*Journal of Financial Econometrics*, 2020, Vol. 18, No. 3, 502–531
doi: 10.1093/jjfinec/nbaa008
Advance Access Publication Date: 13 June 2020
Article



# Realized Volatility Forecasting with Neural Networks

**Andrea Bucci**[1,2]

[1]Università Politecnica delle Marche and [2]Università degli Studi G. d'Annunzio Chieti e Pescara

Address correspondence to Andrea Bucci, Università degli Studi "G. d'Annunzio" Chieti-Pescara, Viale Pindaro 42, Pescara, Italy, or e-mail: andrea.bucci@unich.it.

Received September 2, 2019; revised September 2, 2019; editorial decision April 14, 2020; accepted April 14, 2020

## Abstract

In the last few decades, a broad strand of literature in finance has implemented artificial neural networks as a forecasting method. The major advantage of this approach is the possibility to approximate any linear and nonlinear behaviors without knowing the structure of the data generating process. This makes it suitable for forecasting time series which exhibit long-memory and nonlinear dependencies, like conditional volatility. In this article, the predictive performance of feed-forward and recurrent neural networks (RNNs) was compared, particularly focusing on the recently developed long short-term memory (LSTM) network and nonlinear autoregressive model process with eXogenous input (NARX) network, with traditional econometric approaches. The results show that RNNs are able to outperform all the traditional econometric methods. Additionally, capturing long-range dependence through LSTM and NARX models seems to improve the forecasting accuracy also in a highly volatile period.

**Key words:** neural network, machine learning, stock market volatility, realized volatility

**JEL classification:** C22, C24, C58, G17

Measuring and predicting stock market volatility has received growing attention from both academics and practitioners over the last years. It is well known that stock return volatility varies over time (Engle, 1982; Bollerslev, 1986) and asymmetrically responds to unexpected news (Black, 1976; Nelson, 1990), which may cause distortions in the estimation of volatility and in the definition of its underlying process. For these reasons, some authors suggested to estimate stock market volatility through a smooth transition or a threshold model (De Pooter, Martens, and Van Dijk, 2008; McAleer and Medeiros, 2008). The nonlinearity makes the estimation of these models difficult, since the sample log-likelihood can exhibit local maxima and may be generally hard to solve with confidence. Furthermore, this class of models is in general greedy in requiring a substantial amount of data to identify the states

and presents poor out-of-sample forecasting performance (Clements and Krolzig, 1998; Pavlidis, Paya, and Peel, 2012).

In this framework, this article aims to capture the nonlinear relationships between aggregate stock market volatility, measured by realized volatility, and a set of financial and macroeconomic variables through artificial neural networks (ANNs). Realized volatility has been shown to be subject to structural breaks and regime-switching, hence the need to use a nonlinear adaptive modeling approach such neural networks. This method allows approximating arbitrarily well a wide class of linear and nonlinear functions without knowing the data generating process. Furthermore, ANNs are found to be particularly useful to forecast volatile financial variables exhibiting nonlinear dependence, such as stock prices, exchange rates, and realized volatility; see Donaldson and Kamstra (1996a,b). Even if neural networks have been around since 1950s, only in the last two decades they have been used in finance, showing that they can outperform linear models in capturing complex relationships in which linear models fail to perform well. In particular, they seem to be suitable for modeling the dynamics of realized volatility in relation to macroeconomic and financial determinants, that may drive the dynamics of realized volatility not linearly.

ANNs have been commonly implemented for predicting stock prices (White, 1988; Kamijo and Tanigawa, 1990; Khan, 2011), while there has been little effort on forecasting volatility through neural networks. Moreover, neural networks have been mostly employed in combination with GARCH models (Hajizadeh et al., 2012; Maciel, Gomide, and Ballini, 2016). For instance, Donaldson and Kamstra (1997) investigated the usefulness of a semi-nonparametric GARCH model to capture nonlinear relationships, proving that the ANN model performs better than all competing models. Hu and Tsoukalas (1999), instead, combined the forecasts from four conditional volatility models within a neural network's architecture, showing that the ANNs predict accurately well the targeted variable during crisis periods. Arnerić, Poklepovic, and Aljinović (2014) based their neural networks on the squared innovations deriving from a GARCH model. They relied on a Jordan neural network (JNN) and showed that an neural networks (NN) model provides superior forecasting accuracy in comparison with other linear and nonlinear models. An early contribution to this literature is Hamid and Iqbal (2004), which compared the forecasting performance of neural networks using implied volatility and realized volatility. In their manuscript, neural networks were able to outperform implied volatility forecasts and were in line with realized volatility. Kristjanpoller, Fadic, and Minutolo (2014) also made use of neural networks to forecast, as in this manuscript, monthly realized volatility and returns of three Latin-American stock market indexes.

Fernandes, Medeiros, and Scharth (2014) extended these studies by specifying a neural-network heterogeneous autoregressive (HAR) with exogenous variables to improve implied volatility forecasts. A recent paper by Vortelinos (2017) implemented a neural network to forecast a nonparametric volatility measure. The author concluded that the persistence in realized volatility is not well approximated by a feed-forward network. More recently, Rosa et al. (2014) and Miura, Pichl, and Kaizoji (2019) relied on the use of neural networks to provide out-of-sample forecasts of realized volatility, both showing that neural networks were able to outperform linear models.

This article contributes to this literature investigating whether a totally nonparametric model is able to outperform econometric methods in forecasting realized volatility. In particular, the analysis performed here compares the forecasting accuracy of time series models with

several neural networks architectures, as the feed-forward neural network (FNN), the Elman neural network (ENN), the JNN, a long short-term memory (LSTM) neural network, and the nonlinear autoregressive model process with eXogenous input (NARX) neural network.

The latent volatility is estimated through the ex-post measurement of volatility based on high-frequency data, namely realized volatility; see Andersen et al. (2001) and Barndorff-Nielsen and Shephard (2002). Since macroeconomic and financial variables, which are sampled at lower frequencies, are included in the model, realized volatility is estimated on a monthly basis from daily squared returns.

The remainder of this article is organized as follows: Section 1 illustrates the data set, the estimation method of the volatility, and the set of macroeconomic and financial predictors. Section 2 introduces the neural network models. The choice of the architecture of the neural networks is presented in Section 3. In Section 4, the performance of the ANNs is assessed in terms of forecasting accuracy, while Section 5 concludes.

## 1 Data and Volatility Measurement

The data set employed in this study comprises monthly observations from February 1950 to December 2017 for a total of 815 observations. The realized variance for month $t$ is computed as the sum of squared daily returns, $\sum_{i=1}^{N_t} r_{i,t}^2$, where $r_{i,t}$ is the $i$th daily continuously compounded return in month $t$ and $N_t$ denotes the number of trading days during month $t$. Given that the natural logarithm of realized volatility is approximately Gaussian (Andersen et al., 2001), the realized volatility is here defined as the log of the square root of the realized variance (RV):

$$RV_t = \ln \sqrt{\sum_{i=1}^{N_t} r_{i,t}^2}, \tag{1}$$

where $r_{i,t}$ is the daily return of the Standard & Poor's (S&P) index. The logarithm of the realized volatility is highly persistent, as indicated by the time series plot in Figure 1 and by the autocorrelation function (ACF) in Figure 2, suggesting that a long-memory detecting model should be implemented (Rossi and Santucci de Magistris, 2014). In order to understand whether a nonlinear model was truly necessary, the nonlinearity tests discussed in Terasvirta (1994) and Keenan (1985) were performed. The tests rejected the null of linearity with a $p$-value lower than 0.001.

