PARTIAL INDEX TRACKING: A META-LEARNING APPROACH

Yongxin Yang
Queen Mary University of London
United Kingdom
yongxin.yang@qmul.ac.uk

Timothy M. Hospedales
University of Edinburgh & Samsung AI Centre, Cambridge
United Kingdom
t.hospedales@ed.ac.uk

ABSTRACT

Partial index tracking aims to cost effectively replicate the performance of a benchmark index by using a small number of assets. It is usually formulated as a regression problem, but solving it subject to real-world constraints is non-trivial. For example, the common ℓ_1 regularised model for sparse regression (i.e., LASSO) is not compatible with those constraints. In this work, we meta-learn a sparse asset selection and weighting strategy that subsequently enables effective partial index tracking by quadratic programming. In particular, we adopt an element-wise ℓ_1 norm for sparse regularisation, and meta-learn the weight for each ℓ_1 term. Rather than meta-learning a fixed set of hyper-parameters, we meta-learn an inductive predictor for them based on market history, which allows generalisation over time, and even across markets. Experiments are conducted on four indices from different countries, and the empirical results demonstrate the superiority of our method over other baselines. The code is released at https://github.com/qmfin/MetaIndexTracker.

1 INTRODUCTION

Index tracking is a popular passive investment strategy. It attempts to construct a portfolio of assets to replicate the performance of a given index (Benidis et al., 2018). The constructed portfolio is called the tracking portfolio, and the chosen index is called the benchmark index. Performance is measured by the tracking error between the benchmark index and the constructed tracking portfolio. Methods of constructing a tracking portfolio can be divided into full- and partial replication approaches (Beasley et al., 2003a; García et al., 2017).

Full replication is simply holding all the assets at the same proportions as the market index. It is the most intuitive index tracking approach and provides perfect tracking performance in a frictionless market. However, it leads to high transaction costs due to a large number of index constituents, frequent rebalancing (e.g., daily rebalance), churn in index members, and illiquid assets (Zheng et al., 2020a; Strub & Baumann, 2018).

In contrast, partial replication selects a subset of assets from the index and rebalances at a lower frequency. This leads to a loss of tracking accuracy but significantly reduces transaction costs. Two fundamental problems need to be addressed in partial replication: asset selection and capital allocation. The former decides which assets should be selected for the tracking portfolio, while the latter assigns capital to those selected assets. Asset selection and capital allocation can be solved independently or jointly through optimisation techniques, depending on the designed algorithms and we will provide a short review in the following section.

In general, partial index tracking optimisation can be written as constrained regression problem, which can be solved by either linear programming (Guastaroba & Speranza, 2012; Guastaroba et al., 2016) or quadratic programming (Sant'Anna et al., 2017; Ruiz-Torrubiano & Suárez, 2009; Scozzari et al., 2013), subject to the choice of loss function. In the real world, partial index tracking weights (solution to the regression problem) are subject to constraints, and the three most commonly used ones include: (i) a long-only: each weight is non-negative; (ii) full capital allocation: all weights must sum to one; (iii) sparsity constraint: only a small set of weights are nonzero (few assets are selected).

The most challenging constraint is (iii), i.e., how to achieve the sparse solution. The ℓ_1 norm applies to many portfolio constructing problems (Brodie et al., 2009), but it has a fatal conflict with long-only and full capital allocation constraints, thus it is not practical for real-world trading. An alternative choice is ℓ_0 norm, but it usually needs heuristic-based solvers such as evolutionary algorithms (Chang et al., 2000; Beasley et al., 2003b; Woodside-Oriakhi et al., 2011). Inspired by (Benidis et al., 2018; Zheng et al., 2020b), we adopt the element-wise ℓ_1 norm for sparse regression. Then the open question is how to choose the optimal hyper-parameters, i.e., the weight for each ℓ_1 norm term.

In this work, we meta-learn these hyper-parameters via bi-level optimisation. The concept of meta-learning has a long-standing history. According to (Hospedales et al., 2021), meta-learning can be understood as *learning to learn*, which refers to the process of improving a learning algorithm over multiple learning episodes. This involves two interacting components: (i) A **base learner**, in which an *inner* algorithm solves a task defined by dataset and objective; (ii) a **meta-learner**, where an *outer* algorithm updates the inner algorithm such that the model it learns improve an outer objective. Meta-learning approaches have earned popularity in many areas including few-shot learning (Ravi & Larochelle, 2017) and imitation learning (Finn et al., 2017b).

In our setting, the choice of sparse regularisation hyper-parameters corresponds to a strategy for selecting which assets should be included in the tracking portfolio. However, these hyper-parameters cannot be directly (meta) learned because index constituents change over time. We therefore lift the problem from one of hyper-parameter estimation to one of predicting hyper-parameters from historical market data, and meta-learn this prediction function. To calculate the gradient of this hyper-parameter generator, we need to differentiate through the inner loop optimisation process, where we build a differentiable quadratic programming (QP) solver following (Amos & Kolter, 2017). We evaluate the proposed approach with four indices and compare our method with several baselines. These indices are from both emerging and developed markets, and have different numbers of constituents. Our approach outperforms other baselines by a significant margin (40% tracking error reduced on average).

To summarise, the main contribution in this paper is to provide the first meta-learning perspective on the partial index tracking problem. Prior studies all hand-crafted various heuristics for asset selection. We highlight for the first time that one can meta-learn an asset selection strategy by directly minimising partial index tracking error over a large set of historical windows. We provide an efficient and effective algorithm to do so that substantially improves on state of the art in practice.

2 Related Work

2.1 Index Tracking

Asset selection and capital allocation are two fundamental problems for partial index tracking. The former concerns which assets should be included in the tracking portfolio, while the latter aims to optimally allocate capital among the chosen assets to minimise the tracking error. Previous studies can be broadly clustered into two groups according to whether these two problems are solved independently or jointly.

The first group of methods address the selection and allocation issues in two stages. First, an asset selection method is proposed and a subset of assets is selected. Second, an optimisation algorithm allocates capital among the chosen subset of assets, usually by regression. The main differences in the literature are the various and ad-hoc ways of selecting the assets. For example, some works employed clustering, including hierarchical clustering (Focardi & Fabozzi, 2004; Dose & Cincotti, 2005) and market graph clustering (Hong et al., 2021), to select assets based on their return performance similarities. Other works selected assets based on different factors, such as co-integration and factor replication ability (Alexander & Dimitriu, 2005; Dunis & Ho, 2005; Corielli & Marcellino, 2006). Besides, a deep neural network was employed to identify assets whose original returns and reconstructed returns are the most similar based on their Euclidean distance (Ouyang et al., 2019). However, for an optimal solution for partial index tracking, both asset selection and capital allocation should be tackled jointly because whether a selection of assets is promising or not depends on the actual allocations (Krink et al., 2009; Shu et al., 2020).

The second group of methods unifies asset selection and capital allocation by adding a sparsity constraint on the portfolio weights so that the two issues can be optimised jointly. A natural way is adding an ℓ_0 norm to the objective function to derive a sparse portfolio. However, imposing the ℓ_0 constraint makes the regularised regression problem NP-hard and often requires heuristic solvers, such as Tabu search (García et al., 2017), genetic algorithms (Ni & Wang, 2013; Li et al., 2011), and simulated annealing (Chang et al., 2000; Woodside-Oriakhi et al., 2011). However, these heuristic algorithms have a fatal disadvantage: they are not guaranteed to find the optimal solution, and the search space grows super-linearly in many situations (Zheng et al., 2020a). An alternative is to use the fractional ℓ_p norm (Fastrich et al., 2014; Xu et al., 2015) where $0 < p < 1$. However, it is a non-convex relaxation of the ℓ_0 norm, increasing the difficulty of solving the optimisation problem. In addition, the popular approximation ℓ_1 norm penalty is used to derive a sparse and stable portfolio in a Markowitz mean-variance framework (Brodie et al., 2009). Unfortunately, the ℓ_1 norm does not work in partial index tracking since we require the weights to satisfy sum-to-one and long-only constraints.

Another alternative is the element-wise ℓ_1 norm. For example, (Benidis et al., 2018) proposed a ℓ_0 norm approximation function to replace the ℓ_1 norm that iteratively leads to a element-wise ℓ_1 norm minimisation algorithm, and in (Song et al., 2015; Benidis et al., 2016), this element-wise ℓ_1 norm is used for sparse eigenvector extraction. (Zheng et al., 2020b) introduced an element-wise ℓ_1 realising for both sparsity and diversity, with the hyper-parameter determined by spectral clustering. Inspired by (Benidis et al., 2018; Zheng et al., 2020b), we choose the element-wise ℓ_1 norm to realise sparsity. However, we directly learn the weights by minimising tracking error; and to achieve this using a changing set of historical constituents, we lift the problem to one of learning a weight predictor.

2.2 Meta Learning

Meta-learning, or learning to learn, aims to improve the learning algorithm itself. The concept of meta-learning has a long history (Schmidhuber, 1995; Thrun & Pratt, 2012); and more recently it has achieved outstanding performance in many fields, including few-shot learning (Ravi & Larochelle, 2017), imitation learning (Finn et al., 2017b), visual question answering (Teney & van den Hengel, 2018), and neural architecture search (Real et al., 2019; Zoph & Le, 2017). However, there are relatively few studies applying meta-learning in finance, and our work is the first to adapt meta-learning for index tracking.

There are different frameworks to formalise meta-learning, among which we adopt bi-level optimisation (BLO). BLO originated from economic game theory before its introduction to the optimisation community, and it has received significant attention in the meta learning community recently (Franceschi et al., 2018; Zügner & Günnemann, 2018; Ji et al., 2020). For example, the famous model-agnostic meta-learning is a BLO approach to meta-learning initial conditions (Finn et al., 2017a; Rajeswaran et al., 2019). We adopt a BLO approach to meta-learning for training an asset selection and weighting strategy in the form of a sparse regularisation weight generator.

As we frame the optimisation problem as BLO, the choice of regularisation hyper-parameters determines the sparse solution on meta-train data, and thus the tracking error on meta-test data. Therefore, to optimise with respect to hyper-parameters, we need to differentiate through the learning process. Following (Amos & Kolter, 2017), we formulate the index tracking problem as a QP problem as in (Sant'Anna et al., 2017; Ruiz-Torrubiano & Suárez, 2009; Scozzari et al., 2013), and then calculate the partial derivative of hyper-parameter w.r.t. the loss on meta-test data.

3 Methodology

3.1 Index tracking and meta task

Index tracking, in its simplest form, is a linear regression problem,

$$\min_{w} \ \|Xw - y\|_2^2 \tag{1}$$

where $X \in \mathbb{R}^{D\times N}$ is the return of assets and $y \in \mathbb{R}^D$ is the target index. D is the number of timesteps (e.g., $D = 750$ trading days in three consecutive years), and N is the number of assets (e.g., $N = 500$ assets). $w \in \mathbb{R}^N$ is the weight of each asset to hold to approximate the index y.

In practice, there are two constraints on w: (i) *long only*, which means $w_i \geq 0, \forall i$, (ii) utilise *all* of the capital, which means $\sum_{i=1}^N w_i = 1$. Therefore, the objective function becomes,

$$\min_{w\geq \mathbf{0},\ \sum_i w_i=1} \quad \|Xw - y\|_2^2 \tag{2}$$

Eq. 2 is known as a non-negative regression problem with sum-to-one constraint, which can be efficiently solved by the general QP solvers.

