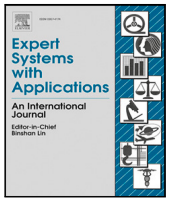
Expert Systems With Applications 229 (2023) 120418

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa



Volatility forecasting with hybrid neural networks methods for Risk Parity investment strategies

Luca Di Persio [a], Matteo Garbelli [a,b,*], Fatemeh Mottaghi [c], Kai Wallbaum [d]

[a] Department of Computer Science, University of Verona, Strada le Grazie 15, Verona, 37134, Italy
[b] Department of Mathematics, University of Trento, Via Sommarive 14, Povo (Trento), 38123, Italy
[c] Department of Mathematics, Iran University of Science and Technology, Tehran, Iran
[d] RiskLab, Allianz Global Investors, Seidlstraße 24, 80335, Munchen, Germany

ARTICLE INFO

Keywords:
Volatility
Forecast
Time series
Neural Networks
Risk Parity

ABSTRACT

We present a hybrid method for computing volatility forecasts that can be used to implement a risk-controlled strategy for a multi-asset portfolio consisting of both US and international equities. Recent years have been characterized by extremely low yields, with 2022 marked by rising interest rates and an increasing inflation rate. These factors produced new challenges for both private and institutional investors, including the need for robust forecast methods for financial assets' volatilities. Addressing such task, our research focuses on a hybrid solution that combines classical statistical models with specific classes of Recurrent Neural Networks (RNNs). In particular, we first use the Generalized Autoregressive Conditional Heteroscedasticity (GARCH) approach within the preprocessing phase to capture volatility clustering, striking an efficient balance between computational effort and accuracy, to then apply RNN architectures, namely GRU, LSTM, and a mixed model with both units, as to maximize performances of volatility forecasts later used as input factors for risk-controlled investment strategies. In terms of portfolio allocation, we focus on a simplified version of the Risk Parity method that was first proposed by the Research division of S&P Global. This version ignores the contribution of cross-correlations among assets, nevertheless providing encouraging results. Indeed, we show the effectiveness of the chosen approach by providing forward-looking risk parity portfolio strategies that outperform standard risk/return portfolio structures.

1. Introduction

The capital market environment of the last ten years has been characterized by falling interest rate levels and rising equity markets. Furthermore, during last 12 months, volatility levels have increased significantly as a result of rising inflation risks and growing interest rate levels in developed markets. As a consequence, efficient risk management techniques are getting an increasing attention since both institutional and private investors are looking for new ways to assessing the risks produced by markets' change.

Accordingly, we focus our attention on the so-called *risk parity portfolio* looking for new approaches to lower risk levels linked to equity markets by means of better forecast computed by Machine Learning (ML) approaches. We remark that risk-parity techniques form the foundation of many portfolio allocation approaches for specific asset classes and equity indices. These methods consider the relative risk contribution of each asset, which is typically measured by its realized volatility, to the overall risk of the portfolio.

As a consequence, an asset class with low risk level, i.e. lower realized volatility, has a higher weight in the overall portfolio if compared with an alternative characterized by higher risk levels, respectively higher volatility. Moving from this landmark, in our analysis we consider hybrid solutions to estimate risk levels by employing both classical statistical model, namely Generalized Autoregressive Conditional Heteroscedasticity (GARCH) type models, and Neural Networks (NNs) methods, specifically different classes of Recurrent Neural Networks (RNNs) architectures, to forecast volatility levels as input factor for a portfolio allocation.

As a result, we obtain an improved performance of the forward-looking risk estimators showing better risk-return profiles for investment strategies. Our main contributions can be summarized as follows:

- We use different RNN architectures (LSTM, GRU and a model with mixed hidden units) combined with a *faster* forecast computed by a GARCH model to provide accurate and robust volatility

* Corresponding author at: Department of Mathematics, University of Trento, Via Sommarive 14, Povo (Trento), 38123, Italy.
E-mail addresses: luca.dipersio@univr.it (L. Di Persio), matteo.garbelli@unitn.it (M. Garbelli), mottaghi@alumni.iust.ac.ir (F. Mottaghi), Kai.Wallbaum@allianzgi.com (K. Wallbaum).

https://doi.org/10.1016/j.eswa.2023.120418
Received 29 November 2022; Received in revised form 6 May 2023; Accepted 6 May 2023
Available online 12 May 2023

predictions. In particular, we use the GARCH model at the level of pre-processing stage to get information about the position of volatility clusters, while the RNNs architectures are later used to improve the precision of the forecasts themselves;
- We perform a comparison analysis between different RNNs solutions in terms of their performances as well as considering a convex ensemble of them;
- Later, we move from volatility forecast to feed a data-driven version of a Risk Parity portfolio firstly presented in Berlinda et al. (2001), in a historical perspective for the reference period of October 2020–December 2022 where we consider both weekly and monthly re-balancing procedures.

Organization of the paper

The article is divided into five sections. In Section 2, we provide details about the general Risk Parity framework and recall basic facts about both the GARCH process and NN tools. We also present an in-depth discussion of GRU and LSTM hidden units, as well as the specific models we use in our simulation. In Section 3, we present the numerical results obtained for volatility forecasting of the S&P 500 index, which serves as our reference, and conduct a comparison analysis among different ML architectures. In Section 4, we present our findings regarding the risk-adjusted strategy in light of the obtained ML-based volatility estimates. We then perform a historical simulation of the Risk Parity portfolio and re-balancing on both a weekly and monthly basis. Finally, in Section 5, we provide our concluding remarks, highlighting the most interesting aspects of our research and suggesting possible future research directions.

1.1. Related literature

Over the last years, ML, and in particular Artificial Neural Networks (ANNs) solutions have received an increasing attention within a plethora of industrial sectors. Indeed, the fast development of efficient, reliable and relatively not expensive, hardware solutions, such as ensemble, distributed architectures or the use of combined GPUs, has offered the opportunity to implement a series of well known ML-based solutions for the solution of *practical industrial tasks*. This is the case, in particular, of the applications of forecasting solutions in the sectors of manufacturing, transportation, pharmacy as well as within the financial one.

Regarding the implications of machine learning (ML) in finance, it is nearly impossible to provide even a partial summary of the vast amount of studies that have been conducted by practitioners, mathematicians, and computer scientists from diverse backgrounds. Notably, the monographs by Cohen and Dixon serve as important landmarks in this field. Additionally, it is worth noting that there is a considerable body of literature that explores the integration of recurrent neural networks (RNNs), particularly Long Short-Term Memory (LSTM) networks, with specific technologies across a wide range of fields. Examples include the use of LSTM networks for malware threat detection in Internet of Things (IoT) (Woźniak et al., 2021), energy consumption forecasting (Peng et al., 2021), automated diagnosis support in healthcare (Woźniak et al., 2023), and the prediction of air pollutant concentrations (Ju et al., 2022).

We will continue by recalling some of the most significant contributions in forecasting realized volatility. In Nguyen et al. (2019), the authors proposed a novel approach to model stochastic volatility dynamics by combining Stochastic Volatility (SV) models with Long Short-Term Memory (LSTM). This resulted in the development of the LSTM-SV model, which overcomes the short-term memory problem encountered when dealing with numerical approximations of financial time series where the volatility term is modeled through a stochastic process. The proposed model has the ability to capture non-linear dependence in the latent volatility process, providing better forecasting performance than standard SV solutions, such as approximating Heston-type models using Monte Carlo approaches.

In Lu et al. (2016), the authors analyzed the performance of different hybrid models that combine Artificial Neural Networks (ANN) with GARCH-type models for volatility forecasting. They demonstrated that the GARCH-ANN model performs better than standard solutions for volatility forecasting in a Chinese energy market index. In Kristjanpoller and Minutolo (2018), a similar approach was applied with a pre-processing stage to forecast price volatility levels of cryptocurrencies like Bitcoin.

In Elsworth and Güttel (2020), a combination of a Recurrent Neural Network (RNN) with a dimension-reducing symbolic representation was applied to overcome two fundamental limitations of Machine Learning (ML) methods trained on raw numerical time series data: high sensitivity to hyperparameters and the initialization of random weights. The proposed solution was used for time series forecasting.

Finally, in Kim and Won (2018), the authors proposed a new hybrid model that combines the LSTM model with various generalized GARCH-type models to forecast stock price volatility. The provided solution showed promising results.

Concerning the integration of LSTM with existing well-known financial method, we mention Mao et al. (2016) where the authors integrate the Merton determinants model to forecast the credit default swap. Another example of a modification of LSTM is the modified adaptive LSTM model (consisting of two LSTM layers followed by a pair of batch normalization (BN) layers, a dropout layer and a binary classifier) presented in Fang et al. (2023). We also cite Ribeiro et al. (2022), Stefenon et al. (2022) to have an overview of possible ensemble techniques following a comparison analysis.

More recently, in Zhang et al. (2022), the authors provide a comparison between different ML method finding that NNs dominate linear regressions and tree models for predicting realized volatility, due to their ability to model complex uncovered interactions in time series. Further, in Di Persio et al. (2021), a VolTarget approach has been proposed as a risk-adjusted procedure alternative to Risk Parity, moving from historical realized volatility values then exploiting LSTM steered solutions, see also Albeverio et al. (2013, 2018), Cao et al. (2020) for further details. Previous examples are worth to be blended with efforts conducted within the so-called *Risk Parity and Budgeting* approach, see Roncalli (2013), as a general (set of) method(s) to treat the Risk Parity setting, namely the scenario we are dealing with.

2. Methodology and Neural Networks models

Financial applications such as risk assessment and forecasting have all benefited from the widespread use of ML, particularly because of the ability of NNs-based algorithms to find connections among data points in view of reconstructing time-series patterns of interest. Throughout this section, we provide a theoretical description of models and methods that we implemented to obtain an effective and robust Risk Parity strategy.

2.1. The Risk Parity strategy

A Risk Parity portfolio aims at equalizing the risk allocation across asset classes by over-weighting safer assets and under-weighting riskier ones in terms of their weights in the market portfolio. There exist different ways and methods to construct a risk parity portfolio, basically depending on how we measure and define risk. It is worth mentioning that risk parity weighting does not require investors to make assumptions about expected returns representing a significant advantage over mean–variance optimization.

In this article, we consider a multi-asset portfolio formed by N equities indexed by $i = \{1, \dots, N\}$ whose risk we assume to be proportional to their historical volatility σ_i.

The log returns of the ith asset are calculated as a time series r_t^i by

$$r_t^i = \log p_{t+1}^i - \log p_t^i . \tag{1}$$

with p_t^i being the value of daily closing price evaluated of equity ith at time t. The log returns series are used as opposed to the daily closing prices as they often fit a Gaussian distribution (Andersen et al., 2001).