Since volatility exhibits a highly variable behaviur, one may also suspect that its dynamics are partly driven by several economic variables. A strand of literature has focused on the identification of economic drivers of volatility. In a seminal work, Schwert (1989) found that volatility behaves in a countercyclical way respect to economic activity. Afterward, both Engle, Ghysels, and Sohn (2009) and Diebold and Yilmaz (2009) showed a strong link between macroeconomic fundamentals and stock return volatility. More recently, Paye (2012) and Christiansen, Schmeling, and Schrimpf (2012) examined the role of a large set of macroeconomic and financial variables on the dynamics of realized volatility. They proved that the presence of exogenous variables helps increasing forecasting accuracy. The same findings are showed by Bucci, Palomba, and Rossi (2019), which analyzed the forecasting accuracy of realized covariance through a Vector Logistic Smooth Transition Autoregressive model.

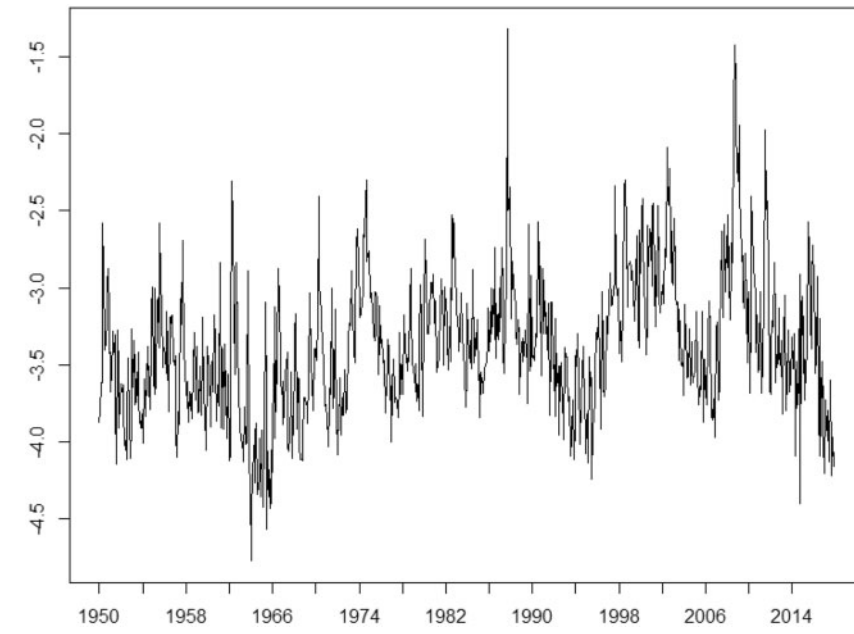



**Figure 1** log RV from February 1950 through December 2017.

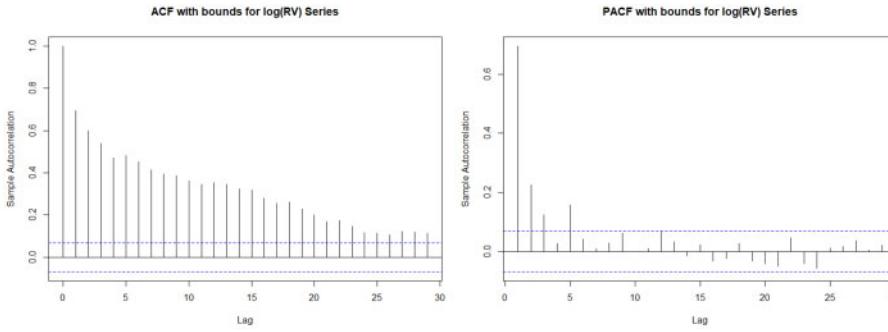



**Figure 2** ACF and partial autocorrelation function of RV.

Understanding which are the volatility predictors can be crucial for investment decisions, and for policy makers and monetary authorities. Thus, this analysis relies on a comprehensive set of macroeconomic and financial variables as volatility predictors.

As in Paye (2012) and Christiansen, Schmeling, and Schrimpf (2012), I include in the analysis many predictive variables from return predictability literature (Mele, 2007, 2008).

First, the set of determinants comprehends the dividend-price (DP) and the earnings-price ratio (EP), commonly included in the set of the excess returns predictors, see also Welch and Goyal (2008). The well-known leverage effect (i.e., negative returns reflect higher volatility) is gathered through the equity market excess return (MKT). As a measure

of risk factors, the Fama and French's (1993) factors (High Minus Low, HML, and Small Minus Big, SMB) are considered in the analysis. The short-term reversal factor (STR) is included to capture the component of stock returns unexplained by "fundamentals."

A set of bond market variables enriches the set of determinants, as the Treasury bill (T-bill) rate, the rate of return on long-term government bond and the term spread difference (TS) of long-term bond yield three-month T-bill rate. The default spread (DEF) completes the set of financial determinants to approximate credit risk.

The inclusion of macroeconomic variables, as inflation rate and industrial production growth, follows Schwert (1989) and Engle, Ghysels, and Sohn (2009). Including these variables permits to assess whether volatility is countercyclical or not. A description of the variables in the data is shown in Table 1.

## 2 Neural Networks

ANNs can be seen as nonparametric tools, inspired by the structure of the human brain, for modeling and predicting the unknown function generating the observed data (Arnerić, Poklepovic, and Aljinović, 2014). The structure of the network can be modified to approximate a wide range of statistical and econometric models. For this reason, ANNs have been widely employed to forecast time series in different areas, like finance, medicine, biology, engineering, and physics. Empirical research indicates that ANNs are particularly suitable for forecasting volatile financial variables that exhibit nonlinear behaviors, like stock market returns or stock market volatility (Maheu and McCurdy, 2002), since they are capable of detecting nonlinear structure that linear models cannot detect. In this way, the researcher can implement neural networks without any a priori knowledge of the data generating process.

The neural network is specified as a collection of neurons (or nodes), grouped in layers, that connect to each other. The nodes of a layer are connected to the nodes of the following layer through weights and an activation function.[1] There exists a wide variety of learning algorithms to obtain these weights, the most popular being the backpropagation (BP). This algorithm is based on the gradient descent rule and allows to update the weights at each iteration, until there is no improvement in the error function, which is typically defined as the *mean squared error* (MSE).[2]

When the size of the network is too large, because of the number of hidden layers and hidden nodes, the training algorithm can be very slow. Although some rules have been suggested in the literature to find the optimal number of hidden layers and neurons (Gnana Sheela and Deepa, 2013), there is no commonly agreed solution to this issue. Stinchcombe and White (1992) proved that a single hidden neural network is a *universal approximator*, meaning that the network can approximate a wide range of linear and nonlinear functions,

1 An activation function is implemented in order to introduce nonlinearity to the network. Many activation functions, like sigmoid, hyperbolic tangent, and exponential, can be used in this framework, provided that they satisfy the condition of differentiability to apply the chain rule in the BP algorithm.

