Learning a Multi-Factor Covariance Matrix for Portfolio Optimisation

Rory Creedon*[1]

Abstract
Whilst mean-variance optimisation finds the optimal in-sample portfolio, it typically exhibits poor out-of-sample performance. This paper proposes a novel methodology to incorporate multiple factors (such as historical profit ratios) into constructing a covariance matrix for mean-variance optimisation, inspired by Autowarp [1]. An autoencoder is first trained on features relevant to each asset. Secondly, an automatically selected distance metric is used to measure the distance between the latent representations for each pair of assets. Finally, this distance matrix is standardised and multiplied by the standard deviations of returns to create a covariance matrix. Using a selection of the constituent stocks in the S&P 500, this method can lead to much improved out-of-sample performance compared to other covariance matrices, such as the sample covariance matrix, covariance shrinkage and the exponentially weighted covariance matrix.

**Department of Computer Science, University College London*
[1]**Email**: rory.creedon.22@ucl.ac.uk

Contents

1. Introduction

Quantitative techniques for efficient portfolio optimisation are a topic that investors and researchers have long been interested in. Of the techniques proposed, the most classic strategy is mean-variance optimisation, first introduced by Markowitz in a seminal paper in 1952 [2]. When constructing a portfolio from a basket of assets, the mean-variance optimisation problem is to find the weights for each asset that minimises risk for a given portfolio return (or equivalently to maximise portfolio returns for a given level of risk).

Whilst mean-variance portfolios results in the optimal portfolio (the one that maximises return over risk) in-sample, they often exhibit poor out-of-sample performance [3]. One of the key reasons for this poor performance is that mean-variance optimisation is highly sensitive to any estimation errors in the two inputs - expected returns and the covariance matrix. Of the two inputs, the covariance matrix is often viewed as the major source of estimation error [4].

Most covariance matrices used in mean-variance optimisation, such as the sample covariance, covariance shrinkage and the exponentially weighted covariance matrix, are based solely on historical asset returns. Relying solely on historical asset returns to assess similarity (intrinsically through correlation) means that epistemic sources of risk, such as different economic climates and regulatory regimes, which may affect assets heterogeneously, are not taken into account. This is in part an unavoidable problem with attempting to estimate future covariance, but incorporating multiple related variables into estimating the covariance matrix may limit how much epistemic sources of risk are unaccounted for.

In this paper, a novel methodology for constructing a multi-factor covariance matrix using the distance between the latent space from an autoencoder is proposed. When tested on long-only portfolios constructed from the constituent stocks within the S&P 500, its performance is a significant improvement over traditional covariance matrices[1].

2. Literature Review

2.1 Mean-variance portfolios

The mean-variance optimisation problem [2] can be formulated as a constrained optimisation problem for long-only portfolios, which is laid out in equation 1. The variance $w^T \Sigma w$ is minimised, subject to given returns. The weights for each asset are represented by w, the expected returns for each asset are μ, the covariance matrix is Σ, and the given portfolio returns (measured by variance) are r^*. As only long-only optimisation is considered in this paper, there is a final constraint that all weights must be greater than or equal to 0.

$$\begin{aligned} \min_w \; & w^T \Sigma w \\ \text{subject to } & w^T \mu \geq r^* \\ & w^T \mathbf{1} = 1 \\ & w_i \geq 0 \end{aligned} \tag{1}$$

Equation 1 shows risk being minimised, subject to portfolio returns of at least r^*, where short positions are not allowed ($w_i \geq 0$) and the total sum of all weights must be equal to 1.

This problem can also be transformed to find the tangency portfolio, i.e., the portfolio that maximises the Sharpe ratio [5] (excess return divided by standard deviation). The constrained optimisation problem to maximise the Sharpe ratio is shown in equation 2, where r_f is the risk-free rate.

$$\begin{aligned} \min_w \; & \frac{w^T \mu - r_f}{(w^T \Sigma w)^{1/2}} \\ \text{subject to } & w^T \mathbf{1} = 1 \\ & w_i \geq 0 \end{aligned} \tag{2}$$

In this paper, two variants of mean-variance optimisation will be considered. Firstly, maximising the Sharpe ratio (as laid out in equation 2) and minimising variance, regardless of portfolio returns (equivalent to removing the first constraint in equation 1). This is because there is a body of literature showing that minimum variance portfolios can outperform mean-variance portfolios out-of-sample [6].

2.2 Alternative portfolio optimisation methods

Mean-variance portfolios have been shown to perform poorly out-of-sample in several studies [3, 7, 8]. The portfolios produced by classic mean-variance strategies are often unstable and underdiversified, where small changes in the inputs (the expected returns and covariance matrix) lead to large changes in the optimised portfolio. This led to Michaud coining the term "estimation error maximizers" for mean-variance portfolios [8], as these portfolios tend to favour assets that have higher estimate returns, negative covariances and low variances.

One relevant response to the poor performance of mean-variance portfolios has been the use of alternative covariance matrices. The sample covariance matrix often has extreme values, and mean-variance optimisation is very sensitive to these extreme values. As such, one approach is the use of shrinkage, a form of regularisation which results in extreme values moving towards more central values. This method has been shown to achieve superior performance to the sample covariance matrix [9].

