ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

# REPORT

## REPLICATING PORTFOLIO USING LIMITED NUMBER OF STOCKS

---

Dusana Milinkovic
Hippolyte Croué-Maës
Alexander Schaller

23rd December 2023

# CHAPTER 1

# INTRODUCTION

## 1.1 MOTIVATION FOR INDEX REPLICATION

Indices are an excellent and cost-effective way to produce insights on broader industry topics and set benchmarks for whole industries. The index creation process is fairly adhoc, typically such an index is defined based on a selection of assets, then their weights are defined. The weighting is typically either equal-weighted, size-weighted, or risk-weighted. In all cases, this is typically an easy rule. Take the S&P 500 index, the rules in the big picture are fairly simple, take the 500 largest stocks by market capitalization, apply weighting by size and the index is ready. The traditional way to replicate such an index is also not very complex, allocate your investment capital to each asset based on the weighting criteria, round each share ownership to the nearest integer, and rebalance based on the schedule of the index' schedule.

In a simple economy with no competitors, this method works very well but often requires large capital commitments to cater to expensive shares. In a competitive economy, this can push smaller players out of the game and make differentiation of products hard. A solution to this problem of differentiation would be to lower yearly fees in a way so as to undercut the competition. However, fees in this scenario are highly dependent on the holdings defined by the index, to mitigate this one could for example only hold the 5 highest-weighted stocks in the index. In this manner, the fees and capital allocation are lower than before. On the downside, the replication may perform rather poorly, in terms of tracking error.

This simple example shows that for competition to take place, there need to be methods to lower capital allocation and fees. This report proposes several methods to achieve this feat by carefully selecting which assets are held in the replicating portfolio to minimize fees, tracking error, and capital allocation.

An additional step to consider may be that certain indices, such as the Refinitiv Venture Capital Index, track private market assets with undefined weights and assets. In this scenario, there is no way to replicate it using its constituents. This leads us to our second motivation, which tackles exactly this problem.

## 1.2 MOTIVATION FOR PORTFOLIO REPLICATION

In a more complex scenario, we can look at general portfolio replication, where one has a universe of assets and weights that make up a portfolio. These are unknown and cannot be traded. There exists a second, larger universe of assets which is known and can be traded. Then there must be a way to define weights such that one can replicate this portfolio.

For example, as mentioned in the previous section, you are an investor who has heard from a friend who works in the venture capital industry that VC is currently performing very well and has outperformed the

general market over the last few years. In this scenario, if one has sufficient funds available one can make direct investments in VC funds and reap the benefits. However, if one lacks this sort of funding, such investments may be unreachable. A solution would be to offer a financial product that tracks the general VC industry. This is exactly the problem of portfolio replication outlined above.

We have seen multiple reasons why such an approach may be useful, the argument for lower transaction costs and lower capital allocation and the argument for tracking private portfolios using public market assets. The methods outlined in the next few chapters try to tackle this exact problem of asset selection for the replication and then portfolio construction using these assets with low tracking errors.

Our work will be based on mainly 2 reference papers covering non negative lasso regression for index replication 'Nonnegative-lasso and application in index tracking'(Wu, Yang and Liu 2014) as well as a more recent paper 'Stock-Index Tracking Optimization Using Auto-Encoders'(Zhang et al. 2020) showing growing interests in using autoencoders for index tracking purpose.

# CHAPTER 2

# METHODS

## 2.1 DATASET, INDEX, AVAILABLE STOCKS FOR REPLICATION

The project is based on daily returns of consituents of the MSCI World Index (*Prospectus of MSCI World Index (USD)* 2023) from March 6th 2002 to September 20th 2023. In a first step the data was cleaned to remove any assets that were not traded over the full 20 year period, that reduced the number of available assets to 984 assets. Finally, the original dataset included data for weekends which where non-trading days, so we removed all weekends from the dataset. Leaving us with 5621 days of data.

In a second step, we selected 100 of the assets to generate a completely random index. The weights were selected on the first date and then not changed over the 20 year period, that is the index was never rebalanced. The index has an average performance over the 20 years of approximately 7.44% per annum, leading to a 20-year performance of 4.2x. This performance is comparable to the historical annualized performance of the MSCI World Index which lies at 7.9% over the past 20 years. (*Historical Performance of MSCI World Index (USD)* 2023)

After the index was created we removed those 100 assets from our replicating universe, which yields 884 assets for possible replication. In this manner, we created two asset universes, one for the index and one for the replication. This is great for robustness as well as for the simulation of a scenario in which the weights and constituents of the index are unknown or untradeable.

