Stock-Index Tracking Using Autoencoders

MFE 230T-2 Project

Philip Spalding
Prarthana Shetty
Kunal Chakraborty
Pankaj Kumar
Karan Desai

October 9, 2021

Abstract

In "Stock-Index Tracking Optimization Using Auto-Encoders" (Zhang, Liang, Lyu, & Fang, 2020) (the "Paper"), the authors evaluate the ability of various autoencoders to identify portfolio constituents that are able to track a broad equity index. The methodology, originally proposed by (Heaton, Polson, & Witte, 2017), uses an autoencoder to determine the amount of "communal information" between each stock in the index and the broad market; then, the 10 stocks with the highest and X stocks with the lowest communal information are included in the tracking portfolio in an attempt to replicate the index using a small subset of the total index constituents. We evaluate the author's methodology and attempt to replicate the results of the Paper using the S&P 500 index and several self constructed indices as test cases. While the authors conclude that the autoencoder-based strategies are generally superior to conventional strategies, we find that this result varies based on the index being tracked and the relative performance of existing, simple baselines.

1. Introduction

Portfolio management can be divided into two main schools of thought: active management and passive management. Since the 2007-2008 financial crisis and with the advent of low cost ETFs and robo-advisors, there has been a huge influx of funds to passive strategies. Investors, however, cannot directly invest in market indices; instead, they must invest in portfolios that are designed to track the index of interest. Index tracking can be accomplished using either full replication, in which all index constituents are included in the portfolio, or optimization techniques to replicate the index with only a subset of the index constituents. Full replication is ideal when the index has a small number of constituents, and it ensures that tracking error is minimized compared to other methodologies. However, full replication strategies are expensive to maintain, rebalance, and manage since the portfolio contains all constituents in the target index with the same weights as the index. Optimization techniques, on the other hand, use historical data to identify constituents and weights that replicate the portfolio using only a few constituents. While this approach might lead to an increase in tracking error as compared to the full replication portfolio, it reduces transaction, implementation, and management costs. The autoencoder approach proposed in the Paper falls under the umbrella of optimization methods.

At present, widely used market techniques to replicate portfolios include approaches such as market-value ranking, weight ranking, liquidity ranking, correlation coefficient ranking, random sampling, stratified sampling, and genetic algorithms. (Zhang et al., 2020) proposes that these existing techniques fail to adequately use the historical data about constituent stocks, target indexes, and the correlations between them, and, therefore, new approaches need to be developed.

2. Background and Related Work

With the recent advances in deep learning, it is not surprising that the financial industry has explored ways to use such techniques to tackle traditional financial problems, including portfolio construction. The methodology used by the Paper was originally proposed by (Heaton et al., 2017), who demonstrates the efficacy of the approach using the IBB biotechnology index. (Heaton et al., 2017) measures the Euclidean distance between the original returns and the reconstructed returns from a trained autoencoder for each stock in the index. This distance is a measure of the communal information each stock shares with the broad market, and the 10 stocks with the highest and X stocks with the lowest communal information are selected for inclusion in the tracking portfolio. (Ouyang, Zhang, & Yan, 2019) subsequently expand on the original framework by including a dynamic asset-weight calculation and test their approach using the Hang Seng Index (HSI). However, the approach proposed by (Ouyang et al., 2019) can lead to negative weights, contrary to traditional index tracking methodologies that generally have a long only constraint. (Kim & Kim, 2020) argue that such an asset selection criterion is artificial. They modify it by constructing an autoencoder in such a way that the deepest hidden layer has only one node, a proxy for the market index, and measuring the similarity of this latent representation to individual asset returns. (Zhang et al., 2020) disagree with this approach, however, reasoning that this method abstracts away from economic intuition even though it does a good job at replicating the index of interest.