Another approach involves placing higher weighting on more recent observations when calculating the covariance matrix [10, 11]. An EWMA (exponentially weighted moving average) model is used to calculate the covariance matrix, which also has been shown to outperform using the sample covariance matrix [12].

Both previous amendments to the covariance matrix only involved the use of asset returns. As mentioned in section 1, incorporating multiple factors into portfolio optimisation may help to reduce exposure to sources of risk that affect groups of assets that are may contain otherwise uncorrelated assets, such as regulatory changes or increases in commodity prices. Multiple variables can be used to calculate the covariance matrix in multi-factor models. First introduced with the single-index factor model [13] and later extended to incorporate multiple factors with the Fama-French three-factor model [14], multi-factor models first regress asset returns on the factors as shown in equation 3, where there are K factors F_n and r_i is the

[1] Accompanying reproducible code is available at https://github.com/rorycreedon/portfolio_metric_learning

return of each asset i.

$$r_i = \alpha_i + \sum_{n=1}^{K} \beta_{in} F_n + \varepsilon_i \tag{3}$$

The covariance matrix is then computed as shown in equation 4:

$$\text{cov}(i,j) = \sum_{m=1}^{K} \sum_{n=1}^{K} \beta_{im}\beta_{jn}\text{cov}(F_m, F_n) + \delta_{ij}\text{var}(\varepsilon_i)$$
$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Multi-factor models have been shown to sometimes outperform the sample covariance matrix [15]. , but this approach still relies heavily on asset returns, as asset returns are regressed in the first step. The approach proposed in this paper incorporates multiple factors, whilst relying less heavily on historical asset returns.

There exist other prominent methodologies that do not rely as heavily on the mean-variance method, namely Black-Litterman Allocation [16] and Hierarchical Risk Parity [17]. Black-Litterman Allocation takes a Bayesian approach to portfolio optimisation, incorporating a prior estimate of returns with 'views' on a subset of asset returns. As this paper is focused on solely quantitative techniques for portfolio optimisation, Black-Litterman Allocation has not been included. Hierarchical Risk Parity is another recent method, which optimises weights clustering assets through a distance matrix and then finding the minimum variance portfolio. There is evidence to suggest that the use of Hierarchical Risk Parity leads to more robust portfolios.

3. Methodology

The methodology proposed is heavily inspired by Autowarp [1], a two-step process to learn a distance metric between time series. First, an autoencoder is trained to minimise the reconstruction error. Secondly, the distance between the latent representation of each time series is calculated. The distance metric used is found by the Autowarp algorithm, which finds the optimal distance metric from a family of warping distances. After Autowarp, a distance matrix is constructed using the optimal warping distance for each pair of assets. This distance matrix is then standardised and converted into a correlation matrix with maximum correlation 1 and minimum correlation C. It is then multiplied by the standard deviations of each pair of assets to form the covariance matrix. The end-to-end methodology is also shown in Figure 1.

3.1 Autoencoders

An autoencoder is a model that consists of an encoder network, which maps the input into a lower dimensional space (often referred to as the latent representation), and a decoder network, which maps the latent representation into the output, which is typically meant to be a reconstruction of the input. A graphical intuition can be seen at the top of Figure 1 (before optimising α, β and γ).

Autoencoders are used in the proposed methodology because condensing the input data into a latent representation and then reconstructing the input from the latent representation only preserves the features within the input data that are the most important. In this case, they are used for both reducing noise and extracting the most useful features from the input data, which is then used to create a covariance matrix which is less biased by noise.

Generally, both the encoder and decoder networks are neural networks. A more formal mathematical formulation follows. As time series data is used, let the input series be $X = (x_1, x_2, ..., x_T)$ where $x_t \in \mathbb{R}^n$ (there are n time series) with T periods.

The encoder network maps the input $X \in \mathbb{R}^{n \times t}$ to a latent representation $X \in \mathbb{R}^{n \times s}$ and the decoder network maps the latent representation $X \in \mathbb{R}^{n \times s}$ to the output $X \in \mathbb{R}^{n \times t}$ where $s < n$. The encoder network can take the form of any neural network. In this paper, three variants are used: a 2-layer fully connected network, a 2-layer convolutional neural network and 3 layer network with 1 fully connected and 2 convolutional layers.

A fully connected layer has a weights matrix W and a bias vector b. The output of the layer z can be calculated as shown in equation 5, where x is the input. Weights W and biases b are updated every epoch when losses are backpropagated.

$$z = Wx + b \tag{5}$$

In a convolutional layer, a convolution operation is applied to the input x using a kernel h. The kernel, generally a small vector or matrix of weights, depends on the dimension of the convolution operation. It slides across the input x, and at each position, the kernel performs an element-wise multiplication with the corresponding segment of the input, generating a component of the output z.