We measure the corresponding historical volatility σ_t^i as the standard deviation of the log return process conditioned to the σ-algebra $\mathcal{F}_{t-1}^i$ generated by log returns $r_0^i, \dots, r_{t-1}^i$. Specifically, the volatility time series are computed as the standard deviation over n days, i.e. the number of samples within that time window, of the log return series, over the time window given by the interval $[\tau, \tau + n]$

$$\sigma_t^i = \sqrt{\frac{1}{n}\sum_{t=\tau}^{\tau+n}\left(r_t^i - \mu_{\tau,\tau+n}^i\right)^2}, \tag{2}$$

where $\mu_{\tau,\tau+n}^i = \frac{1}{n}\sum_{t=\tau}^{\tau+n} r_t^i$ is the mean of the log returns within a time window of n days. Throughout the article, we set n equal to 21, dealing with 21 days standard deviations.

Generally speaking, the marginal contribution MC_i of the ith asset to the total risk of the portfolio is computed by

$$MC_t^i = w_t^i \sigma_t^i \beta_t^i \tag{3}$$

where the coefficients β_i depends on all the assets in the portfolio by means of the covariance matrix

$$\beta_t^i = \frac{\mathrm{Cov}(\sigma_t^i, \sigma_t^p)}{\left(\sigma_t^p\right)^2}. \tag{4}$$

being σ_t^p the overall volatility of the portfolio. The goal is to optimize all asset classes' weights ω^i to have equal marginal contributions MC^i for all t in the time horizon.

However, the computational complexity of the covariance matrix estimation is proportional to the square of the assets number, resulting in a computational, time consuming process. Thus, in this project, we use a simpler version of risk parity, introduced in Berlinda et al. (2001), to approximate Eq. (3). We construct the portfolio by neglecting the contribution of coefficient β_t^i, namely by dropping the assumption of correlation between equities. We assume each asset's weight to be proportional to the inverse of its standard deviation, i.e. the ith asset weight for each time step t is computed according to

$$w_t^i = \frac{\frac{1}{\sigma_t^i}}{\sum_{l=1}^{N}\frac{1}{\sigma_t^l}} \tag{5}$$

for a N equities portfolio. The result is that on a relative basis, lower volatile assets will have a higher weight than higher volatile assets, see, e.g., Berlinda et al. (2001), Chaves et al. (2011).

2.2. The GARCH(1,1) model

Modeling asset volatility is a crucial challenge since accurate forecasts of asset returns volatility contribute to a precise risk assessment. Some financial models, such as the Black–Scholes one, have been extensively used to estimate the fair price of European-style options, often overlooking the presence of heteroskedasticity related effects by assuming an homogeneous volatility. Latter assumption produces elegant closed-form formula, despite assuming the stationarity of variance is far from being able to realistically depict financial application especially within the *investment management arena*. To forecast prices and rates of financial instruments, in 1982 Robert F. Engle introduced the Auto-Regressive Conditional Heteroskedasticity (ARCH) model to provide a more realistic model to estimate financial market volatility, see, e.g., Engle (1982) for more details.

In this model, the log-returns r_t defined by Eq. (1) are computed in approximated form $\hat{r}_t$ as white noise multiplied by the historical volatility σ_t, while the conditional variance process is given by mean of an auto regressive structure

$$\hat{r}_t = \delta_t \hat{\sigma}_t, \tag{6}$$

$$\hat{\sigma}_t^2 = \omega + \sum_{i=1}^{p} \alpha_i \hat{r}_{t-i} \tag{7}$$

where δ_t are independent and identically distributed (i.i.d.) random variables with $\mathbb{E}(\delta_t) = 0$ and $Var(\delta_t) = 1$, independent of σ_k, for all $k \le t$. The lag length $p \ge 0$ is itself a parameter for the model, and the case $p = 0$ represents the *basic* scenario characterized by a white noise.

In 1986, Tim Bollerslev improved the ARCH model by allowing ${\sigma_t}^2$ to have its own autoregressive structure, see Bollerslev (1986), Teräsvirta (2009) for further details. The equation for the GARCH(p, q) (generalized ARCH) model is:

$$\hat{r}_t = \delta_t \hat{\sigma}_t, \tag{8}$$

$$\hat{\sigma}_t^2 = \omega + \sum_{i=1}^{p} \alpha_i \hat{r}_{t-i} + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2. \tag{9}$$

Practically, in applications, the GARCH$(1, 1)$ model has become widely used in financial time series modeling because of its relatively simple implementation. Indeed, the likelihood function is easier to manage if compared with continuous-time models, the latter needing numerical approximations for the solution of associated stochastic difference equations, see, e.g., Hansen and Lunde (2005), Kat and Heynen (1994), Williams (2011), for further details. Taking advantage of the GARCH $(1, 1)$ implementation agility, we use it to forecast the volatility of 4 equities index over the period 1999–2020. More precisely, we apply the GARCH-approach within the preprocessing phase aiming to capture the so-called volatility clustering with a good compromise in terms of computational efforts (computational time needed for the task) and accuracy. Indeed, it is known, see, e.g., Cont (2007), Hanck et al. (2019), that financial time series often exhibit a behavior where large price changes tend to cluster together, resulting in the persistence of the amplitudes of related financial movements.

Using historical data, the weighted sum of the observed volatility is computed by Eq. (12). As shown in Fig. 1, forecasts obtained by GARCH$(1, 1)$ are poor in terms of accuracy, while catching high variance peaks as well as corresponding volatility clusters.

2.3. The forecast task

We aim at deriving an algorithm able to exploit GARCH-based predictions to get information about volatility clusters, then entrusting a NN-steered solution to increase the accuracy of the forecasts.

The task of the forecast algorithm concerns approximating the non-linear mapping f represented as

$$\mathcal{Y}^i = f\left(\mathcal{X}^i\right) \tag{10}$$

overall the K batches indexed by ith in a iterative way. At this stage, we remark that a possible limitation of the model is the fact is assuming that is existing a function linking volatility inputs with targets over distinct batches.

We perform a rolling window mechanism to model each observation of the series on its past recent values, that called lags. Specifically, we can rewrite Eq. (10) at the level of each minibatch as

$$y_{[t,t+5]} = f\left(\mathbf{X}_{[t-25,t-1]}\right) \tag{11}$$

The choice of the *look-back window* has been empirically provided. Indeed, the best lag parameter, which turned to be equal to 25 days, corresponds to such days providing the lowest MSE value among different tested configurations.

We tried different lag parameter values, namely 25, 50 and 100 days, obtaining rather similar results in terms of errors, hence proving the stability of the proposed algorithm w.r.t the look-back window. Indeed, the predictability of a historical series depends largely on its most

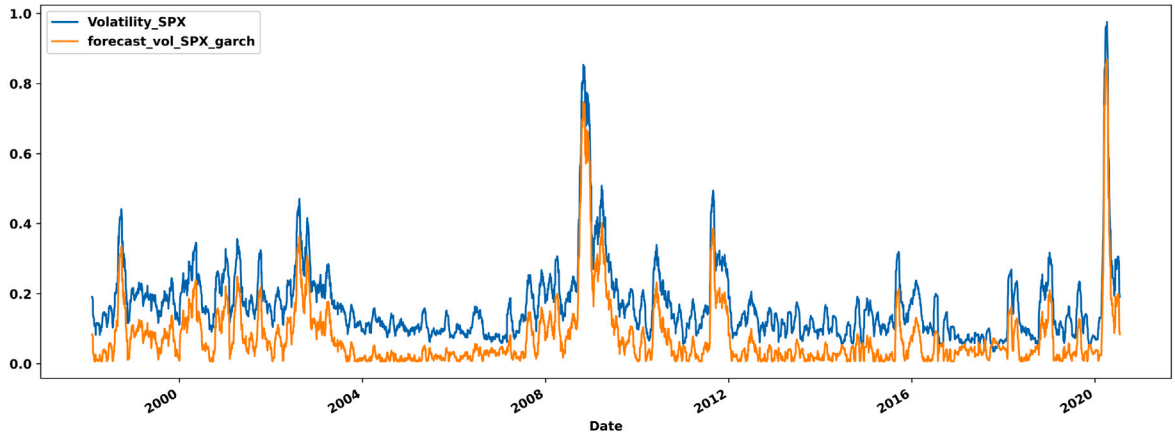


Fig. 1. GARCH(1, 1) volatility output w.r.t. S&P500 data from February 1999 to June 2020.

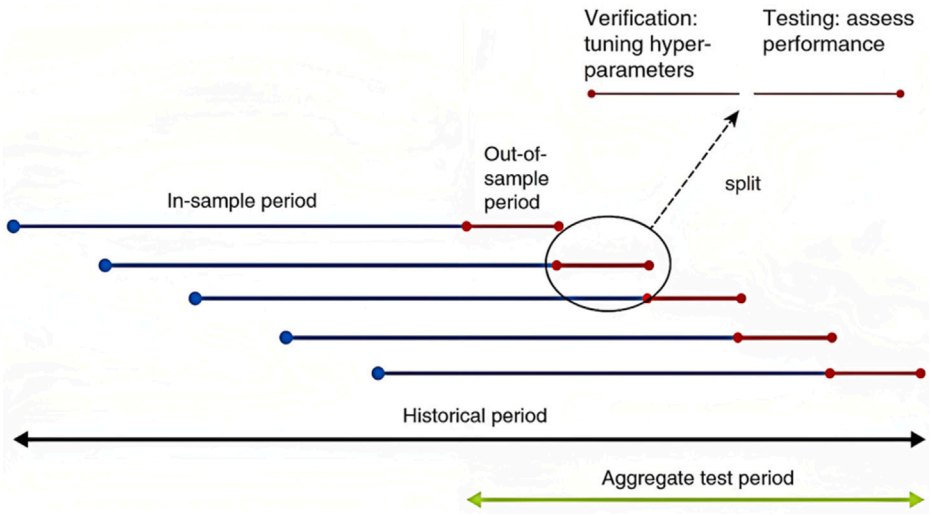


Fig. 2. Rolling Mechanism with the in-sample period equals to 100 days and the out-of-sample period equals to 5 days.

recent values, which are considered to be the most important for the network during the training phase. This fact is supported by empirical evidence and is widely recognized in the field.

We report in Fig. 2 the rolling mechanism used to apply the Supervised Learning procedure for time series.

In the input **X**, then subdivided into a training and a testing setì, we consider supplementary features, hence goign beyond the time series of historical volatility. Specifically, the final dataset consists of the following time series:

- historical volatility σ computed by Eq. (2);
- the volatility forecast $\hat{\sigma}$ computed via the GARCH(1,1) model. Eq. (8) with $p = q = 1$ reads

$$\hat{\sigma}_t^2 = \alpha\sigma_{t-1}^2 + \beta\delta_{t-1}^2 + \omega \tag{12}$$

  that is adapted since it depends on volatility values at time $(t-1)$;
- weekly averaged volatility, i.e. the mean of the volatility of last 5 business days;
- monthly averaged volatility, i.e. the mean of the volatility of last 20 business days;

In Fig. 3, we provide a graphical representation for a subset of the training set for the considered horizon (January 2000–December 2020). Then, we evaluate the model over a test set corresponding from October 2020 to December 2022.