3. Autoencoders

Before detailing the stock selection methodology, we briefly provide an overview of the 6 autoencoders used by the (Zhang et al., 2020) and the additional 2 autoencoders we use for our evaluation of the proposed methodology. An autoencoder is a neural network that is trained to copy its input. It consists of two parts: an encoder function $\boldsymbol{h} = f(\boldsymbol{x})$ and a decoder function $g(\boldsymbol{h}) = \boldsymbol{x}$. The networks is constrained in some way so that it cannot simply learn to perfectly copy its input, which would not be very useful, and we hope that during the training process $\boldsymbol{h}$, the (usually) lower dimension, latent representation of the input, takes on useful properties. In many cases, the loss used to train the autoencoder is simply the Euclidean distance (i.e., the l^2 norm) between the input and reconstructed output:

$$L(\boldsymbol{x}) = \sum_i \|x_i - g(f(x_i))\|_2$$

The output of an auto-encoder is a lossy version of the input as there is generally some information loss during the encoding-decoding process. Finally, the reconstructed input or latent representation can be used for various downstream tasks, in this case, portfolio construction. There are various ways to constrain an autoencoder, which has lead to multiple popular architectures. We consider the following:

1. **Undercomplete Autoencoders**: Undercomplete autoencoders are vanilla autoencoders with one hidden layer that contains a fewer number of neurons than the input layer. The

primary objective of these kind of architectures is to map the input space to a latent representation in lower dimensions.

2. **Spare Autoencoders**: Sparse autoencoders are regularized autoencoders where, generally, the hidden layer has the same number of neurons as the input layers, but some neurons are not activated. This activation of neurons is data driven and the architecture is particularly useful because rather than limiting the capacity of the network, we can improve the generalization capability by enforcing some penalty. A common approach to encourage sparsity is to use an l^1 penalty on the latent space representation, which is the approach we use:

$$L(\boldsymbol{x}) = \sum_i \|x_i - g(f(x_i))\|_2 + \lambda \|\boldsymbol{h}\|_1$$

3. **Stacked or Deep Autoencoders**: This architecture increases the number of layers and reduces the number of neurons as the encoder gets deeper and increases the number of neurons as the decoder gets deeper until the original input dimension is obtained. Figure 1 illustrates the stacked autoencoder architecture used by (Zhang et al., 2020). A stacked autoencoder should be able to better capture non-linear relationships within the input data.

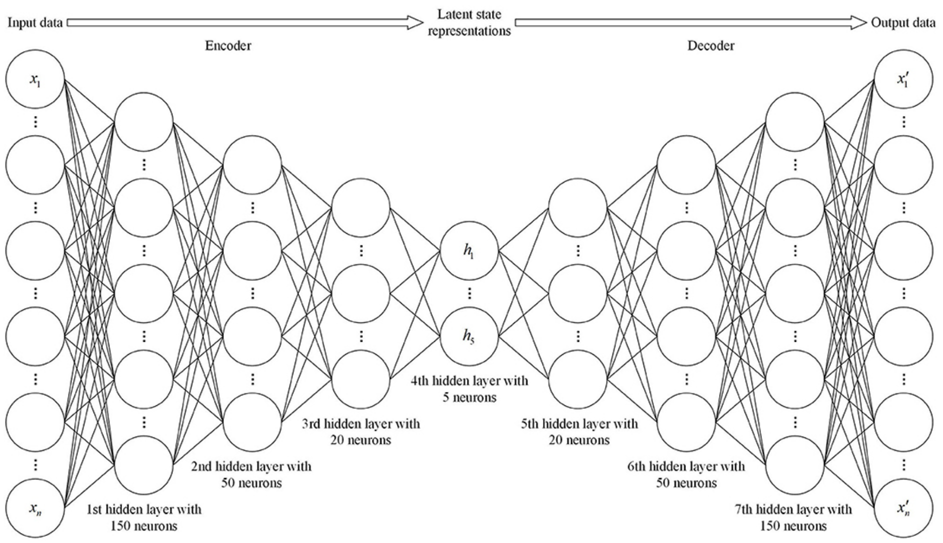
Input data
Latent state representations
Output data
Encoder
Decoder
x_1
x_n
h_1
h_5
x'_1
x'_n
1st hidden layer with 150 neurons
2nd hidden layer with 50 neurons
3rd hidden layer with 20 neurons
4th hidden layer with 5 neurons
5th hidden layer with 20 neurons
6th hidden layer with 50 neurons
7th hidden layer with 150 neurons

Figure 1: Stacked autoencoder architecture used in the Paper.

4. **Denoising Autoencoders**: This architecture is particularly useful in identifying robust relationships with the input data. The architecture is the same as a stacked autoencoder, but the input data is modified to contain some noise and the reconstructed data is measured against the actual data. This regularization mechanism encourages the network to learn robust, "noise free" relationships in the dataset. The Paper uses a denoising autoencoder that adds independent, multivariate gaussian noise to the input; we use the same approach.

