Expert Systems With Applications 183 (2021) 115298

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Neural network with fixed noise for index-tracking portfolio optimization

Yuyeong Kwak [a], Junho Song [b,c], Hongchul Lee [a,*,1]

[a] School of Industrial Management Engineering, Korea University, 145 Anam-ro, Seongbuk-gu, Seoul 02841, Republic of Korea
[b] Graduate School of Convergence Science and Technology, Seoul National University, 18-dong, Gwanak-ro 1, Gwanak-gu, Seoul 08826, Republic of Korea
[c] ZeroOne AI, Next Canada HQ 175 Bloor St. E., 2nd floor (North Tower), Toronto, Ontario, Canada

ARTICLE INFO

Keywords:
Deep learning
Index-tracking portfolio optimization
Fixed noise

ABSTRACT

Index tracking portfolio optimization is popular form of passive investment strategy, with a steady and profitable performance compared to an active investment strategy. Due to the revival of deep learning in recent years, several studies have been conducted to apply deep learning in the field of finance. However, most studies use deep learning exclusively to predict stock price movement, not to optimize the portfolio directly. We propose a deep learning framework to optimize the index-tracking portfolio and overcome this limitation. We use the output distribution of the softmax layer from the fixed noise as the portfolio weights and verify the tracking performance of the proposed method on the S&P 500 index. Furthermore, by performing the ablation studies on the training-validation dataset split ratio and data normalization, we demonstrate that these are critical parameters for applying deep learning to the portfolio optimization problem. We also verify the generalization performance of the proposed method through additional experiments with another index of a major stock market, the Hang Seng Index (HSI).

1. Introduction

A fund is a financial instrument that collects investors' money and generates profits by operating assets such as stocks and bonds. There are two main types of investment strategies: active investment and passive investment.

An active investment fund aims to beat the average return on the stock market through careful stock selection. It assumes that investors can select stocks professionally to generate an excess return and construct a portfolio to beat the market (Zheng et al., 2019). However, its long-term performance is unstable because it can change dramatically over time.

In contrast, a passive investment fund simply aims to achieve the average market returns. Often referred to as a buy-and-hold strategy, it is less complex than active investment. Because many active investment funds have not outperformed the market in recent decades, a significant shift has occurred from active to passive investment strategy (Anadu, Kruttli, McCabe, Osambela, & Shin, 2019). Furthermore, a passive investment fund requires less investment research and often has a higher net income than an active investment fund (Busse, Goyal, Amit, Wahal, & Sunil, 2010). Index-tracking is a popular form of passive investment strategy and aims to track the performance of a benchmark index (Dose & Cincotti, 2005). Furthermore, index-tracking is a relatively stable and effective strategy compared with active investment (Ouyang, Zhang, & Yan, 2019).

Fig. 1 illustrates the taxonomy of the index-tracking portfolio optimization problem. The easiest way to construct an index-tracking portfolio is to hold whole stocks in the index at the relative rate within the index, referred to as full replication. However, research has been conducted primarily on partial replication (Dose & Cincotti, 2005; Shu, Shi, & Tian, 2020; Zhao & Palomar, 2019; Edirisinghe, 2013; Fang & Wang, 2005; Sant'Anna, Filomena, & Caldeira, 2017; Yu, Zhang, & Zhou, 2006) because full replication is not practical due to the high transaction cost. Furthermore, depending on whether rebalancing is considered, the problem can be categorized into a single-period portfolio optimization problem and a multi-period portfolio optimization problem.

In the single-period portfolio optimization problem, portfolio rebalancing is not considered. Accordingly, solving the multi-period portfolio optimization problem is more practical because we need to rebalance the portfolio to track the benchmark index more accurately. In the past several decades, numerous methods have been proposed for index-tracking portfolio optimization. Mathematical programming (especially integer programming and mixed-integer programming)

* Corresponding author.
[1] ORCID ID: https://orcid.org/0000-0002-4407-0348

https://doi.org/10.1016/j.eswa.2021.115298
Received 1 June 2020; Received in revised form 3 April 2021; Accepted 25 May 2021
Available online 4 June 2021

dominates the research (Strub & Baumann, 2018; Chen & Kwon, 2012; Sant'Anna et al., 2019; Gnägi & Strub, 2020; Canakgoz & Beasley, 2009). Although mathematical programming applies to small problems, it is challenging to solve real-world problems because the time complexity is too high to use in practice (Benidis, Konstantinos, Feng, Yiyong, & Palomar, 2018).

Recently, many studies have been conducted on meta-heuristics and deep learning. The majority of meta-heuristics studies solve the multi-objective problem rather than the single-objective problem (Chen et al., 2020; Ni & Wang, 2013; Chiam, Tan, & Mamun, 2013; Jürgen Branke, Scheckenbach, Deb, & Schmeck, 2009; Streichert et al., 2004). Moreover, most studies based on deep learning are prediction-based methodologies to predict stock prices (Sezer, Gudelek, & Ozbayoglu, 2020; Lv et al., 2019; Al-Thelaya, El-Alfy, & Mohammed, 2019; Abrishami, Turek, Choudhury, & Kumar, 2019) and optimize the portfolio with existing optimization methods based on the predicted price (Day & Lin, 2019). Although Heaton, Polson, and Witte, 2016; Ouyang et al., 2019 apply deep learning to optimize the index-tracking portfolio, they did not suggest a method to obtain the specific weight of each stock in the portfolio.

This study aims to develop a deep learning framework to optimize the index-tracking portfolio problem. We propose a neural network with fixed noise to optimize the portfolio weight of each stock and track the benchmark index. Using this model, we can directly optimize a portfolio for tracking the benchmark index.

We can construct the portfolio using the softmax probability from the trained model. The concept of using the softmax probability as the portfolio weight was proposed by Yun, Lee, Kang, and Seok, 2020. However, the primary difference is that we design the neural network to optimize the portfolio weight directly. Furthermore, we add fixed noise in the neural network. Without fixed noise, the portfolio weight changes every day during training, which violates the assumption that the portfolio weight should remain unchanged until the rebalancing period arrives. By using fixed noise, the portfolio weight of each stock remains unchanged even as the input data changes during training.—essential to the proposed model in solving the index-tracking portfolio optimization problem. The main contributions of this paper are as follows:

- We propose a neural network with fixed noise, which optimizes the weight of the portfolio to track the index directly.
- We perform additional experiments to reveal the impact of the critical parameters in index-tracking portfolio optimization problem such as the split ratio of the training-validation dataset.
- We illustrate the generalizability of the proposed model by performing the experiments with another dataset.

This paper proceeds as follows. In Section 2.1, we describe the problem that is addressed in this paper. In Section 2.2, we present a brief review of related studies, and in Section 2.3, we explain the motivations and the novelties of this study. We present the full details of the proposed methods in Section 3. In Sections 4.1 and 4.2, we summarize the data and experimental setting for this study, and in Section 4.3, we demonstrate the tracking performance of the proposed method. In Section 4.4, we investigate the impact of the data normalization and split ratio of the training-validation dataset through additional experiments. In Section 4.5, we demonstrate the generalizability of the proposed method by performing the experiments with another dataset. Finally, in Section 5, we discuss the experimental results and in Section 6 we conclude this study and propose future research.

2. Background

2.1. Problem description

Index-tracking portfolio optimization is a process of asset selection and asset allocation to minimize the tracking error between the constructed portfolio and the benchmark index (Prigent, 2007). The problem of asset allocation can be summarized as follows:

$$W^* = \underset{W\in\{w_1,w_2,\ldots,w_N\}}{\operatorname{argmin}} \frac{1}{T}\sum_{t=1}^{T}\left|C_t - \sum_{i=1}^{N} w_i c_{i,t}\right| \tag{1}$$

$$\textit{subject to} \quad \sum_{i=1}^{N} w_i = 1 \ \textit{and} \ 0 \leqslant w_i \leqslant 1 \ \textit{for} \ \forall i \tag{2}$$

N is the total number of the stocks included in the portfolio and T is the total duration of the dataset. Furthermore, w_i is the portfolio weight of asset i and W is a set of the portfolio weights for the assets included in the portfolio. $c_{i,t}$ is the cumulative daily change of asset i on day t, and C_t is the cumulative daily change of benchmark index on day t. The optimal portfolio weight of each stock W^* can be found with this equation, which minimizes the tracking error between the constructed portfolio and the benchmark index.

