Quantitative Finance

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/rquf20

Deep learning for enhanced index tracking

Zhiwen Dai & Lingfei Li

To cite this article: Zhiwen Dai & Lingfei Li (2024) Deep learning for enhanced index tracking, Quantitative Finance, 24:5, 569-591, DOI: 10.1080/14697688.2024.2356239

To link to this article: https://doi.org/10.1080/14697688.2024.2356239

© 2024 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.

Published online: 06 Jun 2024.

Submit your article to this journal

Article views: 1651

View related articles

View Crossmark data

Full Terms & Conditions of access and use can be found at
https://www.tandfonline.com/action/journalInformation?journalCode=rquf20

Quantitative Finance, 2024
Vol. 24, No. 5, 569–591, https://doi.org/10.1080/14697688.2024.2356239

Routledge
Taylor & Francis Group
Check for updates

Deep learning for enhanced index tracking

ZHIWEN DAI and LINGFEI LI *

Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Hong Kong SAR, People's Republic of China

(*Received 14 July 2023; accepted 11 May 2024; published online 7 June 2024*)

We develop a novel deep learning method for the enhanced index tracking problem, which aims to outperform an index while effectively controlling the tracking error. We generate a dynamic trading policy from a neural network that accepts a set of features as inputs. We design four blocks in the neural network architecture to handle different types of features, including regimes of the index and stocks, their short-term characteristics, and the current allocation. Outputs from the blocks are integrated into the final output that changes the portfolio allocation. We test our model on several indexes in empirical studies based on real market data. Out-of-sample results reveal the importance of different features and demonstrate the ability of our method in obtaining excess returns while effectively controlling the tracking error, downside risk, and transaction costs.

Keywords: Deep learning; Index tracking; Enhanced index tracking; Portfolio selection; Regime switching

1. Introduction

Index-related funds have been developing rapidly over the past decades and enjoy great popularity among investors (Chabakauri and Rytchkov 2021, Coles *et al.* 2022). One type of index-related fund tries to minimize the error of tracking an index, hence known as index replication or index tracking (IT) funds. Another type of index-related fund considers enhanced index tracking (EIT), which attempts to obtain excess returns by benchmarking against an index while still controlling the deviation from it. EIT funds are appealing to investors as they combine the strengths of both passive and active management. On one hand, they provide investors some assurance as the trend of the fund does not deviate significantly from that of an index. On the other hand, they offer the potential to capture higher returns than the index. It is reported that EIT funds have grown more quickly than IT funds; see e.g. Filippi *et al.* (2016).

In this paper, we study the EIT problem for a stock index by trading some of its constituents. We also consider the IT problem as a special case of EIT by ignoring the objective of generating excess returns. In general, solving the EIT problem requires selecting a list of stocks for trading and determining their weights in the portfolio. Without considering dynamic trading, various papers formulate the problem as a mixed-integer program with cardinality and other types of constraints; see e.g. Coleman *et al.* (2006), Guastaroba and Speranza (2012), Filippi *et al.* (2016), Strub and Baumann (2018), Benidis *et al.* (2018), and Hong *et al.* (2022). This type of optimization problem is generally hard to solve, especially in high dimensions, but heuristic and approximate methods are available. Due to the inherent difficulty in tackling the two aspects of the problem simultaneously, many papers that study IT or EIT solve the problem in two steps: first find a list of stocks for trading by some method and then determine the weight allocation for these selected stocks.

Stock selection is usually done based on some criterion. One widely used criterion is market capitalization; see Meade and Salkin (1990), Larsen and Resnick (1998), Bamberg and Wagner (2000), and van Montfort *et al.* (2008). Larsen and Resnick (1998) point out that portfolios consisting of large-cap stocks have a smaller tracking error than those based on small-cap ones. Another popular criterion is the beta of a stock. Ling *et al.* (2014) and Huang *et al.* (2018) choose stocks based on the proximity of their beta to one. Other types of methods have also been developed and applied for stock selection. Dose and Cincotti (2005) cluster stocks based on a similarity measure and form the tracking portfolio based on the clusters. Corielli and Marcellino (2006) and Jiang and Perez (2021) apply principal component analysis (PCA) of the asset price or return to rank and select stocks. Bradrania *et al.* (2022) use a neural network to choose the best stock selection criterion among several proposed criteria. Ouyang *et al.* (2019), Kim and Kim (2020), and Zhang *et al.* (2020) propose models based on auto-encoders as a generalization of PCA for ranking and selection of stocks. The aforementioned

*Corresponding author. Email: lfli@se.cuhk.edu.hk

© 2024 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.
This is an Open Access article distributed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives License (http://creativecommons.org/licenses/by-nc-nd/4.0/), which permits non-commercial re-use, distribution, and reproduction in any medium, provided the original work is properly cited, and is not altered, transformed, or built upon in any way. The terms on which this article has been published allow the posting of the Accepted Manuscript in a repository by the author(s) or with their consent.

papers consider stock selection in the context of IT but some of them also apply the same method to EIT; see e.g. Dose and Cincotti (2005) and Huang *et al.* (2018). For EIT, Kim (2021) use the long short-term memory (LSTM) model and its variations to forecast stock returns to select stocks, while Yi *et al.* (2022) apply PCA and clustering.

In this paper, we select stocks for EIT by following the widely used criterion of market capitalization, which is natural for tracking capitalization-weighted indexes like S&P 500. Furthermore, many large-cap stocks in S&P 500 displayed impressive returns over a relatively long period in history, and thus they are suitable candidates for the goal of beating the index. Our research focuses on the second step, i.e. how to determine the weight allocation in the tracking portfolio once a list of stocks is selected.

In practice, the manager of a tracking portfolio needs to rebalance it over time and thus faces a dynamic control problem. Nevertheless, most papers in the literature on IT or EIT formulate the allocation problem as a single-period optimization model (see the papers mentioned above) and solve it using an optimization algorithm. The manager then constructs a portfolio using the optimal weight found and holds it until the next date when the optimization problem is solved again with new data to yield a new portfolio allocation. In this periodic re-optimization approach, typically the most recent one or two years of data is used to estimate the quantities required in the optimization model. For it to work well requires the market to resemble the recent past, which sometimes fails to hold. This is evident if we consider shifts in the market regime between bull and bear, which is well documented in the literature (Guidolin and Timmermann 2007, 2008, Tu 2010). When the market regime switches, a substantial change in the allocation is often required, which may not be produced by the re-optimization approach because the policy it generates is trained based on data in a different regime. This illustrates the drawback of straightforward re-optimization, which can lead to unstable and inferior performance over time.

We propose a novel deep learning method to construct and dynamically rebalance the tracking portfolio for EIT. We are motivated by the remarkable success of deep learning in solving dynamic control problems in various financial applications; see e.g. Buehler *et al.* (2019), Carbonneau and Godin (2021), Nian *et al.* (2021), Dai *et al.* (2022), and Chen and Li (2023) for hedging of options, Guéant and Manziuk (2019) and Lai *et al.* (2023) for market making, and Li and Forsyth (2019) and Ni *et al.* (2022) for some investment problems. The deep learning approach has two appealing features that makes it suitable for solving control problems. First, unlike the model-based stochastic control approach, it does not require strong model assumptions and can be essentially model free. Second, neural networks are capable of approximating highly nonlinear functions. In financial applications, the optimal control often shows nonlinear dependence on inputs.

In our framework, we consider transaction costs and three types of constraints: no short selling of stocks, no leverage, and limiting conditional value at risk (CVaR) of the portfolio return. Few papers in the existing literature consider controlling transaction costs for IT or EIT, but it is an important issue in portfolio management. CVaR is a popular measure of tail risk in finance and is considered in Wang *et al.* (2012), Huang *et al.* (2018), and Guastaroba *et al.* (2020) as a constraint for IT or EIT. These papers show that by constraining CVaR the downside risk of the tracking portfolio can be effectively controlled. To overcome the drawback of re-optimization, we introduce features that characterize the states of the market represented by the index and stocks in the tracking portfolio. In addition, we consider short-term characteristics of the stocks and the current allocation as features. An important research question is how to design a neural network using these features as inputs to generate the trading policy. Although one can apply standard neural network architectures, they can be problematic as we explain below.

When designing a neural network for our problem, there are three major concerns. First, the real market data we can use for training the neural network is quite limited compared with other fields such as computer vision and natural language processing. Thus, a very sophisticated design can easily lead to overfitting. Second, the neural network should be scalable for training. The dimension of the input feature vector is $\mathcal{O}(n)$, where n is the number of stocks. If we simply stack all the features into a vector as inputs to the standard feedforward neural network (FNN), it would result in a large network when n is large, thus difficult to train. Third, the neural network should have some interpretability for financial investment.

To address these concerns, we use two ideas to design the architecture, which leads to a novel neural network model. (1) We build four blocks to process information from various types of features, and we call them *main*, *score*, *gate*, and *memory*. Each block has its own unique role and functions in the given order. In the first step, the main block determines a proper weight based on the current market regime to combine the long-term average weight vectors in bull and bear markets. Then, the score block considers the influence of short-term behavior on allocation. It first assigns a score to every stock in the portfolio based on short-term characteristics of them and the index, and then converts the score vector into a weight vector, which is further combined with the weight vector from the main block. The output weight vector from the score block reflects both long-term and short-term information. Subsequently, the gate block makes adjustments to the output weight vector from the score block based on the regime of each stock. Finally, the memory block moderates the proposed weight change given by the gate block to control transaction costs and yields the final weight vector as the new allocation among stocks. This block-based design not only reduces the network size, but also improves interpretability. Furthermore, this architecture is flexible to accommodate new types of features by adding new blocks for them. (2) We use parameter sharing, where the score and gate blocks are shared among all assets in the portfolio. This reduces the network size and makes our model scalable. It remains computationally efficient to train our model even when the number of assets is large. Our design also relieves the data insufficiency problem as data of different stocks can be used to train the same block.

We conduct an out-of-sample empirical study using S&P 500 as the benchmark index. Our framework can easily generate different trading policies by disabling some blocks and fixing the values of some parameters in the neural network.

Thus, we can understand the importance of different features by comparing the performances of various policies. Our study reveals that the importance of a feature varies with the problem. Short-term characteristics are crucial for good performance in IT, whereas regimes of the index and stocks are much more important in EIT. In particular, considering these regime features allows our trading policy to generate the well-known phenomenon of flight to safety as the market switches from bull to bear. It also enables the policy to better ride the bull wave for faster growth as the market enters bull from bear. Overall, our results demonstrate that the proposed deep learning model can obtain impressive excess returns while effectively controlling the tracking error, downside risk, and transaction costs. It can also outperform re-optimization by a large margin for EIT.

Our paper is closely related to Ni *et al.* (2022), who solve the problem of outperforming a stochastic benchmark using deep learning. Our work differs from theirs in a number of ways. First, there are differences in the problem formulation. They consider the objective of outperforming without tracking. Furthermore, they study the problem in a finite time horizon and care about the terminal performance, which is natural in their problem. In comparison, the EIT problem is long term in nature as there is typically no specified end date for an EIT fund. Consequently, the fund manager would care about the long-term average performance as opposed to the terminal one. Second, the neural network that generates the trading policy is different in inputs and architecture. They use an FNN with time to maturity, portfolio wealth, and benchmark wealth as inputs, but do not consider market and stock features. We consider these features and develop a novel architecture to overcome the potential problems of using FNNs with high-dimensional feature vectors.

The number of studies that use deep learning to generate the portfolio allocation for IT or EIT is still quite limited. Kwak *et al.* (2021) employ a deep neural network with fixed noise to obtain the portfolio weight for IT, while Yi *et al.* (2022) use an LSTM model to output the portfolio weight for EIT. Compared with our work, there are substantial differences in the neural network design. Additionally, these works do not address the issues considered by us, such as how to control transaction costs and deal with the CVaR constraint, the importance of the index and stock regimes, and interpretability of the deep learning model.

We also notice that several papers approach IT or the problem of outperforming a stochastic benchmark using techniques from continuous-time stochastic control theory in a model-based way. Yao *et al.* (2006) assume the asset prices follow a multidimensional geometric Brownian motion (GBM) and formulated IT as a stochastic linear quadratic control problem, which was solved by semi-definite programming. Al-Aradi and Jaimungal (2018, 2021) solve the control problem of outperforming a stochastic benchmark under some models in closed form using dynamic programming and convex analysis, respectively. These works do not consider no-shorting, no-leverage, and CVaR constraints. Compared to the model-based stochastic control approach, deep learning is data driven and can be more flexible in dealing with constraints. Recently, Peng *et al.* (2023) propose an interesting reinforcement learning approach, which is also model free. However, they only consider IT.

The rest of this article is organized as follows. We first present the formulation of EIT as a stochastic control problem in section 2. We then discuss the features and architecture of our neural network in section 3. We show the empirical results in section 4 for four indexes and conclude in section 5. The appendix contains supplementary information on the filtering algorithm. Throughout the paper, we denote vectors and matrices in bold to differentiate them from scalars.

2. Problem formulation

We consider the EIT problem for a given index and formulate it as a stochastic control problem. We assume that a number of stocks have been chosen and focus on how to construct and update the portfolio consisting of these stocks and cash.

2.1. *Portfolio return and weight dynamics*

Suppose that we have chosen n stocks. The proportion invested in stock i at time t is denoted by $w_{i,t}$ for $i = 1, \dots, n$, and it is also called the weight of stock i at t. We also introduce $w_{0,t}$ as the weight of cash at t. These weights always satisfy

$$\sum_{i=0}^{n} w_{i,t} = 1, \quad \forall\, t \geq 0.$$

We define the weight vector of all the assets $\boldsymbol{w}_t := (w_{0,t}, \dots, w_{n,t}) \in \mathbb{R}^{n+1}$ and the weight vector of the stocks $\boldsymbol{w}_{S,t} := (w_{1,t}, \dots, w_{n,t}) \in \mathbb{R}^n$.

We assume that stock trading incurs a transaction cost that is proportional to the traded value with percentage ρ. There is no cost of moving money into or out of the cash account.

At time t, we rebalance the portfolio by changing the weights of the assets. For asset i, its weight after rebalancing is denoted by $\tilde{w}_{i,t}$, which is calculated relative to $v_{P,t}$, the portfolio value before rebalancing. We consider the corresponding vectors $\tilde{\boldsymbol{w}}_t := (\tilde{w}_{0,t}, \dots, \tilde{w}_{n,t})$ and $\tilde{\boldsymbol{w}}_{S,t} := (\tilde{w}_{1,t}, \dots, \tilde{w}_{n,t})$. After rebalancing, we have

$$v_{P,t} - \rho \sum_{i=1}^{n} |\tilde{w}_{i,t} v_{P,t} - w_{i,t} v_{P,t}| = v_{P,t} \sum_{i=0}^{n} \tilde{w}_{i,t}. \tag{1}$$

Eliminating $v_{P,t}$, we obtain

$$1 - \rho \sum_{i=1}^{n} |\tilde{w}_{i,t} - w_{i,t}| = \sum_{i=0}^{n} \tilde{w}_{i,t},$$

which implies $\sum_{i=0}^{n} \tilde{w}_{i,t} \leq 1$ as a result of transaction cost.

The after-rebalancing weights of stocks $\tilde{w}_{i,t}, i = 1, \dots, n$ are control variables in the EIT problem, and the weight of the cash account is simply determined by

$$\tilde{w}_{0,t} = 1 - \sum_{i=1}^{n} \tilde{w}_{i,t} - \rho \sum_{i=1}^{n} |\tilde{w}_{i,t} - w_{i,t}|. \tag{2}$$

We do not allow short selling of stocks in this paper, which is often a constraint in practice. We also do not consider taking leverage to finance stock investment, i.e. the total weight of stocks cannot exceed one. Thus, the space of admissible controls is given by

$$\left\{\tilde{w}_{S,t} \in \mathbb{R}^n : 1 - \sum_{i=1}^{n} \tilde{w}_{i,t} \geq 0,\ \tilde{w}_{i,t} \geq 0,\ i = 1, \ldots, n\right\}. \quad (3)$$

Using (3) and the estimate

$$\rho \sum_{i=1}^{n} |\tilde{w}_{i,t} - w_{i,t}| \leq \rho \sum_{i=1}^{n} (|\tilde{w}_{i,t}| + |w_{i,t}|) \leq 2\rho,$$

it follows from (2) that

$$\tilde{w}_{0,t} \geq -2\rho.$$

Thus, in the worst case the weight of cash can become negative, which means we need to pay for a dollar amount of $|\tilde{w}_{0,t}|v_{P,t}$. As ρ is very small, this amount is insignificant relative to the portfolio value and we allow injecting cash or borrowing to cover it. Since the probability of the worst-case happening and the amount that needs to be covered are both very small, it does not present a serious issue in practice.

