Recurrent Neural Network GO-GARCH Model for Portfolio Selection

Martin Burda*[ab] Adrian K. Schroeder[ac]

June 25, 2024

Abstract

We develop a hybrid model of multivariate volatility that uses Recurrent Neural Networks to capture the conditional variances of latent orthogonal factors in a GO-GARCH framework. Our approach seeks to balance model flexibility with ease of estimation and can be used to model conditional covariances of a large number of assets. The model performs favourably in comparison with relevant benchmark models in a Minimum Variance Portfolio (MVP) scenario.

JEL: C32, C45, G11, G12
Keywords: LSTM, machine learning, nonlinear time series, multivariate volatility forecasting

*Corresponding author.
[a]Department of Economics, University of Toronto, 150 St. George St., Toronto, ON, M5S 3G7, Canada
[b]E-mail: martin.burda@utoronto.ca
[c]E-mail: adrian.schroeder@mail.utoronto.ca
This project is supported in part by funding from the Social Sciences and Humanities Research Council (SSHRC). Computations were performed on the Mist supercomputer at the SciNet HPC Consortium. SciNet is funded by: the Canada Foundation for Innovation; the Government of Ontario; Ontario Research Fund - Research Excellence; and the University of Toronto.

1 Introduction

Modelling multivariate financial asset returns volatility plays a key role in financial portfolio management. Analysis of the dynamic evolution of variances and covariances of asset portfolios is used for a number of purposes, such as forecasting value-at-risk (VaR) thresholds to determine compliance with the Basel Accords, tracking spillover effects in studies of contagion, or determining optimal asset pricing and hedging positions.

Many different types of multivariate volatility models have been proposed in the literature (for a recent survey, see Boudt et al., 2019). Popular specifications include the BEKK GARCH model (Engle and Kroner, 1995) and the dynamic conditional correlation (DCC) model (Engle, 2002; Aielli, 2013). BEKK is typically used to forecast conditional covariances, although it may also be used to forecast conditional correlations indirectly. DCC is primarily intended for forecasting conditional correlations, although it can be applied to forecast conditional covariances (Caporin and McAleer, 2012).

It is well known that many fully parametrized multivariate volatility models suffer from the so-called curse of dimensionality, whereby the number of parameters increases at an order greater than the number of assets m. Specifically, in both the commonly used implementation of BEKK and in the DCC model the number of parameters increases at order $O(m^2)$ (Caporin and McAleer, 2012). Practical implementation of the DCC model is aided by estimating univariate models of variance separately from the correlation matrix, although the latter still represents a challenging multivariate object for optimization when the number of assets is large.

Parameter parsimony can be achieved by imposition of parametric restrictions, resulting in diagonal or scalar model versions (Silvennoinen and Teräsvirta, 2009). However, enforcement of such restrictions is traded off with model flexibility. For example in the BEKK case the diagonal and scalar versions are substantially dominated in terms of model fit by the unrestricted model with targeting (Burda and Maheu, 2012). Indeed, the types of restrictions needed to achieve parameter parsimony generally operate on the parameters driving the model dynamics.

Factor (Engle et al., 1990) and generalized orthogonal (GO-) (van der Weide, 2002) multivariate

volatility models achieve a degree of pasimony by imposing a common dynamic structure on all the elements of the covariance matrix (Bauwens et al., 2006). The key assumption of these models is that an observed vector of returns can be expressed as a non-singular linear transformation of uncorrelated latent factors. The time-varying variance dynamics of these factors can then be modelled and estimated individually, typically by a univariate GARCH model. In a restricted version only a subset of the latent factors has a time-varying conditional variance (Lanne and Saikkonen, 2007).

Despite the rich covariance and correlation structure exhibited by BEKK and DCC, their dynamics are specified as linear functions of past information. Aided by recent advances in machine learning (ML) methods, a growing stream of literature has been exploring the use of nonlinear ML-based volatility models and time series dynamics in general. In particular, Recurrent Neural Networks (RNNs) have shown remarkable ability in detecting both short-term and long-term dependencies in time series applications in various fields, such as natural language processing, speech and handwriting recognition, genotype sequencing, and drug response prediction (Abraham and Kumar, 2023). The demonstrated success of RNNs at capturing dynamic dependence patterns along with feasibility of implementation in accessible software libraries (Chollet, 2015; Gulli and Pal, 2017) have fuelled the influx of ML approaches to time series and volatility modelling (Hewamalage et al., 2021). RNN models have been shown to generate predictions that outperform mainstream variants of the univariate GARCH family in stock price volatility (Luo et al., 2018) or commodity price volatility (Vidal and Kristjanpoller, 2020). In particular, the RNN variant Long Short Term Memory neural networks (LSTMs) that alleviate the vanishing gradient problem in RNN optimization have shown remarkable ability in detecting both short-term and long-term dependencies. LSTMs have also been shown to be more accurate for predicting volatility than feedforward neural networks that are not well-suited for time series analysis (Kim and Won, 2018).