2.4. RNNs

Neural network models (NNs) are essentially inspired by the functioning of the human brain, in terms of mimicking the connections and transmissions of signals among neurons. Indeed, a hidden unit inside a neural network essentially mimics the output transmission of real neurons to other neurons.

In terms of their applications within the machine learning scenario, the main advantage of neural networks lies in their ability to reconstruct non-linear functions while recognizing internal patterns within data.

Most used NNs-based forecasting solutions when dealing with time-series data are those belonging to the large family of the so-called Recurrent Neural Network (RNN) approaches, which are widely applied for stock market projections as well as for sales forecasting, see, e.g., Dunke and Nickel (2020), Papadopoulos et al. (2021), Sharma and Chopra (2013), Wong and Selvi (1998). In this work we focus on two specific RNN variants, i.e. LSTM and GRU.

Our work has been focused on exploiting both the GRU and the LSTM approach, since they have been shown to be very effective when dealing with time series. For the sake of completeness, ANNs-based solutions have been also tried, the discarded because of poor obtained results. We have planned to implement Temporal Convolutional NNs (TCNN), see, e.g., Chen et al. (2020), Wan et al. (2019), type solutions in a future research, particularly trying to capture possible volatility clusters and huge variations well before they really realized within real markets.

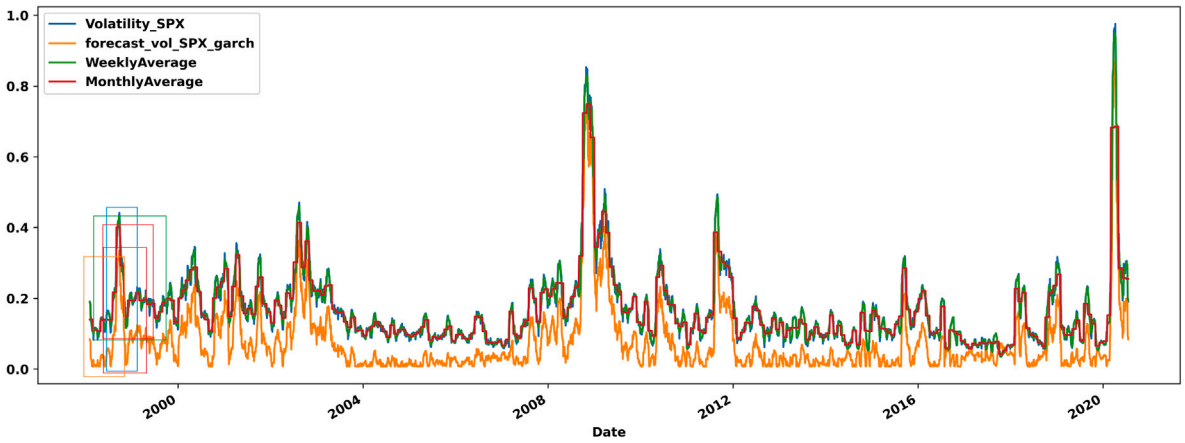


Fig. 3. An extract from the Training Set for S& P 500.

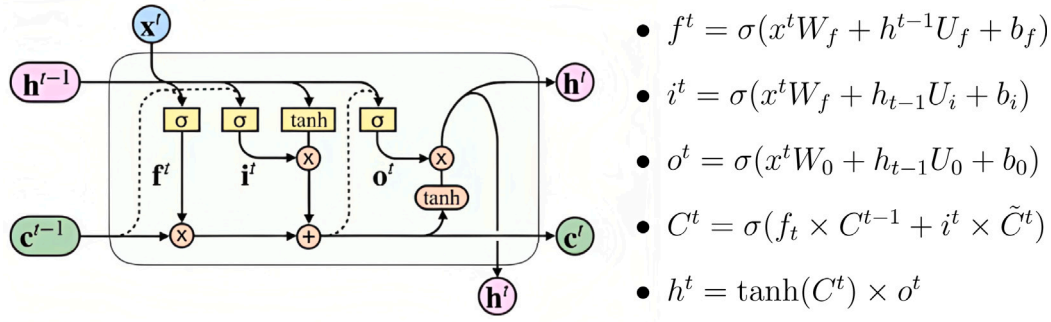


Fig. 4. Structure of a LSTM cell.

2.4.1. LSTM hidden units

The Long Short-Term Memory (LSTM) units were designed in 1997 by Hochreiter and Schmidhuber (1997) to address long-term dependence issues. The main idea behind the LSTM architecture is to replace the recurrent neural network's (RNN) hidden layer neurons with a single set of memory cells, called the "elaboration unit". Additionally, the LSTM model uses a gate structure to filter inputs and keep memory cells up to date. The gate structure includes input, forget, and output gates. Each memory cell is characterized by three sigmoid layers and a tanh layer, as shown in Fig. 4.

In the context of a Long Short-Term Memory (LSTM) network, the notation is typically defined as follows: x^t represents the input vector at time t, f_t is the activation vector of the forget gate at time t, i^t is the activation vector of the input gate at time t, and o^t is the activation vector of the output gate at time t. The output vector of the LSTM unit at time t is denoted by h^t. Additionally, C^t represents the cell state vector at time t. The weight matrices and bias vector parameters of the network are denoted by W, U, and b, respectively.

The number for parameters to be learned during the training process by a hidden layer with n LSTM units is given by:

$$\#_{parameters,LSTM} = 4 * ((n+m) * +n) * n \tag{13}$$

where we are considering 4 functions, namely: 3 sigmoids (f^t i^t, o^t) and 1 hyperbolic tangent (h^t), whit 1 bias parameter, while n, resp. m, represents the size of the input vector, resp. the dimension of the output vector, for a given hidden layer, see, e.g., Hopp (2021), Mateus et al. (2021), for further details.

2.4.2. GRU hidden units

As a modification of the standard LSTM approach, Gated Recurrent Units (GRUs), as presented in Cho et al. (2011), preserve the LSTM's ability to avoid the problem of vanishing gradients. However, they achieve this with a simpler internal structure, resulting in a more straightforward training phase, as fewer computations are required to update the internal states. The updated port controls how much of the past state information is retained in the current state, while the reset port determines whether the current state should be combined with previous information. Fig. 5 illustrates the structure of the GRU memory cells, which are characterized as follows:

W^z, W^r and W being the weight matrices associated with the input vector U^z, U^r and U being the weight matrices from previous step, and b_r, b_z and b representing the biases. Moreover, in this model the logistic sigmoid function is represented by σ, the reset gate is given by r^t, and the update gate is denoted by z^t.

According to Dey and Salem (2021), the total number of parameters in the GRU-RNN for a given layer equals

$$\#_{parameters,GRU} = 3(n^2 + mn + 2n), \tag{14}$$

where n, resp. m, represents the size of input, resp. of the output, vector.

2.4.3. Neural Networks architectures

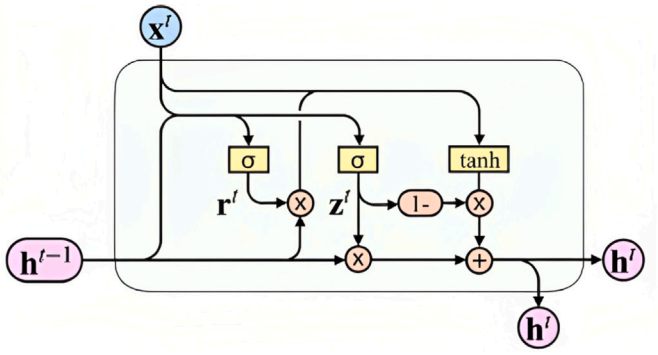
The key difference between GRU and LSTM is that a GRU unit has two gates, namely the reset and the update gate, whereas an LSTM has three gates, namely input, output and the forget gate. Nevertheless, it has been show that the performance of a GRU is comparable to a LSTM if limited to specific applications, like, e.g., classification problems, see Lynn et al. (2019), or prediction tasks, see Mateus et al. (2021).

In this article, we consider the following NN architectures:

1. *LSTM-GARCH*: 2 layers with LSTM units plus a 1 dense layer;
2. *GRU-GARCH*: 2 layers with GRU hidden units plus a 1 dense layer;
3. *LSTM-GRU-GARCH*: a mixed architecture with 3 layers combining different units: a first one with LSTM units followed with a layer with GRU units and a dense layer.

Table 1
Neural Network architecture and parameters.

	LSTM-GARCH	GRU-GARCH	LSTM-GRU-GARCH
First Layer Units	LSTM	GRU	LSTM
First Layer # Neurons	48	48	48
First Layer # Hidden parameters	10176	7776	10176
Second Layer Hidden Units	LSTM	GRU	GRU
Second Layer # Neurons	16	16	16
Second Layer # Hidden parameters	4160	3168	3936
Dense Layer # Hidden parameters	85	85	85
Total # Hidden parameters	14421	11029	13429
Dropout	0.1	0.1	0.1
Learning Rate	0.01	0.01	0.01
Batch Size	32	32	32
Epochs	80	80	80



- $z^t = \sigma(x^t W^{z^t} + h^{t-1} U^z + b_z)$
- $r^t = \sigma(x^t W^{r_t} + h^{t-1} U^r + b_r)$
- $h^t = (1 - z^t) \times \tilde{h}^t + z^t \times h^{t-1}.$

Fig. 5. Structure of a GRU cell.

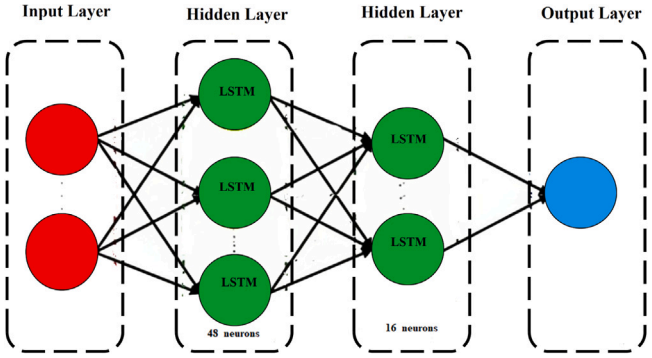


Fig. 6. Graphical representation of the LSTM-GARCH model.

4. An *ENSEMBLE* model aggregating the results from all previous NNs.

We remark that the presence of *GARCH* into names would emphasize that considered model are hybrid, since the presence of the GARCH(1,1) forecast into the NN inputs (see Fig. 6).