For partial index tracking, sparsity is the crucial property because it reduces transaction costs significantly compared with full index tracking. As the ℓ_0 norm often needs heuristic methods, the ℓ_1 norm (known as LASSO) is more common for realising sparsity. However, under the long-only and sum-to-one constraints, the ℓ_1 norm constantly equals one – a fatal incompatibility. To this end, we place an element-wise ℓ_1 norm for constrained regression Eq. 2 following (Benidis et al., 2018; Zheng et al., 2020b), because the objective is still of a QP form with the element-wise ℓ_1 norm.

$$\min_{w\geq \mathbf{0},\ \sum_i w_i=1} \quad \|Xw - y\|_2^2 + \lambda^T w \tag{3}$$

The hyper-parameter λ implicitly governs asset selection, as it determines which elements of w are non-zero. In contrast to (Zheng et al., 2020b) setting λ by some handcrafted rules, we employ a meta-learning framework for hyper-parameter λ optimisation (Balaji et al., 2018; Lorraine et al., 2020) using a large set of historical windows, which correspond to learning *episodes* in meta-learning terminology (Hospedales et al., 2021; Finn et al., 2017a; Snell et al., 2017).

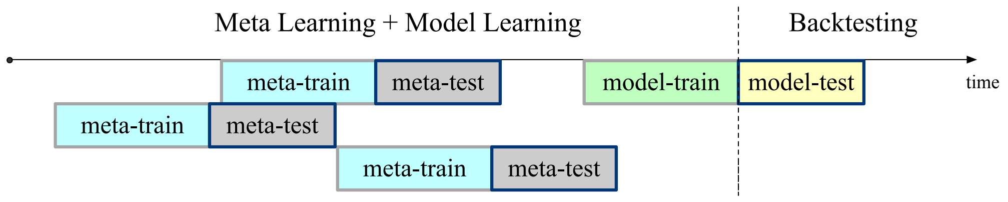


Figure 1: Learning episode design. For a given rebalance day of real testing (dotted line), contiguous support (train, blue) and query (test, grey) windows are sampled from historical data and used to train hyperparameters. These are then used for real training (green) before testing (yellow). The whole process is repeated in a sliding window for testing. The query/support episode length used for learning matches that of the final evaluation.

The way we construct episodes strictly follows the actual trading practice. As an illustrative example, we may use $3Y$ (three years) data for training the constrained regression model (Eq. 3), use the learned weights w to guide our buy-and-hold operation in the next $3M$ (three months), and after that we will rebalance with the same strategy. These two splits are also known as *support* and *query* sets in meta-learning community (Hospedales et al., 2021; Finn et al., 2017a; Snell et al., 2017). To create the set of learning episodes for this $(3Y, 3M)$ pattern, we will take the following steps: (i) On a rebalance day, we collect a sufficiently long, e.g., ten consecutive years data as our full training data. (ii) We can generate arbitrarily many $(3Y, 3M)$ episode pairs, by sampling a random $3Y + 3M$ period over ten-year full training data, to form the set of meta-train tasks, where each task has $3Y$ as support data and $3M$ as query data.

With this set of learning tasks, we can learn the hyper-parameter λ such that the weight produced by solving Eq. 3 on the support data will have small tracking error on its paired query data. Finally, the learned λ is applied for solving Eq. 3 on the last $3Y$ data, and the solution is used for the actual trading (buy-and-hold), since the last day of ten year corresponds to the rebalance day. The flow of task-sampling for meta-training and deployment (meta-testing) is illustrated in Fig. 1.

Mathematically, if we denote (X, y) and (X', y') as the support and query data for a certain episode, the corresponding meta objective function can be written as a bi-level optimisation problem,

$$\begin{aligned} &\min_{\lambda} \ \|X' \cdot w^*(\lambda, X, y) - y'\|_2^2 \\ \text{subject to: } w^*(\lambda, X, y) \ &= \underset{w \geq \mathbf{0}, \sum_i w_i = 1}{\operatorname{argmin}} \ \|Xw - y\|_2^2 + \lambda^T w \end{aligned} \tag{4}$$

The lower-level optimisation task can be tackled by an off-the-shelf quadratic programming solver, i.e., $w^*(\lambda, X, y) = \text{QP\_Solver}(\lambda, X, y)$. In practice, we use iterative optimisation: for every iteration, we sample one (or few) episodes, calculate the gradient of the upper-level optimisation task's objective function w.r.t. λ, and update λ by gradient descent. By doing so, we will encounter two challenges: (i) To calculate the gradient of Eq. 4 w.r.t. λ (this is sometimes called meta gradient), we need to differentiate through the QP solver. In Sec. 3.2, we present a solution by implicit function theorem so that we can avoid going through the internal process of QP solver. (ii) For a sufficiently long period, e.g., 10-year in the above example, the entities in λ, X and y are not consistent, as the benchmark constituents change over time. Crucially, this means that λ should vary with any given temporal episode (Fig. 1). Therefore, we lift λ from a vector of parameters to the output of a predictive model which takes meta-train data as input, i.e., $f_\theta(X, y) \to \lambda$. We will discuss the design of f_θ in Sec. 3.3.

In summary, the full objective function for learning a meta model f_θ can be written as

$$\min_{\theta} \frac{1}{|\mathcal{T}|} \sum_{((X,y),(X',y')) \sim \mathcal{T}} \|X' \cdot \text{QP\_solver}(f_\theta(X, y), X, y) - y'\|_2^2 \tag{5}$$

3.2 Differentiating through the QP Solver

The QP form of Eq. 3 is,

$$\min_{w} \frac{1}{2} w^T P w + q^T w$$
$$\text{subject to: } Gw \leq h \text{ and } Aw = b$$

where $P = 2X^T X$, $q = \lambda - 2X^T y$, $G = -I$, $h = \mathbf{0}$, $A = \mathbf{1}^T$, $b = 1$. Assuming that the optimal solution w^* is produced by an offline QP solver,

$$w^* = \text{QP\_Solver}(P, q, G, h, A, b) \tag{6}$$

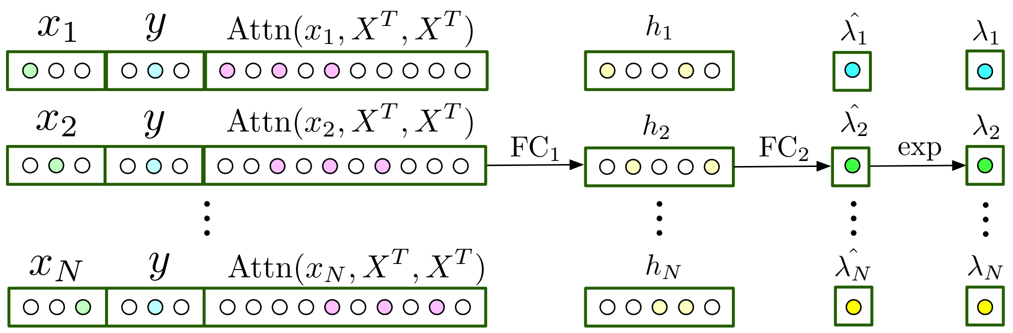


Figure 2: Design of asset selection strategy function $\lambda = f_\theta(X, y)$. Learnable θ comprises parameters in fully connected layers FC_1, FC_2, and Attention module *Attn*.

and we need to compute the gradient of w^* w.r.t. λ, i.e., $\frac{\partial w^*}{\partial \lambda}$. Following (Amos & Kolter, 2017; Lorraine et al., 2020), we derive the gradient via implicit function theorem. Note that λ appears in the q term only, thus we apply the chain rule,

$$\frac{\partial w^*}{\partial \lambda} = \frac{\partial w^*}{\partial q}\frac{\partial q}{\partial \lambda} \tag{7}$$

The second term $\frac{\partial q}{\partial \lambda} = I$. If the QP problem is solved by some generalised Lagrange multiplier based methods, the *full* output of QP_Solver will be $s = [w^*, \alpha_1^*, \alpha_2^*]$ where $\alpha_1^* \in \mathbb{R}^1$ is the Lagrange multiplier for the equality constraint and $\alpha_2^* \in \mathbb{R}^N_{\geq 0}$ is the Lagrange multiplier for the inequality constraint.

If we have the gradient $g = \frac{\partial s}{\partial q} \in \mathbb{R}^{(2N+1)\times N}$, then the sought gradient is simply the first N rows in g. Meanwhile $\frac{\partial s}{\partial q}$ itself can be obtained by *implicit function theorem* (IFT)

$$\frac{\partial s}{\partial q} = -(\frac{\partial f}{\partial s})^{-1}\frac{\partial f}{\partial q} \tag{8}$$

Here f, q, and s must satisfy $f(q, s(q)) = \mathbf{0}$, and $f(q, s(q)) = \mathbf{0}$ can be obtained from KKT conditions

$$f(q, s(q)) = \begin{bmatrix} Pw^* + q + A^T\alpha_1^* + G^T\alpha_2^* \\ Aw^* - b \\ \alpha_2^* \odot (Gw^* - h) \end{bmatrix} = \begin{bmatrix} Pw^* + q + \mathbf{1}\alpha_1^* - \alpha_2^* \\ \mathbf{1}^T w^* - 1 \\ -\alpha_2^* \odot w^* \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ 0 \\ \mathbf{0} \end{bmatrix} \tag{9}$$

Thus, the corresponding derivatives are

$$\frac{\partial f}{\partial s} = \begin{bmatrix} P & \mathbf{1} & -I \\ \mathbf{1}^T & 0 & \mathbf{0}^T \\ -\operatorname{Diag}(\alpha_2^*) & \mathbf{0} & -\operatorname{Diag}(w^*) \end{bmatrix} \to K \tag{10}$$

and

$$\frac{\partial f}{\partial q} = \begin{bmatrix} I \\ \mathbf{0}^T \\ \mathbf{0}\dots\mathbf{0} \end{bmatrix} \to v \tag{11}$$

To avoid the computation of inverse matrix K^{-1}, we connect $K^{-1}v$ with the existing gradient passed from back-propagation, i.e., $\frac{\partial L}{\partial w^*}$ where L is the final (meta-test) loss. First, we pad $\frac{\partial L}{\partial w^*}$ with zeros as

$$\frac{\partial L}{\partial s} = [\frac{\partial L}{\partial w^*}, 0, \mathbf{0}] \tag{12}$$

Then we can compute $\frac{\partial L}{\partial \lambda}$ by

$$\frac{\partial L}{\partial \lambda} = \frac{\partial L}{\partial s} \cdot \frac{\partial s}{\partial \lambda} = -[\frac{\partial L}{\partial w^*}, 0, \mathbf{0}] \cdot K^{-1} \cdot v \tag{13}$$

The dot product of the first two terms is the solution to a system of linear equations where the matrix is K^T and the column vector is $[\frac{\partial L}{\partial w^*}, 0, \mathbf{0}]^T$. Finally, the dot product of the obtained solution and v is simply the first N entities from the obtained solution.

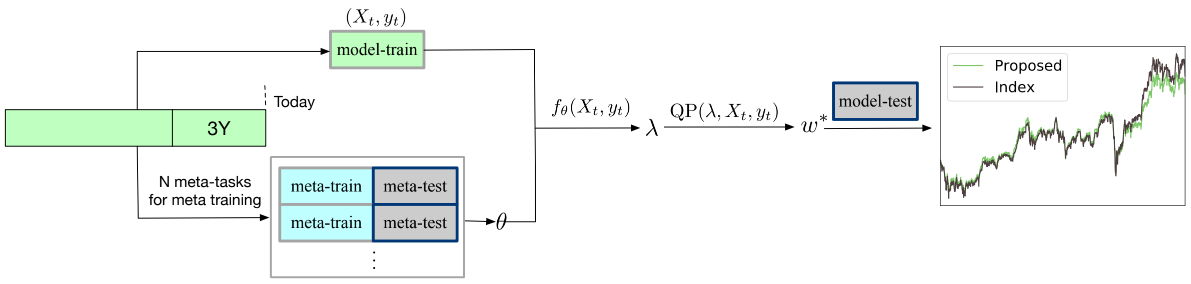


Figure 3: Full pipeline for one rebalancing. (1) Meta-learning process produces θ, which provides regulariser $\lambda = f_\theta(\cdot)$ for the QP. (2) QP produces asset weights w^*, which are bought and compared to the index.

