Asymmetric Autoencoders for Factor-Based Covariance Matrix Estimation

Kevin Huynh*
kevin.huynh@unibas.ch
University of Basel
Basel, Switzerland

Gregor Lenhard*
gregor.lenhard@unibas.ch
University of Basel
Basel, Switzerland

ABSTRACT

Estimating high dimensional covariance matrices for portfolio optimization is challenging because the number of parameters to be estimated grows quadratically in the number of assets. When the matrix dimension exceeds the sample size, the sample covariance matrix becomes singular. A possible solution is to impose a (latent) factor structure for the cross-section of asset returns as in the popular capital asset pricing model. Recent research suggests dimension reduction techniques to estimate the factors in a data-driven fashion. We present an asymmetric autoencoder neural network-based estimator that incorporates the factor structure in its architecture and jointly estimates the factors and their loadings. We test our method against well established dimension reduction techniques from the literature and compare them to observable factors as benchmark in an empirical experiment using stock returns of the past five decades. Results show that the proposed estimator is very competitive, as it significantly outperforms the benchmark across most scenarios. Analyzing the loadings, we find that the constructed factors are related to the stocks' sector classification.

CCS CONCEPTS

• **Computing methodologies** → **Factor analysis**; **Neural networks**; • **Mathematics of computing** → **Dimensionality reduction**; Time series analysis; • **Applied computing** → *Economics*.

KEYWORDS

Covariance matrix estimation, Portfolio optimization, Autoencoder neural networks, Factor modeling

ACM Reference Format:
Kevin Huynh and Gregor Lenhard. 2022. Asymmetric Autoencoders for Factor-Based Covariance Matrix Estimation. In *3rd ACM International Conference on AI in Finance (ICAIF '22), November 2–4, 2022, New York, NY, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3533271.3561715

1 INTRODUCTION

Covariance matrix estimation is a crucial part of portfolio allocation problems, more specifically mean-variance optimization by Markowitz [32]. As the number of assets increases above the number of observed returns, the typical sample covariance estimator becomes singular if no additional structure is imposed. In the economics and finance literature, two popular solutions have been proposed to tackle this problem. One of those methods is to shrink the sample covariance matrix towards a full rank shrinkage target. For a comprehensive summary, see Ledoit and Wolf [27]. A different approach is to reduce the dimensionality of the estimation problem using factor models which was theoretically analyzed in Fan et al. [11]: They showed that imposing a factor structure is beneficial for estimating the inverse of the covariance matrix. Factor models are also widely used in asset pricing and are theoretically motivated by e.g. the arbitrage pricing theory (APT) [36, 37] and the capital asset pricing model (CAPM) [29, 38]. They either employ well established factors, e.g. those of Fama and French [9], or estimate data-driven (latent) factors using unsupervised machine learning methods, e.g. in Lettau and Pelger [28] and Gu et al. [15]. After constructing the factors, the factor loadings then are typically estimated in a second step using ordinary least squares (OLS).

The aim of this paper is to provide a dimension reduction procedure suited for factor modeling using autoencoder neural networks. We propose an asymmetric autoencoder with a single linear decoding layer and thereby simplify the usual 'two-step' procedure by jointly estimating the factors and their corresponding loadings in a single step. The proposed procedure yields highly competitive minimum variance portfolios with respect to the standard deviation that are robust across different scenarios compared to alternative methods from the literature. Analyzing the sectors of the stocks that correspond to the highest factor loadings, we are able to interpret the constructed latent factors.

Asymmetric autoencoders were introduced by Majumdar and Tripathi [31], arguing that the asymmetric structure reduces the number of parameters to be estimated and potential overfitting. Applications include Feng et al. [13] and Wang et al. [44] with a focus on time series forecasting. Andreini et al. [2] also propose a neural network with asymmetric structure to generate factors in a dynamic factor model for macroeconomic modeling. Moreover, Gu et al. [15] propose an asymmetric autoencoder for asset pricing with variable factor loadings. We do not use their model as we restrict ourselves to static factor models which do not allow time-varying betas.

The remainder of this paper is structured as follows: Section 2 gives a brief introduction to covariance matrix estimation using factor models and proposes our method to estimate the latent factors. Section 3 describes the rolling window portfolio optimization experiment where we compare our method with well-known competitors and Section 4 presents the results. Section 5 concludes and gives an outlook for possible future research.

*Both authors contributed equally to this research.

ICAIF '22, November 2–4, 2022, New York, NY, USA

ACM ISBN 978-1-4503-9376-8/22/11.
https://doi.org/10.1145/3533271.3561715

2 METHODOLOGY

We focus on static factor models to simplify the estimation problem as our primary goal is to develop an autoencoder that extracts latent factors. Further, we restrict our analysis to exact factor models to improve interpretability: The factors, along with the factor loadings should explain the entire covariance structure of the asset returns.

2.1 Covariance Matrix Estimation with Factor Models

A (static) factor model for the returns $r_{i,t}$ of assets $i = 1, \dots, N$ with (common) factors $f_t = (f_{1,t}, \dots, f_{K,t})'$ assumes

$$r_{i,t} = \alpha_i + \beta_i' f_t + u_{i,t},$$

where $\beta_i = (\beta_{i,1}, \dots, \beta_{i,K})'$ are the factor loadings and $u_{i,t}$ is the error term with $E(u_{i,t}|f_t) = 0$ for all dates $t = 1, \dots, T$. Stacking the factor loadings columnwise to the $K \times N$ matrix B, the returns $r_{i,t}$, intercepts α_i and error terms $u_{i,t}$ over all assets to the N-vectors r_t, α and u_t, respectively, allows us to compactly express the factor structure for all assets as

$$r_t = \alpha + B' f_t + u_t. \tag{1}$$

Define $\Sigma_{r,t} := \mathrm{Cov}(r_t|\mathcal{I}_{t-1})$, $\Sigma_{f,t} := \mathrm{Cov}(f_t|\mathcal{I}_{t-1})$ and $\Sigma_{u,t} := \mathrm{Cov}(u_t|\mathcal{I}_{t-1})$ as the covariance matrices conditional on the information set $\mathcal{I}_{t-1}$ in date $t-1$. Similar to Ledoit and Wolf [22] and Fan et al. [11], we focus on static factor models according to the definition in De Nard et al. [6, p. 3 f.]. The key assumptions that make a factor model static are time-invariant intercepts α and factor loadings B, as well as time-invariant conditional covariance matrices of the returns, factors and errors. Combined with the underlying assumption of (weak) stationarity for the factors $\{f_t\}$ and errors $\{u_t\}$, the latter implies $\Sigma_r := \mathrm{Cov}(r_t) = \Sigma_{r,t}$, $\Sigma_f := \mathrm{Cov}(f_t) = \Sigma_{f,t}$ and $\Sigma_u := \mathrm{Cov}(u_t) = \Sigma_{u,t}$ for all $t = 1, \dots, T$.