However, unlike the univariate volatility literature, analysis of nonlinear dynamics in conditional covariance matrices of multivariate models has been relatively sparse. A recent survey paper of neural network–based financial volatility forecasting by Ge et al. (2022) found only one study among the predominantly univariate field, Bucci (2020), that predicted a volatility covariance matrix with RNNs. Its application to daily returns on three assets shows an improvement in out-

of-sample forecasting accuracy over benchmark models in terms of mean absolute error and root mean square error. Subsequently, Liu et al. (2022) followed a similar approach and in an application to five technology stocks found that RNN-based models generated higher mean portfolio returns than those from the DCC model. Both papers apply RNNs to the lower-triangular matrix of the Cholesky decomposition of the covariance matrix, which limits the number of potential assets due to the curse of dimensionality.

Boulet (2021) notes that rather than trying to predict the whole conditional covariance matrix with a neural network model, it seems more promising to follow a "hybrid" aproach that combines neural nets with a multivariate GARCH volatility decomposition such as the CCC or the DCC. He proposes several variants of a hybrid model with a DCC conditional covariance decomposition and LSTM dynamics of individual volatilities. The hybrid models are shown to improve on the DCC in terms of several Minimum Variance Portfolio performance metrics on a large sample of several hundred assets. In the univariate volatility literature hybrid models have also been outperforming pure neural network specifications (Kim and Won, 2018; Ge et al., 2022). Notably, a hybrid Exponential Smoothing – LSTM model became the winner of a recent M4 forecasting competition (Smyl, 2020).

In this paper we develop the hybrid multivariate GARCH – RNN approach by modeling latent factors in a multivariate GO-GARCH model with LSTM dynamics. Specifically, we adopt the Boswijk and van der Weide (2011) GO-GARCH framework that was designed to balance generality against ease of estimation. The model uses a three-step estimation method that is easy to implement and is numerically attractive relative to DCC-based hybrid models. The first two steps consist of a method of moments estimator for the linear transformation that only involves iterated matrix rotations free of numerical convergence problems regardless of the dimension. The third step involves estimation of univariate LSTM models for each of the factors instead of parametric GARCH-type models used in the original GO-GARCH model. As a parsimonious special case we also consider a model variant where where all factors share a common LSTM cell.

Instead of attempting to capture the full complexity of a high-dimensional Cholesky decomposition used in the aforementioned studies, the factor decomposition of the GO-GARCH framework enables

us to focus the flexible nonlinear RNN models directly on the low-dimensional univariate dynamics of the latent factors. As a result, we can greatly expand the overall portfolio size from a handful to potentially hundreds of assets.

A potential drawback of GO-GARCH framework is that the latent factors do not have a direct economic interpretation, just like the RNNs that we use to model the factors. Lack of interpretability is a typical feature of the machine learning literature which is generally focused on reduced-form predictive performance rather than structural outcomes, with the former being our perspective as well.

We apply our model to a portfolio of 100 assets and compare its performance with relevant benchmark models in a Minimum Variance Portfolio (MVP) scenario. Our model variant with an individual LSTM for each factor achieves the best outcome by minimizing a key standard deviation metric, while the variant with a common LSTM for all factors is shown to perform favourably in terms of complementary metrics.

The paper is organized as follows. In Section 2 we review the GO-GARCH model, the RNN framework, and based on these formulate the new hybrid model. In Section 3 we lay out the details of the MVP scenario, discuss our application, and perform model comparison with relevant benchmark models. Section 4 concludes.

2 Model

2.1 GO-GARCH Model

Consider an m-vector time series $\{x_t\}_{t\geq 1}$ that represents a vector of returns on m different assets. We assume that a conditional mean has been subtracted from x_t so that $E(x_t|\mathcal{F}_{t-1}) = 0$ where $\{\mathcal{F}_{t-1}\}_{t\geq 0}$ denotes the filtration generated by $\{x_t\}_{t\geq 1}$. The GO-GARCH model imposes a structure on the conditional variance matrix $\Sigma_t = var(x_t|\mathcal{F}_{t-1}) = E(x_t x_t'|\mathcal{F}_{t-1})$ implied by

$$x_t = Z y_t = Z H_t^{1/2} \varepsilon_t \tag{1}$$

with $H_t = diag(h_{1t}, \ldots, h_{mt})$ where Z is an $m \times m$ non-singular matrix, $\{\{h_{it}\}_{t\geq 1}, i = 1, \ldots, m\}$ are positive, $\{\mathcal{F}_{t-1}\}$-adapted processes with $E(h_{it}) = 1$, and $\{\varepsilon_t\}_{t\geq 1}$ is a vector martingale difference sequence with $E(\varepsilon_t|\mathcal{F}_{t-1}) = 0$ and $var(\varepsilon_t|\mathcal{F}_{t-1}) = I_m$.

