Quantitative Finance

ISSN: 1469-7688 (Print) 1469-7696 (Online) Journal homepage: https://www.tandfonline.com/loi/rquf20

Index tracking through deep latent representation learning

Saejoon Kim & Soong Kim

To cite this article: Saejoon Kim & Soong Kim (2019): Index tracking through deep latent representation learning, Quantitative Finance, DOI: 10.1080/14697688.2019.1683599

To link to this article: https://doi.org/10.1080/14697688.2019.1683599

Published online: 18 Nov 2019.

Quantitative Finance, 2019
https://doi.org/10.1080/14697688.2019.1683599

Index tracking through deep latent representation learning

SAEJOON KIM* and SOONG KIM

Department of Computer Science and Engineering, Sogang University, 35 Baekbeom-ro, Mapo-gu, Seoul 04107, Korea

(*Received 24 January 2019; accepted 11 October 2019; published online 18 November 2019*)

We consider the problem of index tracking whose goal is to construct a portfolio that minimizes the tracking error between the returns of a benchmark index and the tracking portfolio. This problem carries significant importance in financial economics as the tracking portfolio represents a parsimonious index that facilitates a practical means to trade the benchmark index. For this reason, extensive studies from various optimization and machine learning-based approaches have ensued. In this paper, we solve this problem through the latest developments from deep learning. Specifically, we associate a deep latent representation of asset returns, obtained through a stacked autoencoder, with the benchmark index's return to identify the assets for inclusion in the tracking portfolio. Empirical results indicate that to improve the performance of previously proposed deep learning-based index tracking, the deep latent representation needs to be learned in a strictly hierarchical manner and the relationship between the returns of the index and the assets should be quantified by statistical measures. Various deep learning-based strategies have been tested for the stock market indices of the S&P 500, FTSE 100 and HSI, and it is shown that our proposed methodology generates the best index tracking performance.

Keywords: Index tracking; Index replication; Deep learning; Deep autoencoder; Stacked autoencoder

JEL Classification: C45, G11

1. Introduction

A market index is a representation of an underlying market through, in general, a market capitalization-weighted sum of a selected set of assets' values in the underlying market. As such, it is a good indicator of the condition of the underlying market and thus serves as a benchmark for various portfolios which are based on that underlying market. For example, when an active management portfolio is created, one of its goals invariably is to surpass the return of a market index. Although this may not sound like an extremely difficult task, empirical studies indicate that achieving this goal remains elusive to most active management investors if considered over a long period of time net of fees (Malkiel 1995, Montfort *et al.* 2008). One of the most well-known market indices is the S&P 500 which consists of a well diversified set of stocks listed on the New York Stock Exchange. While a large number of funds have been launched to beat this index, most of them have failed to do so over the long run, making the high research costs incurred in the process futile (Frino and Gallagher 2001).

For this reason, an investment vehicle that closely tracks the market index is highly sought after, and the strategy that generates such a vehicle has been termed *index tracking* or *index replication*. Thus, index tracking aims to follow the performance of a market index as closely as possible through the progression of time. It is a special form of passive management and portfolios managed through this strategy are called index funds. A measure of performance discrepancy between the market index and an index fund is called the tracking error, and one of the largest managers of S&P 500 index funds, Barclays Global Investors, which managed $22 billion to replicate the S&P 500 in the year 2000, achieved a 2.5% standard deviation of tracking error (Blume and Edelen 2002). Index funds not only have lower associated costs, that include managerial fees and transaction costs, than those of actively managed funds, but, by construction, they provide superior performance to many actively managed funds net of fees. Advantages of index funds over actively managed funds have been extensively documented in Dorocakova (2017). For these favorable reasons, index funds have become increasingly popular with about $2 trillion having been invested in US-based index funds alone as of 2015 (Strub and Baumann 2018).

*Corresponding author. Email: saejoon@sogang.ac.kr

Index tracking is conceptually a very simple strategy as buying every constituent in the market index in proportion to its market capitalization value would clearly 'replicate' the market index with zero tracking error. This implementation is called full replication. While this approach achieves perfect index replication in theory, it is rarely implemented in practice due to many unrealistic constraints that cannot easily be met. For the S&P 500, for example, the composition changed 60 times in the year 2000 (Beasley *et al.* 2003) and 235 times between the years 1995 and 2000 (Blume and Edelen 2002), which translates to transaction costs for the updates or a reduction in return for the index fund relative to the market index. Transaction costs can incur for a variety of reasons including the need to update the number of shares of an asset outstanding and to allocate new capital injections. Other reasons that make the full replication approach very difficult to implement include the lack of sufficient funds for perfect capital allocation with respect to each asset's market capitalization value and the wide bid-ask spread for less liquid assets. Practical considerations that render full replication unrealizable have been described in detail in Frino and Gallagher (2001).

To this end, a more implementable approach to index tracking exists and is called partial replication. This implementation targets a parsimonious index that approximates the market index, wherein the parsimonious index is obtained through constructing a tracking portfolio. The measure of approximation of the tracking portfolio to the market index can vary depending on the portfolio's objectives. A seminal work on partial replication index tracking is Roll (1992) in which the tracking portfolio that minimizes the volatility of the tracking error was considered. In Rudolf *et al.* (1999), a number of tracking error definitions were provided using linear error models to solve the index tracking problem. A main difficulty in the implementation of partial replication is that it implies a cardinality constraint on the tracking portfolio. As this yields a mixed integer quadratic programming problem which is well known to be intractable (Mutunge and Haugland 2018), an extensive body of research has been devoted into investigating an approximate solution of this problem. In Chang *et al.* (2000), cardinality-constrained portfolio optimization using heuristic algorithms has been presented, and in Jansen and van Dijk (2002), optimal index tracking with small portfolios was considered, wherein assumptions for selecting optimal parameters for optimization were employed. Applying the framework of Arbitrage Pricing Theory explains return as a linear function of factors (Ross 1976, Fama and French 1996); a factor-based approach to index tracking has been investigated in Corielli and Marcellino (2006). In Chen and Kwon (2012), an integer programming method was developed to construct a robust tracking portfolio through maximizing the similarity between the assets in the tracking portfolio and the index.

Constraints to index tracking other than the tracking portfolio's cardinality have been considered in the literature. One particular area with extensive research is the minimization of the transaction costs arising from frequent rebalancing. In Beasley *et al.* (2003), transaction costs incurred as rebalancing costs are incorporated into the objective function for tracking portfolio optimization. Gaivoronski and Krylov (2005) showed that their optimal portfolio rebalancing scheme is robust with respect to ranging values of transaction costs, specifically, 0.1%, 0.2% and 0.5%. Montfort *et al.* (2008) proposed a heuristics-based optimal sampling method that incorporates upper limits on transaction costs. Return curves generated under various transaction cost limits have been demonstrated in Guastaroba and Speranza (2012). In Strub and Baumann (2018), optimization of the tradeoff between tracking error and transaction cost has been investigated in detail. In particular, they proposed a new mixed integer linear programming formulation of the index tracking problem that satisfies various practical constraints that include a short-selling restriction to obtain tracking accuracy that is superior to those of previously considered formulations.

Another line of research in the direction of practical aproximations to the index tracking problem has been the development of machine learning algorithms. Machine learning algorithms have the ability to capture chracteristics of the problem not readily inferred from the traditional optimization frameworks. For this reason, a wide body of research has been devoted to machine learning-based index tracking. In particular, in Oh *et al.* (2005) a genetic algorithm was utilized to obtain an optimized tracking portfolio that produced improved performance of an index fund through particular chosen variables. In Beasley *et al.* (2003), an evolutionary heuristic was presented to generate the tracking portfolio, and more recently in Chiam *et al.* (2013), evolutionary computation was utilized to simultaneously optimize tracking performance and transaction costs. Machine learning algorithms have been synthesized for other areas in finance as well. For example, in Roh (2007), neural networks were used to forecast the volatility of stock prices, and in Fernandez and Gomez (2007), they were used for optimization in portfolio selection. In Kim (2018), support vector regression was utilized to generate low-volatility portfolios that beat portfolios optimized by traditional, i.e. not machine learning-based, approaches by a significant margin. In all machine learning-based constructions, the identification of more relevant characteristics of the problem has played the integral role in providing outperformance in comparison to the traditional optimization-based frameworks.

Most recently, with the advent of deep learning technologies in the field of artificial intelligence (Lecun *et al.* 2015), deep learning algorithms have also found applications in index tracking. Most notably, index tracking through deep learning was first conceived in Heaton *et al.* (2016) and this avenue of research presented as having great potential. Building on the framework of Heaton *et al.* (2016), Ouyang *et al.* (2019) recently presented a deep neural network along with a dynamic weight calculation method providing the weights on the edges of the network for index tracking of the Hang Seng Index (HSI). In that paper, in regard to asset selection for inclusion in the tracking portfolio, a basic deep autoencoder was constructed to identify assets whose original returns and the reconstructed returns are the most similar with respect to the Euclidean distance.