Therefore, under the static factor model, the covariance matrix of the returns follows easily from (1) as

$$\Sigma_r = B' \Sigma_f B + \Sigma_u.$$

As our paper focuses on the estimation of latent factors, we further restrict our attention to exact factor models which assume that Σ_u is a diagonal matrix. Following De Nard et al. [6], the estimation of Σ_r is straightforward: The intercepts and factor loadings are estimated in a linear regression of the returns on the factors using ordinary least squares (OLS), resulting in the residuals $\{\hat{u}_t\}$. Then, $\hat{\Sigma}_f$ follows as the sample covariance matrix of the factors $\{f_t\}$ and $\hat{\Sigma}_u = \mathrm{diag}(S_{\hat{u}})$, where $S_{\hat{u}}$ is the sample covariance. The estimator of Σ_r is then given by

$$\hat{\Sigma}_r := \hat{B}' \hat{\Sigma}_f \hat{B} + \hat{\Sigma}_u. \tag{2}$$

Note that $\hat{\Sigma}_r$ will usually have full rank because typically $\mathrm{rank}(\hat{\Sigma}_u) = N$, except e.g. when the factors perfectly explain some assets and their respective residual sum of squares becomes zero.

Fan et al. [11] compared the asymptotic behavior of $\hat{\Sigma}_r$ to the sample covariance matrix of the returns S_r under the static and exact factor model and the assumption of independent and identically distributed (i.i.d.) samples of the returns and *observable* factors $(r_1', f_1'), \dots, (r_T', f_T')$ when the number of assets N tends to ∞ as the sample size T increases. They conclude under technical assumptions and when the number of factors K is allowed to grow with N at $K = o(N)$, that the inverse $\hat{\Sigma}_r^{-1}$ exhibits a faster convergence rate in probability to Σ_r^{-1} than the inverse of the sample estimator S_r^{-1}. Therefore, the factor structure in $\hat{\Sigma}_r$ is useful for portfolio optimization as, e.g., the global minimum variance portfolio involves Σ_r^{-1}.

In contrast to an approximate factor model (allows Σ_u to be any finite and positive definite matrix), the assumption of an exact factor model may seem very restrictive. However, first, the assumption improves the interpretability of our estimators as the entire covariance structure of the returns are explained by the factors and factor loadings. Second, the method we propose can easily be extended to allow for an approximate factor model by e.g. applying the nonlinear shrinkage estimator of Ledoit and Wolf [26] or the principal orthogonal complement thresholding method (POET) by Fan et al. [12] to the residuals $\{\hat{u}_t\}$ as in De Nard et al. [6, p. 10 ff.]. The authors further propose a simple generalization to dynamic factor models by estimating the dynamic conditional correlation model with nonlinear shrinking (DCC-NL) [8] on $\{\hat{u}_t\}$. Another approach used in Conlon et al. [5] is to apply the dynamic conditional correlation model (DCC) [7] on the (estimated latent) factors $\{f_t\}$ combined with univariate GARCH models on $\{\hat{u}_t\}$ for the diagonal elements of $\hat{\Sigma}_{u,t}$.

The factors may be observed, such as the Fama-French factors [9], or latent in which case they have to be estimated from the data. A common approach for the latter is principal components analysis (PCA) [3, p. 205]. Our contribution is to develop an estimation method based on autoencoders.

2.2 Asymmetric Autoencoder

Autoencoders are unsupervised neural networks that can be used for dimension reduction and may be viewed as a (nonlinear) generalization of PCA [4, 15]. They consist of the encoder function g, which reduces the dimensionality of the returns r_t to the 'code' $z_t = g(r_t)$, and of the decoder function h which decodes the 'code' z_t from the encoder to the original dimension of the data via $\hat{r}_t = h(z_t)$. The output $\hat{r}_t$ is called the reconstruction. In this work, we focus on undercomplete autoencoders, where $g : \mathbb{R}^N \to \mathbb{R}^K$ with $K \ll N$ and consequently $h : \mathbb{R}^K \to \mathbb{R}^N$. The idea is that the autoencoder should capture the most salient features of the data thanks to the dimension reduction through g [14].

In order to fit the neural network, we parametrize the functions g and h with weight vectors θ_g and θ_h, respectively, which are then trained by minimizing a loss function $\mathcal{L}$, i.e.

$$\min_{\theta_g, \theta_h} \sum_{t=1}^{T} \mathcal{L}(r_t, h(g(r_t; \theta_g); \theta_h)). \tag{3}$$

Typically, autoencoders are symmetric, i.e. in the simple case where g and h both only consist of linear layers followed by a nonlinear activation function that translates to having the same number of layers for the encoder and decoder [18]. However, since our goal is to estimate latent factors $\{f_t\}$ via the encoder g to be used in factor models and covariance matrix estimation in (2), this would involve an additional step where we have to estimate the loadings B for the estimated latent factors $\{\hat{f}_t\}$ via a linear regression (as for the autoencoder of Conlon et al. [5]). We propose to integrate this last step into our autoencoder so we can *jointly* estimate the factors

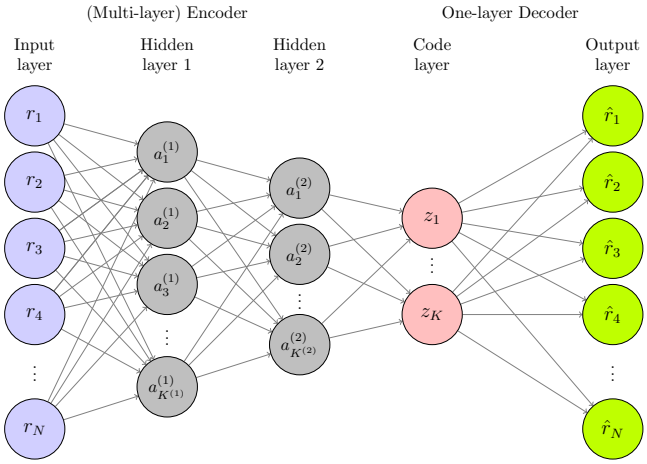

Figure 1: Architecture of the autoencoder

The cross-section of returns (blue) at each time period t is used as input of the encoder. All hidden units (grey) are computed using (4). The code layer contains latent factors z (red) which are decoded using a linear output layer (green).