3.3 Designing sparsity regulariser weight predictor f_θ

Recall that regularisation weights λ correspond to the asset selection strategy that we wish to learn in Eq. 4, since they determine which assets have non-zero weights after the QP. However, the ideal asset selection strategy varies over time (choice of support set (X, y)). More fundamentally, since the benchmark constituents change over time, it cannot literally be represented by a vector of constant weights. To learn an asset selection strategy over an extended time-period, we therefore lift the problem to one of predicting weights λ from historical data, i.e., $\lambda = f_\theta(X, Y)$ and learn θ instead.

To determine which assets should be included in the tracking portfolio, several factors are important. (i) Obviously, the information of asset itself should be involved, e.g., the asset's return. (ii) Since we aim to minimise the tracking error with the benchmark index, the benchmark index information, more generally, the whole market information, should be considered. (iii) The tracking portfolio comprises multiple assets, and decision to include one asset should also consider other assets, and this suggests that the asset-to-asset correlation needs to be considered as well.

We present a way to construct these factors using only return data, i.e., X and y. From the view of the ith asset, (i) is simply the ith column of X, denoted as x_i; (ii) is the return of benchmark index, i.e., y; (iii) can be a similarity measured by x_i against all assets' returns X. We simply concatenate three factors and form a long vector for each asset as the input for f_θ.

To further improve (iii) by introducing more learnable parameters, and more importantly, to align it with (i) and (ii) which are both return data, we employ a self-attention mechanism (Vaswani et al., 2017), where key is x_i, query is X^T, and value is X^T. Finally, we can write the designed f_θ as

$$\begin{aligned} u_i &= [x_i, y, \text{Attn}(x_i, X^T, X^T)] \\ \lambda_i &= \exp(\text{FC}_2(\text{FC}_1(u_i))) \end{aligned} \tag{14}$$

where the learnable parameters θ include the attention layer and the following fully-connected (FC) layer parameters (see Fig. 2). To ensure the non-negativity on λ, we use the exponential function as the activation function of the final layer. To demonstrate the effectiveness of our design, we perform ablation studies over different combinations of (i), (ii), and (iii) in the experiment section.

This architecture could be further improved by introducing exogenous factors such as Macro and asset's meta data in the input to f_θ, but for ease of direct comparison we leave this to future work.

The full pipeline can be found in Fig. 3, and the PyTorch-style pseudo-code can be found in App. A.1.

4 Experiments

4.1 Datasets and Experimental Design

Datasets To rigorously evaluate the performance of the proposed method, we conduct extensive backtesting experiments over four indices from both emerging and developed countries: SENSEX 30 (India), CAC 40 (France), NIKKEI 225 (Japan), and S&P 500 (USA). The date range is from 2005-11-30 to 2021-11-30. Pricing data is obtained from the Center for Research in Security Prices (CRSP), which is known to be high-quality research data. We use the daily closing prices adjusted for dividends and other corporate actions (e.g., splits and mergers).

Table 1: Tracking error (↓). TE and TEV are multiplied by 10^4. Bold numbers are best for each column.

	SENSEX 30		CAC 40		NIKKEI 225		S&P 500	
	TE$_{\pm\text{std.}}$	TEV$_{\pm\text{std.}}$	TE$_{\pm\text{std.}}$	TEV$_{\pm\text{std.}}$	TE$_{\pm\text{std.}}$	TEV$_{\pm\text{std.}}$	TE$_{\pm\text{std.}}$	TEV$_{\pm\text{std.}}$
Proposed	**0.66**$_{\pm 0.04}$	**0.07**$_{\pm 0.01}$	**0.52**$_{\pm 0.05}$	**0.04**$_{\pm 0.01}$	**0.49**$_{\pm 0.03}$	**0.04**$_{\pm 0.01}$	**0.80**$_{\pm 0.08}$	**0.09**$_{\pm 0.04}$
SNN (Zheng et al., 2020a)	1.39$_{\pm 0.10}$	0.31$_{\pm 0.04}$	1.11$_{\pm 0.10}$	0.19$_{\pm 0.04}$	1.33$_{\pm 0.04}$	0.28$_{\pm 0.02}$	1.44$_{\pm 0.19}$	0.32$_{\pm 0.09}$
FN1/8 (Kabán, 2013)	1.46$_{\pm 0.09}$	0.33$_{\pm 0.04}$	1.06$_{\pm 0.10}$	0.18$_{\pm 0.03}$	1.09$_{\pm 0.02}$	0.19$_{\pm 0.01}$	1.09$_{\pm 0.03}$	0.18$_{\pm 0.01}$
SPC (Zheng et al., 2020b)	1.93	0.59	1.23	0.24	1.49	0.35	1.19	0.21
NCO (DeMiguel et al., 2009)	1.06	0.17	0.77	0.09	1.19	0.22	1.10	0.18
NCO_ℓ_2 (DeMiguel et al., 2009)	1.04	0.17	0.74	0.09	1.07	0.18	1.08	0.17

Training and Testing Split The backtesting period starts from 2015-12-01 and ends on 2021-11-30, and is carried out in a sliding window fashion. Following the common practice in industry, we choose three consecutive years[1] ($3Y$) for training and the consequent three months ($3M$) for testing. $3M$ means that we adopt a quarterly rebalancing strategy.

We adopt the proposed meta-learning scheme optimising the hyper-parameter generating function f_θ. For example, on each trading day, e.g., on 1st July 2016, we collect the total training data for ten years ($10Y$), from 1st July 2006 to 1st July 2016. From this period, we randomly generate $3Y + 3M$ windows as episodes for meta-learning: $3Y$ as the support and $3M$ as the query data. After θ is optimised, we use the last $3Y$ data (the same length as meta-train) as the model-train data, and get the weight vector w^* given f_θ. We then buy corresponding the assets and hold them until the last trading day 2021-11-30. The full pipeline for one rebalancing step is illustrated in Fig. 3. Different rebalancing frequencies and total training lengths are considered in the ablation studies in Sec. 4.3.

A transaction fee of 0.02% per trade is taken into account to simulate real trading. Backtesting is performed with the help of the python module Backtrader, which accounts for all necessary trading details, including transaction costs, strategies, and indicators.

Baseline Methods To demonstrate the efficacy of our method, we compare it with five baseline methods: (i) **SNN** (Stochastic Neural Network) (Zheng et al., 2020a), a reparametrisation based method for partial index tracking with cardinality constraint by using stochastic neural network. (ii) **FN1/8**, the fractional norm ℓ_p norm where $p = \frac{1}{8}$ regularised index tracking method in (Fastrich et al., 2014; Xu et al., 2015). Following (Kabán, 2013), we adopt a smooth approximation for the ℓ_p norm: $\|w\|_q^q = \sum_{i=1}^N |w_i|^q \approx \sum_{i=1}^N (w_i^2 + \gamma)^{q/2}$. (iii) **SPC** (Spectral Clustering) (Zheng et al., 2020b), an index tracking method that considers both sparsity and diversity jointly. (iv) **NCO** (Naive Constrained Regression). As the Eq. 2, this is a non-negative regression with the sum-to-one constraint. (v) **NCO_ℓ_2** (Naive Constrained Regression with ℓ_2-norm). NCO with an ℓ_2-norm added to Eq. 2, as suggested by (DeMiguel et al., 2009). SNN, FN1/8, and our method are non-deterministic due to non-linearity and/or network initialisation, thus we run these methods 10 times repeatedly with different random seeds and present the consistency over performance by the standard deviation.

Performance Measures We choose the following metrics to evaluate the trackers produced by different methods. (i) **TE** (Tracking Error). The direct measure for tracking accuracy is the root mean squared error between the index and partial replication portfolio performance, which is defined as: $\frac{1}{T}\sum_{t=1}^T (\hat{y}_t - y_t)^2$, where $\hat{y}_t$ and y_t are the returns of the tracking portfolio and benchmark index respectively at time t. (ii) **TEV** (Tracking Error Variance). We also report the variance between tracking portfolio and index returns, which is defined as $\text{Var}(\hat{y}_t - y_t)$. We aim to track the benchmark index accurately and stably, so we prefer lower numbers for TE and TEV. (iii) **Number of Stocks**. As we solve the partial index tracking problem, we aim to select a subset of all stocks. This is measured by the number of stocks with the weight larger than 10^{-6} on each rebalance day.

Apart from the above measures, we report two other measures for general portfolio optimisation: (i) **VO**: the volatility of return measures the quantitative risk for the tracking portfolio. (ii) **SR**: the Sharpe ratio (Sharpe, 1994) measures the risk-adjusted return. From the perspective of portfolio performance, we prefer lower and higher numbers for these two measures. However, our primary goal is to track the benchmark index accurately. Therefore, they are of secondary importance to tracking accuracy. We present the results for the above measures in Sec. 4.2.

4.2 Result Analysis

Our main result is Tab. 1, which compares all methods' index tracking performance over a six year backtesting period across several indices in terms of TE and TEV. Our meta-learning approach achieves the most accurate tracking

[1]The length for model training is actually another meta-learnable hyper-parameter, and we demonstrate this in App. A.2

Table 2: General portfolio performance report in SR (↑,%) and VO (↓,%). The bold number represents the best result, and underscore represents the closest result to the benchmark index.

	SENSEX 30		CAC 40		NIKKEI 225		S&P 500	
	SR$_{\pm\text{std.}}$	VO$_{\pm\text{std.}}$	SR$_{\pm\text{std.}}$	VO$_{\pm\text{std.}}$	SR$_{\pm\text{std.}}$	VO$_{\pm\text{std.}}$	SR$_{\pm\text{std.}}$	VO$_{\pm\text{std.}}$
Proposed	98.63$_{\pm 4.33}$	**16.28**$_{\pm 0.23}$	<u>**49.31**</u>$_{\pm 2.07}$	**17.54**$_{\pm 0.36}$	42.05$_{\pm 4.38}$	**18.43**$_{\pm 0.28}$	<u>**77.46**</u>$_{\pm 7.95}$	**16.28**$_{\pm 1.49}$
SNN (Zheng et al., 2020a)	83.72$_{\pm 6.37}$	16.75$_{\pm 0.43}$	36.02$_{\pm 11.83}$	19.00$_{\pm 0.60}$	25.80$_{\pm 6.57}$	19.96$_{\pm 0.53}$	64.87$_{\pm 8.25}$	19.20$_{\pm 1.01}$
FN1/8 (Kabán, 2013)	86.85$_{\pm 17.50}$	16.88$_{\pm 0.65}$	43.62$_{\pm 9.28}$	19.23$_{\pm 1.03}$	29.29$_{\pm 3.29}$	19.71$_{\pm 0.12}$	73.16$_{\pm 3.78}$	18.58$_{\pm 0.19}$
SPC (Zheng et al., 2020b)	**106.27**	17.73	47.06	18.12	<u>**46.94**</u>	19.99	68.67	18.59
NCO (DeMiguel et al., 2009)	83.66	<u>17.00</u>	48.23	<u>18.89</u>	32.34	20.33	66.28	18.56
NCO_ℓ_2 (DeMiguel et al., 2009)	<u>88.70</u>	16.72	44.66	19.54	30.17	<u>19.53</u>	68.94	<u>18.53</u>
Index	87.72	17.40	51.75	18.84	46.37	19.57	90.13	18.37

Table 3: Further analysis: Ablation, sensitivity, and transferability. All results are on NIKKEI255 index. Proposed refers to our standard f_θ design, 3 monthly re-balancing, 10 year train data and within-index training.

		f_θ design		Rebalance Freq.			Train Data				Source Index for Transfer		
	Proposed	-Corr	-Index	1M	6M	12M	$8Y$	$6Y$	$5Y$	$4Y$	S30→	CAC40→	S&P500→
TE (↓)	**0.49**	0.53	0.51	0.53	0.52	0.53	0.50	0.52	0.53	0.53	0.58	0.55	**0.37**
TEV (↓)	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.06	0.05	0.02

performance compared with competitors in all indices. The tracking errors for the remaining baselines are significantly larger than our approach, which verifies the effectiveness for our framework. As we aim to track the benchmark index by using a sparse tracking portfolio, we also analyse the number of stocks in App. A.3. Besides, equity curves for different methods are attached in App. A.4.