The model implies that the observed vector of returns x_t can be written as a non-singular transformation of a latent vector process y_t with $dim(x_t) = dim(y_t) = m$, whose components y_{it} are conditionally uncorrelated, satisfying $E(y_{it}|\mathcal{F}_{t-1}) = 0$, $var(y_{it}|\mathcal{F}_{t-1}) = h_{it}$, $cov(y_{it}, y_{jt}|\mathcal{F}_{t-1}) = 0$ for $i \neq j = 1, \ldots, m$. Another model implication is covariance stationarity of y_t and hence x_t, with

$$\Sigma_t = var(x_t|\mathcal{F}_{t-1}) = ZH_tZ'$$
$$\Sigma = var(x_t) = ZZ'.$$

In the original GO-GARCH model of van der Weide (2002) the conditional variances h_{it} were assumed to follow the GARCH(1,1) process

$$h_{it} = (1 - \alpha_i - \beta_i) + \alpha_i y_{i,t-1}^2 + \beta_i h_{i,t-1}, \quad \alpha_i, \beta_i \geq 0, \ \alpha_i + \beta_i < 1. \tag{2}$$

Fan et al. (2008) and Boswijk and van der Weide (2011) considered a more flexible structure, where h_{it} may depend on $y_{j,t-k}, j \neq i, k \geq 1$, specified as

$$h_{it} = \left(1 - \sum_{j=1}^{m} \alpha_{ij} - \beta_i\right) + \sum_{j=1}^{m} \alpha_{ij} y_{j,t-1}^2 + \beta_i h_{i,t-1}, \quad \alpha_{ij}, \beta_i \geq 0, \ \sum_{j=1}^{m} \alpha_{ij} + \beta_i < 1. \tag{3}$$

Boswijk and van der Weide (2011) emphasize that the model structure (1) along with their estimation method allows for various other specifications of the conditional variance process. Instead of (2) or (3) we propose to leverage the dynamic properties of an LSTM RNN cell detailed in the next Section in modeling the conditional variances.

2.2 Recurrent Neural Network Model of Volatility

RNNs are mathematical models structured in layers of connected functional units called cells. The connections between nodes can create a cycle, allowing output from some nodes to affect subsequent input to the same nodes. This feature allows RNNs to exhibit temporal dynamic behavior and

capture dependencies in sequential data. Although the theoretical foundations have been developed much earlier (Amari, 1972; Hopfield, 1982; Elman, 1990), real-time practical implementation of RNNs in wide-scale applications was only enabled in the mid-2000s by advances in machine learning methodology. Since then, RNNs have been widely applied in areas such as language modelling, machine translation, speech recognition, and time series analysis (Yu et al., 2019).

An standard RNN cell takes the form

$$\begin{aligned} a_t &= s(w_f a_{t-1} + w_z z_t + b) \\ q_t &= a_t \end{aligned} \tag{4}$$

where z_t, a_t, and q_t denote the input, a recurrent latent state, and the output of the cell at time t, respectively, w_f, w_z are the model parameters (so-called weights), b is the intercept (so-called bias), and $s(\cdot)$ denotes the so-called activation function that renders the model nonlinear. Popular forms of $\sigma(\cdot)$ include the sigmoid function

$$s(x) = \sigma(x) \equiv \frac{1}{1 + e^{-x}}$$

and the tanh function

$$s(x) = tanh(x) \equiv \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

A dynamic linear model with a recurrent hidden state results from (4) as a special case when $s(x) = x$.

RNNs that only contain the standard cells are not capable of handling long-term dependencies; as the time distance between the related input grows it is difficult for the standard cell to retain the connecting information. The Long Short Term Memory (LSTM) version of RNN (Hochreiter and Schmidhuber, 1997) feature an enhanced memory retention capacity by introducing "gate"

functions into the standard cell. The LSTM cell can be expressed as follows:

$$f_t = \sigma(w_{fa} a_{t-1} + w_{fz} z_t + b_f) \tag{5}$$

$$n_t = \sigma(w_{na} a_{t-1} + w_{nz} z_t + b_n) \tag{6}$$

$$c_t = tanh(w_{ca} a_{t-1} + w_{cz} z_t + b_c) \tag{7}$$

$$d_t = f_t \cdot d_{t-1} + n_t \cdot c_t \tag{8}$$

$$o_t = \sigma(w_{oh} a_{t-1} + w_{oz} z_t + b_o) \tag{9}$$

$$a_t = o_t \cdot tanh(c_t) \tag{10}$$

$$q_t = a_t \tag{11}$$

where the functions f_t, n_t and o_t denote the forget, input, and output gates, respectively. The forget gate function determines what information will be retained or discarded from the latent recurrent cell state (Yu et al., 2019).

