Expert Systems With Applications 143 (2020) 113041

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Portfolio management via two-stage deep learning with a joint cost

Hyungbin Yun[a,1], Minhyeok Lee[a,1], Yeong Seon Kang[b], Junhee Seok[a,*]

[a] School of Electrical Engineering, Korea University, 145 Anam-ro, Seongbuk-gu, Seoul 02841, Republic of Korea
[b] Department of Business Administration, University of Seoul 163 Seoulsiripdaero, Dongdaemun-gu, Seoul 02504, Republic of Korea

ARTICLE INFO

Article history:
Received 21 May 2019
Revised 24 September 2019
Accepted 16 October 2019
Available online 18 October 2019

Keywords:
Deep learning
Long short-term memory
Portfolio management
Joint cost function

ABSTRACT

Portfolio management is a series of processes that maximize returns and minimize risk by allocating assets efficiently. Along with the developments in machine learning technology, it has been studied to apply machine learning methods to prediction-based portfolio management. However, such methods have a few limitations. First, they do not consider the relations between assets for the prediction. In addition, the studies commonly focus on the prediction accuracy, neglecting the construction of portfolios. Furthermore, the methods have usually been evaluated with index data, which hardly represent actual prices to buy or sell an asset. To overcome these problems, Exchange Traded Funds (ETFs) are employed for base assets for the evaluation, and we propose a two-stage deep learning framework, called Grouped-ETFs Model (GEM), with a joint cost function. The GEM is designed to learn the features of inter-asset and groups in each stage. Also, the proposed joint cost can consider relative returns for the training while the relative returns are a crucial factor to construct a portfolio. The results of a rigorous evaluation with global ETF data indicate that the proposed GEM with the joint cost outperforms the equally weighted portfolio and the ordinary deep learning model by 33.7% and 30.1%, respectively. An additional experiment using sector ETFs verifies the generality of the proposed model where the results accord with those of the previous experiment.

1. Introduction

Prediction of the stock market is not straightforward since psychological factors of economic agents and unpredictable non-economic external factors affect a considerable influence (Barberis, 2013; Kaplanski & Levy, 2010). Even though it is challenging, the prediction of stock prices is essential for portfolio management of which objective is to maximize portfolio returns and minimize risks. Consequently, there have constantly been numerous attempts and studies in the fields of economics, statistics, and Machine Learning (ML) (De Prado, 2018; García-Galicia, Carsteanu & Clempner, 2019; Granger & Newbold, 2014; Heaton, Polson & Witte, 2017; Li & Yi, 2019; Mansour, Cherif & Abdelfattah, 2019).

An Exchange Traded Fund (ETF), a tradable fund that combines multiple assets, such as stocks, bonds, and raw materials, has a competitive advantage compared to conventional funds, as an investor can easily buy and sell ETFs in the stock market. ETFs also benefit from low transaction costs, tax efficiency, transparency, and low management fees (Deville, 2008). Due to such advantages, the ETF market size in the United States has grown from $2.2 billion to about $3.0 trillion, from 1997 to 2017 (Box, Davis & Fuller, 2018). For these reasons, instead of the index which is a synthetic value of an asset, such ETFs that actually exist in real market compose the assets of the portfolio that this study addresses, in order to demonstrate the method can be applied to the real market.

For portfolio management, numerous methods have been proposed over the decades. Mathematical and statistical methods constitute a dominant part among the methods (Morton & Pliska, 1995; Varga-Haszonits, Caccioli & Kondor, 2016; Yang, Couillet & McKay, 2015). Since the mean-variance analysis (Markowitz, 1991), the first modern portfolio management method using the mathematical and statistical approach, has been introduced, a variety of variants of the methods has been studied (Kalayci, Ertenlice & Akbay, 2019; Silva, Herthel & Subramanian, 2019; Tayalı & Tolun, 2018).

Recently, however, the ML-based methods are increasingly considered as alternatives and supplements of the statistical methods, due to a superb estimation performance resulting from rapid developments in ML and computing hardware. Among the ML methods, deep learning demonstrates astonishing results in many tasks, including image classification (Huang, Liu, Van Der Maaten & Weinberger, 2017), natural language processing (Devlin, Chang, Lee & Toutanova, 2019), and image synthesis (Lee & Seok, 2019),

* Corresponding author.
E-mail addresses: hb_yun@korea.ac.kr (H. Yun), suam6409@korea.ac.kr (M. Lee), yskang2014@uos.ac.kr (Y.S. Kang), jseok14@korea.ac.kr (J. Seok).
[1] This paper has two first authors who contributed equally to this work.

https://doi.org/10.1016/j.eswa.2019.113041
0957-4174/

compared to the other ML methods. Therefore, there have been many studies that attempt to apply deep learning to the stock market, in the last few years (Chong, Han & Park, 2017; Fischer & Krauss, 2018; Hoseinzade & Haratizadeh, 2019; Jagric, Bojnec & Jagric, 2015; Liu, 2019; Moews, Herrmann & Ibikunle, 2019; Orimoloye, Sung, Ma & Johnson, 2019).

This study aims to develop portfolio management methods using deep learning. Specifically, in this paper, a two-stage deep learning process is proposed, where predictions of individual assets and groups are separately performed in each stage, then the portfolio weights are calculated by combining the results of each stage. In addition, a joint cost function using the training of a deep learning model is proposed. The proposed joint cost function can simultaneously address both the relative return, compared to the other assets, and the fluctuation of the price of an asset. Furthermore, while previous studies have commonly evaluated the methods with the index data (Cervelló-Royo, Guijarro & Michniuk, 2015; Chen & Chen, 2016; Chiang, Enke, Wu & Wang, 2016; Chourmouziadis & Chatzoglou, 2016; Niaki & Hoseinzade, 2013; Orimoloye et al., 2019; Patel, Shah, Thakkar & Kotecha, 2015), which are synthetic and can hardly represent actual costs to buy/sell an asset, the evaluation in this study is conducted with market prices of ETFs, which can reflect the real market more precisely. Also, the evaluation is performed by sliding window method, where the training is repeatably conducted with the most recent data at the training time, which also greatly reflects the condition that the methods are applied to the real market.

The main contributions of this paper are as follows:

- A two-stage deep learning process for portfolio management, in which the estimations of each asset and group are separately performed, is proposed.
- A joint cost function to train a deep learning model that can consider both the absolute return and the relative return is proposed.
- The evaluation is performed with ETFs that reflect actual buy/sell costs, while previous studies have commonly employed indices of assets. The performances of conventional ML models, conventional deep learning models using a recurrent neural network, and the proposed two-stage deep learning model with the joint cost are reported under the condition.

This paper proceeds as follows. In Section 2.1, the problems that are addressed in this paper are described. Also, a brief review of related studies is provided in Section 2.2. Then the motivations and the novelty of the proposed methods are specified in Section 2.3. Section 3 elaborates the proposed methods, called the GEM and the joint cost function. Sections 4.1 and 4.2 explains the market data and the simulation settings that are used in the experiments. In Sections 4.3 and 4.4, the proposed methods are evaluated, compared to the other ML models, including a single-stage deep learning model. Finally, Section 5 discusses the results, and Section 6 concludes and proposes future research.

2. Background

2.1. Problem description

Portfolio management is a process of asset allocation to maximize returns and minimize risks. Let N be the number of assets composing a portfolio, then the portfolio management can be summarized as solving the following problem:

$$\mathrm{W} = \underset{\mathrm{W}=\{w_1,\ldots,w_N\}}{\operatorname{argmax}} \left\{ \sum_k w_k E(r_k) - \lambda\sigma\left(\sum_k w_k E(r_k)\right)\right\}, \tag{1}$$

$$subject\ to\ \sum_k w_k = 1\ and\ 0 \le w_k \le 1, \tag{2}$$

where w_k denotes a portfolio weight for the kth asset, r_k denotes a return of kth asset, $k \in \{1, \ldots, N\}$, σ is a method to evaluate the risk, λ is a weight parameter for the risk value, and $E(r_k)$ is an expected return of the kth asset.

Therefore, the prediction of $E(r_k)$ is essential in order to solve the portfolio management problem. Conventionally, the $E(r_k)$ is estimated by a statistical method using a weighted average of historical returns of the asset with a given time period from t to T:

$$E_{stat}(r_k) = \frac{1}{\sum_i \omega_i} \sum_i \omega_i r_{k,i} \tag{3}$$

where ω_i is the weight for historical returns, and $r_{k,i}$ is the historical return of the kth asset at the time i, and $i \in \{t, \ldots, T\}$.

Recently, with the rapid developments in ML, an alternative approach has been made to employ the ML models to estimate the expected returns in (1). In this approach, the ML models are trained with historical data, which include not only the historical returns used in the statistical approach but also any other information that is related to the asset. Then, the model predicts the future returns with given sequences of the data:

$$E_{ML}(r_k) = \hat{r}_{k,T+\epsilon} = f_\theta\left(D_{k,T}\right), \tag{4}$$

where $\hat{r}_{k,T+\epsilon}$ denotes the expected future returns of the asset at time $T+\varepsilon$, f_θ is the ML model trained with the historical data, and $D_{k,T}$ is a set of data given at time T.

This paper aims to solve the problem of (4). by using a deep learning model, which is one of the most popular ML models in recent years. In this paper, a novel method using two-stage deep learning with a joint cost is proposed to solve (4), thereby demonstrating the superiority of the proposed method compared to the other ML models, including the conventional single-stage deep learning model.

2.2. Literature review of related studies

Most modern portfolio management methods are based on the mean-variance model proposed by Markowitz (1952, 1991). The mean-variance model constructs a set of portfolios based on the past returns of various stocks and selects the most efficient portfolio that is expected to maximize returns and minimize risks (Zhou & Li, 2000).

While the concept of the mean-variance model was introduced more than 60 years ago, variants of the mean-variance model are still used by adopting novel techniques; Kalayci et al. (2019), an extensive review study, addresses such an issue. Among the techniques, population-based algorithms, such as Genetic Algorithm (GA), are the most popular algorithms that have been studied to solve (1). For example, Dreżewski and Doroz (2017) introduce a variant of GA for multi-objective portfolio management, where two sexes are employed in which each sex aims to achieve different objectives. Liagkouras (2019) also proposes a multi-objective portfolio management algorithm using a population-based evolutionary method which uses a three-dimensional encoding. The proposed method, called Three-Dimensional Encoding Multi-objective Evolutionary Algorithm (TDMEA), demonstrates superior performances compared to two well-known variants of GA, when the algorithms are evaluated with seven different tasks.

On the other hand, a few studies aim to apply ML algorithms to portfolio management. Fernández and Gómez (2007) introduce the artificial neural network model for the portfolio selection problem. They aim to use Hopfield network for the selection of efficient frontiers of portfolios. The Hopfield network

is compared with three other metaheuristic algorithms, including GA and Tabu Search (TS) algorithm. Freitas, De Souza and de Almeida (2009) propose a prediction-based mean-variance model that uses an artificial neural network. This model assumes that the predicted returns are the expected returns and uses the discrepancies between the expected returns and the actual returns as expected risks. They demonstrate that a neural network prediction-based mean-variance model outperforms an ordinary mean-variance model.