2 Other loss functions can be also implemented, such as mean absolute error and mean absolute percentage error.

**Table 1** Variables description

| Symbol | Variable | Data source | |
|---|---|---|---|
| | | Description | Source |
| DP | Dividend yield ratio S&P | Dividends over the past year relative to current market prices; S&P 500 index | Robert Shiller's website |
| EP | Earning price ratio S&P 500 | Earnings over the past year relative to current market prices; S&P 500 index | Robert Shiller's website |
| MKT | Market excess return | Fama–French's market factor: return of U.S. stock market minus one-month T-bill rate | Kenneth French's website |
| HML | Value factor | Fama–French's HML factor: average return on value stocks minus average return on growth stock | Kenneth French's website |
| SMB | Size premium factor | Fama–French's SMB factor: average return on small stocks minus average return on big stocks | Kenneth French's website |
| STR | Short-term reversal factor | Fama–French's STR: average return on stocks with low prior return minus average return on stock with high prior return | Kenneth French's website |
| TB | T-bill rate | Three-month T-bill rate | Datastream |
| TS | Term spread | Difference of long-term bond yield and three-month T-bill | Datastream |
| DEF | Default spread | Measure of default risk of corporate bonds: difference of BAA and AAA bond yields | Datastream |
| INF | Monthly Inflation | US inflation rate | Datastream |
| IP | Monthly industrial production growth rate | US industrial production growth | OECD database |

if a sufficient number of hidden nodes is included. For this reason, a single hidden layer network is assumed throughout the present article. Assuming then a three-layer neural network and a single output variable, the output function is of the form:

$$f_t(x_t, \theta) = F\left(\beta_0 + \sum_{j=1}^{q} G\left(x_t \gamma_j'\right)\beta_j\right), \tag{2}$$

where $F$ is the output activation function, $G$ is the hidden units' activation function, $\beta_j$, with $j = 1, \ldots, q$, are the weights from hidden unit $j$ to the output unit, $x_t = \{1, x_{1,t}, \ldots, x_{s,t}\}$ is the $1 \times m$ vector of input variables at time $t$ (with $m = s + 1$), $\beta_0$ is the bias of the final output, $\gamma_j = \{\gamma_{1,j}, \ldots, \gamma_{m,j}\}$ is the $1 \times m$ vector of weights for the connections between the inputs and the hidden neuron $j$, $q$ is the number of hidden units, and $\theta = \{\beta_0, \ldots, \beta_q, \gamma_1', \ldots, \gamma_q'\}$ is the vector of all network weights. This version, with three input variables including the bias and two hidden nodes (i.e., $m = 3$ and $j = 2$), is depicted in Figure 3 and assumes that information moves forward from the input layer to the output layer. Accordingly, it is also called *FNN*.

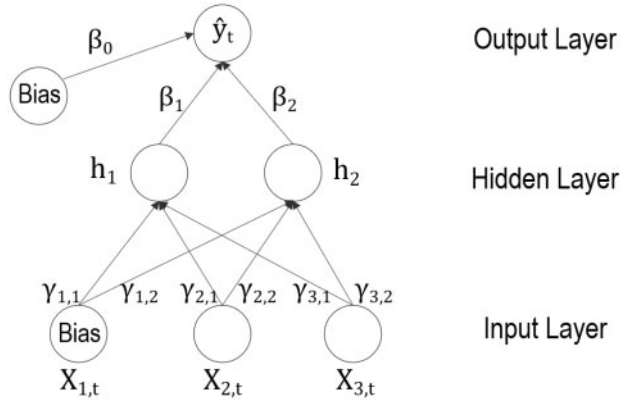



**Figure 3** FNN with a single hidden layer.

Modern practice allows choosing $F$ and $G$ among a variety of functions. In the most used form of FNN, the output activation function is an identity function, that is, $F(a) = a$. In this case, Equation (2) can be written as follows

$$f_t(x_t, \theta) = \beta_0 + \sum_{j=1}^{q} G\left(x_t \gamma_j'\right) \beta_j. \tag{3}$$

A common choice for $G$ is the logistic function, that is, $G(a) = \frac{1}{1+e^{-a}}$, although any continuous, differentiable, and monotonic function may be implemented. This function, bounded between 0 and 1, permits the network to reproduce any nonlinear pattern and replicate the way a real neuron becomes active. In particular, the neuron shows a high level of activation for $G$ close to 1, while it exhibits a poor response when $G$ is close to 0.

Researchers usually refer to FNN as a *static network*, since a given set of input variables is used to forecast the target output variable at time $t$. Hence, feed-forward networks show no memory, even when sample information exhibits temporal dependence. The so-called *recurrent neural networks* (RNNs) overcome this shortcoming by allowing internal feedbacks. This type of networks allows propagating data from input to output, but also from later layers to earlier layers. Such models have many potential applications in economic and finance, when nonlinear time dependence and long-memory exist. For this reason, the use of RNN in forecasting volatility has attracted a large number of researchers (Schittenkopf, Dorffner, and Dockner, 2000; Tino, Schittenkopf, and Dorffner, 2001). This article focuses on four recurrent architectures: Elman and Jordan recurrent networks, LSTM networks, and NARX neural networks.

In the ENN, proposed by Elman (1990), the input layer has additional neurons which are fed back from the hidden layer (see Figure 4). The output of the ENN, with an identity function as output activation function, can be represented as

$$\begin{aligned} f_t(x_t, \theta) &= \beta_0 + \sum_{j=1}^{q} h_{tj} \beta_j \\ h_{tj} &= G\left(x_t \gamma_j' + h_{t-1} \delta_j'\right) \qquad j = 1, \ldots, q \end{aligned} \tag{4}$$

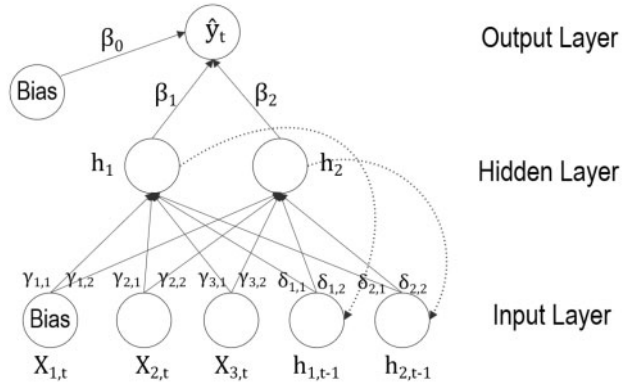



**Figure 4** ENN with a single hidden layer.

where $h_{t-1} = (h_{t-1,1}, \ldots, h_{t-1,q})$ is the vector of lagged hidden-unit activations, and $\delta_j = \{\delta_{1,j}, \ldots, \delta_{q,j}\}$ is the vector of connection weights between the $j$th hidden unit and the lagged hidden units.

Jordan (1986), instead, introduced an RNN with a feedback from the output layer, as in Figure 5. Thus, the network output at time $t-1$ is used as additional input for the network at time $t$. Specifically, the output of the JNN can be specified as follows

$$f_t(x_t, \theta) = \beta_0 + \sum_{j=1}^{q} G\left(x_t \gamma_j' + \hat{y}_{t-1}\psi_j\right)\beta_j \tag{5}$$

where $\hat{y}_{t-1}$ is equal to $f_{t-1}(x_{t-1}, \theta)$ and $\psi$ is the weight between the lagged output and the $j$th hidden unit.

Equations (4) and (5) indicate that the outputs of these RNNs can be expressed in terms of current and past inputs. This makes them similar to the distributed lag model or Autoregressive (AR) representation of an Autoregressive Moving Average (ARMA) model. Furthermore, differently from FNNs, RNNs are able to incorporate information of past observations without including them in the network.

Although extremely appealing, ENN and JNN suffer from the so-called "vanishing gradient problem." In such methods, the network weights are updated through a training algorithm based on the gradient descent rule. When this kind of algorithm is implemented, the magnitude of the gradients gets exponentially smaller (vanishes) at each iteration, making the steps very small and resulting in an extremely slow learning process. In such cases, a local minimum might be reached.