## 2.2 NONNEGATIVE-LASSO

For feature selection as the benchmark approach, we have decided to use the Nonnegative-lasso method for a couple of reasons. This method stands out as a popular approach for both model selection and parameter estimation within linear regression models. The Lasso method is preferred for preventing overfitting, while the Nonnegative-lasso method imposes constraints on short-selling.

Lasso Regression, though not as popular as other machine learning algorithms like linear or logistic regression, offers unique advantages. LASSO stands for Least Absolute Shrinkage and Selection Operator, which is a modification of Ridge Regression. As opposed to Ridge Regression, which penalizes based on the squared value of parameters, Lasso Regression uses the absolute value. This means that in Lasso Regression, some variables can be entirely excluded from the model's penalty, allowing it to be less sensitive to training data and focusing only on relevant variables.

Lasso Regression was shown to be a useful method when dealing with datasets that have numerous variables. Such datasets can result in overfitting, where the model fits the training data too closely but

performs poorly on unseen data. Lasso Regression helps address this by not only preventing overfitting but also by selecting only the most meaningful variables for the model, effectively ignoring those that contribute little or no value. This unique approach makes Lasso Regression our choice as we want the model with better predictive accuracy.

The goal of the algorithm is to find weights of selected features, such that we **minimize** the following objective function:

$$\sum_{i=1}^{n} \left( y_i - \sum_j x_{ij}\beta_j \right)^2 + \alpha \sum_{j=1}^{p} |\beta_j|$$

Tuning the parameter $\alpha$ is crucial as it determines the penalty's strength in Lasso Regression. When $\alpha = 0$, no features are excluded, while a large $\alpha$ eliminates all features. Finding the right balance for $\lambda$ is key to achieving optimal model performance.

We choose the $\alpha$ value through a cross-validation process. We divide the dataset into multiple folds, and $\alpha$ values ranging from $10^{-6}$ to $5 \cdot 10^{-5}$ are linearly spaced across these folds. For each fold, we train the regression model using a particular $\alpha$ value, and its performance is evaluated on a validation set. We compute the score for each $\alpha$, which is the sum of mean squared error and drawdown. The $\alpha$ giving the lowest score across all folds is considered the best and is selected for the next 28-day prediction.

We used the rolling windows of 5 months. We use the data points of the most recent 5 months to determine the features and their weights for the next month. Therefore, rebalancing occurs every month based on the previous 5 months of data.

As suggested in the paper 'Nonnegative-lasso and application in index tracking' (Wu, Yang and Liu 2014), we have also implemented and tested another method:

### 2.2.1 NONNEGATIVE-LASSO AND NONNEGATIVE LEAST SQUARES

This is a two-stage method where we use Lasso only to determine which features should be included. The second step is to use LS to determine the weights of these selected features. However, LS may determine the weight of some features to be 0. In this case, we do not use LS as we don't want to exclude features, but we use the weights determined by Lasso.

## 2.3 AUTO ENCODER

**Feature selection process**

The methodology utilizes autoencoders (AEs) to create a metric for feature selection, diverging from the Lasso approach that selects features based on an L1 penalty in optimization. The core concept of AE involves compressing input data into a smaller deep latent representation, then decompressing and attempting to reconstruct the data with high accuracy. The key metric here is the reconstruction error, which aids in identifying stocks with unusual or idiosyncratic behaviors.

For training, the AE will process a time series of returns from 100 stocks, following a rolling window approach similar to Lasso. The data from the preceding year will be split into 10 months for training and 2 months for testing. The average reconstruction error for each stock's test time series of returns will be calculated as follows:

$$D^j_{\text{error}} = \sum_{t_{\text{Test}}} \left| X_j(t) - \hat{X}_j(t) \right|$$

The selection process, based on reference papers, involves constructing a portfolio foundation using features with the lowest and highest test decoding errors, denoted as $X$ and $Y$ respectively ($X < Y$). Stocks with lower decoding errors, sharing more communal information, are easier to decode and thus suitable for tracking index trends. Conversely, stocks with higher decoding errors, having less communal information, are ideal for capturing the index's more erratic behaviors.

