that only one CNN layer exists for simplified visualization. The downside of the figure applies *column-wise random shuffling,* and the upside does not. As shown in Fig. 4, the same 6x1 CNN filters are applied variable-wise at each time step to all companies; thus, CNN layer extracts the same features regardless of changes in company orders. CNN layers of the *NuNet* are designed to focus on extracting the common features of each company without considering the time step and the order of companies. However, Max-pooling layer realizes the changes, because it is performed company-wise with filter size 1×2 (stride 1×2). In the Max-pooling layer, data augmentation is achieved by performing pooling on feature maps of two neighboring companies. The model perceives the changes in feature maps from Max-pooling layer which continues down to ConvLSTM and FC layers in *market feature extractor*. As stated in Section 4.2, the ConvLSTM layer is designed to extract features of inter-company relations. The changes of the company feature map orders prevent the model from remember-

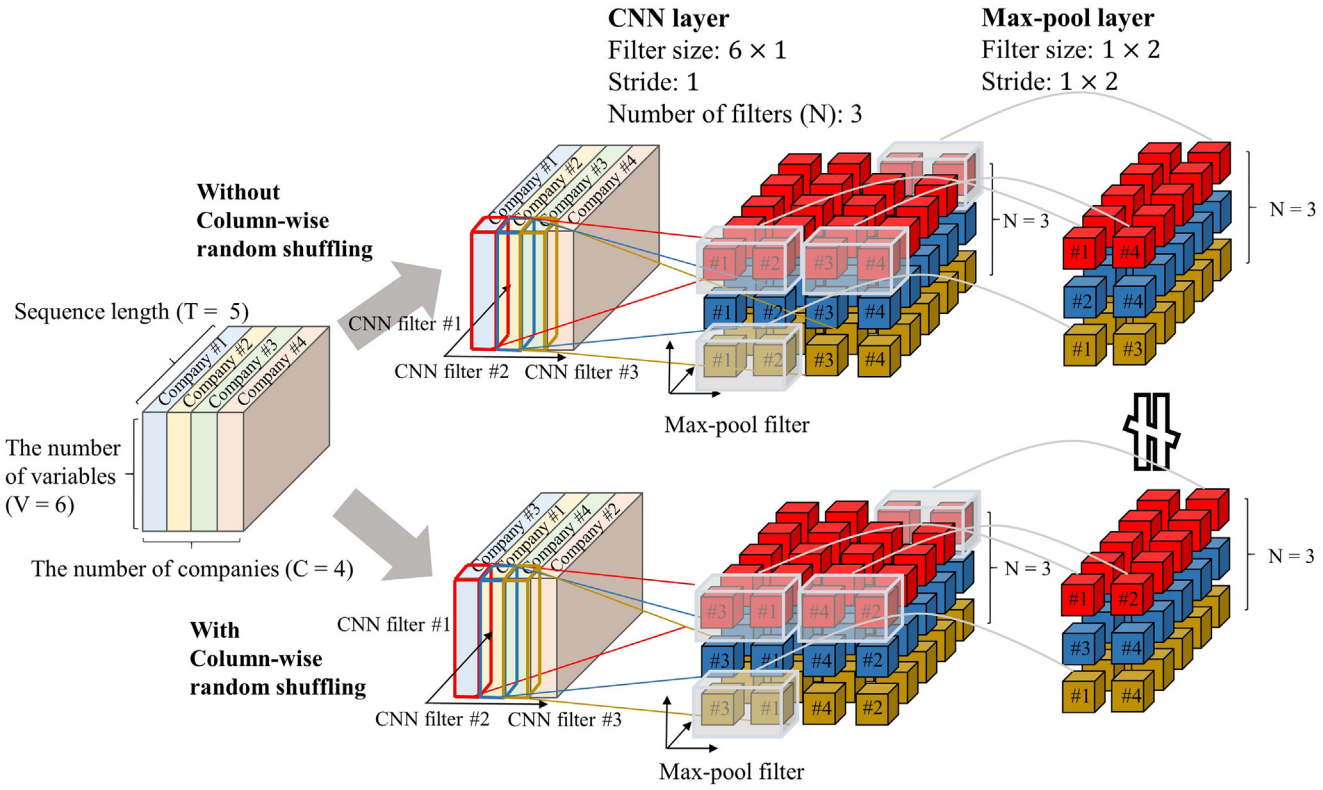

Fig. 4. An example of *column-wise random shuffling* in CNN and Max-pooling layer of *market feature extractor* in *NuNet* with $C = 4$, $V = 6$, $T = 5$.

ing the order of the company features, thereby inducing the module to extract general features of inter-company relationships. For example, if the first layer of the ConvLSTM layer has filter size of $k \times k$, then the layer looks at k^2 company's feature maps at a time and compresses the information to one value. This indicates that the ConvLSTM layer summarizes the inter-company relationships and learns to extract key temporal features that link to the target value. This enables the ConvLSTM layer to cope with changing orders in companies. In summary, the ConvLSTM layer is looking at inter-company relationships and the changes in orders of companies will have more generalization effect rather than confusing the model.

The number of different datasets that can be generated by the *column-wise random shuffling* theoretically is equal to $C!$ (factorial of number of input companies), which is close to infinity. The advantage is that we can generate as many similar datasets as desired, and the disadvantage is that a model with such variation may confuse the model and make the training stage too difficult. Despite this disadvantage, we believe that appropriate architecture of the model (like our proposed *NuNet*) will overcome these shortcomings by demonstrating generalization performance increase.

4.4. Proposed mini-batch sampling method: Trend sampling

When training a DNN model, data samples are assumed independent identically distributed. Sampling the training batch from the sliding window can produce highly correlated data samples, which may cause bias during training. Thus, using sliding window without shuffling the order of training samples is usually not considered generally. In most cases, plain-vanilla sampling method is used to de-correlate the training datasets. Random batch sampling is a random mini-batch sampling method without replacement per epoch, which enables the model to learn patterns robust to historical sequence patterns. This study suggests *trend sampling*, which is a simple extraction method of mini-batch sampling for probabilistically training a model on more recent datasets. Each mini-batch sampling is based on the probability mass function, which is an arithmetic sequence with initial term $\frac{1}{5N}$ and the final term $\frac{40}{N}$, where N refers to number of training time steps. The initial and final terms are defined empirically to keep the sum of all terms equal to 1. Fig. 5 describes the probability mass function of *trend sampling*. The x-axis shows the division of the time step and the mini-batch size, which indicates the number of mini-batch samples in the total training time steps. The y-axis is the sampling probability of the mini-batches. The probability rises as the time step comes close to recent time. We believe this sampling method will enable the model to cope with dynamic changes in the market, resulting into better forecasting performance.

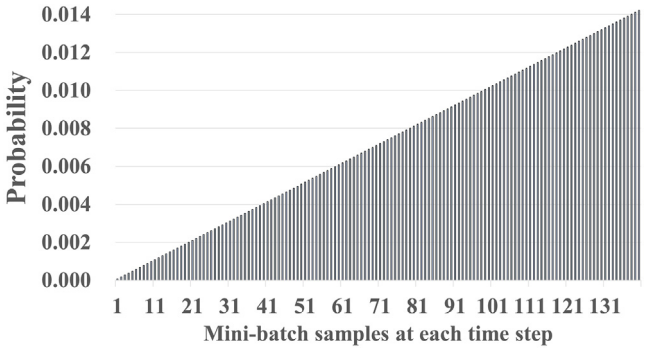

Fig. 5. Probability mass function of *trend sampling*.

4.5. Training & implementation details

Online learning is mainly used when data become available in a sequential order, and it is infeasible to train over the entire dataset or in situations in which the algorithm must adapt to dynamic patterns in the dataset. It is also used when evaluating a time series dataset like stock market data (Bao et al., 2017; Chen & Hao, 2017). Fig. 6 describes our online learning technique. One big block refers to 400 time-steps of data and is equal to the four small blocks. The test dataset is divided into eight small blocks, which indicates that the testing would be done using eight sliding windows. The first block is tested after initial training. Thereafter, the sliding window moves one block ahead, and the training set of the model has access to more training dataset size of the sliding window. At the beginning of each online learning window, we adaptively renormalize the training, validation, and test dataset with new global min and max values of the newly extended training period. We applied this adaptive normalization process to reflect the dynamical and non-stationary characteristics of the market. Training and testing using the sliding window is performed until the model tests the final block. This kind of online learning technique enables the model account for recent trends of market movements and patterns.

Fig. 7 describes our overall training and evaluation steps when online learning, *trend sampling*, and *column-wise random shuffling* are applied. We first build a model and start the training by sampling mini-batches using *trend sampling*, then we shuffle the company orders using *column-wise random shuffling*. Thereafter, the input data feeds to the model for training at the current online learning window until determined epochs are completed. After the training, we evaluate the model on the current test dataset in the window and move to the next online learning window until it reaches the final window.

Practical training details are as follows. The mini-batch size is set to 20, the initial learning rate is set to 0.002, and the time-based decay coefficient of 2×10^{-9} is applied to each iteration for a total 350 epochs. After the initial training window, the online learning rate and decay rate are set to 0.0005 and 3×10^{-8}, respectively for 100 epochs. L2 regularization (Nowlan & Hinton, 1992) of coefficient 10^{-6} is applied to the last fully connected layer of the model. The test model for each sliding window is selected based on the min point of validation loss. Euclidean loss is used for the loss function. For the CNN layers, the rectified linear unit (ReLU) activation function (Nair & Hinton, 2010) is applied. For LSTM and ConvLSTM layers, we use the default activation function setting, which is the hyperbolic tangent and sigmoid activation function as described in Section 3.1 and 3.2. For all the fully connected layers, ReLU is applied, except for the last layer, which has a linear activation function. We apply Adadelta (Zeiler, 2012) for the initial training for 50 epochs and for the rest of the training, we apply Adam (Kingma & Ba, 2014). The codes are available at https://github.com/siwoonlee/NuNet.