The main contribution of this paper is the application of the latest results from deep learning to execute an asset selection strategy defined for the tracking portfolio. We contribute to the literature in this paper by providing novel constructions for deep learning architectures explicitly designed for

the index tracking problem, along with effective algorithms for these architectures. Specifically, in Ouyang *et al.* (2019), the asset selection criteria was somewhat artificial as the value of the index played no role in training the deep autoencoder. Furthermore, the integral power of deep learning which lies in the learned 'deep representations' of the inputs for the problem has not been appropriately exploited as the architecture employed there was not fully hierarchical. We provide remedies for these deficiencies in this paper by constructing a hierarchically designed deep autoencoder whose deepest latent representation functions as a proxy for the market index. Moreover, novel measures of similarity of this latent representation of the market index with individual assets are presented as the criteria by which inclusion in the tracking portfolio is determined. To the best of our knowledge, neither the hierarchical architecture nor the similarity measure presented here has been previously considered for the index tracking problem. Empirical results indicate that these measures defined by the hierarchically learned latent representation play a crucial role in the provision of superior tracking performance relative to Ouyang *et al.* (2019). In particular, our index tracking strategy will be demonstrated to outperform the previously attempted deep learning-based strategies in all the cases we considered, and this improvement was larger than 33% in most cases.

The organization of this paper is as follows. In the next section, we formalize our problem and then give a brief summary of the previously attempted deep learning-based index tracking strategies of Heaton *et al.* (2016) and Ouyang *et al.* (2019). Section 3 presents the main contributions of this paper, and in section 4, we give performance results of our proposed index tracking strategies and compare them with those of existing strategies. We provide the conclusions of this paper in section 5.

2. Preliminaries

Let us first give the definition of the return of a (market) index on day t, R_t, which is as follows:

$$R_t = \sum_{a\in\mathcal{I}} w^*_{a,t} r_{a,t}, \tag{1}$$

where

$$\begin{aligned} w^*_{a,t} &= \text{weight of asset } a \text{ on day } t \\ \mathcal{I} &= \text{set of assets in the index pool} \\ r_{a,t} &= \text{return of asset } a \text{ on day } t \end{aligned}$$

and

$$\sum_{a\in\mathcal{I}} w^*_{a,t} = 1, \quad w^*_{a,t} \geq 0 \quad \text{for all } a, t. \tag{2}$$

Equation (1) indicates that the return of the index is determined solely by how the weight of each asset is defined given the returns of assets in the index pool.

2.1. Problem formulation

The problem of index tracking is basically constructing a tracking portfolio that is parsimonious such that the difference between returns of the index and the tracking portfolio is minimized. In other words, the goal is to select the assets and their weights under a cardinality constraint so that the return of the obtained tracking porfolio follows closely to that of the index. This is normally solved by finding the tracking portfolio with the described attribute from the historical, or the formation period's, data. The measure of this closeness is determined by the tracking error which has a number of different definitions in the literature, and we will consider the one which is the sum of the squared deviations as in Montfort *et al.* (2008). Specifically, the goal is to select a small subset of the assets in the index pool $\mathcal{I}$ so that the following mean squared error (MSE) minimization is attained:

$$\begin{aligned} \min_{w_{a,1}} \frac{1}{T}\sum_{t=1}^{T}\left(R_t - \sum_{a\in\mathcal{I}} w_{a,t} r_{a,t}\right)^2 \\ \text{such that } |\{a \in \mathcal{I} : w_{a,1} \neq 0\}| \leq K, \end{aligned} \tag{3}$$

where

$$w_{a,t} = \frac{(1 + r_{a,t-1})w_{a,t-1}}{\sum_{a'\in\mathcal{I}}(1 + r_{a',t-1})w_{a',t-1}} \quad \text{for } t \geq 2,\ a \in \mathcal{I} \tag{4}$$

and $w_{a,t}$ is the weight of asset a on day t. We note that $w^*_{a,t}$'s in equation (1) are the deterministic values specified by how the market index is defined, and $w_{a,1}$'s in equation (3) are the decision variables. Here, $t = 1$ and T represent the first and the last days of the formation period, respectively, and K is the constraint on the cardinality that is a parameter of the problem. For this index replication, the two constraints of equation (2) need also be satisfied with $w^*_{a,t}$ replaced by $w_{a,t}$. Above minimization problem of equation (3) is a mixed integer quadratic programming problem which is well known to be computationally infeasible (Mutunge and Haugland 2018). The hardness of the problem stems from the cardinality constraint, and for this reason, extensive research based on heuristics has been performed as described in the previous section.

In this paper, we take the approach to solving the index tracking problem through the latest developments from deep learning. Results presented in Heaton *et al.* (2016), Ouyang *et al.* (2019) shed light to the deep learning approach to have the potential to produce excellent tracking performance, and a brief description of this is the topic of next subsection. We note that a more direct way to solving the index tracking problem is to choose the largest composition assets in the index. However, in some cases, investing in assets that are constituents of the index is not possible or the exact index composition may not be known. For example, a Spot Energy Index contains components that are not traded in an exchange (Andriosopoulos and Nomikos 2014) and one needs to look to assets in other exchanges in this case which our approach is capable of handling. In this respect, our approach provides a more general solution to the index tracking problem as it can be applied to settings in which the index composition is not completely known or some index constituents are not tradeable.

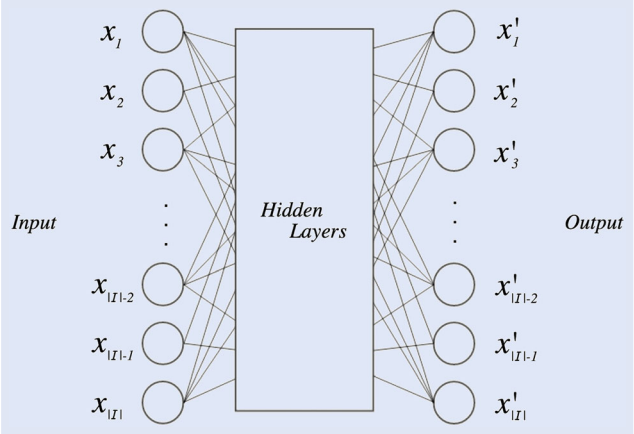

Figure 1. General deep autoencoder.

2.2. Asset selection using deep autoencoder

One approach to providing a solution to the aforementioned intractable problem is to construct a deep neural network architecture to identify the most relevant assets. One architecture that stands out for this purpose is autoencoders, or *deep* autoencoders if the autoencoder has many layers, whose goal is to reconstruct at the output the input as closely as possible. In other words, if x is the input to the autoencoder and x' is the output from the autoencoder, the goal of the autoencoder is to obtain x' through a set of nonlinear transformations of x so that the distance of the two becomes very small. We require the transformations to be nonlinear so that training of the network does not result in a trivial transformation such as the identity mapping.

For such an autoencoder, the 'representativeness' of the input (Bengio *et al.* 2013) can be interpreted to have been well-preserved throughout its propagation in the network, first being compressed as the propagation approaches the middle layer and then being expanded to produce the reconstruction as it approaches the other side of the network. This characteristic of the autoencoder has been exploited in Heaton *et al.* (2016), Ouyang *et al.* (2019) to identify the most relevant assets for index tracking, and the general architecture used there is shown in figure 1. The middle part of the network consisting of hidden layers has not been specified because different implementations can be used. For example, in Heaton *et al.* (2016), a single hidden layer consisting of 4 nodes was used, and in Ouyang *et al.* (2019), three hidden layers with 1 node in the middle layer has been adopted.

In figure 1, $x_i \in \mathbb{R}^T$ represents the daily returns vector of the ith asset for the time period of T days and is the input to the ith node on the left, and $x'_i \in \mathbb{R}^T$ is the output from the ith node on the right and it represents the reconstruction of x_i, $i \in \mathcal{I}$. In figure 1 and throughout this paper, x and x' represent cross-sectional data per day, i.e. $x, x' \in \mathbb{R}^{|\mathcal{I}|}$, and $x_1, x_2, \ldots$ represent time-series data for 1st, 2nd, $\cdots$ assets, respectively. In Heaton *et al.* (2016), Ouyang *et al.* (2019), assets for which the mean squared error of the reconstruction, i.e.

$$\|x_i - x'_i\|_2^2, \quad i \in \mathcal{I}, \tag{5}$$

takes among the K smallest values is selected as the constituent for the tracking portfolio of size K as those assets were claimed to have higher 'communal' information content.

Once the constituents of the portfolio are selected, equal-weighted tracking portfolio was used in Heaton *et al.* (2016) and a deep neural network was utilized for weight calculation in Ouyang *et al.* (2019). However, the weight calculation method of the latter produced weights of negative value which implies shorting some of the assets in the tracking portfolio is required. Therefore, this method seems to be more applicable to synthetic index tracking as it does not conform to the traditional implementation of partial replication. Nonetheless, both schemes in Heaton *et al.* (2016), Ouyang *et al.* (2019) have shown strong tracking portfolio performances, and for this reason, we have pursued further investigation on this topic of index tracking through deep learning.

3. Index tracking by deep neural networks

In this section, we describe a number of ways by which asset allocation strategies can be created through deep neural networks. The goal of these strategies is to select the assets for inclusion in the tracking portfolio. We will present a discussion on selecting the weights for these assets in the last part of this section.

As presented in the previous section, the deep neural network that appropriately suits the goal of index tracking is deep autoencoders. In this section, we will investigate the efficacy of deep autoencoder and its variant along with a new objective function, and of new criteria for asset selection that have not been presented before for the index tracking problem.