We selected the set of parameters in Table 1 through a trial-and-error procedure. Our goal was to optimize the model's predictive accuracy while also ensuring a comparable number of parameters across different models. This allowed us to perform a consistent analysis across all models.

The number of parameters is computed accordingly to Eq. (13), for layers with LSTM units, e.g. $4 * ((48 + 16) * 16 + 16) = 4160$), and via Eq. (14), for layers with GRU units, leading to $3 * (16^2 + 16 * 64 + 2 * 16) = 3936$.

We recall that for Dense layer with n neurons, the number of parameters involved equals

$$\#_{parameters,DENSE} = n * (\#_{previouslayer} + 1)$$

giving $85 = 5 * (16 + 1)$.

The number of parameters is selected empirically using a trial-and-error approach to optimize predictions and ensure consistency in the analysis across different models, considering a training dataset size of 22,604 entries. It should be noted that there are alternative methods available for parameter selection. For example, grid search involves testing all possible combinations of predefined parameter values and selecting the best-performing set based on a performance metric. Another alternative is random search, which involves randomly selecting parameter values from predefined ranges and evaluating the model for each combination. Bayesian optimization uses probabilistic models to determine the most promising parameter values for improving performance. Gradient-based optimization involves iteratively updating parameter values using gradients of the objective function, while evolutionary algorithms generate a population of parameter sets and iteratively mutate the best-performing sets to create a new population. The choice of method depends on the specific problem and available resources. All the aforementioned approaches require proper procedures to be implemented, and we plan to use a selection of these methods in future research. The results for the present case are sufficient for our purposes, even though they required a time-consuming activity due to the specific exploratory nature of the research.

2.4.4. Measuring predictions errors

To evaluate the model prediction performances of the test error, we use the following error metrics

- Mean Square Error (MSE)

$$MSE = \frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2, \tag{15}$$

- Mean Absolute Error (MAE)

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|, \tag{16}$$

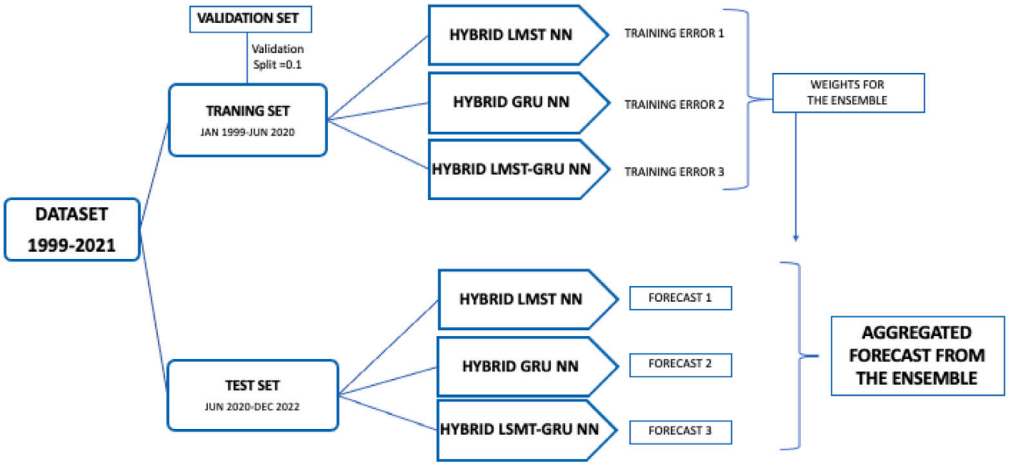


Fig. 7. Scheme of the aggregated Ensemble forecast.

- Mean Absolute Percentage Error (MAPE)

$$MAPE = \frac{1}{n}\sum_{j=1}^{n}\frac{|\mathcal{Y}_j - \hat{\mathcal{Y}}_j|}{|\mathcal{Y}_j|}, \tag{17}$$

where n is the number of samples used in the test set, $\mathcal{Y}_j$ represents the real data value and $\hat{\mathcal{Y}}_j$ shows the forecast, i.e. the NN output from Eq. (11).

We considered two different loss functions during the training process, namely Mean Squared Error (MSE) and Mean Absolute Error (MAE). It is worth noting that using MSE implies giving more weight to larger errors than smaller ones, due to the squaring operation. This can lead to a skewed error estimate, where the contribution of outliers is overvalued. In contrast, MAE considers the absolute value of errors, so all errors are weighted on the same linear scale. This provides a more generic measure of error without overvaluing outliers. However, we observed that using MAE as the loss function generally results in higher test errors, since it does not penalize large discrepancies caused by outliers.

2.5. Ensemble of Neural Networks

An ensemble method relies on a learning paradigm where many NNs-based architectures are jointly used to solve a problem as, e.g., shown by Fig. 7, where we illustrate our (ensemble) forecasting scheme, see, e.g., Borovkova and Tsiamas (2019), Ribeiro et al. (2022), Stefenon et al. (2022) for further details.

Extension of the Bates & Granger Technique. Based on the portfolio diversification theory, the Bates & Granger Technique ignores correlation between forecasting models. The idea is to use the estimated mean squared forecast errors to obtain the combining weights, see, Cheng et al. (2019), Hsiao and Wan (2014) for further details. Following latter approach, we consider a convex combination of forecasts obtained by different models, associated coefficients being weighted according to training loss, with higher coefficient assigned to the network with lower test error. We use a power of grade 5, hence penalizing more terms with higher errors, by taking

$$\mathrm{W_{LG}} = \frac{(1/MSE_{LG})^2}{\text{Total Error}}, \quad \mathrm{W_{GG}} = \frac{(1/\mathrm{MSE_{GG}})^2}{\text{Total Error}}, \quad \mathrm{W_{GLG}} = \frac{(1/\mathrm{MSE_{GLG}})^2}{\text{Total Error}} \tag{18}$$

where LG stands for LSTM-GARCH, GG for GRU- GARCH, and GLG for GRU-LSTM-GARCH model, respectively, and

$$Total\ Error = \left(\frac{1}{MSE_{LG}}\right)^5 + \left(\frac{1}{MSE_{GLG}}\right)^5 + \left(\frac{1}{MSE_{GG}}\right)^5. \tag{19}$$

3. Data analysis and results

In this section we present the numerical results. We calibrate the model with real market data to according with daily closing prices of 4 equities: Standard and Poor's 500, STOXX Europe 600, Hang Seng Index and Nikkei 225.

During a time interval of 20 years, more precisely:

- *Training Set*: January 2000–June 2020 with validation split set at 0.1. In Fig. 8 we can see the training performance for the NN model with GARCH preprocessing and GRU hidden units;
- *Test Set*: October 2020–December 2022, corresponding also to the period of the historical simulation for portfolio ruled by the Risk Parity technique.

3.1. General information about the dataset

It is essential to emphasize that a Risk Parity strategy usually deals with portfolios made by equities, bonds and commodities because of the different intrinsic risk of these instruments. In our setting we focused on a 4-equities, representing relevant both USA and international markets proxies,risk parity portfolio consisting of:

1. S&P500 index or Standard and Poor's 500 is the market capitalization-weighted index of 500 large companies listed on stock exchanges in the United States. It is widely recognized as one of the most accurate indicators of the performance of major American stocks, and by extension, the stock market as a whole.
2. STOXX Europe 600, or Stoxx600, is the European stock index comprising a fixed number of 600 components representing major, mid, and small-capitalization companies from 17 European nations, also with exposure to Great Britain, Switzerland and the Scandinavian countries. Given the wide market exposure, the STOXX Europe 600 index is often considered the European equivalent of the S&P 500 index;
3. Hang Seng Index as proxy for Asia ex Japan (HSI) is a free-float-adjusted market capitalization-weighted index. It is the primary gauge of Hong Kong's stock market performance since it records and monitors the daily changes in the stock prices of the major businesses listed on the city's stock exchange;
4. Nikkei is an abbreviation for Japan's Nikkei 225 Stock Average, being a price-weighted index comprised of the Tokyo Stock Exchange's top 225 blue-chip companies, therefore the Nikkei is similar to the USA centered index called Dow Jones Industrial Average (DJIA).

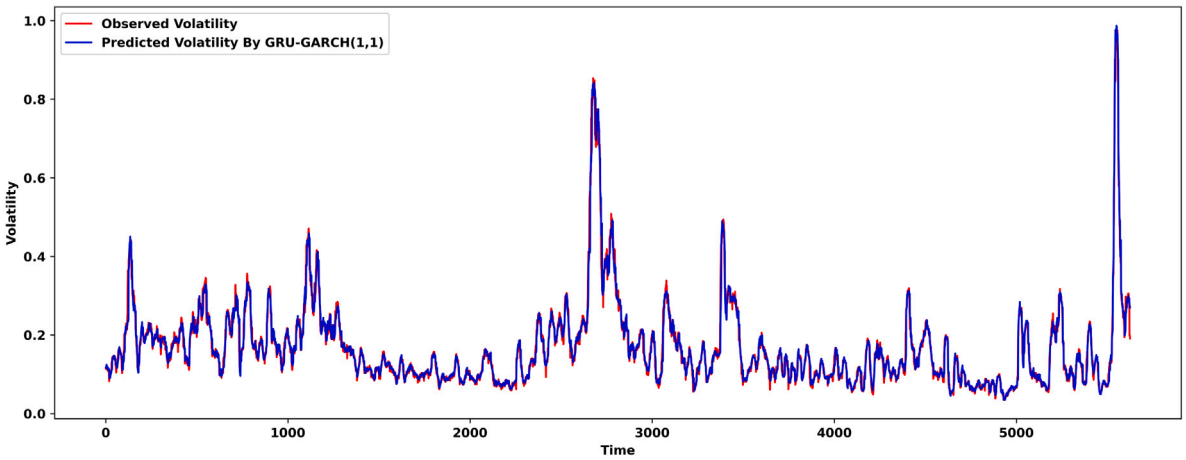


Fig. 8. Calibration from the training set with GRU-GARCH model for the S&P 500 index.

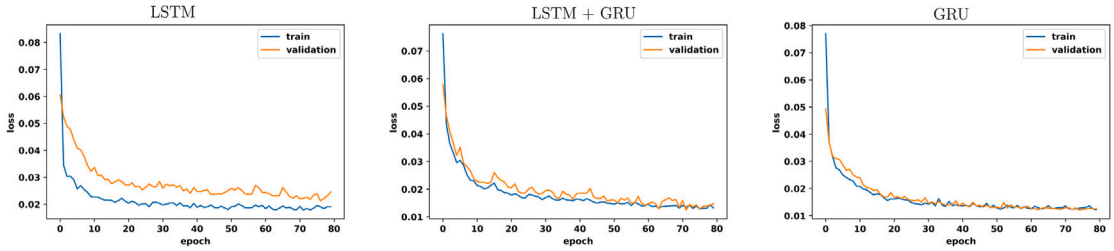


Fig. 9. S& P500, Validation and Train errors for all models with 80 epochs.