We propose to model the conditional variances h_t of the GO-GARCH model of Section 2.1 with the LSTM cell (5)-(11), using $h_t = a_t$ from (10) as the latent recurrent state with sequence of inputs $z_t = y_{t-1}^2$. We consider two versions of the hybrid model. First, a "GO-LSTM" version where each latent factor i is assigned its own LSTM cell, and second, a "GO-LSTM-Common" version where all factors share a common LSTM cell. The former features greater flexibility at the expense of more parameters to be estimated while the latter imposes a common restriction on the factor dynamics with computationally cheaper relative parsimony. The model nests the traditional GO-GARCH with GARCH(1,1) conditional variances (2) as a special case in the absence of gate functions and with a linear activation function $s(x) = x$.

2.3 Estimation and Statistical Properties

Boswijk and van der Weide (2011) note that likelihood-based estimation of the GO-GARCH model family would require numerical maximization of a criterion function over a high-dimensional parameter space. As an alternative, they propose a method of moments estimator that only requires the calculation of common eigenvectors of a sequence of sample moment matrices and "can be applied to arbitrary dimensions m". We adopt the same modelling framework and estimation ap-

proach except that in the last step we extend their estimation of univariate GARCH models to more flexible but still parametric univariate RNN models.

Boswijk and van der Weide (2011) established consistency of their estimator under the following Assumptions.

Assumption 1 *The process $x_{tt\geq 1}$ satisfies the representation*

$$x_t = Zy_t = ZH_t^{1/2}\varepsilon_t,$$
$$H_t = diag(h_{1t}, \ldots, h_{mt})$$

where Z is an $m \times m$ non-singular matrix, where $h_{itt\geq 1}, i = 1, \ldots, m$ are positive, $\mathcal{F}_{t-1}$-adapted processes with $E(h_{it}) = 1$ and where $\varepsilon_{t\geq 1}$ is a vector martingale difference sequence, with $E(\varepsilon_t|\mathcal{F}_{t-1}) = 0$ and $Var(\varepsilon_t|\mathcal{F}_{t-1}) = I_m$.

Assumption 2 *The process $x_{tt\geq 1}$ is strictly stationary and ergodic, and has finite fourth moments $\kappa_i = E(y_{it}^4) < \infty, i = 1, \ldots, m$. Furthermore, the autocorrelations $\rho_{ik} = corr(y_{it}^2, y_{i,t-k}^2)$ and cross-covariances $\tau_{ijk} = cov(y_{it}^2, y_{i,t-k}y_{j,t-k})$ satisfy, for some integer p,*

$$\min_{1\geq i\geq m} \max_{1\geq k\geq p} |\rho_{ik}| > 0, \qquad \max_{1\geq k\geq p, 1\geq i\geq j\geq m} |\tau_{ijk}| = 0.$$

Assumption 3 *In the model defined by Assumptions 1 and 2,*

$$\max_{1\geq k\geq p, 1\geq i\geq j\geq m} |\rho_{ik} - \rho jk| > 0.$$

Assumption 1 defines a representation of the observed vector of returns x_t in terms of a non-singular transformation of a latent vector process y_t of the same dimension in the Generalized Orthogonal framework that we use here as well. Assumption 2 ensures the existence of kurtosis and non-zero autocorrelation of the latent processes y_t while excluding dependence in h_{it} on whether $y_{i,t-k}$ and $y_{j,t-k}$ have the same sign. Similarly to the Boswijk and van der Weide (2011) GARCH case, we verify the finiteness of kurtosis empirically in our application for our LSTM model, as detailed in

the Appendix. Boswijk and van der Weide (2011) note that "it would be hard to think of processes" that display volatility clustering but violate the non-zero autocorrelation condition of Assumption 2. Finally, Assumption 3 excludes the possibility that two squared components y_{it}^2 and y_{jt}^2 have the same autocorrelation function for $k = 1, \ldots, p$. As the probability of squares of components of y_t exhibiting exactly the same dynamic behaviour is deemed vanishingly small in applications, we maintain Assumption 3 as well, and can thus invoke the consistency result of Boswijk and van der Weide (2011).

An asymptotic distribution of the method of moments estimator under the GO framework can be obtained by a numerical bootstrap procedure (Hahn, 1996). It is not required for our application within the Minimum Variance Portfolio (MVP) framework that seeks to find the weights minimizing the conditional variance of a portfolio based on covariance predictions, and therefore not undertaken here.