In terms of general portfolio performance, i.e., Sharpe ratio and volatility measures, our method also provides generally superior performance to the alternatives in Tab. 2. Although the goal is to track the index accurately, our approach also achieves the best general portfolio performance in most cases.

What is learned? We also study possible connections between model generated λ and some heuristics. A standard strategy is to select assets that are close to the target index in aspects such as return and price (Focardi & Fabozzi, 2004; Dose & Cincotti, 2005). Thus we explore the relationship between λ and the correlation $\mathrm{Corr}(x_i, y)$ between returns of each asset and the benchmark index. Fig. 4 shows that λ is negatively correlated with the asset's similarity to the benchmark index. Since smaller λ means larger non-zero weight, negative correlation suggests that f_θ prefers assets with returns close to the benchmark return.

We further explore the relationship between λ and market capitalisation Cap_i for each selected asset. This is inspired by (Zheng et al., 2020a) as they sorted all stocks according to their market capitalisation, and then select the top K stocks to realise a sparse solution. Consequently, we expect a negative correlation between λ and Cap_i, as the latter tends to help select assets with larger market capitalisation (Zheng et al., 2020a). However, Fig. 4 does not reflect this negative correlation. The reason behind may be that solely selecting the tracking portfolio according to market capitalisation is not enough to track the benchmark index accurately. This limitation of (Zheng et al., 2020a) may explain our better tracking performance in Tab. 1.

4.3 Ablation Studies and Further Analysis

Ablations and Sensitivity Studies We perform ablation studies for different aspects of model design on NIKKEI 225. First, we evaluation different designs for f_θ. In Sec. 3.3, we remark that f_θ requires three cues: the asset itself, the benchmark index, and correlation between assets. To see the contribution of each factor, we implement two simplified versions: (i) assets and benchmark index: $u_i = [x_i, y]$; (ii) assets and correlation between assets, $u_i = [x_i, \mathrm{Attn}(x_i, X^T, X^T)]$. The results in Tab. 3 show the value of our full $f_\theta(X, y)$ design, as considering only benchmark index or relationship between assets alone leads to the degraded performance.

Second, while quarterly rebalancing is common practice, we also evaluate different rebalancing frequency: monthly ($1M$), half-yearly ($6M$), and yearly ($12M$). Tab. 3 suggests that the choice of rebalancing frequency has a moderately positive effect. Third, we set the length for total training data (from which we generate meta-train and meta-test pairs) to be ten years ($10Y$) in our main experimetns in Sec. 3.1. Now we compare four different training length settings between eight ($8Y$) and four years ($4Y$). The longer total training period brings a slightly better performance, and we can see in Tab. 3.

Transferability Analysis We learned the predictor $\lambda = f_\theta(\cdot)$ rather than a fixed vector λ to support data-dependent asset selection strategies and evolving index constituents. This means that an asset selection strategy trained on one

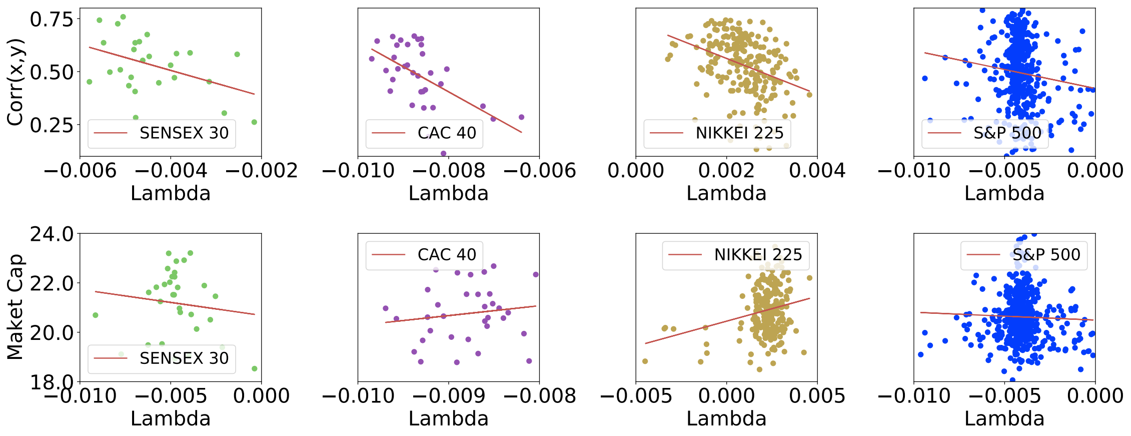


Figure 4: Top plot reflects relationship between $\lambda(x_i, y)$ and $\text{Corr}(x_i, y)$. Negative correlation indicates preference for assets with returns close to the benchmark return. Bottom plot reflects relationship between $\lambda(x_i, y)$ and $\text{Capitalisation}(x_i)$, and the correlation is weaker in this case.

index can be applied to other indices. We consider the f_θ function trained on SENSEX 30, CAC 40, and S&P 500, and apply them to NIKKEI 225. Tab. 3 implies that a model trained on a large index (i.e., S&P 500) generalises better than models trained on smaller indices. Overall this echoes the success of large data pre-training in broader machine learning.

5 Conclusion

When index-tracking is formulated as a constrained regression with an element-wise ℓ_1 norm term, the element-wise weighting corresponds to an asset selection strategy. We provide the first meta-learning perspective on this problem by learning this asset selection strategy that minimises historical tracking error, rather than hand-crafting a heuristic as in prior approaches. To deal with evolving market constituents over time, we further replaced a specific weight vector with a learned weight predictor. Comprehensive backtesting experiments showed our proposed method exhibits superior performance compared to many baselines on four indices from different markets. We hope that this success inspires more interaction between the meta-learning community and financial AI community.

For the future work, we will explore other use cases for this framework, for example, applying it to the portfolio optimisation problems such as Markowitz's mean-variance portfolio (Markowitz, 1952) and minimum variance portfolio (Jagannathan & Ma, 2003). Furthermore, we will investigate other kinds of meta knowledge, such as meta-learning a feature extractor (Snell et al., 2017) for time-series data.

Disclaimer: All authors are faculty. Neither graduate students nor small animals were hurt while producing this paper.

References

Carol Alexander and Anca Dimitriu. Indexing and statistical arbitrage. *The Journal of Portfolio Management*, 31(2): 50–63, 2005.

Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *ICML*, 2017.

Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. Metareg: Towards domain generalization using meta-regularization. In *NeurIPS*, 2018.

John E Beasley, Nigel Meade, and T-J Chang. An evolutionary heuristic for the index tracking problem. *European Journal of Operational Research*, 148(3):621–643, 2003a.

John E Beasley, Nigel Meade, and T-J Chang. An evolutionary heuristic for the index tracking problem. *European Journal of Operational Research*, 148(3):621–643, 2003b.

K. Benidis, Y. Feng, and P. Palomar D. Sparse portfolios for high-dimensional financial index tracking. *IEEE Transactions on Signal Processing*, 66(1):155–170, 2018.

Konstantinos Benidis, Ying Sun, Prabhu Babu, and Daniel P Palomar. Orthogonal sparse pca and covariance estimation via procrustes reformulation. *IEEE Transactions on Signal Processing*, 64(23):6211–6226, 2016.

Joshua Brodie, Ingrid Daubechies, Christine De Mol, Domenico Giannone, and Ignace Loris. Sparse and stable markowitz portfolios. *Proceedings of the National Academy of Sciences*, 106(30):12267–12272, 2009.

T-J Chang, Nigel Meade, John E Beasley, and Yazid M Sharaiha. Heuristics for cardinality constrained portfolio optimisation. *Computers & Operations Research*, 27(13):1271–1302, 2000.

Francesco Corielli and Massimiliano Marcellino. Factor based index tracking. *Journal of Banking & Finance*, 30(8): 2215–2233, 2006.

Victor DeMiguel, Lorenzo Garlappi, Francisco J. Nogales, and Raman Uppal. A generalized approach to portfolio optimization: Improving performance by constraining portfolio norms. *Management Science*, 55(5):798–812, 2009.

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

Christian L Dunis and Richard Ho. Cointegration portfolios of european equities for index tracking and market neutral strategies. *Journal of Asset Management*, 6(1):33–52, 2005.

Björn Fastrich, Sandra Paterlini, and Peter Winker. Cardinality versus q-norm constraints for index tracking. *Quantitative Finance*, 14(11):2019–2032, 2014.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017a.

Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *CoRL*, 2017b.

Sergio M Focardi and Frank J Fabozzi. A methodology for index tracking based on time-series clustering. *Quantitative Finance*, 4:417–425, 2004.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *ICML*, 2018.

Fernando García, Francisco Guijarro, and Javier Oliver. Index tracking optimization with cardinality constraint: a performance comparison of genetic algorithms and tabu search heuristics. *Neural Computing and Applications*, 2017.

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

Gianfranco Guastaroba, Renata Mansini, Wlodzimierz Ogryczak, and Maria Grazia Speranza. Linear programming models based on omega ratio for the enhanced index tracking problem. *European Journal of Operational Research*, 251(3):938–956, 2016.

S. W. Hong, P. Miasnikof, R. Kwon, and Y Lawryshyn. Market graph clustering via qubo and digital annealing. *Journal of Risk and Financial Management*, 14(1):34, 2021.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Ravi Jagannathan and Tongshu Ma. Risk reduction in large portfolios: Why imposing the wrong constraints helps. *The Journal of Finance*, 58(4):1651–1683, 2003.

Kaiyi Ji, Jason D Lee, Yingbin Liang, and H Vincent Poor. Convergence of meta-learning with task-specific adaptation over partial parameters. In *NeurIPS*, 2020.

Ata Kabán. Fractional norm regularization: Learning with very few relevant features. *IEEE transactions on neural networks and learning systems*, 24(6):953–963, 2013.

Thiemo Krink, Stefan Mittnik, and Sandra Paterlini. Differential evolution and combinatorial search for constrained index-tracking. *Annals of Operations Research*, 172(1):153, 2009.

Qian Li, Linyan Sun, and Liang Bao. Enhanced index tracking based on multi-objective immune algorithm. *Expert Systems with Applications*, 38(5):6101–6106, 2011.

Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *AISTATS*, 2020.

Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7:77–91, 1952.

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

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

Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *NeurIPS*, 2019.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI*, 2019.

Rubén Ruiz-Torrubiano and Alberto Suárez. A hybrid optimization approach to index tracking. *Annals of Operations Research*, 166(1):57–71, 2009.

Leonardo Riegel Sant'Anna, Tiago Pascoal Filomena, Pablo Cristini Guedes, and Denis Borenstein. Index tracking with controlled number of assets using a hybrid heuristic combining genetic algorithm and non-linear programming. *Annals of Operations Research*, 258(2):849–867, 2017.

Jürgen Schmidhuber. On learning how to learn learning strategies. Technical report, 1995.

Andrea Scozzari, Fabio Tardella, Sandra Paterlini, and Thiemo Krink. Exact and heuristic approaches for the index tracking problem with ucits constraints. *Annals of Operations Research*, 205(1):235–250, 2013.