To ensure a comprehensive representation of the input data, including its edges, padding is frequently utilised. This technique appends extra data, typically zeros, to the

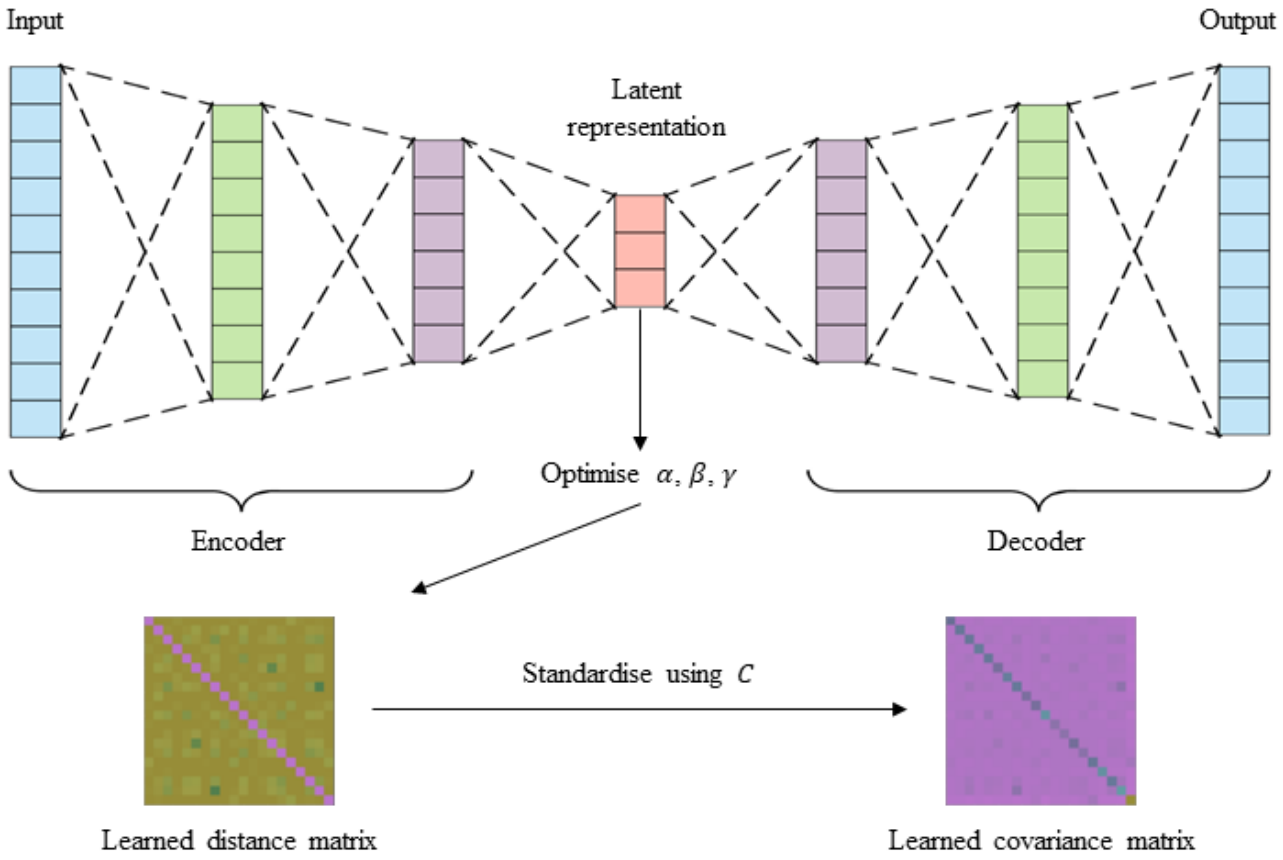

Figure 1. Model overview

peripheries of the input. A padding size of p denotes the addition of p zeros to each edge of the input data. To reduce dimensionality, a stride is commonly employed, particularly in encoder networks that aim to reduce the dimensionality of the input data. This approach means that the kernel does not slide across every data point in the input, advancing only one data point at a time. With a stride size of s, the kernel progresses by s units each time it conducts an element-wise multiplication with the relevant portion of the input. In the autoencoders in this paper, one-dimensional convolution operations are performed due to the utilisation of time series data. Equation 6 provides a mathematical definition of a 1-dimensional convolution operation.

$$z_n = (x \circledast h)n = b_n + \sum k = 0^{M-1} x_{n-k} * h_k \qquad (6)$$

The output z is computed as the sum of element-wise multiplications between the input x and the kernel h. In this case, the kernel's length is M since it is one-dimensional. A bias term b is also incorporated into the output z. In a 1-dimensional convolution, the kernel is 'flipped', as illustrated in Equation 6. The kernel's index spans from $k = 0$ to $k = M - 1$, while the input data's index ranges from $k = n$ to $k = n - (M - 1)$. This reveals that the kernel's index increases as the input data's index decreases. A visual representation of a 1-dimensional convolutional operation can be observed in Figure 2. During the convolution operation $(x \circledast h)$, the kernel h is 'flipped' as it traverses the input data x.

Between layers, regardless of their type, a non-linear activation function is applied. The primary motivation for using activation functions is to introduce non-linearities into the network, enabling it to learn non-linear relationships between inputs and outputs. This pertains to the relationships between the inputs and the latent representation, as well as between the latent representation and the output. Without non-linear activation functions, the network's ability to model complex patterns would be substantially restricted, even in shallower architectures.