3.1. Deep autoencoder

For simplicity and to possess symmetry property, we have constructed a 3-hidden layer deep autoencoder shown in figure 2 which is essentially the architecture proposed in Ouyang *et al.* (2019). It shows that $x \in \mathbb{R}^{|\mathcal{I}|}$ is the input representing the returns of the $|\mathcal{I}|$ assets and $y \in \mathbb{R}^{\lfloor 0.25 \cdot |\mathcal{I}| \rfloor}$, $z \in \mathbb{R}^{\lfloor 0.0625 \cdot |\mathcal{I}| \rfloor}$ and $\omega \in \mathbb{R}$ are the latent representations of the input in which z represents a 'deeper' representation than y and ω represents a deeper one than z. So, in this deep autoencoder architecture, ω is the deepest latent representation of x. The dimensions of y and z were slighly arbitrarily chosen so that they do not decrease too abruptly going from the dimension $|\mathcal{I}|$ of x to dimenstion 1 of ω. $x' \in \mathbb{R}^{|\mathcal{I}|}$ is the output representing the reconstruction of the input x.

The figure indicates that if x' that is a close approximation of x can be generated, then ω can be viewed as a good simple description of x. This is the key idea that we base on for the asset selection criterion in the tracking portfolio. In particular, the figure indicates that ω can be interpreted as the market value from which each coordinate of x' can be generated. In a way, this ω serves an analogous role as the market factor in the Capital Asset Pricing Model (Sharpe 1964, Fama and French 2004) which states that each asset's price can be expressed as a function of the market value. For this reason,

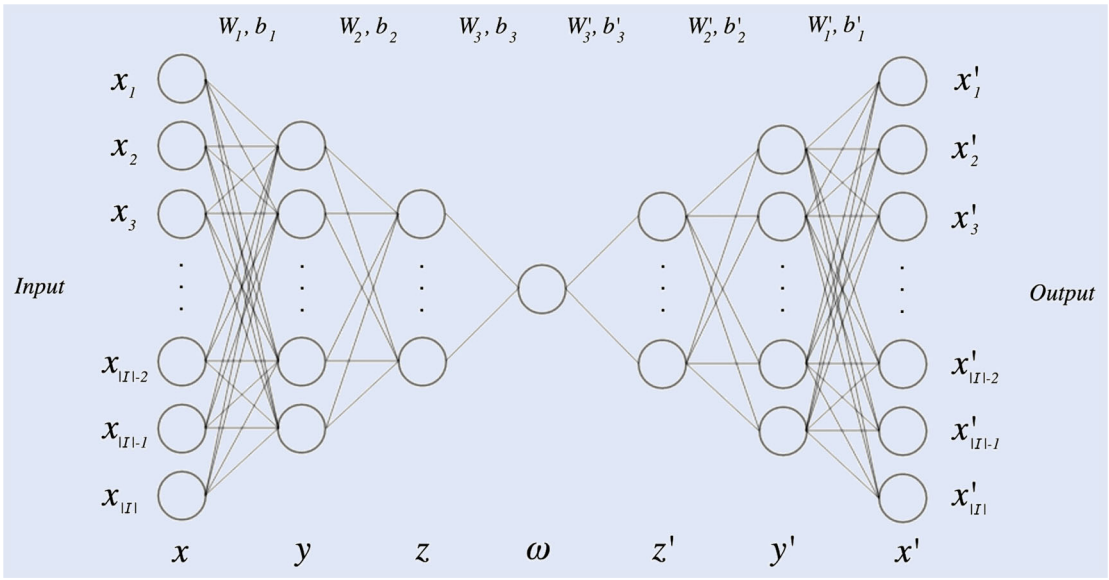

Figure 2. Proposed deep autoencoder architecture.

in our proposed architecture, the center layer necessarily has exactly one node.

The variables x, y, z, ω, z', y' and x' are related by the following set of equations:

$$\begin{aligned} y &= f(W_1 x + b_1) \\ z &= f(W_2 y + b_2) \\ \omega &= f(W_3 z + b_3) \\ z' &= f(W_3' \omega + b_3') \\ y' &= f(W_2' z' + b_2') \\ x' &= f(W_1' y' + b_1') \end{aligned}$$

wherein f is some activation function such as the sigmoid function or the rectified linear unit. We used the former as its nonlinearity better suits in the deep autoencoder architecture for financial data (Bao *et al.* 2017, Ouyang *et al.* 2019, Kim 2019). In particular, we have used the hyperbolic tangent function $f(x) = (e^x - e^{-x})/(e^x + e^{-x})$ for $f(\cdot)$ as its range from -1 to 1 better represents the daily returns than the more commonly used sigmoid function which is the logistic function whose range is between 0 and 1. $W_1, W_2, W_3, W_1', W_2', W_3'$, collectively denoted by W, are weight matrices and $b_1, b_2, b_3, b_1', b_2', b_3'$, collectively denoted by b, are weight vectors to be found. The goal of the deep autoencoder is to train the network so that the output produces a close replica of the input through extracting nonlinear features of the input in each hidden layer. In other words, to train the autoencoder, the following optimization is carried out:

$$\min_{W,b} \|x - x'\|_2^2. \tag{6}$$

A regularization term such as $\lambda \left(\sum_{i=1}^{3} \|W_i\|_F^2 + \|W_i'\|_F^2\right)$ can be added to this objective function to avoid possible overfitting, as was done in Ouyang *et al.* (2019), where λ is the regularization constant and $\|\cdot\|_F^2$ is the Frobenius norm. However, in our constructions, where tens or hundreds of input nodes are compressed to a single node before reconstruction, overfitting was not an issue. In fact, including the regularization term in the objective function had no effect in performance, so we suppressed this term in the objective function throughout our experiments.

The aim of index tracking is to construct the tracking portfolio that approximates the index. As the single node in the middle layer, namely, ω, can be interpreted as the market index, in our proposed tracking portfolio construction, we pick the ith asset for which a similarity between the return vector of the ith asset, x_i, and the latent vector ω is among the K largest. To this end, for variables x_i and ω, we employ the notions of *correlation coefficient*, $\rho_{x_i\omega}$, which has been demonstrated to be an effective similarity measure for asset selection application in Kim and Kim (2018), and *mutual information*, $I(x_i; \omega)$, which is a widely used measure to represent the relationship between two random variables (Kingma and Welling 2014) to represent the similarity. For two return vectors x_i and ω, the correlation coefficient is defined as

$$\rho_{x_i\omega} = \frac{\sigma_{x_i\omega}}{\sigma_{x_i}\sigma_{\omega}}, \tag{7}$$

where the numerator represents the covariance of variables x_i and ω, and the denominator represents the product of the standard deviations of the two variables. The mutual information for the two return vectors is defined as

$$I(x_i; \omega) = \sum_{x_i} \sum_{\omega} p(x_i, \omega) \log \frac{p(x_i, \omega)}{p(x_i)p(\omega)}, \tag{8}$$

where $p(x_i, \omega)$ represents the joint probability distribution of x_i and ω, and $p(\cdot)$ represents the marginalized probability distribution for $\cdot$. Note that correlation coefficient takes values between -1 and 1 with the former and the latter representing the highest negative and highest positive correlation, respectively, and mutual information assumes the value 0 if x_i and ω are probabilistically independent.

Summing up, in one of our asset selection strategies, we pick the K assets for which the correlation coefficient $\rho_{x_i\omega}$

between the return vector x_i and the latent vector ω is one of the K largest values for $i \in \mathcal{I}$. In another asset selection strategy, we pick the K assets for which the mutual information $I(x_i;\omega)$ between the two arguments is one of the K largest values for $i \in \mathcal{I}$. We note that we only take this new asset selection criteria as a part of our asset allocation strategy and we will use a different implementation, to be presented in the next subsection, for the deep autoencoder. This modification of the deep autoencoder is necessary since ω as trained by the deep autoencoder just described does not capture the property of the market index as well as ω of the next subsection.

Note carefully that equation (5) of the previous works (Heaton *et al.* 2016, Ouyang *et al.* 2019) does not capture this relationship between the return vector of an asset and that of the latent variable. It only calculates the similarity between the return vector of an asset and its reconstructed version which is not addressing the relationship between an asset and the market index. This replacement of equation (5) by equations (7) and (8) as the criteria for asset selection is one of our main contributions of this paper. Equations (7) and (8) show that exactly one node needs to exist in the middle layer of the deep autoencoder in constrast to those employed in Heaton *et al.* (2016), Ouyang *et al.* (2019) which does not provide such guidance. The overall asset selection algorithm using the 3-layer deep autoencoder is stated in Algorithm 1. In Algorithm 1, and also in Algorithm 2 to be shown in the next subsection, $x_{(t)}$ represent the cross-sectional data x on day t. So parenthesis was used to distinguish the time index from the asset index. In Algorithm 1, lines $1 \sim 11$ describe the process by which the deep autoencoder is trained. In particular, these lines show that, in this algorithm, the deep autoencoder is not trained layer-wise but all of its layers are trained simultaneously. Lines $12 \sim 14$ describe how the latent vector ω is calculated, and the remainder of the algorithm describes how the K constituents of the tracking portfolio are selected.

Algorithm 1 CC/MI-3LAE

Require: W_l's and b_l's, initial decoder's parameters. W_ls and b_ls, initial encoder's parameters. M, the minibatch size. T, train data set size. CC/MI, correlation coefficient or mutual information.