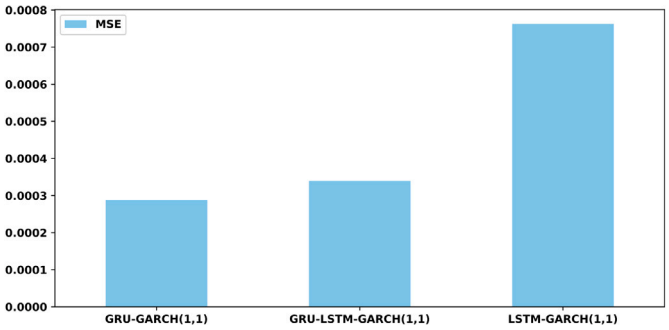


Fig. 10. Ranked training MSE for training set of the S&P index.

Table 2
Training Errors for 2000–2020 simulation, for S&P500.

Models	MSE	RMSE	MAE	MAPE [%]
GARCH	0.00908571	0.0953190	0.09150850	345.56987
LSTM-GARCH	0.00076309	0.0953190	0.01614862	10.246528
GRU-GARCH	**0.0002882**	**0.095319**	**0.0915085**	**6.329703**
LSTM-GRU-GARCH	0.00033954	0.0953190	0.01105917	6.7951889

Table 3
Errors on daily forecast, for S&P500.

Model	MSE	RMSE	MAE	MAPE
LSTM-GARCH	0.023628595	0.153715	0.0177804	12.043454
GRU-GARCH	0.013872029	0.120729	**0.0013675**	**5.7662236**
LSTM-GRU-GARCH	0.014445018	0.120187	0.0059371	6.6963885
ENSEMBLE	**0.01384343**	**0.11765**	0.009488	5.922075

3.2. Calibration results from the training set

For the sake of simplicity, both for the training results and for the predictions ones in Section 3.3, we presents plots and results related to S&P 500 index, ignoring other indexes since they follow a similar pattern over years.

We can see form the second and third picture from Fig. 9, which correspond to LSTM-GRU-GARCH and GRU-GARCH, that we obtain comparable training and validation errors that ensure a good behavior for the model. Conversely, we obtain for LSTM-GARCH a training loss higher than the validation one as it appears from the first loss function in Fig. 9. This behavior can be due to over-fitting or it can also be related to the fact that validation set does not consider dropout.

As we highlight within previous table, on the training set the highest MSE error has been obtained by the GARCH model. The 345% MAPE in Table 2 is due to the bad-performance related to the GARCH forecast as clearly visible in the plot of Fig. 10. The predictive performance is really improved by adding NN with the best model being the GRU-GARCH model.

3.3. Forecast results of the hybrid algorithms for the S&P500 volatility

In this section, we present daily, weekly and monthly frequency forecast for historical volatility for 2021–July 2022.

3.3.1. Daily forecast

As highlighted in Table 3, on the test set for daily forecast, the highest MSE error has been obtained by the LSTM-GARCH model, and modifying this model by NN models significantly reduces the MSE error. Best performances have been obtained by the GRU-GARCH solution, the latter showing the lowest MSE error (see Fig. 11).

3.3.2. Weekly forecast

As highlighted in the Table 4, the highest MSE error has been obtained by the LSTM-GARCH model. Modifying the latter by NN models reduces the MSE error. In particular, best performances have been those obtained by the GRU-GARCH solution showing the lowest MSE error (see Figs. 13 and 14).

Fig. 11. Daily 21 days RV vs predicted volatility for Ensemble model.

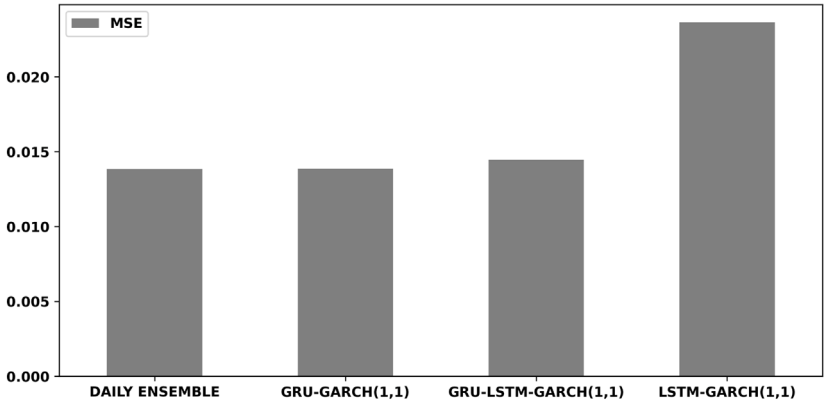


Fig. 12. Ranked MSE errors on daily forecast for S&P500.

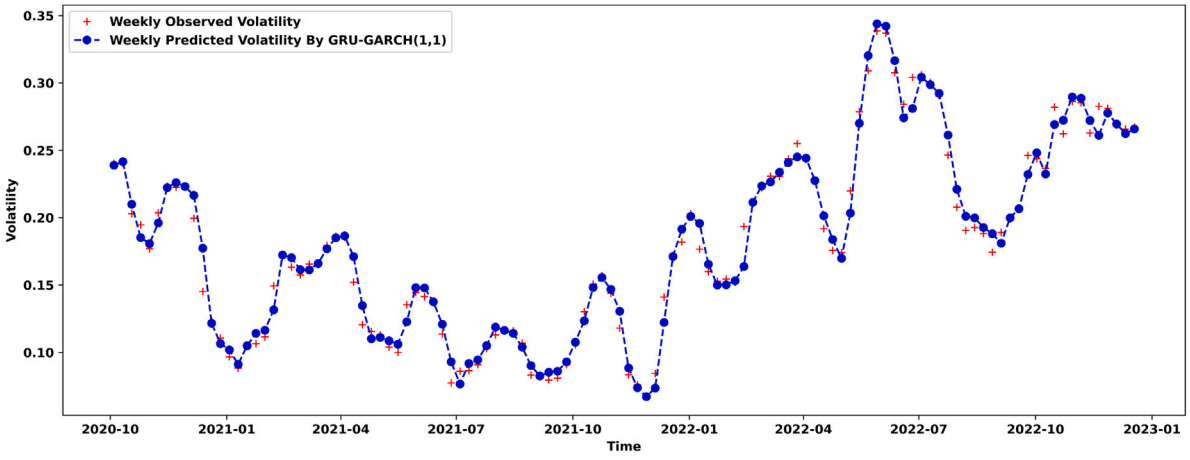


Fig. 13. Weekly predictions for the GRU-GARCH(1,1) model.

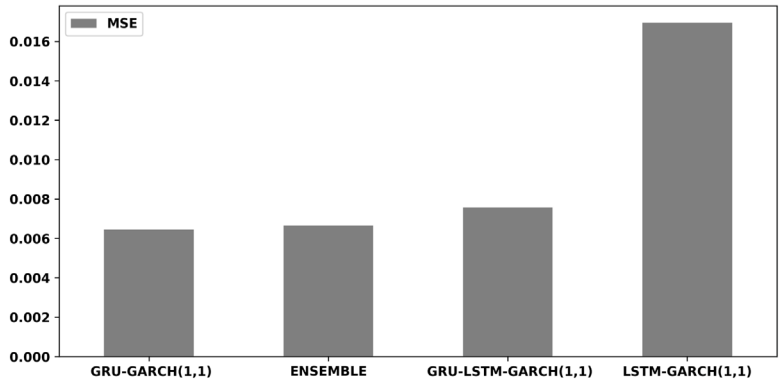


Fig. 14. MSE errors on weekly forecast for S&P 500 index.

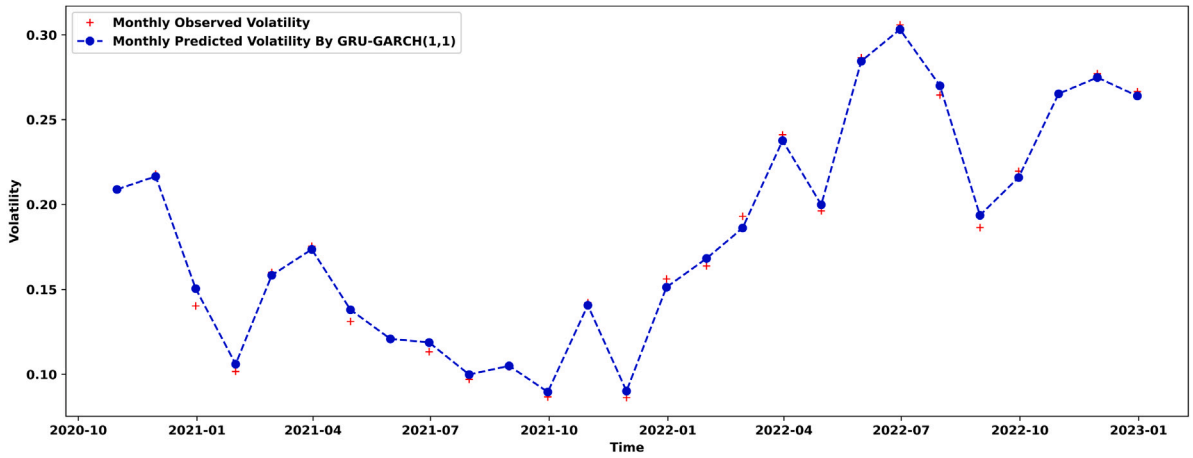


Fig. 15. Monthly predictions for GARCH-GRU model.

Table 4
Errors on weekly forecast, for S&P500.

Models	MSE	RMSE	MAE	MAPE
LSTM-GARCH	0.021117	0.145316	0.016951	10.92226
GRU-GARCH	**0.00890**	**0.09435**	**0.00644**	**4.03471**
LSTM-GRU-GARCH	0.009724	0.098611	0.007565	5.036304
ENSEMBLE	0.009004	0.094884	0.006649	4.258328

Table 5
Errors on monthly forecast, for S&P 500.

Models	MSE	RMSE	MAE	MAPE
LSTM-GARCH	0.0177986	0.133411	0.01510	9.380881
GRU-GARCH	**0.004206**	**0.06845**	**0.00346**	**2.25129**
LSTM-GRU-GARCH	0.0054082	0.073540	0.004821	3.379187
ENSEMBLE	0.0044469	0.066685	0.006649	2.600597

3.3.3. Monthly forecast

As highlighted in Table 5, the highest MSE on the test set for monthly forecast has been obtained by the LSTM-GARCH model, while the best results are related to the GRU-GARCH model, showing the lowest MSE error (see Figs. 15 and 16).