For full replication, N is the total number of the stocks included in the benchmark index, but for partial replication, we set the specific number N to construct the portfolio. Furthermore, for single-period portfolio construction, we only need to optimize the portfolio weight once, but for multi-period portfolio optimization, we need to set the specific rebalancing cycle and perform rebalancing periodically.

This paper aims to solve both the full replication problem and partial replication problem in a multi-period using a deep learning model. We use the output probability distribution of the softmax layer as portfolio weights and solve the partial replication problem by simply applying the model in two steps. Moreover, we evaluate the performance of the proposed model for partial replication for multiple portfolio sizes and rebalancing periods.

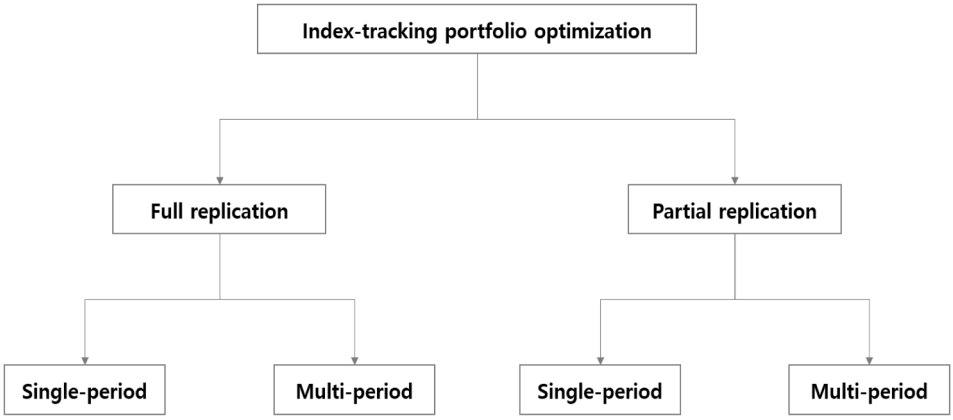

Fig. 1. Taxonomy of the index-tracking portfolio optimization problem.

2.2. Literature review of related studies

Research on index-tracking portfolio optimization can be categorized into three classes: mathematical programming, meta-heuristics, and deep learning.

Studies based on mathematical programming to solve the index-tracking portfolio optimization problem remain active. Based on the function used to measure the tracking error between the benchmark index and index-tracking portfolio, formulations can be categorized as either return-based or value-based (Strub & Baumann, 2018). In a return-based formulation, the objective is to minimize the function for historical return to optimize the index-tracking portfolio (Rudolf, Wolter, & Zimmermann, 1999; Gaivoronski, Krylov, & Van der Wijst, 2005). The return-based formulation assumes a "constant portfolio weight," which indicates the weights of stocks are constant over time. This assumption is required to calculate the return of the tracking portfolio as the convex function of the decision variable.

In a value-based formulation, the objective is to minimize the function of the historical trajectory of the price of the benchmark index and the index-tracking portfolio (Gaivoronski et al., 2005; Guastaroba & Speranza, 2012). The most significant difference between the return-based formulation and value-based formulation is whether the assumption of a constant portfolio weight is applied. There is no assumption of a constant portfolio weight in the value-based formulation because it minimizes the difference between the normalized prices.

Several studies have attempted to use meta-heuristics to optimize the portfolio to track the benchmark index. Ni and Wang (2013) used a hybrid genetic algorithm (GA) with a self-adaptive evolving mechanism that includes the Pareto efficiency as a utility measure. Chiam et al. (2013) optimized both tracking performance and transaction costs, demonstrating the reasonable tracking performance by evaluating in a single period and proving the generalization performance of the model by evaluating it in multiple periods. Sadjadi, Gharakhani, and Safari (2012) proposed a formulation, including uncertain data, to solve the asset allocation problem, which is a new framework for robust portfolio optimization under cardinality constraints. Díaz et al. (2019) proposed the hybrid model which combines a GA to select stocks in the portfolio with mixed-integer non-linear programming (MINLP) to estimate the stocks'weights.

In recent years, deep learning has exhibited state-of-the-art performance in diverse fields. However, in the field of portfolio optimization, the focus is on predicting stock prices using deep learning. A few studies have directly addressed portfolio optimization problems using deep learning. Yun et al. (2020) proposed a methodology for designing a joint cost function by exploiting the advantages of deep learning in managing a multi-objective function and used the deep learning model for prediction. Heaton et al. (2016) proposed the concept of a deep factor considering the non-linear relationship of the input data using an auto-encoder and solved both the index tracking problem and the enhanced index tracking problem. However, Heaton et al. (2016) estimated the portfolio price using a multi-layer perceptron, with no explanation on how to obtain the specific portfolio weight of each stock. In contrast, Ouyang et al. (2019) proposed a dynamic calculation method to interpret the effect of the stocks on the estimated portfolio price; however, it also did not calculate the portfolio weights for each stock.

2.3. Motivations and novelties of this study

Few studies have been conducted to solve the index-tracking portfolio optimization problem directly using the deep learning model. The previous studies on the use of deep learning for portfolio optimization are focused on predicting the stock price, not on constructing the portfolio. We propose a deep learning framework to solve the index-tracking portfolio optimization problem directly and overcome this limitation. In this framework, the neural network is used to optimize the portfolio weight directly to track the benchmark index and the portfolio weight of each stock can be calculated precisely.

Furthermore, few studies outline the critical parameters that must be considered for applying deep learning to optimize an index-tracking portfolio. Therefore, we perform ablation studies on the training-validation split of the dataset and data normalization to demonstrate that these are critical parameters when applying a deep learning model to a portfolio construction problem.

We evaluate both the full replication performance and the partial replication performance of the proposed model. Furthermore, with ablation studies, we demonstrate that several parameters have significant effects on the performance of the proposed model. We train the model using the sliding window method to reflect how the portfolio is managed in practice.

3. Method

3.1. Overview

Fig. 2 illustrates the process of portfolio construction and the rebalancing, proposed in this study. In full replication and partial replication, the input of the neural network, Deep NNF, is the cumulative change of each stock, and the output is the cumulative change of the constructed portfolio. The neural network is trained to minimize the difference between the cumulative change of the constructed portfolio and the cumulative change of the benchmark index.

For full replication, the portfolio is constructed from the softmax layer in the neural network. The output of softmax layer is used as the portfolio weight of each stock. When the rebalancing time arrives, the neural network is trained again with more recent data to rebalance the portfolio. For partial replication, the specific number of stocks are selected by the proposed model and the portfolio weights are optimized so that the portfolio can track the index. The given number of stocks are selected in descending order based on the portfolio weights of the neural network. The neural network is trained with the chosen companies again and the final portfolio is constructed. We describe this process in detail, including input data and network architecture in the following sections.

3.2. Input data

A change is the difference between today's close price and yesterday's close price. Instead of using change as the input for the proposed model, we use the cumulative change:

$$X_t = \begin{bmatrix} C_{1,t} \\ C_{2,t} \\ \cdots \\ C_{N,t} \end{bmatrix} \tag{3}$$

where $C_{i,t}$ is the cumulative change of the i^{th} stock at time t. Moreover, we can obtain the cumulative return by adding 1 to the cumulative change.

$$change_t = \frac{P_t - P_{t-1}}{P_{t-1}} \tag{4}$$

$$return_t = change_t + 1 \tag{5}$$

$$cumulative\ return_T = \prod_{t=1}^{T} return_t \ (= cumulative\ change_T + 1) \tag{6}$$

where P_t is the close price at time t and T is the total duration. For full replication, N is the total number of stocks in the benchmark index, and for partial replication, N is the number of stocks to include in the portfolio.