3 Empirical Application and Model Comparison

In this Section, we apply our model to a portfolio of 100 assets and compare its performance with several relevant benchmark models using the Minimum Variance Portfolio (MVP) framework. The application framework relies on finding the weights $w_t = (w_{1,t}, \ldots, w_{m,t})$ which minimize the conditional variance of the portfolio $\sigma_{p,t}^2 \equiv var(x_{p,t})$ for a sample of daily portfolio returns $x_{p,t} = w_t' x_t$. The MVP without short selling solves the following minimization problem:

$$\min_{w_t} \sigma_{p,t}^2 = w_t' \Sigma_t w_t \text{ s.t. } w_t' \iota_m = 1,\ w_{i,t} \geq 0,\ i = 1, \ldots, m \tag{12}$$

where ι_m is a vector of ones. The problem has the analytical solution

$$w_t = \frac{\Sigma_t^{-1} \iota_m}{\iota_m' \Sigma_t^{-1} \iota_m}. \tag{13}$$

The MVP framework has the advantage of being entirely based on covariance predictions and is a common choice in the literature on the subject (Engle et al., 2019; Boulet, 2021). In practical implementations the unknown Σ_t is replaced with an estimator. We consider the following models for Σ_t:

- Equally weighted portfolio;
- DCC with $\Sigma_t = D_t R_t D_t$ where D_t is a diagonal matrix of univariate conditional variances, here modeled by $GARCH(1,1)$, and R_t is a matrix of conditional correlations;
- GO-GARCH (Boswijk and van der Weide, 2011);
- G-LSTM-DCC-OH where the forecast of D_t is obtained using an LSTM taking past conditional volatilities and $GARCH(1,1)$ features as inputs, best performing model of Boulet (2021);
- GO-LSTM-Common where each h_t in the GO-GARCH model (1) is modelled by the same LSTM process;
- GO-LSTM where each h_t in the GO-GARCH model (1) is modelled by a separate LSTM process.

We have not included the BEKK model among the benchmark models due to its severe curse of dimensionality that renders its implementation infeasible for the large number of assets in our application.

3.1 Data and Portfolio Construction

For our application we chose the top 100 assets from Yahoo finance as ranked by their market capitalization as of the end of the year 2022. We sought to preclude well-known biases arising from including smaller firms' assets (Roll, 1983). The data comprise the daily adjusted closing price, spanning the time period from January 2004 to December 2022. Log returns were calculated and normalized to the interval $[0,1]$ as is common in empirical implementation of neural networks to prevent gradient scaling problems.

The construction of our portfolios was inspired by the empirical exercise in Boulet (2021). The initial 85% of our data was used as a training sample and the remaining data, about 32 months comprised of 672 observations, were used to evaluate the models in a monthly rolling window fashion. We obtained forecasts of the conditional correlation and covariance matrices using the

standard definition of a month covering 21 trading days. Subsequently, we constructed MVPs along with metrics of their performance. This approach mimics evaluating the results from the stakeholder perspective. We re-evaluated each model on a monthly basis in contrast to the yearly frequency of Boulet (2021). Further technical details of the implementation are provided in the Appendix.

3.2 Results

The performance of the portfolios is compared using three different out-of-sample metrics: the annualized average return (AV), annualized standard deviation (SD), and the information ratio (IR) obtained as IR=AV/SD. Within the minimum variance portfolio framework, SD is the main metric of interest. The MVP weights (13) are designed to minimize the variance (and equivalently the standard deviation) rather than to maximize the expected return or the information ratio. Therefore, evaluation of the MVP implementation focuses on SD minimization. A high out-of-sample average return (AV) and a high out-of-sample information ratio (IR) are also desirable, but are considered of secondary importance from the perspective of evaluating the quality of a covariance matrix estimator (Engle et al., 2019).

The results for the performance metrics for our application data are reported in Table 1, with best outcomes for each metric highlighted in bold. The Equal Weights and DCC baseline models are dominated by the factor-based models in terms of the SD metric. Importantly, the GO-LSTM model displays the lowest portfolio SD while the GO-LSTM-Common model achieves the highest AV and IR.

Model	*AV*	*SD*	*IR*
Equal Weights	0.236	0.245	0.964
DCC	0.247	0.245	1.011
GO-GARCH	0.216	0.224	0.966
G-LSTM-DCC-OH	0.189	0.237	0.800
GO-LSTM-Common	**0.268**	0.233	**1.148**
GO-LSTM	0.227	**0.222**	1.024

Table 1: Portfolio measures

Further insights into the relative model performance can be gleaned by examining the portfolio

weights assigned to each asset. Figure 1 shows the extent of heterogeneity in the portfolio asset SD (left) and the asset AV (right). The asset SD and AV exhibit small but significant positive correlation, reflecting a trade-off between SD minimization and AV maximization.

The assets in Figure 1 are indexed by the rank of the their market capitalization (Apple 1, Microsoft 2, etc.) and the reordering of their indices reflects their relative position in terms of SD. Using this SD-based index on the x-axis, Figure 2 presents the weights that the factor-based models allocate to each asset. The best-performing GO-LSTM model that minimizes the overall portfolio SD allocates more weight to low SD assets than other models. The GO-LSTM-Common model appears to achieve its AV and IR maximization due to a more even distribution of weights over the SD ranking. The trailing DCC and G-LSTM-DCC-OH models allocate weight to clusters of observations with higher SD without capitalizing on assets with higher AV.