1: **while** objective function has not converged **do**
2: Randomly select minibatch of M examples $(x_{(t_1)}, x_{(t_2)}, \ldots, x_{(t_M)})$ from train dataset.
3: **for** k from 1 to M **do**
4: $\omega_{(k)} \leftarrow tanh(W_3 \cdot tanh(W_2 \cdot tanh(W_1 \cdot x_{(t_k)} + b_1) + b_2) + b_3)$
5: $x'_{(k)} \leftarrow tanh(W_1' \cdot tanh(W_2' \cdot tanh(W_3' \cdot w_{(k)} + b_3') + b_2') + b_1')$
6: **end for**
7: Calculate loss of the decoder: $D_{loss} \leftarrow \frac{1}{M}\sum_{k=1}^{M} \|x'_{(k)} - x_{(t_k)}\|_2^2$
8: Calculate loss of the encoder: $E_{loss} \leftarrow \frac{1}{M}\sum_{k=1}^{M} \|x'_{(k)} - x_{(t_k)}\|_2^2$
9: Update W_l's and b_l's of the decoder with respect to D_{loss}
10: Update W_ls and b_ls of the encoder with respect to E_{loss}
11: **end while**
12: **for** t from 1 to T **do**
13: Calculate encoded representation: $\omega_{(t)} \leftarrow tanh(W_3 \cdot tanh(W_2 \cdot tanh(W_1 x_{(t)} + b_1) + b_2) + b_3)$
14: **end for**
15: $\omega \leftarrow \{\omega_{(t)}\}_{t\in[1,T]}$
16: **for** $a \in \mathcal{I}$ **do**
17: $x_a \leftarrow \{r_{a,t}\}_{t\in[1,T]}$
18: Calculate CC/MI(x_a, ω)
19: **end for**
20: select K assets of largest CC/MI(x_a, ω), $a \in \mathcal{I}$.

3.2. *Stacked autoencoder*

In the stacked autoencoder (SAE), in contrast to the deep autoencoder of the previous subsection, the network is trained layer-wise (Bengio *et al.* 2006). The SAE that we used for our asset selection strategy is shown in figure 2. It has the same architectural layout as before, however, the training of the network is processed differently to better extract deep features.

First, the first and the last layers of figure 2 involving W_1, W_1' and b_1, b_1' only are trained to obtain y from x. Rephrased, a single-hidden layer autoencoder with the parameters W_1, W_1' and b_1, b_1' is constructed to produce the output x' from the input x through the hidden layer whose nodes are collectively labelled as y. Then, the second and the second to the last layers involving W_2, W_2' and b_2, b_2' only are trained to obtain z from y where y is obtained from the previous training step. Finally, the third and the fourth layers involving W_3, W_3' and b_3, b_3' only are trained to obtain ω from z which is obtained from the previous training step. Therefore, the training process for the stacked autoencoder is more hierarchical than that for the deep autoencoder of the previous subsection, and it is this characteristic of SAE that establishes ω to possess superior latent representation of x, and hence of its individual components x_i, $i \in \mathcal{I}$, to ω of the previous subsection.

To summarize, for our proposed asset selection strategy, the described SAE is implemented as the deep neural network in our deep learning architecture instead of the 3-layer deep autoencoder of the previous subsection. After three separate training steps are completed, the latent variable ω is calculated, and the constituents of the tracking portfolio are selected according to the condition of equations (7) or (8). This implementation of SAE as the deep autoencoder constitutes the other part of our main contributions.

To examine the performance characteristics driven by the architecture and the similarity measure employed in our deep autoencoders rather than by parameter optimization, not much effort has been put into optimizing the tunable parameters. As a consequence, we believe some of the improvements presented in this paper can even further be stretched. The overall asset selection algorithm using the stacked autoencoder is shown in Algorithm 2. The parameter values and the normalizations used, that will be described in the next section, in Algorithm 2 are the same as those used in Algorithm 1.

The main difference in Algorithm 2 lies in lines $6 \sim 18$ that roughly correspond to lines $3 \sim 10$ in Algorithm 1. In Algorithm 1, in each while loop, ω's are computed before

the deep neural network is trained for optimization. In Algorithm 2, however, in each while loop, each layer in the deep neural network is trained for optimization before ω's are computed. Therefore, in Algorithm 2, the ω's are computed in a more hierarchical manner which contributes to generating better deep representation of each x_i, $i \in \mathcal{I}$.

Algorithm 2 CC/MI-SAE

Require: W_l's and b_l's, initial decoder's parameters. W_ls and b_ls, initial encoder's parameters. c, index regression indicator. M, the minibatch size. T, train dataset size. CC/MI, correlation coefficient or mutual information.

while objective function has not converged **do**
  Randomly select minibatch of M examples $(x_{(t_1)}, x_{(t_2)}, \ldots, x_{(t_M)})$ from train dataset.
  for k from 1 to M **do**
    $x''_{(k)} \leftarrow x_{(t_k)}$
  end for
  for l from 1 to 3 **do**
    for k from 1 to M **do**
      $z_{(k)} \leftarrow tanh(W_l \cdot x''_{(k)} + b_l)$
      $x'_{(k)} \leftarrow tanh(W'_l \cdot z_{(k)} + b'_l)$
    end for
    Calculate loss of the decoder: $D_{loss} \leftarrow \frac{1}{M} \sum_{k=1}^{M} \|x'_{(k)} - x''_{(t_k)}\|^2$
    Calculate loss of the encoder: $E_{loss} \leftarrow \frac{1}{M} \sum_{k=1}^{M} \|x'_{(k)} - x''_{(t_k)}\|^2$
    Update W_l's and b_l's of the decoder with respect to D_{loss}
    Update W_ls and b_ls of the encoder with respect to E_{loss}
    for k from 1 to M **do**
      $x''_{(k)} \leftarrow tanh(W_l \cdot x''_{(k)} + b_l)$
    end for
  end for
end while
for t from 1 to T **do**
  Calculate encoded representation: $w_{(t)} \leftarrow tanh(W_3 \cdot tanh(W_2 \cdot tanh(W_1 \cdot x_{(t)} + b_1) + b_2) + b_3)$
end for
$w \leftarrow \{w_{(t)}\}_{t \in [1,T]}$
for $a \in \mathcal{I}$ **do**
  $x_a \leftarrow \{r_{a,t}\}_{t \in [1,T]}$
  Calculate CC/MI(x_a, w)
end for
select K assets of largest CC/MI(x_a, w), $a \in \mathcal{I}$.

3.3. Weight selection

In this subsection, we discuss some of the weight selection schemes to possibly improve the equal-weighted scheme employed in Heaton *et al.* (2016) or the scheme of Ouyang *et al.* (2019) that necessitates negative weights or short selling. A simple alternative to equal weights are those defined by 'correlation' of returns which has been utilized before, for example, in Chen and Kwon (2012).

In this paper, we will use the correlation coefficient for this purpose and, in particular, in one of our weight selection schemes, more weight was given to assets in the tracking portfolio that are more correlated with other assets in the index pool. This scheme identifies these assets with high correlation coefficient as being more representative of the index pool and therefore we endow higher weight to those assets. Specifically, in this scheme, we have defined the weight of asset a, w_a, in the tracking portfolio $\mathcal{TP}$ as the following: $w_a = k \sum_{b \in \mathcal{I}} e^{\rho_{ab}}$ where k is the normalizing constant such that $\sum_{a \in \mathcal{TP}} w_a = 1$. We exponentiated the correlation coefficient so that the value becomes non-negative and thus, in this expression, negative correlation coefficient implies less correlation than zero correlation coefficient.

In the other weight selection scheme that utilizes the correlation coefficient, more weight was given to assets that are less correlated with other assets in the tracking portfolio. In other words, we defined the weight of an asset to be inversely proportional to its correlation with other assets in the tracking portfolio so that similar assets in terms of returns do not dominate the behavior of the entire tracking portfolio. This scheme was chosen to complement the aforementioned scheme. Specifically, in this scheme, we define the weight of asset a in the tracking portfolio as the following: $w_a = k \sum_{b \in \mathcal{TP}} (1/e^{\rho_{ab}})$.

For another alternative to equal weights, we computed the weights by optimizing the objective function of equation (3) using only the tracking portfolio assets' returns. Specifically, once the constituents of the tracking portfolio are determined by the foregoing asset selection algorithms, the minimization of the objective function of equation (3), i.e. the left half of equation (3) only, becomes a quadratic programming problem for which computationally efficient ways to solve exist. Thus, in this weight selection scheme, the weights of assets in the tracking portfolio are defined by those that minimize the objective function of equation (3) using only these tracking portfolio assets' returns.

In particular, in accordance with the notation of equation (3), if $\{w_{a,T}\}_{a \in \mathcal{TP}}$ represent the optimized weights of the tracking portfolio assets at the end of the formation period, these become the weights defined for the holding period.

In summary, in this paper, we tested the described three weight selection schemes along with the equal-weighted scheme in which $w_a = 1/K, \forall\, a \in \mathcal{TP}$. It turns out that the two correlation coefficient-derived unequal-weighted schemes generated roughly equal weights in the experiments we tested in this paper. In particular, for one data set, one of the two unequal-weighted schemes produced slightly better results than the equal-weighted scheme while in another data set, the opposite was true. For the weight selection scheme defined by solving the quadratic programming problem, the associated tracking portfolio performed inferior to that of the equal-weighted scheme for many of the experiments conducted while it performed superior for the rest of the experiments in which tracking portfolio size was larger. It seems that if the portfolio size is smaller, then optimizing the weights in the formation period has the undesirable effect of overfitting. As all three unequal-weighted schemes did not generate substantial difference or superiority with respect to the equal-weighted scheme in the tracking portfolio performance, all

results shown in this paper have been generated from the simplest equal-weighted scheme.