5. **Contractive Autoencoders**: Contractive autoencoders introduce regularization that encourage the derivatives of the encoder to be small:

$$L(\boldsymbol{x}) = \sum_i \|x_i - g(f(x_i))\|_2 + \lambda \left\| \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} \right\|_F^2$$

Unfortunately, penalizing the Jacobian of the encoder is relatively expensive, and we found that training this architecture takes over 5x as long as the other architectures considered. Because the authors do not find that the contractive autoencoder performs particularly better than any of the other autoencoders, we exclude this architecture from our analyses.

6. **Variation Autoencoders (VAE)**: While VAEs are similar to the other autoencoders discussed, they are more commonly used as generative models. The authors find that VAEs perform worse than the other autoencoders considered, and, therefore, we exclude this architecture from our analyses.

7. **Overcomplete Autoencoders**: Unlike undercomplete autoencoders that restrict $\boldsymbol{h}$ to be a lower dimension than the input, overcomplete autoencoders actually make the dimension of $\boldsymbol{h}$ larger than the input. While the authors only examine autoencoders that restrict the capacity of the model so that the autoencoder does not learn to simply copy the input, we also experimented with overcomplete autoencoders as they are relatively inexpensive to train and test.

8. **Linear Undercomplete Autoencoders**: When the encoder-decoder is linear and mean squared error (i.e., Euclidean distance) is used as the loss function, it has been shown that undercomplete autoencoders learn "to span the same subspace as PCA. In this case, an autoencoder trained to perform the copying task has learned the principal subspace of the training data as a side-effect" (Goodfellow, Bengio, & Courville, 2016). Given this autoencoders's similarity to PCA, we include it in our experiments.

4. Methodology

Once an autoenocoder architecture has been selected, the stock selection and index tracking methodologies proceeds as follows:

1. First, the autoencoder is trained to reconstruct the cross-section of daily stock returns for the index constituents during the training period, which is usually the previous 3 to 5-years of data.

2. Once the autoencoder is trained, the entire training set is reconstructed, and the amount of information loss is calculated for each stock in the dataset. Information loss is calculated as the Euclidean distance between the original time series of returns and the reconstructed time series of returns:
$$L_j = \sum_i \|x_j^{(i)} - {x'}_j^{(i)}\|$$
where L_j represents the information loss of the jth constituent and x and x' are the original and reconstructed returns, respectively. This is done on a stock-by-stock basis so that each stock has a single score for the period. The lower (higher) the information loss, the higher (lower) the communal information the stock has. The authors define communal information as, "the amount of information that they share with the stock index market" (Zhang et al., 2020).

3. Stocks are ranked by their communal information and the stocks with the 10 highest communal information scores (lowest information loss) and the X lowest communal information scores (most information loss) are included in the tracking portfolio. The authors use a fixed

number of the most-communal stocks plus a variable number of the least-communal stocks to construct a tracking portfolio as, “is not beneficial for improving index tracking performance to include too many stocks contributing the same information” (Zhang et al., 2020). Figure 2 presents an example of high and low communal information stocks from one of our initial experiments.