For asset i, let $r_{i,t+1}$ be its return from t to $t+1$. Consider the stock return vector $\boldsymbol{r}_{S,t+1} := (r_{1,t+1}, \ldots, r_{n,t+1})$. At time $t+1$, the investment portfolio value is given by

$$v_{P,t+1} = \sum_{i=0}^{n} \tilde{w}_{i,t} v_{P,t}(1 + r_{i,t+1}),$$

The portfolio return from t to $t+1$, denoted by $r_{P,t+1}$, satisfies

$$1 + r_{P,t+1} = \frac{v_{P,t+1}}{v_{P,t}} = \sum_{i=0}^{n} \tilde{w}_{i,t}(1 + r_{i,t+1}).$$

Thus, from (2) we obtain

$$\begin{aligned} r_{P,t+1} &= \sum_{i=0}^{n} \tilde{w}_{i,t}(1 + r_{i,t+1}) - 1 \\ &= \sum_{i=0}^{n} \tilde{w}_{i,t} r_{i,t+1} - \rho \sum_{i=1}^{n} |\tilde{w}_{i,t} - w_{i,t}|, \end{aligned} \quad (4)$$

After rebalancing at time t, the dollar amount invested in asset i is given by $v_{P,t}\tilde{w}_{i,t}(1 + r_{i,t+1})$. Hence, at time $t+1$ before next rebalancing, the weight of asset i becomes

$$w_{i,t+1} = \frac{\tilde{w}_{i,t}(1 + r_{i,t+1})}{\sum_{i=0}^{n} \tilde{w}_{i,t}(1 + r_{i,t+1})}. \quad (5)$$

Remark 2.1 Proportional transaction cost is commonly considered in the literature on portfolio selection; see e.g. Balduzzi and Lynch (1999), Dai *et al.* (2010a), and Soner and Touzi (2013). Our method can also deal with other forms of transaction cost by modifying the cost term $\rho \sum_{i=1}^{n} |\tilde{w}_{i,t} v_{P,t} - w_{i,t} v_{P,t}|$ in (1).

2.2. *Objectives*

Let $r_{I,t}$ be the return of the index from time $t-1$ to t. We assume that $\{r_{P,t}, t = 1, 2, \ldots\}$ and $\{r_{I,t}, t = 1, 2, \ldots\}$ are both weakly stationary, i.e. their mean, variance and autocovariance functions are invariant with respect to time shifts (Tsay 2010). Although price data are typically nonstationary, the return data often pass the commonly used stationarity tests. For this reason, it is common to assume that an asset return is at least weakly stationary in the finance literature.†

We are concerned about two objectives: one is the expected tracking error and the other is the expected excess return of the portfolio over the index. For the first objective, we consider the root mean squared tracking error per period as in Huang *et al.* (2018), defined by

$$L_{TE} := \sqrt{\mathbb{E}[(r_{P,t} - r_{I,t})^2]}.$$

For the second objective, we consider the expected excess return per period defined by

$$L_{\mathrm{ER}} := \mathbb{E}[r_{P,t} - r_{I,t}].$$

Both measures are the same in different periods due to the weak stationarity.

Following Beasley *et al.* (2003), Dose and Cincotti (2005), and Huang *et al.* (2018), we combine these two measures by introducing a weight for the second one and obtain the objective of EIT as

$$\begin{aligned} L_{\mathrm{EIT}} &:= L_{TE} - \lambda L_{\mathrm{ER}} \\ &= \left(\mathbb{E}[(r_{P,t} - r_{I,t})^2]\right)^{1/2} - \lambda \mathbb{E}[r_{P,t} - r_{I,t}], \end{aligned} \quad (6)$$

where $\lambda \geq 0$. When $\lambda = 0$, the problem becomes tracking the index only without considering obtaining excess returns. Alternative formulations of EIT also exist. Wu *et al.* (2007) and Filippi *et al.* (2016) formulate EIT as a bi-objective optimization problem while Canakgoz and Beasley (2009) treat one objective as a constraint. We use (6) as it is more tractable for deep learning.

Controlling CVaR is important for portfolio management. The value at risk (VaR) of a random return r at confidence level $1 - \alpha$ is defined as the negative α-quantile of the return distribution (McNeil *et al.* 2015), i.e.

$$\mathrm{VaR}_\alpha(r) := -\inf\{u \in \mathbb{R} : \Phi(u) \geq \alpha\},$$

where $\Phi(\cdot)$ is the cumulative distribution function of r. CVaR is defined as the expected loss conditioned on the loss exceeding VaR_α, i.e.

$$\mathrm{CVaR}_\alpha(r) := \mathbb{E}[-u \mid -u \geq \mathrm{VaR}_\alpha(r)].$$

As in Huang *et al.* (2018), we incorporate the constraint $\mathrm{CVaR}_\alpha(r_{P,t}) \leq c$ for some prespecified c in the EIT problem. To estimate CVaR of the tracking portfolio return from its time series, we assume it is strictly stationary (see Cai and Wang 2008), i.e. the joint distribution of returns is invariant with respect to time shifts.

† See remark 4.1 where we perform statistical tests about stationarity on the portfolio return process generated by the learned policy.

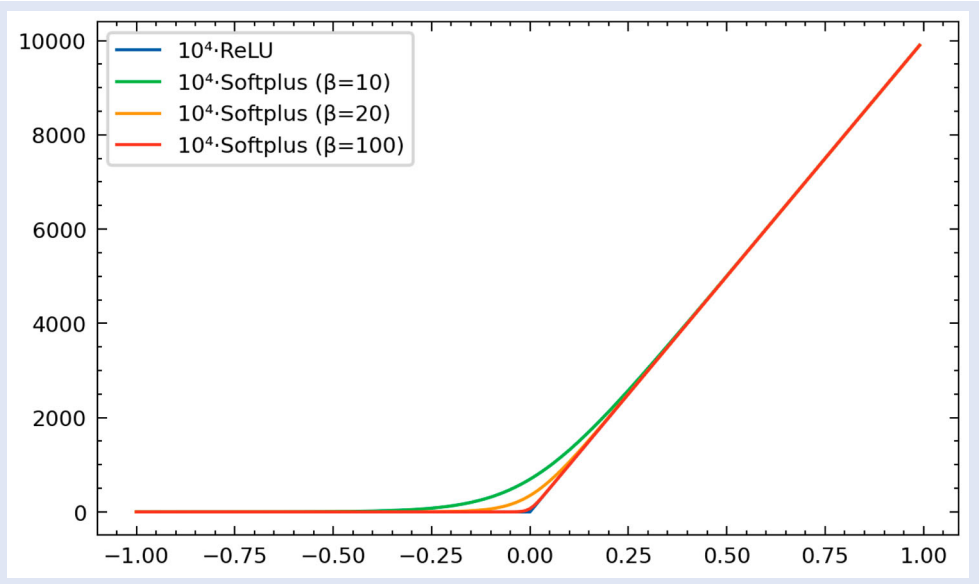


Figure 1. ReLU and Softplus functions.

In general, directly dealing with a constrained optimization problem using deep learning can be difficult. To make the problem more tractable for deep learning, we reformulate the CVaR constraint as a penalty term in the objective function. A natural penalty term is given by

$$\gamma(\mathrm{CVaR}_\alpha(r_{P,t}) - c)_+,$$

where $\gamma > 0$ is the penalty coefficient and $x_+ = \max\{x, 0\}$ is known as the Rectified Linear Unit (ReLU) function. The ReLU function can potentially cause problems for training as the penalty term is zero when the CVaR is below c. This would result in zero gradient in this region, while the gradient can suddenly become large when the CVaR gets greater than c. To address this issue, we replace ReLU with the Softplus function defined as

$$g_\beta(x) := \frac{\log(1 + e^{\beta x})}{\beta},$$

where β is the sharpness parameter. This function has a positive gradient on $\mathbb{R}$ and is close to ReLU. An illustration is given in figure 1, which shows the larger β is, the closer Softplus is to ReLU. Note that the Softplus function is always above ReLU, especially around $x = 0$, thus using Softplus approximation leads to a stricter constraint. Now, the penalty term is given by

$$P_{\mathrm{CVaR}} := \gamma g_\beta(\mathrm{CVaR}_\alpha(r_{P,t}) - c).$$

For the EIT problem with the CVaR constraint, the objective becomes

$$\begin{aligned} L_{\mathrm{EIT-CVaR}} &= L_{\mathrm{TE}} - \lambda L_{\mathrm{ER}} + P_{\mathrm{CVaR}} \\ &= \sqrt{\mathbb{E}[(r_{P,t} - r_{I,t})^2]} - \lambda \mathbb{E}[r_{P,t} - r_{I,t}] \\ &\quad + \gamma g_\beta(\mathrm{CVaR}_\alpha(r_{P,t}) - c). \end{aligned}$$

We summarize the key ingredients of our problem below.

- *State* $\boldsymbol{X}_t$: For $t = 0$, $\boldsymbol{X}_t = \boldsymbol{w}_0$, which is a fixed vector, e.g. one in the cash account and zero weight in the stocks. For $t \geq 1$, $\boldsymbol{X}_t = (r_{P,t}, \boldsymbol{w}_t)$.
- *Control* $\tilde{\boldsymbol{w}}_{S,t}$: Suppose that we rebalance every T_{rb} trading days. Thus, rebalancing only happens at those t that are multiples of T_{rb}. At a reblancing time t, the control $\tilde{\boldsymbol{w}}_{S,t}$ depends on a feature vector $\boldsymbol{F}_t$. Furthermore, it should satisfy the constraints given in (3). Note that the weight of cash $\tilde{w}_{0,t}$ is not a control variable as it is determined by (2).
- *Noise* $\boldsymbol{r}_{t+1}$: The noise term $\boldsymbol{r}_{t+1} = (\boldsymbol{r}_{S,t+1}, r_{I,t+1})$, which is the returns of stocks and the index from time t to $t + 1$ for $t \geq 0$. We assume the return of the cash account is a constant over time and hence not a noise.
- *State transition*: We can write it in the form $\boldsymbol{X}_{t+1} = f(\boldsymbol{X}_t, \tilde{\boldsymbol{w}}_{S,t}, \boldsymbol{r}_{t+1})$. Specifically, we have two transition equations given by (4) and (5).
- *Objective function* L: We minimize L, which is given by L_{IT} for IT, L_{EIT} for EIT, and $L_{\mathrm{EIT-CVaR}}$ for EIT with the CVaR constraint. In our implementation, the objective is estimated using data, which will be discussed in section 4.2.

Remark 2.2 It is important to observe from (4) that $r_{P,t}$ for any t is the portfolio's return after transaction costs. Thus, the objective function L is aware of the negative impact of transaction costs on the returns. It follows that by minimizing L, we can control the transaction costs.

3. Deep learning

We solve the stochastic control problem by deep learning. We design a neural network that takes a feature vector $\boldsymbol{F}_t$ as input to output the control $\tilde{\boldsymbol{w}}_{S,t} \in \mathbb{R}^n$ at time t, and use the

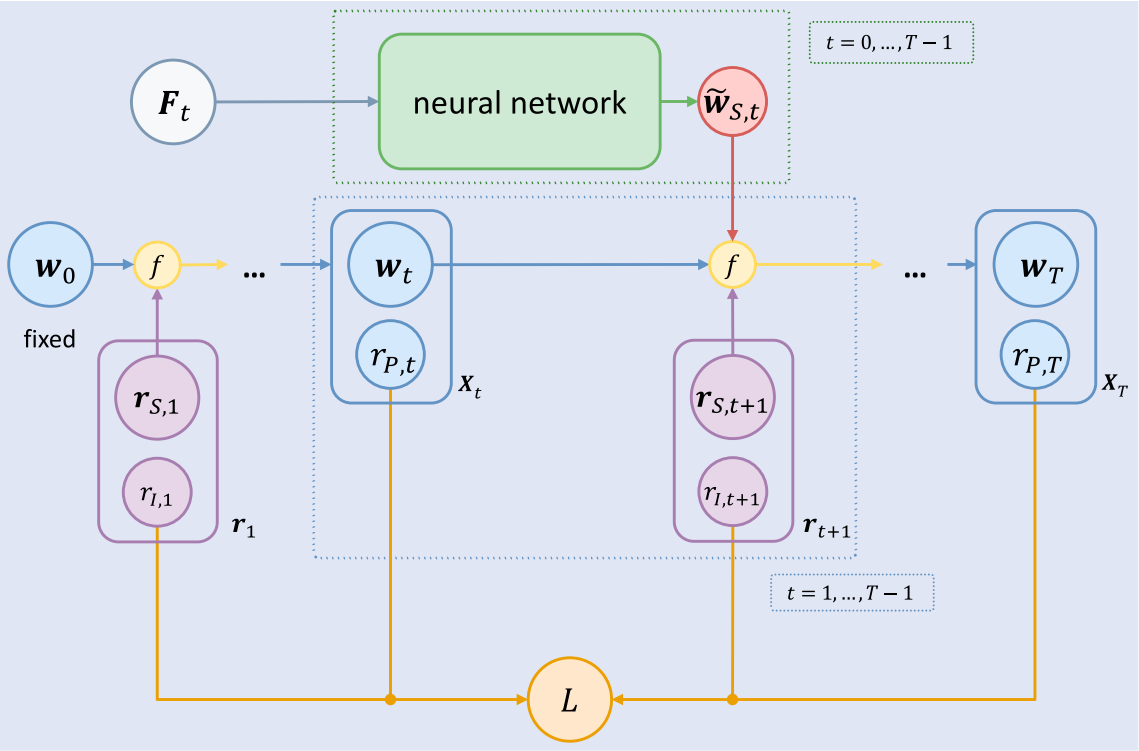


Figure 2. Forward propagation.

same neural network for different times. Figure 2 illustrates the workflow for forward propagation. We first discuss features that are input to the neural network and then present our design of its architecture.

3.1. Features

The regime of an index and of a stock can change over time, which has an obvious impact on investment. However, it is not directly observable but can be inferred from the return data. To capture the regime-switching behavior, for daily returns of the index and every stock, we build a separate hidden Markov model (HMM) with Gaussian noise as follows:

$$r_t = \mu_{q_t} + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma_{q_t}^2),$$

where the regime $q_t \in \mathcal{Q} = \{1, 2\}$ and its dynamics is described by a time-homogeneous Markov chain. We only consider two states for the regime, which is commonly used in modeling the switching behavior of stocks between bull and bear conditions (Dai *et al.* 2010b, 2016). Specifically, we identify the bull condition as state 1. For simplicity, we assume the noise is normally distributed to facilitate estimation and filtering, but more realistic heavy-tailed distributions can be used. As the regime q_t is unobservable, the HMM model is estimated using an expectation-maximization (EM) algorithm. After obtaining the parameters, we use the forward algorithm to filter the regime probability. We provide details for filtering in appendix 1.

We consider four types of features as components of $\boldsymbol{F}_t$ and discuss them below.

(1) *Smoothed version of filtered index regime probability* $\bar{p}_{I,t} \in [0, 1]$. This feature represents the belief of the index in regime 1. Let $p_{I,t} := \mathbb{P}(q_{I,t} = 1 \,|\, \boldsymbol{r}_{I,1:t})$. As there are only two regimes, it suffices to consider the filtered probability of one regime. Considering that the filtered probability values can be quite noisy, we use the smoothed version $\bar{p}_{I,t}$, which is obtained by averaging the most recent k filtered probabilities.

(2) *Smoothed version of filtered stock regime probabilities* $\bar{\boldsymbol{p}}_{S,t} \in [0, 1]^n$. We consider the filtered probability of regime 1 for every stock. For $i = 1, \ldots, n$, the ith entry of $\boldsymbol{p}_{S,t}$ is given by $p_{i,t} := \mathbb{P}(q_{i,t} = 1 \,|\, \boldsymbol{r}_{i,1:t})$. The smoothed probability vector $\bar{\boldsymbol{p}}_{S,t}$ is obtained in the same way as for the index regime probability.

(3) *Short-term features* $\boldsymbol{F}_{st,t} \in \mathbb{R}^{3n+2}$. For stock i, we consider its estimated mean $\hat{\mu}_i^{(k)}$, volatility $\hat{\sigma}_i^{(k)}$, and beta $\hat{\beta}_i^{(k)}$ in the most recent k days. For the index, we consider its estimated mean $\hat{\mu}_I^{(k)}$ and volatility $\hat{\sigma}_I^{(k)}$ in the most recent k days. These values are stacked as a vector $\boldsymbol{F}_{st,t} \in \mathbb{R}^{3n+2}$. The mean and volatility provide a basic description of the recent behavior of the stocks and index, which may continue in the near future. The beta of a stock shows its relationship with the index, which is potentially important for tracking the index.