In the field of deep learning, z-normalization or min–max normalization is applied to the input data for aligning the input distribution from 0 to 1 or from −1 to 1. Cumulative change is a normalized version

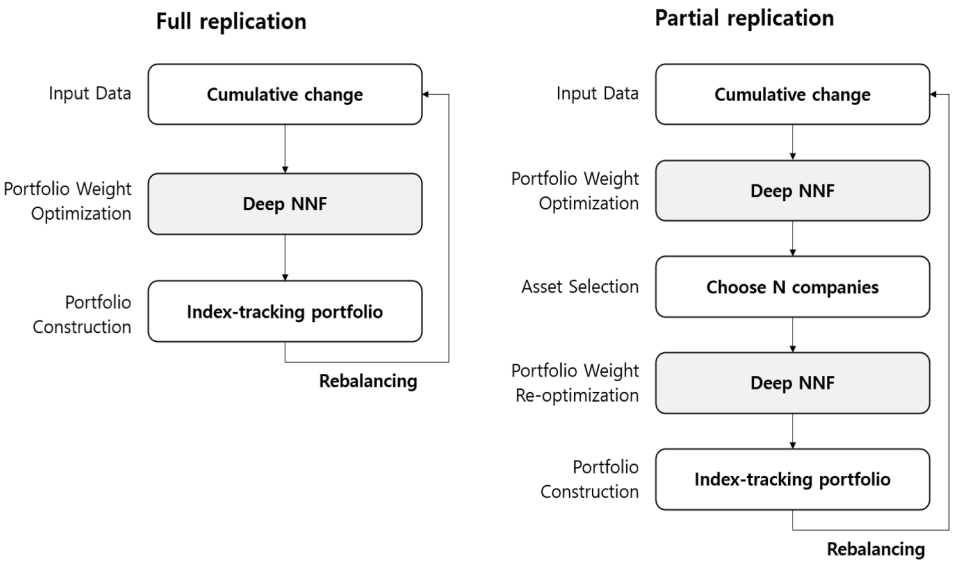

Fig. 2. Flow chart of the proposed model in this study.

of cumulative return. So the cumulative change is used as input data rather than cumulative return. Depending on whether data normalization is applied, the performance of the neural network can vary significantly and we perform a related ablation study about data normalization in Section 4.4.1.

3.3. Deep NNF: Deep Neural Network with Fixed noise

3.3.1. Portfolio weight from fixed noise

As depicted in Fig. 3, the input of the neural network is the cumulative daily change of the companies in the benchmark index. The output of softmax layer is used as the portfolio weight of each stock:

$$softmax(\widehat{y}_i) = \frac{\widehat{e^{y_i}}}{\sum_{j=1}^{N} \widehat{e^{y_j}}} \tag{7}$$

N is the size of the softmax layer, and $\widehat{y}_i$ is the i^{th} output value of the previous layer of the softmax layer. By performing elementwise-multiplication of the output of the softmax layer with the input data, and then summing the results, we can obtain the cumulative change of the constructed portfolio. The output of the softmax layer ranges from 0 to 1, and each element represents the portfolio weight of each stock. The loss of the neural network is the difference between the cumulative daily change of the constructed portfolio and the cumulative daily change of the benchmark index, referred to as the tracking error. Accordingly, we can optimize the portfolio to minimize the tracking error by training the neural network.

The motivation of this architecture is the attention mechanism (Bahdanau, Dzmitry, Cho, Kyunghyun, & Bengio, 2014) in deep learning, which is a recently proposed technique to focus on the most relevant the information in the input data, rather than using all available information. It has been successfully used for various problems (Yin, Schütze, Xiang, & Zhou, 2016; Gui et al., 2019). The output from the softmax layer in the attention module is referred to as the attention probability. Moreover, by multiplying the attention probability with the input data, we can obtain the attention input data, which highlights the information required to solve the task.

However, in contrast to the attention mechanism, the attention probability is calculated not from the input data, but from the fixed noise in the proposed model. Therefore, we name the output of the softmax layer in the proposed model "softmax probability," not attention probability. In the proposed model, the softmax probability is calculated from the fixed noise: if the softmax probability is calculated from the input data, it changes whenever the input data changes. Consequently, the portfolio weight changes every day during training. This violates the assumption that the portfolio weights are unchanged until the rebalancing period arrives.

The softmax probability used as the portfolio weight should not change even if the input data changes during training. As depicted in Fig. 3, the proposed model obtains the softmax probability not from the input data, but from the fixed noise, so we can solve the problem described previously. By applying elementwise-multiplication to the input data and softmax probability and adding them together, we can obtain the cumulative change of the constructed portfolio. The loss of the neural network, the tracking error between the constructed portfolio and the benchmark index, is now calculated correctly. Therefore, the model is trained to minimize the difference (e.g., mean absolute error) between the cumulative change of the constructed portfolio and the cumulative change of the benchmark index.

3.3.2. Neural network with fixed noise

For evaluating the performance of the proposed model—Deep Neural Network with Fixed noise (Deep NNF)—we use Shallow NNF as a baseline. The main difference between Shallow NNF and Deep NNF is the depth of the layer between the fixed noise and the softmax layer.

We set Shallow NNF as the baseline of Deep NNF because few studies are using deep learning in the index-tracking portfolio construction. Heaton et al. (2016) propose a non-linear portfolio that cannot calculate the portfolio weight of each stock, so it is not suitable as a comparison model for the Deep NNF. Kim and Kim (2020) and Ouyang et al. (2019) construct an index-tracking portfolio using deep learning, but constraints such as single-period or multi-period construction differ from those in study, so they are not the proper baseline for the proposed model.

As depicted in Fig. 3, the layers of the neural network (e.g. fully connected layer, activation layer) are located between the fixed noise and the softmax layer. Shallow NNF is a neural network consisting of three layers: fully connected layer, ReLU activation layer, and fully connected layer. In contrast, the proposed model, Deep NNF, has six fully connected layers and a ReLU activation layer between the fully connected layers. Because the neural network of Deep NNF is deeper than Shallow NNF, we apply dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) to each fully connected layer to prevent overfitting. Dropout is a technique to drop units from the neural network during training randomly, and it is a popular technique to address the overfitting problem of the neural network.

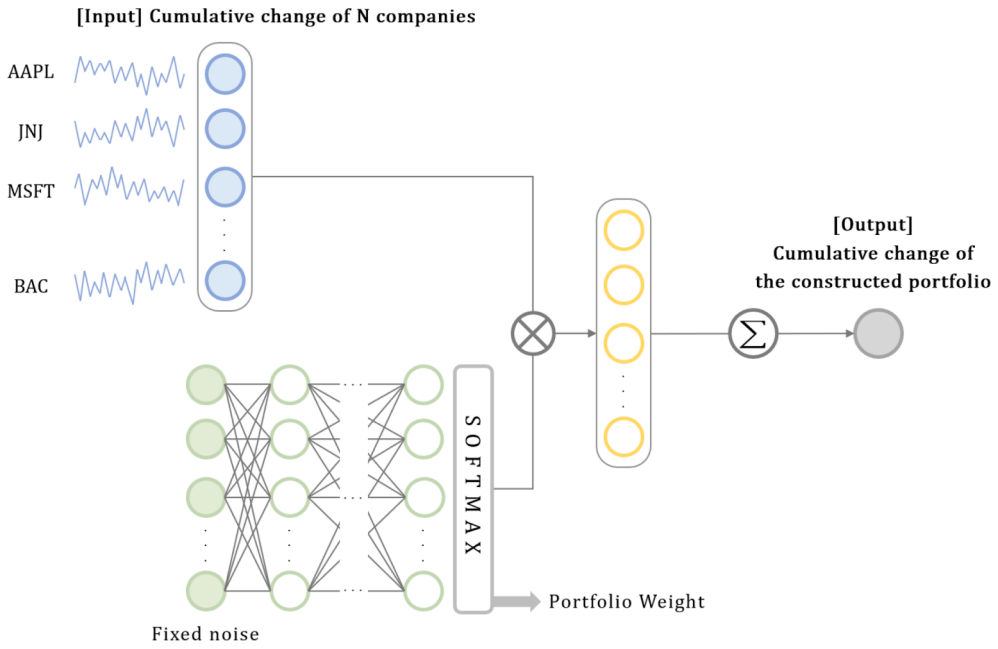

Fig. 3. Index-tracking Portfolio Optimization based on Neural Network with Fixed Noise.