4. Tracking portfolio performance

In this section, we present the performance results of tracking portfolios constructed by various strategies we considered in this paper. For completeness, we include the performance results of strategies that are neither the baseline strategy nor our proposed strategy to confirm the underlying rationale of our proposed constructions. The daily return of a portfolio on day t is given as follows:

$$\sum_{a\in\mathcal{I}} w_{a,t}^{(j)} r_{a,t}$$

in which the weights $w_{a,t}^{(j)}$, $j \in$ {a,b,c,d,e,f}, are determined through various tracking portfolio strategies. In fact, only $w_{a,1}^{(j)}$, where $t = 1$ denotes the first day of the holding period, is determined per strategy and $\{w_{a,t}^{(j)}\}_{t\geq 2}$ are updated similarly as in equation (4). Specifically, for a fixed cardinality constraint K, the tracking porfolio strategies under consideration are the following:

(a) 3-Layer Autoencoder
(b) CC-3-Layer Autoencoder
(c) MI-3-Layer Autoencoder
(d) Stacked Autoencoder
(e) CC-Stacked Autoencoder
(f) MI-Stacked Autoencoder

In this list, strategy (a) is the deep autoencoder strategy of Ouyang *et al.* (2019) that used the architecture of figure 2 with asset selection criterion of equation (5). This will be the baseline strategy that our proposed strategies will be compared and constrasted with. Strategy (b) is the strategy that uses the deep autoencoder architecture of figure 2 with asset selection criterion of equation (7). Strategy (c) is the same as strategy (b) except with asset selection criterion of equation (8). We remark that the criteria of equations (7) and (8) were not proposed to be used in conjunction with the deep autoencoder architecure described in section 3.1. Therefore, we do not expect strategies (b) and (c) to improve the performance of the strategy (a). Strategies (b) and (c) were included, in fact, to illustrate this point.

Strategy (d) is the stacked autoencoder (SAE) described in section 3.2 with asset selection criterion of equation (5). This strategy was included to illustrate that SAE alone may not be sufficient to improve the performance of the baseline strategy (a). In fact, SAE does not match naturally with the criterion of equation (5). However, when the same SAE is used in conjuction with the asset selection criterion of equations (7) or (8), index tracking performance improvement is expected which was the motivation of this paper. Therefore, the final two strategies in the list are the proposed asset selection strategies of this paper. Strategy (e) is the SAE with asset selection criterion of equation (7) and strategy (f) is the SAE with asset selection criterion of equation (8). We will demonstrate shortly that these strategies beat the baseline strategy by a significant margin in all data sets considered in this paper.

Specifications of the parameter selection in Algorithm 1 and Algorithm 2 are in order. The minibatch size M was set to 200 and the train dataset size T was 504 or 2 years of daily data as mentioned previously. Elements in the matrix W and the vector b were selected at random from the uniform distribution in the interval $[-(1/\sqrt{M}), 1/\sqrt{M}]$ which is the default setting in PyTorch (Paszke *et al.* 2017). We used the Adam optimizer (Kingma and Ba 2015) for the optimization in lines 7 and 8 in Algorithm 1 and lines 11 and 12 in Algorithm 2. Just before calculating for the correlation coefficient of equation (7) or the mutual information of equation (8), we applied normalization so that both x_a and ω have zero mean and unit variance. Finally, to calculate for the mutual information of equation (8), we divided the real line into ten adjoining intervals of endpoints at $-3, -1.75, -1, -0.5, 0, 0.5, 1, 1.75$, and 3. To examine the performance characteristics driven by the selection of our deep autoencoder rather than by parameter optimization, we have not put too much effort in optimizing the parameters such as the interval selection in the mutual information calculcation.

The data set used to generate the performance results of the strategies is the following: S&P 500 of the US, FTSE 100 of the UK, and HSI of Hong Kong. The source of S&P 500 and HSI data sets was Yahoo! Finance (finance.yahoo.com) and that of FTSE 100 was Investing.com (investing.com). For all data sets, the experiments were performed for a single-period index tracking of the holding period of one year from 2017/11/1 to 2018/10/31 using the immediately preceeding two years from 2015/11/1 to 2017/10/31 as the formation period. As only the assets whose prices were available over the entire formation and holding periods are included in the index pool, the number of constituents in the index pool was 497, 95, and 49 for the S&P 500, FTSE 100 and HSI data set, respectively, in our experiments. If validation period was necessary as part of the experiment which was true in some cases, we have used the immediately preceeding one year to the aforementioned holding period as the validation period and the immediately preceeding two years to this validation period as the formation period.

As a side note, we have also performed a multiple-period index tracking over the same holding period divided into four equal-length periods. Specifically, a four quarter-length periods index tracking has been tested with the rebalancing cost of 0.2% incurred at the end of each quarter-length period. However, the results were not nontrivially different from those of the single-period experiments to be demonstrated in section 4, and therefore, we included only the single-period experimental results in this paper.

4.1. Root mean squared error and tracking error volatility

As a measure of tracking error, a widely used one is the root mean squared error (RMSE) of returns which is defined as

$$\left(\frac{1}{T}\sum_{t=1}^{T}\left(R_t - \sum_{a\in\mathcal{I}} w_{a,t}^{(j)} r_{a,t}\right)^2\right)^{1/2}. \tag{9}$$

Table 1. S&P 500 index tracking RMSE (%) and TEV (%).

	(K_1)	RMSE	TEV	(K_2)	RMSE	TEV	(K_3)	RMSE	TEV
(a)		0.3428	0.3427		0.2963	0.2963		0.2590	0.2586
(b)		0.6712	0.6685		0.5527	0.5504		0.4137	0.4114
(c)		0.6129	0.6084		0.4682	0.4652		0.3668	0.3648
(d)		0.3779	0.3778		0.3299	0.3298		0.3026	0.3022
(e)	⋆	**0.2843**	**0.2842**	⋆	**0.2477**	**0.2476**		0.2153	0.2151
(f)		0.3000	0.2991		0.2735	0.2730	⋆	**0.1993**	**0.1993**

Table 2. FTSE 100 index tracking RMSE (%) and TEV (%).

	(K_1)	RMSE	TEV	(K_2)	RMSE	TEV	(K_3)	RMSE	TEV
(a)		0.5010	0.5010		0.4429	0.4428		0.3641	0.3635
(b)		0.6333	0.6332		0.5031	0.5031		0.3948	0.3945
(c)		0.7651	0.7651		0.4743	0.4742		0.3702	0.3702
(d)		0.5702	0.5702		0.4303	0.4300		0.3141	0.3139
(e)		0.5029	0.5029	⋆	**0.2729**	**0.2728**	⋆	0.2736	0.2735
(f)	⋆	**0.3942**	**0.3939**		0.3143	0.3142		**0.2508**	**0.2506**

In equations (9) and (10) below, $t = 1$ and T represent the first and the last days of the holding period, respectively. This RMSE provides a direct measure of the deviation of the return of the tracking portfolio from that of the index. Another popular measure is the volatility of deviation of returns, also sometimes referred to as the *tracking error volatility* (TEV). TEV is defined as

$$\left(\frac{1}{T}\sum_{i=1}^{T}\left(R_t - \sum_{a\in\mathcal{I}} w_{a,t}^{(j)} r_{a,t} - E\left[R_t - \sum_{a\in\mathcal{I}} w_{a,t}^{(j)} r_{a,t}\right]\right)^2\right)^{1/2} \tag{10}$$

and it provides with the standard deviation property of the tracking error. So, RMSE in combination with TEV provides a good succinct measure of the deviation of returns as was also mentioned in Gnagi and Strub (2019). We list below the values of both tracking error measures for the index tracking strategies discussed in this paper.

Table 1 shows the tracking error values for the S&P 500 data set. For this data set, we let $K_1 = 0.05{*}|\mathcal{I}|$, $K_2 = 0.1{*}|\mathcal{I}|$, $K_3 = 0.2{*}|\mathcal{I}|$. So the first part of the table shows the values when the cardinality constraint $K = K_1$, the middle part when $K = K_2$, and the last part when $K = K_3$. Note that strategies (b) and (c) do not necessarily improve the strategy of (a), however, the strategies (e) and (f) do improve the strategy of (d) as we expected. In particular, comparing the baseline strategy of (a) with our proposed strategies (e) and (f), our strategies both beat the strategy of (a) for all cases of K considered. To choose between our proposed two strategies, we utilized the validation period and the strategy that was selected for execution in the holding period is marked with ⋆ in the table. Furthermore, the lowest value in each column of the table is shown in boldface font. This marking by ⋆ and boldface font is applied to all three tables presented in this section. Consequently, the table indicates that our proposed strategy for each K produced the best result amongst all strategies considered. The numbers shown in the table are in percentages and therefore they represent very small values.

Table 2 shows the tracking error values for the FTSE 100 data set. As the number of index constituents for this data set is smaller than that for S&P 500, we let $K_1 = 0.1{*}|\mathcal{I}|$, $K_2 = 0.2{*}|\mathcal{I}|$, $K_3 = 0.3{*}|\mathcal{I}|$ for this data set. As in table 2, strategies (b) and (c) do not improve the strategy of (a), however, the strategies (e) and (f) do improve the strategy of (d). In particular, comparing the baseline strategy of (a) with our proposed strategies (e) and (f), our strategies both beat the strategy of (a) by significant margins, for all cases of K considered. As in table 1, we utilized the validation period to choose between our two proposed strategies, and the one selected is denoted by ⋆. Our proposed strategy marked by ⋆ for each K produced either the best or the near-best result amongst all strategies considered.