(4) *Current stock weights* $\boldsymbol{w}_{S,t} \in \mathbb{R}^n$. The weights of the stocks before rebalancing influence the transaction cost at t as shown in (2). Thus, $\boldsymbol{w}_{S,t}$ is potentially useful in providing information to control transaction costs.

3.2. Neural network architecture

We propose a block-based design, where there are four blocks, each playing a unique role to incorporate information from one type of feature. Details of the blocks are given below.

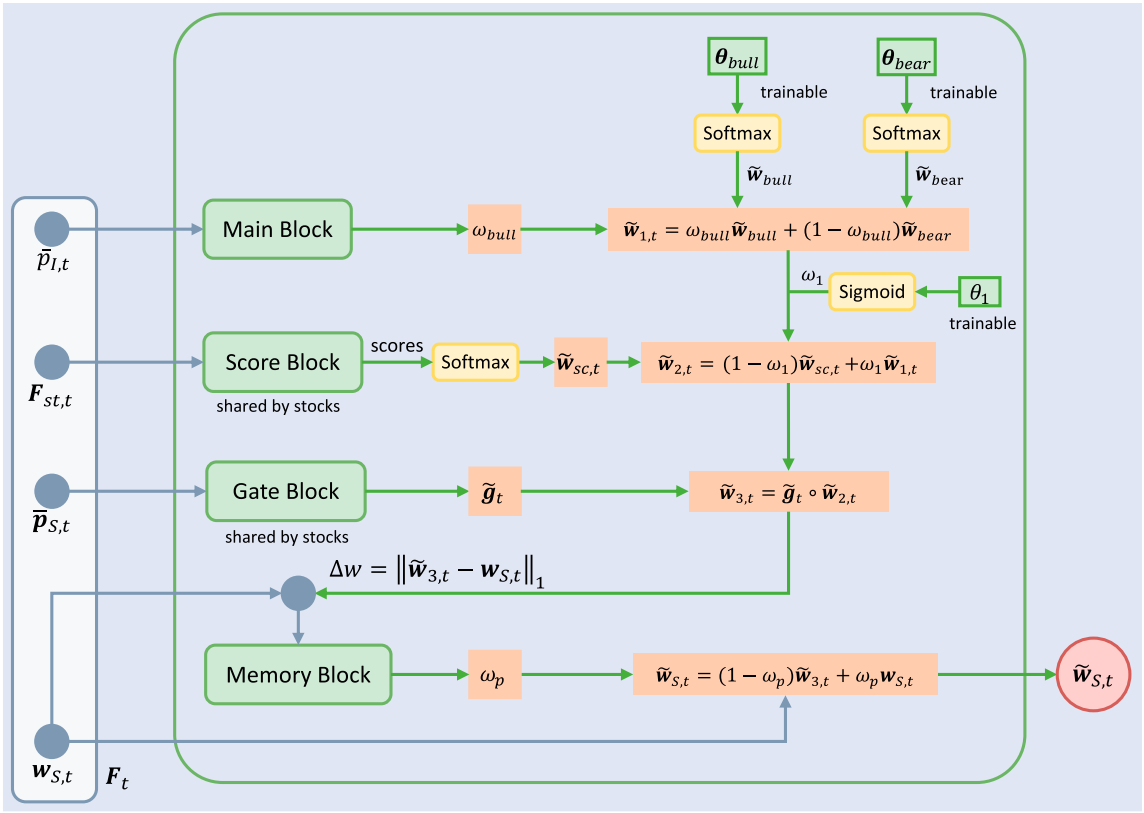


Figure 3. Architecture of the neural network. For the output of each softmax block, we drop the first entry.

(1) *Main block*: The purpose of the main block is to learn the stock weights based on long-term performance and the current condition of the market represented by the index. Intuitively, the allocations to stocks would vary in bull and bear markets. We introduce two weight vectors $\tilde{\boldsymbol{w}}_{bull} \in \mathbb{R}^n$ and $\tilde{\boldsymbol{w}}_{bear} \in \mathbb{R}^n$ to learn, which represent the weights of the stocks in the bull and bear markets, respectively. These vectors reflect the average long-term holdings in these stocks in different market conditions. The main block is designed as an FNN that accepts the smoothed index regime probability $\bar{p}_{I,t}$ as input, which indicates how likely the market is in the bull state. The FNN generates a scalar $\omega_{bull} \in (0,1)$ using the Sigmoid function to activate the output layer, which serves as a weight to combine the stock weights in bull and bear markets. That is, we calculate the final output of the main block as

$$\tilde{\boldsymbol{w}}_{1,t} = \omega_{bull}\tilde{\boldsymbol{w}}_{bull} + (1-\omega_{bull})\tilde{\boldsymbol{w}}_{bear}. \tag{7}$$

Among the two stock weight vectors, we define the one that has a larger weight in (7) when $\bar{p}_{I,t} \approx 1$ as $\tilde{\boldsymbol{w}}_{bull}$. The two vectors $\tilde{\boldsymbol{w}}_{bull}$ and $\tilde{\boldsymbol{w}}_{bear}$ should satisfy the constraints given in (3). To meet this requirement, we introduce two unconstrained parameters $\boldsymbol{\theta}_{bull} \in \mathbb{R}^{n+1}$ and $\boldsymbol{\theta}_{bear} \in \mathbb{R}^{n+1}$ to train, and set

$$\tilde{\boldsymbol{w}}_{bull} = \mathrm{Softmax}(\boldsymbol{\theta}_{bull})_{2:n+1},$$
$$\tilde{\boldsymbol{w}}_{bear} = \mathrm{Softmax}(\boldsymbol{\theta}_{bear})_{2:n+1},$$

where the subscript $2{:}n+1$ means that we retrieve the last n entries of the $(n+1)$-dimensional vector obtained from the Softmax function. This construction automatically guarantees the satisfaction of the constraints.

(2) *Score block*: This block is designed to score every stock and the cash account based on short-term features to determine their weights. We use an FNN for the block, which is shared among all the stocks. This design has the advantage that the network size can remain unchanged when n increases. Moreover, the data of different stocks are used to train the same structure, which relieves the problem of limited training data.

The input of the score block is a 5D vector containing $\hat{\mu}_i^{(k)}$, $\hat{\sigma}_i^{(k)}$, and $\hat{\beta}_i^{(k)}$ for the asset, and $\hat{\mu}_I^{(k)}$ and $\hat{\sigma}_I^{(k)}$ for the index. We treat the cash account as a special stock with the short-term mean, volatility, and beta all set as zero. The output of the block is a real-valued score for the asset under consideration. We stack these scores as a vector $\boldsymbol{sc}_t \in \mathbb{R}^{n+1}$ (starting from the score of cash and ending with the score of stock n), and pass it to the Softmax function, which outputs a vector in $\mathbb{R}^{n+1}$ that sums to one and all its elements are between 0 and 1. We only keep the last n entries of the vector and write it as $\tilde{\boldsymbol{w}}_{sc,t}$, which is the weight vector of the stocks based on the information provided by the short-term features. We then combine it with the weight vector of the stocks given by the main block as

$$\tilde{\boldsymbol{w}}_{2,t} = (1-\omega_1)\tilde{\boldsymbol{w}}_{sc,t} + \omega_1\tilde{\boldsymbol{w}}_{1,t}, \tag{8}$$

where $\omega_1 \in (0,1)$ measures the relative importance of the long-term allocation to the short-term one. To learn

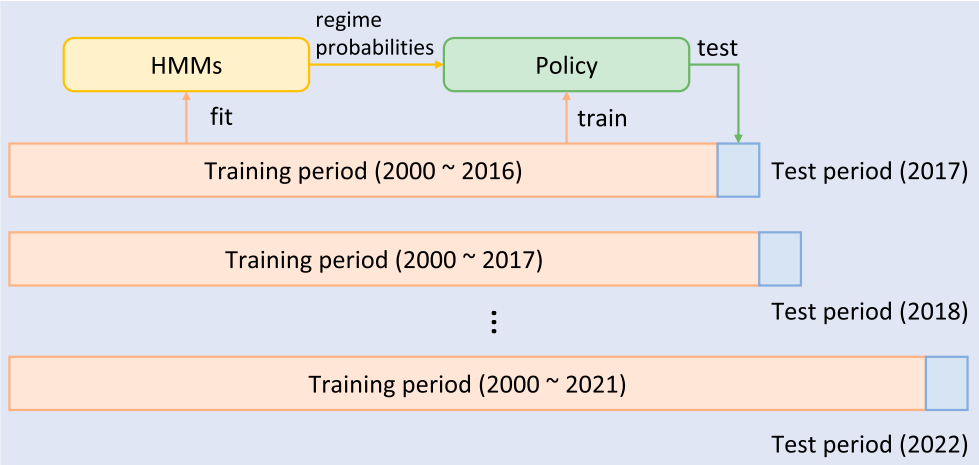


Figure 4. Training and testing scheme.

ω_1, we train a parameter $\theta_1 \in \mathbb{R}$ and map it by the Sigmoid function to obtain ω_1. By the construction in (8), $\tilde{\boldsymbol{w}}_{2,t}$ satisfies the constraints in (3).

(3) *Gate block*: This block is used to adjust the weight for each stock based on its own state. As shown in figure A1, a stock is not necessarily in the same state as the market. The gate block is an FNN that takes in the smoothed stock regime probability $\bar{p}_{i,t}$ and outputs a scalar $\tilde{g}_{i,t} \in (0,1)$ by using the Sigmoid function to activate the output layer. This block is shared by every stock to limit the network size and solve the data insufficiency problem. We stack all $\tilde{g}_{i,t}$ into a gate vector $\tilde{\boldsymbol{g}}_t \in (0,1)^n$ and update the weight vector of the stocks by

$$\tilde{\boldsymbol{w}}_{3,t} = \tilde{\boldsymbol{g}}_t \circ \tilde{\boldsymbol{w}}_{2,t}, \tag{9}$$

where $\circ$ denotes the elementwise product of two vectors. Thus, the weight of a stock can be significantly reduced compared with the result from the previous two blocks when it is in a bear state. From the construction in (9), $\tilde{\boldsymbol{w}}_{3,t}$ satisfies the constraints in (3).

(4) *Memory block*: If $\tilde{\boldsymbol{w}}_{3,t}$ is far from the current weight vector $\boldsymbol{w}_{S,t}$, trading would incur a significant transaction cost. Thus, knowing the current weights of the stocks is necessary for controlling transaction costs. We design a memory block to determine how much of $\boldsymbol{w}_{S,t}$ should be kept. It accepts the total turnover of stocks, defined by $\Delta w = \|\tilde{\boldsymbol{w}}_{3,t} - \boldsymbol{w}_{S,t}\|_1$ (vector L^1-norm), as the input and outputs a weight $\omega_p \in (0,1)$ by using the Sigmoid function to activate the output layer. The final output is given by a linear combination of $\tilde{\boldsymbol{w}}_{3,t}$ and $\boldsymbol{w}_{S,t}$ as

$$\tilde{\boldsymbol{w}}_{S,t} = (1-\omega_p)\tilde{\boldsymbol{w}}_{3,t} + \omega_p \boldsymbol{w}_{S,t},$$

where ω_p moderates the deviation from the current weight vector to reduce the impact of transaction costs. While the proposed weight change is $\tilde{\boldsymbol{w}}_{3,t} - \boldsymbol{w}_{S,t}$, after the memory block is applied, the weight change becomes $(1-\omega_p)(\tilde{\boldsymbol{w}}_{3,t} - \boldsymbol{w}_{S,t})$, i.e. a proportion of the proposed weight change given by ω_p is canceled by the memory block.

The output $\tilde{\boldsymbol{w}}_{S,t}$ depends on $\boldsymbol{w}_{S,t}$ and the weight vector $\boldsymbol{w}_{S,t+1}$ at the next time is obtained by (5). Training such structure involves backward propagation through time, similar to training recurrent neural networks, which may experience convergence issues. Note that when there is no transaction cost, i.e. $\rho = 0$, the memory block is not needed and our neural network only has the first three blocks, which can be trained relatively easily. Thus, we can adopt a convenient two-step training strategy. In the first step, we train the first three blocks in an environment with $\rho = 0$. In the second step, we train the memory block with the parameters in the first three blocks fixed.

We would like to understand the relative importance of different types of features. Therefore, we need to compare different policies generated by the neural network using different features. We consider four neural network policies, which are listed in table 1. The policies using only a subset of the features can be easily generated in our architecture by fixing some parameters to certain values and do not use some blocks. As an example, for NN-ST, we do not use the main and score blocks and fix $\omega_1 = 0$. See table 1 for a summary.

A natural benchmark for our dynamic neural network policies is the policy obtained by re-optimization. We find an optimal weight vector at the initial time and then buy and hold until a future date when the optimization problem is resolved with updated data and a new optimal weight vector is found. We can easily obtain the optimal weight vector in our framework by training a parameter vector $\boldsymbol{\theta} \in \mathbb{R}^{n+1}$ to minimize the objective function L and then set $\tilde{\boldsymbol{w}}_{S,t} = \mathrm{Softmax}(\boldsymbol{\theta})_{2:n+1}$.

Remark 3.1 In our design, we combine the weight vectors from some blocks in a linear way. One can certainly combine them in a nonlinear fashion, e.g. input two weight vectors into an FNN to generate a new weight vector. We choose linear combination for simplicity, which allows us to obtain a relatively parsimonious model.

4. Empirical study

We first consider three tracking problems for the S&P 500 index: IT, EIT, and EIT-CVaR, and provide in-depth analysis

Table 1. Neural network policies.

Policy	Features	Main block	Score block	Gate block	Memory block
NN-ST	$\boldsymbol{F}_{st,t}, \boldsymbol{w}_{S,t}$		✓		✓
NN-IR	$\bar{p}_{I,t}, \boldsymbol{w}_{S,t}$	✓			✓
NN-ISR	$\bar{p}_{I,t}, \bar{\boldsymbol{p}}_{S,t}, \boldsymbol{w}_{S,t}$	✓		✓	✓
NN-All	all	✓	✓	✓	✓

(see sections 4.1–4.3). We then consider another three important indexes, namely, S&P 100, FTSE 100, and Nikkei 225 to provide additional comparison between our method and RO.

4.1. Data, design of study, and performance measures

We collect daily adjusted closing prices from January 3, 2000, to December 30, 2022, and the free-float market capitalizations of all the constituent stocks of S&P 500 on December 30, 2016, from Refinitiv. The free-float market capitalization is the total market value of all outstanding shares in the market.

We filter out those stocks listed after January 3, 2000. For the remaining stocks, we rank them in terms of their market capitalization in decreasing order and select the top 20 stocks for trading in our study. Their tickers are given by AAPL, MSFT, XOM, BRK-B, JNJ, JPM, AMZN, WFC, GE, T, BAC, VZ, CVX, PG, PFE, KO, INTC, C, MRK, CMCSA. We consider two cases: trading with the top 5 stocks and trading with all 20 stocks. For the cash account, we assume its return is zero.

To make the most use of the data, we adopt a rolling training and testing scheme as shown in figure 4. At the beginning of every test year, we refit the HMMs of the index and the stocks and train a neural network policy using all the available training data dating back to January 3, 2000. We apply block bootstrap with overlapping blocks to generate training data. Specifically, given a long path of daily returns in $(n+1)$-dimension with a total of T_{total}, we collect all sub-periods with length $T_{train} = 200$ to construct $(T_{total} - T_{train} + 1)$ paths. We sample 500 paths as the validation set and use the remaining paths as the training set.

To evaluate and compare different policies, we consider the following performance measures, which are defined based on the daily portfolio returns $\{r_{P,t},\ t = 1, \ldots, T_{test}\}$ and daily index returns $\{r_{I,t}, t = 1, \ldots, T_{test}\}$ in the test period spanning a total of 6 years from the beginning of 2017 to the end of 2022. This period includes the infamous market meltdown in 2020 and the bear market in 2022 as well as several bullish sessions.