The detailed architecture of the *NuNet* model is as follows. The shape of the input data of the *market feature extractor* is (*V, C, T*), while the input data of the *target index feature extractor* takes the

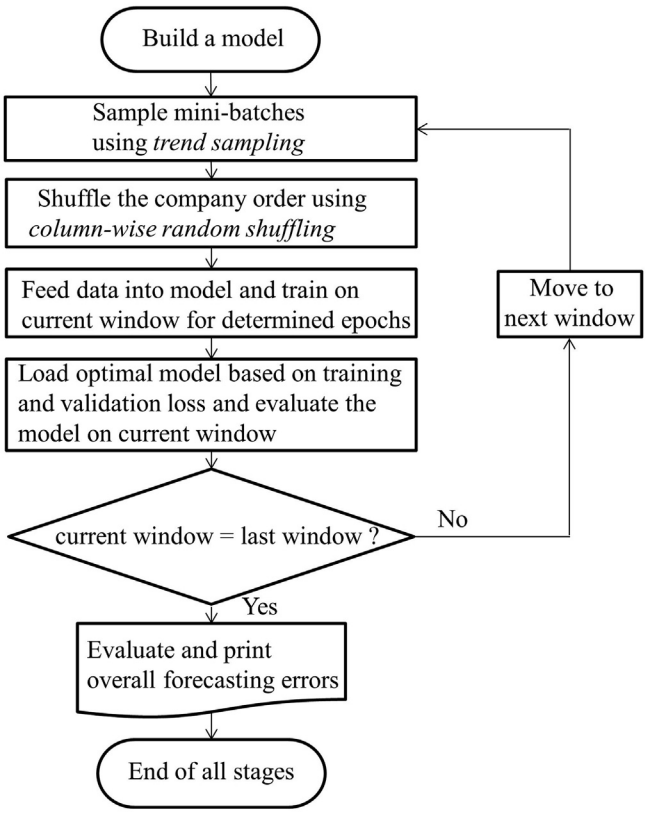

Fig. 7. Overview of training and evaluation steps in flow chart.

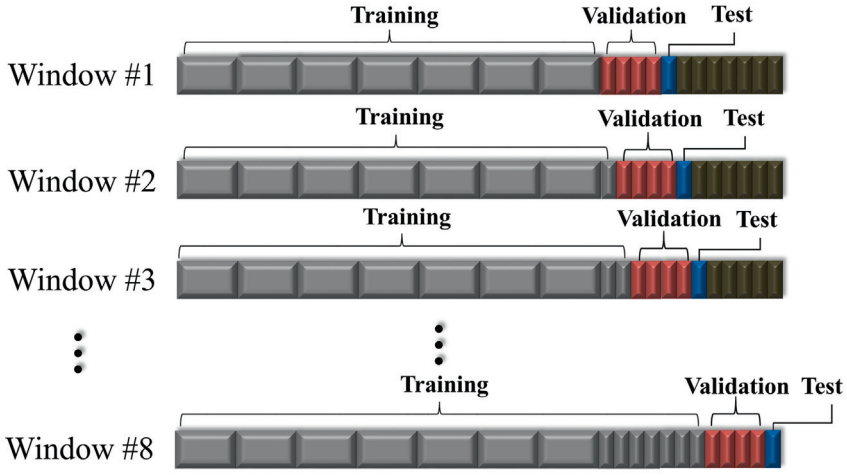

Fig. 6. Online learning technique.

shape of (*V, T*), where *C, V,* and *T* denote the number of companies listed in the corresponding country's open market, number of variables, and sequence length, respectively. For all experiments, the number of variables (*V*) has six dimensions. The length of sequence (*T*) is set to 20. The number of companies listed in the corresponding country's open market (*C*) is the only variable among each experiment: 5790, 2591, and 1259 for the S&P500, KOSPI200, and FTSE100 indexes, respectively. Thus, the *market feature extractor* takes input size of (*6, C, 20*). Since the CNN layer does 1D convolutional operation, the inputs of the first CNN layer are reshaped into $(6, C \times 20)$. The first CNN layer of the *market feature extractor* has output of N = 30 feature maps made by filter size of 6x1 and stride of 1. The second CNN layer also has output of N = 30 feature maps made by filter size of 1x1 and stride of 1. The feature map after the CNN layer has shape of $(N, C \times 20)$. The Max-pooling layer has a pooling filter size of 1×2 and stride of 1×2 with zero-padding if needed, which has output of $(N, C/2 \times 20)$. After the Max-pooling layer, the feature map is reshaped back to $(N, C/2, 20)$, which preserves the time step. The ConvLSTM layer consists of four layers and extracts feature maps of 10, 20, 40, 40, respectively. Each layer has filter size of 3×3, 3×2, 3×1, 2×2 and stride of 3×3, 2×2, 2×2, 2×2, respectively. Thus, each layer has output shape of (10, *C/6*, *N/3*, *T*), (20, *C/12*, *N/6*, *T*), (40, *C/24*, *N/12*, *T*), and (40, *C/48*, *N/24*, *1*), respectively. The model takes only the last sequence of the ConvLSTM layer's last outputs. The last sequence is flattened and is connected to a fully connected layer that consists of 120 nodes. Meanwhile, the hidden state h_t for each LSTM layer is set to 4 for the *target index feature extractor*. The final output of the last sequence of the *target index feature extractor* is concatenated with the output of the *market feature extractor*. The concatenated features are connected to another fully connected layer that consists of 30 nodes connected to the final output layer, which has 1 node of the target price data.

4.6. Measures of forecasting performance

We measure forecasting performance by the terms of three errors, namely, mean squared error (MSE), mean absolute percentage error (MAPE), and mean absolute error (MAE). Note that the MAPE we use in this study is in percentage (%) form. Formulas (14)–(16) indicate the MSE, MAPE, and MAE, respectively, where N refers to the number of observations and g_t and p_t are actual and predicted value, respectively, at time step t. All the errors are calculated based on predicted price and actual price values.

$$MSE = \frac{1}{N}\sum_{t=1}^{N}(g_t - p_t)^2 \tag{14}$$

$$MAPE(\%) = \frac{1}{N}\sum_{t=1}^{N}\left|1 - \frac{p_t}{g_t}\right| \times 100(\%) \tag{15}$$

$$MAE = \frac{1}{N}\sum_{t=1}^{N}|g_t - p_t| \tag{16}$$

4.7. Statistical testing

The Diebold–Mariano (DM) test (Diebold & Mariano, 2002) and Wilcoxon signed rank (WS) test is applied to test the statistical difference between the main models' predictions on the test dataset. The DM test assumes the null hypothesis that there is no difference in prediction error between the two models. The prediction errors of the i th prediction model at time t, are denoted as $e_{i,t}$, defined as the difference between the ground truth value $g_{i,t}$ and the predicted value $p_{i,t}$; thus, $e_{i,t} = g_{i,t} - p_{i,t}$. The equal level of accuracy is $E(a_t) = 0$ under the null hypothesis, where $a_t = l(e_{1,t}) - l(e_{2,t})$ and $l(*)$ is any kind of given loss function. The DM value is calculated as formula (17), where $\bar{a}$ = $\sum_{t=1}^{T}(a_t)$and $\dot{f}_a(0)$ are consistent estimates of mean loss and $f_a(0)$, respectively, $f_a(0)$ being the spectral density of the loss differential at frequency 0. Meanwhile, the WS test is a non-parametric statistical hypothesis test used to compare two related samples to assess whether their population mean ranks differ. Consequently, the WS test can be used to test whether a_t tends to be positive. The assumption is that the observed loss differentials can be tested using the null hypothesis of zero median loss differential. If the exact finite sample tests are available, median(a_t) = 0. If the sampled loss differential has a symmetrical distribution, then the null hypothesis is consistent with the DM test, indicating an equal level of accuracy. The WS statistics are defined as formulas (18) and (19).

$$DM = \frac{\bar{a}}{\sqrt{2\pi\dot{f}_a(0)/T}} \tag{17}$$

$$WS = \sum_{t=1}^{T} I_+(a_t)rank(|a_t|) \tag{18}$$

$$I_+(a_t) = \begin{cases} 1 if a_t > 0 \\ 0 otherwise \end{cases} \tag{19}$$

For both the DM and WS statistical analysis, we make conclusions based on the p-values. We reject the null hypothesis and conclude that the difference in prediction accuracy between the two models is significant if the p-value is less than 0.05.

5. Experimental results and discussion

The forecasting experiments are performed on the market composite indexes of the S&P500, KOSPI200, and FTSE100. Owing to limitations of obtaining a dataset of delisted companies for the whole experiment period, the number of input companies rises, which may reduce the model prediction accuracy. Fig. 8 shows the number of companies as input data of the *market feature extractor*. Original values are plotted on the left-hand side and Min-Max scaled values are plotted on the right-hand side. To solve this problem, we employ online learning as a base learning framework for all experiments, which enables the model to access and learn temporal patterns from relatively recent datasets.

The experiments are carried out in three steps. The first experiment is performed to check feasibility of our proposed model, *NuNet,* by comparing the performance with baseline models. Second, we measure the efficiency of our *trend sampling* method by comparing it with a vanilla random sampling method. In third experiment, we examine the effect of the proposed data augmentation through forecasting error comparison with models without applying our data augmentation approach. For all the experiments, models that employ random sampling are denoted as *R*, and those that employ *trend sampling* as *T*. The model that employs *column-wise random shuffling* is denoted as *DA* and the model that does not as *NDA*.