Table 3 shows the tracking error values for the HSI data set. For this data set, we let $K_1 = 0.1{*}|\mathcal{I}|$, $K_2 = 0.2{*}|\mathcal{I}|$, $K_3 = 0.3{*}|\mathcal{I}|$ as in table 2. For this data set, strategies (b) and (c) do improve the strategy of (a) in most cases unlike the previous two data sets. However, the values achieved by our proposed strategies (e) and (f) beat those achieved by strategies (b) and (c). Of course, our proposed strategies (e) and (f) improve the values achieved by strategy (d). Comparing the baseline strategy of (a) with our proposed strategies (e) and (f), both of our strategies again beat the strategy of (a). As before, we utilized the validation period to choose between our two proposed strategies and denoted the one selected by ⋆. As in table 2, our proposed strategy marked by ⋆ produced either the best or the near-best result amongst all strategies considered for every K.

Tables 1–3 indicate that whether strategies (b) and (c) improve the performance of the strategy (a) is inconclusive, however, strategies (e) and (f) unequivocally improve the performance of the strategy (d). Furthermore, the tables all indicate that more stocks in the tracking portfolio lead to lower tracking error as intuition would suggest.

4.2. Cumulative returns

In this subsection, we compare the cumulative return curves of the market index, the baseline strategy of (a) (3LAE), and our proposed strategy of either (e) (CC-SAE) or (f) (MI-SAE). Furthermore, we include the cumulative return

Table 3. HSI index tracking RMSE (%) and TEV (%).

	(K_1)	RMSE	TEV	(K_2)	RMSE	TEV	(K_3)	RMSE	TEV
(a)		0.8495	0.8479		0.5324	0.5314		0.3410	0.3392
(b)		0.5341	0.5336		0.4558	0.4542		0.3650	0.3624
(c)		0.5854	0.5848		0.4558	0.4542		0.3308	0.3300
(d)		0.8495	0.8479		0.4595	0.4584		0.3316	0.3304
(e)		**0.5492**	**0.5492**	⋆	**0.3852**	**0.3848**	⋆	0.3068	0.3068
(f)	⋆	0.5596	0.5595		0.3856	0.3852		**0.3010**	**0.3008**

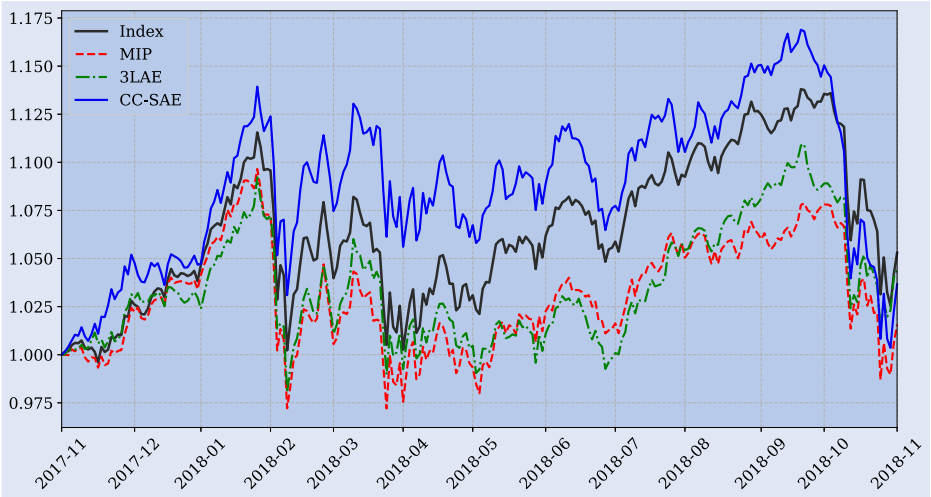

Figure 3. Cumulative return curves for S&P 500 at $K = 0.05^*|\mathcal{I}|$.

curves for the portfolio obtained by solving the mixed integer programming problem of equation (3). Specifically, while solving for a general mixed integer programming problem is intractable, state-of-the-art commercial mixed integer programming solver such as CPLEX can solve equation (3) exactly for problems of reasonable size. As such, CPLEX has been utilized in the past in a similar vein of work in Filippi *et al.* (2016), Gnagi and Strub (2019), Guastaroba and Speranza (2012), and the sizes of the evaluated data sets in this paper are within its tractable boundary. We used the latest version CPLEX 12.9 and all parameters were set to default values. We will use the label 'MIP' to represent the curve generated by CPLEX in the figures.

As in the previous subsection, we used the validation period to pick between (e) and (f), and therefore, the strategy marked by ⋆ in tables 1–3 represent our proposed strategy in this subsection for the respective data set and K value. The figures will illustrate that not only do our proposed strategies beat the baseline strategy in terms of the tracking error values as witnessed in the previous subsection, which will also be made apparent by the 'distance' of the curves in this subsection, but our strategies also present much more similar 'shape' of the return curve to the curve of the index than that of the baseline strategy.

Figures 3–5 show the curves for the S&P 500 data with $K = 0.05^*|\mathcal{I}|$, $0.1^*|\mathcal{I}|$, $0.2^*|\mathcal{I}|$, respectively. The figures clearly show that the return curve of our proposed strategy follows that of the index much more closely than that of the baseline strategy. A favorable aspect found in the figures is that while the return curve of the baseline strategy lies below that of the index, the curve of our proposed strategy lies on or above that of the index, for most part of the holding period. These figures, and also figures for other data sets to be shown shortly, indicate that more stocks in the portfolio lead to lower tracking error as also was shown in the tables of the previous subsection. Figures 3 and 4 show that our proposed strategy follows the index curve slightly better than the optimal tracking portfolio curve shown by MIP. In figure 5, the curves of our proposed strategy and of MIP performed almost identically with respect to the index curve.

Figures 6–8 show the cumulative return curves of the FTSE 100 data with $K = 0.1^*|\mathcal{I}|$, $0.2^*|\mathcal{I}|$, $0.3^*|\mathcal{I}|$, respectively. In these figures, the curves of our strategy follow those of the index progressively much closer as the value of K is increased. While similar trait can be found for the baseline strategy, our proposed strategy performs clearly superior. As before, the curves for our strategies not only are closer to those of the index, but they tend to lie above those of the baseline strategies. Figure 6 shows that the MIP curve tracks the index curve slightly better than our proposed curve, however, in figures 7 and 8, the situation seems to get reversed with our proposed curve tracking the index slightly closer.

Figures 9–11 show the cumulative return curves of the HSI data with $K = 0.1^*|\mathcal{I}|$, $0.2^*|\mathcal{I}|$, $0.3^*|\mathcal{I}|$, respectively. Figures for this data set also reveal that our proposed index tracking strategy follows the index much better than the baseline strategy. Specifically, not only are the curves for our strategy plotted much closer to those of the index, but they show much more similar shape to those of the index than those of the baseline strategy. Note in particular that, post-July 2018, the curve of the baseline strategy diverges away from that of the index while the curve of our proposed strategy remains close to that of index as in most part of the entire holding period. For the performance comparison between our proposed strategy and

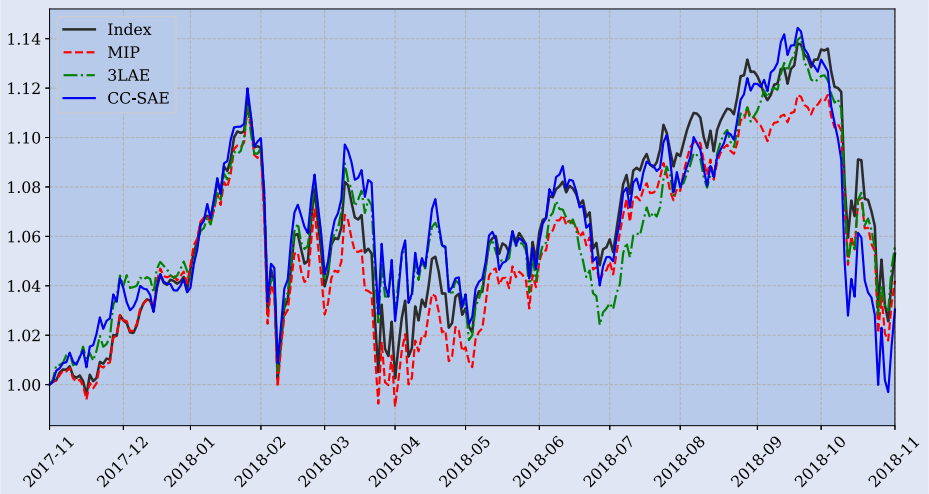

Figure 4. Cumulative return curves for S&P 500 at $K = 0.1*|\mathcal{I}|$.