- *Tracking error (TE)*: $\sqrt{\frac{1}{T_{test}} \sum_{t=1}^{T_{test}} (r_{P,t} - r_{I,t})^2}$.
- *Mean excess return (MER)*: $\frac{1}{T_{test}} \sum_{t=1}^{T_{test}} (r_{P,t} - r_{I,t})$.
- *Loss*: For IT and EIT, the loss is given by TE $- \lambda \times$ MER. For EIT-CVaR, the loss is given by TE $- \lambda \times$ MER $+ \gamma (\text{CVaR}_\alpha(r_P) - c)_+$.
- *Information ratio (IR)*: The sample mean of daily excess returns divided by the sample standard deviation of daily excess returns, i.e.

$$\text{IR} := \frac{\text{MER}}{\text{std}(r_P - r_I)}.$$

- *Cumulative return (CR)*: $\prod_{t=1}^{T_{test}} (1 + r_t) - 1$, where $\{r_t,\ t = 1, \ldots, T_{test}\}$ is the daily return path.
- *Sharpe ratio*: We define the Sharpe ratio of our portfolio as

$$\text{Sharpe} := \frac{\text{mean}\,(r_P)}{\text{std}(r_P)}.$$

  Here, we ignore the risk-free rate because it is near zero for most of the test period.
- *Maximum drawdown (MDD)*: Let the running wealth and the maximum of wealth be v_t and M_t. We define

$$\text{MDD} := \max_{1 \le t \le T} \frac{M_t - v_t}{M_t},$$

  which gives the percentage drop from the most recent peak.
- *CVaR*: We consider the 95%-CVaR, which is estimated from the daily returns in the test period.
- *Average transaction cost per trade (ATC)*: If a trade occurs at time t, we incur a transaction cost of $\rho \sum_{i=1}^{n} |\tilde{w}_{i,t} - w_{i,t}|$ relative to the portfolio value $v_{p,t}$ before rebalancing. The average transaction cost per trade is the sample average of the costs incurred over all the trades in the 6 year test period.

MER, IR, CR, and Sharpe ratio are measures of return, while MDD and CVaR show the risk of the policies. Note that all the return measures are calculated by deducting transaction costs.

4.2. Experiment specifications

We implement all the policies, including the RO policy, using PyTorch (version 1.12), and train the models on a GPU (NVIDIA Tesla V100 with CUDA version 10.2).

We approximate the loss function for each problem by using the training data. Suppose that we have m paths of daily data with T_{train} days. We use the following sample-based loss functions as the objective for minimization (a scaling factor of 100 is applied for better convergence).

- *IT*:

$$\hat{L}_{\text{IT}} = 100 \times \sqrt{\frac{1}{mT_{train}} \sum_{t=1}^{T_{train}} \sum_{j=1}^{m} \left(r_{P,t}^{(j)} - r_{I,t}^{(j)} \right)^2}$$

- *EIT*:

$$\hat{L}_{\text{EIT}} = \hat{L}_{\text{IT}} - 100 \times \frac{\lambda}{mT_{train}} \sum_{t=1}^{T_{train}} \sum_{j=1}^{m} \left(r_{P,t}^{(j)} - r_{I,t}^{(j)} \right).$$

We use $\lambda = 20$ based on the result of the IT problem, which shows that the daily tracking error is at the 10^{-3} level while the daily excess return is at the 10^{-4} level. To balance the two goals, we use a relatively large value for λ, which does not underweigh the tracking objective from our experiment.

- *EIT-CVaR*:

$$\hat{L}_{\text{EIT-CVaR}} = \hat{L}_{\text{EIT}} + \gamma g_{\beta} \left(\frac{1}{\lceil \alpha m T_{train} \rceil} \sum_{t=1}^{T_{train}} \sum_{j=1}^{m} -r_{P,t}^{(j)} \mathbb{1}_{\left\{ r_{P,t}^{(j)} < r_{\alpha} \right\}} - c \right),$$

where r_α is the $\lceil \alpha m T_{train} \rceil$-smallest value for all $r_{P,t}^{(j)}, t = 1, \ldots, T_{train}, j = 1, \ldots, m$, which is the sample estimate of the negative of VaR. For other hyperparameters, we set $c = 3\%, \alpha = 5\%, \gamma = 10^6$ and $\beta = 2000$. We choose $c = 3\%$ as the 95%-CVaR of S&P 500 daily returns from 2000 to 2016 is around 3.0%. The choice of γ and β guarantees that the penalty term is comparable in size to the term $\hat{L}_{\text{EIT}}$ while remaining sufficiently strict for the CVaR constraint to be satisfied.

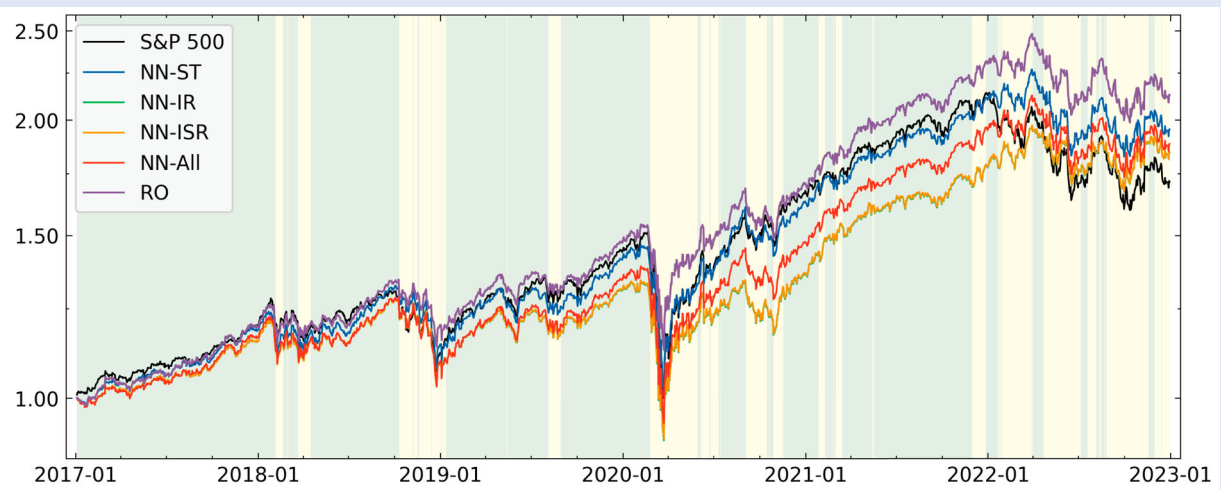
We present and discuss our choice of hyperparameters below.

- *Network hyperparameters*: We have tuned the network sizes on the validation set and report here the sizes used. For all the neural network policies, the main, gate, and memory blocks only have one hidden layer with 5 nodes, as their input is only one-dimensional. For the score block, as its input dimension is 5, we use two hidden layers with 5 nodes on each layer. In our tuning, we do not observe significant improvement by increasing the number of layers or number of neurons.

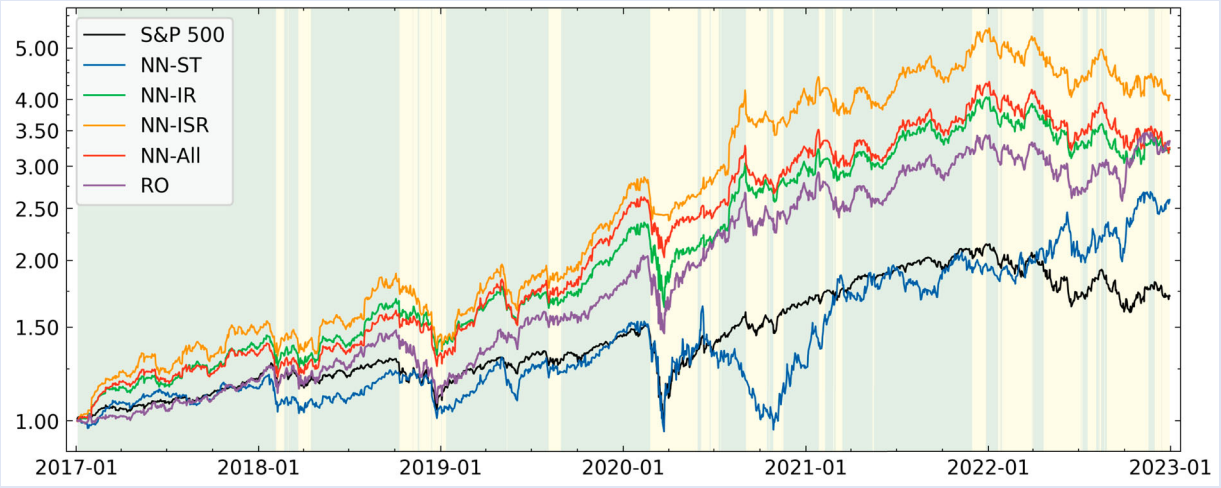
  For all the hidden layers, we use GELU (Hendrycks and Gimpel 2016) for activation. For the main, gate, and memory blocks, the activation function of the output layer is Sigmoid, while for the score block, it is the identity function. The numbers of model parameters to learn under different policies are listed in table 2. All the models are quite parsimonious. If $\rho = 0$, the model does not involve the memory block and hence has fewer parameters than the other case with positive ρ.
- *Rebalancing frequency*: We have tried different rebalancing frequencies on the training and validation set. While rebalancing frequently allows the investor to quickly adapt her portfolio to changes in the market, it can incur too high transaction costs. We find that setting the rebalancing time interval $T_{rb} = 5$ trading days achieves a similar performance in obtaining excess returns and controlling CVaR as $T_{rb} = 1$ trading day, but experiences much lower transaction costs. Rebalancing at lower frequencies can lead to significant downgrades in investment performance. Thus, we consider rebalancing every 5 trading days.
- *Time window*: The filtered regime probabilities can be too noisy to be directly used. We smooth the regime probability by taking an average of the probabilities in the last 2 trading days for the index and stocks. We also tried longer smoothing windows but did not observe improvement. For the short-term features, we calculate them using the data in the most recent 63 trading days, which is about 3 months.
- *Percentage of trading cost*: We consider two cases: $\rho = 0$ and $\rho = 0.005$ or 0.5%. In general, the transaction cost consists of market impact cost and trading fees. Frazzini *et al.* (2018) report that the average trading cost of large-cap US stocks does not exceed 20 basis points and Engle *et al.* (2012) also report similar cost levels. Thus, our choice of ρ is sufficient for understanding the impact of trading cost, but it can be more conservative than what is actually incurred in practice.
- *Initial weights*: The initial weight vector $\boldsymbol{w}_0$ before trading on the first test day, which is January 3, 2017, is initialized as $w_{i,0} = 0, i = 1, \ldots, n$ and $w_{0,0} = 1$, i.e. all the wealth is in the cash account. As we roll over, the initial weight vector for the next test year is given by the last weight vector in the year preceding it.
- *Stochastic gradient descent (SGD)*: To minimize the loss function, we use the SGD method with minibatches of 128 paths (each path has 200 days) and the Adam optimizer with learning rate given by 10^{-2}. All the neural network policies are trained for 50 epochs. For every epoch, we test the learned policy on the validation set, and the best set of model parameters in the 50 tests is kept for out-of-sample testing.

Table 2. Number of model parameters to learn for different policies.

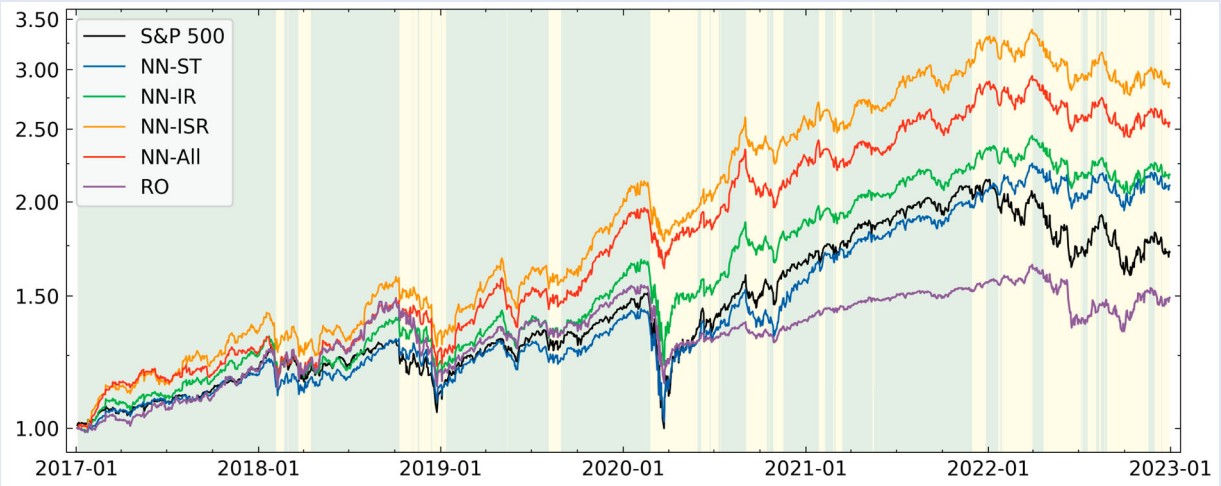
(a) $\rho = 0$		
Policy	$n = 5$	$n = 20$
NN-ST	66	66
NN-IR	28	58
NN-ISR	44	74
NN-All	111	141
RO	6	21
(b)$\rho = 0.005$		
Policy	$n = 5$	$n = 20$
NN-ST	82	82
NN-IR	44	74
NN-ISR	60	90
NN-All	127	157
RO	6	21



(a) IT



(b) EIT



(c) EIT-CVaR

Figure 5. Log-plots of cumulative wealth from trading 5 stocks for (enhanced) tracking of S&P 500 with $\rho = 0.005$. Areas shaded in green and yellow refer to bull and bear periods, respectively. (a) IT (b) EIT (c) EIT-CVaR.

Table 3. Policy performances (IT, S&P 500).

	TE	MER	IR	CVaR	CR	Sharpe	MDD	ATC
(a) 5 Stocks ($\rho = 0$)								
NN-ST	$\mathbf{3.82 \times 10^{-3}}$	1.22×10^{-4}	0.032	0.030	1.086	0.046	0.314	n.a.
NN-IR	4.62×10^{-3}	7.67×10^{-5}	0.017	0.030	0.943	0.042	0.329	n.a.
NN-ISR	4.62×10^{-3}	7.71×10^{-5}	0.017	0.030	0.946	0.042	0.327	n.a.
NN-All	3.94×10^{-3}	1.06×10^{-4}	0.027	0.031	1.021	0.044	0.323	n.a.
RO	3.87×10^{-3}	$\mathbf{1.48 \times 10^{-4}}$	**0.038**	**0.028**	**1.193**	**0.051**	**0.268**	n.a.
S&P 500				0.032	0.715	0.035	0.339	
(b) 5 Stocks ($\rho = 0.005$)								
NN-ST	$\mathbf{3.82 \times 10^{-3}}$	7.90×10^{-5}	0.021	0.030	0.956	0.043	0.315	2.15×10^{-4}
NN-IR	4.61×10^{-3}	4.18×10^{-5}	0.009	0.030	0.845	0.039	0.329	1.60×10^{-4}
NN-ISR	4.61×10^{-3}	4.18×10^{-5}	0.009	0.030	0.846	0.040	0.327	1.61×10^{-4}
NN-All	3.93×10^{-3}	5.96×10^{-5}	0.015	0.031	0.885	0.040	0.324	2.28×10^{-4}
RO	3.87×10^{-3}	$\mathbf{1.29 \times 10^{-4}}$	**0.033**	**0.028**	**1.130**	**0.049**	**0.269**	$\mathbf{9.67 \times 10^{-5}}$
S&P 500				0.032	0.715	0.035	0.339	
(c) 20 Stocks ($\rho = 0$)								
NN-ST	3.98×10^{-3}	-1.18×10^{-4}	-0.030	0.030	0.450	0.026	0.333	n.a.
NN-IR	4.37×10^{-3}	-1.23×10^{-4}	-0.028	0.030	0.438	0.026	0.341	n.a.
NN-ISR	4.31×10^{-3}	-1.34×10^{-4}	-0.031	**0.029**	0.422	0.026	0.331	n.a.
NN-All	3.61×10^{-3}	-9.94×10^{-5}	-0.028	0.030	0.486	0.027	0.344	n.a.
RO	$\mathbf{2.81 \times 10^{-3}}$	$\mathbf{-1.33 \times 10^{-5}}$	**−0.005**	0.030	0.696	0.035	**0.295**	n.a.
S&P 500				0.032	**0.715**	**0.035**	0.339	
(d) 20 Stocks ($\rho = 0.005$)								
NN-ST	4.00×10^{-3}	-1.49×10^{-4}	-0.037	0.030	0.384	0.024	0.334	1.34×10^{-4}
NN-IR	4.36×10^{-3}	-1.49×10^{-4}	-0.034	0.029	0.388	0.024	0.335	$\mathbf{1.09 \times 10^{-4}}$
NN-ISR	4.26×10^{-3}	-1.54×10^{-4}	-0.036	**0.029**	0.380	0.024	0.330	1.14×10^{-4}
NN-All	3.61×10^{-3}	-1.34×10^{-4}	-0.037	0.030	0.411	0.025	0.345	1.47×10^{-4}
RO	$\mathbf{2.81 \times 10^{-3}}$	$\mathbf{-4.62 \times 10^{-5}}$	**−0.016**	0.030	0.614	0.032	**0.296**	1.65×10^{-4}
S&P 500				0.032	**0.715**	**0.035**	0.339	

We implement the standard re-optimization approach and call the policy as RO. We can view RO as a special case in our framework, where the weight vector $\tilde{\boldsymbol{w}}_{S,t} := (\tilde{w}_{1,t}, \ldots, \tilde{w}_{n,t})$ after rebalancing gives us the decision variables to optimize and they do not depend on any features. In comparison, for our method $\tilde{\boldsymbol{w}}_{S,t}$ is generated by a neural network model whose parameters are the decision variables to optimize. We use the same rebalancing frequency for RO as for the neural network policies, i.e. rebalance every 5 trading days. At each rebalancing time, we minimize the objective by using the most recent two years of daily return data, from which we bootstrap with block size equal to 10 days. We train RO with the same minibatch size but for 200 epochs because there are fewer minibatches.