$\{f_t\}$ and its loadings B by using a simple linear layer without any activation as decoder h.

In order to reduce clutter in our notation, we omit the time subscript t in the exposition of our autoencoder. Implicitly, we apply the encoder and decoder over all time periods t. The encoder is allowed to be nonlinear, i.e. it may have multiple linear layers followed by a nonlinear activation function. Let $K^{(l)}$ denote the number of hidden units in the l-th layer with $l = 0, 1, \ldots, L$. The output of unit k in layer l is given by $a_k^{(l)}$ which is collected in $a^{(l)} := \left(a_1^{(l)}, \ldots, a_{K^{(l)}}^{(l)}\right)'$. For the input layer $l = 0$, the input are the returns $r = a^{(0)}$. The remaining $a^{(l)}$ are computed via

$$a^{(l)} = \sigma\left(W^{(l)} a^{(l-1)} + b^{(l)}\right), \text{ for } l = 1, \ldots, L-1, \tag{4}$$

where $W^{(l)}$ is the $K^{(l)} \times K^{(l-1)}$ weight matrix, $b^{(l)}$ is the $K^{(l)}$-vector of biases and σ denotes an element-wise activation of the outputs in $a^{(l)}$. We use the hyperbolic tangent tanh as activation function $\sigma(x) = 2/(1 + \exp(-2x)) - 1$ which is a zero-centered function whose range lies between -1 and 1. Typically, factors for asset returns do not have a fixed range, thus, we omit the activation for the last encoding layer: $a^{(L)} = W^{(L)} a^{(L-1)} + b^{(L)} = z = f$ which are our latent factors.

The decoder is a simple linear layer $h(z) = Wz + b$, where b is an N-vector of biases and W is the $N \times K$ weight matrix. Since the input z of the decoder are the series of latent factors $\{f_t\}$ (extracted by the autoencoder), the weight matrix W corresponds to the factor loadings, i.e. $W = B'$. Figure 1 illustrates the idea of this autoencoder architecture using a three-layer encoder. The asymmetric architecture of the autoencoder should help improve the performance of our covariance estimator as the estimated latent factors are 'optimized' directly for the factor structure.

The parameters of the aforementioned autoencoders are estimated by minimizing the mean quadratic loss function where we regularize the training using weight decay (L_2-regularization). Thus, the optimization problem of (3) for our autoencoder is extended to

$$\min_{\theta_g, \theta_h} \left[\frac{1}{T}\sum_{t=1}^{T} ||r_t - h(g(r_t; \theta_g); \theta_h)||^2\right] + \lambda\left(||\theta_g||^2 + ||\theta_h||^2\right),$$

where $\lambda \geq 0$ is the weight decay parameter. Moreover, we apply dropout [40] to the code layer to further regularize the estimation and therefore prevent overfitting. In brief, dropout independently sets observations of the factors to zero with probability p during training, but rescales the remaining ones by $1/(1-p)$ to stand in for the omitted factors. In the simple case of linear regression, it may be viewed as a generalized ridge penalty [17, 43]. Note that dropout does not force the same factors over all time periods t to zero, as it *independently* chooses observations over all t and factors to zero. Dropout is not used after training when generating the estimated factors $\{\hat{f}_t\}$ with the fitted network.

2.3 Minimum Variance Portfolios

We apply our covariance matrix estimators to construct minimum variance portfolios which have been frequently used in the portfolio optimization literature and date back to Markowitz [32]. The optimization problem is described as

$$\begin{aligned} &\min_w w'\hat{\Sigma}_r w, \\ &\text{s.t. } w'\mathbf{1} = 1, \end{aligned} \tag{5}$$

where $w = (w_1, \ldots, w_N)'$ is a vector of portfolio weights, $\mathbf{1}$ is an $N \times 1$ vector of ones, and $\hat{\Sigma}_r$ is the estimated covariance matrix. Different estimators are considered for $\hat{\Sigma}_r$ which are described in Section 3.2. The constraint ensures that the portfolio is fully invested, i.e. the portfolio weights must add up to one. Depending on whether one allows short selling, another constraint is imposed on the problem, which is formulated as

$$w_i \geq 0, \text{ for } i = 1, \ldots, N, \tag{6}$$

which means that all portfolio weights are non-negative. There are many portfolio optimization applications, some of which include the constraint of (6), such as Conlon et al. [5] or Ledoit and Wolf [23]. De Nard et al. [6], on the other hand, allow short selling in their work. In this paper we consider both cases: The portfolio resulting from (5) is denoted by GMV (global minimum variance portfolio) and if the short-sale constraint of (6) is imposed, then it is denoted by GMV$^+$.

3 EMPIRICAL ANALYSIS

3.1 Data and Portfolio Construction Rules

The data set for this empirical analysis consists of individual stock return data from the Center for Research in Security Prices (CRSP) downloaded from January 1973 to December 2021. All ordinary common shares (share codes 10 and 11) listed on the NYSE, AMEX, and NASDAQ (exchange codes 1, 2, and 3) were selected. We also downloaded the five factor data series of Fama and French [10] as well as the one-month U.S. Treasury bill rate (risk-free rate) for the same time period from Kenneth French's website.[1]

We adopt a monthly sliding window procedure similar to the common literature e.g. Ledoit and Wolf [23], De Nard et al. [6], and

[1] http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html

Conlon et al. [5]. At the beginning of each investment date we select all stocks with a history of $T = 120$ months. We find the stocks that have an almost complete return history (availability of > 97.5% observations) and an observable next month out-of-sample return. Whilst this "forward-looking" restriction is not perfectly realistic, it corresponds to the standard in the literature. Finally, stocks with a share price below 5 USD have been filtered out. The out-of-sample period therefore ranges from January 1983 to December 2021, resulting in a total of 468 months.

We denote investment dates $s = 1, \ldots, 468$. At any investment date s the covariance matrix is estimated based on the historic $T = 120$ monthly returns using the N largest stocks by market capitalization. The considered portfolio sizes are $N \in \{50, 100, 200\}$.