Along with such attempts to apply ML to portfolio management, in recent years, the ML techniques have been astonishingly advanced with modern developments in computing hardware. Among the techniques, deep neural networks, namely deep learning, have demonstrated fine performances to handle time-series data, including financial data. For example, Long Short-Term Memory (LSTM) is a conventional deep learning model to deal with time-series data (Greff, Srivastava, Koutník, Steunebrink & Schmidhuber, 2016), and has shown numerous superior results compared to the other ML models. The LSTM has been commonly employed for many different tasks to address time-series data, such as speech recognition (Sak, Senior & Beaufays, 2014; Zhang et al., 2016), language understanding (Zazo, Lozano-Diez, Gonzalez-Dominguez, Toledano & Gonzalez-Rodriguez, 2016), analyzing electrocardiogram (Chauhan & Vig, 2015, October), and predicting traffic flow (Tian & Pan, 2015).

In terms of predicting future prices of stocks, ML models have been frequently employed as well. Numerous previous studies aim to predict stock market prices or their directions using ML methods, including deep learning (Adebiyi, Adewumi & Ayo, 2014; Patel et al., 2015; Rather, Agarwal & Sastry, 2015). For example, Nelson, Pereira, and de Oliveira (2017) demonstrate that the LSTM model shows higher accuracy than the multi-layer perceptron or the random forest model for predicting the direction of stock prices. Chatzis, Siakoulis, Petropoulos, Stavroulakis and Vlachogiannakis (2018) study an interesting topic regarding the prediction of stock market crisis by ML models. Many ML models that are conventionally used, including Support Vector Machine (SVM), Random Forest (RF), eXtreme Gradient Boosting (XGBoost), and artificial neural networks, are evaluated for this problem. Jeong and Kim (2019) employ reinforcement learning for stock market trading. They apply the deep Q-learning model, which is a recently introduced deep learning model for reinforcement learning, to this problem. Moews et al. (2019) aim to use deep learning model to estimate the changing points of stock market prices. To handle the chaotic nature of the stock market, feature engineering is utilized as a pre-processing method before the feed-forward neural network.

2.3. Motivations and novelties of this study

However, these existing studies regarding the use of ML for the stock market prediction have three main shortcomings, which motivate us to investigate this study. First, many previous studies do not take the relations among assets into account (Qiu, Song & Akagi, 2016; Zhong & Enke, 2017). They generally aim to predict individual asset prices given the historical returns and volatility of each asset; thus, these methods cannot directly consider the construction of a portfolio. Second, while a few studies provide a framework for prediction-based portfolio management, these studies do not apply recent developments in prediction models, such as novel deep learning techniques and LSTM. Third, for the evaluation of the methods, indices of the stock market are employed. Since the indices are synthetic, they can hardly represent actual buy/sell costs of a market, although prices of ETFs reflect the actual costs more precisely.

To overcome these problems, we propose a Grouped-ETF Model (GEM) that corresponds to a two-stage deep-learning framework for portfolio management. GEM is designed to combine ETFs in two stages according to correlation. First, highly correlated ETFs are grouped, and thus, each prediction model is constructed with the grouped ETFs. We refer to this model as an in-group model, which determines the asset-allocation weights of the ETFs belonging to the corresponding groups.

In addition, we generate a model that determines the portfolio weights of each group, which we call a total-group model. The portfolio weights of each asset are determined by multiplying the predicted weight values of the asset in the total-group model and the in-group model. Employing GEM, we can determine the relation among ETFs and use a future-prediction model, which enables actual portfolio simulation and can thereby improve the performance of the model.

We employ LSTM for the prediction model in the proposed framework from among the conventional deep learning structures. LSTM is an artificial neural network structure that not only learns the relation between each variable but also learns the changes between the previous and the current time (Greff et al., 2016). Therefore, LSTM can potentially outperform the other neural network models in predicting market prices, for which changes over time are usually a significant factor in future prices.

We also propose a joint cost function for the training of LSTM used in the GEM, through utilizing the advantage that a deep learning model can be trained to optimize multiple objectives by simply adding multiple cost functions for the training. The joint cost function is designed to simultaneously learn normalized targets that express the relative degree of fluctuation in the price of each ETF, in addition to the return target used in conventional models. Employing the joint cost function, the model can learn not only the price fluctuation of each ETF asset but also the price fluctuation relation between ETFs assets.

We evaluate the proposed model using rigorous experiments, in which we separate the training set and the validation set precisely using real datasets, specifically Morgan Stanley Capital International (MSCI) All Country World Index (ACWI) ETFs and US Sector ETFs. We utilize the 32 ETFs that compose the majority of the MSCI ACWI ETFs. For the US Sector ETFs, we select four of the 16 ETFs with the longest operating period by sector, listed in the New York Stock Exchange (NYSE) and National Association of Securities Dealers Automated Quotations (NASDAQ) markets. Furthermore, we apply the sliding window method, which assumes the actual operation of a fund, to evaluate the proposed model precisely.

3. Methods

3.1. Overview

Fig. 1 illustrates the process of the portfolio management method which we propose in this study. First, using the raw price data of each asset, we calculate various market indicators that we use as input features for the deep learning models. Second, we reduce the dimension of the calculated indicators through principal component analysis (PCA) to prevent overfitting. Third, we train Integrated ETF Model (IEM), which corresponds to a conventional deep learning model for ETFs, and the proposed GEM using the preprocessed data. In the proposed GEM, the model is composed of two stages, where the in-group model predicts the price direction of highly correlated assets and the total-group model predicts the price direction of groups with low correlation. The proposed joint cost function is used to train the model in this process. Then, we construct portfolios using a softmax function with the outputs

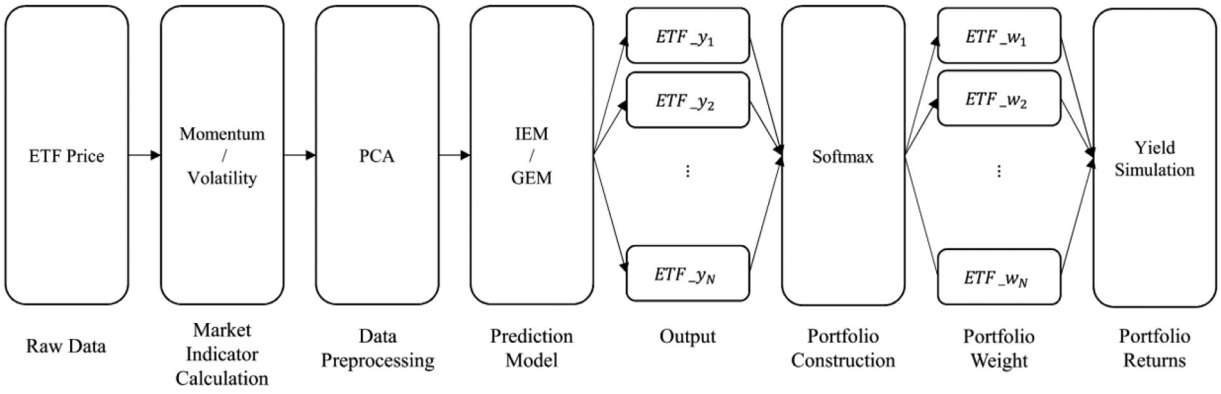

Fig. 1. Evaluation process of the prediction-based portfolio construction methods in this study.

Table 1
Lists of market indicator variables that are used for the input variables of machine learning models.

Indicators	Formulas	
Momentum	3-month momentum	$m_{t,3} = P_t - P_{t-90}$
	6-month momentum	$m_{t,6} = P_t - P_{t-180}$
	9-month momentum	$m_{t,9} = P_t - P_{t-270}$
	12-month momentum	$m_{t,12} = P_t - P_{t-360}$
	14 days RSI (relative strength index)	$d = 14$ $U_t(n) = \begin{cases} P_{t-n} - P_{t-1-n}, & P_{t-n} - P_{t-1-n} > 0 \\ 0 & , \ P_{t-n} - P_{t-1-n} \le 0 \end{cases}$ $D_t(n) = \begin{cases} 0 & , P_{t-n} - P_{t-1-n} > 0 \\ P_{t-n} - P_{t-1-n}, & P_{t-n} - P_{t-1-n} \le 0 \end{cases}$ $RSI_t = 100 - \frac{100}{1+(\sum_{i=0}^{N-1} \frac{U_t(t-i)}{d})/(\sum_{i=0}^{N-1} \frac{D_t(t-i)}{d})}$
	12-month RSM (return signal momentum)	$d = 360$ $q_t(n) = \begin{cases} 1, P_{t-n} - P_{t-1-n} > 0 \\ 0, P_{t-n} - P_{t-1-n} \le 0 \end{cases}$, $RSM_t = \frac{1}{d} \sum_{i=0}^{d-1} q_t(i)$
Volatility	500-day price SD (standard deviation)	$d = 500, SD_t = \sqrt{\frac{1}{d} \sum_{i=0}^{d-1} (P_{t-i} - \frac{1}{d} \sum_{j=0}^{d-1} P_{t-j})}$
	500-day historic VaR (Value at Risk) 99%	$d = 500$, $VaR_t = P_t \times SD_t \times 2.33 \times d$
	IVOL (idiosyncratic volatility)	$d = 500, \quad IVOL_t = \sqrt{RE_t} \times 100$ RE_t = Regression error between Benchmark and each ETF price data

of each model. We describe each step of the process in detail in the following sections.

3.2. Input data and dimension reduction through PCA

Instead of using market price data itself as the input for the proposed model, we use momentum and volatility features that directly represent trends or risks of an asset. Momentum is originally a physics term that refers to the slope of a point on a curve. In economics, momentum implies a rate of change in prices. This metric is used to measure the acceleration of trends when prices fluctuate in the market. By using various types of momentum as input features, we design the model to capture and learn the patterns of prices that actually affect future prices of assets.

Volatility indicates the price variance over a period of time. In stock-market predictions, volatility is referred to as risk because it is difficult to predict the price as price volatility increases. We use market indicators such as momentum and volatility as input data for the proposed model to not only maximize the returns but also minimize the volatility of the portfolio. Such conventional market indicators using prices of assets are used in this study (Fu, 2009; Rodríguez-González, García-Crespo, Colomo-Palacios, Iglesias & Gómez-Berbís, 2011). Table 1 summarizes the various momentum and volatility indicators that we use as the input features to train the model.

We reduce the dimensions of the extracted features using PCA to prevent overfitting. PCA can reduce the dimensionality of data while minimizing data loss (Jolliffe & Cadima, 2016). For each model, the number of principal components is set at the minimum value of which explanatory rate exceeds 90%.