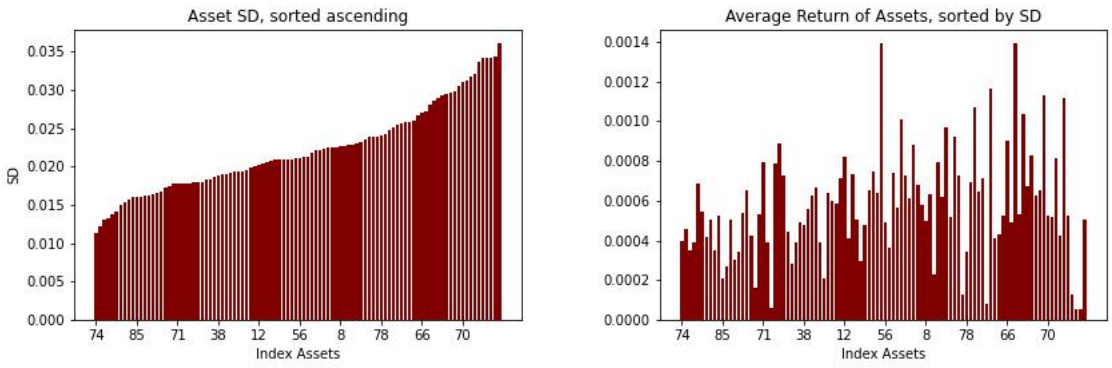

Figure 1: Standard Deviation (SD) and Average Returns (AV)

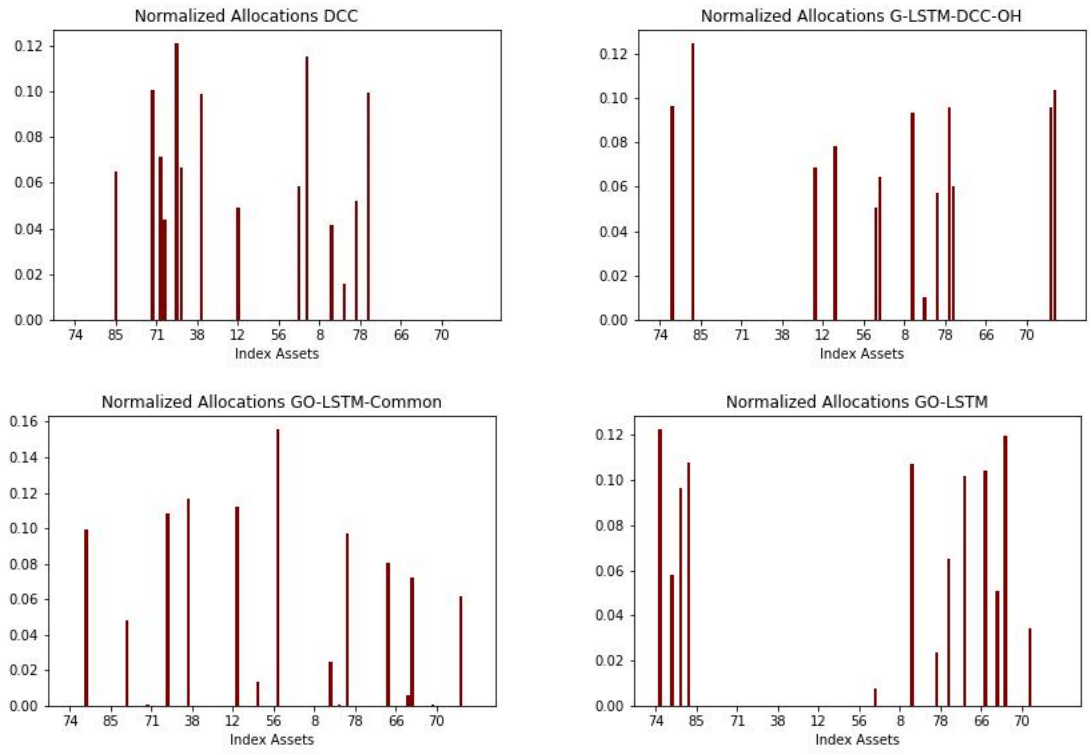

Figure 2: Portfolio Weights Allocation

4 Conclusion

In this paper we developed a hybrid model of multivariate volatility that employs the LSTM variant of Recurrent Neural Networks for fitting the conditional variances of latent orthogonal factors in a GO-GARCH framework. Relative to hybrid RNN approaches based on Cholesky factorization of the conditional covariance matrix or the DCC decomposition our approach is easy to implement and can be used to model conditional covariances of a large number of assets. Our model is shown to perform favourably in comparison to relevant benchmark models in a Minimum Variance Portfolio (MVP) scenario on a portfolio of 100 assets.