3.4. Discussion of the results

The main improvements in terms of MSE-decreasing have been obtained by implementing the following:

- enlarging the space of features: the initial model only uses the daily volatility as input, while by including the weekly and monthly means as features, the model predictions ended up being smoother.
- Optimizing the *Dropout* hyperparameter: this regularization technique improved significantly the model performance on test data. It was noted that without it, the model performed much better on trained data, but worse in test ones.
- Adding a *Dense layer*: we note that adding as final layer a *dense layer*, whose neurons receive input from all neurons in the previous one, considerably improves the overall accuracy.
- Comparing LSTM and GRU performances: LSTMs remember longer sequences than GRUs and should outperform GRUs in tasks requiring modeling long-distance relations. On the other hand, GRUs train faster and perform better than LSTMs on less training data. Thus, it is crucial to select the opportune fraction of the dataset to have a proper *train-validation split*. The choice of this parameter (we have considered values from 0.08 to 0.2) directly affects the specific models giving the best predictions.

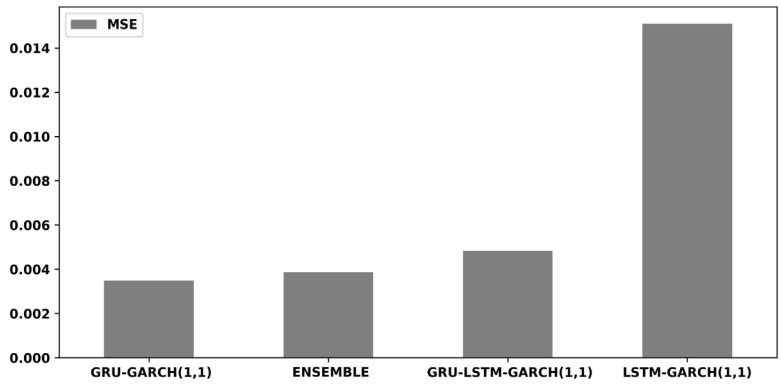


Fig. 16. MSE errors on monthly forecast.

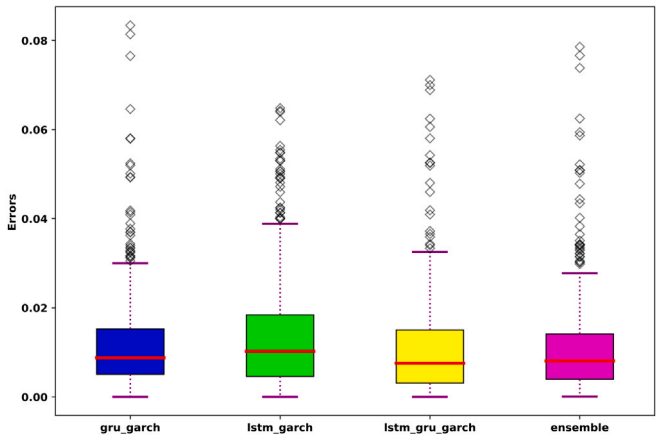


Fig. 17. Box plot of the comparison for the MSE on daily forecast for different NNs architecture with S&P500 data.

- Considering an ensemble algorithm: there are two main reasons to use an ensemble over a single model. The first one is related to performance: the ensemble has good predictions behind only the GARCH-GRU representing our benchmark,as shown in Figs. 12 and 17. The second advantage deals with robustness since an ensemble reduces the spread or dispersion of the predictions and model performance, limiting the bias connected to a single model.

Moreover, comparing Tables 3 and 4, we can see how monthly predictions portfolio are better than the weekly ones in terms of overall of percentage error, with a MAPE of $2,25\%$ and $4,03\%$ respectively. This behavior may be explained by the fact that averaging over a monthly basis provides a smoothing effect over the noise that directly affects the volatility prediction (see Fig. 15).

To summarize, as we can clearly see from Tables 2, 3 and 5, reporting the forecast errors, and from the box plot in Fig. 17, the GRU-GARCH solution outperforms other hybrid models in terms of overall MSE. We also obtain a good performance from the ensemble model, dealing with the aggregated forecast.

We report the errors for different simulations in the aggregated results presented in Appendix.

4. Risk Parity portfolio

In this section we compare an equally-weighted equities (S& 500, STOXX Europe 600, Hang Seng, Nikkei) basket with a risk-parity weighted portfolio of the considered equities over the same reference period October 2020–December 2022.

We use a bottom-up data-driven approach to determine the weights for the portfolio assets. We refer to Section 2.1 for a detailed description of the re-balancing rules.

We begin by calculating the long-term historical volatility for each index, as discussed in Section 2.3. Volatility time series of Fig. 18 are then used as input to update the weights according to Eq. (5).

For the sake of simplicity, we do not consider transactions costs.

The contract position weight within each asset class is proportional to the inverse of its standard deviation, see Eq. (5).

According to weights of Fig. 19, we compute the return in Fig. 20 of an investment over the considered horizon of 112 weeks (October 2020–December 2022).

In Fig. 21, we report the monthly volatility used to compute the corresponding weights reported in Fig. 22.

Fig. 23 reports plot of returns for a monthly re-balanced portfolio over the considered 27 months. We also compare its performances to an equally balanced one.

4.1. Performance analysis

We report some financial considerations from the analysis of the portfolio performance:

- According to Figs. 20 and 23 where we consider the evolution of different portfolios, we can see how Risk Parity techniques outperformed equally balanced portfolios over the considered period October 2020–December 2022. The latter suggests that a diversification based on equalizing risk is able to offer a good protection for investments in periods of market turbulence.
- Figs. 19 and 22 suggest a possible explanation of how the discrete allocation of the Risk Parity mechanism can benefit overall returns. The real advantages from risk parity appear only at the end of the first year. For example, we can see that during 2021 positions on the American and European indices are strengthened up to more than 70% of the allocation in December 2021. While for the previous period with the redistribution mechanism not rewarding any particular stocks, the difference in returns between the two portfolio does not grow considerably as it happens during the peaks of volatility within all the 2022 year.

5. Conclusions

We presented a hybrid method to compute volatility forecast to perform a risk-controlled strategy over a portfolio of 4 indexes, namely S&P 500, STOXX Europe 600, Hang Seng, Nikkei, as proxy of different international markets.

We prove how the version of Risk Parity proposed by Berlinda et al. (2001), moving from Eq. (5), effectively produces encouraging results beyond its simplicity, also suggesting to develop more advanced

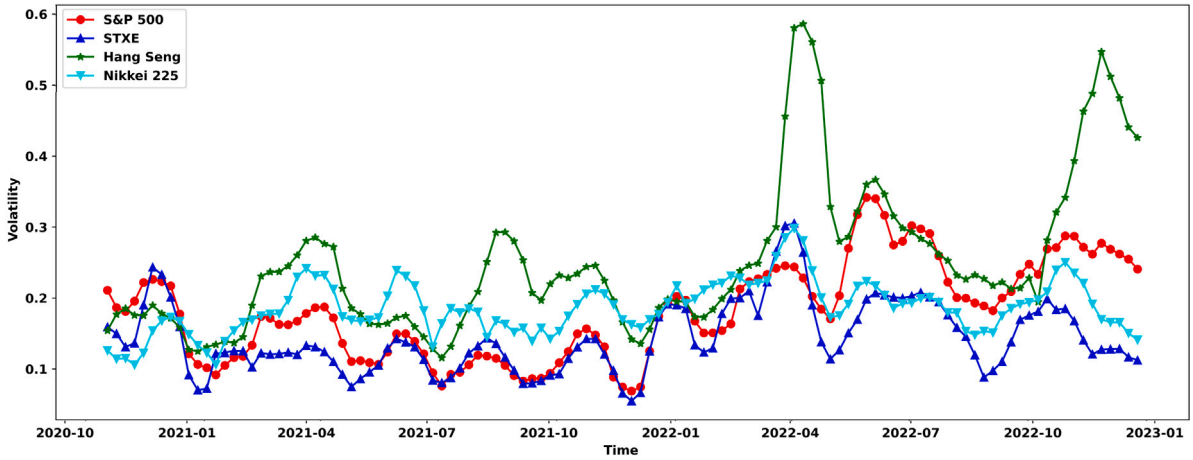


Fig. 18. Weekly Forecast for the 4 considered equities.

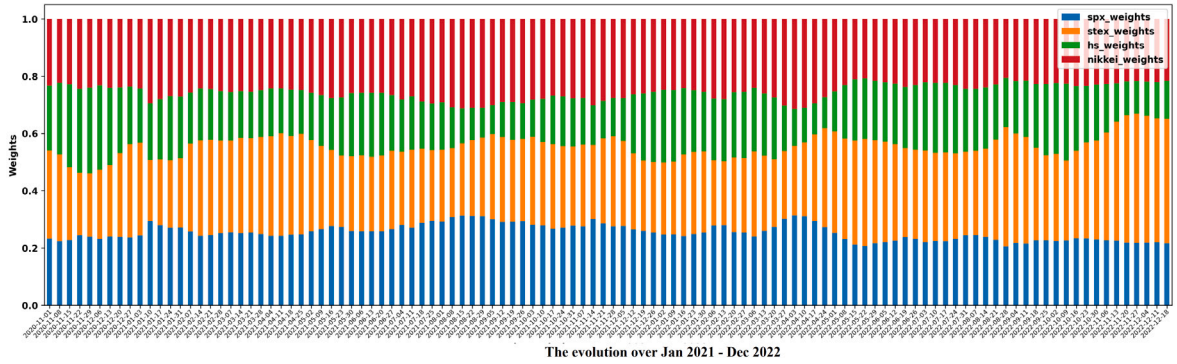


Fig. 19. Weekly Weights for the 4 considered equities.

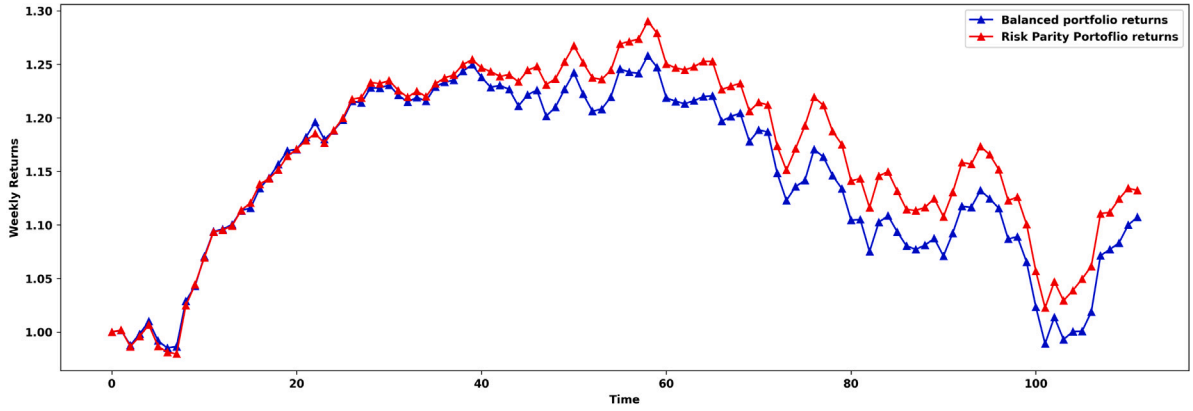


Fig. 20. Weekly rebalanced portfolio.

techniques to include cross-correlation of assets in addition to individual volatility. Considering an ensemble method helps improving the accuracy of predictions.