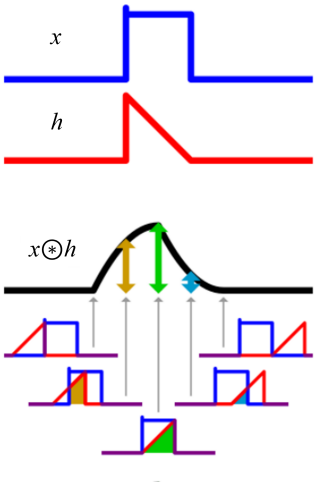

Figure 2. 1D Convolution operation

In the autoencoders employed in this paper, the rectified linear activation function (ReLU) is utilized. The ReLU function introduces non-linearity and offers computational efficiency, making it a popular choice for a wide range of network depths. The formula for the ReLU function is presented in Equation 7.

$$\text{ReLU}(x) = \max(0, x) \tag{7}$$

Autoencoders are usually trained using gradient descent. After weights and biases have been initialised and the training data has been passed through both the encoder and decoder networks, the reconstruction loss is calculated. In the autoencoders used in this paper, the mean squared error is used to calculate the difference between the input data and reconstruction data. Gradients are then backpropagated through the encoder and decoder networks and weights are biases are updated through an optimiser. In this case, the optimiser Adam has been used, largely because of its adaptive learning rate, meaning that it calculates different learning rates for different parameters. These steps are repeated many times (in this paper 20 epochs have been used).

It is worth noting that more complex neural networks, such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Unit (GRU) networks are often used in time series, but are not used in this paper. This is because the features in this paper have different frequencies. The price and returns data are daily, whereas the financial ratios used are quarterly. As such the financial ratios have been 'stretched' into daily data, which causes recurrent neural networks such as LSTMs and GRUs to not perform as well when minimising the reconstruction loss.

Additionally, a common extension to autoencoders is variational autoencoders (VAEs), which take a Bayesian approach [18]. These are similar to autoencoders, but the crucial difference is that the latent representation is sampled from a distribution. These are not used in this paper as it is more difficult to learn an appropriate distance metric (which is detailed in section 3.2) between distributions as opposed to the numeric latent representation from an autoencoder.

3.2 Distance metrics

After the autoencoder has been trained and run on data in the training set, the Autowarp algorithm is used to find the distance metric to measure the distance between the latent representations for each pair of assets.

When comparing different time series, often the Euclidian distance is not a meaningful distance metric for time series. There are several reasons why this is the case - for example misalignment in the time series and temporal distortions (i.e., time shifts, stretching, compression, etc.). To account for this, a distance metric such as Dynamic Time Warping [19] and Edit Distance for Real Sequences [20] is used. The Autowarp algorithm, which is used in this paper, learns a distance metric from a family of warping distances.

A warping path is a way to connect two sets of points (i.e., two different time series), which will be referred to as trajectories. The warping path p between trajectories t^A and t^B, where p is a tuple of indexes. The first entry must be an index from t^A and the second must be an index from t^B. The warping path p must satisfy the following conditions:

- *Boundary conditions* - the path must start with the first values of each trajectory and end with the last value of each trajectory. If t^A is of length n and t^B is of length m, then $p_0 = (t^A_0, t^b_0)$ and $p_L = (t^A_n, t^b_m)$, where L is the length of the warping path.
- *Valid steps* - Starting at $p_k = (t^A_i, t^b_j)$, the next tuple in the warping path has to be one the following $p_{k+1} \in \{(t^A_{i+1}, t^B_j), (t^A_{i+1}, t^B_{j+1}), (t^A_i, t^B_{j+1})\}$.

A warping distance d is a function that maps a warping path p for trajectories t^A and t^B to a real number. The warping distance d is defined as the minimum total cost over the warping path, where the cost function c is a function of two tuples within the warping path. The

formula for the warping distance is shown in equation 8 where P is the set of all possible paths between t^A and t^B

$$d(t^A, t^B) = \min_{p \in P} \sum_{i=1}^{L} c(p_{i-1}, p) \tag{8}$$

The cost function takes three parameters $\alpha, \gamma, \varepsilon$ and is defined in equation 9, where $\sigma(x,y) = y \cdot \tanh(x/y)$. This cost function is generalised such that other warping distances can be specified using it. For example, if $\alpha = 0.5, \gamma = 0, \varepsilon = 1$, it is equivalent to Dynamic Time Warping.

$$c\big((t^A_{i_1}, t^B_{j_2}), (t^A_{i_2}, t^B_{j_2})\big) = \begin{cases} \sigma(\|t^A_{i_2} - t^B_{j_2}\|, \frac{\varepsilon}{1-\varepsilon}) & \text{if } i_2 > i_1, j_2 > j_1 \\ \frac{\alpha}{1-\alpha} \cdot \sigma(\|t^A_{i_2} - t^B_{j_2}\|, \frac{\varepsilon}{1-\varepsilon}) + \gamma & \text{if } i_2 = i_1 \text{ or } j_2 = j_1 \end{cases} \tag{9}$$