5 Appendix

5.1 Implementation Details

The GO-LSTM and GO-LSTM-Common models were implemented as in Boswijk and van der Weide (2011, Summary 1) except instead of their last the GARCH-type models were replaced with the LSTM cell (5)-(11). LSTM layers and hyperparameters were tuned using a Bayesian optimization algorithm as described in (Snoek et al., 2012) in contrast to Boulet (2021) who uses values given by previous literature. The algorithm has the advantage of searching across the most promising parameter space, and does not require a full evaluation of the neural network, which speeds up cross-validation significantly. For the factor decomposition we used the method of moments estimator of the Flurry-Gautschi algorithm. The neural network parameters were estimated using the back-propagation algorithm, which is a popular modern machine learning method (Nielsen, 2015, chapter 2). This procedure calculates the derivative of the loss function of the neural network model with respect to each parameter. Since neural nets can be seen as the composition of as many functions as layers, this derivative is "propagated" by use of the chain rule (Hastie et al., 2009, Chapter 11). The derivative value is then used to update the network parameters in each layer over many iterations until the model loss function is minimized the at the output layer.

As is typical in the machine learning literature, the back-propagation algorithm was implemented with early stopping and an adaptive learning rate adjustment to preclude model overfitting and a vanishing gradient vector. These monitor the loss function across training iterations and are known in the literature to boost performance significantly. We used the common 85%/15% training to test set ratio.

The model was compiled with an ADAM optimizer that takes information from the first two moments of the gradient into account. The activation function, the compiler, and the architecture itself were all tuned using Bayesian optimization, which seeks to attain optimal precision. The parameters are as follows: patience = 6, maximum epochs = 32, batch size = 40. The best performing architecture was an LSTM layer with 249 neurons using a tanh activation function, with a 0.25 dynamic drop-out. This was followed by 4 fully connected layers with 446, 314, 367,

26 neurons, followed by the output layer. All fully connected layers employed the RELU activation function. After each layer, a dropout stage with $p = 0.6$ proved most efficient. The learning rate starting at 0.006 was adjusted at a rate of 0.001 across iterations, with the MSE used as loss function. The implementation was run with a Python code on a 40-core 2.4 GHz Linux cluster (Loken et al., 2010; Ponce et al., 2019).

5.2 Verification of the GO-LSTM Assumption

Before implementing the model, we tested Assumption 2 of Boswijk and van der Weide (2011) by empirically verifying the sufficient condition of a finite kurtosis for independent GARCH processes (He and Teräsvirta, 1999) when translated into the LSTM stochastic evolution. Due to computational constraints, this test was performed on a random subset of 10 assets. An LSTM model was fit on the individual assets and the fitted model was then used to simulate the evolution of the stochastic process for further 10,000 time periods. The kurtosis values were then calculated for this simulated time series, confirming the validity of the assumptions necessary for GO-LSTM (Boswijk and van der Weide, 2011; He and Teräsvirta, 1999).

References

Abraham, A. and T. A. Kumar (2023). *Recurrent neural networks : concepts and applications*. CRC Press.

Aielli, G. P. (2013). Dynamic conditional correlation: On properties and estimation. *Journal of Business & Economic Statistics 31*(3), 282–299.

Amari, S. I. (1972). Learning patterns and pattern sequences by self-organizing nets of threshold elements. *IEEE Transactions on Computers 21*(11), 1197–1206.

Bauwens, L., S. Laurent, and J. V. K. Rombouts (2006). Multivariate GARCH models: a survey. *Journal of Applied Econometrics 21*(1), 79–109.

Boswijk, H. P. and R. van der Weide (2011). Method of moments estimation of go-garch models. *Journal of Econometrics 163*(1), 118–126. Factor Structures in Panel and Multivariate Time Series Data.

Boudt, K., A. Galanos, S. Payseur, and E. Zivot (2019). Chapter 7 - multivariate GARCH models for large-scale applications: A survey. In H. D. Vinod and C. Rao (Eds.), *Conceptual Econometrics Using R*, Volume 41 of *Handbook of Statistics*, pp. 193–242. Elsevier.

Boulet, L. (2021). Forecasting high-dimensional covariance matrices of asset returns with hybrid garch-

lstms.

Bucci, A. (2020). Cholesky-ann models for predicting multivariate realized volatility. *Journal of Forecasting 39*(6), 865–876.

Burda, M. and J. M. Maheu (2012). Bayesian adaptively updated Hamiltonian Monte Carlo with an application to high-dimensional BEKK GARCH models. *Studies in Nonlinear Dynamics & Econometrics 17*(4), 345–372.

Caporin, M. and M. McAleer (2012). Do we really need both BEKK and DCC? a tale of two multivariate GARCH models. *Journal of Economic Surveys 26*(4), 736–751.

Chollet, F. (2015). Keras. `https://github.com/fchollet/keras`.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science 14*(2), 179–211.