Simple models such as the sample covariance estimator take the resulting $T \times N$ matrix to construct the covariance matrix. Covariance matrix estimators based on factor models that do not require hyperparameter tuning apply (2) using the factors (which may be estimated by a dimension reduction method), and their corresponding loadings and residuals estimated by OLS. For machine learning-based models that require hyperparameter tuning we adopt a simple validation sample approach, also used in Conlon et al. [5, p. 17 ff.], that maintains the temporal ordering of the data: In each iteration of the rolling window the data is split into two disjoint periods, namely the first 80% of the sample running from $t = 1, \ldots, T^\tau$, with $T^\tau = 96$, for model training and the following 20% of the data running from $t = T^\tau + 1, \ldots, T$ for model validation. The hyperparameter selection of each model is given in Section 3.2.

More specifically, methods that require hyperparameter tuning are estimated using information of the training sample from the following regression equation:

$$r_t = \hat{\alpha}^\tau + \hat{B}^{\tau\prime}\hat{\psi}^\tau(r_t) + \hat{u}_t, \text{ for } t = 1, \ldots, T^\tau,$$

where $\hat{B}^\tau$ is the estimated matrix of factor loadings (by OLS) using only data up to T^τ, and $\hat{\psi}^\tau$ is the estimated dimension reduction function using a particular set of hyperparameters which depends on returns r_t. Next the covariance over the training subsample is calculated by

$$\hat{\Sigma}_r^\tau = \hat{B}^{\tau\prime}\hat{\Sigma}_f^\tau\hat{B}^\tau + \hat{\Sigma}_u^\tau,$$

where $\hat{\Sigma}_f^\tau$ is the sample covariance matrix of the factors $\hat{\psi}^\tau(r_t)$ over the training data and $\hat{\Sigma}_u^\tau$ is a diagonal matrix containing the sample variances of residuals $\hat{u}_t$ for $t = 1, \ldots, T^\tau$. For validation, the covariance matrix is calculated, again using the estimated matrix of loadings $\hat{B}^\tau$ as

$$\hat{\Sigma}_r^\nu = \hat{B}^{\tau\prime}\hat{\Sigma}_f^\nu\hat{B}^\tau + \hat{\Sigma}_u^\nu,$$

where $\hat{\Sigma}_f^\nu$ is the sample covariance matrix of the estimated factors $\hat{\psi}^\tau(r_t)$ over the validation data and $\hat{\Sigma}_u^\nu$ is a diagonal matrix containing the sample variance of the residuals derived by $\hat{u}_t = r_t - \left(\hat{\alpha}^\tau + \hat{B}^{\tau\prime}\hat{\psi}^\tau(r_t)\right)$ for $t = T^\tau + 1, \ldots, T$.

The covariance matrix is evaluated by finding portfolio weights $\hat{w}$ for the GMV and GMV$^+$ portfolio (see Section 2.3) using the covariance matrix $\hat{\Sigma}_r^\tau$ estimated from the training set. The validation portfolio variance is then calculated as $\hat{w}'\hat{\Sigma}_r^\nu\hat{w}$. This procedure is repeated for each set of hyperparameters.

The set that yields the lowest portfolio variance over the validation sample is the optimal hyperparameter setting. This setting is then used to re-estimate the latent factors and factor loadings using all the observations in the rolling window. Finally, the actual out-of-sample performance is evaluated by constructing a portfolio at investment date s and observing the portfolio performance in $s+1$ using the asset returns that have not been part of the parameter estimation process. This rolling window approach is repeated until November 2021 ($s = 468$). Following De Nard et al. [6], transaction costs throughout all experiments are ignored.

3.2 Competing Estimators

In order to compare the performance of our model, we include different methods for the empirical analysis. Importantly, we examine popular alternative dimension reduction approaches from the literature as well as some other reference portfolios. Note that the factor models for this comparison are restricted to exact factor models. To achieve parsimonious model specifications that still capture the main (co-)variation in the data, we decided to use 5 latent factors for all dimension reduction methods. Competing factor-based estimators are the following:

- *FF5*: The five factors of Fama and French [10]: market return in excess of the risk-free rate, size premium (small-minus-big), value premium (high-minus-low), profitability (robust-minus-weak), and investment (conservative-minus-aggressive).
- *AE*: Our estimator as described in Section 2.2 and shown in Figure 1.
- *AE-SYM*: A symmetric autoencoder, i.e. a comparable architecture as *AE*, but with a decoder that is symmetric to the encoder.
- *PCA*: Principal Components Analysis.
- *SPCA*: Sparse Principal Components Analysis as suggested by Mairal et al. [30].
- *CCK*: A symmetric autoencoder neural network by Conlon et al. [5] that uses ranks of stock returns which are mapped on a $[-0.5, 0.5]$ interval as input features. The authors claim that this transformation focuses on the ordering of the data and is insensitive to outliers.

Note that FF5 does not use dimension reduction but employs observable factors. It serves as a natural benchmark to assess whether latent factors from dimension reduction methods lead to better portfolio performance.

We also include some simple portfolios:

- *Sample*: The minimum-variance portfolio based on the sample covariance matrix estimator.
- *VW*: A value-weighted portfolio based on each stock's market capitalization at investment date s.
- *EW*: An equally-weighted portfolio, i.e. a weight of $w_i = 1/N \; \forall \; i$.

All factor-based estimators apart from AE are two-step procedures, i.e., the latent factors are first estimated by the corresponding dimension reduction method which are then plugged into (1) to estimate the factor loadings by OLS. For our method AE, the loadings are directly extracted from the decoding layer of the neural network. Furthermore, all dimension reduction methods, except for CCK, are estimated based on standardized returns and their

Table 1: Hyperparameter tuning - Parameters and choice of values

Method(s)	Hyperparameters	Values
AE, AE-SYM	encoding layers	[1, 2]
	λ	$[1e^{-4}, 1e^{-3}]$
	dropout ratio	[0.20, 0.25, 0.30]
SPCA	α	[0.1, 0.2, 0.5, 0.8, 1.0, 1.2, 1.5, 1.8, 2.0, 2.2, 2.5, 2.8, 3.0, 3.5, 4.0, 4.5, 5.0]
CCK	λ	$[0.0, 1e^{-5}, 1e^{-4}]$
	δ_1	$[0.0, 1e^{-5}, 1e^{-4}]$
	δ_2	$[0.0, 1e^{-5}, 1e^{-4}]$

resulting covariance matrix estimate is then rescaled. The standardization is carried out in each rolling window. For methods that need hyperparameter tuning, the required moments are estimated from the training set when determining the optimal hyperparameters. The final model then uses the moments estimated from the entire window. For CCK, standardization is irrelevant as it uses ranks of returns as input.