One of the cause of this shortcoming is the choice of the activation function. For example, a logistic activation function maps all the input values in a relatively small range, that is [0,1]. As a result, even a large change in the input will produce a small change in the output, vanishing the gradient very fast.

LSTM was introduced by Hochreiter and Schmidhuber (1997) to alleviate the vanishing gradient problem through a mechanism based on memory cells. LSTM extends the RNN architecture by replacing each hidden unit with a memory block. Each block contains one or more self-connected memory cells and is equipped with three multiplicative units called

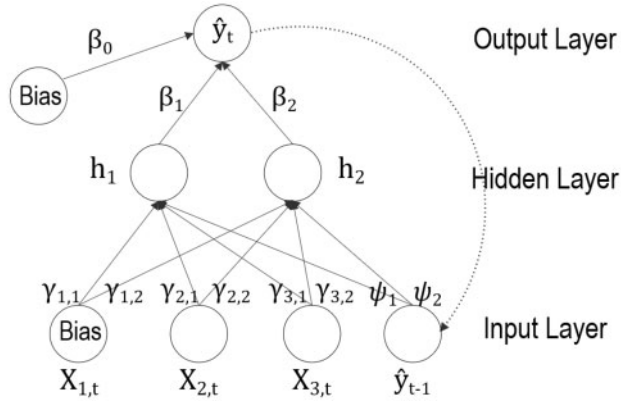



**Figure 5** JNN with a single hidden layer.

input, forget, and output gates. These gates allow the memory cells to store and access information, in order to determine which information should be persisted. In this way, LSTMs are capable of retaining relevant information of input signals, overlooking the unnecessary parts.

Figure 6 illustrates the structure of a simple LSTM memory block with a one cell architecture. In the figure, $x_t$ denotes the vector of input variables at time $t$, $c_t$ and $c_{t-1}$ correspond to the cell state at time $t$ and $t-1$, respectively, while $h_t$ and $h_{t-1}$ denote the hidden state or output of the cell at time step $t$ and $t-1$, respectively. The input gate is identified by $i_t$, $f_t$ indicates the forget gate, while $o_t$ is the output gate. Both input and output gates have the same role as in the RNNs. The new instance, that is, the forget gate, is responsible for removing the unnecessary information from the cell state. The information at time $t$, given by $x_t$ and $h_{t-1}$, is passed through the forget gate $f_t$, which determines if the information should be retained or not using a sigmoid function. Basically, a zero response of the sigmoid function means that the information should be discarded, while a value close to 1 implies that the information should be stored. Meanwhile, the same information is processed by the input gate to add information to the cell state $c_t$. Additionally, a nonlinear layer, $\phi = \tanh$, is introduced to generate a vector of candidate values, $\tilde{c}_t$, to update the state of $c_t$. The output gate is used to regulate the output values of an LSTM cell, using a logistic function to filter the output. The final output of the memory cell, $h_t$, is then computed by feeding the cell state, $c_t$, into a *tanh* layer and multiplying it by the value of the output gate. The entire process can be synthesized by the following equations:

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f) \tag{6}$$
$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i) \tag{7}$$
$$\tilde{c}_t = \tanh(W_c h_{t-1} + U_c x_t + b_c) \tag{8}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{9}$$
$$o_t = \sigma(W_o h_{t-1} + U_o x_t + V_o c_t + b_o) \tag{10}$$
$$h_t = o_t \odot \tanh(c_t) \tag{11}$$
$$\hat{y}_t = h_t \tag{12}$$

where $W_f$, $W_i$, $W_c$, $W_o$, $U_f$, $U_i$, $U_c$, and $U_o$ are the weight matrices of forget, input, memory

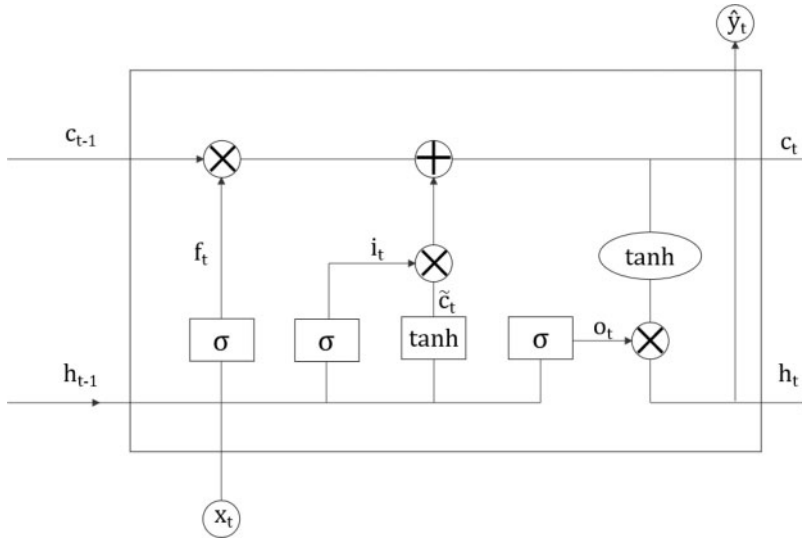



**Figure 6** Basic LSTM memory cell.

*Notes*: The memory cell has four key components: an input gate, a neuron with self-current connection, a forget gate, and an output gate. The inputs (the predictors at time $t$ and the outputs of the previous steps) are passed through the memory cell with some non-linear and linear interactions. Linear interactions of the cell state are point-wise addition $\oplus$ and point-wise multiplication $\otimes$. Non-linear interactions are logistic functions, $\sigma$.

cell state, and output gates, respectively, $V_c$ is the weight matrix of the cell state, $\hat{y}_t$ is the output of the neural network, $b_f$, $b_i$, $b_c$, and $b_o$ are the biases of the related gates, $\sigma$ is a sigmoid or logistic function and $\odot$ is the Hadamard product function.

LSTM can operate where long-memory effects are present in the underlying structure of the times series, similarly to HAR or Autoregressive Fractional Integral Moving Average (ARFIMA) models. Accordingly, there are numerous applications of LSTM models in finance, see, for example, Heaton, Polson, and Witte (2016), Bao, Yue, and Rao (2017), Pichl and Kaizoji (2017), Kim and Won (2018), Di Persio and Honchar (2017), and Xiong, Nichols, and Shen (2016).

A further way to deal with long-term dependencies and mitigate the effect of the vanishing gradient problem is the NARX neural network. This network, introduced by Lin et al. (1996), addresses the vanishing gradient problem by using an orthogonal mechanism with direct connections or delays from the past. Some authors (Bianchi et al., 2017) showed that NARX networks accurately predict time series with long-term dependencies, while others (Menezes and Barreto, 2006) demonstrated that this method accurately forecasts nonlinear time series.

NARX networks can be specified in a two-fold way. The first mode is called *parallel (P) architecture*, in which the output is fed back to the input of the FNN. The NARX-P architecture behaves like a JNN where, at each training epoch, the output is trained and used in the subsequent time steps (differently from JNN, this architecture relies on a greater number of lags). The second mode is called *series-parallel (SP) architecture*, here the observed output is used as additional input instead of feeding back the estimated output. The

structure is that of a regular FNN with $d$ additional inputs equal to $d$ delays of the real target variable.