5.1. Verifying effectiveness of NuNet

5.1.1. Comparison with baseline models

In this subsection, we compare the performance of forecasting results of the S&P500 index using four models—*CNN, LSTM, SingleNet*, and *NuNet*-to test the feasibility of our proposed model. The models are designed from a general perspective using the most basic neural network architecture that we can apply to our task. The three models of *CNN, LSTM,* and *SingleNet* are a modified struc-

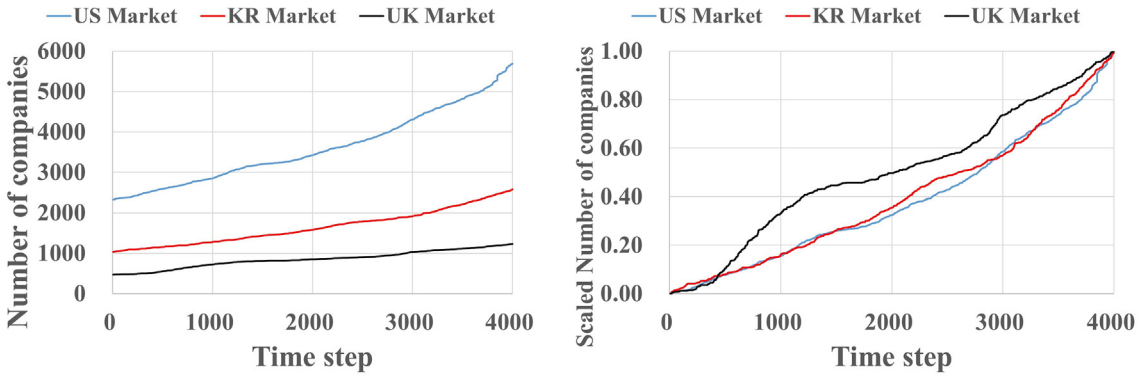

Fig. 8. Number of companies as input data of market feature extractor.

ture of *NuNet*. The name *SingleNet* is adopted from Baek and Kim (2018), as the model has a similar architecture and plays the same rule in this study. *SingleNet* is constructed by removing the total *market feature extractor* of *NuNet* and modifying the structure of the *target index feature extractor*. It is a two-layer stacked LSTM with four dimensions of hidden state, in which the output of the hidden state is connected to a fully connected layer with four nodes that are connected to one node (output). This model takes only the target index as input. The *SingleNet* model is used as a baseline model, and we assume that if our model has better forecast performance than *SingleNet*, then our model succeeds in extracting successful features from large-scale sequential datasets. If the model performs worse than *SingleNet*, we assume that it fails to learn from the large-scale input. The *CNN* model is developed by removing the ConvLSTM layer of the *market feature extractor*, which makes the model directly connected to the fully connected layer after the Max-pooling layer. The *LSTM* model is developed by removing the CNN layer and replacing the ConvLSTM layer with the LSTM layer, which has 30 dimensions of hidden state. While the *CNN* model does not consider the input data as sequential datasets, the *LSTM* model does, but it is essentially a fully connected network that may fail to extract robust features from large-scale sequential datasets.

For this experiment, the containment of the effect of the sampling method is undertaken to observe only the changes in the model forecasting performance when the architecture differs. All the training is performed using a vanilla random sampling and *column-wise random shuffling*. Table 2 shows the overall forecasting results on S&P500 and the number of parameters of four baseline models. As shown in Table 2, the number of parameters of the *CNN* and *LSTM* model are 164.8 and 3.3 times larger than that of *NuNet*, respectively, while the number of parameters of *SingleNet* is 3,623 times smaller than that of *NuNet*. Comparing the overall forecasting results of the four models, the *LSTM* model performs the worst, followed closely by the *CNN* model. Both the *CNN* and *LSTM* models fail to outperform the baseline model *SingleNet*, indicating that the architecture of these models is not adequate for the task. The proposed *NuNet* model with the *column-wise random shuffling* outperforms the baseline *SingleNet* model, which shows that the model architecture of *NuNet* can learn the robust features from large scale financial time-series datasets.

5.1.2. Measuring trend sampling effect

In this subsection, we perform comparative experiments to analyze the effect of our proposed sampling method, *trend sampling*. The experiments are performed on the three indexes of S&P500, KOSPI200, and FTSE100. We compare the forecasting results of the *SingleNet* and *NuNet* models, both trained using random sampling and *trend sampling*. The *NuNet* models are employed with the *column-wise random shuffling*. In addition, we compare the simple moving average (SMA) prediction model, a most simple and basic prediction model, to examine the model performance with respect to historical average. The SMA model uses the average closing price of the last 20 time steps to predict the next closing price.

Tables 3–5 report the closing price prediction results for the S&P500, KOSPI200, and FTSE100 for all online learning test windows and the overall forecasting errors (the errors during the total test period) for each model. To present the predictive power of the proposed model more accurately when capturing market dynamics, Figs. 9–11 visualize the actual returns against the forecasted returns by converting the predicted prices to daily returns at each time step for all models. Since we calculate the predicted return based on the predicted price, the figures demonstrate that the lower the predicted price error, the lower the predicted return error. The *x*-axis represents time steps, and the *y*-axis represents daily returns (left side) and the cumulative squared error (SE) of the predicted returns at each time step (right side). The blue curve shows the cumulative SE at each time step to provide an easy performance comparison between the models. We should remark that we plot Figs. 9–11 based on the returns only to improve the visualization of the performance; however, we analyze all forecasting results using the price values. In all three indexes, the overall MSE, MAPE, and MAE of *SingleNet*(*T*) are 1.38, 1.10, 1.12 times lower on average than *SingleNet*(*R*), respectively, and those of *NuNet*(*DA, T*) are averagely 1.49, 1.22, 1.32 times lower than *NuNet*(*DA, R*), respectively. The *SMA*(20) model showed significantly high overall errors, which indicates that our baseline and proposed models surpass the historical average. Figs. 9–11 show *SMA*(20) and *SingleNet(R)* significantly fails the prediction, while *SingleNet*(*T*) makes more stable predictions. Likewise, predictions of *NuNet*(*DA, T*) show more stable results than *NuNet*(*DA, R*). These results show that *trend sampling* is effective in stock market forecasting models.

Table 2
Forecasting results and number of parameters of four baseline models on the S&P500 index.

	CNN	LSTM	SingleNet	NuNet
Overall MSE	2187.4	2507.8	860.5	592.1
Number of parameters	205,421,361	4,120,581	345	1,246,561

Table 3
Forecasting results of *SMA*(20), *NuNet*(*DA, T*), *NuNet*(*DA, R*), *SingleNet*(*R*), and *SingleNet*(*T*) for S&P500.

Model	Metric	Time step								Overall
		1–100	101–200	201–300	301–400	401–500	501–600	601–700	701–800	
SMA(20)	MSE	2196.2	794.9	786.3	609.4	541.5	479.8	3421.6	991.8	1394.2
	MAPE	1.72	1.91	1.03	0.88	0.79	0.77	1.76	0.98	1.23
	MAE	34.4	37.6	21.8	19.5	18.8	19.2	47.5	27.4	28.3
SingleNet(R)	MSE	737.3	794.9	453.9	767.2	461.1	821.5	1483.5	1364.6	860.5
	MAPE	0.92	0.98	0.82	0.76	0.60	0.80	1.04	1.00	0.86
	MAE	18.4	19.1	12.9	15.1	12.5	24.0	32.1	33.5	21.0
SingleNet(T)	MSE	588.6	523.5	249.8	297.4	785.4	780.6	950.5	828.6	625.6
	MAPE	0.87	0.92	0.55	0.63	0.97	0.96	0.99	0.90	0.85
	MAE	17.5	17.5	10.5	11.9	23.7	23.5	28.1	26.1	19.8
NuNet(DA, R)	MSE	436.8	457.7	258.4	170.1	695.5	884.5	994.6	839.0	592.1
	MAPE	0.90	0.84	0.53	0.47	0.74	0.81	0.93	0.96	0.77
	MAE	13.9	13.6	10.1	8.4	19.4	23.1	29.5	27.2	18.2
NuNet(DA, T)	MSE	512.2	466.9	237.7	120.9	114.4	98.3	915.3	233.6	337.4
	MAPE	0.80	0.87	0.51	0.39	0.34	0.28	0.80	0.43	0.55
	MAE	16.1	17.2	10.7	8.7	8.0	6.9	21.4	11.8	12.6

Table 4
Forecasting results of *SMA*(20), *NuNet*(*DA, T*), *NuNet*(*DA, R*), *SingleNet*(*R*), and *SingleNet*(*T*) for KOSPI200.

Model	Metric	Time step								Overall
		1–100	101–200	201–300	301–400	401–500	501–600	601–700	701–800	
SMA(20)	MSE	18.4	25.9	18.0	9.6	11.5	33.8	25.0	33.1	22.5
	MAPE	1.39	1.77	1.54	1.02	1.00	1.59	1.25	1.35	1.36
	MAE	3.6	4.2	3.7	2.6	2.7	4.8	4.1	4.3	3.8
SingleNet(R)	MSE	4.6	5.0	3.1	3.6	4.8	16.7	16.8	7.6	7.8
	MAPE	0.69	0.72	0.51	0.52	0.57	1.12	1.12	0.75	0.75
	MAE	1.7	1.7	1.3	1.3	1.7	3.6	3.6	2.3	2.2
SingleNet(T)	MSE	4.9	4.9	3.5	3.8	3.9	4.5	9.0	6.0	5.1
	MAPE	0.68	0.71	0.51	0.53	0.57	0.56	0.76	0.59	0.61
	MAE	1.6	1.8	1.6	1.5	1.4	1.6	2.3	1.8	1.7
NuNet(DA, R)	MSE	4.9	5.3	3.2	3.7	3.0	4.7	9.0	5.3	4.9
	MAPE	0.71	0.76	0.54	0.55	0.46	0.49	0.73	0.60	0.60
	MAE	1.7	1.8	1.3	1.4	1.3	1.6	2.3	1.8	1.7
NuNet(DA, T)	MSE	3.6	4.1	3.1	3.3	3.1	3.4	5.5	4.2	3.8
	MAPE	0.60	0.61	0.51	0.51	0.42	0.43	0.57	0.52	0.52
	MAE	1.3	1.3	1.3	1.3	1.2	1.2	1.8	1.5	1.4

Table 5
Forecasting results of *SMA*(20), *NuNet*(*DA, T*), *NuNet*(*DA, R*), *SingleNet*(*R*), and *SingleNet*(*T*) for FTSE100.