To select optimal values of $\alpha, \gamma, \varepsilon$, batched gradient descent is used. The loss function used is an adapted version of BetaCV - an index typically used to evaluate the quality of clustering assignments given a fixed distance between points. However, cluster assignments are not available (but distances are). To proxy for cluster assignments, if two points have a Euclidian distance below a threshold level δ in their latent representation (calculated as a percentile $\bar{p}$ of all Euclidian distances in the latent representation), this is considered to be of the same cluster assignment. The formula for the adapted BetaCV is shown in equation 10, where Z is a normalising constant, T is the number of trajectories (in the case of this paper the number of assets) and h_i and h_j are the latent representations of assets i and j.

$$\hat{\beta} = \frac{\frac{1}{Z}\sum_{i=1}^{T}\sum_{j=1}^{T} d(t^i, t^j)\mathbb{I}[\|h_i - h_j\|_2 \leq \delta]}{\frac{1}{T^2}\sum_{i=1}^{T}\sum_{j=1}^{T} d(t^i, t^j)} \tag{10}$$

The learned distance metric d^* chosen is the one that minimises $\hat{\beta}$. It is minimised through batch gradient descent. In this paper, the Adam optimiser [21] is used for optimisation.

Once the distance metric d^* has been learned, it is used to calculate a distance matrix, where each pair of assets has a scalar value $d^*(t^A, t^B)$.

3.3 Covariance matrices

After Autowarp has been run and a distance matrix has been constructed, it is used to calculate the covariance matrix used in mean-variance optimisation.

In a covariance matrix Σ, typically each element is calculated as follows.

$$\Sigma_{ij} = \rho_{ij}\sigma_i\sigma_j \tag{11}$$

It is the correlation coefficient ρ_{ij} which contains the information on how similar two assets are. Therefore, the correlation coefficient ρ_{ij} is replaced with a scaled version of the learned distance matrix.

As the covariance matrix will contain the covariance between asset i and asset j (with a correlation coefficient of 1), the upper bound of the scaled distance metric $\hat{d}^*_{ij}$ is set at 1. However, the lower bound is less clear. Whilst ρ_{ij} has a minimum of -1, it is unlikely that there are assets in the data (or for the vast majority of real-world datasets) where there are assets that are the opposite of each other. Therefore, after multiplying the distance matrix by -1 (as it represents distance, and the correlation coefficient represents similarity), the minimum value that $\hat{d}^*_{ij}$ is left as a parameter C. Therefore, the learned covariance matrix M is calculated as follows, where d^*_{ij} is the learned distance between asset i and asset j, K is the set of all indices along the first axis and L is the set of all indices along the second axis.

$$M_{ij} = 1 + (C-1)\left(\frac{d^*_{ij} - \min_{k \in K, l \in L} d^*_{kl}}{\max_{k \in K, l \in L} d^*_{kl} - \min_{k \in K, l \in L} d^*_{kl}}\right) \tag{12}$$

The learned covariance matrix M is then used in mean-variance optimisation. The end-to-end process of learning the covariance matrix is shown in Figure 1.

4. Experiments

The learned covariance matrix is tested on the constituent stocks within the S&P 500. Four rolling training, validation and testing sets are defined, where the training set is two years long, the validation set is one year long and the testing set is 18 months long. The stocks used are the constituent stocks at the test/train cutoff date, meaning that if a stock was not in the S&P 500 on 1 September 2021 (the first test/train cutoff date) but was in the S&P 500 on 1 March 2022 (the second test/train cutoff date), it would not be included in the constituent stocks for the first period, but it would be included in the second. This approach is taken to mimic real-world conditions. Investors are not aware of which stocks will enter the S&P 500 before they have entered. To include all stocks at a specific date (for example, the time of

writing) would mean the set of stocks used would be intrinsically biased, as it does not include companies who had a similar market capitalisation to stocks that have entered the S&P 500 but did not enter the S&P 500.

In each test/train/validation period, two experiments are run. Firstly, where the Sharpe ratio is maximised during mean-variance optimisation (because the Sharpe ratio is the primary criterion that is used to evaluate different methods) and where volatility is minimised during-variance optimisation (because, as mentioned in section 2.1, there is evidence to suggest that minimum variance portfolio out-perform out-of-sample).

Whilst there are typically (close to) 500 stocks within the S&P 500 at any time, 500 stocks have not been used in this paper. Firstly, in addition to closing prices of all stocks, several features relevant to each stock are used. The features used in this paper are cash flow divided by operating revenue, current ratio, net assets turnover and profit margin. The features used are all ratios or percentages, to ensure that they are comparable across stocks with different market capitalisations. In addition to these features, the historical price of each stock is used. All features (including price data) are standardised, such that the maximum value is 1 and the minimum value is 0. This data is not available for roughly 200 of the constituent stocks within the S&P 500.

Of the remaining roughly 300 stocks, a third of them are set aside for validation. They are not used in either the train or test sets, in order to avoid overfitting the validation set. These stocks are first trained during the 2-year training period, and the performance of portfolios selected from these stocks is evaluated during the validation period. There are a large number of hyper-parameters to tune, namely the latent size, the hidden size(s), batch size and learning rate in the autoencoders, the batch size, threshold percentile $\bar{p}$, and batch size in Autowarp and the scaling parameter C for constructing a covariance matrix. These are optimised through Bayesian optimisation on the validation set, individually for each period, autoencoder and whether the Sharpe ratio is maximised or volatility is minimised.