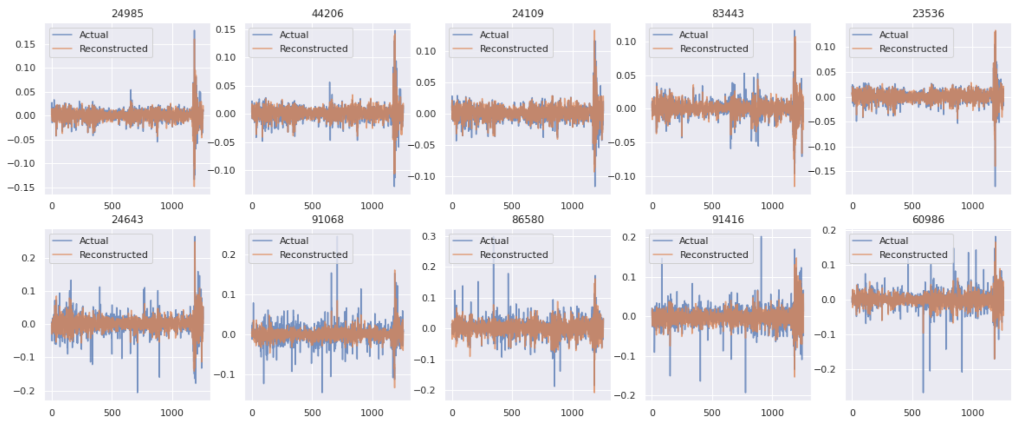

Figure 2: Original and reconstructed return series for the five stocks with the highest communal information (top) and five stocks with the lowest communal information (bottom) from our initial experiments with an undercomplete autoencoder and the S&P 500. We can clearly see that high communal information stocks overlap much more with their original series (in blue) and low communal information stocks have notable deviations.

4. Once stocks are selected for inclusion in the tracking portfolio, the weights for the portfolio are determined by finding weights that minimize the tracking error between the tracking portfolio and the actual index during the training period. The problem is formulated as the following quadratic program:

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \|\boldsymbol{r_i} - \boldsymbol{R_p}\boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{w}\|$$
$$s.t. \quad \boldsymbol{w} \geq 0, \quad \mathbf{1}^T\boldsymbol{w} = 1$$

where $\boldsymbol{r_i}$ is a vector of index returns during the training period and $\boldsymbol{R_p}$ is a matrix of constituent returns over the training period. These weights are then used to formulate the tracking portfolio for the following 6-month test period. During the test period, the weights remain unchanged, which corresponds to daily rebalancing of the portfolio in practice. While other test period rebalancing strategies could be formulated, we use the same procedure as the authors for simplicity.

5. To create the portfolio for the next 6-month period, the training and test sets are shifted 6-months forward and the entire process is repeated.

5. Stock and Index Data

In the Paper, the authors evaluate their autoencoder based index tracking methodology on the CSI 300 index. The CSI 300 is a market capitalization weighted index designed to replicate the performance of the top 300 stocks traded on the Shanghai Stock Exchange and the Shenzhen Stock

Exchange. Unfortunately, we were not able to obtain the CSI 300 index data required to reproduce the authors results. Instead, we evaluated the methodology using the S&P 500. The S&P 500 and CSI 300 share many characteristics: both are market capitalization weighted, both include a similar number of constituents, and both are designed to track the performance of some of the most prominent stocks in their respective markets. Given the difficulty of obtaining data on historical index constituents, we also created 9 indices of our own using the data for the S&P 500 constituents in order to test the robustness of the methodology to various index constructions.

We obtained 11-years - 2010 through 2020 - of historical price data and return data on the S&P500 and its constituents. While the index includes about 500 constituents at any point in time, constituents are added or subtracted from the index over time. Over 750 different stocks have been included in the index over the 11-year period, and we mapped each stock to the days during which it is included in the index. Historical data for the S&P500 and its constituents was obtained from the Center for Research in Security Prices (CRSP). This data is very high quality, so little data cleaning is required. Less than 0.1% of days have missing return information, and such missing data is generally the result of stock splits on these days. Given how few days have missing entries, missing values were filled in with 0 as daily returns are also generally centered around 0.

We explore the construction of indices along 2 different dimensions. Namely:

Index Weighting Method:
- Market capitalization weighted
- Price weighted
- Equal weighed

Index Size (Number of Constituents):
- 50 stocks
- 250 stocks
- 500 stocks

Stocks were selected randomly for inclusion in each index as we found that this minimized the correlation between the constructed indices compared to other approaches. The three weighting methodologies and index sizes are also representative of other popular indices. For example, the 50 and 250 constituent price weighted indices would be comparable to the Dow Jones Industrial Average, a price weighted index of 30 prominent U.S. companies, and the Nikkei 225, a price weighted index of 225 companies included on the Tokyo Stock Exchange, respectively.

In order to test the autoencoder based stock selection methodology, the authors utilize rolling train-test periods. Specifically, 4-years of data is used to train the autoencoder and determine the portfolio weights, and the next 6-month period is then used as the test period. Both the train and test sets are then shifted 6-months and the process is repeated. This process corresponds to a 6-month rebalancing frequency of both the index constituents and weights. Figure 3 (originally from the Paper) outlines the rolling train-test methodology. Because we have a slightly longer time series of returns than (Zhang et al., 2020), we use a 5-year train period followed by a 6-month test period.