Model	Metric	Time step								Overall
		1–100	101–200	201–300	301–400	401–500	501–600	601–700	701–800	
SMA(20)	MSE	9446.2	27630.3	17140.1	20873.7	10136.2	6134.5	5403.1	25489.2	14723.4
	MAPE	1.17	2.00	1.77	1.74	1.17	0.81	0.82	1.81	1.37
	MAE	79.6	123.9	106.8	114.5	82.8	60.1	60.5	133.7	93.3
SingleNet(R)	MSE	6695.7	6130.4	4907.2	2724.5	2372.8	2855.3	5782.7	4807.0	4534.5
	MAPE	0.98	1.03	0.68	0.68	0.53	0.58	0.85	0.83	0.77
	MAE	61.0	61.8	44.7	43.7	39.2	43.4	51.3	48.2	49.2
SingleNet(T)	MSE	6811.3	6107.6	4228.1	2243.3	2031.9	2324.6	4487.6	6093.0	4290.9
	MAPE	0.99	1.03	0.74	0.53	0.46	0.53	0.73	0.86	0.73
	MAE	61.8	61.9	47.5	36.5	34.1	39.8	50.6	53.1	48.2
NuNet(DA, R)	MSE	6939.4	6151.0	4258.5	2178.8	1931.9	1622.2	3217.8	3174.8	3684.3
	MAPE	0.99	1.03	0.70	0.50	0.43	0.44	0.60	0.61	0.66
	MAE	61.8	62.1	45.4	35.1	31.7	32.5	43.8	46.7	44.9
NuNet(DA, T)	MSE	3658.5	4335.7	2963.3	1960.2	1326.0	1405.9	2648.7	2248.1	2568.3
	MAPE	0.86	0.91	0.56	0.52	0.41	0.42	0.54	0.53	0.59
	MAE	38.0	46.9	32.5	31.5	26.0	28.0	39.8	37.9	35.0

Tables 6–8 show the p-values of each model for the S&P500, KOSPI200, and FTSE100. The notation * represents p-values below 0.05. The results of DM and WS tests on our proposed model *NuNet*(*DA, T*) showed p-values lower than 0.05 on all the indexes. This indicates that the forecast results of *NuNet*(*DA, T*) model statistically differ with those of other models for all error measures.

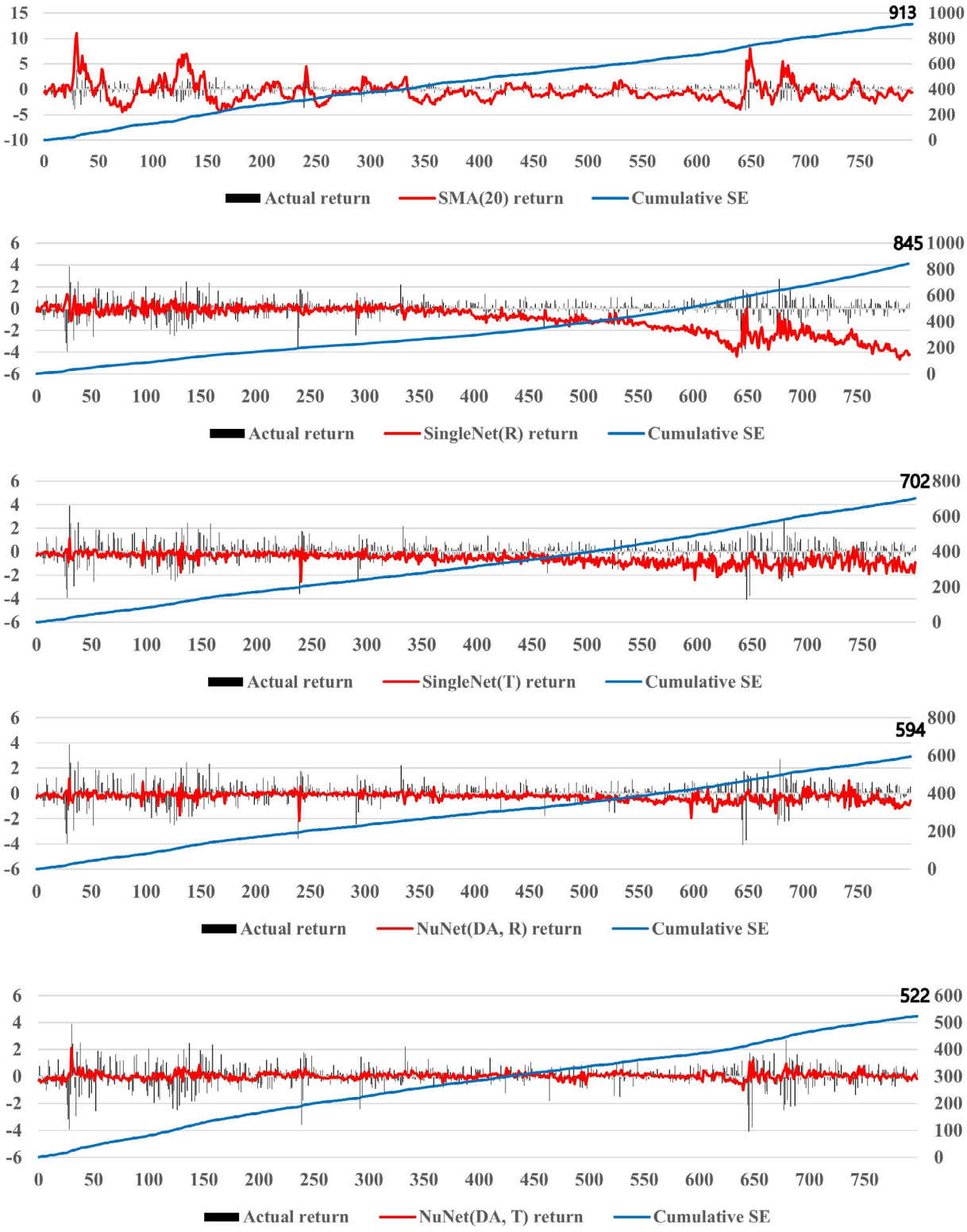

Fig. 9. Comparison of the predicted returns from the *SMA*(20), *SingleNet*(*R*), *SingleNet*(*T*), *NuNet*(*DA, R*), and *NuNet*(*DA, T*) models with the actual returns for S&P500.

5.1.3. Measuring data augmentation effect

In this subsection, we examine the effects of our proposed data augmentation approach, *column-wise random shuffling*. We compare the overall forecasting results of *NDA* and *DA* models using either *trend sampling* (T) or random sampling (R).

Table 9 shows the forecasting results of the S&P500, KOSPI200, FTSE100 for all test periods and the overall forecasting errors of *NuNet*(*NDA, R*) and *NuNet*(*NDA, T*). For all three indexes, the overall MSE of *NuNet*(*NDA, R*) and *NuNet*(*NDA, T*) has increase by average of 8.36 times compared to *NuNet*(*DA, T*). The increase rate is still 5.93 times when compared to *NuNet*(*DA, R*). The errors are significantly higher than those of the baseline model *SingleNet* in Tables 3–5, indicating that the model faces overfitting problems when *column-wise random shuffling* is not applied. We can extrapolate

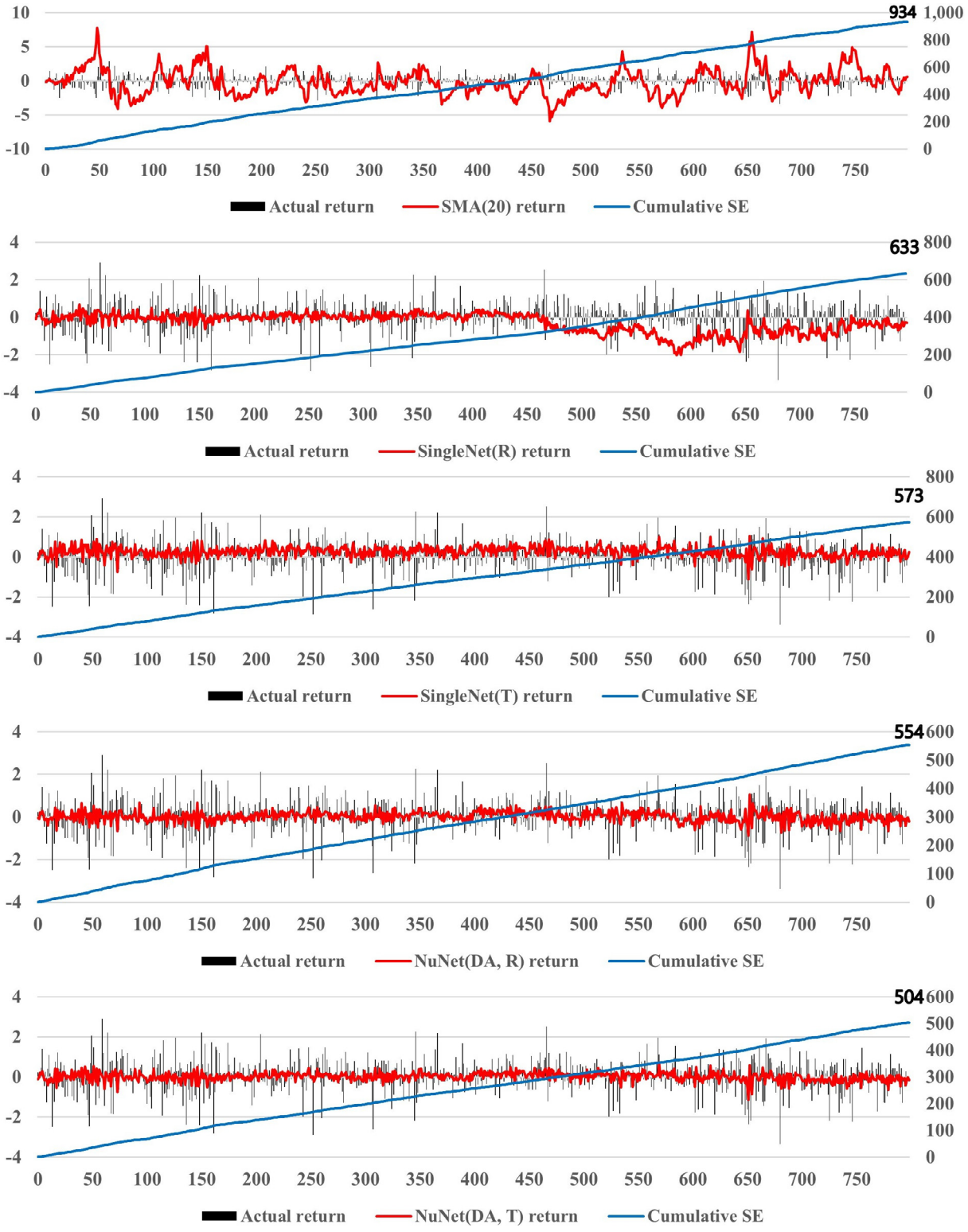

Fig. 10. Comparison of the predicted returns from the *SMA*(20), *SingleNet*(*R*), *SingleNet*(*T*), *NuNet*(*DA, R*), and *NuNet*(*DA, T*) models with the actual returns for KOSPI200.