For an in-group model of GEM, we perform a PCA on the indicators of each ETF asset to maintain the features of each individual asset. Then, we use only the calculated principal components belonging to each group as input data for each in-group model.

In the total-group model of the GEM, we group the market indicators of each ETF and then perform a PCA for each group separately. Unlike the previous process, we do not use a PCA for the market indicators of each ETF, though we use it within groups to maintain the features of each group. The calculated principal components for each group are bound and used as the input data of the total-group model. We can express the indicator data format I

and the input data X as follows:

$$I = \begin{bmatrix} m_{t,3,1} & \cdots & m_{t-s,6,1} \\ m_{t,6,1} & & m_{t-s,6,1} \\ \vdots & & \vdots \\ VaR_{t,1} & & VaR_{t-s,1} \\ IVOL_{t,1} & & IVOL_{t-s,1} \\ \vdots & \ddots & \vdots \\ m_{t-s,3,N} & & m_{t-s,3,N} \\ m_{t-s,6,N} & & m_{t-s,6,N} \\ \vdots & & \vdots \\ VaR_{t,N} & & VaR_{t-s,N} \\ IVOL_{t,N} & & IVOL_{t-s,N} \end{bmatrix}, \tag{5}$$

$$X = \begin{bmatrix} pc_{1,t} & \cdots & pc_{1,t-s} \\ pc_{2,t} & & pc_{2,t-s} \\ \vdots & & \vdots \\ pc_{k,t} & \ddots & pc_{k,t-s} \\ P_{t,1} & & P_{t-s,1} \\ \vdots & & \vdots \\ P_{t,N} & \cdots & P_{t-s,N} \end{bmatrix}, \tag{6}$$

where t is the time step parameter; N is the ETF indexing number; s is the number of samples; k is the number of PCs, which depend on the explanatory rate; $P_{t,i}$ is the ETF price of the i^{th} ETF at time t; $X_t = \{pc_{1,t}, pc_{2,t}, \cdots, P_{t,N}\}$; and $i = 1, 2, \cdots, N$.

We do not directly use asset prices, that is, $\{P_{t,1}, P_{t,2}, \cdots, P_{t,N}\}$, for the total-group model that determines the weights of each group because we expect that the group prices are difficult to represent as each ETF asset has a different price scale. In contrast, we utilize input data in the form of (6) in all the other models.

3.3. Two-stage grouped ETF model for LSTM structure

LSTM is a neural network model for time-series data that accounts for both current and historical data. LSTM is a type of Recurrent Neural Network (RNN) that originated from a basic RNN called the Elman network (Jia, Zhao, Zheng & Hou, 2019). RNN saves the output values of the hidden layer of a previous point in the learning process as a state layer and uses them as input data in the next training step (Zhang, Wang & Liu, 2014). By contrast, LSTM can consider not only a certain past point but also a long sequence of past points data through a forget gate (Greff et al., 2016). The size of the past time series is called the sequence size.

In this study, we set the sequence size to 50. In a basic LSTM model, when the sequence size exceeds 50, while the training time greatly increases, it has been known that the performance does not improve much (Pascanu, Mikolov & Bengio, 2013; Sutskever, Vinyals & Le, 2014). Because we design the model to predict the price direction after 30 days without forecasting the flow of the price, we use only the last output among the whole sequence to predict the target value.

We use a multilayer LSTM structure, which is a stack of LSTM cells using the previous output values of the low-layer LSTM cell as the inputs of the high-layer LSTM cell. Specifically, we use three layers of LSTM cells in all models. Depending on the model, the number of hidden nodes is determined from 50% to 60% of the input dimensions to prevent overfitting (Panchal & Panchal, 2014). We use the hyperbolic tangent function, which is traditional for an LSTM model, as the activation function. The activation function at the last output layer is set to the sigmoid function to generate an output value between 0 and 1.

According to the portfolio selection method proposed by Markowitz, constructing a portfolio with low correlated assets reduces risk and increases returns (Lee, Cheng & Chong, 2016). Based on this method, we propose a model that determines the weights of each individual group when constructing a portfolio, then reassign the weights of each specific asset in each group, which corresponds to the two-stage deep learning model that we propose in this study. If we group portfolios according to their regional characteristics and assign different weights to each group, then we expect that the portfolios will have low variance as the correlation between regions is commonly lower than the correlation within regions. By utilizing this intuition for the proposed model, we expect to reduce risk.

In addition, when we consider each group and ETF to construct an ML model, the model can account for local characteristics, and therefore improve the accuracy of the model. Instead of constructing one model that predicts the weight of the portfolio at once, we use a single model that predicts the weights of each group and multiple models that predict the weights of each ETF within the group.

Constructing separate models can be considered as assigning different objectives to learn for each model. Specifically, the total-group model learns the relation between low-correlated data, and the in-group model learns the relation between highly correlated data. It is expected to improve prediction accuracy by generating a specialized model by dividing the role performed in one model into several models. For convenience, we call the model that predicts the ETF weights by dividing them into two stages as the GEM. We calculate the portfolio weights constraint of the GEM as follows:

$$\sum_{i=1}^{G} \sum_{j=1}^{N_G} g_{t,i} c_{t,i,j} = 1, \tag{7}$$

$$subject\ to\ 0 \le g_{t,i} \le 1\ and\ 0 \le c_{t,i,j} \le 1, \tag{8}$$

where G is the group index number, N_G is the number of ETFs belonging to the G^{th} group, $g_{t,i}$ is the predicted weight of the i^{th} group at time t, and $c_{t,i,j}$ is the predicted weight of the j^{th} ETF belonging to the i^{th} group at time t.

In the GEM, we construct a group-specific portfolio using the predicted values from each in-group model, then multiply each in-group portfolio by the weights calculated by the total-group model, as in (7). In summary, for the GEM, the total-group model predicts the weights of each group and the in-group models predict the weights within the groups. The final portfolio of assets consists of the weights calculated by this GEM process.

3.4. Single-stage IEM

To evaluate the GEM, we use an integrated deep learning model as the baseline of the proposed model. The Integrated ETF Model (IEM) corresponds to a one-group, single-stage model; therefore, IEM becomes a baseline of the evaluation since IEM predicts each ETF price with each model, which is a conventional manner of using deep learning model to predict stock market prices, as described in the previous sections. There are some differences in the PCA process between the IEM and GEM due to a characteristic of IEM that considers individual assets separately. For the IEM, we perform the PCA on the indicators of each ETF asset to maintain the features of each individual asset. Then, we use all calculated principal components as the input data. We derive the portfolio weights constraint of the IEM as follows:

$$\sum_{k=1}^{N} w_{t,k} = 1, \tag{9}$$

$$subject\ to\ 0 \le w_{t,k} \le 1, \tag{10}$$

where $w_{t,k}$ is the predicted weight of the k^{th} ETF at time t.

3.5. Joint cost function

We propose the joint cost function that uses two cost functions with different types of target data simultaneously for a single model. Therefore, the two cost functions share the weight parameters of the model. In the training process, the cost functions have equal weights; thus, we can train the weight parameters of the model using both cost functions equally.

The two cost functions use different types of target data: directional data and normalized continuous data. These data types represent the price increase/decrease and the relative profit between the assets, respectively. We can improve performance by using a simultaneous learning process to estimate the price fluctuations and the relative profit of an asset.

To construct a model to predict the direction of 30-day returns, we set the directional target data to one if the returns are above 0, and zero otherwise:

$$y_{t,i} = \begin{cases} 1, & r^{30}_{t,i} > 0 \\ 0, & r^{30}_{t,i} \le 0 \end{cases}, \tag{11}$$

where $r^{30}_{t,i}$ is the monthly return of asset i, which we can calculate by $r^{30}_{t,i} = \frac{P_{t,i}}{P_{t-30,i}} - 1$; and $y_{t,i} \in Y_t$, where Y_t is a set of directional target data for N number of assets at time t.Whereas we use the directional target data to represent the price increase/decrease of each individual asset, the normalized continuous target data indicate the relative profit of an asset at a certain time. To set these target values, the monthly returns for each ETF are normalized between 0 and 1 according to the profit relative to the other assets:

$$z_{t,i} = \frac{r^{30}_{t,i} - min(R_t)}{max(R_t) - min(R_t)}, \tag{12}$$

where $R_t = \{r^{30}_{t,1}, r^{30}_{t,2}, \cdots, r^{30}_{t,N}\}$, $min(R_t)$ is the minimum value in R_t, $max(R_t)$ is the maximum value in R_t, and $z_{t,i} \in Z_t$, where Z_t is a set of normalized target data for N number of assets at time t.

One of the main advantages of neural network models is that the cost function is easy to modify. We use this advantage to propose a model that is trained simultaneously by two cost functions. The ordinary mean squared error (MSE) cost function C_t, which is conventionally used for deep learning model, is defined as follows:

$$C_t = \frac{1}{N}\sum_{i=1}^{N}\left(y_{t,i} - \hat{y}_{t,i}\right)^2, \tag{13}$$

where $\hat{y}_{t,i}$ is the model prediction value of the i^{th} ETF at time t.

In contrast, we can calculate our proposed joint cost function, JC_t, as follows:

$$JC_t = \frac{1}{N}\sum_{i=1}^{N}\left(y_{t,i} - \hat{y}_{t,i}\right)^2 + \frac{1}{N}\sum_{j=1}^{N}\left(z_{t,j} - \hat{y}_{t,j}\right)^2. \tag{14}$$

The targets in the joint cost function are the direction of asset prices ($y_{t,i}$) and relative prices ($z_{t,j}$). $y_{t,i}$ is either 0 or 1, and $z_{t,j}$ is a continuous value between 0 and 1. Both the relative asset prices and the directions are crucial factors in constructing a portfolio with multiple assets. Many previous studies use only either of these as the main target. The proposed joint cost function makes it possible to learn the price direction and relative price of each asset simultaneously. Moreover, by considering relative prices instead of absolute prices, we can use the joint cost function to learn the relations among multiple assets, which is another essential factor in portfolio construction.

4. Result

4.1. Data description

We use ETF price data from 32 countries included in the MSCI ACWI. The MSCI ACWI is a traditional global portfolio that launched on May 31, 1990, and is currently in operation. The entire dataset spans approximately 15 years, from May 20, 2002, to June 8, 2017, with a sample of 5499 daily closing prices, which are obtained from Yahoo finance database. We define the portfolio rebalancing interval as one month, assuming the actual operation of a fund. To predict 1590 daily test datasets from January 31, 2013, through June 8, 2017, we divide the dataset into the sliding window structure.

We set an equally weighted portfolio and IEM as baselines and conduct an experiment to evaluate whether the proposed method outperforms the baselines. In this evaluation, we use ETF price data instead of the stock price data from each country for several reasons.