We emphasize that our findings remain robust when dealing with different indexes, providing an empirical evidence for a universal volatility mechanism among indexed.

Eventually, we present some future research directions:

- When dealing with real applications, standard deviation is a dangerously limited estimate of the true risk of an asset class. We believe the model could benefit from using other risk measures (such as sampled entropy or cross-entropy) as well as by improving the risk analysis;
- We would like to analyze other NN algorithms such as Temporal CNNs or Regression Trees to analyze the problem even if the considered models already give acceptable results.
- We assume a portfolio composed only of equities. It is worth to mention that dealing with different assets, e.g., bonds, futures, options, may increase the effectiveness of the proposed methodology. It would also raise the potential diversification benefits of a Risk Parity strategy in terms of risk efficiency, namely the risk-adjusted returns.
- Finally, in Di Persio et al. (2021) the authors consider a portfolio analysis focused on a VolTarget strategy that, moving from volatility predictions, iteratively balances a portfolio between a risky investment and a risk-less one. It might be worthy of further

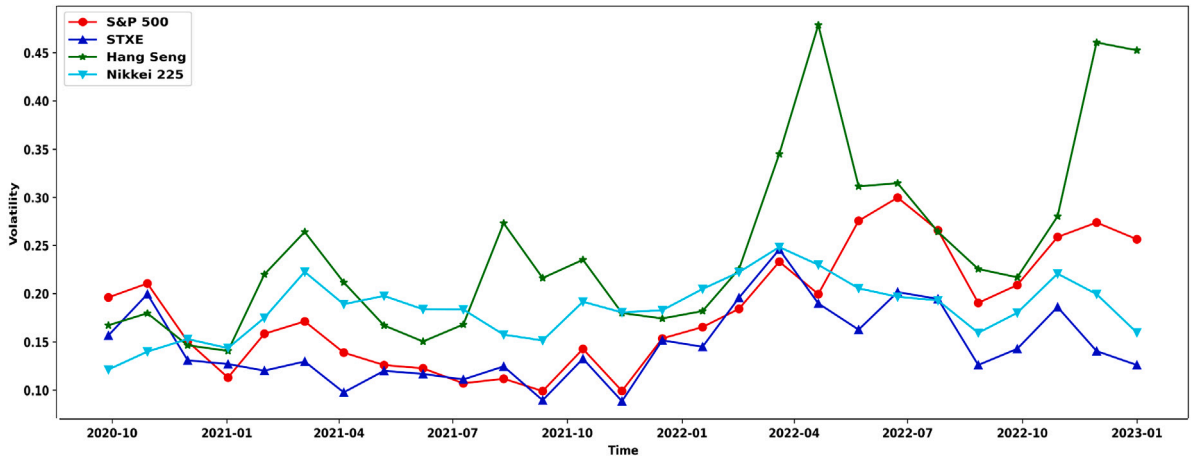


Fig. 21. Monthly Forecast for the 4 considered equities.

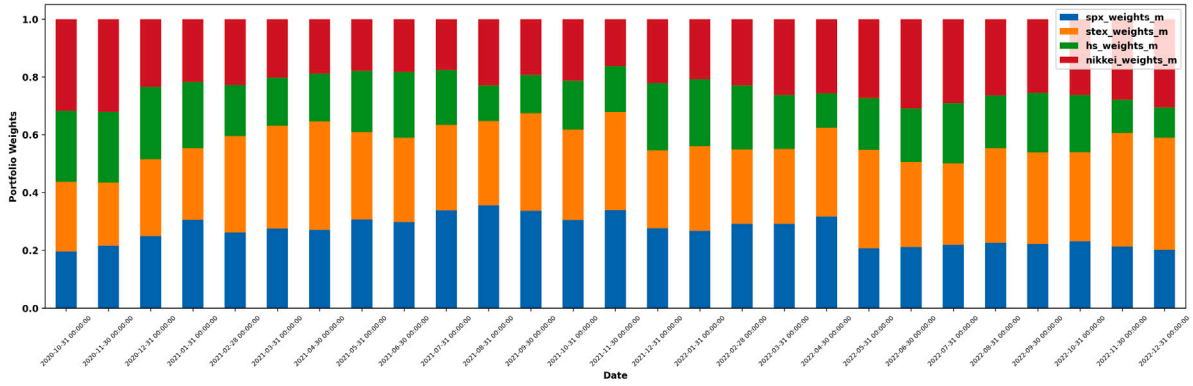


Fig. 22. Monthly Weights for the 4 considered equities.

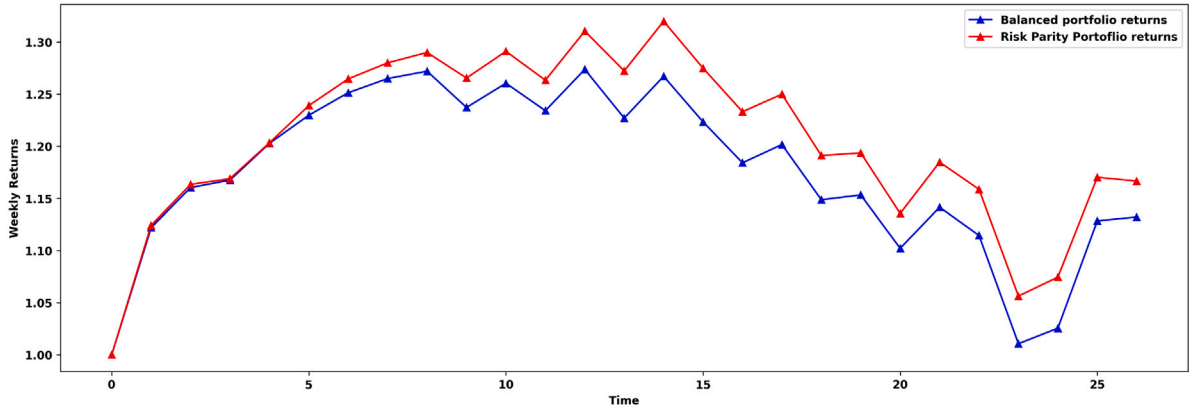


Fig. 23. Monthly re-balanced portfolio.

investigation running a comparison analysis between VolTarget and Risk Parity to understand analogies over long-term horizons and, moreover, to consider a mixed algorithm that includes both the 2 risk-controlled strategies.

CRediT authorship contribution statement

Luca Di Persio: Study conception and design. **Matteo Garbelli:** Study conception and design. **Fatemeh Mottaghi:** Study conception and design. **Kai Wallbaum:** Study conception and design.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

We use data from Yahoo Finance.

Acknowledgments

The authors would like to thank the great and stimulating work done by the referees whose advice has definitely helped us to improve our contribution. All authors approved the version of the manuscript to be published.

Appendix. Aggregate tables

Errors on daily forecast				
Indexes	Models	MSE	MAE	MAPE
S&P500	LSTM-GARCH	0.0236280	0.0177804	12.04345
	GRU-GARCH	0.0138720	**0.0013675**	**5.766220**
	LSTM-GRU-GARCH	0.0144450	0.0059371	6.696388
	ENSEMBLE	**0.0138434**	0.0094885	5.922075
STXE 600	LSTM-GARCH	0.0248722	0.0380490	0.025007
	GRU-GARCH	**0.0153641**	0.0053930	**7.529229**
	LSTM-GRU-GARCH	0.0162282	**0.0042649**	7.796152
	ENSEMBLE	0.0156058	0.0105275	7.496465
HANG SENG	LSTM-GARCH	0.0336795	0.0967208	9.459470
	GRU-GARCH	**0.0236604**	**0.0018508**	**6.612040**
	LSTM-GRU-GARCH	0.0250805	0.0118126	7.155621
	ENSEMBLE	0.0239497	0.0155571	6.698893
NIKKEI 225	LSTM-GARCH	0.0185359	0.1957084	7.503521
	GRU-GARCH	0.0162912	0.0126196	0.012076
	LSTM-GRU-GARCH	**0.0152084**	0.0196809	**0.010959**
	ENSEMBLE	0.0155269	**0.0110254**	5.960312

Errors on weekly forecast				
Indexes	Models	MSE	MAE	MAPE
S&P500	LSTM-GARCH	0.021117	0.016951	10.92226
	GRU-GARCH	**0.008903**	**0.006445**	**4.034713**
	LSTM-GRU-GARCH	0.009724	0.007565	5.036304
	ENSEMBLE	0.009004	0.006649	4.258328
STXE 600	LSTM-GARCH	0.022576	0.018520	13.11474
	GRU-GARCH	**0.010610**	**0.007814**	**5.827596**
	LSTM-GRU-GARCH	0.0116497	0.008487	6.179462
	ENSEMBLE	0.0109887	0.008023	5.888007
HANG SENG	LSTM-GARCH	0.0287128	0.018974	8.164507
	GRU-GARCH	**0.0154987**	**0.010919**	**4.730918**
	LSTM-GRU-GARCH	0.0179641	0.012909	5.534155
	ENSEMBLE	0.0164694	0.011768	5.054182
NIKKEI 225	LSTM-GARCH	0.0152089	0.011774	6.542322
	GRU-GARCH	0.0119356	0.009215	0.043689
	LSTM-GRU-GARCH	**0.0106270**	**0.007949**	**0.038849**
	ENSEMBLE	0.0110109	0.008236	4.473878

Errors on monthly forecast				
Indexes	Models	MSE	MAE	MAPE
S&P500	LSTM-GARCH	0.0177986	0.015100	9.380881
	GRU-GARCH	**0.0042063**	**0.003464**	**2.251295**
	LSTM-GRU-GARCH	0.0054082	0.004821	3.379187
	ENSEMBLE	0.0044460	0.006649	2.600597
STXE 600	LSTM-GARCH	0.0193511	0.016567	9.380881
	GRU-GARCH	0.0071429	0.005047	3.442933
	LSTM-GRU-GARCH	**0.0063606**	**0.004917**	**3.413815**
	ENSEMBLE	0.0067136	0.008023	3.431308
HANG SENG	LSTM-GARCH	0.0191462	0.015369	6.413808
	GRU-GARCH	**0.0091628**	**0.006991**	**2.928284**
	LSTM-GRU-GARCH	0.0114515	0.008595	3.618165
	ENSEMBLE	0.0131702	0.011768	4.199611
NIKKEI 225	LSTM-GARCH	0.0094146	0.008254	4.343089
	GRU-GARCH	0.0080447	0.006747	0.030392
	LSTM-GRU-GARCH	**0.0062053**	**0.005135**	**0.023992**
	ENSEMBLE	0.0067475	0.008236	2.780131

References

Albeverio, S., Steblovskaya, V., & Wallbaum, K. (2013). Investment instruments with volatility target mechanism. *Quantitative Finance*, *13*(10), 1519–1528. http://dx.doi.org/10.1080/14697688.2013.804943.