William F Sharpe. The sharpe ratio. *Journal of portfolio management*, 21(1):49–58, 1994.

Lianjie Shu, Fangquan Shi, and Guoliang Tian. High-dimensional index tracking based on the adaptive elastic net. *Quantitative Finance*, 20(9):1513–1530, 2020.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NIPS*, 2017.

Junxiao Song, Prabhu Babu, and Daniel P Palomar. Sparse generalized eigenvalue problem via smooth optimization. *IEEE Transactions on Signal Processing*, 63(7):1627–1642, 2015.

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

Damien Teney and Anton van den Hengel. Visual question answering as a meta learning task. In *ECCV*, 2018.

Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

Maria Woodside-Oriakhi, C Lucas, and John E Beasley. Heuristic algorithms for the cardinality constrained efficient frontier. *European Journal of Operational Research*, 213(3):538–550, 2011.

Fengmin Xu, Zongben Xu, and Honggang Xue. Sparse index tracking based on $l_{1/2}$ model and algorithm. *arXiv preprint arXiv:1506.05867*, 2015.

Yu Zheng, Bowei Chen, Timothy M Hospedales, and Yongxin Yang. Index tracking with cardinality constraints: A stochastic neural networks approach. In *AAAI*, 2020a.

Yu Zheng, Timothy M Hospedales, and Yongxin Yang. Diversity and sparsity: A new perspective on index tracking. In *ICASSP*, 2020b.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017.

Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. In *ICLR*, 2018.

A Appendix

A.1 Pseudo code

The following pseudo-code reflects the full pipeline for one-time rebalancing.

Algorithm 1 Full pipeline for one-time rebalancing

```
Input: Total training data $\mathcal{X}\mathcal{Y}$
Input: Length for meta-train and meta-test data $D$ and $d$
Input: Initialised parameter $\theta$
 1: $N \leftarrow$ Number of assets in $\mathcal{X}$
 2: $T \leftarrow$ Total length for $\mathcal{Y}$
 3: $\mathrm{Attn} \leftarrow \mathrm{Attention}(\mathrm{num\_heads} = 1)$        ▷ Scaled Dot-Product Attention
 4: $FC_1 \leftarrow$ The first fully connected layer
 5: $FC_2 \leftarrow$ The second fully connected layer
 6: $X_t, y_t \leftarrow \mathcal{X}[T\text{-}D{:}T], \mathcal{Y}[T\text{-}D{:}T]$        ▷ $X_t$ and $y_t$ are the training data
 7: repeat
 8:     $R \leftarrow$ Random start        ▷ Random integer in the range of [0:$T$-$D$-$d$]
 9:     $X, y \leftarrow \mathcal{X}[R{:}R{+}D], \mathcal{Y}[R{:}R{+}D]$        ▷ Prepare the meta-train data
10:     $X', y' \leftarrow \mathcal{X}[R{+}D{:}R{+}D{+}d], \mathcal{Y}[R{+}D{:}R{+}D{+}d]$        ▷ Prepare the meta-test data
11:     $Y \leftarrow [y; \cdots ; y]$        ▷ $N$ copies of $y$ stacked by row
12:     $u \leftarrow [X^T, Y, \mathrm{Attn}(X^T, X^T, X^T)]$
13:     $\lambda \leftarrow \exp(\mathrm{FC}_2(\mathrm{FC}_1(u)))$
14:     $w^*, \alpha_2^*, P \leftarrow \mathrm{QP\_Solver}(\lambda, X, y)$        ▷ Offline QP solver (cvxopt)
15:     $L \leftarrow ||X'w^* - y'||_2^2$        ▷ Tracking error for meta-test data
16:     $\frac{\partial L}{\partial \lambda} \leftarrow \mathrm{metagradient}(w^*, \alpha_2^*, P, \frac{\partial L}{\partial w^*})$        ▷ $\frac{\partial L}{\partial w^*}$: Passed from back-propagation
17:     $\frac{\partial L}{\partial \theta} \leftarrow \frac{\partial L}{\partial \lambda} \cdot \frac{\partial \lambda}{\partial \theta}$        ▷ $\frac{\partial \lambda}{\partial \theta}$: Calculated by automatic differentiation
18:     Update $\theta$
19: until Convergence
20: $Y_t \leftarrow [y_t; \cdots ; y_t]$        ▷ $N$ copies of $y_t$ stacked by row
21: $u \leftarrow [X_t^T, Y_t, \mathrm{Attn}(X_t^T, X_t^T, X_t^T)]$
22: $\lambda \leftarrow \exp(\mathrm{FC}_2(\mathrm{FC}_1(u)))$
23: $w \leftarrow \mathrm{QP\_Solver}(X_t, y_t)$
24: return $w$
```

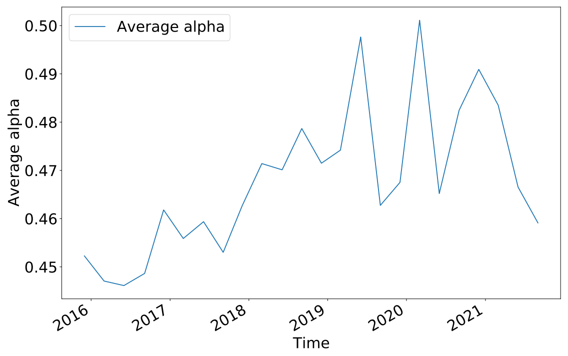
Algorithm 2 metagradient($w^*, \alpha_2^*, P, \frac{\partial L}{\partial w^*}$)

```
Input: $w^*, \alpha_2^*, P, \frac{\partial L}{\partial w^*}$
 1: $N \leftarrow$ the length of $w^*$
 2: $\frac{\partial L}{\partial s} \leftarrow [\frac{\partial L}{\partial w^*}, \mathrm{Zeros}(1,1), \mathrm{Zeros}(1,N)]$
 3: $K \leftarrow \begin{bmatrix} P & \mathrm{Ones}(N,1) & -\mathrm{Identity}(N) \\ \mathrm{Ones}(1,N) & \mathrm{Zeros}(1,1) & \mathrm{Zeros}(1,N) \\ -\mathrm{Diag}(\alpha_2^*) & \mathrm{Zeros}(N,1) & -\mathrm{Diag}(w^*) \end{bmatrix}$
 4: $v \leftarrow \begin{bmatrix} \mathrm{Identity}(N) \\ \mathrm{Zeros}(1,N) \\ \mathrm{Zeros}(N,N) \end{bmatrix}$
 5: $g^T \leftarrow \mathrm{LP\_Solver}(\mathrm{K}^\mathrm{T}, \text{-}(\frac{\partial \mathrm{L}}{\partial \mathrm{s}})^\mathrm{T})$        ▷ Offline LP solver (cvxopt)
 6: $\frac{\partial L}{\partial \lambda} \leftarrow g[0{:}N] \cdot v[0{:}N]{=}g[0{:}N]$
 7: return $\frac{\partial L}{\partial \lambda}$
```

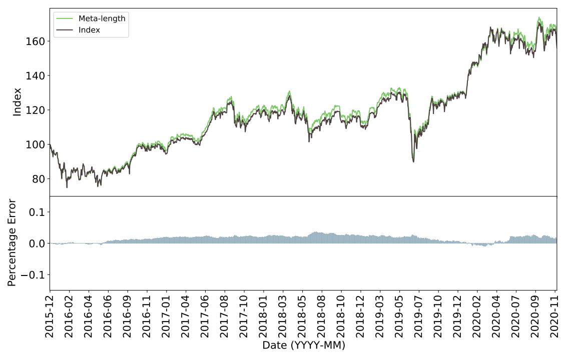
A.2 Meta learn the length of model training.

As mentioned in Sec. 4.1, the length for model training also can be meta learned. The meta learning framework for meta-learning the model training length can be modelled as:

$$\min_{w \geq \mathbf{0},\ \sum_i w_i = 1} \quad (Xw - Y)^T \Lambda (Xw - Y) \tag{15}$$



(a)



(b)

Figure 5: (a)The average α for different time. (b) The tracking accuracy for meta-train's changing lengths. Shadow area covers one standard deviation.

where $\Lambda \in \mathbb{R}^{D\times D}$ is a diagonal matrix that the off-diagonal elements are 0 (D is the predetermined length, such as $5Y$). When $D_{ii} = 1$, we choose the ith day for training and else drop the day. We use the meta-learning framework to learn a length smaller than D, such as $3Y$, as our model training length.

We adopt a parameter-driven approach to decide Λ. Define v as an equally spaced sequence from 0 to 1 with a spacing of $\frac{1}{D}$ from small to large:

$$v = [v_1, \cdots, v_D] \tag{16}$$

Define α as,

$$\alpha = [\text{sigmoid}(\frac{v_1 - \tau}{\sigma}), \cdots, \text{sigmoid}(\frac{v_D - \tau}{\sigma})] \tag{17}$$

Where τ is the parameter that we need to optimise and σ is a hyper-parameter, and in this part, we set $\sigma = 0.05$. We know α is composed of numbers between 0 and 1. Then, we can define $\Lambda = \text{Diag}(\alpha)$. Λ is a diagonal matrix, and we use these elements that equals 1. Through such operations, we can select a subset of the predetermined D days. After these settings, Eq. 15 can be expressed as:

$$\min_{w\geq\mathbf{0},\ \sum_i w_i=1} \quad (Xw - Y)^T\Lambda(\tau)(Xw - Y) \tag{18}$$

We know in the QP form of Eq. 18, $P = 2X^T\Lambda(\tau)X$, $q = \lambda - 2X^T\Lambda(\tau)y$, $G = -I$, $h = \mathbf{0}$. $A = \mathbf{1}^T$, $b = 1$. Assuming that the optimal solution w^* is produced by an offline QP solver: $w^*(\tau) = \text{QP\_Solver}(P, q, G, h, A, b)$ as Eq. 6, and we need to compute the gradient of w^* w.r.t. τ, i.e., $\frac{\partial w^*}{\partial \lambda}$. Following Amos & Kolter (2017); Lorraine et al. (2020), we derive the gradient via implicit function theorem (IFT). Note that λ appears in the P and q term only, thus we apply the chain rule,

$$\frac{\partial w^*}{\partial \tau} = \frac{\partial w^*}{\partial P}\frac{\partial P}{\partial \tau} + \frac{\partial w^*}{\partial q}\frac{\partial q}{\partial \tau} \tag{19}$$

The two terms $\frac{\partial P}{\partial \tau}$ and $\frac{\partial q}{\partial \tau}$ can be obtained from the existing gradients by using a neural network. If the QP problem is solved by some generalised Lagrange multiplier based methods, the *full* output of QP_Solver will be $s = [w^*, \alpha_1^*, \alpha_2^*]$ where $\alpha_1^* \in \mathbb{R}^1$ is the Lagrange multiplier for the equality constraint and $\alpha_2^* \in \mathbb{R}^N_{\geq 0}$ is the Lagrange multiplier for the inequality constraint.