First, as an ETF in each country follows the stock market index as a benchmark, the ETF reflects the country's market index relatively more accurately than the stock price does. While it can be considered to directly employ the index data, however, the index data are synthetic so that they hardly represent the actual cost to buy or sell the market. The price of the ETFs that follows the index can reflex the cost more precisely since the ETFs are actual products which can be bought in the stock market. Second, it is easy to experiment because there is no need to prepare a portfolio of stocks for each country. Table 2 lists the 32 ETFs we use for the analysis. As we described in the previous section and Table 1, we use various features based on the price data of each ETF.

To solve the problem of constructing a global portfolio, we follow prior economic research to construct a forecasting model. Junior, Mullokandov and Kenett (2015) find that regional factors influence a market index; that is, the stock market index of each country depends on the country's location. Therefore, the ETF price depends on each region's factors.

We measure the correlation between each ETF price and find that the correlation within each continent tends to be strong. Fig. 2 illustrates the correlations between ETF prices, demonstrating that correlation with the group to which it belongs is generally high. The average correlation within each continent is 0.79; in contrast, the average inter-continent correlation is 0.70, where a p-value of the two groups of correlation is below 0.001.

We group emerging markets separately because their characteristics are stronger those of the regional groups. Table 3 shows the average correlation within the groups. Although it is not absolute, it is generally understood that the within-group correlation is high. Therefore, we employ the proposed method (GEM) to address the problem of constructing a portfolio.

We divide the ETFs by continent and then apply the GEM to each country (in-group) and continent (total-group). The total-group model determines the weight of each continent, and the in-group models determine the weight of each country's ETF in each continent, as Fig. 3 shows.

4.2. Experiment setting

To assume the actual operation of a fund, we use the sliding window for the simulation. We set the rebalancing period to 30 days. Then, the model learns the latest data at each rebalancing period using the sliding window method. Thus, the model considers the most up-to-date training data at each period. Such an evaluation method using the sliding-window method follows many previous works to show that the method outperforms uniform sampling when it is applied to stock-market data

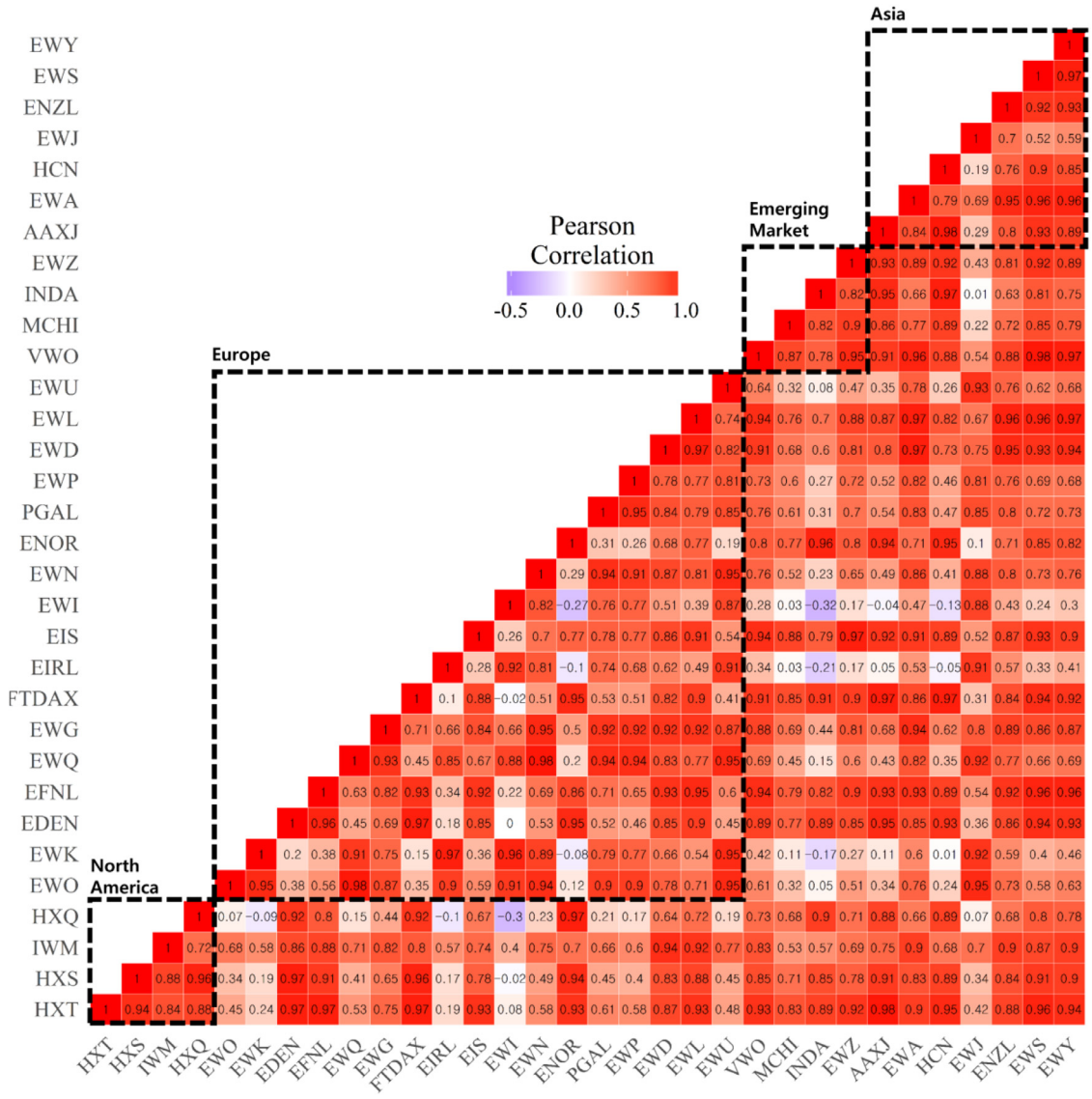

Fig. 2. Pearson correlation coefficient among the ETFs used for the training data. Rows and columns denote abbreviations of ETFs. The analysis was conducted over the training set, from May 20, 2002, to January 30, 2013, with 3909 observations.

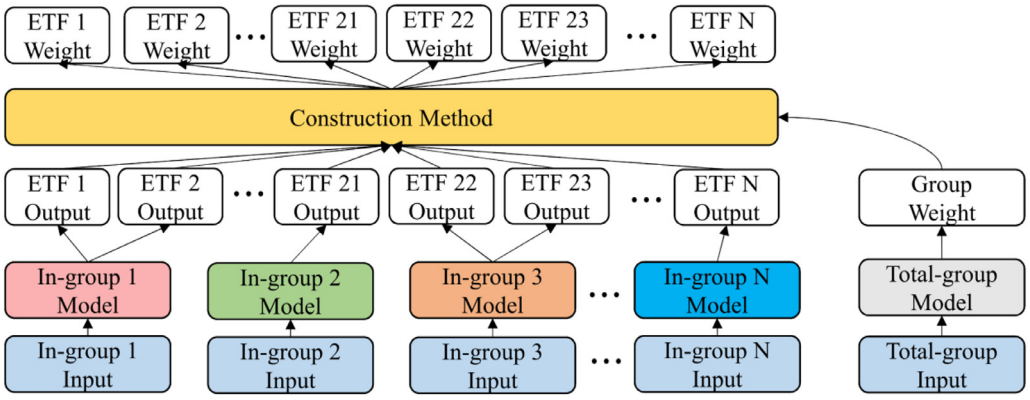

Fig. 3. Architecture of two-stage grouped ETF model. Each in-group model predicts the returns of ETFs in respective groups in the first stage. Then, the total-group model allocates the group weights for each group. The portfolio weights are calculated by multiplying the two estimates from the models.

Table 2
Lists of ETFs and regions that compose a portfolio in the experiment.

Region	Country	ETF Name
North America	Canada	Horizons S&P/TSX 60™ Index ETF
	United States	Horizons S&P 500® Index ETF
	United States	Horizons Nasdaq-100 Index ETF
	United States	iShares Russell 2000 ETF
Europe	Austria	iShares MSCI Austria Capped ETF
	Belgium	iShares MSCI Belgium Capped ETF
	Denmark	iShares MSCI Denmark Capped ETF
	Finland	iShares MSCI Finland Capped ETF
	France	iShares MSCI France ETF
	Germany	iShares MSCI Germany ETF
	Germany	Recon Capital Dax
	Ireland	iShares MSCI Ireland Capped ETF
	Israel	iShares MSCI Israel Capped ETF
	Italy	iShares MSCI Italy Capped ETF
	Netherlands	iShares MSCI Netherlands ETF
	Norway	iShares MSCI Norway Capped ETF
	Portugal	Global X MSCI Portugal ETF
	Spain	iShares MSCI Spain Capped ETF
	Sweden	iShares MSCI Sweden ETF
	Switzerland	iShares MSCI Switzerland Capped ETF
	United Kingdom	iShares MSCI United Kingdom ETF
Emerging Markets	Emerging Markets	Vanguard Emerging Markets Stock Index Fund
	Brazil	iShares MSCI Brazil Capped
	China	iShares MSCI China ETF
	India	iShares MSCI India ETF
Asia Pacific	Asia ex Japan	iShares MSCI All Country Asia ex Japan ETF
	Australia	iShares MSCI Australia ETF
	Hong Kong	Horizons China High Dividend Yield Index ETF
	Japan	iShares MSCI Japan ETF
	New Zealand	iShares MSCI New Zealand Capped ETF
	Singapore	iShares MSCI Singapore ETF
	South Korea	iShares MSCI South Korea Capped ETF

Table 3
Pearson correlation coefficients between each continent.

	North America	Europe	Emerging market	Asia pacific
Asia pacific	–	–	–	0.78
Emerging market	–	–	**0.86**	0.79
Europe	–	0.68	0.59	0.69
North America	**0.87**	0.58	0.78	0.79

Bold indicates a high correlation coefficient (>0.8). The analysis was conducted over the training set, from May 20, 2002, to January 30, 2013, with 3909 observations.

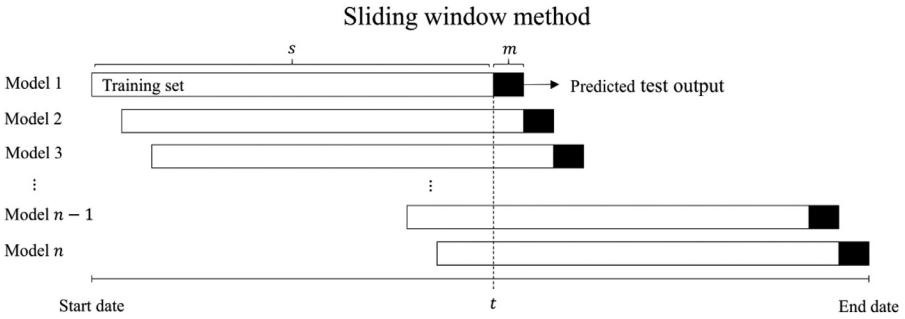

Fig. 4. Sliding window method for evaluating portfolios. In the sliding window method, at each rebalancing time, a new model is trained and constructed with the most recent obtainable data.