4.3. *Findings*

Performance comparison: The performance statistics of every policy for IT, EIT, and EIT-CVaR are shown in tables 3–5, respectively. In each table, the best result under a performance measure is highlighted. Figure 5 shows the wealth log-plots of the policies in the case of trading 5 stocks with $\rho = 0.005$. To save space, we do not show the wealth plots in the other cases as their trends are largely similar.

For IT, table 3 shows that NN-ST achieves the smallest TE with 5 stocks, but its improvement over RO is marginal. When the number of stocks is increased to 20, all the policies except NN-ST have a smaller TE, and RO is the best among all, which leads by a quite significant margin. In the long run, all the policies can track the index trend (see figure 5(a)). The insignificance of the neural network model over re-optimization is not difficult to understand. After all, IT is a regression-type problem. The single-period optimization model does regression exactly to construct the optimal tracking portfolio. Thus, the room for improvement by using a more sophisticated model is quite small given limited data.

For EIT, performances of the policies vary in different measures as demonstrated by table 4. In terms of MER, Sharpe ratio, and CR, NN-ISR, which utilizes both the index and stock regime probabilities, outperforms the others in all four cases (see figure 5(b)). Although it has a larger tracking error, it still tracks the index trend quite closely. Compared with RO, NN-ISR shows significant improvement in the return measures. In particular, we observe an increase about 60% to 70% in CR in the case with transaction cost. The performances of NN-IR and NN-All are also quite good in the return measures, and NN-IR achieves the highest IR in three out of four cases. As for the risk measures, all policies are more risky than the index in 95%-CVaR. They also experience quite large MDDs although some of them have a smaller MDD than the index. These results demonstrate the problem of the EIT formulation, which does not consider controlling risk. Although impressive excess returns are generated in the test period by some neural network policies, the lack of risk control makes it potentially dangerous to deploy them in practice. We also observe that NN-ST, which only considers short-term features, is an inferior choice in the return measures. It is also the most risky policy as measured by 95%-CVaR and MDD. These results highlight the importance of introducing index and stock regimes for EIT.

For EIT-CVaR, as demonstrated in table 5, the 95%-CVaR of all the policies are controlled below 3% (the constraint

Table 4. Policy performances (EIT, S&P 500).

	Loss	TE	MER	IR	CVaR	CR	Sharpe	MDD	ATC
(a) 5 Stocks ($\rho = 0$)									
NN-ST	5.73×10^{-3}	1.54×10^{-2}	4.82×10^{-4}	0.031	0.044	2.041	0.048	0.423	n.a.
NN-IR	$\mathbf{-4.15 \times 10^{-3}}$	$\mathbf{8.01 \times 10^{-3}}$	6.08×10^{-4}	**0.076**	0.035	3.077	0.069	0.309	n.a.
NN-ISR	-2.28×10^{-3}	1.24×10^{-2}	$\mathbf{7.35 \times 10^{-4}}$	0.059	0.034	**3.902**	**0.076**	0.284	n.a.
NN-All	-2.38×10^{-3}	1.01×10^{-2}	6.25×10^{-4}	0.062	0.032	3.246	0.073	**0.235**	n.a.
RO	-2.36×10^{-3}	9.47×10^{-3}	5.92×10^{-4}	0.063	0.039	2.807	0.061	0.283	n.a.
S&P 500					**0.032**	0.715	0.035	0.339	
(b) 5 Stocks ($\rho = 0.005$)									
NN-ST	7.27×10^{-3}	1.46×10^{-2}	3.66×10^{-4}	0.025	0.044	1.595	0.043	0.414	$\mathbf{4.06 \times 10^{-4}}$
NN-IR	$\mathbf{-1.61 \times 10^{-3}}$	$\mathbf{7.41 \times 10^{-3}}$	4.51×10^{-4}	**0.061**	0.035	2.230	0.059	0.306	4.33×10^{-4}
NN-ISR	1.92×10^{-4}	1.24×10^{-2}	$\mathbf{6.12 \times 10^{-4}}$	0.049	0.034	**3.079**	**0.068**	0.281	6.19×10^{-4}
NN-All	3.15×10^{-4}	9.40×10^{-3}	4.54×10^{-4}	0.048	0.033	2.278	0.061	**0.265**	6.51×10^{-4}
RO	-6.39×10^{-4}	9.48×10^{-3}	5.06×10^{-4}	0.053	0.039	2.344	0.056	0.285	4.30×10^{-4}
S&P 500					**0.032**	0.715	0.035	0.339	
(c) 20 Stocks ($\rho = 0$)									
NN-ST	2.72×10^{-2}	1.66×10^{-2}	-5.29×10^{-4}	-0.032	0.050	-0.374	-0.004	0.645	n.a.
NN-IR	$\mathbf{-2.10 \times 10^{-3}}$	$\mathbf{7.62 \times 10^{-3}}$	4.87×10^{-4}	**0.064**	0.035	2.376	0.060	0.366	n.a.
NN-ISR	-5.30×10^{-4}	1.19×10^{-2}	$\mathbf{6.22 \times 10^{-4}}$	0.052	0.038	**3.005**	**0.063**	0.372	n.a.
NN-All	-8.52×10^{-5}	1.16×10^{-2}	5.84×10^{-4}	0.050	0.038	2.801	0.062	0.372	n.a.
RO	-1.19×10^{-3}	9.31×10^{-3}	5.25×10^{-4}	0.057	0.039	2.455	0.057	**0.312**	n.a.
S&P 500					**0.032**	0.715	0.035	0.339	
(d) 20 Stocks ($\rho = 0.005$)									
NN-ST	2.75×10^{-2}	1.58×10^{-2}	-5.84×10^{-4}	-0.037	0.049	-0.418	-0.007	0.623	9.32×10^{-4}
NN-IR	1.21×10^{-3}	$\mathbf{7.42 \times 10^{-3}}$	3.11×10^{-4}	0.042	0.035	1.598	0.049	0.364	7.05×10^{-4}
NN-ISR	$\mathbf{1.18 \times 10^{-3}}$	1.16×10^{-2}	$\mathbf{5.21 \times 10^{-4}}$	**0.045**	0.038	**2.443**	**0.057**	0.359	7.66×10^{-4}
NN-All	1.89×10^{-3}	1.15×10^{-2}	4.82×10^{-4}	0.042	0.038	2.258	0.055	0.367	7.90×10^{-4}
RO	1.20×10^{-3}	9.31×10^{-3}	4.06×10^{-4}	0.044	0.039	1.884	0.050	**0.323**	$\mathbf{6.00 \times 10^{-4}}$
S&P 500					**0.032**	0.715	0.035	0.339	

Table 5. Policy performances (EIT-CVaR, S&P 500).

	Loss	TE	MER	IR	CVaR	CR	Sharpe	MDD	ATC
(a) 5 Stocks ($\rho = 0$)									
NN-ST	2.30×10^{-3}	$\mathbf{4.62 \times 10^{-3}}$	1.16×10^{-4}	0.025	0.028	1.091	0.048	0.317	n.a.
NN-IR	7.16×10^{-4}	5.46×10^{-3}	2.37×10^{-4}	0.043	0.027	1.532	0.062	0.269	n.a.
NN-ISR	$\mathbf{-1.03 \times 10^{-4}}$	8.92×10^{-3}	$\mathbf{4.51 \times 10^{-4}}$	**0.051**	0.025	**2.491**	**0.081**	0.184	n.a.
NN-All	5.72×10^{-4}	8.53×10^{-3}	3.98×10^{-4}	0.047	**0.024**	2.250	0.080	**0.160**	n.a.
RO	7.95×10^{-3}	6.84×10^{-3}	-5.64×10^{-5}	-0.008	0.026	0.658	0.039	0.258	n.a.
S&P 500					0.032	0.715	0.035	0.339	
(b) 5 Stocks ($\rho = 0.005$)									
NN-ST	2.56×10^{-3}	$\mathbf{4.88 \times 10^{-3}}$	1.16×10^{-4}	0.024	0.027	1.105	0.050	0.293	$\mathbf{8.29 \times 10^{-5}}$
NN-IR	2.36×10^{-3}	5.12×10^{-3}	1.38×10^{-4}	0.027	0.027	1.176	0.052	0.276	2.07×10^{-4}
NN-ISR	$\mathbf{1.76 \times 10^{-3}}$	8.09×10^{-3}	$\mathbf{3.16 \times 10^{-4}}$	**0.039**	0.026	**1.845**	**0.068**	0.199	4.98×10^{-4}
NN-All	3.31×10^{-3}	8.06×10^{-3}	2.37×10^{-4}	0.029	**0.024**	1.550	0.064	**0.189**	5.51×10^{-4}
RO	9.36×10^{-3}	6.85×10^{-3}	-1.26×10^{-4}	-0.018	0.026	0.492	0.032	0.260	3.50×10^{-4}
S&P 500					0.032	0.715	0.035	0.339	
(c) 20 Stocks ($\rho = 0$)									
NN-ST	8.63×10^{-3}	5.60×10^{-3}	-1.52×10^{-4}	-0.027	**0.024**	0.428	0.028	0.285	n.a.
NN-IR	7.19×10^{-3}	$\mathbf{5.07 \times 10^{-3}}$	-1.06×10^{-4}	-0.021	0.025	0.522	0.032	0.302	n.a.
NN-ISR	$\mathbf{2.52 \times 10^{-3}}$	7.68×10^{-3}	2.58×10^{-4}	**0.034**	0.025	1.625	**0.065**	**0.249**	n.a.
NN-All	2.65×10^{-3}	7.90×10^{-3}	$\mathbf{2.62 \times 10^{-4}}$	0.033	0.025	**1.635**	0.065	0.266	n.a.
RO	1.01×10^{-2}	6.66×10^{-3}	-1.70×10^{-4}	-0.026	0.027	0.382	0.026	0.263	n.a.
S&P 500					0.032	0.715	0.035	0.339	
(d) 20 Stocks ($\rho = 0.005$)									
NN-ST	9.22×10^{-3}	5.61×10^{-3}	-1.81×10^{-4}	-0.032	**0.024**	0.367	0.026	0.283	1.77×10^{-4}
NN-IR	7.06×10^{-3}	$\mathbf{5.08 \times 10^{-3}}$	-9.91×10^{-5}	-0.020	0.025	0.543	0.033	0.292	$\mathbf{3.40 \times 10^{-5}}$
NN-ISR	$\mathbf{4.43 \times 10^{-3}}$	7.24×10^{-3}	$\mathbf{1.40 \times 10^{-4}}$	**0.019**	0.026	**1.191**	**0.053**	**0.277**	5.36×10^{-4}
NN-All	5.13×10^{-3}	7.57×10^{-3}	1.22×10^{-4}	0.016	0.026	1.131	0.052	0.296	6.29×10^{-4}
RO	1.23×10^{-2}	6.68×10^{-3}	-2.79×10^{-4}	-0.042	0.027	0.173	0.015	0.286	5.41×10^{-4}
S&P 500					0.032	0.715	0.035	0.339	

level), which are significantly lower than in EIT. Under every policy, MDD also shows noticeable decrease compared with the results in EIT although the problem does not directly control MDD. In particular, the MDD of NN-ISR or NN-All is only about 19% in the five-stock case with transaction costs, much lower than the MDD of S&P 500, which is 33.9%. The reduction in risk is achieved by sacrificing excess returns as shown in the drop of MER and CR. Nevertheless, NN-ISR and

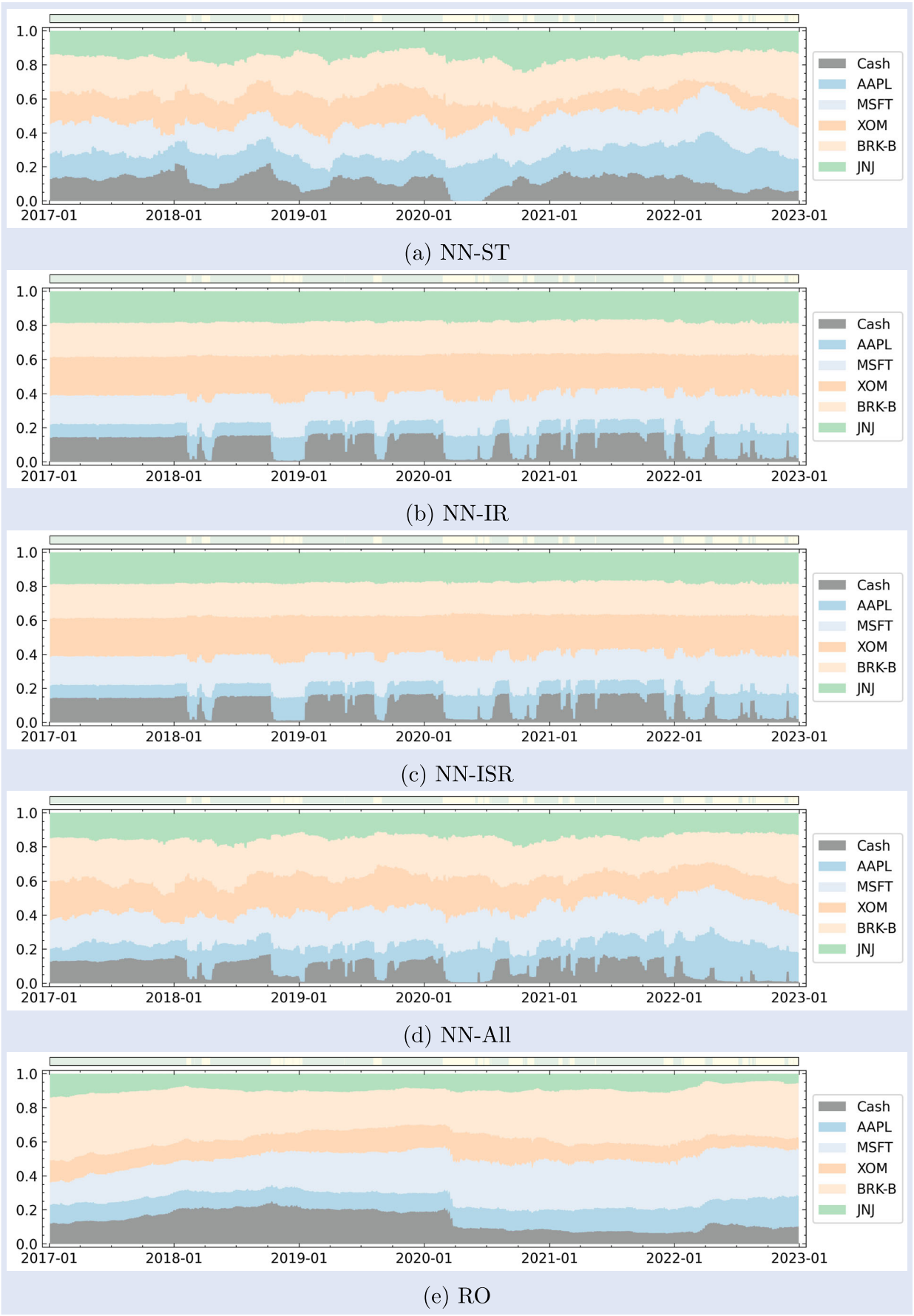


Figure 6. Rebalanced weights $\tilde{\boldsymbol{w}}_t$ (IT, 5 Stocks, $\rho = 0.005$, S&P 500) (a) NN-ST (b) NN-IR (c) NN-ISR (d) NN-All (e) RO.