4. Result

4.1. Data description

Stock price data included in the S&P500 index, which measures the stock performance of 500 large-cap U.S. companies, is used in the experiments. Among the indicators of stock price, the cumulative change is used in the experiments and the cumulative return is used for the comparative experiment. The formulas are described in (6). The data is obtained from Yahoo finance API[2] and the total period of the dataset is five and a half years, from July 1, 2014, to December 31, 2019. The total duration of the dataset is 1,385 days. Furthermore, we use 487 of the stocks included in the S&P 500 index, excluding those that were incorporated or excluded during the entire period. The base rebalancing period is one month, and we conducted additional experiments to verify whether the rebalancing period affects the tracking performance in this model. The test period is from January 1, 2018, to December 31, 2019, and the total duration is 502 days. As depicted in Fig. 4, because the test period includes the time that the trend of the S&P index falls, fluctuates, and rises, it is sufficient to illustrate the proposed model's robustness for the trend of the index.

We set an equally-weighted portfolio and shallow NNF as the baselines and conduct experiments to assess whether the proposed model is superior to the baseline. An equally-weighted portfolio is described as 1/N, which is a portfolio that includes the assets equally based on asset prices. The close price of each stock is used to calculate the cumulative daily change and the model is trained with the cumulative daily change.

4.2. Experimental setting

Sliding window method is used in the simulation to reflect the actual operation of the portfolio, as depicted in Fig. 5. By applying the sliding window method, the model is trained with the most up-to-date training data and reflect the recent trend of the market to construct the portfolio.

Training period is set to two and a half years, and the validation period is set to one year. Furthermore, we conduct an ablation study on the split ratio of the training-validation dataset. The total duration of the test dataset is two years, from January 1, 2018, to December 31, 2019, with 502 observations. With the rebalancing period of 1 month, a total of 24 models are trained using the sliding window technique for simulation during the test period, and with the rebalancing period of 3 months, a total of 8 models are trained. The model is trained for 50,000 epochs with a gradual warm-up learning rate scheduler. During training, the best model is determined when the validation loss is lowest. (Goyal et al., 2017).

$$RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left[C_t - \sum_{i=1}^{N} w_i c_{i,t}\right]^2} \tag{8}$$

$$MEAN(\bar{r}) = \frac{1}{T}\sum_{t=1}^{T} r_t \tag{9}$$

$$VOL = \frac{\sum_{t=1}^{T}(r_t - \bar{r})^2}{T} \tag{10}$$

T is the total duration and N is the total number of the stocks included in the portfolio. w_i is the portfolio weight of the asset i included in the portfolio. C_t and $c_{i,t}$ are the daily change of the benchmark index and the daily change of the asset i at time t, respectively. Moreover, r_t is the return or the constructed portfolio (or the benchmark index) at time t and $\bar{r}$ is the average daily return of the constructed portfolio (or the benchmark index).

Several metrics are used to evaluate tracking performance (Strub & Baumann, 2018): root mean-square error (RMSE) between the returns of the index and the portfolio as in (8), average daily return (MEAN) as in (9), volatility of the daily return (VOL) as in (10), and skewness (SKEW) and kurtosis (KURT) of the distributions of the returns of the index and the portfolio.

RMSE is the primary metric used to compare the tracking error between the benchmark index and the constructed portfolio. MEAN and VOL are the typical performance indicators of the portfolio. The similarity between the benchmark index and the constructed portfolio can be assessed by comparing their MEAN and VOL. SKEW and KURT are representative values of the distribution and are also supplementary indicators to assess the similarity between the benchmark index and the constructed portfolio. We assume that transaction cost is zero in both the baselines and the proposed model.

[2] https://github.com/FinanceData/FinanceDataReader.

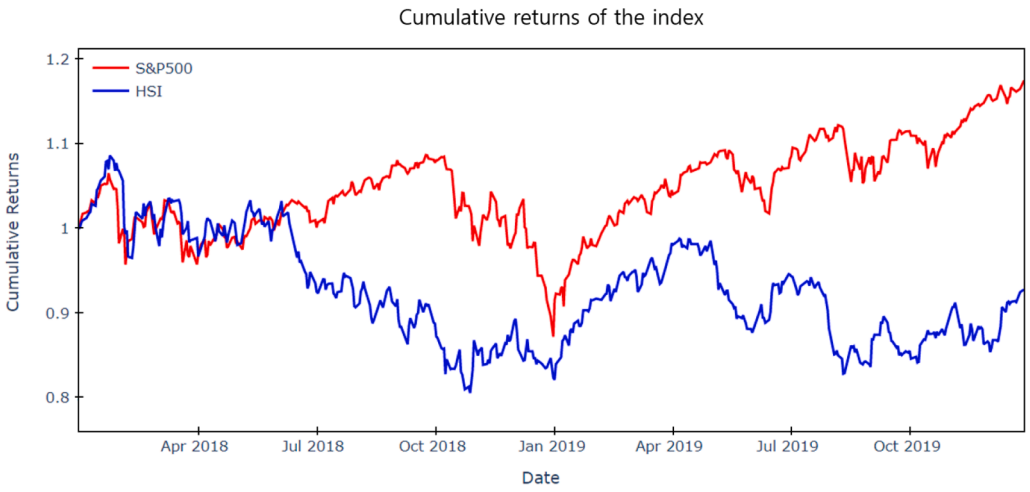

Fig. 4. Cumulative return of S&P 500 index and Hang Seng Index(HSI) index during the test period.

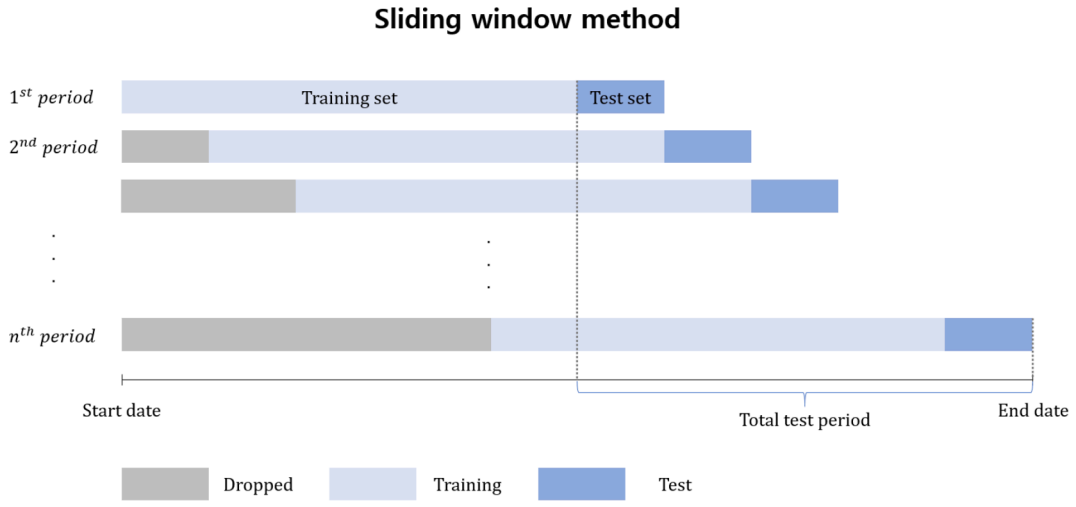

Fig. 5. Sliding window method for training the proposed model. At each rebalancing time, a new model is trained with recent data.

4.3. Performance evaluation of the proposed model

The proposed model is evaluated from two perspectives: full replication and partial replication. In Section 4.3.1, we evaluate the tracking performance of the model with full replication. In Section 4.3.2, we evaluate how the model's performance depends on the size of the portfolio with partial replication.

4.3.1. Tracking performance: Full replication

To evaluate the tracking performance of the proposed model, equally weighted portfolio and shallow NNF are used as the baseline. Furthermore, the experiments are conducted with the rebalancing periods of one month and three months, respectively.