Three AE types will be tested: Undercomplete, Sparse, and Denoising. These are shown to be effective in handling highly variable and noisy financial data. The Sparse AE is crucial for enhancing learning and convergence, achieved by randomly deactivating neurons during training and testing, acting as a form of L1 regularization and defense against overfitting. The Denoising AE aims to increase the network's robustness during the learning process. By adding white noise to the input data, it accommodates the inherent noise in financial data, making the learning process more challenging and robust.

**Portfolio construction:**

Having our feature selected we now aim to find weights for our portfolio. We then run non-negative OLS regression to get portfolio weights. Potential improvements could be to try to run Ridge regression to avoid overfitting but in our case, we had no visible sign of overfitting over each period. It would require tuning one more parameter that could become problematic as we already have to cross-validate crucial parameters for AE's.

It is important to note that X will be fixed to 10 stocks and $Y \in \{10, 20, 30, 40, 50\}$

**Hyperparameter tuning:**

Hyperparameter tuning is a critical aspect when working with autoencoders, necessitating precise adjustments. In this process, the entirety of the time series data is employed for cross-validation. Initially, the data is segmented into several folds (keeping the time order in folds obviously as we work with time series), each corresponding to one hyperparameters value under evaluation. The length of each fold is determined by dividing the total length of the dataset by the number of parameters value we want to cross-validate on.

For cross-validation, mean squared error (MSE) serves as the key metric. Each fold is then splited into two subsets: training and testing. The model specific to each fold is trained on its training subset and subsequently evaluated on the testing subset. The decoding error, assessed on the test set for each fold across all features, represents the test output. Following this, features are selected, regression is performed, and the MSE for that period is calculated. The fold exhibiting the lowest error is indicative of the most suitable hyperparameter.

In the context of sparse autoencoders, the penalty imposed on neuron activation within the hidden layer must be carefully calibrated to avoid underfitting. The denoising autoencoder, similarly, undergoes a cross-validation process to ascertain the appropriate level of noise. The aim is to find a balance where the noise is sufficient to meaningfully impact the learning process without causing data degradation. We build the denoising autoencoder to have the same hidden layer dimension as what was cross validated for the undercomplete autoencoder, following the aforementioned cross-validation procedure.

Note that all of our 3 AE have only 1 hidden layer as it was shown sufficient by 'Stock-Index Tracking Optimization Using Auto-Encoders' (Zhang et al. 2020).

## 2.4 METRICS

### 2.4.1 TRANSACTION COSTS

The transaction costs we included in the calculation where fixed and linear costs at each rebalancing day. The computation is simple, we consider a fixed fee of 5bps and a linear cost of 10bps. These assumptions were made based on the authors' discretion and are not necessarily founded in reality. It is to note that Ledoit and Wolf 2022 mentions an average transaction cost of 3.9bps per dollar. However, it is possible to recompute the methods to include different transaction cost assumptions. The costs are then as such:

$$\text{T-Costs} = \text{ABS\_TCOSTS} + \text{REL\_TCOSTS}(w_t - w_{t-1})$$

### 2.4.2 TRACKING ERROR

Tracking error is an excellent way to show the performance of the replicating portfolio. It gives the standard deviation of the difference between the replicating portfolio $P_r$ and the index $I$.

$$\text{Tracking Error} = \sigma(P_r - I)$$

where $\sigma$ signifies the function that calculates the empirical standard deviation. The lower the tracking error the better the replication.

### 2.4.3 RELATIVE DRAWDOWN

Relative drawdown is a metric to show how much the largest peak to trough movement is. This metric always has to be compared to that of the index. If the two match closely that means that given a bad performance of the index the replicating portfolio does not perform much worse or much better. This is another way to look at how well the tracking is as it shows the replication during the worst drawdowns. We computed it as follows:

$$\text{Rel. Drawdown} = \max_t \left( \frac{\text{Peak Cum. Returns}_t - \text{Cum. Returns}_t}{\text{Peak Cum. Returns}_t} \right)$$

# CHAPTER 3

# RESULTS

## 3.1 BENCHMARK

We can see in Fig. 3.1 the impact of the rebalancing in both methods compared to the Lasso method without rebalancing. This result is expected. We also observe that while the Lasso method more closely follows the index, the Lasso + LS method outperforms the index overall.

However, when the T-costs are taken into account these results change. In this setup, we clearly see that Lasso + LS outperforms the Lasso method in the long run. The effect of the transaction costs is large as none of the methods manages to come close to the index itself. Having that the no-rebalancing method performs the best draws a question of balance between the cost of these transactions compared to how often we should actually rebalance the portfolio.