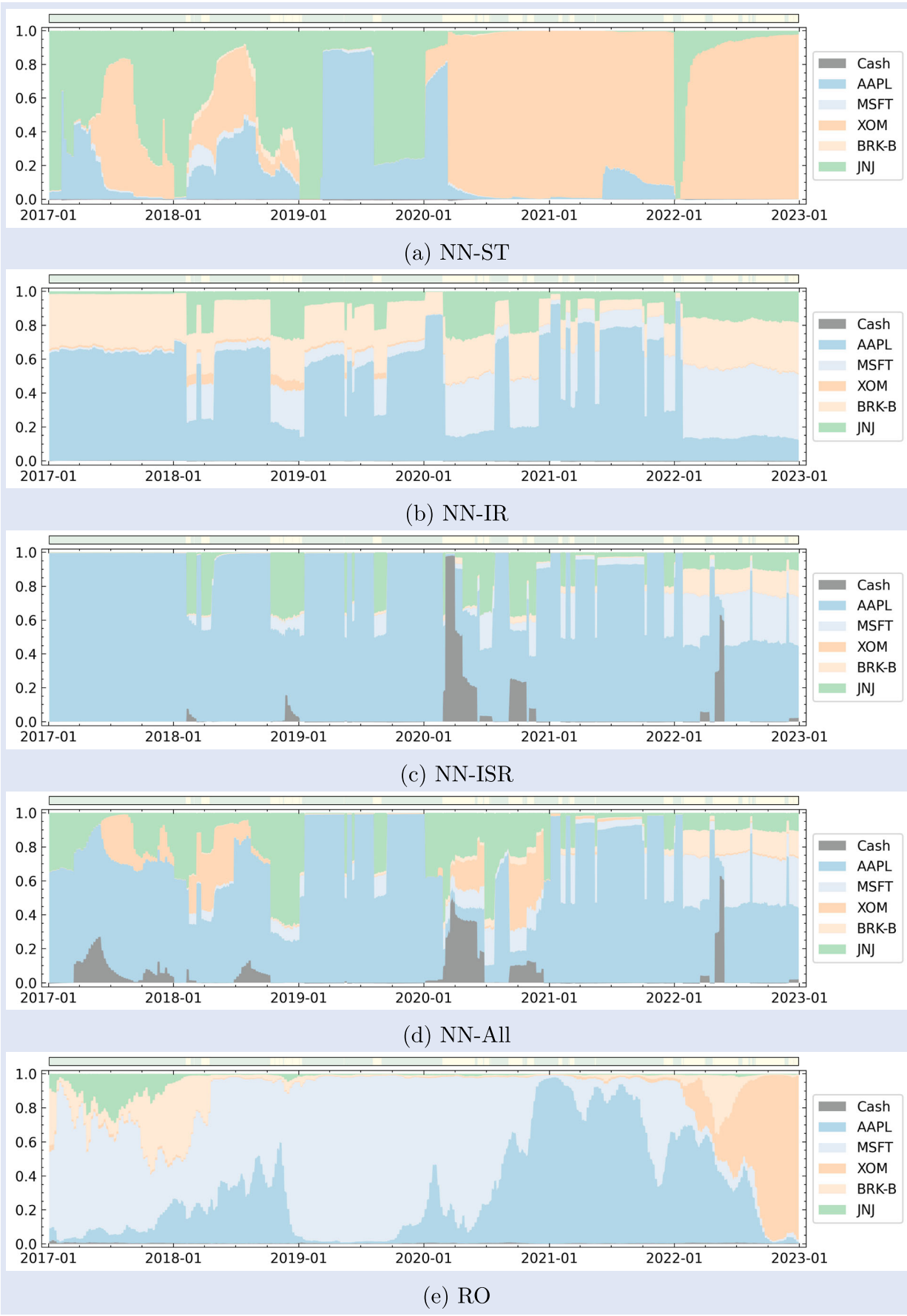


Figure 7. Rebalanced weights $\tilde{\boldsymbol{w}}_t$ (EIT, 5 Stocks, $\rho = 0.005$, S&P 500) (a) NN-ST (b) NN-IR (c) NN-ISR (d) NN-All (e) RO.

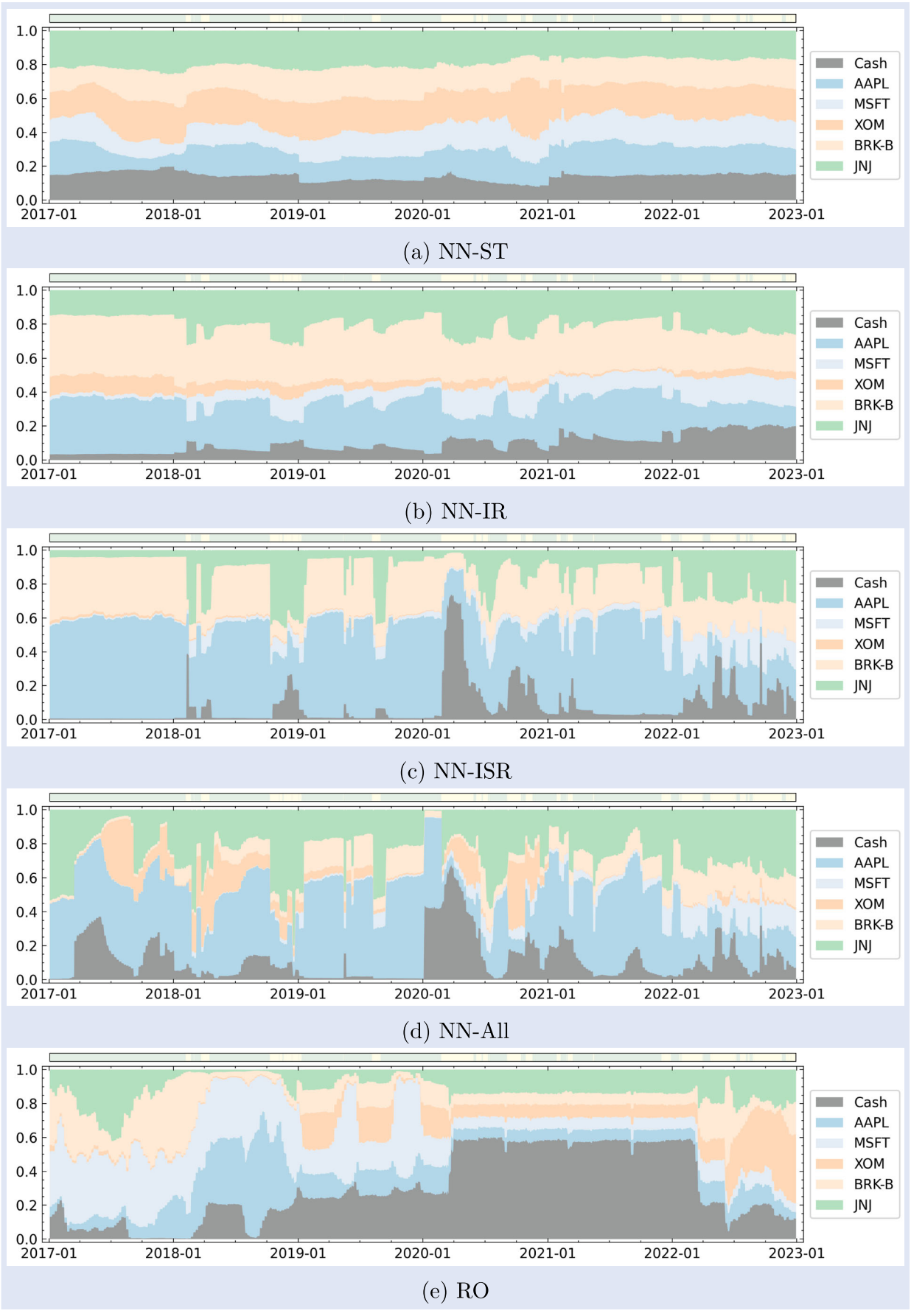
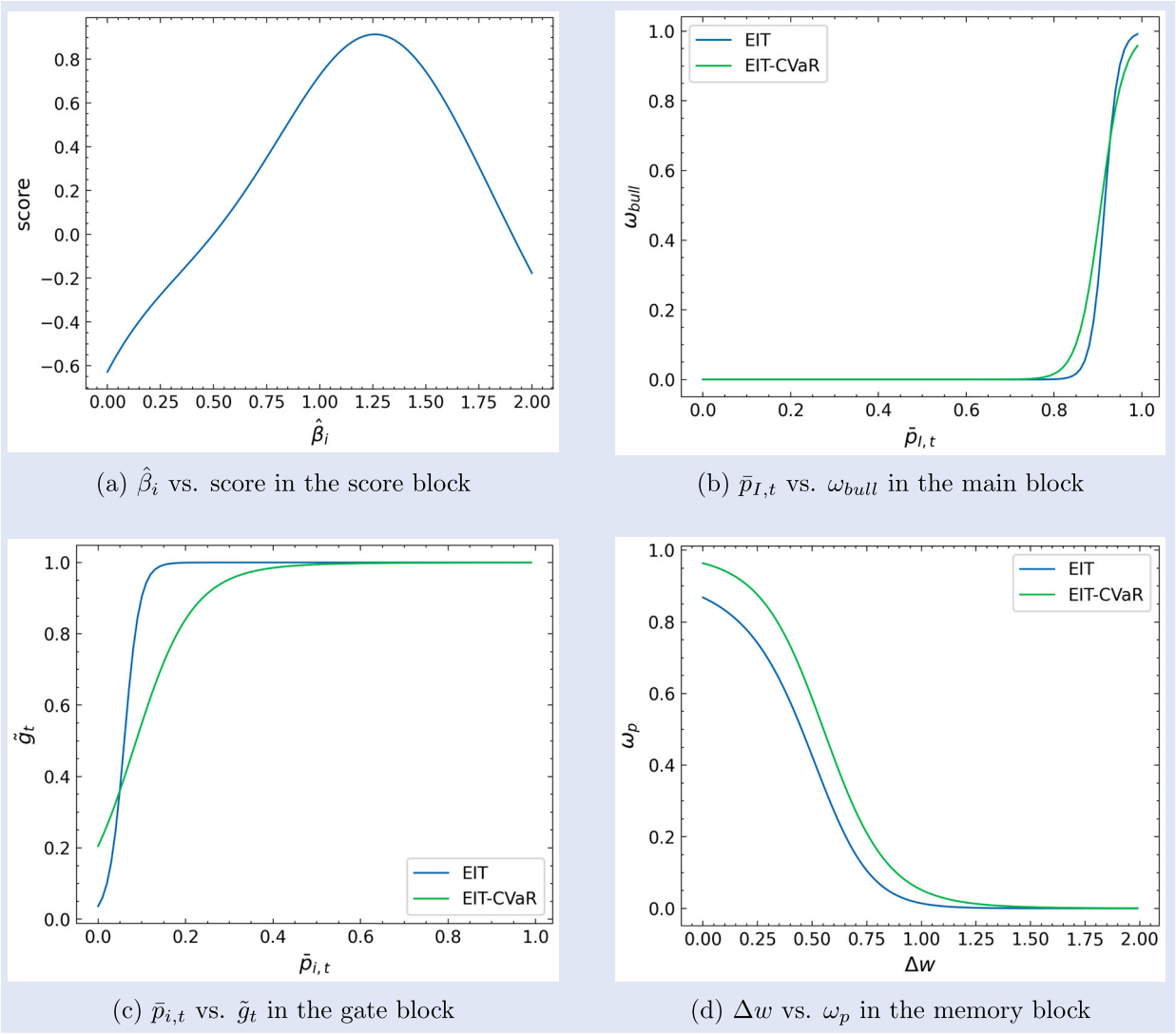


Figure 8. Rebalanced weights $\tilde{\boldsymbol{w}}_t$ (EIT-CVaR, 5 Stocks, $\rho = 0.005$, S&P 500) (a) NN-ST (b) NN-IR (c) NN-ISR (d) NN-All (e) RO.



(a) $\hat{\beta}_i$ vs. score in the score block

(b) $\bar{p}_{I,t}$ vs. ω_{bull} in the main block

(c) $\bar{p}_{i,t}$ vs. $\tilde{g}_t$ in the gate block

(d) Δw vs. ω_p in the memory block

Figure 9. Input-output curves of the neural network blocks in the 5 stock case for (enhanced) tracking of S&P 500 trained with data from year 2000 to 2016. In plot (a), we fix the daily mean as 0 and daily volatility as 1% for the stock and the index, and the curve is obtained from the learned result in the IT problem. (a) $\hat{\beta}_i$ vs. score in the score block (b) $\bar{p}_{I,t}$ vs. ω_{bull} in the main block (c) $\bar{p}_{i,t}$ vs. $\tilde{g}_t$ in the gate block (d) Δw vs. ω_p in the memory block.

NN-All still exhibit impressive excess returns and perform better than the others in terms of MER, IR, Sharpe ratio, and CR (see figure 5(c)). In particular, they demonstrate significant advantages over RO in these return measures. Being mindful of the penalty from violating the CVaR constraint, RO becomes overly conservative, leading to poor performance in obtaining excess returns. NN-IR and NN-ST have a smaller TE than NN-ISR and NN-All, but their performances in the return measures are not so impressive.

Stock selection apparently matters for the return performance in EIT without or with the CVaR constraint. By choosing stocks according to their market capitalization, our results show that increasing the number of stocks from 5 to 20 does not improve the return performance for enhanced tracking of S&P 500.

Learned policies: We plot the rebalanced weights of trading 5 stocks in the test period that are generated by the five policies under $\rho = 0.005$ in figures 6–8 for IT, EIT, EIT-CVaR, respectively.

For IT, figure 6 shows that all neural network policies invest in the 5 stocks and the cash account quite evenly and update their weights slowly. The good performance of NN-ST should be credited to the score block. Figure 9(a) plots the beta-score relationship that shows the score block tends to assign a higher score to the stocks with beta around one. This is expected as such stocks resemble the index more than the others.

For EIT, figure 7 shows that all the policies tend to concentrate on some stocks, thereby increasing risk. Adding the CVaR constraint leads to more diversification as seen from figure 8, but the percentage of AAPL is still very high under NN-ISR and NN-All to pursue excess returns.