In this article, I consider only NARX-SP networks with zero input order and a one-dimensional output. Thus, the output function of the NARX networks with zero input order is defined by

$$\hat{y}_t = \Psi[x_t, y_{t-1}, \ldots, y_{t-d}] \tag{13}$$

where $x_t$ and $y_t$ are, respectively, the input and the output of the network at time $t$, $d$ is the output order and $\Psi$ is a multilayer perceptron as in Figure 7. This architecture can be represented by the following equation:

$$f_t(x_t, \theta) = \beta_0 + \sum_{j=1}^{q} G\left(x_t \gamma_j' + \sum_{d=1}^{n_d} y_{t-d} \psi_{d,j}\right) \beta_j \tag{14}$$

where $\psi_{d,j}$ is the weight associated to the $d$th delay of the output.

In the following section, I specify the architecture for the above models, selecting the final set of inputs, the number of hidden nodes, and the training algorithm.

## 3 Neural Networks Architecture

The overall task of constructing a neural network passes through a process of trial and error. Some authors, Anders and Korn (1996) and Panchal et al. (2010) among others, suggested various ways to define information criteria that could help driving the choice of the neural network architecture. However, the most reliable approach remains the training of different architectures and the choice of the network producing the lowest forecasting error.

First, the researcher should choose a set of inputs. Variable selection represents a crucial phase for the identification of the neural networks' architecture. While the initial set of determinants can be guided by the economic theory (see Section 1), a subset of these predictors should be used to reduce the number of weights to be trained (equal to $(1 + m)q + 1$) and enable algorithms to work properly. In the related literature, there are several methods to optimally detect the relevant explanatory variables. Here, I selected the variables through a Least Absolute Shrinkage and Selection Operator (LASSO) regression, introduced by Tibshirani (1996). This method performs estimation and model selection in the same step by penalizing the absolute size of the regression coefficients, based on a penalty coefficient, $\lambda$; see Zou (2006) for the mathematical details. To assess the results of the analysis, I examined which variables really affected realized volatility for two samples: the entire sample of observations, from January 1950 to December 2017, and a subsample,[3] from February 1973 to June 2009. All independent and control variables were lagged by one year to mitigate the possibility of simultaneity or reverse causality bias, while the number of lags of the dependent variables was assessed through information criteria. As further explained in the

3 This subsample was used to validate the approach in a more volatile period. The starting month of this sample has been determined through a breakpoint analysis, while the final monthly observation coincided with the end of the *Great Recession* according to the National Bureau of Economic Research.

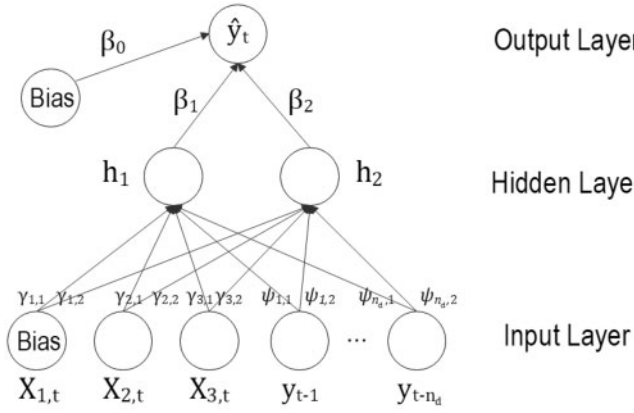



**Figure 7** Architecture of an NARX network.

next paragraph, the final set of variables was selected on the training sample, and the resulting sets of variables were $X_a = \{RV_{t-1}, RV_{t-2}, RV_{t-3}, DP_{t-1}, MKT_{t-1}, STR_{t-1}, DEF_{t-1}\}$ for the entire sample, and $X_b = \{RV_{t-1}, RV_{t-2}, RV_{t-3}, MKT_{t-1}, STR_{t-1}\}$ for the subsample. The lack of significance of pure macroeconomic variables, that is, inflation rate and industrial production growth, is in line with the findings of Schwert (1989) and Christiansen, Schmeling, and Schrimpf (2012), once again underlying the relevance of premium risk's determinants.

Choosing the set of explanatory variables entails a two-fold risk. On the one side, the so-called *look ahead bias*[4] may occur. On the other side, the variables selected through this method, that is, LASSO, may not be relevant in a neural network framework. The choice of two samples and two different sets of explanatory variables may help alleviating these drawbacks. Moreover, the former issue was circumvented by selecting the relevant variables on the training sample (see the following section for details), where the number of observations was approximately equal to two-thirds of the entire number of observations. Furthermore, the neural networks have been implemented without macroeconomic and financial determinants, in order to understand if the lags of the dependent variable, alone, were sufficient to provide accurate forecasts.

Once a set of determinants has been identified, the researcher can proceed to select the number of hidden layers and hidden neurons. To assess the performance of an architecture, the researcher must modify the number of hidden units or by adding or removing certain network connections, and then evaluate them by comparing the MSE attained in compared architectures.

As previously mentioned, a single hidden layer was assumed throughout the article, while the selection of the optimal number of hidden neurons was trickier. Since a standard and accepted method for determining the number of hidden nodes does not exist, I evaluated the performance of the networks[5] for each sample by the lowest training MSE for an

4 Look ahead bias involves using information not available during the period analyzed.

5 LSTM hidden units follow a different setting in comparison with other neural networks; thus, the number of hidden units is set to fifty, comparably to similar studies.

increasing number of hidden nodes, where the maximum number of hidden nodes was equal to the total number of inputs (i.e., 7 and 5, respectively), as suggested by Tang and Fishwick (1993). To avoid the optimization algorithm being trapped in a local minimum, the network weights were re-estimated using 300 sets of random starting values. Table 2 provides the MSE for each architecture in the entire sample, while the results for the subsample are showed in Table 3. Therefore, the number of hidden nodes was selected according to the lowest MSE. As in the case of explanatory variables selection, the choice of the architecture was made on the training samples.

A gradient descent with momentum and adaptive learning rate (*gdx*) BP has been used to train the feed-forward, Elman, Jordan, and LSTM architectures. Despite it converges more slowly in comparison to other algorithms, the trained weights iteratively adapt to the shape of the error surface at each iteration, reducing the risk of a local minimum. The NARX network has been trained using a Bayesian regularization (BR) algorithm, since the predictive performance of the BR algorithm is more robust when an NARX architecture is implemented, see Guzman, Paz, and Tagert (2017).

## 4 Assessing Forecast Accuracy

The forecasting ability of the ANNs was compared to an autoregressive fractionally integrated moving average with the same set of explanatory variables selected in the previous section (ARFIMAX) and without determinants (ARFIMA). The set of competing models also included a logistic smooth transition autoregressive model (LSTAR), also entailing exogenous variables (LSTARX), where the number of lags was set relying on the Akaike and the Bayesian information criteria. The analysis was performed over a period from January 1950 to December 2017 and a period from February 1973 to June 2009.

Lag selection of the ARFIMA was assessed on the training sample through information criteria. ARFIMA(0, d , 1) and ARFIMAX(0, d , 0) were selected for the larger sample, while ARFIMA(0, d , 0) and ARFIMAX(2, d , 2) were used for the subsample. In order to determine the number of regimes of the smooth transition models, the presence of structural breaks was evaluated through the method introduced by Bai and Perron (2003). A single structural break (and two regimes) was identified, while lagged realized volatility was used as transition variable.

The forecasting accuracy of the neural networks model was further compared with the forecasts from an HAR model for realized volatility, discussed in Andersen, Bollerslev, and Diebold (2007) and Corsi (2009). Since the realized volatility used in this article was observed monthly, we made use of quarterly and annual aggregation periods.