If we have the gradients $g_1 = \frac{\partial s}{\partial P} \in \mathbb{R}^{(2N+1)\times(D\times D)}$ and $g_2 = \frac{\partial s}{\partial q} \in \mathbb{R}^{(2N+1)\times D}$, then the sought gradients $\frac{\partial w^*}{\partial P}$ and $\frac{\partial w^*}{\partial q}$ are simply the first N rows in g_1 and g_2, respectively. Meanwhile $\frac{\partial s}{\partial P}$ and $\frac{\partial s}{\partial q}$ can be obtained by *implicit function theorem* (IFT)

$$\frac{\partial s}{\partial P} = -(\frac{\partial f}{\partial s})^{-1}\frac{\partial f}{\partial P},\ \frac{\partial s}{\partial q} = -(\frac{\partial f}{\partial s})^{-1}\frac{\partial f}{\partial q} \tag{20}$$

Here f, P, q, and s must satisfy $f(P, q, s(P, q)) = \mathbf{0}$, and $f(P, q, s(P, q)) = \mathbf{0}$ can be obtained from KKT conditions as Eq. 9. Thus, the corresponding derivatives are $\frac{\partial f}{\partial s} = K$ as Eq. 10, and

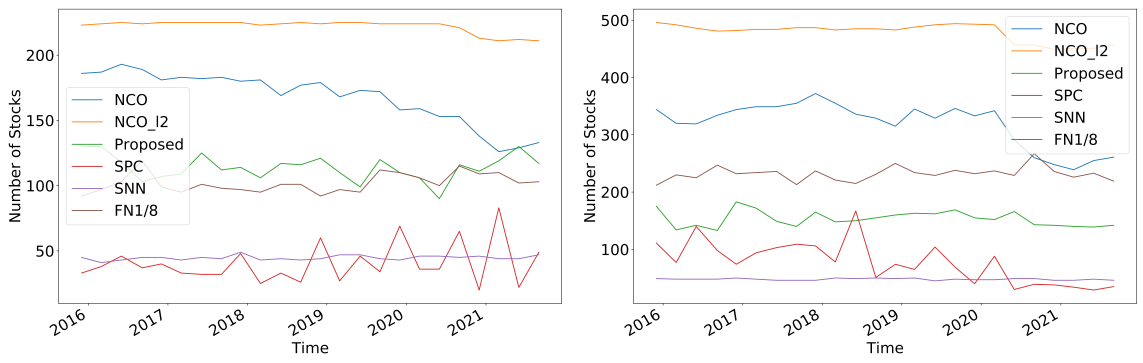


Figure 6: Sparsity over w. The left and right plots represent the NIKKEI 225 and S&P 500 index.

$$\frac{\partial f}{\partial P} = \begin{bmatrix} \frac{\partial (Pw*)}{\partial P} \\ 0 \otimes [\mathbf{0} \dots \mathbf{0}] \\ \mathbf{0} \otimes [\mathbf{0} \dots \mathbf{0}] \end{bmatrix} \rightarrow v_1, \; \frac{\partial f}{\partial q} = \begin{bmatrix} I \\ \mathbf{0}^T \\ \mathbf{0} \dots \mathbf{0} \end{bmatrix} \rightarrow v_2$$

where $\otimes$ represents the Kronecker product for two matrix. We connect $\frac{\partial w^*}{\partial \tau}$ with the existing gradient passed from back-propagation, i.e., $\frac{\partial P}{\partial \tau}$, $\frac{\partial q}{\partial \tau}$, and $\frac{\partial L}{\partial w^*}$ where L is the final (meta-test) loss, and . First, we pad $\frac{\partial L}{\partial w^*}$ with zeros as $\frac{\partial L}{\partial s} = [\frac{\partial L}{\partial w^*}, 0, \mathbf{0}]$, Then we can compute $\frac{\partial L}{\partial \tau}$ as Eq. 21, and finally optimise the parameter τ.

$$\begin{aligned} \frac{\partial L}{\partial \tau} &= \frac{\partial L}{\partial s} \cdot \frac{\partial s}{\partial \tau} = \frac{\partial L}{\partial s} \cdot (\frac{\partial s}{\partial P} \cdot \frac{\partial P}{\partial \tau} + \frac{\partial s}{\partial q} \cdot \frac{\partial q}{\partial \tau}) \\ &= -\frac{\partial L}{\partial s}(K^{-1} v_1 \frac{\partial P}{\partial \tau} + K^{-1} v_2 \frac{\partial q}{\partial \tau}) \end{aligned} \tag{21}$$

We report the average value for α in Fig. 5(a), and we can conclude that the optimal length for meta-train is changing over time. As a result, length for meta-train should also be data-dependent. Besides, this setting can results a outstanding tracking performance, and result is shown in Fig. 5(b).

A.3 Figures for sparsity

As the partial index tracking problem aims to replicate the benchmark using a small number of assets, it is necessary to obtain a sparse solution. We plot the number of selected stocks (weight larger than 10^{-6}) for different methods based on two large indices: NIKKEI 225 and S&P 500, and Fig. 6 shows that our method produces a moderately sparse solution. Competitors SNN and SPC produce even more sparse solutions, but at the cost of reduced tracking accuracy.

A.4 Equity curves

We report the equity curves for different methods according to the indices, and Fig. 7, Fig. 8, Fig. 9, and Fig. 10 exhibits the results for SENSEX 30, CAC 40, NIKKEI 225, and S&P 500 indices, respectively.

A.5 Equity curves for ablation studies

We report the equity curves according to different ablation considerations, and Fig. 11, Fig. 12, Fig. 13, and Fig. 14 represent the results for f_θ designing, rebalancing frequency, total training length, and cross-index f_θ deploying, respectively.

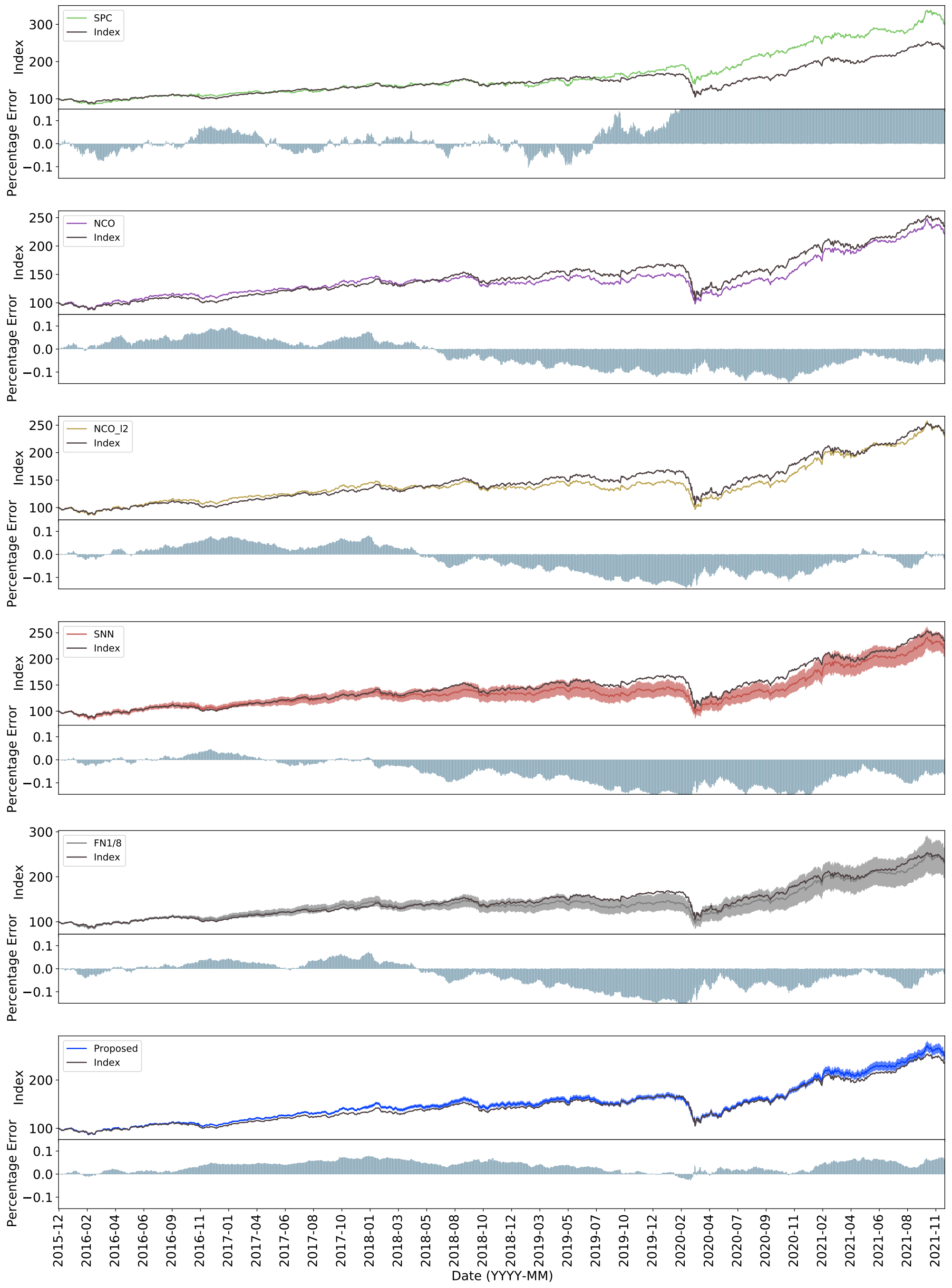


Figure 7: Index tracking performance for different methods based on SENSEX 30 index. From up to the bottom are SPC, NCO, NCO_ℓ_2, SNN, FN1/8, and our proposed methods, respectively. For every single plot, the top plots are the index and the trackers, and the bottom is the percentage tracking error $\frac{\hat{y}-y}{y}$. Shadow area covers one standard deviation for SNN, FN1/8, and our proposed method.

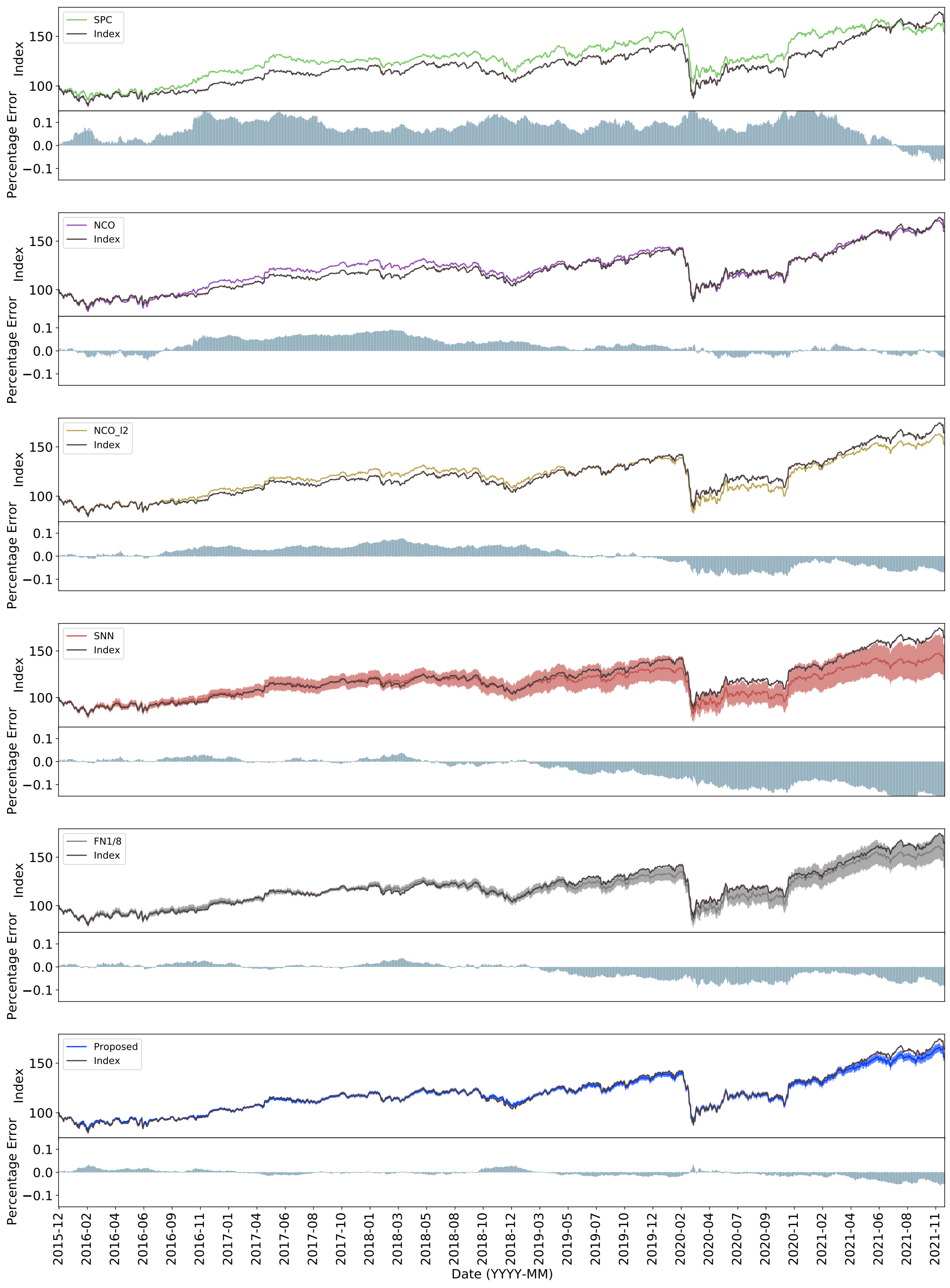


Figure 8: Index tracking performance for different methods based on CAC 40 index. From up to the bottom are SPC, NCO, NCO_ℓ_2, SNN, FN1/8, and our proposed methods, respectively. For every single plot, the top plots are the index and the trackers, and the bottom is the percentage tracking error $\frac{\hat{y}-y}{y}$. Shadow area covers one standard deviation for SNN, FN1/8, and our proposed method.