NN-IR, NN-ISR, and NN-All change their allocations considerably as the market switches between bull and bear. NN-ISR and NN-All, which utilize both the index and stock regime probabilities, show more flexibility in weight allocation by holding more in the high-return stocks in the bull market for faster growth, and investing less in the stocks in the bear market to stem losses. We also observe that during the 2020 market meltdown, which is an extreme crash, these two policies relocate a substantial amount of investment from the stocks to the cash account (even in the case without the CVaR constraint), thereby incurring smaller losses than the

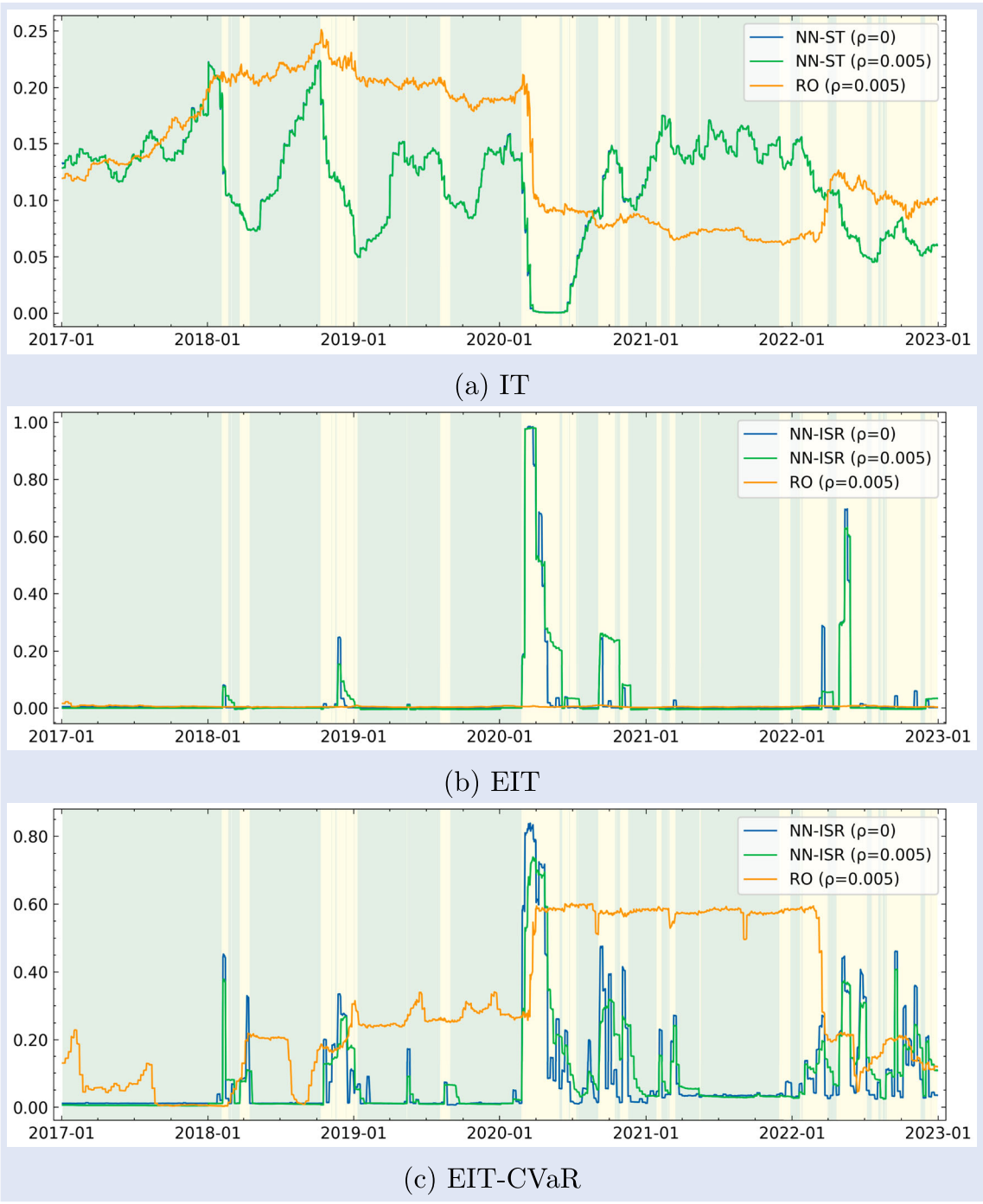


Figure 10. Weight of cash under neural network and RO policies in the 5 stock case for (enhanced) tracking of S&P 500. Areas shaded in green and yellow refer to bull and bear periods, respectively. (a) IT (b) EIT (c) EIT-CVaR.

other policies and the index as shown in figure 5(b,c). In contrast, without considering the CVaR constraint, the phenomenon of flight to safety cannot be generated by RO; see figure 10(b) where the weight of cash remains at very low levels throughout the test period. By adding the CVaR constraint, figure 10(c) shows that RO also generates flight to safety in early 2020, but only at some time after the bear period began.† In comparison, the neural network policy responds to the regime change timely thanks to the signal from the regime probability of the index. Furthermore, as the market recovers from the 2020 crash, the neural network policy perceives this change and substantially increases investment in the stocks to profit from the market rally, whereas RO remains conservative and continues to hold heavily in cash for a prolonged period.‡ These comparisons clearly show the advantages of our approach over RO.

† For RO, we re-optimize every five trading days to rebalance the tracking portfolio. Many papers in the literature consider lower rebalancing frequencies, such as quarterly or semiannually, leading to strategies that are even less responsive to regime changes.

‡ The RO policy remains conservative for a long time due to the existence of large negative returns from the 2020 market crash in the sample, which results in a large estimate of CVaR of the stock portfolio. As RO is unaware of the true regime of the market, the strategy is forced to hold a lot of cash to control tail risk.

Table 6. Policy performances (EIT-CVaR, S&P 100).

	Loss	TE	MER	IR	CVaR	CR	Sharpe	MDD	ATC
(a) 5 Stocks									
NN-ISR	$\mathbf{-3.24 \times 10^{-3}}$	1.01×10^{-2}	$\mathbf{6.69 \times 10^{-4}}$	**0.066**	0.044	**3.130**	**0.069**	0.341	$\mathbf{2.94 \times 10^{-5}}$
RO	1.14×10^{-2}	$\mathbf{6.39 \times 10^{-3}}$	-2.50×10^{-4}	−0.039	**0.021**	0.566	0.046	**0.183**	9.80×10^{-5}
S&P 100					0.033	1.000	0.047	0.315	
(b) 20 Stocks									
NN-ISR	$\mathbf{6.67 \times 10^{-3}}$	$\mathbf{6.20 \times 10^{-3}}$	$\mathbf{-2.34 \times 10^{-5}}$	**−0.004**	0.028	**1.000**	**0.053**	0.260	4.67×10^{-4}
RO	1.66×10^{-2}	7.13×10^{-3}	-4.71×10^{-4}	−0.066	**0.023**	0.177	0.019	**0.248**	$\mathbf{1.73 \times 10^{-4}}$
S&P 100					0.033	1.000	0.047	0.315	
(c) 40 Stocks									
NN-ISR	$\mathbf{7.37 \times 10^{-3}}$	$\mathbf{5.84 \times 10^{-3}}$	$\mathbf{-7.62 \times 10^{-5}}$	**−0.013**	0.030	0.857	0.046	0.327	5.75×10^{-4}
RO	1.63×10^{-2}	6.98×10^{-3}	-4.65×10^{-4}	−0.067	**0.023**	0.184	0.019	**0.241**	$\mathbf{2.73 \times 10^{-4}}$
S&P 100					0.033	**1.000**	**0.047**	0.315	

We illustrate the input-output relationships in the main block and the gate block in figure 9(b,c), respectively. The main block acts like a switch in the sense that only when the filtered probability of the bull state for the index is sufficiently large a significant positive weight is given to $\tilde{\boldsymbol{w}}_{bull}$. The gate block serves as a gate that reduces the proposed weight of the stock based on its current state. The gate is more conservative in EIT-CVaR than in EIT, as it starts to discount the proposed weight at higher levels of the filtered probability of the stock due to the need of risk control.

Feature importance: Based on the discussions above, we highlight the important features. For IT, the short-term features are the most useful in reducing the tracking error. For EIT and EIT-CVaR, it is important to consider both index and stock regime probabilities, given that NN-ISR is the best policy in most of the return measures among all and its improvement over NN-IR is sizable. A comparison of the strategies of NN-ISR and NN-IR in the EIT-CVaR problem further illuminates the importance of stock regimes; compare figure 8(b,c). During the unprecedented stock market meltdown in March 2020, NN-ISR holds considerably more cash than NN-IR. Although NN-IR perceives the bear market, its strategy is learned from historical bear periods dating back to 2000, where the magnitude of crash is much smaller than in 2020. Thus, NN-IR underestimates the cash required. In contrast, after observing the bad performance of the stocks in the meltdown, NN-ISR further reduces their holdings through the gate block and thus increases the cash weight substantially to protect the portfolio. This mechanism enables NN-ISR to avoid heavy losses in a big crash (see figure 5(c)). Finally, it is worth mentioning that only considering short-term features do not lead to satisfactory results for EIT and EIT-CVaR.

Our results underscore the importance of understanding the regimes of the market and stocks for investment. The idea is that we want to catch the bull trend to increase profits as well as liquidate in a bear period to stem losses as early as possible. In hindsight, these regimes are easy to perceive, but the challenge lies in how to identify them at an early stage. One approach is to filter the probabilities of the regimes from price data as trading signals to guide investment as done in this paper. Alternatively, we can consider using additional information besides the price data to construct features to predict the probabilities of the regimes by a machine learning model. Our framework is flexible and the features input into our neural network model can be obtained from other models.

Impact of transaction costs: Overall, the ATC per trade of every neural network policy in all three problems is at most several basis points relative to the portfolio value, hence moderate.

In IT, the transaction costs of all policies are at low levels (see table 3), as the policies tend to change weights slowly (see figure 6).

For EIT and EIT-CVaR, the neural network policies tend to change weights more dramatically and frequently (see figures 7 and 8). Hence, transaction costs have an obvious influence on return performance. Nevertheless, the neural network policies that use appropriate features still show advantages to the RO policy, which confirms that our deep learning model can effectively control transaction costs through the memory block. It is also worth mentioning that NN-ISR, the neural network policy with the best return performance, is not the most costly one.

We plot the input-output relationship of the memory block in figure 9(d). Recall that w_p shows the proportion of proposed change canceled. Thus, the memory block has a larger moderating effect for small weight changes than for large ones. Typically, when large weight changes are proposed, it signals a significant shift in the market conditions that requires such a big change. In this situation, transaction costs are less of a concern.

We further illustrate the effect of the memory block in figure 10 by showing how the weight of cash (equivalently, the total weight of stocks) varies without and with transaction costs. For EIT and EIT-CVaR, the memory block makes the weight change more smoothly, thereby reducing large sudden changes to control transaction costs.

REMARK 4.1 Using data in the test period, we performed three commonly used stationarity tests, augmented Dickey-Fuller, Phillips-Perron, and KPSS on the portfolio return process generated by the best policy: NN-ST for IT and NN-ISR for EIT and EIT-CVaR. The results under all these tests support stationarity.

REMARK 4.2 For EIT, investment in the cash account is only significant when the index and the stocks in the tracking portfolio undergo bear states as seen from figures 7 and 8. In our experiment, we set the cash return to be zero. If it is positive, it

Table 7. Policy performances (EIT-CVaR, FTSE 100).

	Loss	TE	MER	IR	CVaR	CR	Sharpe	MDD	ATC
(a) 5 Stocks									
NN-ISR	$\mathbf{6.14 \times 10^{-3}}$	7.95×10^{-3}	$\mathbf{1.81 \times 10^{-5}}$	**0.002**	0.023	0.223	0.021	0.287	6.43×10^{-5}
RO	6.49×10^{-3}	$\mathbf{6.98 \times 10^{-3}}$	4.93×10^{-6}	0.001	**0.019**	**0.228**	**0.024**	**0.198**	$\mathbf{6.03 \times 10^{-5}}$
FTSE 100					0.028	0.177	0.017	0.350	
(b) 20 Stocks									
NN-ISR	$\mathbf{6.37 \times 10^{-3}}$	5.54×10^{-3}	$\mathbf{-8.34 \times 10^{-6}}$	**−0.002**	0.024	**0.180**	**0.018**	**0.268**	1.17×10^{-4}
RO	1.55×10^{-2}	$\mathbf{4.70 \times 10^{-3}}$	-1.08×10^{-4}	−0.023	**0.023**	0.051	0.009	0.302	$\mathbf{1.09 \times 10^{-4}}$
FTSE 100					0.028	0.177	0.017	0.350	
(c) 40 Stocks									
NN-ISR	$\mathbf{-1.27 \times 10^{-3}}$	9.13×10^{-3}	$\mathbf{1.04 \times 10^{-4}}$	**0.011**	**0.021**	**0.365**	**0.030**	**0.272**	4.46×10^{-4}
RO	1.35×10^{-3}	$\mathbf{3.65 \times 10^{-3}}$	2.29×10^{-5}	0.006	0.026	0.226	0.021	0.319	$\mathbf{1.20 \times 10^{-4}}$
FTSE 100					0.028	0.177	0.017	0.350	

Table 8. Policy performances (EIT-CVaR, Nikkei 225).

	Loss	TE	MER	IR	CVaR	CR	Sharpe	MDD	ATC
(a) 5 Stocks									
NN-ISR	1.13×10^{-2}	$\mathbf{8.34 \times 10^{-3}}$	-3.02×10^{-5}	−0.004	0.027	0.636	0.039	0.297	$\mathbf{6.36 \times 10^{-5}}$
RO	$\mathbf{1.05 \times 10^{-2}}$	8.97×10^{-3}	$\mathbf{-1.58 \times 10^{-5}}$	**−0.002**	0.028	0.673	0.041	**0.264**	9.70×10^{-5}
Nikkei 225					**0.027**	**0.705**	**0.042**	0.313	
(b) 20 Stocks									
NN-ISR	$\mathbf{1.15 \times 10^{-2}}$	$\mathbf{7.23 \times 10^{-3}}$	$\mathbf{-4.30 \times 10^{-5}}$	**−0.006**	0.027	0.609	0.037	**0.250**	1.79×10^{-4}
RO	2.18×10^{-2}	7.51×10^{-3}	-1.43×10^{-4}	−0.019	0.027	0.438	0.031	0.289	$\mathbf{1.45 \times 10^{-4}}$
Nikkei 225					**0.027**	**0.705**	**0.042**	0.313	
(c) 40 Stocks									
NN-ISR	$\mathbf{1.31 \times 10^{-2}}$	$\mathbf{5.82 \times 10^{-3}}$	$\mathbf{-7.37 \times 10^{-5}}$	**−0.013**	**0.026**	0.566	0.037	**0.256**	$\mathbf{1.72 \times 10^{-4}}$
RO	2.12×10^{-2}	6.81×10^{-3}	-1.44×10^{-4}	−0.021	0.027	0.436	0.031	0.290	1.79×10^{-4}
Nikkei 225					**0.027**	**0.705**	**0.042**	0.313	

would hardly change our allocation in the bull periods because in reality the cash return is so small relative to returns provided by stocks. In the bear periods, our portfolio would have a higher value because the money in the cash account now earns some positive return. Thus, we expect that the value of our tracking portfolio at the end of the test period would be even greater than what is reported in tables 4 and 5, hence further boosting the excess return of our portfolio over the index.

4.4. Results for S&P 100, FTST 100, and Nikkei 225

We have observed significant improvement of NN-ISR over the RO method in the study for S&P500. In this subsection, we evaluate their performances for S&P 100, FTST 100, and Nikkei 225 to further compare them.

We adopt the same rolling training and testing scheme used for S&P 500. Our training period starts from year 2003 because the data of many constituent stocks are unavailable before this year. The performance of each method is tested from 2019 to 2023.

We consider the EIT-CVaR problem with transaction costs ($\rho = 0.5\%$). We set the constraint level for the 95% CVaR to be 3% for S&P 100, 2.7% for FTSE 100, and 3.3% for Nikkei 225, which are around the CVaR estimates of these indexes in the training period. For each index, the stock portfolio is constructed by choosing the largest n constituent stocks by market capitalization and we consider $n = 5, 20, 40$. In total, we have 9 cases to compare NN-ISR and RO.

The results are shown in tables 6–8 for S&P 100, FTST 100, and Nikkei 225, respectively. Except for two cases (5 stocks for enhanced tracking of FTSE 100 and Nikkei 225), NN-ISR generates lower test loss (which shows the combined effect of tracking error, excess return, and violation of CVaR constraint) and higher CR and Sharpe ratio than RO. NN-ISR also achieves higher MER and IR than RO in 8 out of 9 cases (except the 5 stock case for Nikkei 225). Thus, the improvement of NN-ISR over RO is quite robust to changes in the index and number of stocks.

We note that both NN-ISR and RO fail to generate excess CR over the Nikkei 225 index. This is mainly due to two reasons. First, we choose stocks by their market capitalization and it turns out the large-cap stocks underlying Nikkei 225 perform quite poorly. Second, we impose the CVaR constraint in the problem, which makes the learned strategy more conservative so that it tends to sacrifice return for lower risk. In general, our method provides a flexible framework to adjust the enhanced tracking strategy between bull and bear markets for selected stocks. Further research should be carried out to improve stock selection to boost return performance in the case where large-cap stocks do not perform well.

5. Conclusion

We have proposed a novel deep learning method for EIT and its novelty is twofold. First, we introduce index and stock regimes as features, which were ignored by previous works on EIT. These features allow our trading policy to quickly adapt

to changes in the market, which is key to boosting investment performance. Second, we propose a novel design of the neural network architecture that makes our deep learning model parsimonious, scalable, and convenient for training and also improves its interpretability. Our approach is essentially data driven as it does not require a specific parametric model for asset returns, except that a hidden Markov model is used to filter the regime probabilities. Our framework is also flexible as it can easily incorporate new features by adding blocks in the neural network architecture. Furthermore, it can deal with various types of constraints by either choosing proper activation functions or formulating them as penalties in the objective. Empirical results demonstrate the significant advantages of our approach over standard re-optimization for EIT. For future research, there are two interesting directions to explore. One is considering alternative models for predicting the stock and index regimes, which is done by filtering in our study. The other direction is improving the selection of stocks, which clearly matters for the performance of EIT.

Acknowledgments

We thank the reviewers for their comments and suggestions which led to significant improvement in the paper.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This research was supported by the Hong Kong Research Grant Council under General Research Fund [grant number 14206020].

ORCID

Lingfei Li http://orcid.org/0000-0002-1761-3458

References

Al-Aradi, A. and Jaimungal, S., Outperformance and tracking: Dynamic asset allocation for active and passive portfolio management. *Appl. Math. Finance*, 2018, **25**(3), 268–294.

Al-Aradi, A. and Jaimungal, S., Active and passive portfolio management with latent factors. *Quant. Finance*, 2021, **21**(9), 1437–1459.