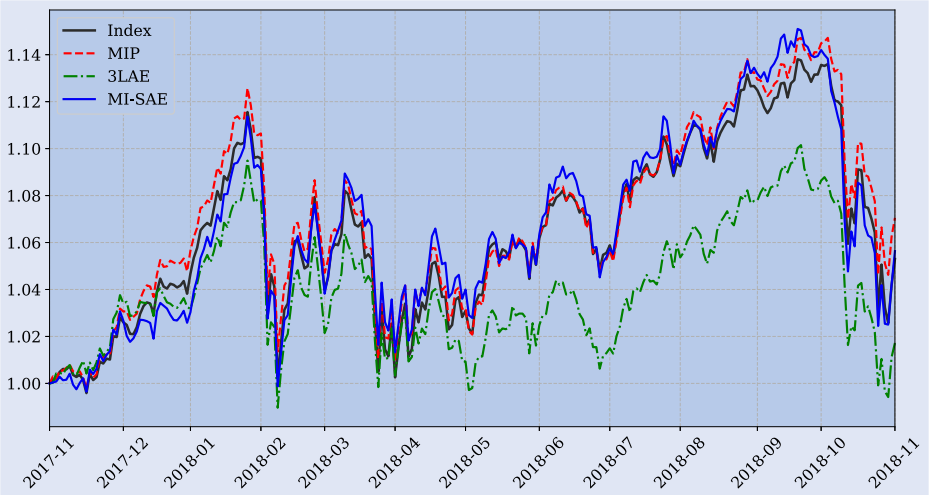

Figure 5. Cumulative return curves for S&P 500 at $K = 0.2*|\mathcal{I}|$.

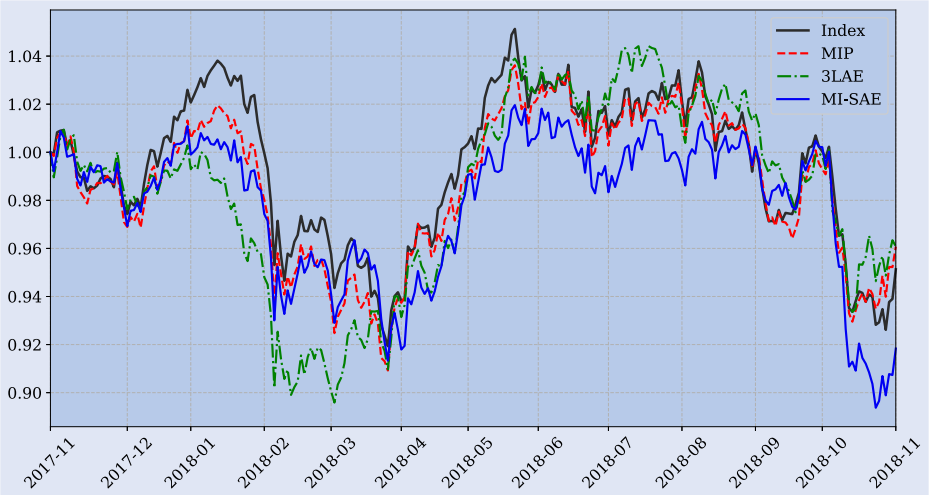

Figure 6. Cumulative return curves for FTSE 100 at $K = 0.1*|\mathcal{I}|$.

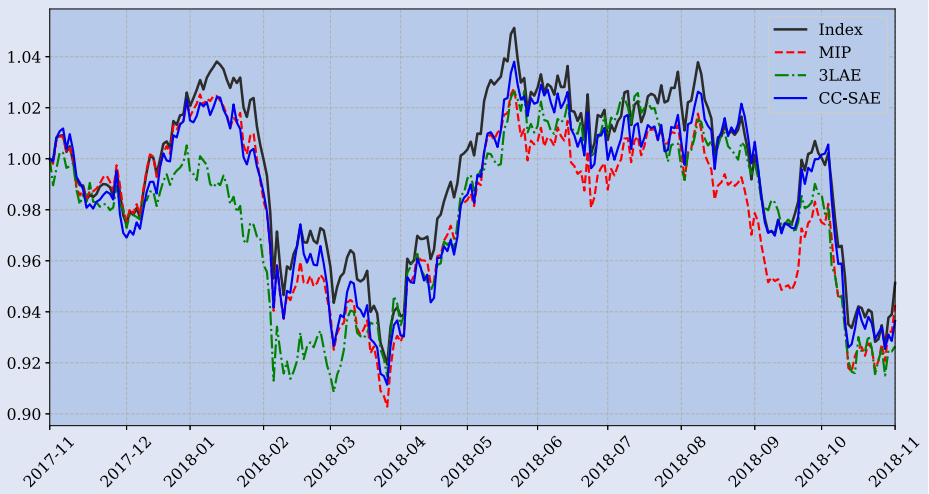

Figure 7. Cumulative return curves for FTSE 100 at $K = 0.2*|\mathcal{I}|$.

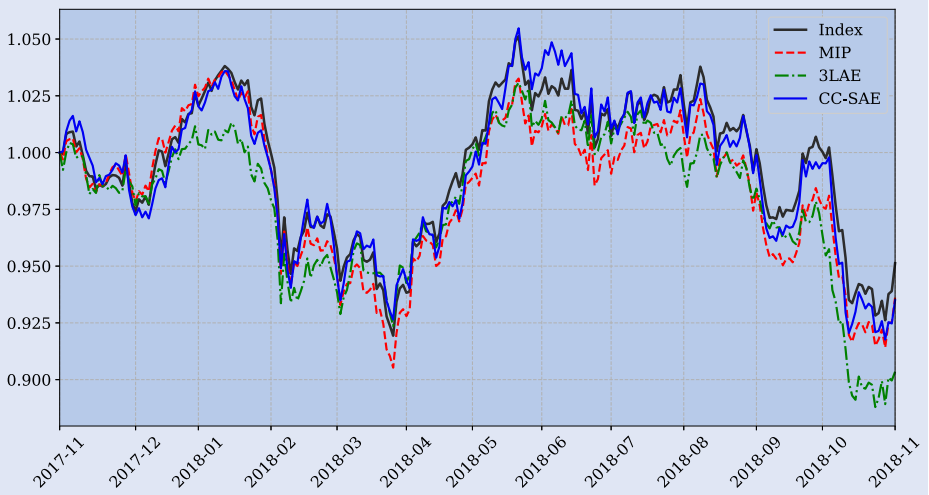

Figure 8. Cumulative return curves for FTSE 100 at $K = 0.3*|\mathcal{I}|$.

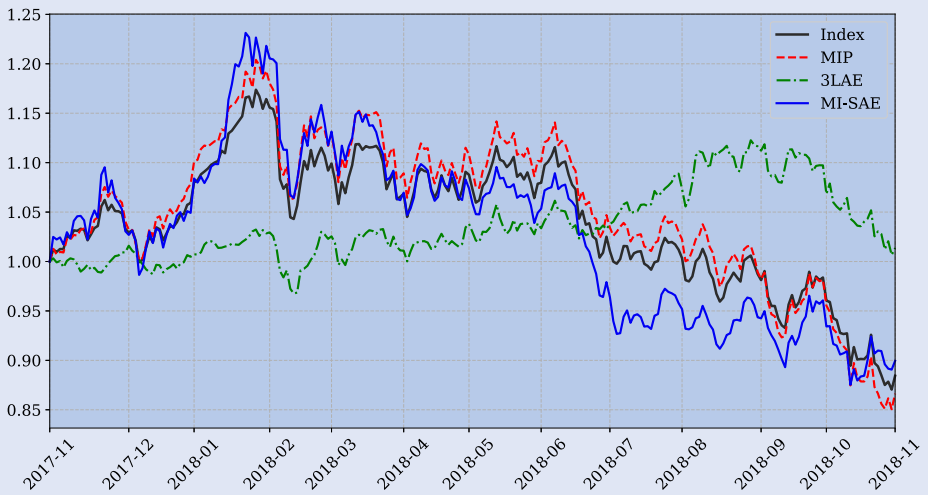

Figure 9. Cumulative return curves for HSI at $K = 0.1*|\mathcal{I}|$.

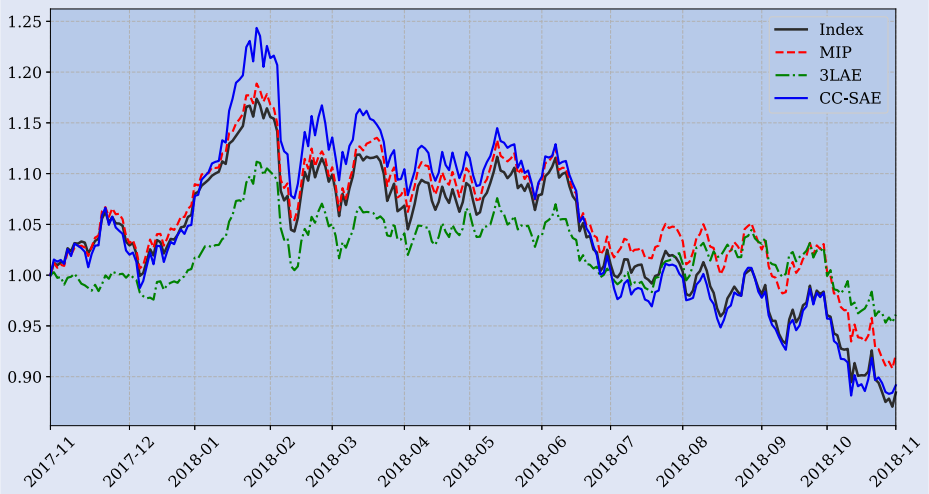

Figure 10. Cumulative return curves for HSI at $K = 0.2^*|\mathcal{I}|$.