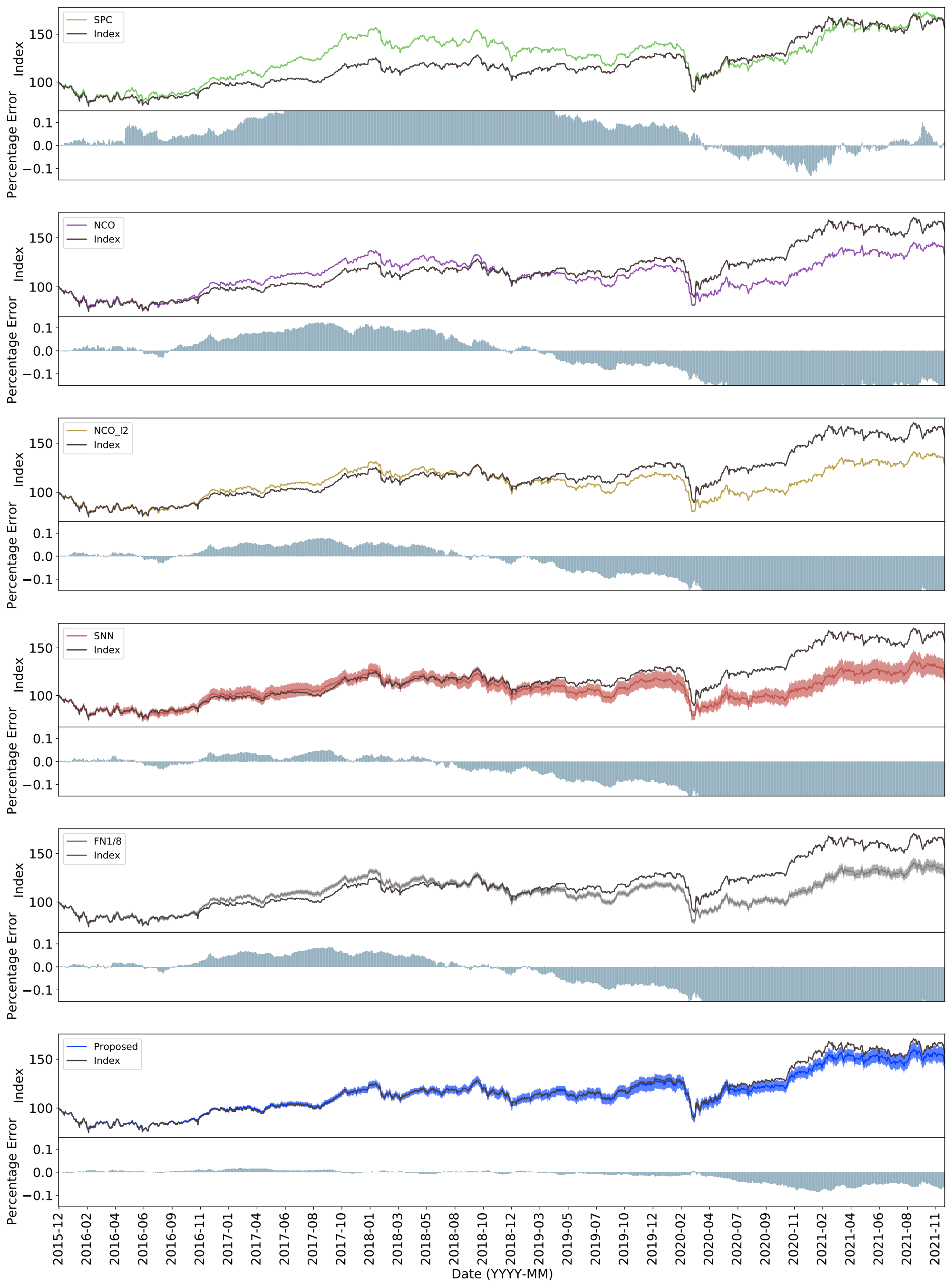


Figure 9: Index tracking performance for different methods based on NIKKEI 225 index. From up to the bottom are SPC, NCO, NCO_ℓ_2, SNN, FN1/8, and our proposed methods, respectively. For every single plot, the top plots are the index and the trackers, and the bottom is the percentage tracking error $\frac{\hat{y}-y}{y}$. Shadow area covers one standard deviation for SNN, FN1/8, and our proposed method.

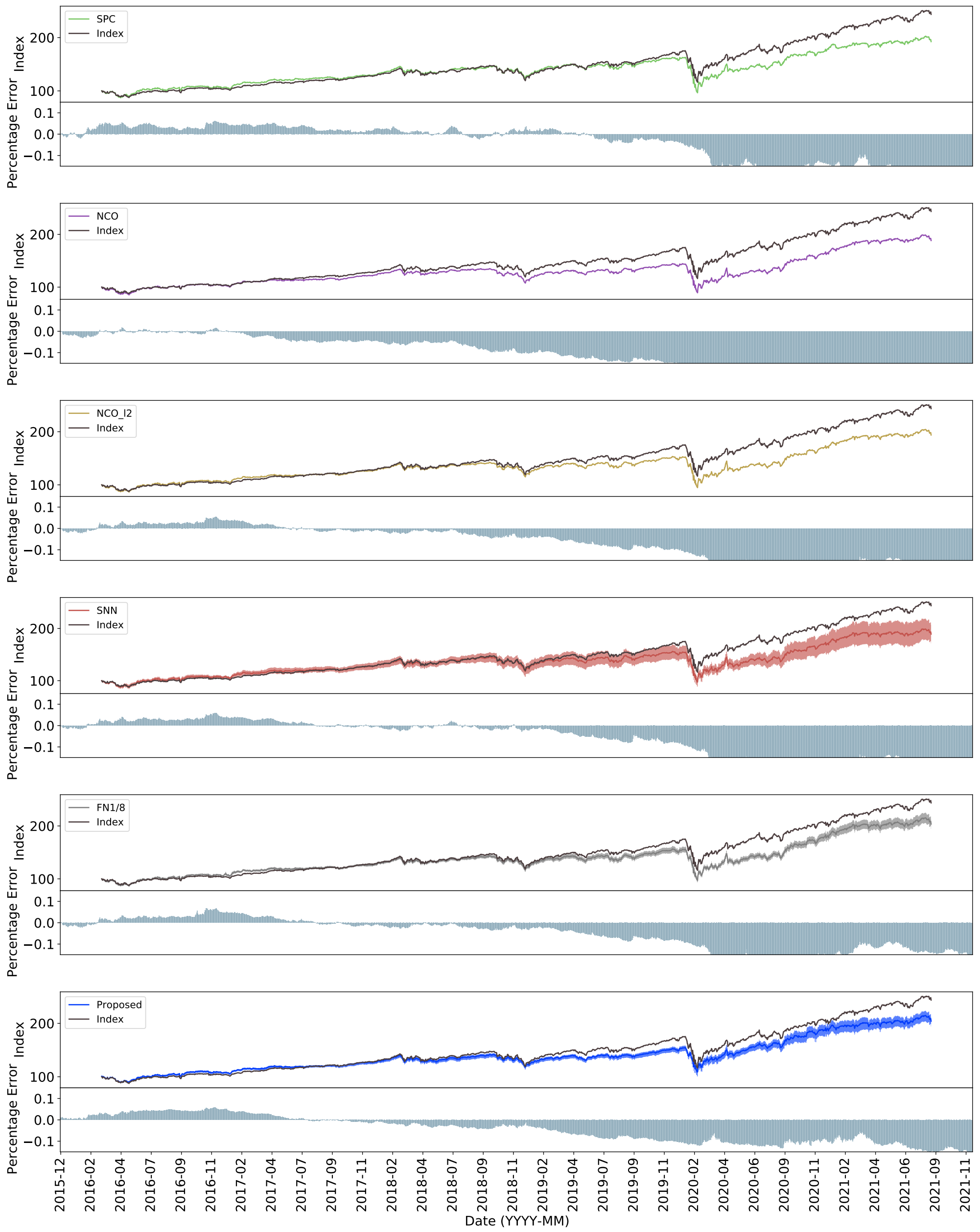


Figure 10: Index tracking performance for different methods based on S&P 500 index. From up to the bottom are SPC, NCO, NCO_ℓ_2, SNN, FN1/8, and our proposed methods, respectively. For every single plot, the top plots are the index and the trackers, and the bottom is the percentage tracking error $\frac{\hat{y}-y}{y}$. Shadow area covers one standard deviation for SNN, FN1/8, and our proposed method.

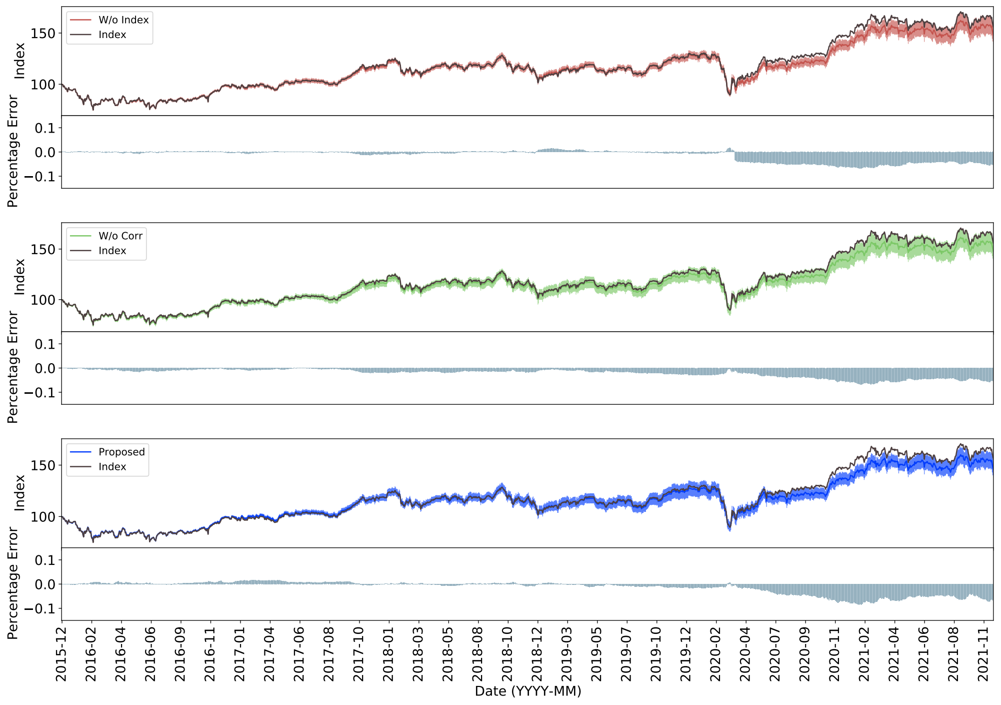


Figure 11: Index tracking performance for different f_θ function designing based on NIKKEI 225 index. From up to the bottom are $u_i = [x_i, \mathrm{Attn}(x_i, X^T, X^T)]$, $u_i = [x_i, y]$, and $u_i = [x_i, y, \mathrm{Attn}(x_i, X^T, X^T)]$. For every single plot, the top plots are the index and the trackers, and the bottom is the percentage tracking error $\frac{\hat{y}-y}{y}$. Shadow area covers one standard deviation.