Fig. 6 and Table 1 illustrate the tracking performance of the baseline models and the proposed model. RMSE of the proposed model is the smallest, both when the rebalancing period is one month and three months. This result illustrates that the tracking performance of the proposed model is superior to that of the baseline. Shallow NNF presents the worse performance than an equally weighted portfolio because it fails to be trained due to the low complexity of the model. Deep NNF addresses these shortcomings and improves tracking performance. Furthermore, this result indicates that tracking performance varies depending on the rebalancing period and the rebalancing period is the hyperparameter of this model that must be optimized.

4.3.2. Tracking performance: Partial replication

To evaluate the tracking performance of the proposed model for partial replication, the experiments comparing the tracking performance based on the size of the constructed portfolio are conducted. Table 2, Figs. 7 and 8 illustrate the tracking performance of the Deep NNF based on the size of the portfolio. We compare the proposed model for rebalancing periods of one month and three months, respectively.

When the rebalancing period is one month, the portfolio with 200 companies is the best tracking model. Moreover, compared to the portfolio with 487 companies, the number of stocks decreases by 58.9%, and the tracking performance is similar. When the rebalancing period is 3 months, the portfolio with 487 companies is the best tracking model, and the tracking performance is maintained similarly even if the number of stocks decreases to 100. In each rebalancing period, the tracking performance degrades rapidly when the size of the portfolio is smaller than the specific number, which can be inferred from the lack of stocks when following the index in the corresponding rebalancing period. The results of this experiment shows that using this simple methodology, we can reduce the size of the portofolio to 200 companies in a rebalancing period of 1 month and 100 companies in a rebalancing period of 3 months.

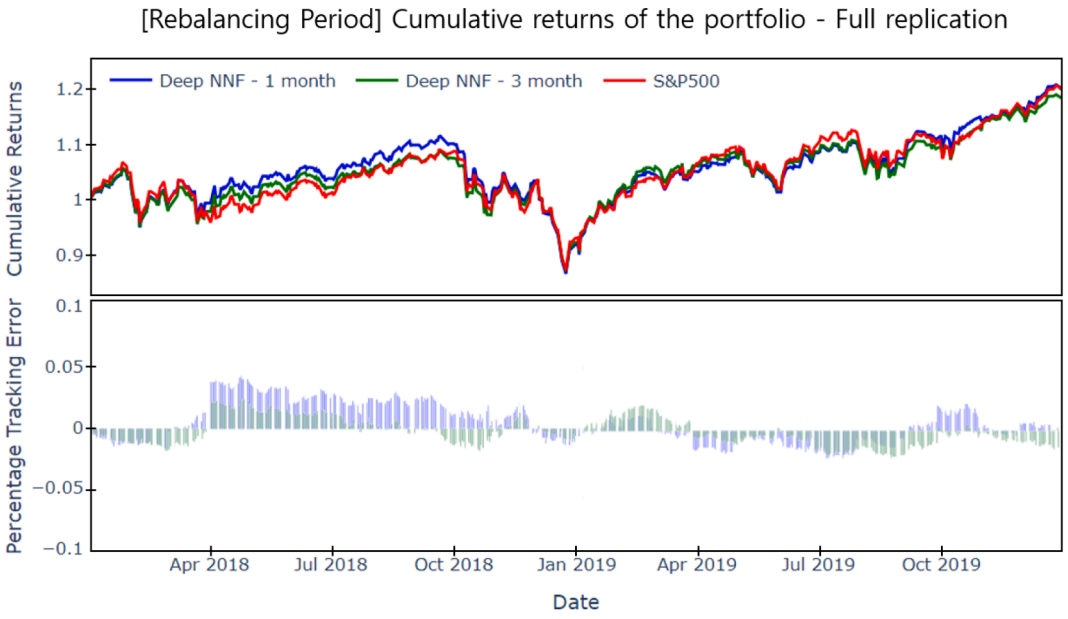

Fig. 6. Cumulative returns and percentage tracking error of each portfolio based on the rebalancing period. The evaluation was conducted over the test period, from January, 2018, to December 31, 2019, with 502 observations.

Table 1
Summary of simulation results for baseline models and proposed model when tracking the S&P 500.

Rebalancing period	Model	RMSE	MEAN	VOL	SKEW	KURT
1 month	1/N	0.0187	1.0557	0.0028	0.0611	0.7957
	Shallow NNF	0.0789	1.0421	0.0026	−0.0159	0.8560
	Deep NNF	**0.0168**	1.0550	0.0030	0.0983	0.6272
3 month	1/N	0.0162	1.0599	0.0028	0.1067	0.3361
	Shallow NNF	0.0203	1.0650	0.0023	−0.0675	0.5453
	Deep NNF	**0.0110**	1.0477	0.0027	0.1096	0.5118
	S&P 500	–	1.0495	0.0032	0.2351	0.0815

Bold indicates the best tracking performance among the models in each rebalancing period (1 month, 3 months). The benchmark index is the S&P 500, and evaluation is conducted over the test dataset, from January 1, 2018, to December 31, 2020, over 502 days. 1/N: equally-weighted portfolio; RMSE: root mean squared error between the returns of the index and the portfolio; MEAN: average daily return of the index and the portfolio; VOL: volatility of the daily returns of the index and the portfolio; SKEW (KURT): skewness (kurtosis) of the distributions of the returns of the index and the portfolio.

4.4. Ablation study

In this section, we investigate the impact of the data normalization and training-validation split ratio on tracking performance. In Section 4.4.1, we evaluate the tracking performance based on the type of input data from the perspective of data normalization in deep learning. In Section 4.4.2, we evaluate the tracking performance based on the split ratio of the training-validation dataset.

4.4.1. Data normalization

Either cumulative return or cumulative change can be used to train the proposed model. Cumulative change is exactly one smaller than cumulative return, as in Eq. (6). In the experiments that the cumulative return is used as input data, the base price of the portfolio is set to 1 at the start day of the training period. In the experiments that the cumulative change is used as input data, the base value of the portfolio is set to 0 at the start day of the training period.

Whether to use the cumulative return or cumulative change as input data is considered data normalization in deep learning. Data normalization is a basic technique to train the neural network efficiently. By subtracting 1 from the cumulative return, we can obtain the cumulative change and this process is similar to data normalization in setting the base of the data value to zero. The effectiveness of data normalization is summarized in Table 3; using cumulative change produces superior results regardless of the rebalancing period.

Table 2
Summary of simulation results for the proposed model according to portfolio size when tracking the S&P 500.

Rebalancing period	Portfolio size	RMSE	MEAN	VOL	SKEW	KURT
1 month	487	<u>0.0168</u>	1.0550	0.0030	0.0983	0.6272
	200	**0.0163**	1.0588	0.0034	0.4960	0.2130
	100	0.0325	1.0764	0.0043	0.3984	−0.1638
	50	0.0409	1.0852	0.0037	0.0720	−0.7796
3 month	487	**0.0110**	1.0477	0.0027	0.1096	0.5118
	200	0.0167	1.0378	0.0026	0.5595	1.2116
	100	<u>0.0139</u>	1.0515	0.0034	0.4807	0.0274
	50	0.0220	1.0368	0.0021	0.0993	1.1638
	S&P 500	–	1.0495	0.0032	0.2351	0.0815

Bold and underline indicate the best and the second-best tracking performance among the models in each rebalancing period. Portfolio size: the number of companies included in the portfolio;