Realized volatility forecasts were produced on 245 out-of-sample observations (from August 1997 to December 2017) for the entire sample, while 22 out-of-sample forecasts (from September 2007 to June 2009) were produced for the subsample. The number of out-of-sample observations was equal to one-third of the entire sample in the former case, while it started from the beginning of the *Great Recession* for the latter. This helped understanding whether NNs were able to outperform econometric models in a highly volatile context and in the presence of greater persistence.

**Table 2** MSE for increasing number of hidden nodes—entire sample

| Model | No. of hidden | Performance | No. of weights | Model | No. of hidden | Performance | No. of weights |
|---|---|---|---|---|---|---|---|
| FNNX | 1* | **0.1029** | 10 | FNN | 1 | 0.1168 | 6 |
| | 2 | 0.1245 | 19 | | 2 | 0.1260 | 11 |
| | 3 | 0.1062 | 28 | | 3 | 0.1177 | 16 |
| | 4 | 0.1074 | 37 | | 4 | 0.1171 | 21 |
| | 5 | 0.1035 | 46 | | 5* | **0.1136** | 26 |
| | 6 | 0.1069 | 58 | | 6 | 0.1168 | 31 |
| | 7 | 0.1063 | 71 | | 7 | 0.1175 | 36 |
| ENNX | 1 | 0.1027 | 11 | ENN | 1 | 0.1177 | 7 |
| | 2 | 0.1111 | 23 | | 2 | 0.1479 | 15 |
| | 3 | 0.1031 | 37 | | 3 | 0.1179 | 25 |
| | 4* | **0.1006** | 53 | | 4 | 0.1230 | 37 |
| | 5 | 0.1131 | 71 | | 5 | 0.1210 | 51 |
| | 6 | 0.1070 | 91 | | 6 | 0.1223 | 67 |
| | 7 | 0.1115 | 113 | | 7* | **0.1170** | 85 |
| JNNX | 1 | 0.1006 | 11 | JNN | 1 | 0.1165 | 7 |
| | 2 | 0.1236 | 21 | | 2 | 0.1315 | 13 |
| | 3* | **0.1004** | 31 | | 3 | 0.1148 | 19 |
| | 4 | 0.1044 | 41 | | 4 | 0.1158 | 25 |
| | 5 | 0.1040 | 51 | | 5* | **0.1147** | 31 |
| | 6 | 0.1062 | 61 | | 6 | 0.1152 | 37 |
| | 7 | 0.1085 | 71 | | 7 | 0.1154 | 43 |
| NARX | 1 | 0.0990 | 10 | NAR | 1 | 0.1141 | 6 |
| | 2 | 0.0981 | 19 | | 2 | 0.1123 | 11 |
| | 3 | 0.0978 | 28 | | 3 | 0.1160 | 16 |
| | 4 | 0.0968 | 37 | | 4 | 0.1123 | 21 |
| | 5* | **0.0953** | 46 | | 5 | 0.1110 | 26 |
| | 6 | 0.0967 | 55 | | 6 | 0.1113 | 31 |
| | 7 | 0.0966 | 64 | | 7* | **0.1100** | 36 |

*Notes*: The table includes the number of hidden nodes, the performance in terms of MSE, and the number of weights trained for each architecture. Each architecture has a maximum of iterations equal to 1000. The presence of the *X* in the name of the model indicates the use of exogenous determinants other than lagged realized variance. The asterisk denotes the selected number of hidden nodes. Values in boldface represent the lowest MSE.

The one-step-ahead ($k = 1$) out-of-sample forecasts were generated from a rolling window scheme, re-estimating the parameters at each step. In addition, multistep-ahead forecasts have been considered. The five-step-ahead ($k = 5$) forecasts were iteratively produced from a rolling window estimation. At each step ahead, the information was updated with the prediction of the previous step. The resulting set of variables used to make the forecasts five-step ahead is the following:

**Table 3** MSE for increasing number of hidden nodes—subsample

| Model | No. of hidden | Performance | No. of weights | Model | No. of hidden | Performance | No. of weights |
|---|---|---|---|---|---|---|---|
| FNNX | 1 | 0.1450 | 8 | FNN | 1 | 0.1534 | 6 |
| | 2 | 0.1745 | 15 | | 2 | 0.1751 | 11 |
| | 3* | **0.1196** | 22 | | 3* | **0.1474** | 16 |
| | 4 | 0.1467 | 29 | | 4 | 0.1491 | 21 |
| | 5 | 0.1534 | 36 | | 5 | 0.1484 | 26 |
| ENNX | 1 | 0.1192 | 9 | ENN | 1 | 0.1298 | 7 |
| | 2 | 0.1693 | 19 | | 2 | 0.1667 | 15 |
| | 3 | 0.1158 | 31 | | 3 | 0.1425 | 25 |
| | 4* | **0.1126** | 45 | | 4* | **0.1420** | 37 |
| | 5 | 0.1205 | 61 | | 5 | 0.1471 | 51 |
| JNNX | 1 | 0.1201 | 9 | JNN | 1* | **0.1425** | 7 |
| | 2 | 0.1289 | 17 | | 2 | 0.1541 | 13 |
| | 3* | **0.1115** | 25 | | 3 | 0.1494 | 19 |
| | 4 | 0.1176 | 33 | | 4 | 0.1498 | 25 |
| | 5 | 0.1169 | 41 | | 5 | 0.1478 | 31 |
| NARX | 1 | 0.1111 | 8 | NAR | 1 | 0.1145 | 6 |
| | 2 | 0.1009 | 15 | | 2 | 0.1291 | 11 |
| | 3 | 0.1119 | 22 | | 3* | **0.1056** | 16 |
| | 4* | **0.0998** | 29 | | 4 | 0.1177 | 21 |
| | 5 | 0.1020 | 36 | | 5 | 0.1158 | 26 |

*Notes*: The table includes the number of hidden nodes, the performance in terms of MSE, and the number of weights trained for each architecture. Each architecture has a maximum of iterations equal to 1000. The presence of the $X$ in the name of the model indicates the use of exogenous determinants other than a lagged variance. The asterisk denotes the selected number of hidden nodes. Values in boldface represent the lowest MSE.

$$\hat{y}_{t+1} = \{y_t, y_{t-1}, y_{t-2}, z_t\}$$
$$\hat{y}_{t+2} = \{\hat{y}_{t+1}, y_t, y_{t-1}, z_t\}$$
$$\hat{y}_{t+3} = \{\hat{y}_{t+2}, \hat{y}_{t+1}, y_t, z_t\}$$
$$\hat{y}_{t+4} = \{\hat{y}_{t+3}, \hat{y}_{t+2}, \hat{y}_{t+1}, z_t\}$$
$$\hat{y}_{t+5} = \{\hat{y}_{t+4}, \hat{y}_{t+3}, \hat{y}_{t+2}, z_t\},$$

where $z_t = \{\mathrm{DP}_{t-1}, \mathrm{MKT}_{t-1}, \mathrm{STR}_{t-1}, \mathrm{DEF}_{t-1}\}$ in the entire sample, and $z_t = \{\mathrm{MKT}_{t-1}, \mathrm{STR}_{t-1}\}$ in the subsample. The set of input variables used in models ARFIMA, LSTAR, HAR, FNN, ENN, JNN, LSTM, and NAR did not include $z_t$.