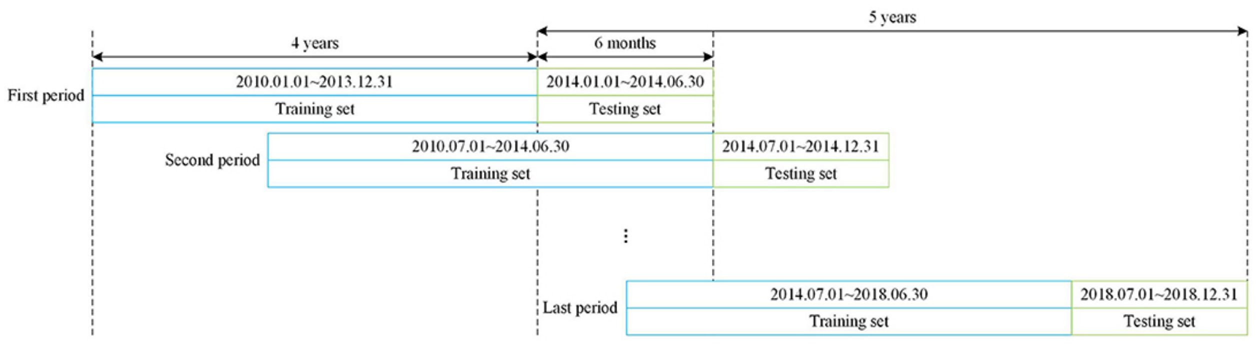

Figure 3: Rolling train-test periods utilized in the Paper.

6. Empirical Results

In order to evaluate the efficacy of the autoencoder based stock selection methodology, the authors use two baseline stock selection methodologies for comparison: weight ranking and market-value (i.e., market capitalization) ranking. Unfortunately, the authors do not provide details for either baseline methodology. However, the weight ranking methodology appears to select the n stocks with the highest index weights at the beginning of each time period for inclusion in the tracking portfolio, and the market-value ranking selects the n stocks with the largest market capitalization at the beginning of each time period. The authors vary the value of n to see how the number of stocks in the tracking portfolio effects tracking error for each methodology.

For market capitalization weighted indices, the weight ranking and market-value ranking approaches should be generally the same with the exception of adjustments made to market capitalization (e.g., free float adjustments). We were not able to pull data for the exact index weights over time for the S&P 500, so we use the following baselines for each type of index tested.

1. **Market Capitalization Weighted Indices**: Select n stocks with the highest market capitalization at the beginning of each rebalance period. Note that this is the same baseline that will be used for the S&P 500 index as well.
2. **Price Weighted Indices**: Select n stocks with the highest prices at the beginning of each rebalance period. This corresponds to selecting the highest index weighted stocks for a price weighted index, without consideration of any idiosyncratic adjustments.
3. **Equal Weighted Indices**: Unlike the market capitalization and price weighted indices, the equal weighted index does not have an obvious baseline selection criteria. Therefore, for these indices, we will use the market capitalization ranking methodology.

To measure the performance of our autoencoder based tracking portfolio, we calculate the tracking error between the portfolio and the index, calculated as:

$$e = \sqrt{\mathrm{Var}(r_p - r_i)}$$

where r_p is the portoflio returns and r_i is the index returns over all 6-month test periods. For each autoencoder and for each 5-year train/6-month test period, we construct multiple portfolios of different sizes, varying the number of low communal information stocks to evaluate how tracking error changes with the number of constituents. We limit the size of each tracking portfolio to approximately 25% of the index's total size; e.g., for the S&P 500, we look at portfolios with between 20 and 130 stocks. In addition to experimenting with different autoencoders, we also compare all results to the baseline portfolio for each index, as described above.

Figure 4 presents our results for the S&P 500. Each autoencoder uses the best hyperparameters determined via a grid search. However, the baseline (the dashed blue line) outperforms all the autoencoders used, achieving the lowest tracking error for any size portfolio.