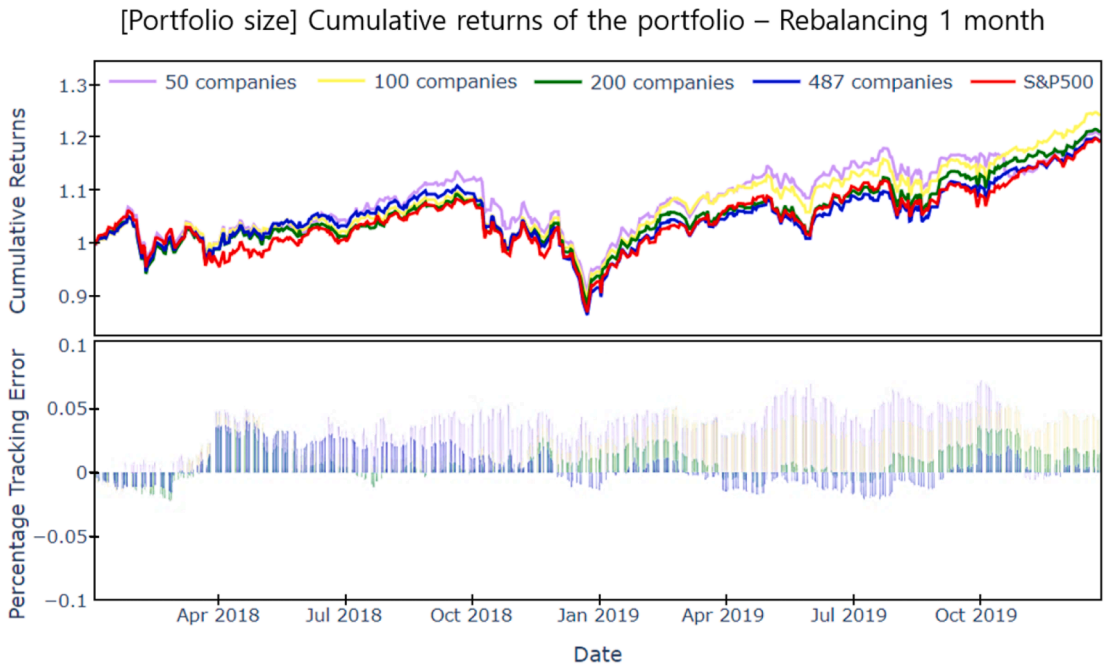

Fig. 7. Cumulative returns and percentage tracking error of each portfolio based on the portfolio size with 1 month of rebalancing period. The evaluation was conducted over the test period, from January 1, 2018, to December 31, 2019, with 502 observations.

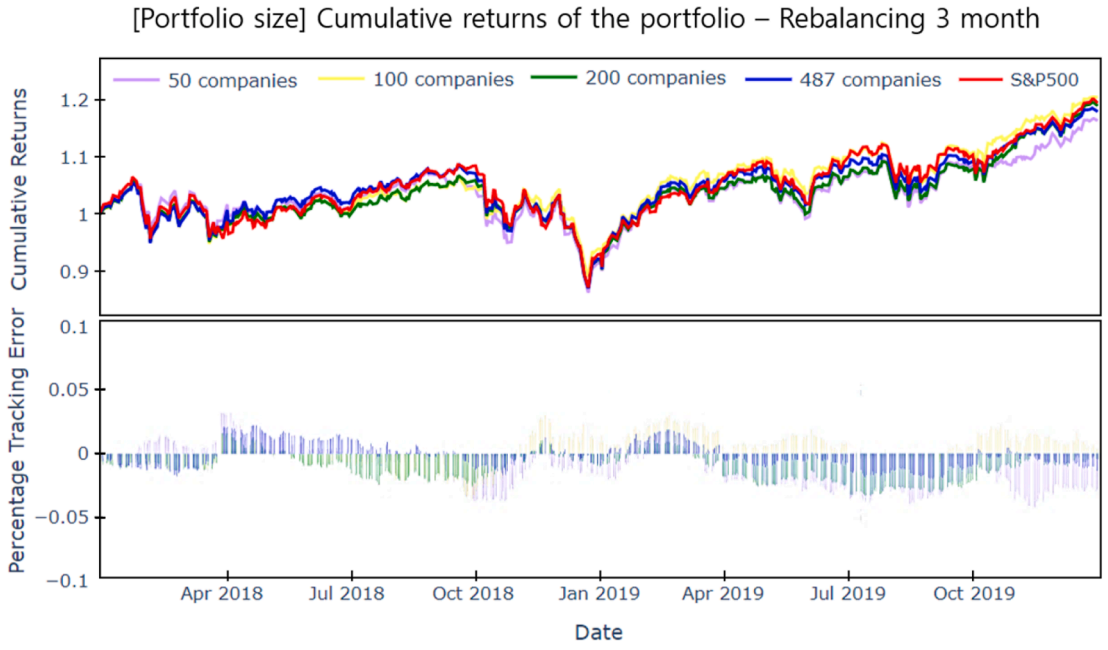

Fig. 8. Cumulative returns and percentage tracking error of each portfolio based on the portfolio size with 3 months of rebalancing period. The evaluation was conducted over the test period, from January 1, 2018, to December 31, 2019, with 502 observations.

4.4.2. Split ratio of training-validation dataset

The performance of the model can differ significantly based on the split ratio of the training dataset and the validation dataset. The split ratio has a significant impact on the performance of the model because the characteristics of the time series data vary based on the timing, especially for financial time series data. We perform additional experiments on the split ratio to verify whether it impacts the performance of the model. As presented in Table 4, the total duration of the training dataset and validation dataset is three and a half years. We divide the same dataset with three different split ratios. For example, for split A, the duration of the training dataset is three years, and the duration of the validation dataset is half a year. In this experiment, the number of stocks included in the portfolio is 487, and the rebalancing period is one month. The sliding window method is used to train the model. Fig. 9 and Table 5 illustrate the performance of the portfolio constructed with the model trained with a dataset having different training-validation split ratios. Among the three portfolios, the tracking performance of the portfolio with split B has the highest performance. This result illustrates that the split ratio of the training-validation dataset affects the tracking performance of the model significantly. According to these experimental results, we should consider the split ratio of the training-validation dataset as the hyperparameter, which must be optimized in the

Table 3
Summary of simulation results for the proposed model according to data normalization on tracking S&P 500.

Rebalancing period	Model	RMSE	MEAN	VOL	SKEW	KURT
1 month	Cumul change	**0.01680**	1.0550	0.0030	0.0983	0.6272
	Cumul return	0.01682	1.0485	0.0029	0.1057	0.7790
3 month	Cumul change	**0.01104**	1.0477	0.0027	0.1096	0.5118
	Cumul return	0.02111	1.0339	0.0023	−0.0454	0.8559
	S&P 500	–	1.0495	0.0032	0.2351	0.0815

Bold indicates the better performance among the model used the cumulative return and the model used the cumulative change. The model "Cumul change" uses the cumulative change as its input and output and the model "Cumul return" uses the cumulative return as its input and output.

Table 4
Description of split A, split B, and split C.

	Training	Validation	Total duration (year)
split A	3	0.5	3.5
split B	2.5	1	3.5
split C	2	1.5	3.5

The value in this table is on a yearly basis. We can train the model differently with the same dataset based on the ratio of splitting the dataset for training and validation.

portfolio optimization problem.

4.5. Additional experiment with Hang Seng Index

We also conduct the additional experiments on Hang Sang Index (HSI)—the major stock market's index—to confirm the generalization performance of this model. The dataset is obtained from Yahoo finance API. Based on the same approach as the experiments on S&P 500 dataset, 47 stocks among the 50 stocks included in the HSI are used in the experiment. The entire period is from July 1, 2014, to December 31, 2019 and the total duration is 1,356 days. The training period is set to two and a half years and the validation period is set to one year. The sliding method is used to construct the portfolio with the rebalancing period of one month.

The results of the experiment are as depicted in Fig. 10 and Table 6. The tracking error of Deep NNF is 0.0239, which illustrates that the tracking performance of the proposed model is superior to the baseline models of the equally weighted portfolio(1/N) and Shadow NNF.

5. Discussion

In this paper, we propose a new deep learning framework—Deep NNF—to construct and rebalance an index-tracking portfolio. We demonstrate that Deep NNF outperforms the baselines—equally-weighted portfolio and Shallow NNF—as described in Section 4.3. We design a model that can solve both full replication and partial replication for tracking the benchmark index. By applying a simple process, we can reduce the size of the portfolio (the number of stocks in the portfolio) efficiently for partial replication.

Furthermore, we explore the effect of data normalization and the split ratio of the training-validation dataset to the performance of the model in Section 4.4. Based on the results of these ablation studies, we suggest considerations when solving the index-tracking portfolio optimization problem with a deep learning framework. The crucial parameters are the split ratio of the training-validation dataset and the rebalancing period. Moreover, from the perspective of data normalization, we demonstrate that the performance of the proposed model is improved after changing the input and output data from the cumulative return to the cumulative change. Furthermore, to verify that the proposed model is generally applicable, we perform an additional experiment with the index of another major stock market, the Hang Seng Index (HSI).