the *trend sampling* effect when *column-wise random shuffling* is not employed by comparing *NuNet*(*NDA, R*) with *NuNet*(*NDA, T*). The average overall MSE of *NuNet*(*NDA, T*) for the three indexes decreases by 22.13%, which indicates that the *trend sampling* is also effective without *column-wise random shuffling*.

For additional analysis on overfitting, we compare the train and test errors of non-data-augmented models (*NuNet(NDA, R)* and *NuNet(NDA, T)*) and data-augmented models (*NuNet(DA, R)* and *NuNet(DA, T)*) on the S&P500 index. Fig. 12 shows the overall MSE of each model on training and test dataset. In Fig. 12, non-data-augmented models showed much worse forecasting results in test data than data-augmented models. In addition, non-data-augmented models have a very small train error and a large test error, which indicates overfitting. Overall results show that

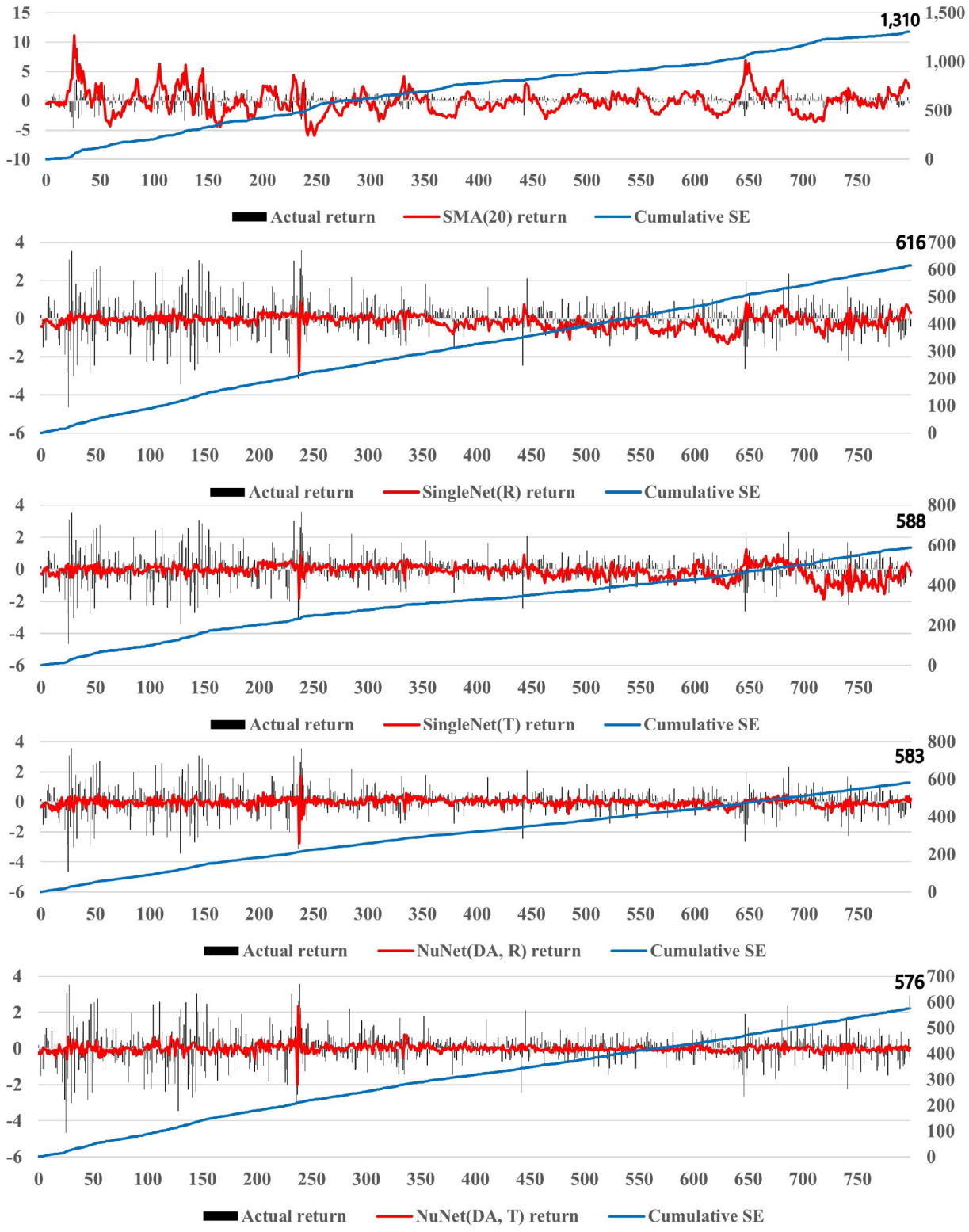

Fig. 11. Comparison of the predicted returns from the *SMA*(20), *SingleNet*(*R*), *SingleNet*(*T*), *NuNet*(*DA, R*), and *NuNet*(*DA, T*) models with the actual returns for FTSE100.

column-wise random shuffling prevents overfitting and increases forecasting performance.

To summarize the results, *NuNet*(*DA, R*) outperforms all *SingleNet* models, which indicates that using market-size data with the architecture of the proposed model helps forecasting. In addition, the model that applied *trend sampling*, *NuNet*(*DA, T*), has the best performance in this study, indicating the effectiveness of our proposed sampling method.

5.1.4. Performance comparison with previous methods

In this subsection, we compare the performance of our proposed model, *NuNet*(*DA, T*), with other recent studies to show the

Table 6
DM and WS test *p*-values of each model for S&P500.

		SMA(20)	SingleNet(R)	SingleNet(T)	NuNet(DA, R)	NuNet(DA, T)
SMA(20)	MSE		0.00*	0.00*	0.00*	0.00*
	MAPE		0.00*	0.00*	0.01*	0.00*
	MAE		0.00*	0.00*	0.00*	0.00*
SingleNet(R)	MSE	0.00*		0.12	0.01*	0.00*
	MAPE	0.00*		0.00*	0.02*	0.00*
	MAE	0.00*		0.43	0.01*	0.01*
SingleNet(T)	MSE	0.00*	0.00*		0.03*	0.01*
	MAPE	0.00*	0.00*		0.00*	0.00*
	MAE	0.00*	0.00*		0.02*	0.01*
NuNet(DA, R)	MSE	0.00*	0.00*	0.00*		0.02*
	MAPE	0.00*	0.00*	0.00*		0.01*
	MAE	0.00*	0.00*	0.00*		0.02*
NuNet(DA, T)	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*	

Notes: The DM test and WS test results are above and below the diagonal, respectively.

Table 7
DM and WS test *p*-values of each model for KOSPI200.

		SMA(20)	SingleNet(R)	SingleNet(T)	NuNet(DA, R)	NuNet(DA, T)
SMA(20)	MSE		0.00*	0.01*	0.00*	0.00*
	MAPE		0.00*	0.00*	0.00*	0.00*
	MAE		0.00*	0.00*	0.00*	0.00*
SingleNet(R)	MSE	0.00*		0.22	0.04*	0.02*
	MAPE	0.00*		0.14	0.03*	0.01*
	MAE	0.00*		0.35	0.03*	0.02*
SingleNet(T)	MSE	0.00*	0.32		0.03*	0.00*
	MAPE	0.00*	0.06		0.00*	0.00*
	MAE	0.00*	0.13		0.02*	0.00*
NuNet(DA, R)	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*
NuNet(DA, T)	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*	

Notes: The DM test and WS test results are above and below the diagonal, respectively.

Table 8
DM and WS test *p*-values of each model for FTSE100.

		SMA(20)	SingleNet(R)	SingleNet(T)	NuNet(DA, R)	NuNet(DA, T)
SMA(20)	MSE		0.00*	0.01*	0.00*	0.00*
	MAPE		0.00*	0.00*	0.00*	0.00*
	MAE		0.00*	0.00*	0.00*	0.00*
SingleNet(R)	MSE	0.00*		0.12	0.25	0.03*
	MAPE	0.00*		0.01*	0.00*	0.00*
	MAE	0.00*		0.43	0.01*	0.01*
SingleNet(T)	MSE	0.00*	0.03*		0.06	0.00*
	MAPE	0.00*	0.00*		0.00*	0.00*
	MAE	0.00*	0.14		0.10	0.00*
NuNet(DA, R)	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*
NuNet(DA, T)	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*	

Notes: The DM test and WS test results are above and below the diagonal, respectively.

superiority of our model. Comparative experiments are performed against two recent leading works in forecasting stock market indexes, namely, the *ModAugNet-c* (Baek & Kim, 2018) and *WSAES-LSTM* (Bao et al., 2017). We re-train and test the model from scratch based on those studies by setting the train, validation, and test periods mentioned therein. The model architecture is not modified. For Baek and Kim (2018), the price information between January 4, 2000 and December 31, 2007 is used as the training

Table 9
Forecasting results (MSE) of models *NuNet(NDA, R)* and *NuNet(NDA, T)* for three indexes (S&P500, KOSPI200, and FTSE100).