When we do not decide to rebalance, it happens that Lasso and Lasso + LS perform the same. This happens due to the situation explained earlier. As LS decided to "cancel out" some of the features pre-selected by Lasso, we chose weights determined by Lasso. That is why it happens that 2 methods in this case provide the same result.

TABLE 3.1
Performance Metrics for the Benchmark Method

| | Lasso Only | Lasso + Least Squares | Rebalancing and Lasso | Rebalancing and Lasso + Least Squares |
|---|---|---|---|---|
| **Tracking Error** | 0.155 | 0.155 | 0.123 | 0.156 |
| **Relative Drawdown** | 1.541 | 1.541 | 1.153 | 1.541 |
| **T-Costs** | 0.022 | 0.022 | 2.392 | 1.943 |

Although Lasso with rebalancing has the most favorable tracking error as can be seen in Table 3.1, the tradeoff between the error and the transaction costs must be taken into consideration.

## 3.2 AUTOENCODERS

The performance of autoencoders (AEs) provides strong evidence of their utility in index replication.

The relationship between feature count, tracking error, and transaction costs is a nuanced tradeoff. The denoising AE, with 40 features (number of features from now on refers to the number of smallest

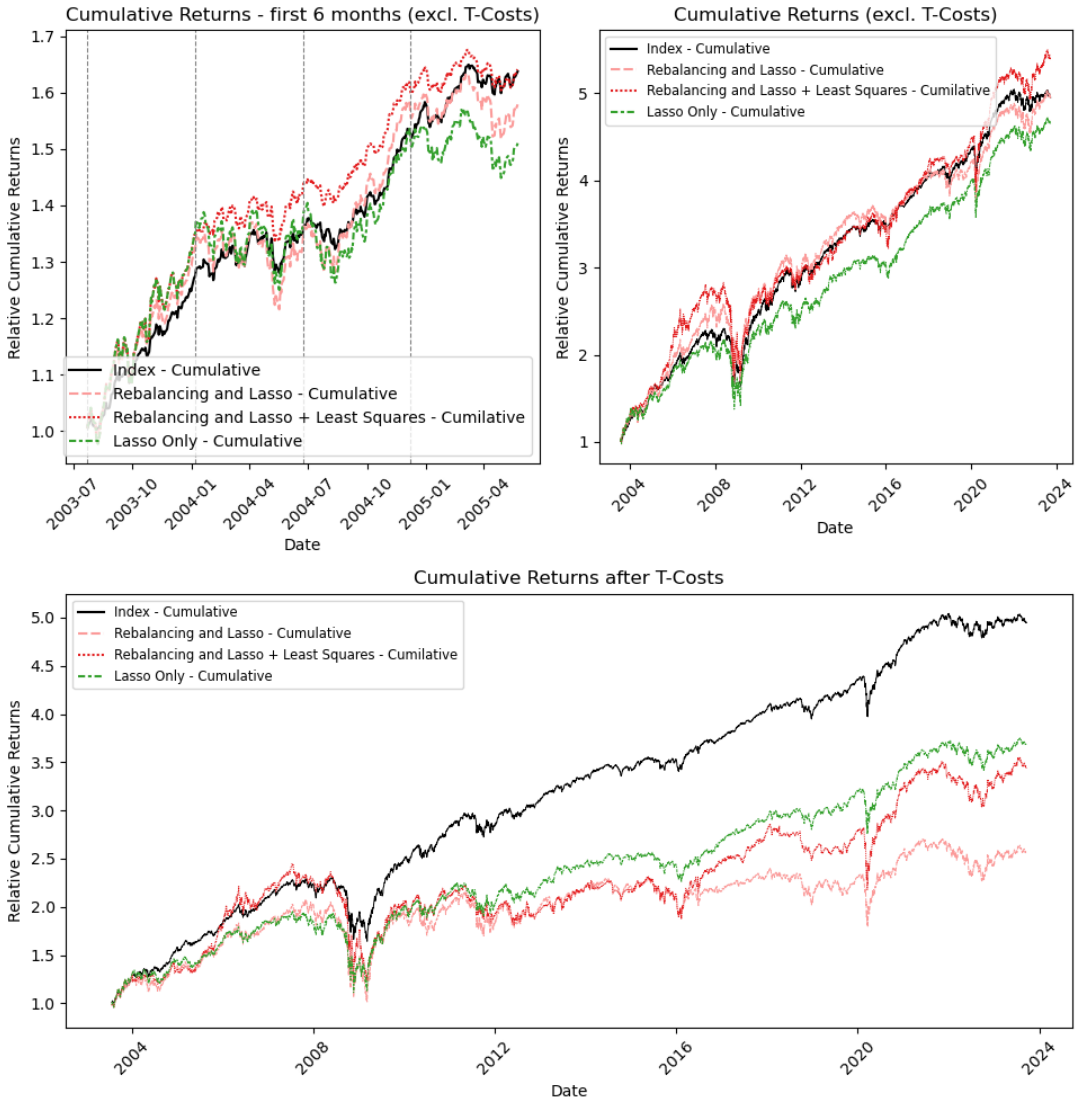


**FIGURE 3.1**
The best benchmark approaches plotted. The bottom plot shows performance net of fees, while the two methods above show performance before any fees.

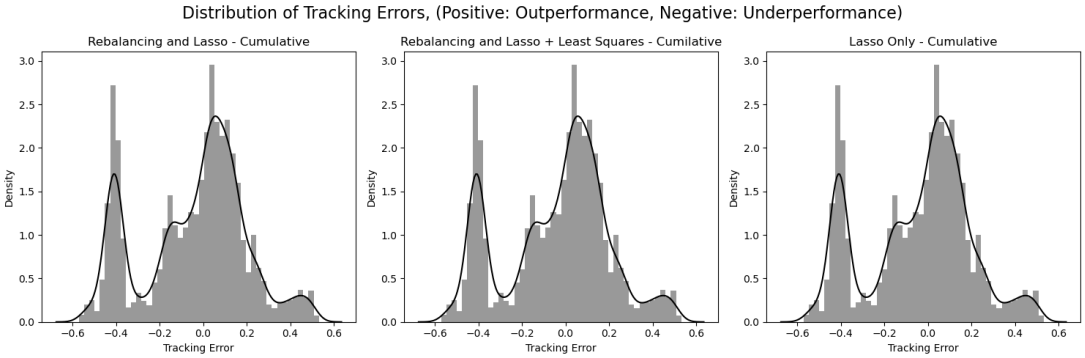


**FIGURE 3.2**
The following show the distribution of daily tracking errors for each of the three most performant methods of the benchmark.

communal information selected features: Y, as X is fixed to 10), presents a compelling case by minimizing tracking error—as seen in Table 3.2, where the tracking error for denoising AE with 40 features is just 0.026 and also managing drawdowns effectively as it is reaching lower values than best benchmark approach. This specific feature count seems to strike an optimal balance, supporting the selection of 40 as a strategic number of features.