However, while we demonstrate the performance of Deep NNF and perform the ablation study on data normalization and the split ratio of the training-validation dataset, this study has a few limitations. First, constraints of the portfolio, including transaction costs, are not considered. For transaction cost, we can add a penalty to the loss function without changing the neural network architecture, which has not been addressed in this study. Second, because we use time-series data, the rebalancing period and the split ratio of the training-validation dataset need to be optimized for improved tracking performance, but the specific method to optimize these parameters is not proposed. We perform a grid search, which is simply a complete search in the manually specified

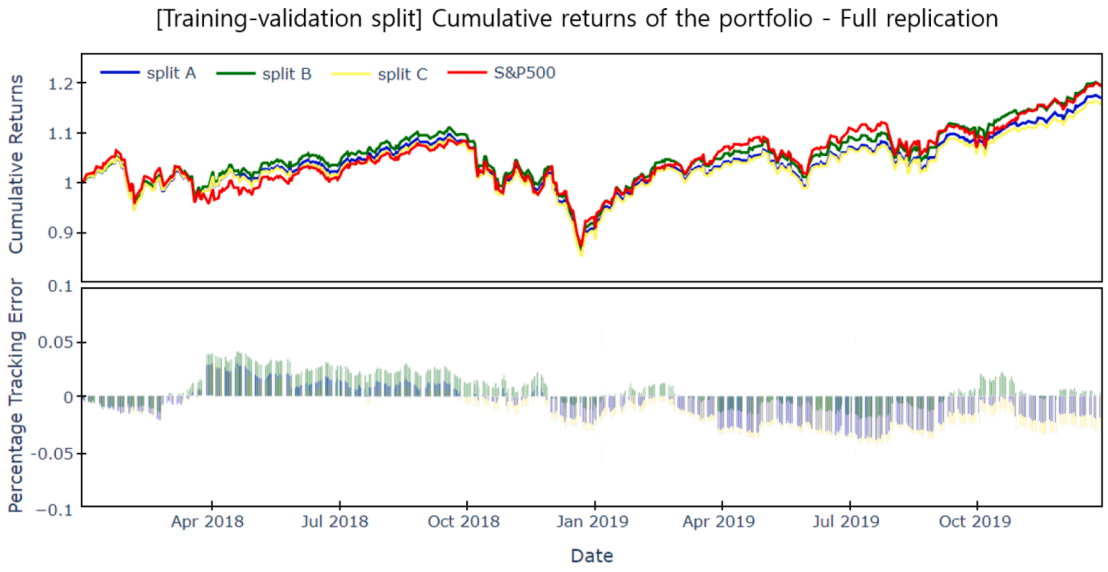

Fig. 9. Cumulative returns and percentage tracking error of each portfolio based on the split ratio of the training-validation dataset. The evaluation was conducted over the test period, from January 1, 2018, to December 31, 2019, with 502 observations.

Table 5
Summary of simulation results for the proposed model according to training-validation split ratio for tracking S&P 500 index.

Model	RMSE	MEAN	VOL	SKEW	KURT
split A	0.0200	1.0403	0.0026	−0.0215	0.8925
split B	**0.0168**	1.0550	0.0030	0.0983	0.6272
split C	0.0234	1.0349	0.0024	−0.1581	1.0690
S&P 500	–	1.0495	0.0032	0.2351	0.0815

Bold indicates the best tracking performance among the training-validation split of A, B, and C.

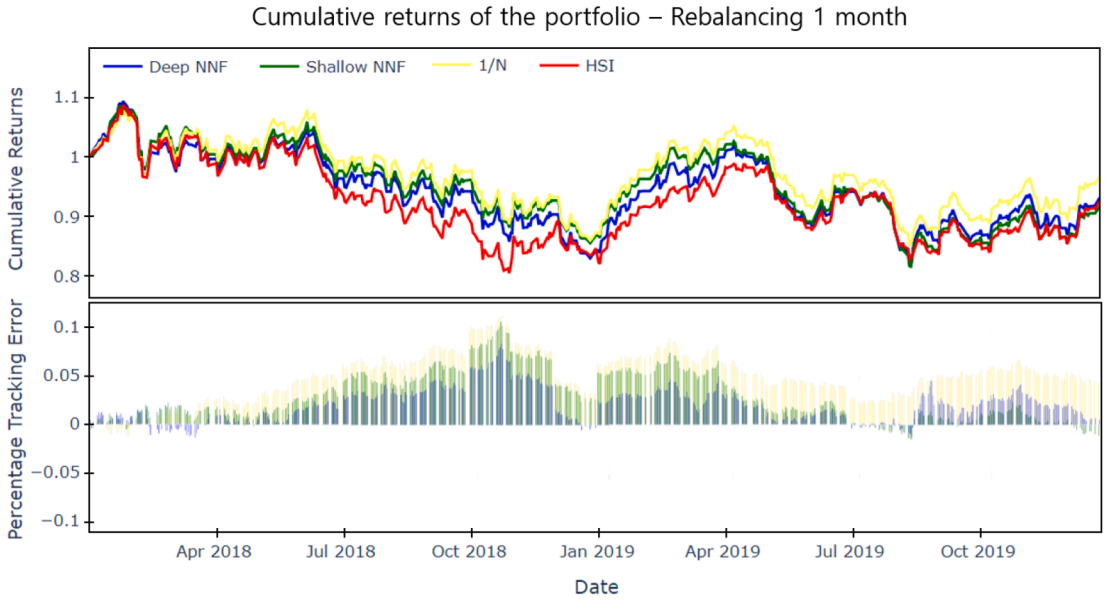

Fig. 10. Cumulative returns and percentage tracking error of each portfolio based on the rebalancing period. The evaluation was conducted over the test period, from January 1, 2018, to December 31, 2019, with 491 observations.

Table 6
Summary of simulation results for the proposed model when tracking the Hang Seng Index.

Model	RMSE	MEAN	VOL	SKEW	KURT
1/N	0.0489	0.9683	0.0028	0.0169	−0.9919
Shallow NNF	0.0347	0.9505	0.0036	0.0600	−1.0055
Deep NNF	**0.0239**	0.9427	0.0032	0.2510	−0.6859
Hang Seng Index	–	0.9245	0.0038	0.4140	−0.7196

Bold indicates the best tracking performance when tracking the Hang Seng Index.

ranges. Optimizing these parameters can be used to extend this study and improve the performance of the proposed model.

6. Conclusion

In this paper, we propose a neural network with fixed noise to solve the portfolio optimization problem. The performance and generality of the proposed method are demonstrated with simulation studies using the S&P 500 and HSI indexes. We demonstrate that both full replication and partial replication can be performed with the proposed model; moreover, we reveal the impact of data normalization and the split ratio of the training-validation dataset.

For future work, we aim to extend this study to construct the enhanced index-tracking portfolio, which outperforms the benchmark index. First, the tracking performance of the proposed model according to the time-series characteristics of training data, such as rising, descending, and fluctuating, should be analyzed to use the discovered pattern for enhancing the constructed portfolio. Second, because the neural network architecture of Deep NNF is simple, we can expand the input data including the cumulative change and train the model with more features. For example, we can simply add the input dimension with the other time-series features, such as the seven-day moving average of the stock.

CRediT authorship contribution statement

Yuyeong Kwak: Conceptualization, Investigation, Methodology, Writing - original draft, Writing - review & editing. **Junho Song:** Conceptualization, Methodology, Supervision, Writing - review & editing. **Hongchul Lee:** Supervision, Writing - review & editing.

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.

Acknowledgements

This research was supported by Brain Korea 21 FOUR, MODULABS and ZeroOne AI.

References

Abrishami, Soheila, Turek, Michael, Choudhury, Ahana Roy, & Kumar, Piyush (2019). Enhancing profit by predicting stock prices using deep neural networks. In *2019 IEEE 31st international conference on tools with artificial intelligence (ICTAI)* (pp. 1551–1556). IEEE.