(Chou & Nguyen, 2018; Guo, Wang, Yang & Miller, 2015). Fig. 4 illustrates the sliding-window method for the simulation, which we conduct using Algorithm 1 below.

Through the sliding window method, the cumulative test dataset becomes the size of $n \times m$ and we use it to verify the performance of the models. With this method, the most up-to-date training data in the rebalancing period can be learned. Finally, we can calculate daily returns and risks with the weight of the ETFs predicted at each rebalancing:

$$Pr_t = \sum_{i=1}^{N} w_{t,i} r_{t,i}^1, \tag{15}$$

$$\sigma_p = \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N} w_{t,i} w_{t,j} \sigma_i \sigma_j cov(i,j)}, \tag{16}$$

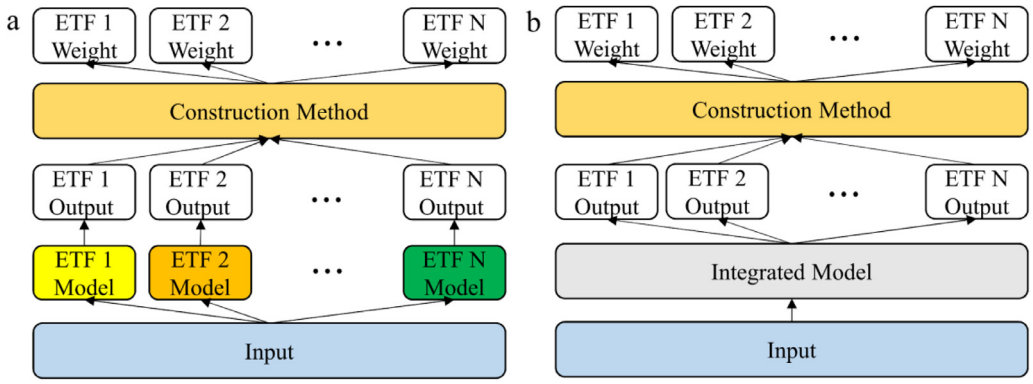

Fig. 5. Architectures of the conventional prediction-based model and integrated ETF model. (a) The conventional prediction-based model used for the random forest and the support vector machine of which output is univariate. (b) The integrated model used for the deep learning methods.

Algorithm 1 Sliding Window Method.

Require: Daily Historical data set h
Input: data set h, rebalancing period m, first rebalancing time t, number of times rebalance n, number of training data s
Output: sliding window set S
1: **Initialization** $t = s,\ i = 1$
2: **while** $i \leq n$ **do**
3: training set $u = \{h_{(t-s+1)}, \ldots, h_{(t)}\}$
4: test set $v = \{h_{(t+1)}, \ldots, h_{(t+m)}\}$
5: set $S_{(i)} = \{u, v\}$
6: $i = i + 1$
7: $t = t + m$
8: **end while**

where $r^1_{t,i}$ is the daily return calculated by $r^1_{t,i} = \frac{p_{t,i}}{p_{t-1,i}} - 1$, σ_i is the standard deviation of the daily returns of the ith ETF, and $cov(i, j)$ is the covariance between ith and jth ETF.

We divide the entire dataset into 53 training and test datasets at intervals of one month. To optimize the structure of each of the 53 models, we set 10% of the training set as the validation set and empirically determine the structure of the model at the level where the MSE value falls to an appropriate level.

Because the GEM is a combination of multiple models and uses the joint cost instead of the ordinary MSE, it is difficult to directly compare the costs of the IEM and GEM. Therefore, we evaluate the performances using the Spearman correlation and the Sharpe ratio as accuracy scores, which can be calculated as follows:

$$Accuracy\ Score = \frac{TP + TN}{TP + FP + TN + FN}, \tag{17}$$

$$Spearman\ Correlation\ Coefficient = \rho_{rg_X, rg_Y} = \frac{cov(rg_X, rg_Y)}{\sigma_{rg_X}\ \sigma_{rg_Y}}, \tag{18}$$

$$Sharpe\ Ratio = \frac{R_p - R_f}{\sigma_p}, \tag{19}$$

where TP is true positive, TN is true negative, FP is false positive, FN is false negative, rg_X is the rank of X, R_p is the portfolio's return, and R_f is the risk-free rate.

The Sharpe ratio (Ledoit & Wolf, 2008) is a key metric that shows the returns and risk relation as proposed by Sharpe (1994). Generally, the higher Shape ratio signifies the better performance of the portfolio. We use the U.S. 3 month treasury bill rate as the risk-free rate for the Sharpe ratio. However, the U.S. 3 month treasury bill was close to 0% during the test period, so we calculate the Sharpe ratio by fixing it to 0.1% for convenience of calculation. Furthermore, we use the other metrics to evaluate the performance, i.e., Information Ratio (IR), Maximum DrawDown (MDD), Value at Risk (VaR), and Conditional Value at Risk (CVaR), which are general metrics to assess a portfolio (Bacon, 2004).

We simulate portfolio returns using real prices with the assumption of zero transaction cost. Because each ETF has different transaction costs that vary depending on the period, it is difficult to apply them in a simulation environment. The baselines also do not account for transaction costs for a fair comparison.

4.3. Performance evaluation of GEM with LSTM

We conduct experiments using the following ML methods as the other baselines to evaluate the performance of the GEM with LSTM: random forest, support vector machine, and multi-layer perceptron (MLP). In the random forest and the support vector machine, it is not ordinary to construct a multi-output model. Therefore, we construct each model for each asset separately, as illustrated in Fig. 5(a). By contrast, as each artificial neural network model can have several output variables, we create an integrated model, as in Fig. 5(b).

When composing a portfolio, the performance depends largely on how much the model correctly ranks the assets. Therefore, we conduct the experiment by setting the accuracy score for the selection of the top N assets as the performance criterion of the models. Fig. 6 shows the results of the accuracy score for the top 10 to 18 selections of the models. As shown in the figure, all GEMs show a higher accuracy score than respective IEMs. When the MLP is used, the performances of the IEM and the proposed GEM are distinct by 11.1% on average. Similarly, the differences are 1.4% when LSTM is used.

We also evaluate the models whether they properly learn the relations between the assets, which is also a significant factor to construct a portfolio, by Spearman's rank correlation between the actual ranks and the estimated ranks for monthly returns. The results indicate that the GEM outperforms the IEM in all models, except for MLP. Table 4 summarizes the averages of the accuracy scores and the Spearman's correlations over the test set.

In order to test the models with a portfolio using the sliding window method, we adopt an ordinary softmax function for the prediction values and determine the weights of each ETF asset. The softmax function can be calculated by:

$$Softmax(\hat{y}_i) = \frac{e^{\hat{y}_i}}{\sum_{j=1}^{N} e^{\hat{y}_j}}. \tag{20}$$

Then, we evaluate the cumulative returns of the portfolios constructed by the models, with the sliding window method over the

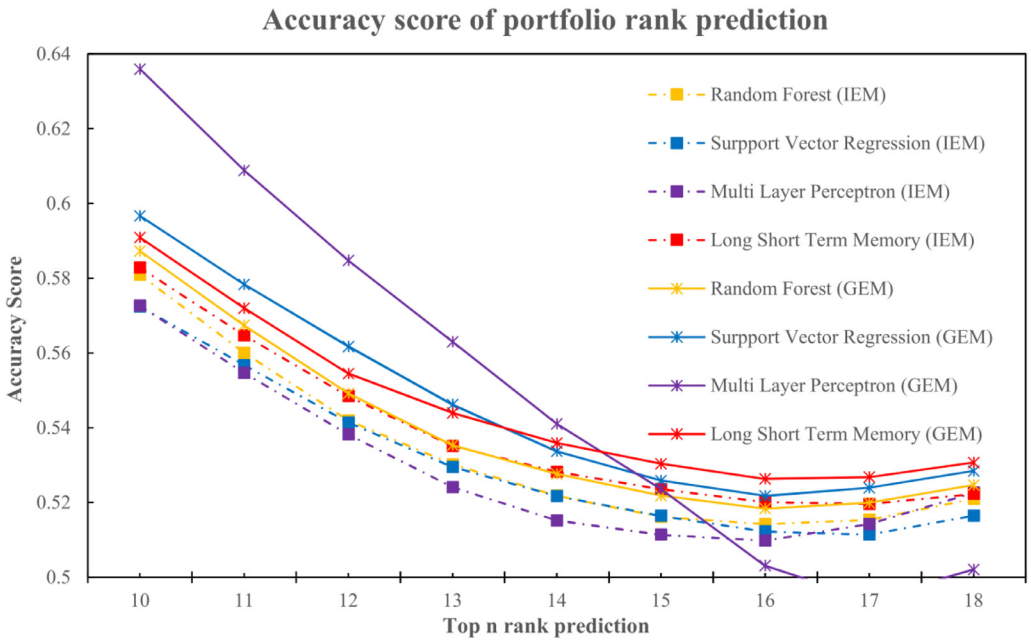

Fig. 6. Accuracy score of the rank predictions by each method. IEM: Integrated ETF Model; GEM: Grouped ETF Model. The evaluation was conducted over the test set, from January 31, 2013, to June 8, 2017, with 1590 observations.

Table 4
Average of accuracy score and average Spearman correlation for each model with different machine learning algorithms.

	Average of accuracy score		Average of Spearman correlation	
	IEM	GEM	IEM	GEM
Random Forest	0.6626	**0.6687**	0.01893	**0.04362**
Support Vector Regression	0.6609	**0.6719**	0.01713	**0.07077**
Multi-Layer Perceptron	0.6646	**0.6817**	**0.02911**	0.01187
Long Short-Term Memory	0.6656	**0.6707**	0.03258	**0.06159**

Bold indicates higher performance among the IEM and the GEM in each algorithm. The evaluation was conducted over the test set, from January 31, 2013, to June 8, 2017, with 1590 observations. IEM: Integrated ETF Model; GEM: Grouped ETF Model.

Table 5
Summary of simulation results for the integrated ETF models with different machine learning algorithms.