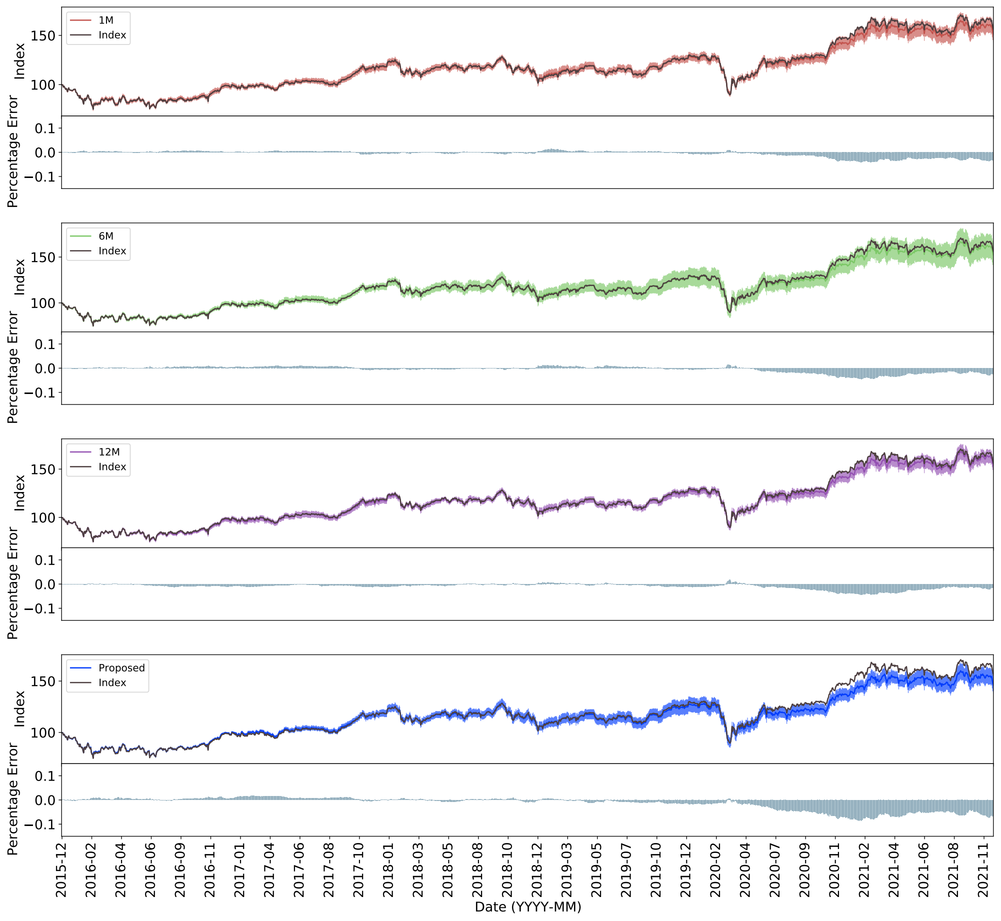


Figure 12: Index tracking performance for different rebalancing frequencies based on NIKKEI 225 index. From up to the bottom are monthly, semi-yearly, yearly, and quarterly. For every single plot, the top plots are the index and the trackers, and the bottom is the percentage tracking error $\frac{\hat{y}-y}{y}$. Shadow area covers one standard deviation.

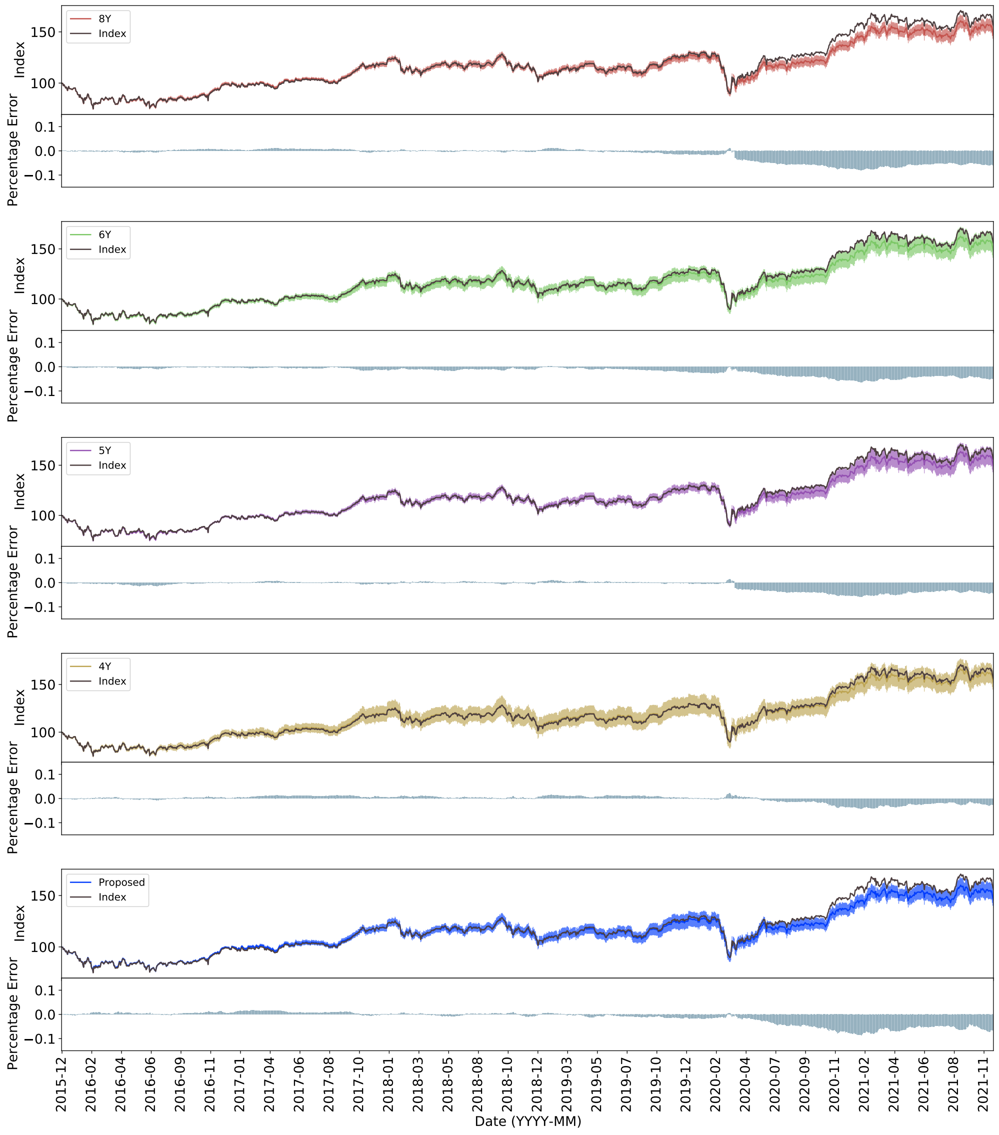


Figure 13: Index tracking performance for different total training lengths based on NIKKEI 225 index. From up to the bottom are $8Y$, $6Y$, $5Y$, $4Y$, and $10Y$. For every single plot, the top plots are the index and the trackers, and the bottom is the percentage tracking error $\frac{\hat{y}-y}{y}$. Shadow area covers one standard deviation.

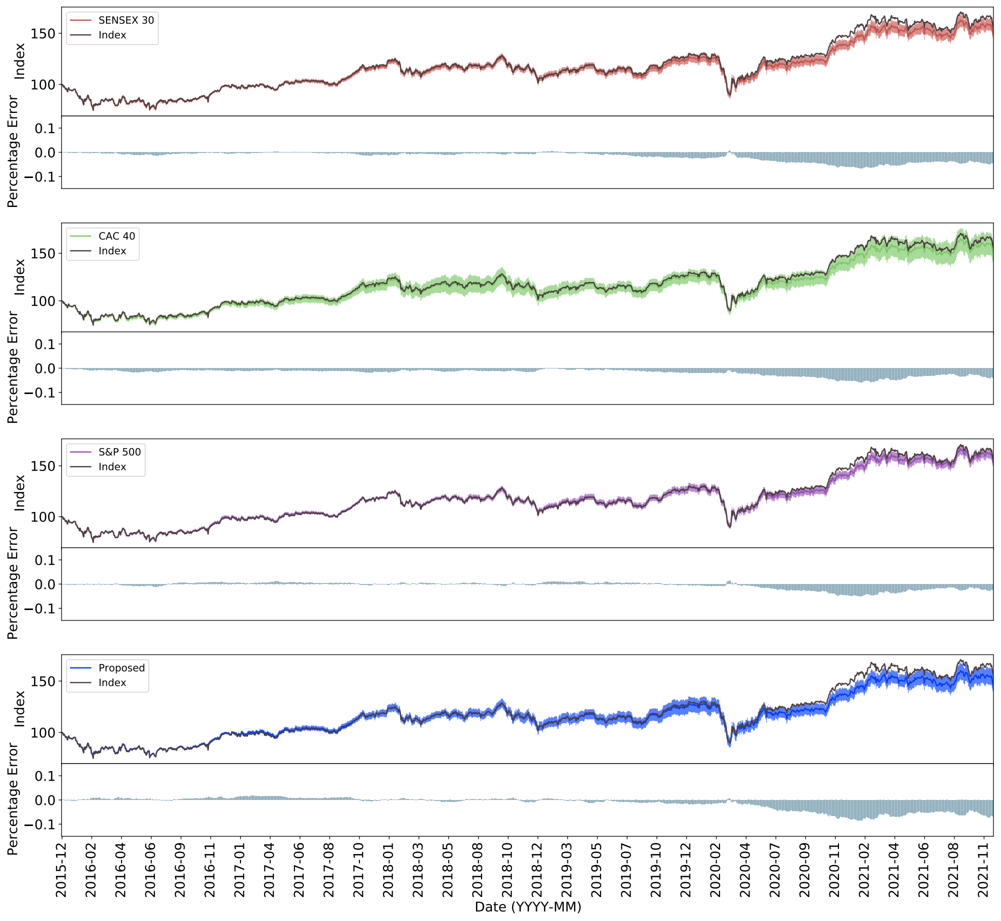


Figure 14: Index tracking performance for cross-index f_θ deploying based on NIKKEI 225 index. From up to the bottom are SENSEX 30, CAC 40, S&P 500, and NIKKEI 225 indices. For every single plot, the top plots are the index and the trackers, and the bottom is the percentage tracking error $\frac{\hat{y}-y}{y}$. Shadow area covers one standard deviation.