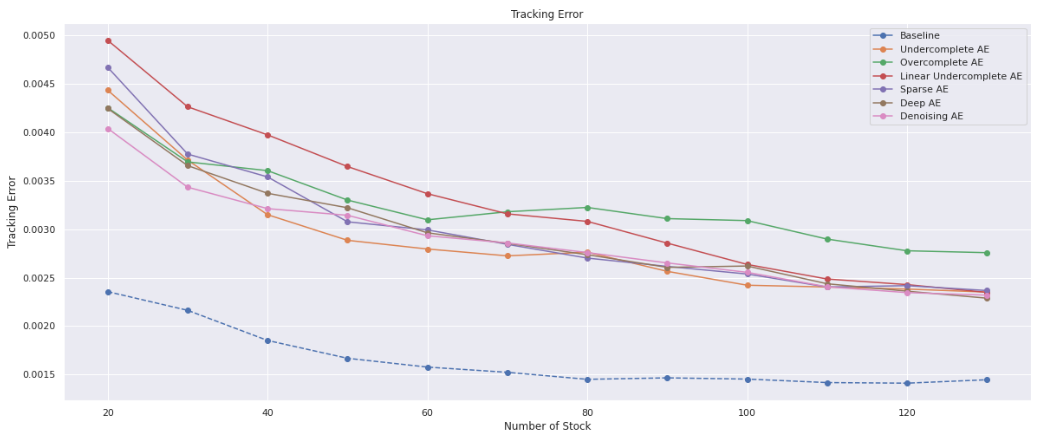

Figure 4: Results for the S&P 500 index. Tracking error for each size portfolio (number of stocks) is presented for each autoencoder and the baseline.

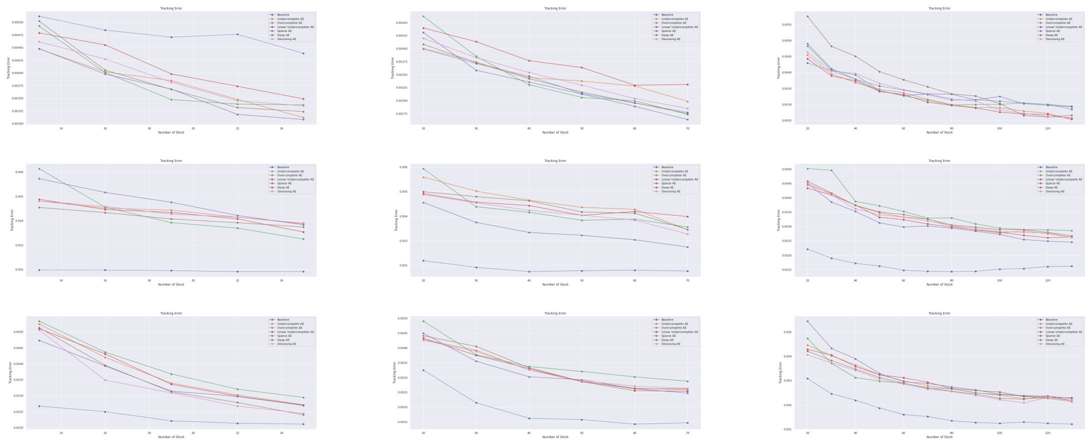

Figure 5: Results for the 9 constructed indices: (Top row) equally weighted indices; (middle row) market capitalization weighted indices; (bottom row) price weighted indices; (first column) 50 constituent indices; (second column) 250 constituent indices; (third column) 500 constituent indices.

Figure 5 presents our results for the 9 constructed indices. Again, each autoencoder uses the best hyperparameters determined via a grid search. While all of the autoencoders outperform the baseline for the equally weighted, 50 constituent index (EW50), only some of the autoencoders outperform the baseline for the EW250 and EW500 indices. Further, none of the autoencoders outperform the baselines for the market capitalization or price weighted indices, similar to the results for the S&P 500. These baselines for the market capitalization and price weighted indices, by design, capture the stocks with the highest index weights. They also have very low tracking error - below 0.003 for almost every size portfolio. In comparison, the author's CSI 300 baselines (and autoencoders) do not achieve tracking error below 0.003 until the portfolio contains at least 50 stocks, and the smaller portfolios have tracking error between 0.004 and 0.008. The Equal weighted indices, however, do not have such obvious baselines - all constituents have the same weight so it is more difficult to identify constituents that represent the index. For these indices, the autoencoders are able to compete with the baselines.