Albeverio, S., Steblovskaya, V., & Wallbaum, K. (2018). The volatility target effect in structured investment products with capital protection. *Review of Derivatives Research*, *21*(2), 201–229. http://dx.doi.org/10.1007/s11147-017-9138-2.

Andersen, T. G., Bollerslev, T., Diebold, F. X., & Ebens, H. (2001). The distribution of realized stock return volatility. *Journal of Financial Economics*, *61*(1), 43–76. http://dx.doi.org/10.1016/S0304-405X(01)00055-1.

Berlinda, L., Phillip, B., & Tianyin, C. (2001). Indexing risk parity strategies, research division of *S&P* global. *Journal of Financial Economics*, *61*(1), 43–76. http://dx.doi.org/10.1016/S0304-405X(01)00055-1.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, *31*(3), 307–327. http://dx.doi.org/10.1016/0304-4076(86)90063-1.

Borovkova, S., & Tsiamas, I. (2019). An ensemble of LSTM neural networks for high-frequency stock market classification. *Journal of Forecasting*, *38*(6), 600–619. http://dx.doi.org/10.1002/for.2585.

Cao, H., Badescu, A., Cui, Z., & Jayaraman, S. K. (2020). Valuation of VIX and target volatility options with affine GARCH models. *Journal of Futures Markets*, *40*(12), 1880–1917. http://dx.doi.org/10.2139/ssrn.3650690.

Chaves, D., Hsu, J., Li, F., & Shakernia, O. (2011). Risk parity portfolio vs. other asset allocation heuristic portfolios. *The Journal of Investing*, *20*(1), 108–118. http://dx.doi.org/10.3905/joi.2011.20.1.108.

Chen, Y., Kang, Y., Chen, Y., & Wang, Z. (2020). Probabilistic forecasting with temporal convolutional neural network. *Neurocomputing*, *399*, 491–501. http://dx.doi.org/10.1016/j.neucom.2020.03.011.

Cheng, X., Wu, J., & Xu, J. (2019). Evaluation study of linear combination technique for SVM related time series forecasting. In *Proceedings of the 52nd Hawaii international conference on system sciences*. http://dx.doi.org/10.24251/HICSS.2019.151.

Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2011). On the properties of neural machine translation: Encoder-decoder approaches. http://dx.doi.org/10.3115/v1/W14-4012, arXiv preprint arXiv:1409.1259.

Cont, R. (2007). Volatility clustering in financial markets: empirical facts and agent-based models. In *Long memory in economics* (pp. 289–309). http://dx.doi.org/10.1007/978-3-540-34625-8_10.

Dey, R., & Salem, F. M. (2021). Gate-variants of gated recurrent unit (GRU) neural networks. In *2017 IEEE 60th international Midwest symposium on circuits and systems, vol. 9, no. 2* (pp. 1597–1600). http://dx.doi.org/10.1109/MWSCAS.2017.8053243.

Di Persio, L., Garbelli, M., & Wallbaum, K. (2021). Forward-looking volatility estimation for risk-managed investment strategies during the covid-19 crisis. *Risks*, *9*(2), 33. http://dx.doi.org/10.3390/risks9020033.

Dunke, F., & Nickel, S. (2020). Neural networks for the metamodeling of simulation models with online decision making. *Simulation Modelling Practice and Theory*, *99*, http://dx.doi.org/10.1016/j.simpat.2019.102016.

Elsworth, S., & Güttel, S. (2020). Time series forecasting using LSTM networks: A symbolic approach. http://dx.doi.org/10.48550/arXiv.2003.05672, arXiv preprint arXiv:2003.05672.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica*, 987–1007. http://dx.doi.org/10.2307/1912773.

Fang, Z., Ma, X., Pan, X., Yang, G., & Arce, G. R. (2023). Movement forecasting of financial time series based on adaptive LSTM-BN network. *Expert Systems with Applications*, *213*, 119–207. http://dx.doi.org/10.1016/j.eswa.2022.119207.

Hanck, C., Arnold, M., Gerber, A., & Schmelzer, M. (2019). *Introduction to econometrics with R* (pp. 1–9). University of Duisburg-Essen, https://www.econometrics-with-r.org/ITER.pdf.

Hansen, P. R., & Lunde, A. (2005). A forecast comparison of volatility models: does anything beat a GARCH (1, 1)? *Journal of Applied Econometrics*, *20*(7), 873–889. http://dx.doi.org/10.2139/ssrn.264571.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Hopp, D. (2021). Economic nowcasting with long short-term memory artificial neural networks (LSTM). arXiv preprint arXiv:2106.08901, https://arxiv.org/ftp/arxiv/papers/2106/2106.08901.pdf.

Hsiao, C., & Wan, S. K. (2014). Is there an optimal forecast combination? *Journal of Econometrics*, *178*, 294–309. http://dx.doi.org/10.1016/j.jeconom.2013.11.003.

Ju, J., Liu, K., & Liu, F. (2022). Prediction of SO2 concentration based on AR-LSTM neural network. *Neural Processing Letters*, http://dx.doi.org/10.1007/s11063-022-11119-7.

Kat, H. M., & Heynen, R. C. (1994). Volatility prediction: A comparison of the stochastic volatility, GARCH (1, 1) and Egarch (1, 1) models. *Journal of Derivatives*, *2*(2), http://dx.doi.org/10.3905/jod.1994.407912.

Kim, H. Y., & Won, C. H. (2018). Forecasting the volatility of stock price index: A hybrid model integrating LSTM with multiple GARCH-type models. *Expert Systems with Applications*, *103*, 25–37. http://dx.doi.org/10.1016/j.eswa.2018.03.002.

Kristjanpoller, W., & Minutolo, M. C. (2018). A hybrid volatility forecasting framework integrating GARCH, artificial neural network, technical analysis and principal components analysis. *Expert Systems with Applications*, *109*, 1–11. http://dx.doi.org/10.1016/j.eswa.2018.05.011.

Lu, X., Que, D., & Cao, G. (2016). Volatility forecast based on the hybrid artificial neural network and GARCH-type models. *Procedia Computer Science*, *91*, 1044–1049. http://dx.doi.org/10.1016/j.procs.2016.07.145.

Lynn, H. M., Pan, S. B., & Kim, P. (2019). A deep bidirectional GRU network model for biometric electrocardiogram classification based on recurrent neural networks. *IEEE Access*, *7*, http://dx.doi.org/10.1109/ACCESS.2019.2939947.

Mao, W., Zhu, H., Wu, H., Lu, Y., & Wang, H. (2016). Forecasting and trading credit default swap indices using a deep learning model integrating merton and LSTMs. *Expert Systems with Applications*, *213*, http://dx.doi.org/10.1016/j.eswa.2022.119012.

Mateus, B. C., Mendes, M., Farinha, J. T., Assis, R., & Cardoso, A. M. (2021). Comparing LSTM and GRU models to predict the condition of a pulp paper press. *Energies*, *14*(21), http://dx.doi.org/10.3390/en14216958.

Nguyen, N., Tran, M. N., Gunawan, D., & Kohn, R. (2019). A long short-term memory stochastic volatility model. arXiv preprint arXiv:1906.02884, https://arxiv.org/pdf/1906.02884.pdf.

Papadopoulos, T., Kosmas, I. J., & Michalakelis, C. (2021). Artificial neural networks: Basic methods and applications in finance, management and decision making, a roadmap. http://dx.doi.org/10.31031/NRS.2021.06.000648.

Peng, L., Wang, L., Xia, D., & Gao, Q. (2021). Effective energy consumption forecasting using empirical wavelet transform and long short-term memory. *Energy*, *238*, http://dx.doi.org/10.1016/j.energy.2021.121756.

Ribeiro, M. H. D. M., da Silva, R. G., Moreno, S. R., Mariani, V. C., & dos Santos Coelho, L. (2022). Efficient bootstrap stacking ensemble learning model applied to wind power generation forecasting. *International Journal of Electrical Power & Energy Systems*, *136*, http://dx.doi.org/10.1016/j.ijepes.2021.107712.

Roncalli, T. (2013). *Introduction to risk parity and budgeting*. CRC Press, http://dx.doi.org/10.2139/ssrn.2272973.

Sharma, A., & Chopra, A. (2013). Artificial neural networks: Applications in management. *Journal of Business and Management*, *12*(5), 32–40. http://dx.doi.org/10.9790/487X-1253240.

Stefenon, S. F., Ribeiro, M. H. D. M., Nied, A., Yow, K. C., Mariani, V. C., dos Santos Coelho, L., & Seman, L. O. (2022). Time series forecasting using ensemble learning methods for emergency prevention in hydroelectric power plants with dam. *Electric Power Systems Research*, *202*, http://dx.doi.org/10.1016/j.epsr.2021.107584.

Teräsvirta, T. (2009). An introduction to univariate GARCH models. In *Handbook of financial time series* (pp. 17–42). http://dx.doi.org/10.1007/978-3-540-71297-8_1.

Wan, R., Mei, S., Wang, J., Liu, M., & Yang, F. (2019). Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting. *Electronics*, *8*(8), http://dx.doi.org/10.3390/electronics8080876.

Williams, B. (2011). *GARCH (1, 1) models*. Ruprecht-Karls-Universitat Heidelberg, https://math.berkeley.edu/~btw/thesis4.pdf.

Wong, B. K., & Selvi, Y. (1998). Neural network applications in finance: A review and analysis of literature. *Information & Management*, *34*(3), 129–139. http://dx.doi.org/10.1016/S0378-7206(98)00050-0.

Woźniak, M., Siłka, J., Wieczorek, M., & Alrashoud, M. (2021). Recurrent neural network model for IoT and networking malware threat detection. *IEEE Transactions on Industrial Informatics*, *17*(8), 5583–5594. http://dx.doi.org/10.1109/TII.2020.3021689.

Woźniak, M., Wieczorek, M., & Siłka, J. (2023). Bilstm deep neural network model for imbalanced medical data of IoT systems. *Future Generation Computer Systems*, *141*, 489–499. http://dx.doi.org/10.1016/j.future.2022.12.004.

Zhang, C., Zhang, Y., Cucuringu, M., & Qian, Z. (2022). Volatility forecasting with machine learning and intraday commonality. https://ssrn.com/abstract=4022147.