The relative performance of the out-of-sample forecasting accuracy was assessed using MSE and the quasi-likelihood (QLIKE), which belong to the family of loss functions robust to a noisy volatility proxy; see Patton (2011). The predictive performance of the competing models was also simultaneously compared via *model confidence set* (MCS), introduced by Hansen, Lunde, and Nason (2011). The MCS procedure consists in a sequence of equal predictive accuracy tests through which a set of superior models (SSM) is defined, given a certain confidence level. For a set of forecasts from $M$ models, MCS tests, through a pairwise

comparison of loss difference $d_{l,j,t}$ from model $l$ and model $j$, whether all models provide equal predictive accuracy. Assuming $d_{l,j,t}$ stationary, the null hypothesis assumes the following form:

$$H_0 : E[d_{l,j,t}] = 0, \quad \forall l, j \in M. \tag{15}$$

Given a confidence level $\alpha$, a model is discarded when the null hypothesis of equal forecasting ability is rejected. The SSM is then defined as the set of models not-rejecting the null hypothesis. The $p$-values were computed using a stationary bootstrap. The lower the $p$-value of an object, the lower the probability of being included in the SSM, see Hansen, Lunde, and Nason (2011) for further details.

As shown by the average of the loss functions in Table 4, the majority of the neural networks were able to outperform the traditional long-memory detecting models from Panel A in the larger sample, when analyzing forecasts for $k = 1$. The unique model exhibiting an MSE and QLIKE in line with NNs, or better in some comparisons with Panel B, was the HAR model. In most cases, the exclusion of the explanatory variables worsened the forecasting accuracy, highlighting that the dynamics of realized volatility are somehow linked to macroeconomic and financial conditions. The lowest loss functions were provided by the models in Panel C, where the long-memory neural networks were stored. The best overall performance in terms of forecasting accuracy measures was exhibited by LSTMX and NARX in Panel C, which exhibited the lowest MSE and QLIKE. NARX was the unique model belonging to the 75% MCS ($\hat{\mathcal{M}}^*_{75\%}$), regardless of the loss function considered. Even though, the average loss functions varied differently according to the kind of loss considered. For example, in Panel C the QLIKE function was almost equal for all the models, while MSE was severely smaller in the models entailing the use of macroeconomic and financial variables. This may be driven by the definition of the loss functions. In fact, MSE is a symmetric measure, while QLIKE penalizes more negative biases. In this context, all the models seemed to underestimate the observed RV (see also Figures 8 and 9), meaning that the the differences between real RV and the forecasts tended to be positive and the QLIKE more in line among compared models.

The analysis of multistep-ahead forecasts, in the entire sample, further highlighted the predictive ability of long-term memory detecting RNNs. In fact, NARX and LSTMX neural networks exhibited the lowest MSE and QLIKE, excluding the overall best performance of the HAR model. Compared to one-step-ahead forecasts, the differences in terms of average losses in multistep-ahead forecasts were much less pronounced between the models compared. This emerged also from the $p$-values, which assumed values not close to zero for almost all the models, and from the fact that the majority of the models was included in the 75% MCS ($\hat{\mathcal{M}}^*_{75\%}$). An explanation of these results derives from the nature of the forecasts, since in multistep-ahead forecasts, the information used is the same for repeated steps, flattening the results from models highly different. The average losses of the models in Panel A were higher than the most part of the models in Panels B and C, with the only exception of HAR model's losses and the QLIKE loss for the ARFIMAX model. Finally, among the models belonging to 75% MCS ($\hat{\mathcal{M}}^*_{75\%}$), the $p$-values were higher for HAR model, JNNs, LSTMX, NAR, and NARX neural networks, both for MSE and QLIKE.

In the more volatile period of time, September 2007–June 2009, in Table 5, classical long-memory detecting models in Panel A seemed not accurate in terms of one-step-ahead

**Table 4** MCS with 10,000 bootstraps (entire sample: 1997.08–2017.12)

| Model | $k=1$ | | | | $k=5$ | | | |
|---|---|---|---|---|---|---|---|---|
| | MSE | | QLIKE | | MSE | | QLIKE | |
| | Loss | $P_{MCS}$ | Loss | $P_{MCS}$ | Loss | $P_{MCS}$ | Loss | $P_{MCS}$ |
| Panel A: time series models | | | | | | | | |
| ARFIMAX | 0.167 | 0.000 | 3.317 | 0.000 | 0.185 | 0.683** | 3.323 | 0.867** |
| ARFIMA | 0.205 | 0.000 | 3.325 | 0.000 | 0.219 | 0.201* | 3.333 | 0.147* |
| LSTARX | 0.145 | 0.000 | 3.316 | 0.000 | 0.323 | 0.000 | 3.352 | 0.000 |
| LSTAR | 0.130 | 0.000 | 3.311 | 0.000 | 0.245 | 0.205* | 3.332 | 0.267** |
| HAR | 0.115 | 0.000 | 3.308 | 0.000 | **0.102** | 1.000** | **3.306** | 1.000** |
| Panel B: FFN, JNN, and ENN | | | | | | | | |
| FNNX | 0.132 | 0.000 | 3.313 | 0.000 | 0.178 | 0.680** | 3.325 | 0.212* |
| FNN | 0.130 | 0.000 | 3.311 | 0.000 | 0.176 | 0.757** | 3.325 | 0.308** |
| ENNX | 0.133 | 0.000 | 3.313 | 0.000 | 0.164 | 1.000** | 3.321 | 1.000** |
| ENN | 0.138 | 0.000 | 3.312 | 0.000 | 0.172 | 0.992** | 3.322 | 0.872** |
| JNNX | 0.136 | 0.000 | 3.314 | 0.000 | 0.158 | 1.000** | 3.315 | 1.000** |
| JNN | 0.134 | 0.000 | 3.312 | 0.000 | 0.161 | 1.000** | 3.314 | 1.000** |
| Panel C: long-term dependence detecting neural networks | | | | | | | | |
| LSTMX | **0.042** | 0.005 | **3.293** | 0.004 | 0.152 | 1.000** | **3.313** | 1.000** |
| LSTM | 0.110 | 0.000 | 3.307 | 0.000 | 0.185 | 0.639* | 3.327 | 0.025 |
| NARX | **0.018** | 1.000** | **3.288** | 1.000** | **0.146** | 1.000** | 3.316 | 1.000** |
| NAR | 0.075 | 0.000 | 3.301 | 0.003 | 0.164 | 1.000** | 3.317 | 1.000** |

*Notes*: This table reports the average loss over the evaluation sample and the MCS *p*-values calculated on the basis of the range statistics. The realized volatility forecasts in $\hat{\mathcal{M}}^{*}_{90\%}$ and $\hat{\mathcal{M}}^{*}_{75\%}$ are identified by one and two asterisks, respectively. Values in boldface represent the lowest average losses. The set of input variables used in models ARFIMA, LSTAR, HAR, FNN, ENN, JNN, LSTM, and NAR did not include exogenous variables other than the lags of the dependent variable.

forecasts. Nevertheless, ARFIMAX model exhibited an average MSE and QLIKE in line with the compared neural networks. Once again, the lowest loss functions were exhibited by two models from Panel C, that is, LSTM and NARX. From a theoretical point of view, this result is not surprising, given that the LSTM has shown stronger performance in similar works in presence of long dependencies (Heaton, Polson, and Witte, 2016; Pichl and Kaizoji, 2017). Furthermore, the prediction differences between neural networks and linear models may indicate a nonlinear behavior of the log-realized variance during financially stressed periods; see also Choudhry, Papadimitriou, and Shabi (2016). In this volatile framework, the five-step-ahead forecasts provided mixed results. All the models were included in the 90% MCS when the MSE was used as loss function, while only two models, LSTARX and LSTM, were not included in the 75% MCS ($\hat{\mathcal{M}}^{*}_{75\%}$). A similar result was obtained with QLIKE as a loss function. Among models without exogenous variables, FNN exhibited the lowest average losses, while the most part of the simple neural networks from Panel B showed average losses lower than the more complex networks from Panel C. This may imply that a simpler model should be implemented when few multistep-ahead forecasts need to be produced.