Al-Thelaya, Khaled A., El-Alfy, El-Sayed M., & Mohammed, Salahadin (2019). Forecasting of bahrain stock market with deep learning: Methodology and case study. In *2019 8th International conference on modeling simulation and applied optimization (ICMSAO)* (pp. 1–5). IEEE.

Anadu, Kenechukwu, Kruttli, Mathias S., McCabe, Patrick E., Osambela, Emilio, & Shin, Chaehee (2019). The shift from active to passive investing: Potential risks to financial stability?. In *Available at SSRN 3244467*.

Bahdanau, Dzmitry, Cho, Kyunghyun, & Bengio, Yoshua. (2014). Neural machine translation by jointly learning to align and translate. In: arXiv preprint arXiv: 1409.0473 25.

Benidis, Konstantinos, Feng, Yiyong, Palomar, Daniel P., et al. (2018). Optimization methods for financial index tracking: From theory to practice. *Foundations and Trends in Optimization, 3*(3), 171–279.

Busse, Jeffrey A., Goyal, Amit, & Wahal, Sunil (2010). Performance and persistence in institutional investment management. *The Journal of Finance, 65*(2), 765–790.

Canakgoz, Nilgun A., & Beasley, John E. (2009). Mixed-integer programming approaches for index tracking and enhanced indexation. *European Journal of Operational Research, 196*(1), 384–399.

Chen, Bilian, Zhong, Jingdong, Chen, Yuanyuan (2020). A hybrid approach for portfolio selection with higher-order moments: Empirical evidence from Shanghai Stock Exchange. *Expert Systems with Applications, 145*, 113104.

Chen, Chen, & Kwon, Roy H. (2012). Robust portfolio selection for index tracking. *Computers & Operations Research, 39*(4), 829–837.

Chiam, Swee Chiang, Tan, Kay Chen, & Mamun, Abdullah Al (2013). Dynamic index tracking via multi-objective evolutionary algorithm. *Applied Soft Computing, 13*(7), 3392–3408.

Day, Min-Yuh, & Lin, Jian-Ting (2019). Artificial intelligence for ETF market prediction and portfolio optimization. In *Proceedings of the 2019 IEEE/ACM international conference on advances in social networks analysis and mining* (pp. 1026–1033).

Díaz, Juan, Cortés, María, Hernández, Juan, Clavijo, Óscar, Ardila, Carlos, & Cabrales, Sergio (2019). Index fund optimization using a hybrid model: Genetic algorithm and mixed-integer nonlinear programming. *The Engineering Economist, 64*(3), 298–309.

Dose, Christian, & Cincotti, Silvano (2005). Clustering of financial time series with application to index and enhanced index tracking portfolio. *Physica A: Statistical Mechanics and its Applications, 355*(1), 145–151.

Edirisinghe, N. C. P. (2013). Index-tracking optimal portfolio selection. *Quantitative Finance Letters, 1*(1), 16–20.

Fang, Yong, & Wang, Shou-Yang (2005). A fuzzy index tracking portfolio selection model. In *International conference on computational science* (pp. 554–561). Springer.

Gaivoronski, Alexei A., Krylov, Sergiy, & Van der Wijst, Nico (2005). Optimal portfolio selection and dynamic benchmark tracking. *European Journal of Operational Research, 163*(1), 115–131.

Gnägi, Mario, & Strub, Oliver (2020). Tracking and outperforming large stock-market indices. *Omega, 90*, Article 101999.

Goyal, Priya, Dollár, Piotr, Girshick, Ross, Noordhuis, Pieter, Wesolowski, Lukasz, Kyrola, Aapo, Tulloch, Andrew, Jia, Yangqing, He, Kaiming (2017). Accurate, large minibatch sgd: Training imagenet in 1 hour. arXiv preprint arXiv:1706.02677.

Guastaroba, Gianfranco, & Speranza, Maria Grazia (2012). Kernel search: An application to the index tracking problem. *European Journal of Operational Research, 217*(1), 54–68.

Gui, Ning, Ge, Danni, Hu, Ziyin (2019). AFS: An attention-based mechanism for supervised feature selection. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 33, pp. 3705–3713).

Heaton, J. B., Polson, Nicholas G., & Witte, J. H. (2016). Deep portfolio theory. In *arXiv preprint arXiv:1605.07230*.

Jürgen Branke, B., Scheckenbach, Michael Stein, Deb, Kalyanmoy, & Schmeck, Hartmut (2009). Portfolio optimization with an envelope-based multiobjective evolutionary algorithm. *European Journal of Operational Research, 199*(3), 684–693.

Kim, Saejoon, & Kim, Soong (2020). Index tracking through deep latent representation learning. *Quantitative Finance, 20*(4), 639–652.

Lv, Shaogao, Hou, Yongchao, & Zhou, Hongwei (2019). Financial market directional forecasting with stacked denoising autoencoder. arXiv preprint arXiv:1912.00712.

Ni, He, & Wang, Yongqiao (2013). Stock index tracking by Pareto efficient genetic algorithm. *Applied Soft Computing, 13*(12), 4519–4535.

Ouyang, Hongbing, Zhang, Xiaowei, & Yan, Hongju (2019). Index tracking based on deep neural network. *Cognitive Systems Research, 57*, 107–114.

Prigent, Jean-Luc (2007). *Portfolio optimization and performance analysis*. CRC Press.

Rudolf, Markus, Wolter, Hans-Jürgen, & Zimmermann, Heinz (1999). A linear model for tracking error minimization. *Journal of Banking & Finance, 23*(1), 85–103.

Sadjadi, Seyed Jafar, Gharakhani, Mohsen, & Safari, Ehram (2012). Robust optimization framework for cardinality constrained portfolio problem. *Applied Soft Computing, 12*(1), 91–99.

Sant'Anna, Leonardo R., Filomena, Tiago P., & Caldeira, Jo ao F. (2017). Index tracking and enhanced indexing using cointegration and correlation with endogenous portfolio selection. *The Quarterly Review of Economics and Finance, 65*, 146–157.

Sant'Anna, Leonardo Riegel, de Oliveira, Alan Delgado, Filomena, Tiago Pascoal, & Caldeira, Jo ao Frois (2019). Solving the index tracking problem based on a convex reformulation for cointegration. *Finance Research Letters*, p. 101356.

Sezer, Omer Berat, Gudelek, Mehmet Ugur, & Ozbayoglu, Ahmet Murat (2020). Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied Soft Computing, 90*, Article 106181.

Shu, Lianjie, Shi, Fangquan, & Tian, Guoliang (2020). High-dimensional index tracking based on the adaptive elastic net. *Quantitative Finance*, 1–18.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, & Salakhutdinov, Ruslan (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research, 15*(1), 1929–1958.

Streichert, Felix, Ulmer, Holger, & Zell, Andreas (2004). Evaluating a hybrid encoding and three crossover operators on the constrained portfolio selection problem. In *Proceedings of the 2004 congress on evolutionary computation (IEEE Cat. No. 04TH8753)* (Vol. 1, pp. 932–939). IEEE.

Strub, Oliver, & Baumann, Philipp (2018). Optimal construction and rebalancing of index-tracking portfolios. *European Journal of Operational Research, 264*(1), 370–387.

Yin, Wenpeng, Schütze, Hinrich, Xiang, Bing, & Zhou, Bowen (2016). Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics, 4*, 259–272.

Yun, Hyungbin, Lee, Minhyeok, Kang, Yeong Seon, & Seok, Junhee (2020). Portfolio management via two-stage deep learning with a joint cost. *Expert Systems with Applications, 143*, 113041.

Yu, Lian, Zhang, Shuzhong, & Zhou, Xun Yu (2006). A downside risk analysis based on financial index tracking models. In *Stochastic finance* (pp. 213–236). Springer.

Zhao, Ziping, & Palomar, Daniel P (2019). *Large-scale regularized portfolio selection via convex optimization*.

Zheng, Yu, Chen, Bowei, Hospedales, Timothy M., & Yang, Yongxin (2019). Index tracking with cardinality constraints: A stochastic neural networks approach. arXiv preprint arXiv:1911.05052.