	IEM							
Model	$E(r_m)$	$\sigma(r_m)$	Sharpe ratio	IR	Jensen's α	MDD	VaR	CVaR
1/N	0.4478	0.0350	0.0993	0.0000	0.0000	0.2066	−0.0515	−0.0637
Random Forest	0.4428	0.0350	0.0980	−0.3791	−0.0006	0.2089	−0.0516	−0.0638
Support Vector Regression	0.4427	0.0348	0.0984	−0.1716	−0.0004	0.2060	**−0.0504**	−0.0646
Multi-Layer Perceptron	0.4384	0.0348	0.0972	−0.1594	−0.0008	0.2068	−0.0530	−0.0639
Long Short-Term Memory	**0.5179**	**0.0346**	**0.1208**	**1.0333**	**0.0096**	**0.1879**	−0.0505	**−0.0630**

Bold and underline indicate the best and the second-best performance in each metric, respectively. The evaluation was conducted over the test set, from January 31, 2013, to June 8, 2017, with 1590 observations. IEM: Integrated ETF Model; 1/N: equally weighted portfolio; r_m: monthly return; $E(\cdot)$: average; $\sigma(\cdot)$: standard deviation; IR: Information Ratio; MDD: Maximum DrawDown; VaR: Value at Risk; CVaR: Conditional Value at Risk.

test set of actual ETF data, as described in the previous section. Fig. 7(a) and (b) show the cumulative returns of the IEMs and the GEMs of the models, respectively. The equally weighted portfolio of GEM, which is one of the baselines of this experiment, is obtained by multiplying the two equal weights, i.e., the equal weight of the total-group and the equal weight the in-group, for a fair comparison with the other GEMs. As shown in the figure, the proposed method, the GEM with LSTM, significantly outperforms the other models.

Specifically, the results assessed by the different metrics are summarized in Tables 5 and 6. As a result, the GEM with LSTM outperforms the equally weighted portfolio by 4.9%, when the models are evaluated by the final cumulative returns of the portfolios. In addition, in the GEMs, all the methods outperform the equally weighted portfolio, although only LSTM is superior to the equally weighted portfolio in the IEM, which empirically demonstrates the proposed grouping method is effective. Overall, all GEMs outperform respective IEMs by 15.0% on average, which also signifies the effectiveness of the proposed GEM.

Furthermore, GEMs also demonstrate superior performances compared to IEMs by 27.5% on average, when they are evaluated by Sharpe ratio which is a metric to assess a portfolio with both the returns and the volatility, as aforementioned. Among them, the GEM with LSTM shows the best performance compared to all other

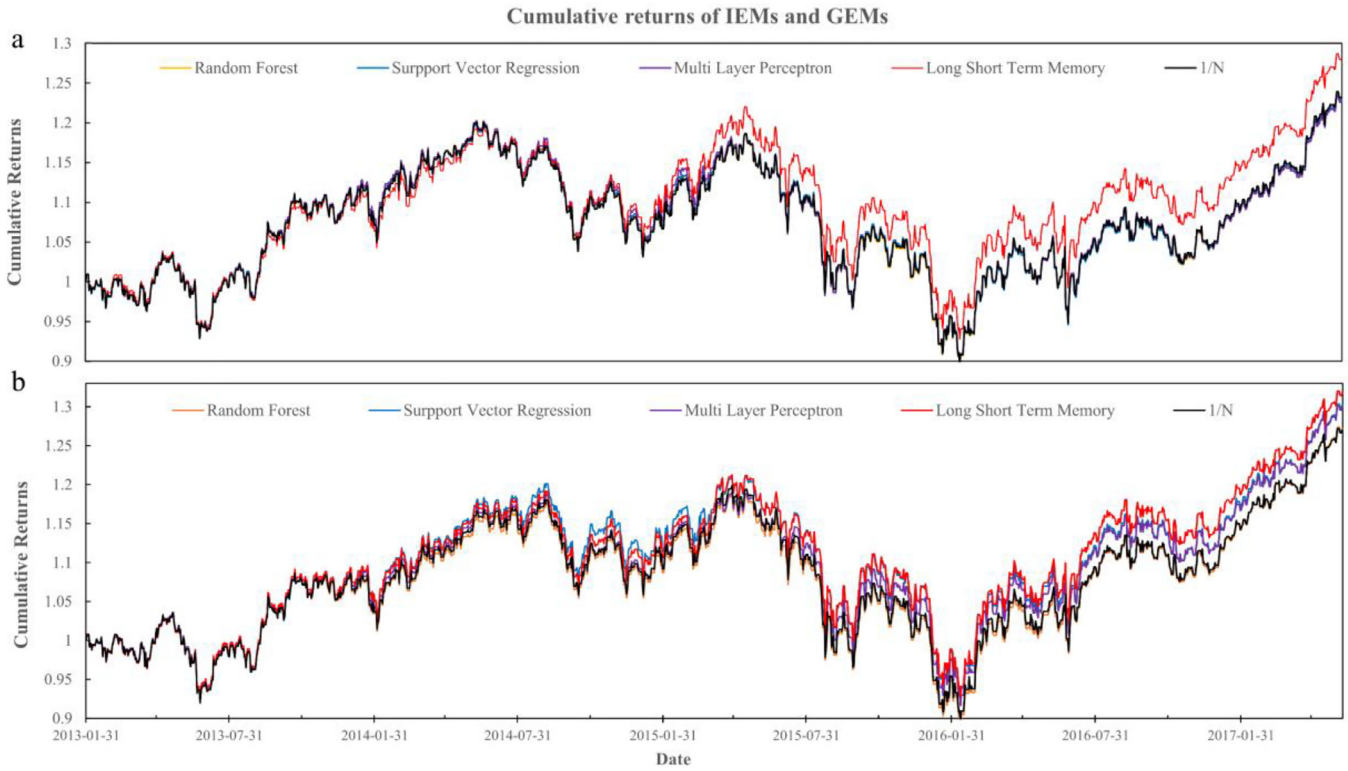

Fig. 7. Cumulative returns for (a) Integrated ETF Model (IEM) and (b) Grouped ETF Model (GEM) with different machine learning methods. The evaluation was conducted over the test set, from January 31, 2013, to June 8, 2017, with 1590 observations.

Table 6
Summary of simulation results for the grouped ETF model with different machine learning algorithms.

	GEM							
Model	$E(r_m)$	$\sigma(r_m)$	Sharpe ratio	IR	Jensen's α	MDD	VaR	CVaR
1/N	0.5008	0.0340	0.1179	0.2689	0.0094	0.2022	−0.0538	−0.0649
Random Forest	0.5028	0.0339	0.1187	0.2771	0.0099	0.1998	−0.0539	−0.0650
Support Vector Regression	0.5443	**0.0330**	0.1347	0.4795	0.0168	0.1871	−0.0522	−0.0641
Multi-Layer Perceptron	0.5412	0.0336	0.1314	0.4750	0.0153	0.1844	−0.0550	−0.0640
Long Short-Term Memory	**0.5659**	**0.0330**	**0.1411**	**0.5730**	**0.0195**	**0.1843**	**−0.0504**	**−0.0629**

Bold and underline indicate the best and the second-best performance in each metric, respectively. The evaluation was conducted over the test set, from January 31, 2013, to June 8, 2017, with 1590 observations. GEM: Grouped ETF Model; 1/N: equally weighted portfolio; r_m: monthly return; $E(\cdot)$: average; $\sigma(\cdot)$: standard deviation; IR: Information Ratio; MDD: Maximum DrawDown; VaR: Value at Risk; CVaR: Conditional Value at Risk.

ML methods. Also, evaluated by Sharpe ratio, the performance of the GEM with LSTM is superior to the GEM with SVM, which is the second-best model, by 4.8%.

4.4. Performance of the joint cost

In this section, we investigate whether the proposed joint cost, which is introduced to consider both the absolute return and the relative return, can further enhance the performance. The GEM with LSTM, which shows the best performance in the previous section, is employed for the baseline of this experiment. The joint cost function is applied to the baseline, then the performance difference between before and after applying the joint cost is compared. Fig. 8 illustrates the cumulative returns of the baseline (GEM with LSTM) and the joint cost applied to the baseline. In addition, the evaluation results of the portfolios are summarized in Table 7.

As shown in the results, the joint cost function demonstrates superior performance compared to the ordinary loss function. Specifically, evaluated by the portfolio returns, it is demonstrated that the joint cost can improve the performance by 10.7%. Although the standard deviation of the returns increases by 1.2% when the joint cost is employed, however, the overall performance of the joint cost outperforms that of the ordinary model, when they are comprehensively evaluated by both the return and the risk. Evaluated with Sharpe ratio, which is a metric to assess both the characteristics, the effectiveness of the joint cost is demonstrated where the joint cost outperforms the baseline by 11.7%. Also, such a result corresponds to enhancement by 33.7%, compared to the equally weighted portfolio with GEM.

4.5. Additional experiment with a sector ETF dataset

To verify that the proposed framework using two-stage learning is generally applicable, we perform an additional experiment with the other dataset composed of sector ETFs. In contrast to the previous experiment where the groups are constructed by the regional characteristic, business sectors are used in this experiment to form groups. Table 8 provides a list of the ETFs for each business sector group. Specifically, we use four sector and factor groups, i.e., health care, technology, real estate, and large-cap, which are com-

Fig. 8. Cumulative returns for the model trained with different cost functions. The evaluation was conducted over the test set, from January 31, 2013, to June 8, 2017, with 1590 observations.

Table 7
Summary of simulation results for each cost function.

Model	$E(r_m)$	$\sigma(r_m)$	Sharpe ratio	IR	Jensen's α	MDD	VaR	CVaR
LSTM+ GEM	0.5659	**0.0330**	0.1411	0.5730	0.0195	**0.1843**	**−0.0504**	**−0.0629**
LSTM+ GEM + JC	**0.6263**	0.0334	**0.1576**	**0.7555**	**0.0268**	0.1970	−0.0524	−0.0645

Bold indicates higher performance among the ordinary cost function and the joint cost function. The evaluation was conducted over the test set, from January 31, 2013, to June 8, 2017, with 1590 observations. LSTM + GEM: Long Short-Term Memory with Grouped ETF Model with the conventional cost function; LSTM + GEM + JC: Long Short-Term Memory with Grouped ETF Model and the Joint Cost function.

Table 8
Lists of ETFs and sectors for the additional experiment using sector ETFs.

Sector	ETF Name
Health Care	ProShares Ultra Health Care (RXL)
	PowerShares DWA Healthcare Momentum ETF (PTH)
	Vanguard Health Care ETF (VHT)
	iShares US Healthcare ETF (IYH)
Technology	iShares North American Tech ETF (IGM)
	Vanguard Information Technology ETF (VGT)
	VanEck Vectors Gaming ETF (BJK)
	PowerShares DWA Technology Momentum ETF (PTF)
Real Estate	First Trust S&P REIT ETF (FRI)
	iShares US Real Estate ETF (IYR)
	ProShares Ultra Real Estate (URE)
	Vanguard Real Estate ETF (VNQ)
Large Cap	ProShares Ultra Dow30 (DDM)
	iShares Morningstar Large-Cap ETF (JKD)
	SPDR S&P 500 ETF (SPY)
	SPDR Portfolio Large Cap ETF (SPLG)

posed of US ETFs. In each sector, four ETFs are selected and employed, as the sub-elements of the sectors; therefore, the 16 ETFs and the four sectors construct a portfolio in this experiment.