Examining the cumulative returns after accounting for transaction costs, it's evident that AEs can maintain satisfactory index tracking at relatively low costs. A leaner approach with only 10 features could also be efficient(Fig. 3.4), offering lower costs as shown in Table 3.3, where T-Costs for denoising AE with 10 features are 0.375133, despite a slight compromise on tracking precision. Drawdowns with this setup are also surprisingly low.

The distribution of tracking errors underscores the stability that denoising AEs offer over sparse AEs. Sparse AEs demonstrate a wider and more negatively skewed error distribution, indicating a tendency towards underperformance (Fig. 3.5). This suggests potential inadequacies in the training phase, and that additional optimization could be beneficial. The denoising AE exhibits a narrower error distribution, which, although slightly more negatively skewed than the undercomplete AE, is still indicative of a more controlled and predictable replication process.

Sparse AE's underperformance, particularly notable with a tracking error of 0.416 for 10 features, may stem from suboptimal training. Improved results could potentially be achieved with further fine-tuning and robust testing.

In conclusion, the choice between tracking error and transaction costs is inherently subjective and depends on individual investment strategies. However, the data suggests that denoising AEs, especially with an intermediate number of features like 40, offer a balanced solution for index replication that considers both accuracy, drawdowns safety and operational efficiency(Fig. 3.3).