Engle, R. F. (2002). Dynamic conditional correlation: A simple class of multivariate generalized autoregressive conditional heteroskedasticity models. *Journal of Business and Economic Statistics 20*, 339–350.

Engle, R. F. and K. F. Kroner (1995). Multivariate simultaneous generalized ARCH. *Econometric Theory 11*(1), 122–150.

Engle, R. F., O. Ledoit, and M. Wolf (2019). Large dynamic covariance matrices. *Journal of Business & Economic Statistics 37*(2), 363–375.

Engle, R. F., V. K. Ng, and M. Rothschild (1990). Asset pricing with a factor-arch covariance structure: Empirical estimates for treasury bills. *Journal of Econometrics 45*(1), 213–237.

Fan, J., M. Wang, and Q. Yao (2008, 02). Modelling multivariate volatilies via conditionally uncorrelated components. *Journal of the Royal Statistical Society Series B 70*, 679–702.

Ge, W., P. Lalbakhsh, L. Isai, A. Lenskiy, and H. Suominen (2022, jan). Neural network-based financial volatility forecasting: A systematic review. *ACM Comput. Surv. 55*(1), 1–30.

Gulli, A. and S. Pal (2017). *Deep learning with Keras*. Packt Publishing Ltd.

Hahn, J. (1996). A note on bootstrapping generalized method of moments estimators. *Econometric Theory 12*(1), 187–197.

Hastie, T., R. Tibshirani, J. H. Friedman, and J. H. Friedman (2009). *The elements of statistical learning: data mining, inference, and prediction*, Volume 2. Springer.

He, C. and T. Teräsvirta (1999). Properties of moments of a family of garch processes. *Journal of Econometrics 92*(1), 173–192.

Hewamalage, H., C. Bergmeir, and K. Bandara (2021). Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting 37*(1), 388–427.

Hochreiter, S. and J. Schmidhuber (1997). Long short-term memory. *Neural Computation 9*(8), 1735–1780.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America 79*(8), 2554–2558.

Kim, H. and C. Won (2018, August). Forecasting the volatility of stock price index: A hybrid model integrating lstm with multiple garch-type models. *Expert Systems with Applications 103*, 25–37.

Lanne, M. and P. Saikkonen (2007). A multivariate generalized orthogonal factor garch model. *Journal of Business & Economic Statistics 25*(1), 61–75.

Liu, W. K., M. K. So, and A. M. Chu (2022). Dynamic covariance modeling with artificial neural networks. *Communications in Statistics: Case Studies, Data Analysis and Applications 8*(1), 15–42.

Loken, C., D. Gruner, L. Groer, R. Peltier, N. Bunn, M. Craig, T. Henriques, J. Dempsey, C.-H. Yu, J. Chen, L. J. Dursi, J. Chong, S. Northrup, J. Pinto, N. Knecht, and R. V. Zon (2010). Scinet: Lessons learned from building a power-efficient top-20 system and data centre. *Journal of Physics: Conference Series 256*(1), 012026.

Luo, R., W. Zhang, X. Xu, and J. Wang (2018, February). A neural stochastic volatility model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Volume 32, Palo Alto, CA, pp. 123–130. AAAI: AAAI Press.

Nielsen, M. A. (2015). *Neural Networks and Deep Learning*. Determination Press.

Ponce, M., R. van Zon, S. Northrup, D. Gruner, J. Chen, F. Ertinaz, A. Fedoseev, L. Groer, F. Mao, B. C. Mundim, M. Nolta, J. Pinto, M. Saldarriaga, V. Slavnic, E. Spence, C.-H. Yu, and W. R. Peltier (2019). Deploying a top-100 supercomputer for large parallel workloads: The niagara supercomputer. In *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (Learning)*, PEARC '19, New York, NY, USA. Association for Computing Machinery.

Roll, R. (1983). On computing mean returns and the small firm premium. *Journal of Financial Economics 12*(3), 371–386.

Silvennoinen, A. and T. Teräsvirta (2009). Multivariate GARCH models. In T. Mikosch, J.-P. Kreiß, R. A. Davis, and T. G. Andersen (Eds.), *Handbook of Financial Time Series*, pp. 201–229. Springer Berlin Heidelberg.

Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting 36*(1), 75–85. M4 Competition.

Snoek, J., H. Larochelle, and R. P. Adams (2012). Practical bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems 25*, 2951–2959.

van der Weide, R. (2002). GO-GARCH: A multivariate generalized orthogonal GARCH model. *Journal of Applied Econometrics 17*(5), 549–564.

Vidal, A. and W. Kristjanpoller (2020). Gold volatility prediction using a cnn-lstm approach. *Expert Systems with Applications 157*, 113481.

Yu, Y., X. Si, C. Hu, and J. Zhang (2019, 07). A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. *Neural Computation 31*(7), 1235–1270.