Balduzzi, P. and Lynch, A.W., Transaction costs and predictability: Some utility cost calculations. *J. Financ. Econ.*, 1999, **52**(1), 47–78.

Bamberg, G. and Wagner, N., Equity index replication with standard and robust regression estimators. *OR Spectrum*, 2000, **22**(1), 525–543.

Baum, L.E., Petrie, T., Soules, G. and Weiss, N., A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Ann. Math. Statist.*, 1970, **41**(1), 164–171.

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

Benidis, K., Feng, Y. and Palomar, D.P., Optimization methods for financial index tracking: From theory to practice. *Foundations Trends Optim.*, 2018, **3**(3), 171–279.

Bradrania, R., Pirayesh Neghab, D. and Shafizadeh, M., State-dependent stock selection in index tracking: A machine learning approach. *Financial Markets Portfolio Manage.*, 2022, **36**(1), 1–28.

Buehler, H., Gonon, L., Teichmann, J. and Wood, B., Deep hedging. *Quant. Finance*, 2019, **19**(8), 1271–1291.

Cai, Z. and Wang, X., Nonparametric estimation of conditional VaR and expected shortfall. *J. Econom.*, 2008, **147**(1), 120–130.

Canakgoz, N.A. and Beasley, J.E., Mixed-integer programming approaches for index tracking and enhanced indexation. *Eur. J. Oper. Res.*, 2009, **196**(1), 384–399.

Carbonneau, A. and Godin, F., Equal risk pricing of derivatives with deep hedging. *Quant. Finance*, 2021, **21**(4), 593–608.

Chabakauri, G. and Rytchkov, O., Asset pricing with index investing. *J. Financ. Econ.*, 2021, **141**(1), 195–216.

Chen, J. and Li, L., Data-driven hedging of stock index options via deep learning. *Oper. Res. Lett.*, 2023, **51**(4), 408–413.

Coleman, T.F., Li, Y. and Henniger, J., Minimizing tracking error while restricting the number of assets. *J. Risk*, 2006, **8**(4), 33–55.

Coles, J.L., Heath, D. and Ringgenberg, M.C., On index investing. *J. Financ. Econ.*, 2022, **145**(3), 665–683.

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

Dai, Z., Li, L. and Zhang, G., Evaluation of deep learning algorithms for quadratic hedging. *J. Derivatives*, 2022, **30**(1), 32–57.

Dai, M., Xu, Z.Q. and Zhou, X.Y., Continuous-time Markowitz's model with transaction costs. *SIAM J. Financial Math.*, 2010a, **1**(1), 96–125.

Dai, M., Yang, Z., Zhang, Q. and Zhu, Q.J., Optimal trend following trading rules. *Math. Oper. Res.*, 2016, **41**(2), 626–642.

Dai, M., Zhang, Q. and Zhu, Q.J., Trend following trading under a regime switching model. *SIAM J. Financial Math.*, 2010b, **1**(1), 780–810.

Dose, C. and Cincotti, S., Clustering of financial time series with application to index and enhanced index tracking portfolio. *Phys. A: Statistical Mechanics and Its Applications*, 2005, **355**(1), 145–151.

Engle, R., Ferstenberg, R. and Russell, J., Measuring and modeling execution cost and risk. *J. Portfolio Manage.*, 2012, **38**(2), 14–28.

Filippi, C., Guastaroba, G. and Speranza, M., A heuristic framework for the bi-objective enhanced index tracking problem. *Omega*, 2016, **65**(1), 122–137.

Frazzini, A., Israel, R. and Moskowitz, T.J., Trading costs. Available at SSRN 3229719, 2018.

Guastaroba, G., Mansini, R., Ogryczak, W. and Speranza, M.G., Enhanced index tracking with cvar-based ratio measures. *Ann. Oper. Res.*, 2020, **292**(2), 883–931.

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

Guidolin, M. and Timmermann, A., Asset allocation under multivariate regime switching. *J. Econ. Dyn. Control*, 2007, **31**(11), 3503–3544.

Guidolin, M. and Timmermann, A., International asset allocation under regime switching, skew, and kurtosis preferences. *Rev. Financ. Stud.*, 2008, **21**(2), 889–935.

Guéant, O. and Manziuk, I., Deep reinforcement learning for market making in corporate bonds: Beating the curse of dimensionality. *Appl. Math. Finance*, 2019, **26**(5), 387–452.

Hendrycks, D. and Gimpel, K., Gaussian error linear units (GELUs), 2016. Preprint arXiv:1606.08415.

Hong, Y., Kim, Y., Kim, J. and Choi, Y., Index tracking via learning to predict market sensitivities, 2022. Preprint arXiv:2209.00780.

Huang, J., Li, Y. and Yao, H., Index tracking model, downside risk and non-parametric kernel estimation. *J. Econ. Dyn. Control*, 2018, **92**(1), 103–128.

Jiang, P. and Perez, M.F., Follow the leader: Index tracking with factor models. *J. Empirical Finance*, 2021, **64**(1), 337–350.
Kim, S., Deep time series forecasting for enhanced index tracking. *Appl. Econ.*, 2021, **53**(17), 1916–1934.
Kim, S. and Kim, S., Index tracking through deep latent representation learning. *Quant. Finance*, 2020, **20**(4), 639–652.
Kim, C.-J. and Nelson, C.R., *State-Space Models with Regime Switching: Classical and Gibbs-Sampling Approaches with Applications*, 2017 (MIT Press: Cambridge, MA).
Kwak, Y., Song, J. and Lee, H., Neural network with fixed noise for index-tracking portfolio optimization. *Expert. Syst. Appl.*, 2021, **183**(1), 1–11.
Lai, Q., Gao, X. and Li, L., A data-driven deep learning approach for options market making. *Quant. Finance*, 2023, **23**(5), 777–797.
Larsen, G.A. and Resnick, B.G., Empirical insights on indexing: How capitalization, stratification and weighting can affect tracking error. *J. Portfolio Manage.*, 1998, **25**(1), 51–60.
Li, Y. and Forsyth, P.A., A data-driven neural network approach to optimal asset allocation for target based defined contribution pension plans. *Insurance: Math. Econ.*, 2019, **86**(1), 189–204.
Ling, A., Sun, J. and Yang, X., Robust tracking error portfolio selection with worst-case downside risk measures. *J. Econ. Dyn. Control*, 2014, **39**(1), 178–207.
McNeil, A.J., Frey, R. and Embrechts, P., *Quantitative Risk Management: Concepts, Techniques, and Tools*, 2015 (Princeton University Press: Princeton, NJ).
Meade, N. and Salkin, G.R., Developing and maintaining an equity index fund. *J. Oper. Res. Soc.*, 1990, **41**(7), 599–607.
Ni, C., Li, Y., Forsyth, P. and Carroll, R., Optimal asset allocation for outperforming a stochastic benchmark target. *Quant. Finance*, 2022, **22**(9), 1595–1626.
Nian, K., Coleman, T.F. and Li, Y., Learning sequential option hedging models from market data. *J. Banking Finance*, 2021, **133**(1), 1–14.
Ouyang, H., Zhang, X. and Yan, H., Index tracking based on deep neural network. *Cogn. Syst. Res.*, 2019, **57**(1), 107–114.
Peng, X., Gong, C. and He, X.D., Reinforcement learning for financial index tracking, 2023. Preprint arXiv:2308.02820.
Soner, H.M. and Touzi, N., Homogenization and asymptotics for small transaction costs. *SIAM J. Control Optim.*, 2013, **51**(4), 2893–2921.
Strub, O. and Baumann, P., Optimal construction and rebalancing of index-tracking portfolios. *Eur. J. Oper. Res.*, 2018, **264**(1), 370–387.
Tsay, R.S., *Analysis of Financial Time Series*, 3rd ed., 2010 (John Wiley & Sons: Hoboken, NJ).
Tu, J., Is regime switching in stock returns important in portfolio decisions? *Manage. Sci.*, 2010, **56**(7), 1198–1215.
van Montfort, K., Visser, E. and van Draat, L.F., Index tracking by means of optimized sampling. *J. Portfolio Manage.*, 2008, **34**(2), 143–152.
Wang, M., Xu, C., Xu, F. and Xue, H., A mixed 0–1 LP for index tracking problem with CVaR risk constraints. *Ann. Oper. Res.*, 2012, **196**(1), 591–609.
Wu, L.-C., Chou, S.-C., Yang, C.-C. and Ong, C.-S., Enhanced index investing based on goal programming. *J. Portfolio Manage.*, 2007, **33**(3), 49–56.
Yao, D.D., Zhang, S. and Zhou, X.Y., Tracking a financial benchmark using a few assets. *Oper. Res.*, 2006, **54**(2), 232–246.
Yi, K., Yang, J., Wang, S., Zhang, Z., Zhang, J., Song, J. and Ren, X., Intelliportfolio: Intelligent portfolio for enhanced index tracking using clustering and LSTM. *Math. Probl. Eng.*, 2022, **2022**(1), 1–15.
Zhang, C., Liang, S., Lyu, F. and Fang, L., Stock-index tracking optimization using auto-encoders. *Front. Phys.*, 2020, **8**(1), 1–15.

Appendix. Estimation and filtering of hidden Markov models

A Q-state HMM model with Gaussian noise is described by the parameter vector $\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{\mu}, \boldsymbol{\sigma})$, where

- $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_Q) \in \mathbb{R}^{1\times Q}$: the initial distribution of the q_t process;
- $\boldsymbol{A} = \{A_{i,j}\} \in \mathbb{R}^{Q\times Q}$: the transition probability matrix of the q_t process;
- $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_Q)^T \in \mathbb{R}^Q$: the collection of mean of r_t in every regime;
- $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_Q)^T \in \mathbb{R}^Q$: the collection of standard deviation of r_t in every regime.

The estimation of HMM models is well studied; see e.g. Kim and Nelson (2017). In our implementation, we use the Baum-Welch algorithm (Baum *et al.* 1970), which is a special EM algorithm. It starts from an initial guess of parameter $\boldsymbol{\theta}$ and converges to a local optimal solution iteratively.

After obtaining an estimate of $\boldsymbol{\theta}$, the filtered probabilities of an observed path $\boldsymbol{r}_{1:T}$ can be obtained iteratively by the forward algorithm. We denote the density function of r_t conditioned on q_t by $b_{q_t}(\cdot)$. Under the Gaussian assumption,

$$b_{q_t}(r_t) = f_{\mathcal{N}}\left(\frac{r_t - \mu_{q_t}}{\sigma_{q_t}}\right),$$

where $f_{\mathcal{N}}(\cdot)$ is the standard normal density function. We define

$$\alpha_i^{\boldsymbol{\theta}}(t) := \mathbb{P}(\boldsymbol{r}_{1:t}, q_t = i \,|\, \boldsymbol{\theta}), \quad i = 1, \ldots, Q.$$

Starting with $\alpha_i^{\boldsymbol{\theta}}(1) = \pi_i b_i(r_1)$, we obtain the recursion,

$$\alpha_i^{\boldsymbol{\theta}}(t+1) = \left(\sum_{j=1}^{Q} \alpha_j^{\boldsymbol{\theta}}(t) A_{j,i}\right) b_i(r_{t+1}), \quad i = 1, \ldots, Q.$$

For each day $t = 1, \ldots, T$, after obtaining $\alpha_i^{\boldsymbol{\theta}}(t)$ for $i = 1, \ldots, Q$, the filtered probability can be obtained by

$$\begin{aligned}\mathbb{P}(q_t = i \,|\, \boldsymbol{r}_{1:t}, \boldsymbol{\theta}) &= \frac{\mathbb{P}(\boldsymbol{r}_{1:t}, q_t = i \,|\, \boldsymbol{\theta})}{\mathbb{P}(\boldsymbol{r}_{1:t} \,|\, \boldsymbol{\theta})} \\ &= \frac{\alpha_i^{\boldsymbol{\theta}}(t)}{\sum_{j=1}^{Q} \alpha_j^{\boldsymbol{\theta}}(t)}, \quad i = 1, \ldots, Q.\end{aligned}$$

In our problem, Q=2. We plot in figure A1 the filtered regimes of the S&P 500 index and the stock AAPL, where the green background indicates the bull market ($\mathbb{P}(q_t = 1 \,|\, \boldsymbol{r}_{1:t}, \boldsymbol{\theta}) \geq 0.5$) and the yellow background indicates the bear market ($\mathbb{P}(q_t = 1 \,|\, \boldsymbol{r}_{1:t}, \boldsymbol{\theta}) < 0.5$). Here, the HMMs are first estimated using the data from the beginning of 2000 to the end of 2016, and then used to filter the regime on the entire return path from the beginning of 2000 to the end of 2022.

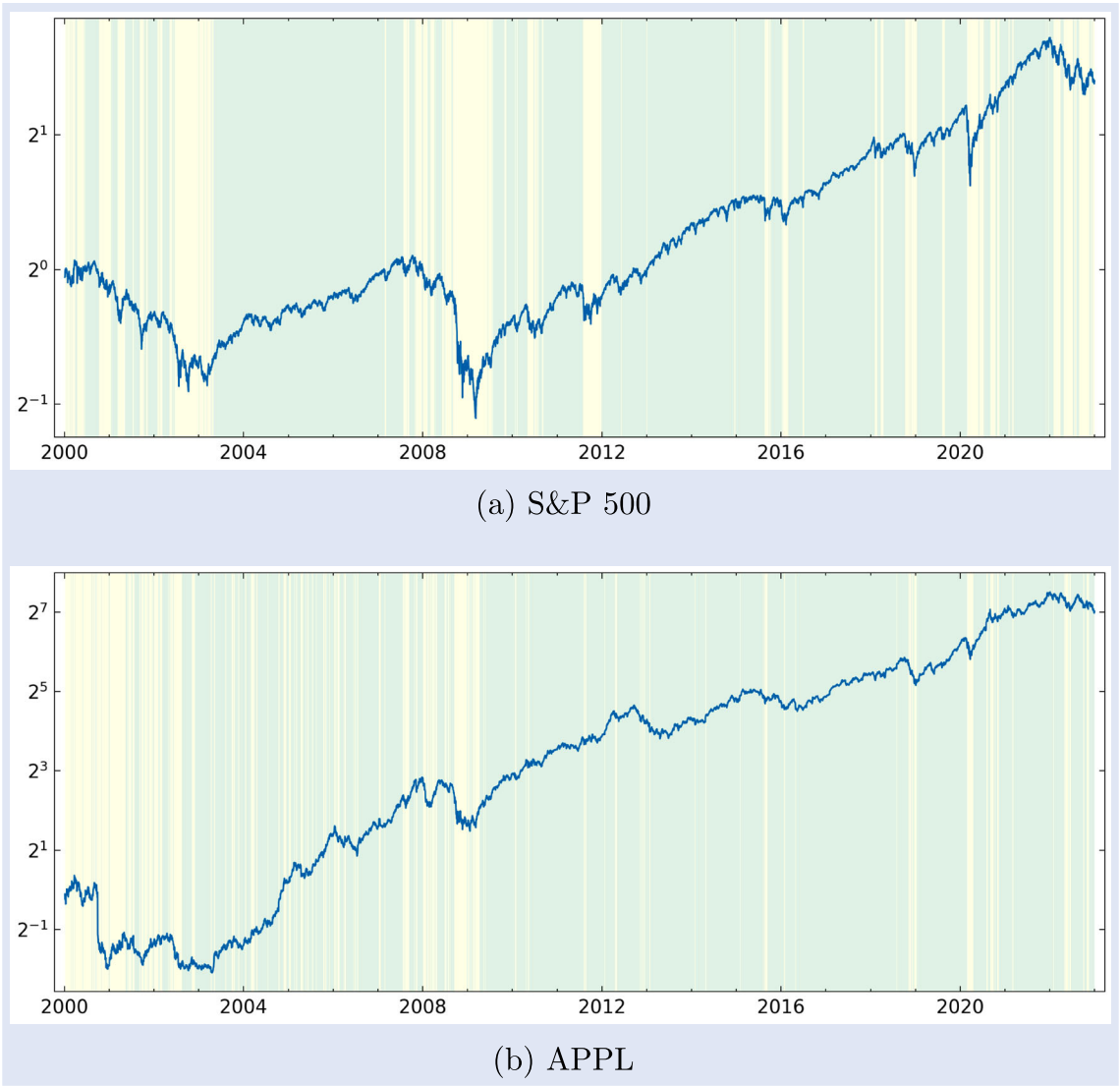


Figure A1. Cumulative wealth log-plots of the S&P 500 index and the stock AAPL with filtered regimes. Areas shaded in green and yellow refer to bull and bear periods, respectively. (a) S&P 500 (b) APPL.