**TABLE 3.2**
Tracking error

| | Simple | Sparse | Denoising |
|---|---|---|---|
| # Least: 10 | 0.061896 | 0.416534 | 0.077577 |
| # Least: 20 | 0.070920 | 0.454620 | 0.073977 |
| # Least: 30 | 0.068745 | 0.453319 | 0.043980 |
| # Least: 40 | 0.035957 | 0.453020 | 0.026589 |
| # Least: 50 | 0.118181 | 0.399026 | 0.081183 |

**TABLE 3.3**
T-Costs

| | Simple AE | Sparse AE | Denoising AE |
|---|---|---|---|
| # Least: 10 | 0.374435 | 0.374351 | 0.375133 |
| # Least: 20 | 0.597303 | 0.598866 | 0.596949 |
| # Least: 30 | 0.817726 | 0.820175 | 0.819637 |
| # Least: 40 | 1.038168 | 1.040561 | 1.039811 |
| # Least: 50 | 1.259181 | 1.261164 | 1.260566 |

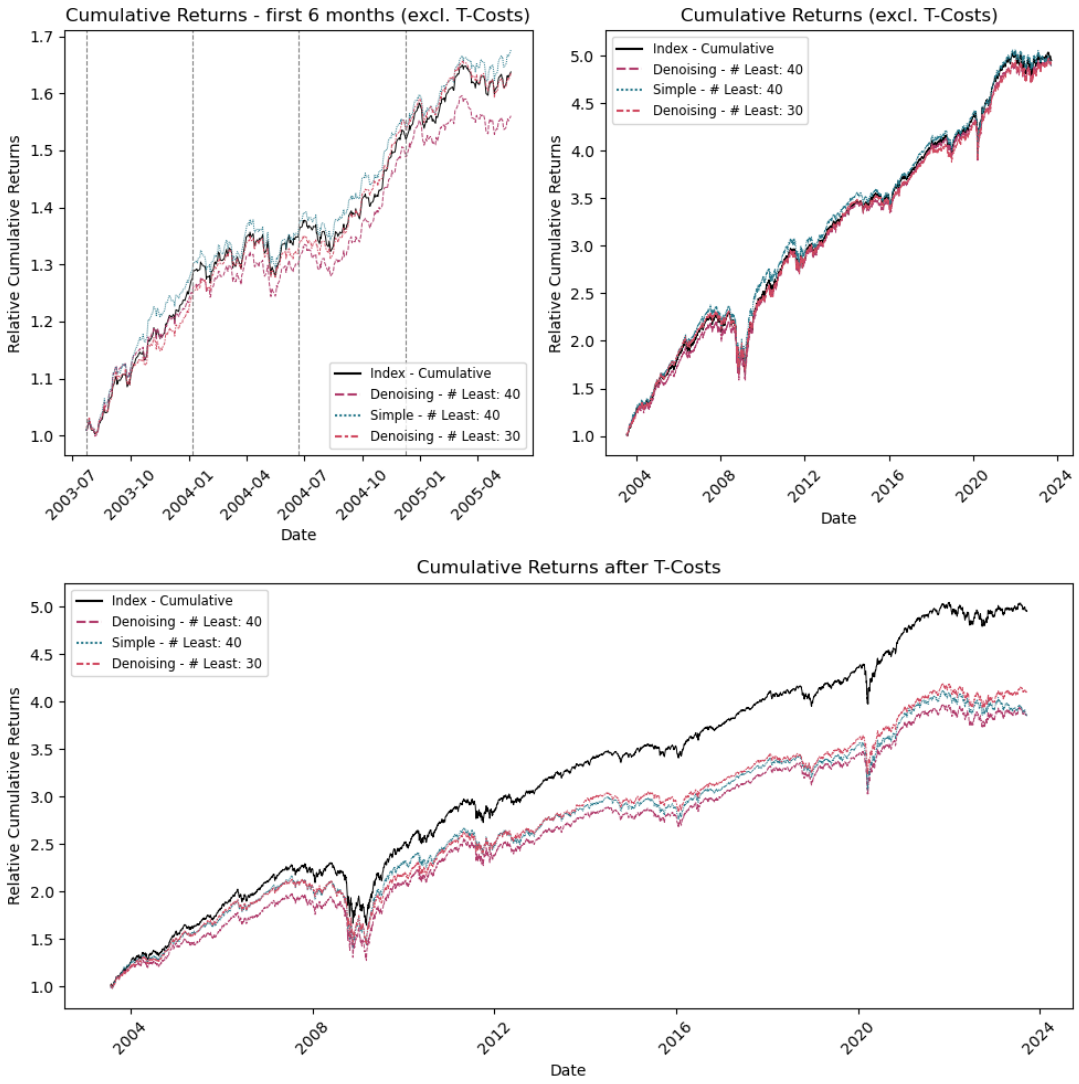


**FIGURE 3.3**

**TABLE 3.4**
Drawdowns

| | Simple AE | Sparse AE | Denoising AE |
|---|---|---|---|
| # Least: 10 | 1.062089 | 0.728466 | 0.379735 |
| # Least: 20 | 0.951157 | 1.697477 | 0.621112 |
| # Least: 30 | 1.141753 | 4.654329 | 0.923161 |
| # Least: 40 | 0.859494 | 1.396487 | 1.034617 |
| # Least: 50 | 0.962166 | 1.037279 | 1.068316 |