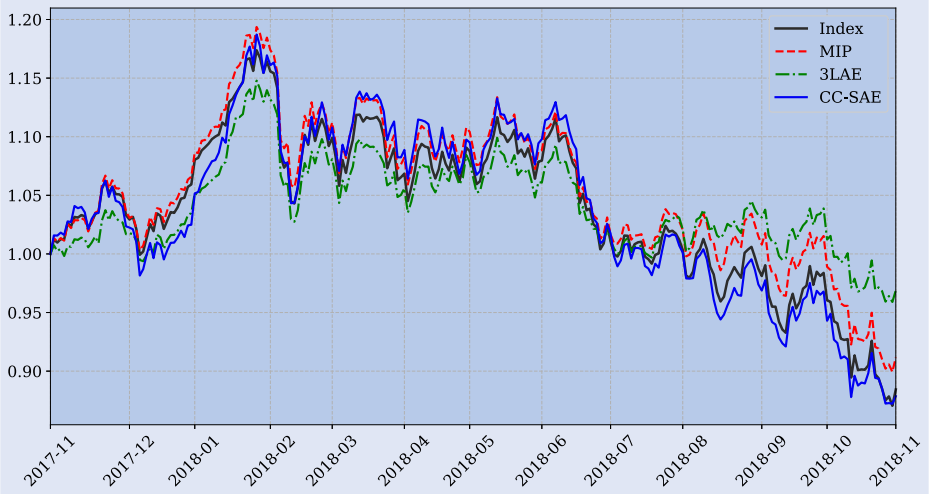

Figure 11. Cumulative return curves for HSI at $K = 0.3^*|\mathcal{I}|$.

the optimal MIP strategy, figures 9–11 show similar behavior as figures 6–8, respectively. Specifically, figure 9 shows outperformance by the MIP strategy whereas figures 10 and 11 show outperformance by our strategy.

5. Conclusion

In this paper, we have considered the index tracking problem which is of both theoretical and practical importance in the field of financial economics. By extending the latest developments in the field of artificial intelligence, and in particular, of deep learning, we have proposed a deep neural network that is designed specifically for the index tracking problem.

The main contributing factor of this deep neural network was the hierarchically learned deep latent representation of individual assets that was compared with its unlearned versions. Empirical results have indicated that rather than an ordinary deep autoencoder a stacked autoencoder is necessary to produce the deep latent representations of individual assets which can uncover the most relevant ones for the index tracking problem. Moreover, our empirical findings show that appropriately defined statistical measures between the return vector of an asset and that of the hierarchically learned latent representation serve to establish the asset selection criterion much better than the distance between the return vectors of an asset and its reconstructed version from the output of the deep autoencoder.

Our proposed index tracking strategies have been compared and contrasted with other deep learning-based strategies over the three major market indices of the S&P 500, FTSE 100 and HSI, and in the experiments conducted, our strategies outperformed all other strategies in every single market and for every cardinality constraint value tested. Furthermore, our strategy has shown cumulative return performance that is comparable if not slightly better than the in-sample optimal strategy provided by the solution to the mixed integer programming problem. In the future, we plan to revise our optimization objective function to address enhanced index tracking.

Acknowledgments

The authors would like to express sincere gratitude to the anonymous reviewers for the detailed and constructive comments that helped to improve the presentation of the paper.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2017R1D1A1B03032722).

References

Andriosopoulos, K. and Nomikos, N., Performance replication of the Spot Energy Index with optimal equity portfolio selection: Evidence from the UK, US and Brazilian markets. *Eur. J. Oper. Res.*, 2014, **234**(2), 571–582.

Bao, W., Yue, J. and Rao, Y., A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLoS ONE*, 2017, **12**(7), e0180944.

Beasley, J.E., Meade, N. and Chang, T.-J., An evolutionary heuristic for the index tracking problem. *Eur. J. Oper. Res.*, 2003, **148**(3), 621–643.

Bengio, Y., Courville, A. and Vincent, P., Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2013, **35**(8), 1798–1828.

Bengio, Y., Lamblin, P., Popovici, D. and Larochelle, H., Greedy layer-wise training of deep networks. In *Neural Information Processing Systems*, edited by B. Schölkopf, J.C. Platt, and T. Hoffman, pp. 153–160, 2006 (MIT Press: Cambridge, MA).

Blume, M.E. and Edelen, R.M., On replicating the S&P 500 index. Technical Report, The Wharton School, University of Pennsylvania, 2002.

Chang, T.-J., Meade, N., Beasley, J.E. and Sharaiha, Y.M., Heuristics for cardinality constrained portfolio optimisation. *Comput. Oper. Res.*, 2000, **27**, 1271–1302.

Chen, C. and Kwon, R.H., Robust portfolio selection for index tracking. *Comput. Oper. Res.*, 2012, **39**, 829–837.

Chiam, S.C, Tan, K.C. and Al Mamun, A., Dynamic index tracking via multi-objective evolutionary algorithm. *Appl. Soft Comput.*, 2013, **13**, 3392–3408.

Corielli, F. and Marcellino, M., Factor based index tracking. *J. Bank. Finance*, 2006, **30**(8), 2215–2233.

Dorocakova, M., Comparison of ETF's performance related to tracking error. *J. Int. Stud.*, 2017, **10**(4), 154–165.

Fama, E.F. and French, K.R., Multifactor explanations of asset pricing anomalies. *J. Finance*, 1996, **51**, 55–84.

Fama, E.F and French, K.R., The capital asset pricing model: Theory and evidence. *J. Econ. Perspect.*, 2004, **10**(3), 25–46.

Fernandez, A. and Gomez, S., Portfolio selection using neural networks. *Comput. Oper. Res.*, 2007, **34**, 1177–1191.

Filippi, C., Guastaroba, G. and Speranza, M.G., A heuristic framework for the bi-objective enhanced index tracking problem. *Omega*, 2016, **65**, 122–137.

Frino, A. and Gallagher, D.R., Tracking S&P 500 index funds. *J. Portfolio Manag.*, 2001, **28**(1), 44–55.

Gaivoronski, A.A. and Krylov, S., Optimal portfolio selection and dynamic benchmark tracking. *Eur. J. Oper. Res.*, 2005, **163**, 115–131.

Gnagi, M. and Strub, O., Tracking and outperforming large stock-market indices. *Omega*, 2019, forthcoming.

Guastaroba, G. and Speranza, M.G., Kernel search: An application to the index tracking problem. *Eur. J. Oper. Res.*, 2012, **217**(1), 54–68.

Heaton, J.B., Polson, N.G. and Witte, J.H., Deep learning in finance, 2016. arXiv: 1602.06561v2.

Jansen, R. and van Dijk, R., Optimal benchmark tracking with small portfolios. *J. Portfolio Manag.*, 2002, **28**(2), 33–39.

Kim, S., Volatility forecasting for low-volatility portfolio selection in the US and the Korean equity markets. *J. Exp. Theor. Artif. Intell.*, 2018, **30**(1), 71–88.

Kim, S., Enhancing the momentum strategy through deep regression. *Quant. Finance*, 2019, **19**(7), 1121–1133.

Kim, S. and Kim, S., Min k-cut for asset selection in risk-based portfolio strategies. In *Artificial Intelligence – Emerging Trends and Applications*, edited by M.A. Aceves-Fernandez, pp. 161–177, 2018 (IntechOpen).

Kingma, D.P. and Ba, J., Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015 (San Diego).

Kingma, D.P. and Welling, W., Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014 (Banff).

Lecun, Y., Bengio, Y. and Hinton, G., Deep learning. *Nature*, 2015, **521**, 436–444.

Malkiel, B.G., Returns from investing in equity mutual funds 1971 to 1991. *J. Finance*, 1995, **50**(2), 549–572.

Montfort, K.V., Visser, E. and van Draat, L.F., Index tracking by means of optimized sampling. *J. Portfolio Manag.*, 2008, **34**(2), 143–152.

Mutunge, P. and Haugland, D., Minimizing the tracking error of cardinality constrained portfolio. *Comput. Oper. Res.*, 2018, **90**, 33–42.

Oh, K.J., Kim, T.Y. and Min, S., Using genetic algorithm to support portfolio optimization for index fund management. *Expert. Syst. Appl.*, 2005, **28**(2), 371–379.

Ouyang, H., Zhang, X. and Yan, H., Index tracking based on deep neural network. *Cogn. Syst. Res.*, 2019, **57**, 107–114.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L. and Lerer, A., Automatic differentiation in PyTorch. In *Neural Information Processing Systems*, 2017 (Long Beach).

Roh, T.H., Forecasting the volatility of stock price index. *Expert. Syst. Appl.*, 2007, **33**, 916–922.

Roll, R., A mean/variance analysis of tracking error. *J. Portfolio Manag.*, 1992, **18**(4), 13–22.

Ross, S., The arbitrage theory of capital asset pricing. *J. Econ. Theory*, 1976, **13**, 341–360.

Rudolf, M., Wolter, H.-J. and Zimmermann, H., A linear model for tracking error minimization. *J. Bank. Finance*, 1999, **23**(1), 85–103.

Sharpe, W.F., Capital asset prices: A theory of market equilibrium under conditions of risk. *J. Finance*, 1964, **19**(3), 425–442.

Strub, O. and Baumann, P., Optimal construction and rebalancing of index-tracking portfolios. *Eur. J. Oper. Res.*, 2018, **264**, 370–387.