All autoencoders are fitted by minimizing the mean quadratic loss function using Adam [21] with a fixed learning rate of 0.005 and model-dependent weight decay λ. We forego batch optimization as our datasets are quite small. Similar to Gu et al. [15] we apply early stopping to reduce the computational burden and further regularize the procedures: The Adam optimizer is allowed to run a maximum number of 3'500 epochs but is stopped early if the validation loss does not improve for 300 consecutive epochs. In addition, CCK applies activity regularization on the output of the encoder using an elastic net penalty term, controlled by δ_1 and δ_2 for the L_1 and L_2 penalty, respectively, which shrinks the estimated latent factors [14].

Finally, the number of neurons in the hidden layers of the autoencoders is chosen according to the geometric pyramid rule [33, p. 176 ff.]. For the symmetric autoencoder, we simply mirror the number of neurons of the corresponding layers of the encoder.

Table 1 covers the choices of hyperparameters as well as the corresponding values. Each hyperparameter combination is tested using the validation sample approach as described in Section 3.1. The choice of hyperparameters was determined in simple pre-experiments in order to find a suitable range.

3.3 Evaluation Measures

Inspired by Conlon et al. [5] and De Nard et al. [6], we compare the out-of-sample performance of the covariance matrix estimators using measures based on the out-of-sample portfolio returns. Those are:

- SD: The standard deviation of the 468 out-of-sample returns annualized by multiplying by $\sqrt{12}$.
- SR: The Sharpe ratio as (AV − RF)/SD, where AV and RF are the annualized (multiplied by 12) arithmetic means of the 468 out-of-sample returns and of the risk-free rate, respectively.

In context of minimum variance portfolios, the most important performance measure is the out-of-sample SD. SR measures the risk-variance tradeoff and is of secondary importance when evaluating the covariance matrix estimators.

We apply the hypothesis test of Ledoit and Wolf [25] to check whether the proposed factor-based estimators deliver significantly different SD than the well-known FF5 model. Let σ_m denote the standard deviation of the out-of-sample returns of method $m \in \{\text{AE, AE-SYM, PCA, SPCA, CCK}\}$ and let σ_{FF5} be the standard deviation of the out-of-sample returns generated by FF5. Then, we test the null hypothesis

$$H_{0,m} : \log(\sigma_m) = \log(\sigma_{\text{FF5}}) \quad \text{vs.} \quad H_{1,m} : \log(\sigma_m) \neq \log(\sigma_{\text{FF5}}),$$

for all m and portfolio sizes N. As recommended in Ledoit and Wolf [25, p. 87], we calculate the p-values using a circular block bootstrap. The block size for each comparison is chosen using the algorithm provided in Ledoit and Wolf [24, p. 854] from the candidates[2] $\{1, 3, 6, 10\}$. As multiple null hypotheses are tested, we control for the family-wise error rate by correcting all p-values using the Holm-Bonferroni procedure [19]. Those adjustments are applied over all m but separately for each scenario, which is the combination of portfolio size N and optimization constraint (GMV or GMV$^+$). Since the major goal of the tests is to compare the performance of factor models to the benchmark FF5, we omit Sample[3], EW, and VW[4] from the tests and achieve more power from the Holm-Bonferroni correction.

3.4 Implementation

All computations are performed in Python 3.8.13. Generally, numerical computations and data preprocessing use NumPy 1.22.3 [16], SciPy 1.8.0 [42], and pandas 1.4.1 [41], while all neural networks are implemented in PyTorch 1.11.0 [34]. Specifically, the CCK autoencoder is reimplemented in PyTorch according to Conlon et al. [5]. The remaining dimension reduction methods PCA and its extension SPCA, as well as the OLS routine for the estimation of the factor loadings for most methods, are taken from scikit-learn 1.0.2 [35]. The portfolio weights are optimized using the quadratic programming solver of CVXOPT 1.3.0 [1]. The hypothesis tests of Section 3.3 are conducted using the MATLAB implementation provided on Michael Wolf's website[5] and the Holm-Bonferroni correction is taken from statsmodels 0.13.2 [39].

Finally, all calculations are performed on the sciCORE scientific computing center at the University of Basel.[6]

4 RESULTS

4.1 Performance Measures

Table 2 shows the SD and SR of the global minimum variance (GMV) portfolios from the optimization problem of (5) for the different methods and varying portfolio sizes. In summary, our proposed method AE yields consistently low SDs for all N. For $N = 50$ and $N = 200$, AE ranks second, with values of 12.57% and 11.61%,

[2] Default values recommended by the authors.
[3] Sample is highly unstable for GMV portfolios and even breaks down for $N = 200$.
[4] EW and VW always yield higher SD than FF5.
[5] https://www.econ.uzh.ch/en/people/faculty/wolf/publications.html#Programming_Code
[6] http://scicore.unibas.ch/

Table 2: Main results for the GMV portfolio

	$N = 50$			$N = 100$			$N = 200$		
model	SD	RK	SR	SD	RK	SR	SD	RK	SR
FF5	13.42	6	0.93	12.74	6	0.85	12.88	6	0.84
AE	12.57**	2	1.00	**11.41*****	1	0.94	11.61***	2	0.98
AE-SYM	13.06	4	1.00	11.89**	4	0.96	11.98**	5	0.93
PCA	13.16	5	0.92	11.47***	2	0.96	**11.59*****	1	0.97
SPCA	12.94	3	1.00	11.59***	3	0.97	11.68***	3	0.98
CCK	**12.41**	1	1.04	12.02	5	1.00	11.78**	4	0.94
Sample	14.85	9	0.77	23.91	9	0.48	–	–	–
EW	14.57	7	0.88	14.72	8	0.83	14.68	8	0.86
VW	14.57	8	0.85	14.51	7	0.84	14.43	7	0.85

This table shows the annualized standard deviation (SD) and Sharpe ratio (SR) over the 468 out-of-sample returns from January 1983 to December 2021 for three portfolio sizes $N \in \{50, 100, 200\}$ across 9 methods (6 factor-based covariance matrix estimators and 3 simple portfolios) of the global minimum variance portfolio without short-sale constraint. RK ranks the methods from smallest to largest according to SD. Asterisks indicate the significance of the hypothesis tests of the dimension reduction methods (AE, AE-SYM, PCA, SPCA, CCK) against the benchmark (FF5). ***, ** and * denote significance at the 0.01, 0.05 and 0.1 level, respectively.