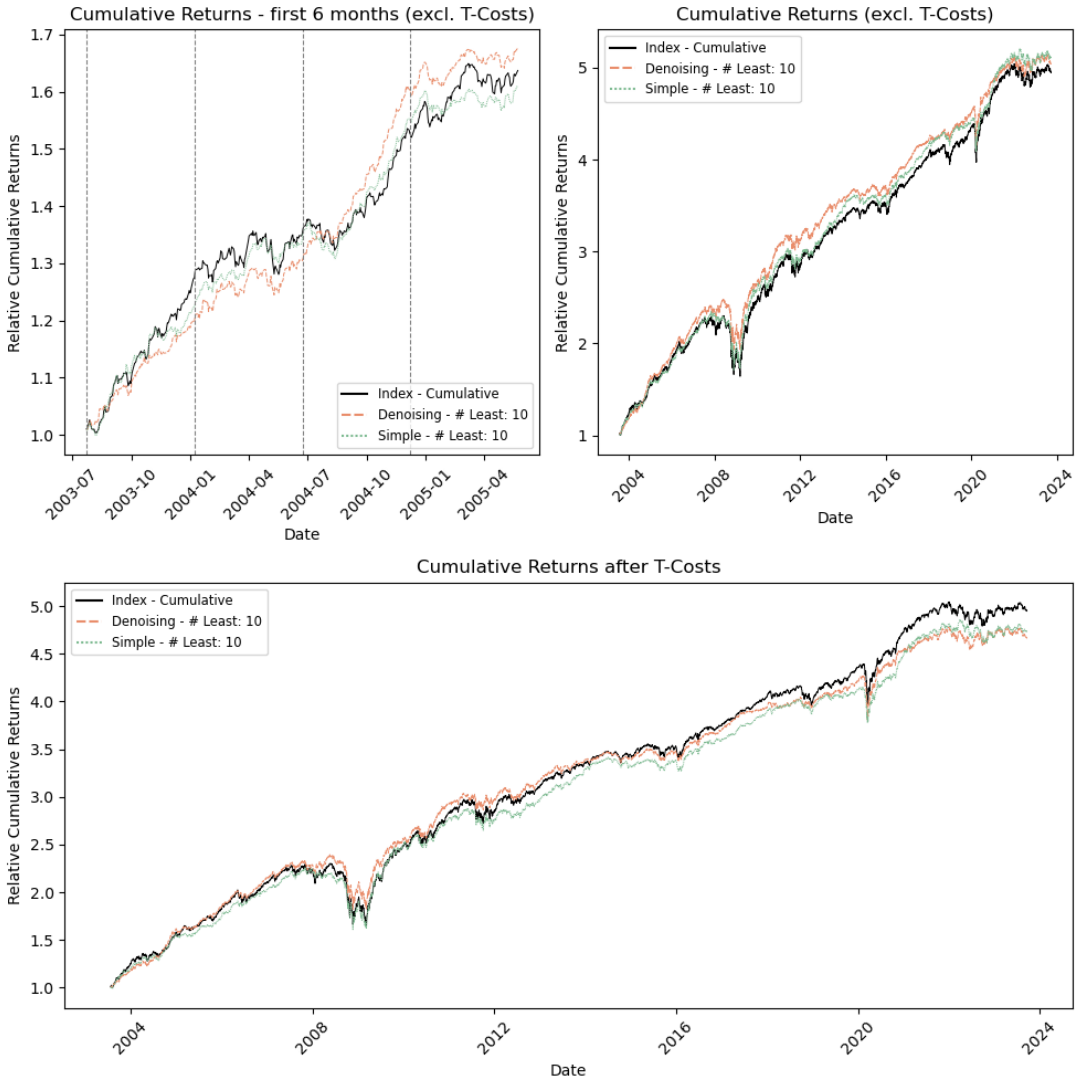


FIGURE 3.4

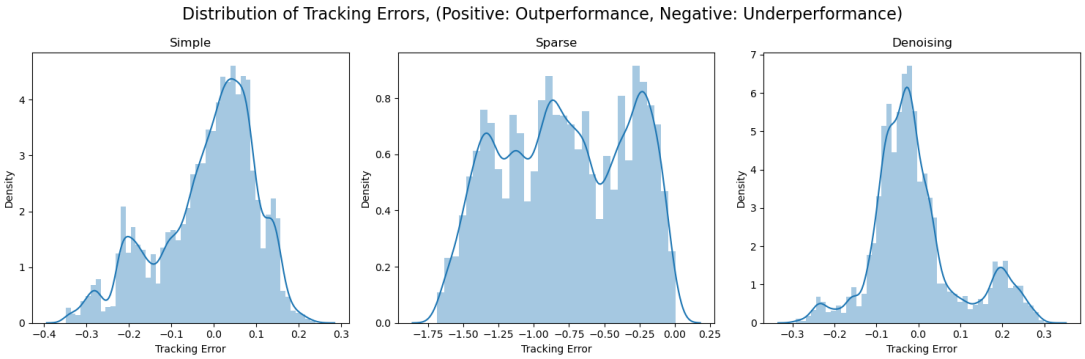


FIGURE 3.5

# CHAPTER 4

# CONCLUSION

The two methods show very interesting results, on the one hand the benchmark method is very easy to explain as is with standard linear regressions. The weights are the best fit for the given data points, with a lasso parameter and a positive restriction. The results before transaction costs seemed very promising, however, this was quickly contradicted after including transaction costs.

On the other hand, we have a very complex and hard to understand method in the Auto Encoder. This results in excellent performance, almost always outperforming the benchmark method. However, this comes at the cost of large computational efforts at every rebalancing date and difficult explainability. The methods also require large amounts of data, leading to long breaks between rebalancing dates. Finally, to address explainability in this report there has been an attempt at making this choice more understandable, which leads to a choice of assets for volatility and assets for the trend.

We can conclude that, when properly tuned, Autoencoders (AE) can significantly outperform Lasso regression, reducing tracking error by an order of magnitude. However, it's important to note that Lasso is a fully automated method that doesn't require users to define any parameters. In contrast, AE requires users to specify the number of features they want to use for tracking. Furthermore, as we anticipated and observed, when transitioning across volatility regimes, AE may require more features to maintain the same tracking performance. This means that AE should be closely monitored and adjusted potentially during each rebalancing period.

## 4.1 FUTURE WORK

We have been able to identify some questions that are important to answer but were infeasible with the computational power at hand.