5. Results

The average Sharpe ratios in the four 18-month-long test sets are shown in Table 1, with separate columns for when the Sharpe ratio is maximised in the training set and when volatility is minimised in the training set. The results show that autoencoder-based portfolios appear to significantly outperform all other models (except Hierarchical Risk Parity) when maximising the Sharpe ratio in the training period, and autoencoder-based models slightly improve upon all other models when minimising volatility in the training period.

	Average Sharpe Ratio	
Model	**Max Sharpe**	**Min Volatility**
Linear	0.49	**0.81**
CNN	**0.68**	0.68
Linear + CNN	**0.68**	0.68
Covariance	0.33	0.64
Cov. Shrinkage	0.34	0.65
EW Covariance	0.35	0.46
Fama-French	0.27	0.52
HRP	**0.68**	0.68
S&P 500	0.02	0.02

Table 1. Average Sharpe Ratio in the test sets when maximising Sharpe ratio and minimising volatility in the training sets.
Note: The values in bold represent the highest average Sharpe ratio for each task.

Additionally, these results have been broken down to show the Sharpe ratios in each of the test sets, shown in 3. The red bars represent portfolios constructed using the autoencoder-based covariance matrices introduced in this paper, the blue bars represent the portfolios constructed using benchmark covariance matrices, the green bars represent portfolios made using the Fama-French 3 factors, the yellow bars represent Hierarchical Risk Parity portfolios and the grey represents the S&P 500.

The results show that the autoencoder-based portfolios perform well in the first two testing period, outperforming all other portfolios when both maximising the Sharpe ratio and minimising volatility. In particular, they significantly outperform the benchmark portfolios in the second testing period. Whilst performance is not as strong in the second two testing periods, at least one of the autoencoder models outperforms all benchmark models in the fourth testing period.

It is worth noting that given that only a subset of the constituent stocks in the S&P 500 was used, comparisons to the performance of the S&P 500 in Figures 3 may not necessarily be fair, as the subset of stocks used may have exhibited different performance levels to the whole of the S&P 500.

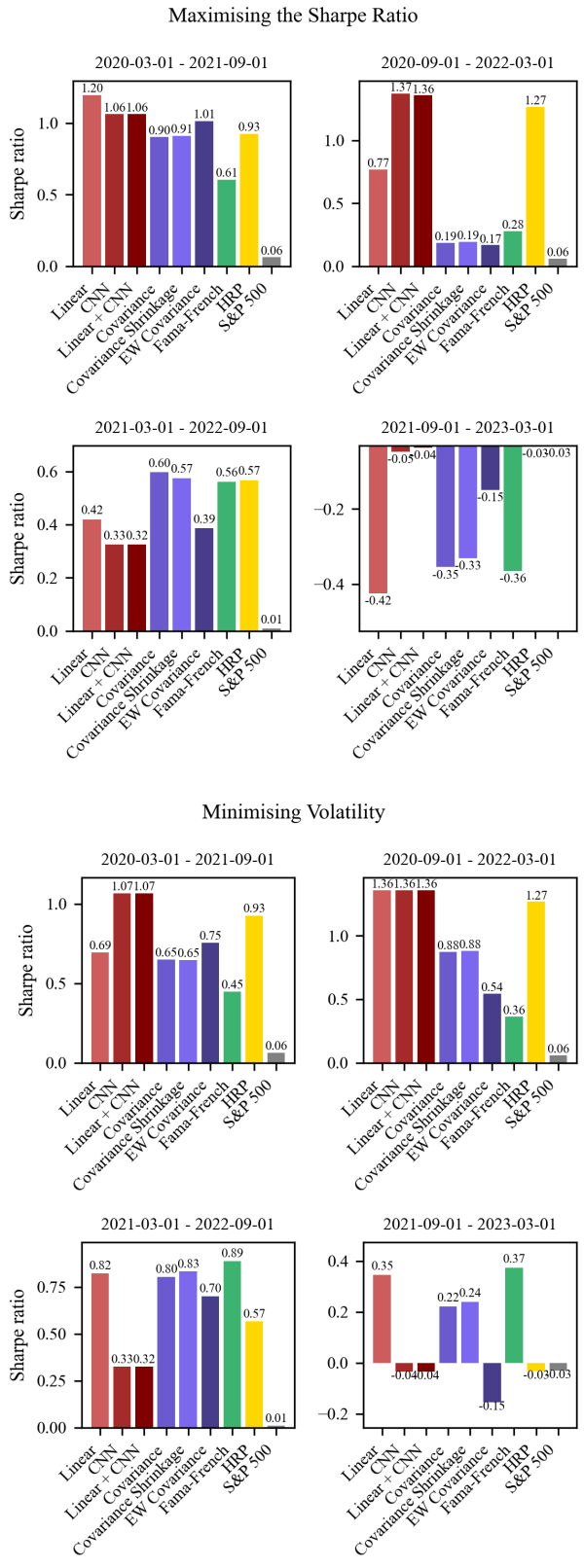

Figure 3. Test set Sharpe ratios from maximising Sharpe ratio and minimising volatility in train sets

In addition to the Sharpe ratio charts, Figure 4 shows the performance of the portfolios for one selected period, 2020-09-01 to 2022-03-01 when maximising the Sharpe ratio. In the background of the plot, all the individual stocks that were available for selection are plotted. Aligning with Figure 3, the CNN and Linear + CNN portfolios, which have the highest Sharpe ratio, also exhibit the highest returns, both with a return of 36% over the 18 months and a maximum drawdown of 10%.