Table 3: Main results for GMV$^+$ portfolio

	$N = 50$			$N = 100$			$N = 200$		
model	SD	RK	SR	SD	RK	SR	SD	RK	SR
FF5	12.65	7	1.00	11.92	6	0.94	12.11	7	0.89
AE	**12.32**	1	1.02	**11.54****	1	0.94	**11.54****	1	0.98
AE-SYM	12.46	4	1.04	11.66	4	0.95	11.57*	2	0.97
PCA	12.55	6	1.04	11.61	2	0.94	11.60	3	0.98
SPCA	12.46	3	1.05	11.70	5	0.92	11.67	5	0.99
CCK	12.53	5	1.03	12.00	7	0.99	11.64	4	0.97
Sample	12.43	2	1.01	11.64	3	0.94	11.93	6	0.94
EW	14.57	8	0.88	14.72	9	0.83	14.68	9	0.86
VW	14.57	9	0.85	14.51	8	0.84	14.43	8	0.85

This table shows the annualized standard deviation (SD) and Sharpe ratio (SR) over the 468 out-of-sample returns from January 1983 to December 2021 for three portfolio sizes $N \in \{50, 100, 200\}$ across 9 methods (6 factor-based covariance matrix estimators and 3 simple portfolios) of the minimum variance portfolio with short-sale constraint (no negative weights). RK ranks the methods from smallest to largest according to SD. Asterisks indicate the significance of the hypothesis tests of the dimension reduction methods (AE, AE-SYM, PCA, SPCA, CCK) against the benchmark (FF5). ***, ** and * denote significance at the 0.01, 0.05 and 0.1 level, respectively.

respectively, and first for $N = 100$ with 11.41%. Moreover, in all scenarios its SD is significantly different from FF5 at the usual significance levels. The estimator with the lowest SD for $N = 50$ is CCK with a value of 12.41%, however, the difference to FF5 is not significant although its SD is lower than AE. This is due to the fact that the performed test incorporates higher-order moment dependencies between the returns of CCK and FF5 which, in this case, results in a higher standard error than for AE. For details see Ledoit and Wolf [25, p. 87]. For $N = 100$ CCK drops to rank 5 with 12.02% which is about 5.3% higher than AE. Moreover, for $N = 200$, CCK ranks 4th with 11.78%.

Autoencoders can be interpreted as (nonlinear) extensions of PCA, so it is interesting to compare AE with PCA. PCA is the closest competitor of AE for $N = 100$ at rank 2 with an SD of 11.47%. However, for $N = 50$, it performs much worse with an SD that is around 4.7% larger than AE. Interestingly, for $N = 200$, AE ranks second behind PCA with marginal differences in SD. We also included SPCA for comparison to check whether the regularized estimation of AE is the main driver for the improved performance or if it is due to the network structure itself. The results are mixed, for $N = 100$ and $N = 200$, SPCA is worse than PCA, but for $N = 50$ the opposite is true. Still, SPCA is worse than AE in all scenarios, suggesting that the model architecture of AE is key for the performance gain of AE compared to PCA and SPCA.

Importantly, AE performs better than AE-SYM for all portfolio sizes N which shows that the asymmetric structure of AE yields factors and loadings that are more suitable for portfolio optimization in our scenarios.

The sample covariance estimator ranks last in terms of SD for small $N = 50$, and becomes highly unstable at $N = 100 \approx T$ with a very high SD of 23.91%. It yields no results for $N = 200$ because the resulting covariance matrix is singular.

Table 3 shows the same out-of-sample measures for the GMV$^+$ portfolios where only positive portfolio weights are allowed. Overall, the SDs are lower across most methods compared to the GMV portfolios, especially for small $N = 50$. This can be explained by the fact, that constraining the weights to be nonnegative effectively shrinks the covariance matrix estimator [20, p. 1657].

AE again yields very competitive and robust SDs for all N, ranking first in every scenario. It also is the only dimension reduction method which is significantly different from FF5 for $N = 100$ and $N = 200$ at the 5% level. However, PCA achieves higher SRs compared to AE for $N = 50$, while its SR is similar to that of AE for $N = 100$ and $N = 200$. Again, the regularization in SPCA does not always improve upon PCA: its SD is higher in both $N = 100$ and $N = 200$. Interestingly, while CCK yields the GMV portfolio with the lowest SD for small $N = 50$, its relative performance is much worse for the GMV$^+$ portfolio with short-sale constraint: it drops to rank 5.

The sample covariance estimator performs much better with short-sale constraint. For $N = 50$ and $N = 100$ it is competitive ranking second and third, but dropping to 6th for larger portfolios.

4.2 Factors and Loadings

The asymmetric structure of our autoencoder leads to a joint estimation of the latent factors and their loadings which should improve the interpretability of the model. The loadings are directly observable as weights of the single linear decoding layer. Figure 2 displays the ten highest loading exposures for each factor in a descending order. The corresponding model is the hyperparameter-tuned version estimated as of 30.11.2021 for $N = 100$ stocks, which is the last investment date ($s = 468$) of our data set. Since the results of the neural network are data-driven and computed without additional knowledge of the stocks, there is no prior convention on how to interpret the latent factors. To provide a possible explanation for this ranking, we have colored each stock based on the MSCI

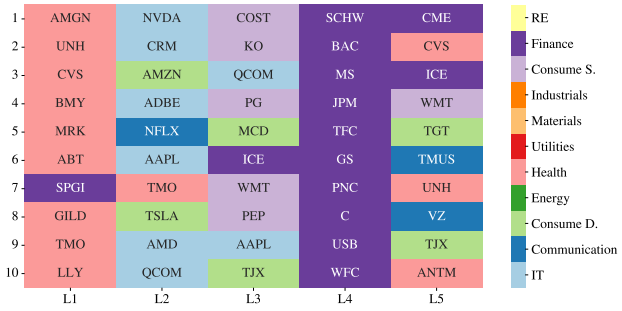

Figure 2: Top 10 loadings for AE model as of 30.11.2021

The model is trained on the last investment date ($s = 468$) using input from the previous 10 years. Each column represents loadings of a specific factor taken from the neural network decoder, ranked by their exposure. Colors represent sectors based on the GICS-codes. Stocks are labeled with their ticker symbol.