The dataset covers about 10 years from Jan 24, 2008, to Mar 23, 2018, with 3712 samples of daily closing price data. The architecture of the deep learning model used in this experiment is basically similar to that of the previous experiment. Also, we use the same parameters to optimize the deep learning models and the same feature variables as the inputs of the deep learning models. A few changes are applied for this experiment due to the differences in the datasets, where the sample size decreases compared to the previous experiment. Therefore, in order to prevent overfitting, a simper deep learning architecture is used for both the total-group model and the in-group models in which the deep learning models are composed of two layers of LSTM cells.

Fig. 9 and Table 9 show the cumulative returns and the evaluation results of portfolios constructed by each method. Evaluated with the final returns of the portfolios, the GEM outperforms the equally weighted portfolio and the IEM by 2.6% and 3.3%, respectively. In addition, the GEM with the joint cost shows superior performance compared to the equally weighted portfolio and the GEM by 6.1% and 3.5%, respectively.

In addition, while the risk of the portfolio generally increases when the joint cost is used, the overall performance is enhanced when both the return and the risk are comprehensively evaluated by Sharpe ratio. The Sharpe ratio of the IEM is 0.2162, which is very similar to that of the equally weighted portfolio (0.2130); in contrast, the Sharpe ratio of the GEM method is 0.2258, which outperforms the equally weighted portfolio by 6.0%. Furthermore, when the joint cost method is adopted, the IEM outperforms the equally weighted portfolio, with a Sharpe ratio of 0.2222, which signifies the effectiveness of the proposed joint cost. The best performance is obtained when the joint cost is applied to the GEM of which the Sharpe ratio is 0.2409, which is a 13.1% improvement compared to the equally weighted portfolio.

Such results accord with the results of the previous experiment, where the GEM trained by the joint cost demonstrates the superiority compared to the other methods. Also, it is empirically verified that using the joint cost to train deep learning models can enhance the performance since the models using the joint cost commonly outperforms respective counterparts in all the experiment. The overall results of the additional experiment also indicate the effectiveness of the GEM and the joint cost that are proposed in this study.

5. Discussion

In this paper, the two-stage deep learning architecture, called GEM, with the joint cost function is proposed to address several shortcomings in the previous studies that employ prediction-based portfolio management models using ML, as described in Section 2. We design the two-stage model to separately learn the features of highly correlated assets and the features of less well-correlated as-

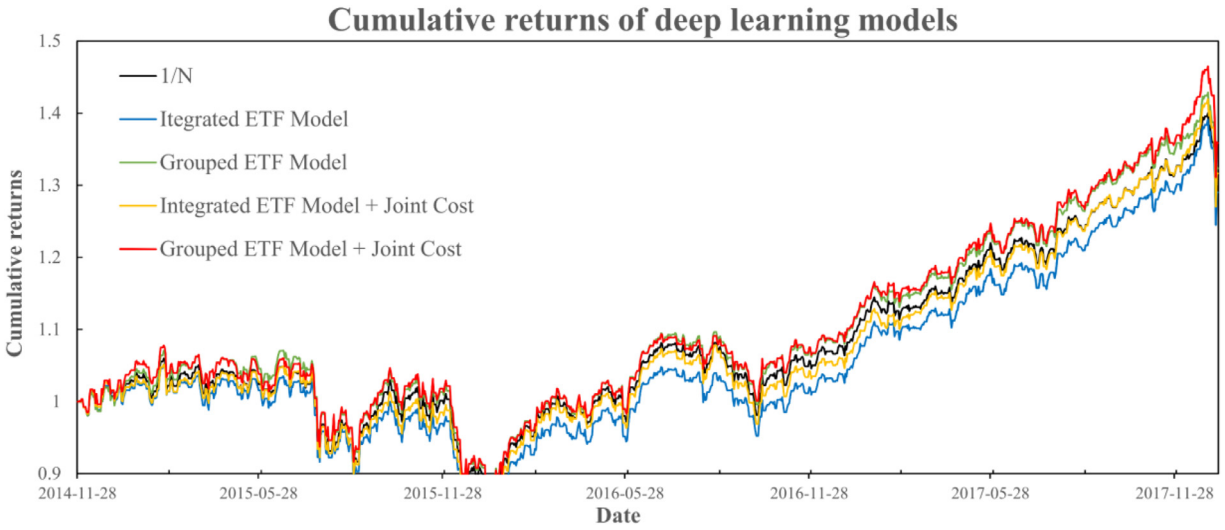

Fig. 9. Cumulative returns for the additional experiment using sector ETFs. The evaluation was conducted over the test set, from November 28, 2014, to March 23, 2018, with 1140 observations.

Table 9
Summary of simulation results for the additional experiment using sector ETFs.

	$E(r_m)$	$\sigma(r_m)$	Sharpe ratio	IR	Jensen's α	MDD	VaR	CVaR
1/N	0.9697	0.0408	0.2130	NA	0.0000	**0.1391**	−0.0632	−0.0888
IEM	0.9453	**0.0391**	0.2162	−0.1149	0.0034	0.1500	<u>−0.0612</u>	<u>−0.0869</u>
IEM + JC	1.0066	<u>0.0408</u>	0.2222	0.2595	0.0059	0.1471	−0.0672	**−0.0867**
GEM	<u>1.0238</u>	0.0409	<u>0.2258</u>	<u>0.7906</u>	<u>0.0071</u>	0.1441	−0.0671	−0.0897
GEM + JC	**1.0913**	0.0411	**0.2409**	**1.1144**	**0.0157**	<u>0.1412</u>	**−0.0587**	−0.0909

Bold and underline indicate the best and the second-best performance in each metric, respectively. The evaluation was conducted over the test set, from November 28, 2014, to March 23, 2018, with 1140 observations. 1/N: equally weighted portfolio; IEM: Integrated ETF Model with the conventional cost function; IEM + JC: IEM with the Joint Cost function; GEM: Grouped ETF Model with the conventional cost function. GEM + JC: GEM with the Joint Cost function.

sets in each stage. In addition, while ordinary cost functions for deep learning, such as the mean squared error and the cross-entropy, only consider the features of each individual asset, the joint cost function is proposed in this paper, in order to also learn the relation between the assets since the relation is a significant factor to construct an efficient portfolio.

With extensive simulations using the actual ETF datasets, the GEM is evaluated and verified that the GEM outperforms conventional manners of using ML, including the single-stage deep learning model. Specifically, evaluated with a global ETF dataset, the GEMs show superior performances compared to IEMs, the single-stage ML models, by 18.4% on average, in terms of the return of the portfolios. Furthermore, assessed with both the return and the risk by Sharpe ratio, all GEMs with different ML models demonstrate the superiority compared to respective IEMs with the ML models, where the improvement is 19.9% on average. Overall, the proposed two-stage deep learning framework significantly outperforms the conventional manner of using deep learning models which directly predict returns of each individual asset.

In addition, the effectiveness of the joint cost function, which is designed to use both the absolute return and the relative return, is demonstrated. Evaluated with the global ETF dataset, the joint cost enhances the montly return and Sharpe ratio by 0.6% and 9.6%, respectively. However, the risk of the portfolio using the joint cost increases in general in the experiment.

The generality of the proposed methods is verified with the other dataset, where the business sectors form a group. We apply the proposed methods, i.e., the GEM and the joint cost, to the sector ETF dataset. The results accord with those of the previous experiment. Specifically, the GEMs outperform the IEMs, and the performances increase by using the joint cost in both the IEM and the GEM. The GEM with the joint cost shows the best performance in common with the previous simulation.

Such results represent a possibility of applying machine learning predictions to portfolio management problems. While statistical methods are dominant in obtaining the expected returns of assets for portfolio management, this study demonstrates that a two-stage deep learning framework can be a fine alternative to such methods and outperforms the other machine learning and deep learning methods.

In addition, in this study, although only the expected returns are predicted by the two-stage deep learning framework, it is also possible to estimate the risks of assets by machine learning algorithms. The estimation of both the expected returns and the risks is essential for portfolio selection problem, as described in Section 2.1. This study aims to solve the problem of (4), which corresponds to applying machine learning to estimating the expected returns in portfolio management problem. In a similar manner, the risks also be estimated by machine learning methods, while statistical methods have commonly been used. Such an extension of this study for the risk estimation should be performed.

However, while we demonstrate the effectiveness of the GEM and the joint cost through extensive simulation studies, a few limitations exist in this study. First, constraints of the portfolio have not been considered although some constraints exist for the construction of a portfolio in real life, due to various limitations to operate a fund (Kalayci et al., 2019). For example, the transaction cost is a specific constraint which can affect the performance of the portfolios. To handle such constraints, as a simple approach, a penalty to deal with the constraint can be added to the cost

function for optimizing a deep learning model, which has not been addressed in this study. Second, due to the rapid advancement in the deep learning technology, state-of-the-art deep learning techniques to handle time-series data, such as the bidirectional LSTM and Convolutional Neural Networks (CNNs), have not been employed in this study. Among the techniques, CNNs have demonstrated superior performances to learn time-series data in recent years while they had been commonly and mostly used for images (Borovykh, Bohte & Oosterlee, 2018; Liu, Hsaio & Tu, 2018). To extend this study, such techniques can be considered to be used to enhance performance.

6. Conclusion

In this paper, we propose a two-stage deep learning framework and a joint cost function to train the deep learning model for portfolio management. The effectiveness and the generality of the proposed methods are demonstrated by the extensive simulation studies using ETF price datasets, which can properly represent the actual cost to buy and sell an asset. In the experiments, the GEM, the proposed two-stage deep learning framework, outperforms the conventional single-stage deep learning process, in all cases. Also, evaluated by both the return and the risk of the portfolio, the joint cost is verified that it can enhance the performance in general. The overall results indicate the superiority and effectiveness of the proposed methods.

For future work, we will aim to address the limitations of this study that are described in the previous section. First, the possibility should be studied whether additional penalties can be adopted for the cost function to address several constraints of portfolios in real life. For example, the transaction cost and the sector capitalization constraints can be handled by the additional penalties based on Lagrange multiplier method. Second, very recent developments in deep learning technology can be applied to this study. We will investigate whether there is a chance to enhance the proposed method by adopting recent developments, such as bidirectional LSTM and CNNs.

Declaration of Competing Interest

To the best of our knowledge, the named authors have no conflict of interest, financial or otherwise.

Credit authorship contribution statement

Hyungbin Yun: Conceptualization, Investigation, Methodology, Writing - original draft, Writing - review & editing. **Minhyeok Lee:** Conceptualization, Investigation, Methodology, Writing - original draft, Writing - review & editing. **Yeong Seon Kang:** Investigation, Methodology. **Junhee Seok:** Conceptualization, Methodology, Supervision, Writing - original draft, Writing - review & editing.

Acknowledgments

This work was supported by the National Research Foundation of Korea grants (NRF-2017R1C1B2002850, NRF-2019R1A2C1084778) as well as a grant from Mirae Asset Global Investments.

References

Adebiyi, A. A., Adewumi, A. O., & Ayo, C. K. (2014). Comparison of ARIMA and artificial neural networks models for stock price prediction. *Journal of Applied Mathematics, 2014*, 614342.