Time step	S&P500		KOSPI200		FTSE100	
	NuNet(NDA, R)	NuNet(NDA, T)	NuNet(NDA, R)	NuNet(NDA, T)	NuNet(NDA, R)	NuNet(NDA, T)
1–100	992.0	897.7	4.67	4.66	6070.6	6838.3
101–200	878.2	732.3	5.14	5.02	6125.2	6264.8
201–300	819.5	729.1	3.22	3.23	3474.6	4076.5
301–400	720.9	670.6	3.76	3.75	2568.7	2140.9
401–500	2651.0	1184.5	6.14	3.85	5368.3	1695.9
501–600	5053.7	2798.9	42.22	23.27	7286.7	1566.8
601–700	8012.6	6582.6	32.98	29.47	8954.3	6502.5
701–800	7381.7	7482.2	8.27	6.44	9435.8	9955.8
Overall	3313.7	2634.7	13.30	9.96	6160.5	4880.2

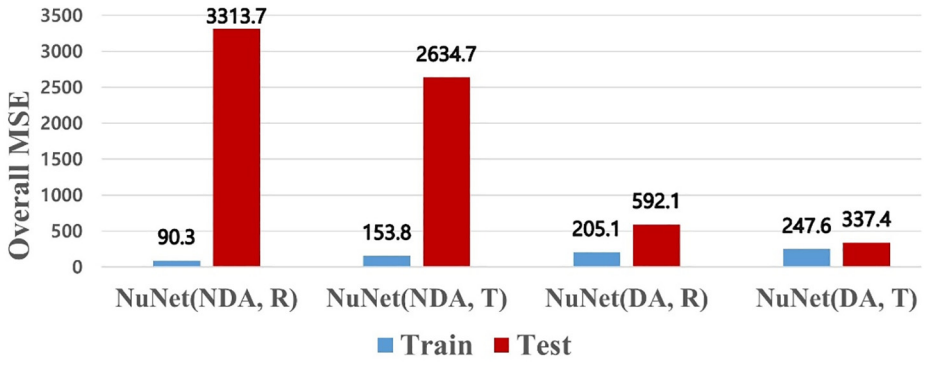

Fig. 12. Comparison of results from train and test data of the *NuNet* with and without data augmentation on S&P500.

dataset, while time steps between January 2, 2008 and July 27, 2017 is used as the test dataset. For Bao et al. (2017), the period between July 1, 2008 and July 1, 2010 is used for training at the initial sliding window. The overall test period is October 1, 2010 to September 30, 2016.

Table 10 shows the overall forecasting errors of *ModAugNet-c* against *NuNet(DA, T)* on two indexes, the S&P500 and KOSPI200. The values of *ModAugNet-c* are from Baek and Kim (2018). All the MSE, MAPE, and MAE of *NuNet*(*DA, T*) outperform that of *ModAugNet-c*. Similarly, Table 11 shows a comparative experiment between *WSAEs-LSTM* from Bao et al. (2017) and *NuNet*(*DA, T*). For all periods, the overall average of MAPE of our proposed model is significantly low. The decrease in forecasting errors compared with both studies shows that our proposed model is superior to comparative works.

5.2. Testing NuNet on various conditions

This section employs additional experiments to determine the effects of various number of companies and sequence lengths. We hypothesize that using all companies within a stock market is more valuable than using a selected group. To investigate our hypothesis, we perform an additional experiment on forecasting S&P500 when inputting company number (*C*) of 200, 400, 600, …, 2000. We assume that the trading value of the companies is an important feature to sort companies (Li, Wang, Havlin, & Stanley, 2011; Shin, 2006). The 5790 companies are sorted in descending order by the mean trading value of total time steps of training period. To verify the proposed model's performance, we experiment from the top 200 to the top 2000 in increments of 200. Fig. 13 shows the results. *NuNet* with C = 5790 shows the best forecasting performance. The forecasting errors of *NuNet*(*DA, T*) in test data tends to decrease as the number of companies increases. Since our model is designed to extract relevant features from large scale of input companies, more information given to the model increases forecasting performance. As a result, we conclude that our proposed model successfully learns to distinguish irrelevant features and important features from market-size high dimensional data for forecasting the market index.

All experiments are performed with sequence length (T) of 20, to fairly compare the forecasting performance to the target study ModAugNet (Baek & Kim, 2018), which used T = 20. Increase in sequence length implies that the model sees more historical data to forecast the market index. The forecasting performance may increase when using sequence length greater than 20 and may capture monthly patterns and seasonal patterns. We employ an additional experiment on forecasting S&P500 using T = 40, 60, 80, … 240 to determine the effect of increase in sequence length. The results are summarized in Fig. 14 containing overall MSE and total training time. Forecasting errors tend to decrease when sequence

Table 10
Comparison of the test performance with ModAugNet-c in terms of MSE, MAPE (%), and MAE.

	S&P500		KOSPI200	
	ModAugNet-c	NuNet(DA, T)	ModAugNet-c	NuNet(DA, T)
MSE	342.48	319.65	7.56	7.06
MAPE	1.0759	0.9896	1.0077	0.9732
MAE	12.058	10.329	1.975	1.735

Notes: The results of *ModAugNet-c* are from the Baek and Kim (2018) study.

Table 11
Comparison of the test performance with WSAEs-LSTM in terms of MAPE.

	Year1	Year2	Year3	Year4	Year5	Year6	Average
WSAEs-LSTM	0.012	0.014	0.010	0.008	0.011	0.010	0.011
NuNet(DA, T)	0.010	0.011	0.008	0.008	0.009	0.007	0.009

Notes: The results of *WSAEs-LSTM* are from the Bao et al. (2017) study.

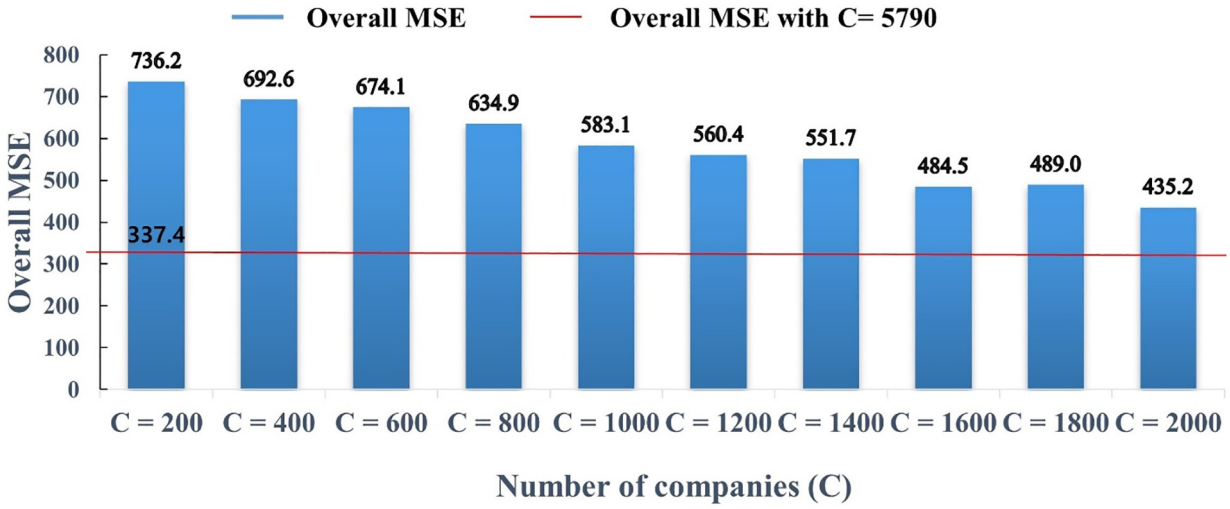

Fig. 13. Forecasting results with *NuNet(DA, T)* on S&P500 by variation of the number of companies.

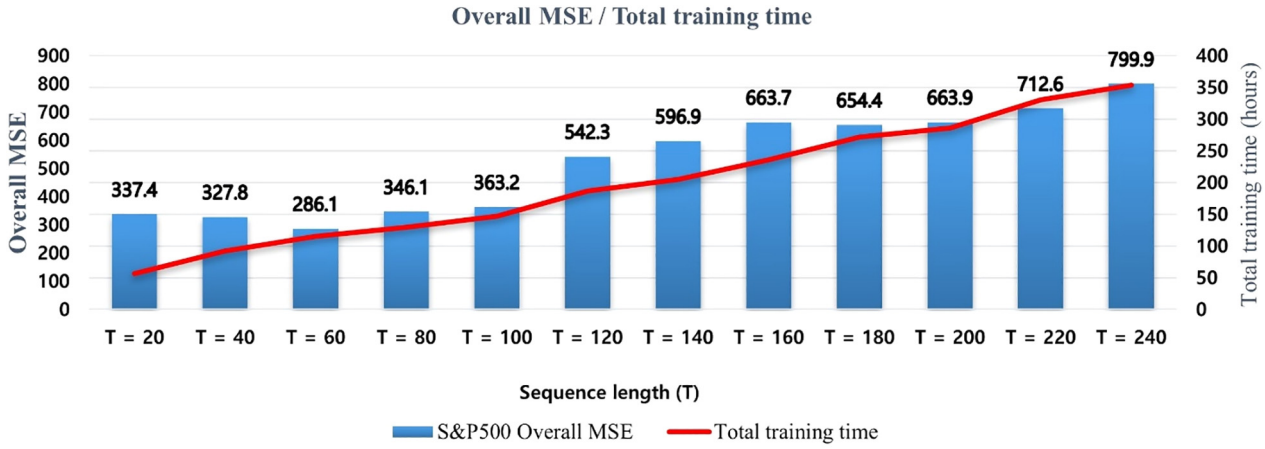

Fig. 14. Forecasting results and training time on S&P500 by variation of sequence length (T).

length is close to $T = 60$ and increase remarkably when $T > 100$. In summary, an increasing sequence length may improve forecasting performance, though not always. As longer sequence lengths contain more historical data, the model should discriminate irrelevant and significant features as the sequence is processed. However, the results show that our model architecture fails to improve forecasting performance when sequence length increases greater than 60. Our model deals with a large scale of features at each time step, which is (5790×6) for S&P500. It is ideal for the model to look at more sequences for prediction; however, the increase in sequence length makes the training stage complex by overwhelming the model with too much information. Our model is initially tuned to relatively smaller sequence lengths. However, we believe that modifying our model architecture to more deep and complex model by adding additional layers and increasing the number of filters of each CNN and ConvLSTM layers may enable it to adapt to larger sequences. Another interpretation of the case about the increase in forecast error when T > 60 may indicate that information older than 60 trading days may not contain useful features, resulting as noise for the model and interfering with learning.