Global Industry Classification Standard (GICS)[7] code downloaded from COMPUSTAT dating from 31.12.2020, which classifies each stock into one of eleven sectors. Additionally, we labeled each stock with its current ticker symbol. The resulting color-matrix suggests the following explanation: Factor 1 mainly drives companies from the health sector, e.g. pharmaceuticals and healthcare companies. Factor 2 shows high exposures for tech companies e.g. NVIDIA, Adobe, Apple or AMD. Factor 3 is related to the consumer industry. Wholesale corporations like Costco or Walmart, and consumer goods corporations like Coca Cola or Procter & Gamble can be found here. Factor 4 visibly summarizes corporations in the financial industry: (investment) banks, insurance companies, and other financial services. Factor 5 is not as characterized by a particular sector as the other factors. It attempts to capture the remaining variation in the returns.

The results appear to be economically reasonable. Evidently, the model also identifies similar companies from different sectors, such as Amazon and Tesla, which are both tech-related even though they are not classified as 'IT' in the GICS codes.

5 CONCLUSION

We proposed an autoencoder structure for factor-based high dimensional covariance matrix estimation applied to portfolio allocation. We focused on exact and static factor models, so the covariance structure of the asset returns is completely determined by the factors and their loadings. The asymmetric architecture of our procedure is tailored for factor modeling and jointly estimates the latent factors along with their loadings, and therefore avoids the usual two-step procedure required by other dimension reduction methods.

We ran our method in an empirical application using monthly stock return data of the past five decades. Using typical risk measures, the performance of the asymmetric autoencoder was compared to well-established covariance matrix estimators from the literature, which include comparable dimension reduction methods and other reference portfolios. To test whether dimension reduction techniques yield better results than observable factors, the well-known five-factors of Fama and French [10] were chosen as the benchmark model. Moreover, we evaluated the decoder weights of the neural network to find a possible interpretation of the latent factors.

Our results can be summarized as follows: The proposed autoencoder is highly competitive across any tested portfolio size and optimization constraints (short-sale). Importantly, it is the only dimension reduction method that consistently outperforms the benchmark with significant difference. Moreover, it always improves upon a symmetric autoencoder and compares favorably with PCA which justifies the selected architecture. Furthermore, we have shown that the asymmetric autoencoder estimates data-driven latent factors that are economically reasonable, even without having prior knowledge of the stock characteristics. Therefore, we shed light on an often regarded *blackbox* procedure by finding meaning in the latent variables. The suggested color-scheme of the factor loadings by sectors is only one way to interpret the latent factors. Future research is required to better understand the economic content of the latent factors.

[7] See https://www.msci.com/our-solutions/indexes/gics for more details on the industries.

ACKNOWLEDGMENTS

The scientific computing center sciCORE at the University of Basel provided us with valuable computing resources that enabled us to crucially accelerate our research progress.

We would also like to thank Dietmar Maringer, Christian Kleiber, and Tim Kroencke for their feedback.

REFERENCES

[1] Martin Andersen, Joachim Dahl, and Lieven Vandenberghe. 2022. *CVXOPT: A Python package for convex optimization.* cvxopt.org Version 1.3.0.
[2] Paolo Andreini, Cosimo Izzo, and Giovanni Ricco. 2020. Deep Dynamic Factor Models. (2020). https://doi.org/10.48550/arXiv.2007.11887 arXiv:2007.11887
[3] Jushan Bai and Shuzhong Shi. 2011. Estimating High Dimensional Covariance Matrices and its Applications. *Annals of Economics and Finance* 12, 2 (2011), 199–215.
[4] Pierre Baldi and Kurt Hornik. 1989. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks* 2, 1 (1989), 53–58. https://doi.org/10.1016/0893-6080(89)90014-2
[5] Thomas Conlon, John Cotter, and Iason Kynigakis. 2021. *Machine Learning and Factor-Based Portfolio Optimization.* Research Paper 21-6. Michael J. Brennan Irish Finance Working Paper Series. https://doi.org/10.2139/ssrn.3889459
[6] Gianluca De Nard, Olivier Ledoit, and Michael Wolf. 2021. Factor models for portfolio selection in large dimensions: The good, the better and the ugly. *Journal of Financial Econometrics* 19, 2 (2021), 236–257. https://doi.org/10.1093/jjfinec/nby033
[7] Robert F. Engle. 2002. Dynamic conditional correlation: A simple class of multivariate generalized autoregressive conditional heteroskedasticity models. *Journal of Business & Economic Statistics* 20, 3 (2002), 339–350. https://doi.org/10.1198/073500102288618487
[8] Robert F. Engle, Olivier Ledoit, and Michael Wolf. 2019. Large dynamic covariance matrices. *Journal of Business & Economic Statistics* 37, 2 (2019), 363–375. https://doi.org/10.1080/07350015.2017.1345683
[9] Eugene F. Fama and Kenneth R. French. 1993. Common risk factors in the returns on stocks and bonds. *Journal of Financial Economics* 33, 1 (1993), 3–56. https://doi.org/10.1016/0304-405X(93)90023-5
[10] Eugene F. Fama and Kenneth R. French. 2015. A five-factor asset pricing model. *Journal of Financial Economics* 116, 1 (2015), 1–22. https://doi.org/10.1016/j.jfineco.2014.10.010
[11] Jianqing Fan, Yingying Fan, and Jinchi Lv. 2008. High dimensional covariance matrix estimation using a factor model. *Journal of Econometrics* 147, 1 (2008), 186–197. https://doi.org/10.1016/j.jeconom.2019.04.026
[12] Jianqing Fan, Yuan Liao, and Martina Mincheva. 2013. Large covariance estimation by thresholding principal orthogonal complements. *Journal of the*

Royal Statistical Society: Series B (Statistical Methodology) 75, 4 (2013), 603–680. https://doi.org/10.1111/rssb.12016

[13] Guanhao Feng, Jingyu He, and Nicholas G. Polson. 2018. Deep Learning for Predicting Asset Returns. (2018). https://doi.org/10.48550/arXiv.1804.09314 arXiv:1804.09314

[14] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[15] Shihao Gu, Bryan Kelly, and Dacheng Xiu. 2021. Autoencoder asset pricing models. *Journal of Econometrics* 222, 1 (2021), 429–450. https://doi.org/10.1016/j.jeconom.2020.07.009

[16] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, et al. 2020. Array programming with NumPy. *Nature* 585, 7825 (2020), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[17] Trevor Hastie. 2020. Ridge Regularization: An Essential Concept in Data Science. *Technometrics* 62, 4 (2020), 426–433. https://doi.org/10.1080/00401706.2020.1791959

[18] Geoffrey E. Hinton and Ruslan R. Salakhutdinov. 2006. Reducing the Dimensionality of Data with Neural Networks. *Science* 313, 5786 (2006), 504–507. https://doi.org/10.1126/science.1127647

[19] Sture Holm. 1979. A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics* 6, 2 (1979), 65–70.