The first questions revolves around the performance of the autoencoder when cross validating the bottleneck, the number of most and least common features, the rebalancing date all while penalizing large changes in transaction costs. Another important question lies in understanding the performance of the sparse Auto Encoder. We believed this would perform well, however, against our expectations this method performed the worst. We theorize that this was caused by a lower than optimal number of training epochs. However, we were unable to increase the number of epochs without making the computation infeasible.

Finally, there are some open optimizations that may be possible with the benchmark approach, such as including penatly parameters for transaction costs. While at first this was attempted, the implementation was not achievable and we were forced to abandon the approach. Since this was identified as one of the painpoints in the benchmark approach it would be great to test such a framework in the future.

# BIBLIOGRAPHY


Wu, Lan, Yuehan Yang and Hanzhong Liu (2014). 'Nonnegative-lasso and application in index tracking'. In: *Computational Statistics & Data Analysis* 70, pp. 116–126. ISSN: 0167-9473. DOI: https://doi.org/10.1016/j.csda.2013.08.012. URL: https://www.sciencedirect.com/science/article/pii/S0167947313003095.

Zhang, Chi et al. (Sept. 2020). 'Stock-Index Tracking Optimization Using Auto-Encoders'. In: *Frontiers in Physics* 8, p. 388. DOI: 10.3389/fphy.2020.00388.

*Prospectus of MSCI World Index (USD)* (Nov. 2023). URL: https://www.msci.com/documents/10199/178e6643-6ae6-47b9-82be-e1fc565ededb.

*Historical Performance of MSCI World Index (USD)* (Nov. 2023). URL: https://curvo.eu/backtest/en/market-index/msci-world?currency=usd.

Ledoit, Olivier and Michael Wolf (Oct. 2022). *Markowitz portfolios under transaction costs*. ECON - Working Papers 420. Department of Economics - University of Zurich. URL: https://ideas.repec.org/p/zur/econwp/420.html.