6. Discussion

The results show that portfolios using autoencoder-based covariance matrices can outperform both portfolios using traditional covariance matrices and portfolios using existing methods for multi-factor covariance matrices. Whilst autoencoder-based portfolios do always outperform other methods, Table 1, shows that they have a higher average Sharpe ratio.

The experiments in this paper have been conducted in a way that keeps as much consistent across the different models, only changing the covariance matrix (if applicable). This has been done in order to effectively compare different covariance matrices. The experiments have not, however, been conducted to construct the optimal portfolio. For example, in all models, the mean of historical returns was used as the expected returns, which is unlikely to be a sensible assumption. Using a separate model to forecast expected returns (or potentially incorporating analysts' forecasts) would likely yield higher Sharpe ratios across the board.

The autoencoder-based models incorporate multiple factors as well as the price of each stock, such as profit ratios and the current ratio. The performance of autoencoder-based models depends heavily on the factors that are included - less relevant features could decrease performance and more relevant features could improve performance. However, the features used in this report may often not be available for different stocks or different types of assets. The features used in this report are all specific to equities, meaning that the experiments in this paper cannot be extended to include different asset classes (such as bonds, real estate and other indices). Nonetheless, a different set of features that is common across different asset classes can be used to construct autoencoder-based portfolios across different asset classes.

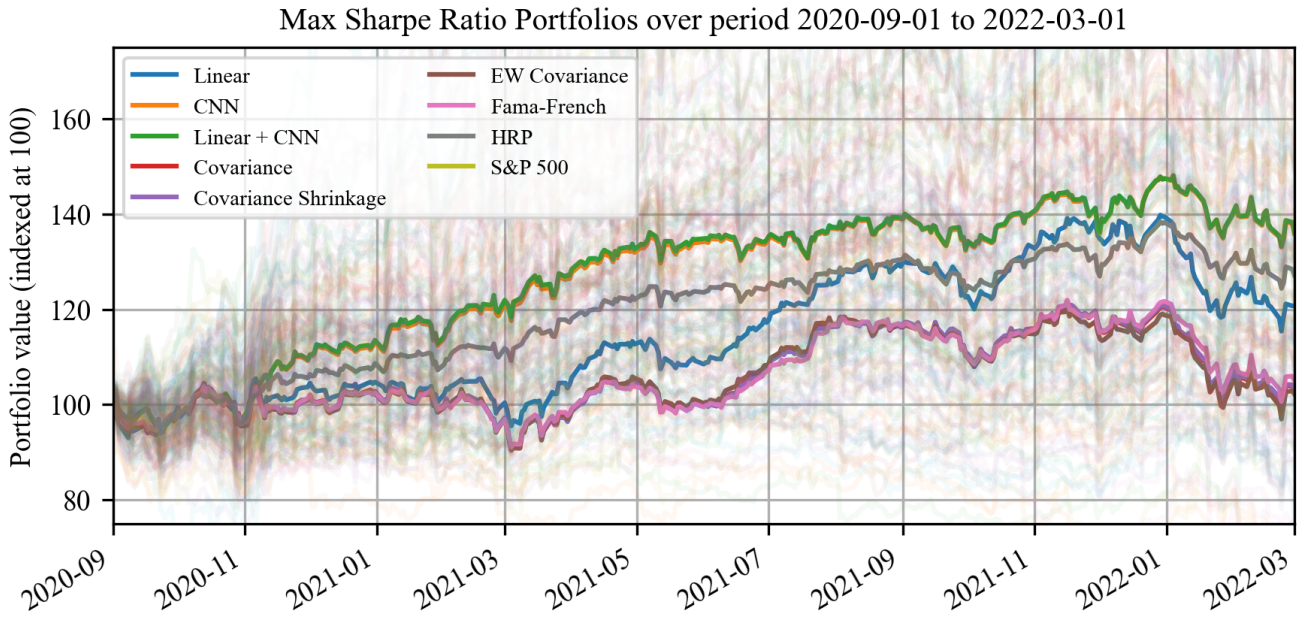

Figure 4. Performance of portfolios constructed by maximising the Sharpe Ratio, evaluated over the period 2020-09-01 to 2022-03-01

Note: The value of the portfolio has been indexed at 100 at the beginning of the test set.