We explored various reasons for why the author's autoencoders were able to outperform the baselines, at least for the smaller portfolios. We did extensive hyperparameter searches, doing a separate grid search for each autoencoder and each index. However, the results for the various parameters explored were relatively consistent, suggesting that the approach is at least somewhat robust to the autoencoder architecture selected. We also tested whether the random initializations and inherent randomness in the training process were a cause of the under-performance. While we saw some variation in the results for each of the different random seeds used, the variation was not large enough to explain the differences in the results. Overall, it simply looks like our baselines are significantly more competitive than the baselines used by the authors for the CSI 300. Figure 6 compares our average autoencoder results for the S&P500 and its baseline next to the authors average autoencoder results for the CSI 300 and their best performing baseline. Our autoencoders achieve similar or better tracking error than the author's autoencoders. However, our baseline achieves notably better tracking error.

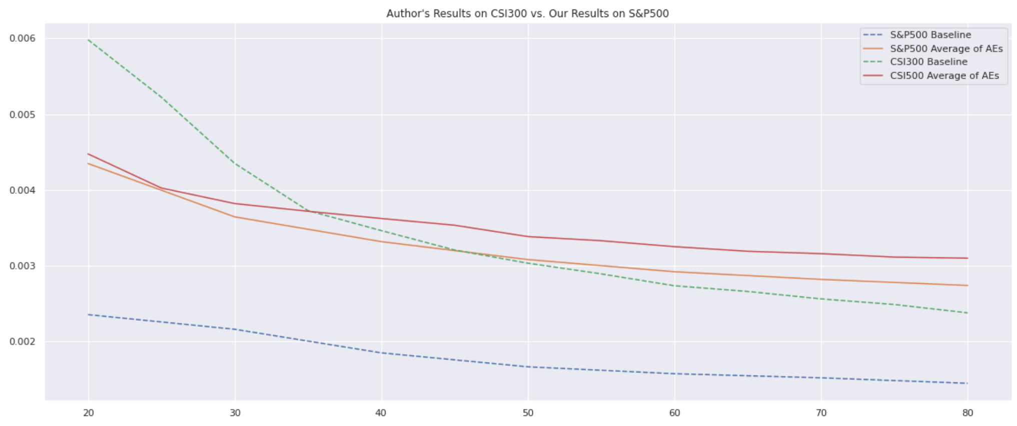

Figure 6: Comparison between our results for the S&P 500 and the author's results for the CSI 300. Solid lines represent the average performance across all autoencoders tested. Dashed lines represent the baselines used.

Even if the autoencoder approach is not always able to outperform the baseline, it is possible that the portfolios constructed have other desirable properties. While the authors do not discuss this aspect of the approach, we decided to look at how many of the tracking portfolio constituents change for each 6-month period when the portfolio is updated. Figure 7 plots the percentage of the constituents in the portfolio that are replaced each period for the S&P500 for various portfolio sizes and for the undercomplete autoencoder and the denoising autoencoder. Note that, while similar, this is not the same measure as portfolio turnover. These graphs capture the number of constituent changes, but not how much the weights change for constituents that remain in the portfolio.