5.3. Trading simulation

To test the feasibility of application in real-time trading, we employ a back-trading test as a profitability test. We assume that the indexes are tradable assets with transaction cost of 0.3% when changing position. The transaction cost assumption includes taxes and commission fees. We apply simple trading simulation by using the following strategy. We first calculate the predicted rate of return, which is the value that we expect according to the forecasting model. Formula (20) shows the calculation of the expected rate

Table 12
Overall cumulative profits (%) of trading simulations with transaction cost on three indexes.

Index	B&H	SingleNet(R)	SingleNet(T)	NuNet(DA, R)	NuNet(DA, T)
S&P500	34.01	3.34	33.67	35.57	69.32
KOSPI200	18.23	15.25	29.49	43.80	61.02
FTSE100	9.88	−1.69	5.72	11.11	22.84

of return using the predicted value of the model. P_t is the closing price at time step t and P_{t+1} is the predicted closing price that the model expects. Thus, r_{t+1} is the expected rate of return. We assume daily-based trading, implying that all positions are closed at end of day, and new positions are made immediately according to the trading signal. However, if the trading signal does not differ from the last signal, we do not make any transactions. Our trading strategy is based on predicted return with a threshold that assumes if $r_{t+1} > threshold$, then we buy (long), and if $r_{t+1} < -threshold$, then we sell (short); otherwise, we make no transition of the position. The threshold we used was 0.15. For cases when the trading signal is the same as the last signal, we assume that no transaction occurs, and the same position is held. For example, if we are already in the buy (short) position and the signal indicates to buy (short) again, we maintain the same position until another signal is predicted. The position changes when the new trading signal differs from the last signal; we close the current position and make a new position according to the new signal. The profits are calculated by the daily trading actions and accumulated. We compare our four models *SingleNet*(*R*), *SingleNet*(*T*), *NuNet*(*DA, R*), and *NuNet*(*DA, T*) against a simple buy-and-hold (B&H) strategy, which uses only a buying strategy for the whole period.

$$r_{t+1} = \frac{P_{t+1} - P_t}{P_t} \tag{20}$$

Table 12 shows the cumulative profits of the B&H strategy and the four models while Fig. 15 shows the cumulative profits (%) by each time step. The overall cumulative profit for B&H strategy, which is the baseline profit we seek to outperform, are 34.01%, 18.23%, and 9.88% for S&P500, KOSPI200, and FTSE100, respectively. Our proposed model, *NuNet*(*DA, R*) and *NuNet*(*DA, T*) outperformed B&H strategy by showing average of 1.52 and 2.57 times more cumulative profits in three indexes, respectively. On the other hand, *SingleNet*(*R*) failed to outperform B&H strategy for all three indexes, while *SingleNet*(*T*) only outperforms B&H strategy

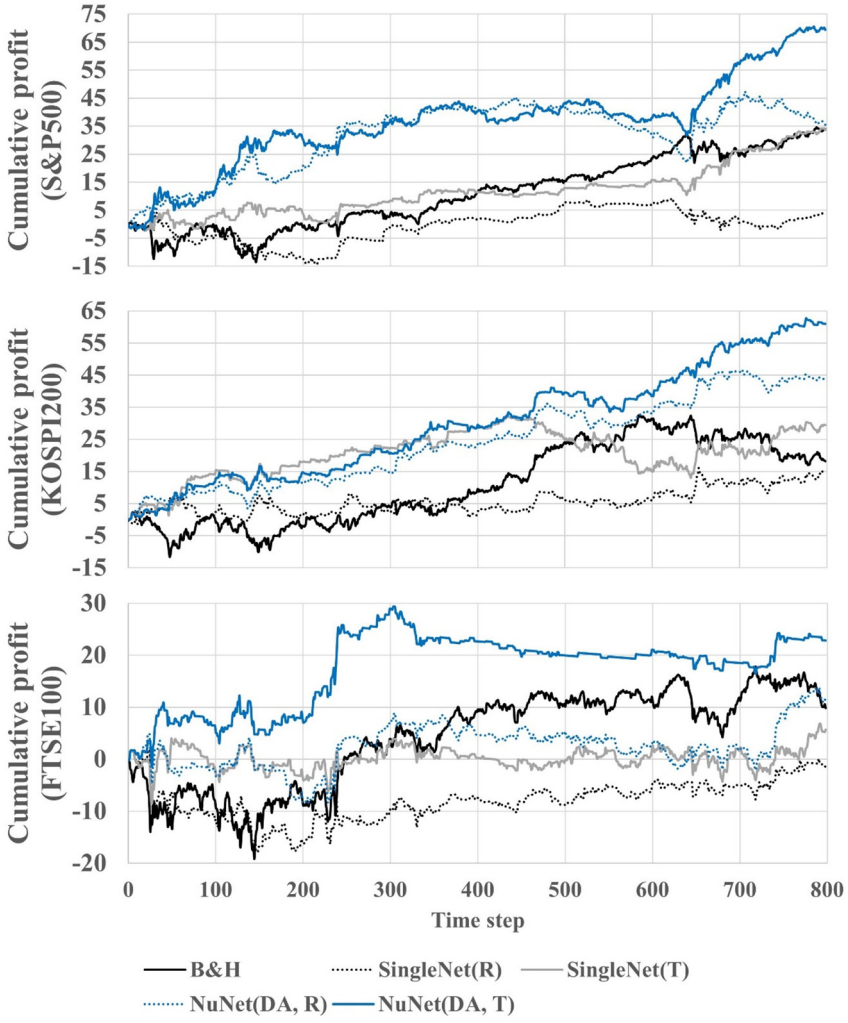

Fig. 15. Cumulative profits of trading simulations with transaction cost on test data of S&P500, KOSPI200, and FTSE100.

Table 13
Computational complexity in FLOPS (floating point operations per second) and inference time in milliseconds.

	SingleNet	NuNet(S&P500)	NuNet(KOSPI200)	NuNet(FTSE100)
FLOPS	510	2,610,088	1,878,548	940,138
Inference time	5	60	45	31

Table 14
Total training time and training time per epoch in seconds.

		SingleNet(R)	SingleNet(T)	NuNet(NDA, R)	NuNet(NDA, T)	NuNet(DA, R)	NuNet(DA, T)
S&P500	Per epoch	1.44	1.51	96.13	101.56	105.11	112.21
	Total time	2592	2718	173,034	182,808	189,198	201,978
KOSPI200	Per epoch	1.43	1.50	86.72	91.08	94.85	101.03
	Total time	2574	2700	156,096	163,944	170,730	181,854
FTSE100	Per epoch	1.43	1.50	44.02	46.26	48.01	51.33
	Total time	2574	2700	79,236	83,268	86,418	92,340

in the KOSPI200 index by 1.62 times more profitable. The results also show effectiveness of *trend sampling*. For all three indexes, the cumulative profit has increased between *SingleNet*(*R*) to *SingleNet*(*T*) and *NuNet*(*DA, R*) to *NuNet*(*DA, T*). These results demonstrate that *trend sampling* increases profitability.

We also perform DM and WS tests on historical average return to test the statistical difference between the main models' trading simulations. All models passed the test with significantly low p-values, below 10^{-5}, on MSE, MAPE, and MAE. This indicates that trading strategies are significantly different between baseline models.

5.4. Computational complexity

We measure the FLOPS (floating point operations per second), inference time, total training time, and training time per epoch for each model to see the computational complexity of our method. The equipment used was Intel® Core™ i7-6850 K CPU @ 3.60 GHz × 12, GeForce GTX 1080Ti. Tables 13 and 14 summarize the results. The proposed model, *NuNet*, shows an average of 3548 times larger FLOPS and 9.07 times slower inference time than the baseline model, *SingleNet*. Considering that the model makes daily predictions (FLOPS and inference time is not critical), the trade-off between forecasting errors and computational cost seems reasonable. The total training time for *SingleNet* and *NuNet* takes an average of 2580 and 146,742 s, which is approximately 43 min and 41 h, respectively. The training time has increased by average of 0.07 and 4.77 s per epoch in *SingleNet* and *NuNet*, respectively, when applying *trend sampling*. In addition, training time has increased by average of 7.80 s per epoch when *column-wise random shuffling* was applied. The training time per epoch has increased by average of 12.56 s when applying both *trend sampling* and *column-wise random shuffling*, which is not a significant issue, since the increase in forecasting performance is more important to the financial industry.

6. Conclusion

Given the rising usage of DNNs in various fields, this study proposed the deep learning-based model, *NuNet*, which is successfully capable of learning inter-company temporal features using market-size high dimensional data for forecasting stock market indexes. Solving the ultra-high dimensional problem, the proposed model architecture is composed of two modules, *market feature extractor* and *target index feature extractor*, which separates and learns information from all companies in the market and target index information in the early stage. In the *market feature extractor*, the architecture was designed with CNN and ConvLSTM to effectively learn patterns with tens of thousands of time-series. The proposed model is easily applicable without heavy preprocessing and hand-crafted feature extraction. Furthermore, the problem of data shortage, which is a limitation of deep learning-based daily-basis prediction modeling in the financial market, was overcome by a simple and powerful data augmentation method, *column-wise random shuffling*. A new mini-batch sampling method, called *trend sampling*, was proposed to reflect the real time, dynamic situation of the financial market, which is greatly shaken by one unexpected event.