7. Conclusion

The key contribution of this paper is a novel methodology to incorporate multiple different factors into a covariance matrix for portfolio optimisation, which appears to be able to exhibit superior out-of-sample performance compared to traditional covariance matrices and traditional multi-factor covariance matrices when tested on the constituent stocks in the S&P 500. When tested using financial ratios as factors, the average Sharpe ratios for the autoencoder-based models introduced in this paper are comparable to state-of-the-art methods such as Hierarchical Risk Parity.

The performance of multi-factor models is highly dependent on the factors used. Ledoit and Wolf described finding the optimal number and type of factors for multi-factors models as "as much an art as it is a science" [9]. Within this description, the methodology introduced in this paper provides a better toolkit to implement the "art" of multi-factor portfolio optimisation.

The autoencoder-based methodology introduced in this paper opens up several avenues for further research. Firstly, the methodology could be further tested with different pools of assets, different training and test set lengths and more backtests could be implemented. This would add robustness to the results presented in this paper.

Secondly, different autoencoder architectures could be explored. Following the finding that increasing autoencoder complexity does not lead to a discernable difference in the distance metric found [1], fully connected and convolutional neural networks with limited depth were used in this paper. Experimenting with different architectures could lead to increased performance. In particular, when all factors chosen have the same frequency (e.g., daily, quarterly, etc.), the use of more complex neural networks commonly used in time series, such as LSTM and GRU networks could be explored.

Finally, this approach can be extended to a multi-period problem, where the portfolio is dynamically updated every day [22]. The inclusion of the standardisation parameter C, which determines the lower bound on how correlated the assets are and affects how many stocks are included in the portfolio may introduce added complexity to this problem as fewer recommend stocks in a portfolio can lead to reduced transaction costs.

References

[1] Abubakar Abid and James Zou. Autowarp: Learning a Warping Distance from Unlabeled Time Series Using Sequence Autoencoders. In *Proceedings of*

the 32nd International Conference on Neural Information Processing Systems, 2018.

[2] Harry Markowitz. Portfolio Selection. *Journal of Finance*, 7(1):77–91, 1952.

[3] Victor DeMiguel, Lorenzo Garlappi, and Raman Uppal. Optimal Versus Naive Diversification: How Inefficient is the 1/N Portfolio Strategy? *The Review of Financial Studies*, 22(5):1915–1953, 2007.

[4] Richard Michaud, David Esch, and Robert Michaud. Deconstructing Black-Litterman: How to Get the Portfolio You Already Knew You Wanted. *Journal of Investment Management*, 11:6–20, 2013.

[5] William Sharpe. Capital Asset Prices: A Theory of Market Equilibrium Under Conditions of Risk. *The Journal of Finance*, 19(3):425–442, 1964.

[6] Ziemowit Bednarek and Pratish Patel. Understanding the outperformance of the minimum variance portfolio. *Finance Research Letters*, 24:175–178, 2018.

[7] Vijay Chopra and William Ziemba. The Effect of Errors in Means, Variances, and Covariances on Optimal Portfolio Choice. *The Journal of Portfolio Management*, 19(2):6–11, 1993.

[8] Richard Michaud. The Markowitz Optimization Enigma: Is 'Optimized' Optimal? *Financial Analysts Journal*, 45(1):31–42, 1989.

[9] Olivier Ledoit and Michael Wolf. Honey, I Shrunk the Sample Covariance Matrix. *The Journal of Portfolio Management*, 30(4):110–119, 2004.

[10] Jacques Longerstaey. RiskMetrics - Technical Document. *RiskMetrics*, 1996.

[11] Gilles Zumbach. The RiskMetrics 2006 methodology. *RiskMetrics*, 2006.

[12] Benedikt Himberta, Julia Kapraunb, and Markus Rudolfa. A Study of Improved Covariance Matrix Estimators for Low and Diversified Volatility Portfolio Strategies. 2017.

[13] William Sharpe. A Simplified Model for Portfolio Analysis. *Management Science*, 9(2):277–293, 1963.

[14] Eugene Fama and Kenneth French. Common Risk Factors in the Returns on Stocks and Bonds. *Journal of Financial Economics*, 33(1):3–56, 1993.

[15] Louis Chan, Jason Karceski, and Josef Lakonishok. On Portfolio Optimization: Forecasting Covariances and Choosing the Risk Model. *The Review of Financial Studies*, 12(5):937–974, 2015.

[16] Fischer Black and Robert Litterman. Global Portfolio optimization. *Financial Analysts Journal*, 48(5):28–43, 1992.

[17] Marcos Lopez De Prado. Building Diversified Portfolios that Outperform Out of Sample. *The Journal of Portfolio Management*, 42(4):59–69, 2016.

[18] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv*, 2013.

[19] Hiroaki Sakoe and Seibi Chiba. Dynamic Programming Algorithm Optimization for Spoken Word Recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

[20] Lei Chen, M Tamer Özsu, and Vincent Oria. Robust and Fast Similarity Search for Moving Object Trajectories. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, pages 491–502, 2005.

[21] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv*, 2017.

[22] Stephen Boyd, Enzo Busseti, Steve Diamond, Ronald N Kahn, Kwangmoo Koh, Peter Nystrup, Jan Speth, et al. Multi-period trading via convex optimization. *Foundations and Trends® in Optimization*, 3(1):1–76, 2017.