[20] Ravi Jagannathan and Tongshu Ma. 2003. Risk Reduction in Large Portfolios: Why Imposing the Wrong Constraints Helps. *The Journal of Finance* 58, 4 (2003), 1651–1683. https://doi.org/10.1111/1540-6261.00580

[21] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. (2015). https://doi.org/10.48550/arXiv.1412.6980 arXiv:1412.6980v9

[22] Olivier Ledoit and Michael Wolf. 2003. Improved estimation of the covariance matrix of stock returns with an application to portfolio selection. *Journal of Empirical Finance* 10, 5 (2003), 603–621. https://doi.org/10.1016/S0927-5398(03)00007-0

[23] Olivier Ledoit and Michael Wolf. 2004. Honey, I shrunk the sample covariance matrix. *The Journal of Portfolio Management* 30, 4 (2004), 110–119. https://doi.org/10.3905/jpm.2004.110

[24] Oliver Ledoit and Michael Wolf. 2008. Robust performance hypothesis testing with the Sharpe ratio. *Journal of Empirical Finance* 15, 5 (2008), 850–859. https://doi.org/10.1016/j.jempfin.2008.03.002

[25] Olivier Ledoit and Michael Wolf. 2011. Robust Performances Hypothesis Testing With the Variance. *Wilmott Magazine* 2011, 55 (2011), 86–89. https://doi.org/10.1002/wilm.10036

[26] Olivier Ledoit and Michael Wolf. 2017. Nonlinear shrinkage of the covariance matrix for portfolio selection: Markowitz meets Goldilocks. *The Review of Financial Studies* 30, 12 (2017), 4349–4388. https://doi.org/10.1093/rfs/hhx052

[27] Olivier Ledoit and Michael Wolf. 2020. The Power of (Non-)Linear Shrinking: A Review and Guide to Covariance Matrix Estimation. *Journal of Financial Econometrics* 20, 1 (2020), 187–218. https://doi.org/10.1093/jjfinec/nbaa007

[28] Martin Lettau and Markus Pelger. 2020. Estimating latent asset-pricing factors. *Journal of Econometrics* 218, 1 (2020), 1–31. https://doi.org/10.1016/j.jeconom.2019.08.012

[29] John Lintner. 1965. Security Prices, Risk, and Maximal Gains From Diversification. *The Journal of Finance* 20, 4 (1965), 587–615. https://doi.org/10.1111/j.1540-6261.1965.tb02930.x

[30] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. 2009. Online Dictionary Learning for Sparse Coding. In *Proceedings of the 26th Annual International Conference on Machine Learning (PMLR '09)*. 689–696. https://doi.org/10.1145/1553374.1553463

[31] Angshul Majumdar and Aditay Tripathi. 2017. Asymmetric stacked autoencoder. In *Proceedings of the 2017 International Joint Conference on Neural Networks* (Anchorage, Alaska, USA) *(IJCNN '18)*. IEEE, 911–918. https://doi.org/10.1109/IJCNN.2017.7965949

[32] Harry Markowitz. 1952. Portfolio Selection. *The Journal of Finance* 7, 1 (1952), 77–91. https://doi.org/10.1111/j.1540-6261.1952.tb01525.x

[33] Timothy Masters. 1993. *Practical neural network recipes in C++*. Academic Press Professional, San Diego, London.

[34] Adam Paszke, Sam Gross, Francisco Massa, et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32 (NIPS '19, Vol. 32)*, H. Wallach, H. Larochelle, A. Beygelzimer, et al. (Eds.). Curran Associates, 1–12. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[35] Fabian Pedregosa, Gael Varoquaux, Alexendre Gramfort, et al. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[36] Stephen A. Ross. 1976. The arbitrage theory of capital asset pricing. *Journal of Economic Theory* 13, 3 (1976), 341–360. https://doi.org/10.1016/0022-0531(76)90046-6

[37] Stephen A. Ross. 1977. The capital asset pricing model (CAPM), short-sale restrictions and related issues. *The Journal of Finance* 32, 1 (1977), 177–183. https://doi.org/10.1111/j.1540-6261.1977.tb03251.x

[38] William F. Sharpe. 1963. A Simplified Model for Portfolio Analysis. *Management Science* 9, 2 (1963), 277–293. https://doi.org/10.1287/mnsc.9.2.277

[39] Skipper Seabold and Josef Perktold. 2010. Statsmodels: Econometric and Statistical Modeling with Python. In *Proceedings of the 9th Python in Science Conference*, Stéfan van der Walt and Jarrod Millman (Eds.). 92 – 96. https://doi.org/10.25080/Majora-92bf1922-011

[40] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, et al. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15, 56 (2014), 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html

[41] The pandas development team. 2022. *pandas-dev/pandas: Pandas*. https://doi.org/10.5281/zenodo.3509134 Version 1.4.1.

[42] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, et al. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17 (2020), 261–272. https://doi.org/10.1038/s41592-019-0686-2

[43] Stefan Wager, Sida Wang, and Percy S. Liang. 2013. Dropout Training as Adaptive Regularization. In *Advances in Neural Information Processing Systems (NIPS '13, Vol. 26)*, C.J. Burges, L. Bottou, M. Welling, et al. (Eds.). Curran Associates, 1–9. https://proceedings.neurips.cc/paper/2013/file/38db3aed920cf82ab059bfccbd02be6a-Paper.pdf

[44] Yuyang Wang, Alex Smola, Danielle Maddix, et al. 2019. Deep Factors for Forecasting. In *Proceedings of the 36th International Conference on Machine Learning (PMLR '19, Vol. 97)*. 6607–6617. https://proceedings.mlr.press/v97/wang19k.html