In this study, the main experiments were performed in three steps to verify model effectiveness. First, we compared the forecasting performance with other baseline models to investigate model feasibility. Second, we estimated the difference of forecasting errors between models that used vanilla random sampling and *trend sampling* to analyze the effects of *trend sampling*. Finally, we explored the effectiveness of the augmentation approach by evaluating the test performance of the model without *column-wise random shuffling*. The results indicate that the proposed model successfully extracted robust features from market-size high dimensional data with *column-wise random shuffling*, while preventing overfitting. Moreover, *trend sampling* enabled the model to learn dynamically changes in market patterns. The results demonstrated that our proposed methods adapt well to real-time dynamic changes, owing to the large amount of accessible market information.

Large amounts of unstructured textual data as well as numerical data are recently accumulated online. Although this study focuses on structured numerical data, we believe that our work provides a baseline for the use of large-scale time-series data. Thus, future research could develop a multi-modular model with inputs of large scale unstructured and structured time-series data to improve forecasting performance.

Financial support: This research was supported by the Technology Advancement Research Program (TARP) funded by the Ministry of Land, Infrastructure and Transport of the Korean government (Grant No. 19CTAP-C152017-01). The funders had no role in the study design, data collection and analysis, decision to publish, or preparation of the manuscript.

CRediT authorship contribution statement

Si Woon Lee: 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.

Declaration of Competing Interest

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

References

Akita, R., Yoshihara, A., Matsubara, T., & Uehara, K. (2016). Deep learning for stock prediction using numerical and textual information. In *Paper presented at the Computer and Information Science (ICIS), 2016 IEEE/ACIS 15th International Conference On* (pp. 1–6).

Al-Jumeily, D., & Hussain, A. J. (2015). The performance of immune-based neural network with financial time series prediction. *Cogent Engineering, 2*(1) 985005.

Ariyo, A. A., Adewumi, A. O., & Ayo, C. K. (2014). Stock price prediction using the ARIMA model. In *Paper presented at the Computer Modelling and Simulation (UKSim), 2014 UKSim-AMSS 16th International Conference On* (pp. 106–112).

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.

Baek, Y., & Kim, H. Y. (2018). ModAugNet: A new forecasting framework for stock market index value with an overfitting prevention LSTM module and a prediction LSTM module. *Expert Systems with Applications, 113*, 457–480.

Ballings, M., Van den Poel, D., Hespeels, N., & Gryp, R. (2015). Evaluating multiple classifiers for stock price direction prediction. *Expert Systems with Applications, 42*(20), 7046–7056.

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.

Barak, S., Arjmand, A., & Ortobelli, S. (2017). Fusion of multiple diverse predictors in stock market. *Information Fusion, 36*, 90–102.

Barak, S., & Modarres, M. (2015). Developing an approach to evaluate stocks by forecasting effective features with data mining methods. *Expert Systems with Applications, 42*(3), 1325–1339.

Cavalcante, R. C., Brasileiro, R. C., Souza, V. L., Nobrega, J. P., & Oliveira, A. L. (2016). Computational intelligence and financial markets: A survey and future directions. *Expert Systems with Applications, 55*, 194–211.

Chen, A., 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 *Paper presented at the Big Data (Big Data), 2015 IEEE International Conference On* (pp. 2823–2824).

Chen, Y., & Hao, Y. (2017). A feature weighted support vector machine and K-nearest neighbor algorithm for stock market indices prediction. *Expert Systems with Applications, 80*, 340–355.

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.

Chollet, F. (2018). *Deep learning mit python und keras: Das praxis-handbuch vom entwickler der keras-bibliothek MITP-Verlags*. GmbH & Co. KG.

Cootner, P. H. (1964). The random character of stock market prices.

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv Preprint arXiv:1810.04805,

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.

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research, 270*(2), 654–669.

Fuertes, A., 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.

Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. Paper presented at the Proceedings of the 34th International Conference on Machine Learning-Volume 70, 1243-1252.

Gorenc Novak, M., & Velušček, D. (2016). Prediction of stock price movement based on daily high prices. *Quantitative Finance, 16*(5), 793–826.

Gündüz, H., Yaslan, Y., & Çataltepe, Z. (2018). Stock market prediction with deep learning using financial news. In *Paper presented at the 2018 26th Signal Processing and Communications Applications Conference (SIU)* (pp. 1–4).

Guo, X., Zhang, H., & Tian, T. (2018). Development of stock correlation networks using mutual information and financial big data. *PLoS One, 13*(4) e0195941.

Hajizadeh, E., Seifi, A., Zarandi, M. F., & Turksen, I. (2012). A hybrid modeling approach for forecasting the volatility of S&P 500 index return. *Expert Systems with Applications, 39*(1), 431–436.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Paper presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770–778).

Ho, C., Damien, P., Gu, B., & Konana, P. (2017). The time-varying nature of social media sentiments in modeling stock returns. *Decision Support Systems, 101*, 69–81.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.

Huang, S. C. (2017). A big data analysis system for financial trading. *Global Business and Finance Review, 22*(3), 32–44.

Ince, H., & Trafalis, T. B. (2017). A hybrid forecasting model for stock market prediction. Economic Computation & Economic Cybernetics Studies & Research, 51(3)

Kara, Y., Boyacioglu, M. A., & Baykan, Ö. K. (2011). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the istanbul stock exchange. *Expert Systems with Applications, 38*(5), 5311–5319.

Kim, H. Y., & Won, C. H. (2018). Forecasting the volatility of stock price index: A hybrid model integrating LSTM with multiple GARCH-type models. *Expert Systems with Applications, 103*, 25–37.

Kim, T., & Kim, H. Y. (2019). Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data. *PloS One, 14*(2) e0212320.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv Preprint arXiv:1412.6980,

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Paper presented at the Advances in Neural Information Processing Systems*, 1097–1105.

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

Li, W., Wang, F., Havlin, S., & Stanley, H. E. (2011). Financial factor influence on scaling and memory of trading volume in stock market. *Physical Review E, 84*(4) 046112.

Li, Z., & Tam, V. (2017). A comparative study of a recurrent neural network and support vector machine for predicting price movements of stocks of different volatilites. Paper presented at the Computational Intelligence (SSCI), 2017 IEEE Symposium Series On, 1-8.

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.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... Ostrovski, G. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529.

Mo, H., Wang, J., & Niu, H. (2016). Exponent back propagation neural network forecasting for financial cross-correlation relationship. *Expert Systems with Applications, 53*, 106–116.

Moghaddam, A. H., Moghaddam, M. H., & Esfandyari, M. (2016). Stock market index prediction using artificial neural network. *Journal of Economics, Finance and Administrative Science, 21*(41), 89–93.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. Paper presented at the Proceedings of the 27th International Conference on Machine Learning (ICML-10), 807-814.

Nassirtoussi, A. K., Aghabozorgi, S., Wah, T. Y., & Ngo, D. C. L. (2015). Text mining of news-headlines for FOREX market prediction: A multi-layer dimension reduction algorithm with semantics and sentiment. *Expert Systems with Applications, 42*(1), 306–324.

Nelson, D. M., Pereira, A. C., & de Oliveira, R. A. (2017). Stock market's price movement prediction with LSTM neural networks. In *Paper presented at the Neural Networks (IJCNN), 2017 International Joint Conference On* (pp. 1419–1426).

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.

Pradeepkumar, D., & Ravi, V. (2017). Forecasting financial time series volatility using particle swarm optimization trained quantile regression neural network. *Applied Soft Computing, 58*, 35–52.

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.

Shin, Y. (2006). An empirical study on stock trading value of each investor type in the korean stock market. *Journal of the Korean Data and Information Science Society, 17*(4), 1099–1106.

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

Toczydlowska, D., & Peters, G. (2018). Financial big data solutions for state space panel regression in interest rate dynamics. *Econometrics, 6*(3), 34.

Villegas, R., Yang, J., Hong, S., Lin, X., & Lee, H. (2017). Decomposing motion and content for natural video sequence prediction. arXiv Preprint arXiv:1706.08033,

Wang, L., Zeng, Y., & Chen, T. (2015). Back propagation neural network with adaptive differential evolution algorithm for time series forecasting. *Expert Systems with Applications, 42*(2), 855–863.

Wang, Q., Jia, W., He, X., Lu, Y., Blumenstein, M., & Huang, Y. (2019). FACLSTM: ConvLSTM with focused attention for scene text recognition. arXiv Preprint arXiv:1904.09405,

Xingjian, S., Chen, Z., Wang, H., Yeung, D., Wong, W., & Woo, W. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Paper presented at the Advances in Neural Information Processing Systems*, 802–810.

Yuan, Z., Zhou, X., & Yang, T. (2018). In *Hetero-ConvLSTM: A deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data* (pp. 984–992). Data Mining.

Zeiler, M. D. (2012). ADADELTA: An adaptive learning rate method. arXiv Preprint arXiv:1212.5701,

Zeng, Y., Zeng, Y., Choi, B., & Wang, L. (2017). Multifactor-influenced energy consumption forecasting using enhanced back-propagation neural network. *Energy, 127*, 381–396.

Zhang, Y., & Wu, L. (2009). Stock market prediction of S&P 500 via combination of improved BCO approach and BP neural network. *Expert Systems with Applications, 36*(5), 8849–8854.

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

Zhou, K., Zhu, Y., & Zhao, Y. (2017). A spatio-temporal deep architecture for surveillance event detection based on ConvLSTM. Paper presented at the 2017 IEEE Visual Communications and Image Processing (VCIP), 1-4.