Bacon, C. (2004). *Practical portfolio performance measurement and attribution*. New York: John Wiley & Sons.

Borovykh, A., Bohte, S., & Oosterlee, C. W. (2018). Dilated convolutional neural networks for time series forecasting. *Journal of Computational Finance* press.

Box, T., Davis, R. L., & Fuller, K. P. (2018). ETF competition and market quality. *Financial Management, 48*, 873–916.

Cervelló-Royo, R., Guijarro, F., & Michniuk, K. (2015). Stock market trading rule based on pattern recognition and technical analysis: Forecasting the DJIA index with intraday data. *Expert Systems with Applications, 42*, 5963–5975.

Chatzis, S. P., Siakoulis, V., Petropoulos, A., Stavroulakis, E., & Vlachogiannakis, N. (2018). Forecasting stock market crisis events using deep and statistical machine learning techniques. *Expert Systems with Applications, 112*, 353–371.

Chauhan, S., & Vig, L. (2015, October). Anomaly detection in ECG time signals via deep long short-term memory networks. *Presentation at IEEE international conference on data science and advanced analytics, Paris, France*.

Chen, T.-l., & Chen, F.-. Y. (2016). An intelligent pattern recognition model for supporting investment decisions in stock market. *Information Sciences, 346*, 261–274.

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.

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.

Chou, J.-. S., & Nguyen, T.-. K. (2018). Forward forecast of stock price using sliding-window metaheuristic-optimized machine-learning regression. *IEEE Transactions on Industrial Informatics, 14*, 3132–3142.

Chourmouziadis, K., & Chatzoglou, P. D. (2016). An intelligent short term stock trading fuzzy system for assisting investors in portfolio management. *Expert Systems with Applications, 43*, 298–311.

De Prado, M. L. (2018). *Advances in financial machine learning*. New York: John Wiley & Sons.

Deville, L. (2008). Exchange traded funds: History, trading, and research. In *Handbook of financial engineering* (pp. 67–98). New York: Springer.

Devlin, J., Chang, M.-. W., Lee, K., & Toutanova, K. (2019, June). BERT: Pre-training of deep bidirectional transformers for language understanding. *Presentation at 2019 annual conference of the North American chapter of the association for computational linguistics: Human language technologies, Minneapolis, MN*.

Drężewski, R., & Doroz, K. (2017). An agent-based co-evolutionary multi-objective algorithm for portfolio optimization. *Symmetry, 9*, 168.

Fernández, A., & Gómez, S. (2007). Portfolio selection using neural networks. *Computers Operations Research, 34*, 1177–1191.

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

Freitas, F. D., De Souza, A. F., & de Almeida, A. R. (2009). Prediction-based portfolio optimization model using neural networks. *Neurocomputing, 72*, 2155–2170.

Fu, F. (2009). Idiosyncratic risk and the cross-section of expected stock returns. *Journal of financial Economics, 91*, 24–37.

García-Galicia, M., Carsteanu, A. A., & Clempner, J. B. (2019). Continuous-time reinforcement learning approach for portfolio management with time penalization. *Expert Systems with Applications, 129*, 27–36.

Granger, C. W. J., & Newbold, P. (2014). *Forecasting economic time series*. Cambridge: Academic Press.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2016). LSTM: A search space odyssey. *IEEE transactions on neural networks learning systems, 28*, 2222–2232.

Guo, Z., Wang, H., Yang, J., & Miller, D. J. (2015). A stock market forecasting model combining two-directional two-dimensional principal component analysis and radial basis function neural network. *PloS One, 10*, e0122385.

Heaton, J., Polson, N., & Witte, J. H. (2017). Deep learning for finance: Deep portfolios. *Applied Stochastic Models in Business Industry, 33*, 3–12.

Hoseinzade, E., & Haratizadeh, S. (2019). CNNpred: CNN-based stock market prediction using a diverse set of variables. *Expert Systems with Applications, 129*, 273–285.

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *Presentation at IEEE conference on computer vision and pattern recognition, Honolulu, HI*.

Jagric, T., Bojnec, S., & Jagric, V. (2015). Optimized spiral spherical self-organizing map approach to sector analysis–The case of banking. *Expert Systems with Applications, 42*, 5531–5540.

Jeong, G., & Kim, H. Y. (2019). Improving financial trading decisions using deep q-learning: Predicting the number of shares, action strategies, and transfer learning. *Expert Systems with Applications, 117*, 125–138.

Jia, W., Zhao, D., Zheng, Y., & Hou, S. (2019). A novel optimized GA–Elman neural network algorithm. *Neural Computing Applications, 31*, 449–459.

Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: A review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical Engineering Sciences, 374*, 20150202.

Junior, L., Mullokandov, A., & Kenett, D. (2015). Dependency relations among international stock market indices. *Journal of Risk Financial Management, 8*, 227–265.

Kalayci, C. B., Ertenlice, O., & Akbay, M. A. (2019). A comprehensive review of deterministic models and applications for mean-variance portfolio optimization. *Expert Systems with Applications, 125*, 345–368.

Ledoit, O., & Wolf, M. (2008). Robust performance hypothesis testing with the Sharpe ratio. *Journal of Empirical Finance, 15*, 850–859.

Lee, H.-. S., Cheng, F.-. F., & Chong, S.-. C. (2016). Markowitz portfolio theory and capital asset pricing model for Kuala Lumpur stock exchange: A case revisited. *International Journal of Economics Financial Issues, 6*, 59–65.

Lee, M., & Seok, J. (2019). Controllable generative adversarial network. *IEEE Access, 7*, 28158–28169.

Li, H.-. Q., & Yi, Z.-. H. (2019). Portfolio selection with coherent Investor's expectations under uncertainty. *Expert Systems with Applications, 133*, 49–58.
Liagkouras, K. (2019). A new three-dimensional encoding multiobjective evolutionary algorithm with application to the portfolio optimization problem. *Knowledge-Based Systems, 163*, 186–203.
Liu, C.-. L., Hsaio, W.-. H., & Tu, Y.-. C. (2018). Time series classification with multivariate convolutional neural network. *IEEE Transactions on Industrial Electronics, 66*, 4788–4797.
Liu, Y. (2019). Novel volatility forecasting using deep learning–Long short term memory recurrent neural networks. *Expert Systems with Applications, 132*, 99–109.
Mansour, N., Cherif, M. S., & Abdelfattah, W. (2019). Multi-objective imprecise programming for financial portfolio selection with fuzzy returns. *Expert Systems with Applications, 138* Article 112810.
Markowitz, H. (1952). Portfolio selection. *The Journal of Finance, 7*, 77–91.
Markowitz, H. (1991). Foundations of portfolio theory. *The Journal of Finance, 46*, 469–477.
Moews, B., Herrmann, J. M., & Ibikunle, G. (2019). Lagged correlation-based deep learning for directional trend change prediction in financial time series. *Expert Systems with Applications, 120*, 197–206.
Morton, A. J., & Pliska, S. R. (1995). Optimal portfolio management with fixed transaction costs. *Mathematical Finance, 5*, 337–356.
Niaki, S. T. A., & Hoseinzade, S. (2013). Forecasting s&p 500 index using artificial neural networks and design of experiments. *Journal of Industrial Engineering International, 9*, 1.
Orimoloye, L. O., Sung, M.-. C., Ma, T., & Johnson, J. E. (2019). Comparing the effectiveness of deep feedforward neural networks and shallow architectures for predicting stock price indices. *Expert Systems with Applications*, 112828.
Panchal, F. S., & Panchal, M. (2014). Review on methods of selecting number of hidden nodes in artificial neural network. *International Journal of Computer Science Mobile Computing, 3*, 455–464.
Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. *Presentation at international conference on machine learning, Atlanta, GA*.
Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock market index using fusion of machine learning techniques. *Expert Systems with Applications, 42*, 2162–2172.
Qiu, M., Song, Y., & Akagi, F. (2016). Application of artificial neural network for the prediction of stock market returns: The case of the Japanese stock market. *Chaos, Solitons Fractals, 85*, 1–7.
Rather, A. M., Agarwal, A., & Sastry, V. (2015). Recurrent neural network and a hybrid model for prediction of stock returns. *Expert Systems with Applications, 42*, 3234–3241.
Rodríguez-González, A., García-Crespo, Á., Colomo-Palacios, R., Iglesias, F. G., & Gómez-Berbís, J. M. (2011). CAST: Using neural networks to improve trading systems based on technical analysis by means of the RSI financial indicator. *Expert Systems with Applications, 38*, 11489–11500.
Sak, H., Senior, A., & Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. *Presentation at fifteenth annual conference of the international speech communication association, Singapore*.
Sharpe, W. F. (1994). The sharpe ratio. *Journal of Portfolio Management, 21*, 49–58.
Silva, Y. L. T., Herthel, A. B., & Subramanian, A. (2019). A multi-objective evolutionary algorithm for a class of mean-variance portfolio selection problems. *Expert Systems with Applications, 133*, 225–241.
Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Presentation at advances in neural information processing systems, Montréal, Canada*.
Tayalı, H. A., & Tolun, S. (2018). Dimension reduction in mean-variance portfolio optimization. *Expert Systems with Applications, 92*, 161–169.
Tian, Y., & Pan, L. (2015). Predicting short-term traffic flow by long short-term memory recurrent neural network. Presentation at IEEE international conference on smart city/socialcom/sustaincom, Chengdu, China.
Varga-Haszonits, I., Caccioli, F., & Kondor, I. (2016). Replica approach to mean-variance portfolio optimization. *Journal of Statistical Mechanics: Theory Experiment, 2016*, 123404.
Yang, L., Couillet, R., & McKay, M. R. (2015). A robust statistics approach to minimum variance portfolio optimization. *IEEE Transactions on Signal Processing, 63*, 6684–6697.
Zazo, R., Lozano-Diez, A., Gonzalez-Dominguez, J., Toledano, D. T., & Gonzalez-Rodriguez, J. (2016). Language identification in short utterances using long short-term memory (LSTM) recurrent neural networks. *PloS One, 11*, e0146917.
Zhang, H., Wang, Z., & Liu, D. (2014). A comprehensive review of stability analysis of continuous-time recurrent neural networks. *IEEE Transactions on Neural Networks Learning Systems, 25*, 1229–1262.
Zhang, Y., Chen, G., Yu, D., Yaco, K., Khudanpur, S., & Glass, J. (2016). Highway long short-term memory rnns for distant speech recognition. Presentation at IEEE international conference on acoustics, speech and signal processing, Shanghai, China.
Zhong, X., & Enke, D. (2017). Forecasting daily stock market return using dimensionality reduction. *Expert Systems with Applications, 67*, 126–139.
Zhou, X. Y., & Li, D. (2000). Continuous-time mean-variance portfolio selection: A stochastic LQ framework. *Applied Mathematics Optimization, 42*, 19–33.