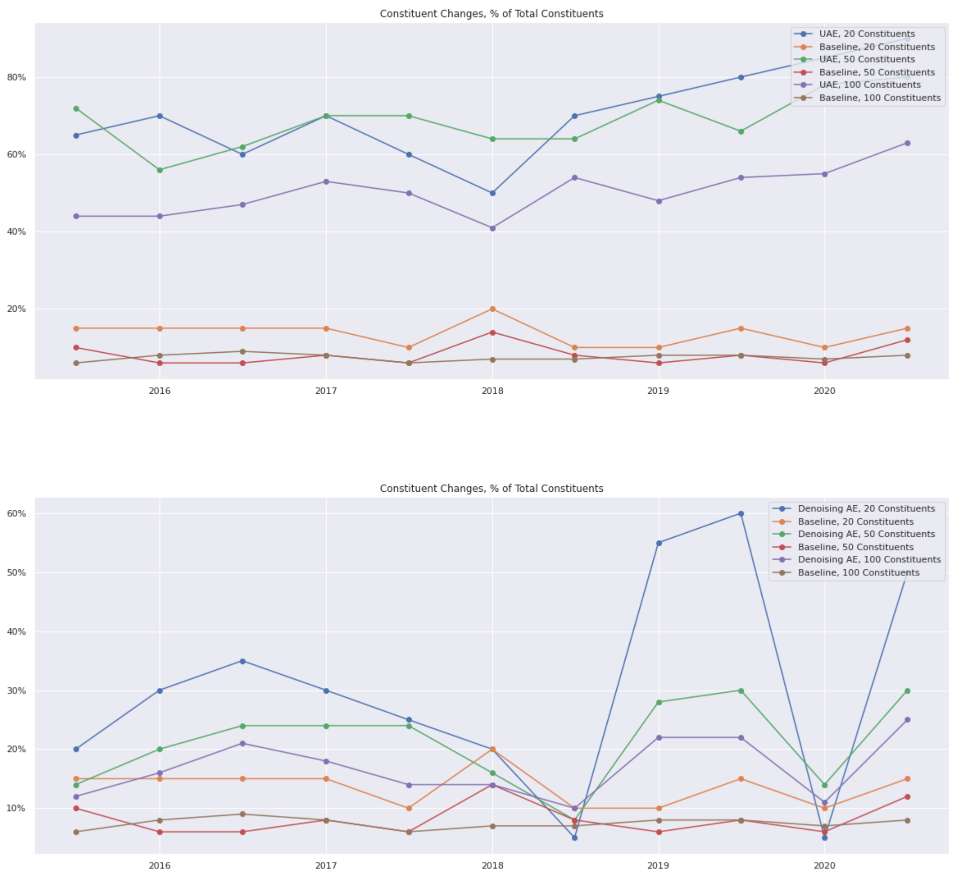

Figure 7: Changes in the portfolio constituents for each 6-month period.

Interestingly, the deeper autoencoders (the denoising autoencoders uses a deep architecture) have relatively few constituent changes compared to their shallow counterparts. It is possible that the deep autoencoders are able to learn a more stable representation of returns as we slide the training period forward over time. Regardless, the autoencoder approach results in more constituent changes than the baselines. This is of real practical concern as it will result in higher transaction costs and a larger number of stocks to monitor over time.

7. Conclusion

While we do see that the author's autoencoder based stock selection methodology is effective in some cases, the efficacy varies by index and the relative performance of the corresponding simple baseline approach. Further, we did not evaluate whether the autoencoders perform better than a cardinality constrained optimization approach, which is common in practice. The autoencoder approach may, however, be particularly suited to cases where we either do not have an obvious baseline or simply do not know the underlying constituents' weights in the index. By inspecting the autoencoders' reconstructed return series, we can measure how similar the stocks are to the broader universe of stocks in our dataset, providing some measure of interpretability of the results, which can be difficult for deep neural network applications.

There also may be some practical concerns to the approach. The autoencoders require some hyperparameter tuning, which can be very time consuming given a separate model must be trained for 6-month period. Further, we have found that the autoencoders result in more variable portfolios, with more constituents being replaced each 6-month period, which would, in practice, result in higher transaction costs and a larger number of stocks to monitor over time.

The main concern with the proposed methodology is that there is not a direct link between stock selection and the target index. While we restrict our universe of potential constituents to stocks included in the index, the communal information measure used does not actually consider the target index itself (i.e., the target index's returns). While autoencoders may be uniquely suited to identify complex, non-linear relationships between index constituents, adjusting the methodology to include a more direct connection to the target index itself may result in lower tracking error.

References

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning.* MIT Press. (`http://www.deeplearningbook.org`)

Heaton, J. B., Polson, N. G., & Witte, J. H. (2017). Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry*, *33*(1), 3-12. doi: https://doi.org/10.1002/asmb.2209

Kim, S., & Kim, S. (2020). Index tracking through deep latent representation learning. *Quantitative Finance*, *20*(4), 639-652. doi: 10.1080/14697688.2019.1683599

Ouyang, H., Zhang, X., & Yan, H. (2019). Index tracking based on deep neural network. *Cognitive Systems Research*, *57*, 107-114.

Zhang, C., Liang, S., Lyu, F., & Fang, L. (2020). Stock-index tracking optimization using auto-encoders. *Frontiers in Physics*, *8*, 388. doi: 10.3389/fphy.2020.00388