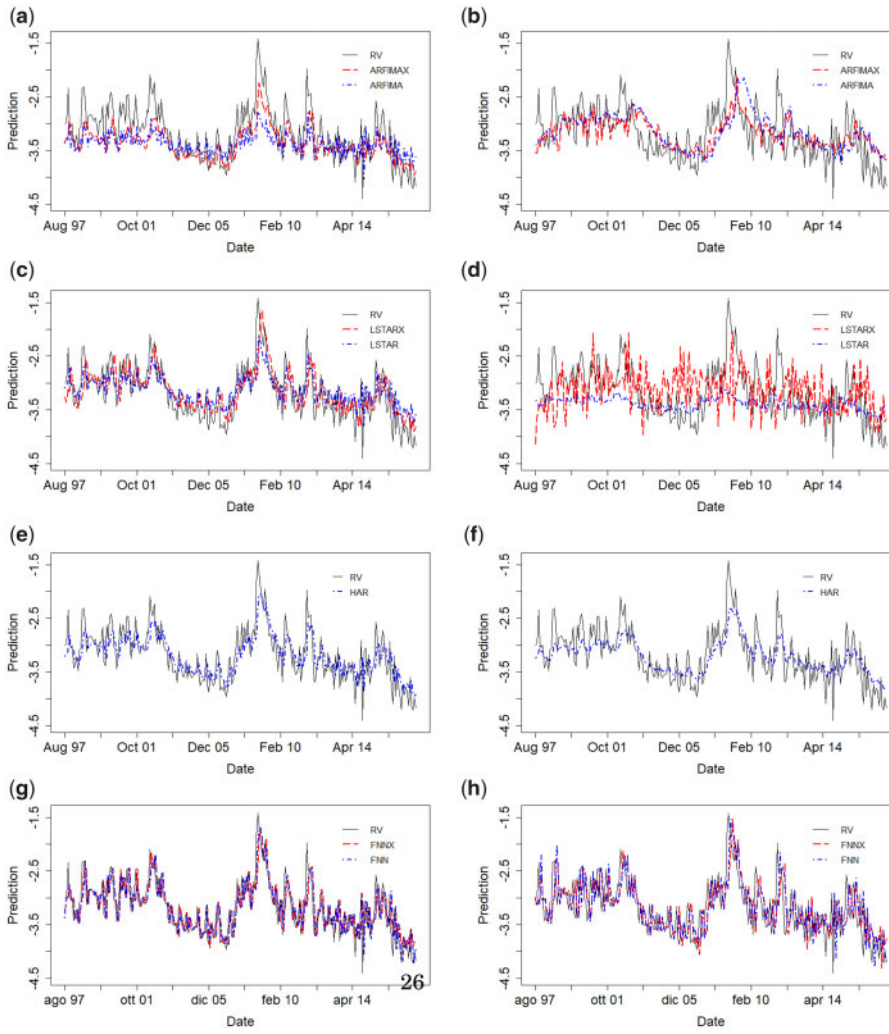



**Figure 8** Entire sample forecasts comparison. The black line represents the realized volatility of S&P 500, the red dashed line describes the out-of-sample forecasts with exogenous variables, while the blue dashed line depicts the forecasts from a model without financial and macroeconomic variables. The left panel entails the one-step-ahead forecasts, while five-step-ahead forecasts are showed in the right column. (a) ARFIMA one-step-ahead forecasts. (b) ARFIMA five-step-ahead forecasts. (c) LSTAR one-step-ahead forecasts. (d) LSTAR five-step-ahead forecasts. (e) HAR one-step-ahead forecasts. (f) HAR five-step-ahead forecasts. (g) FNN one-step-ahead forecasts. (h) FNN five-step-ahead forecasts. (i) ENN one-step-ahead forecasts. (j) ENN five-step-ahead forecasts. (k) JNN one-step-ahead forecasts. (l) JNN five-step-ahead forecasts. (m) LSTM one-step-ahead forecasts. (n) LSTM five-step-ahead forecasts. (o) NARX one-step-ahead forecasts. (p) NARX five-step-ahead forecasts.

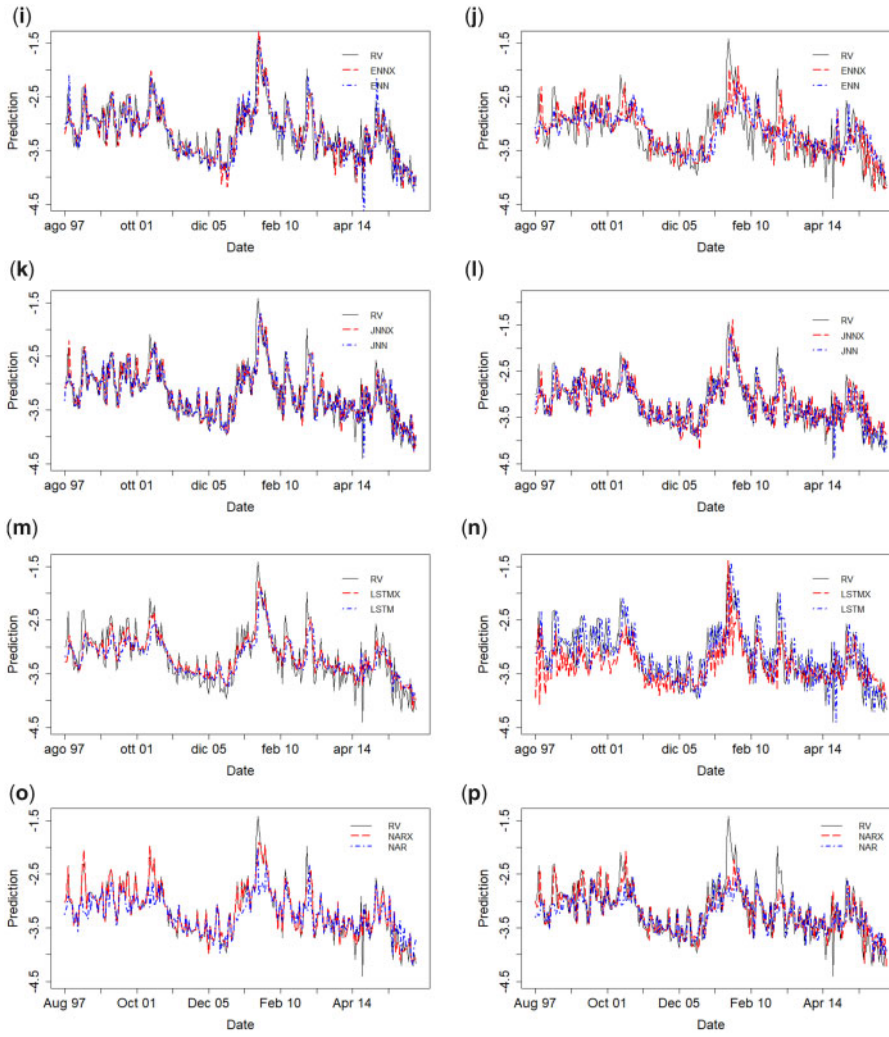
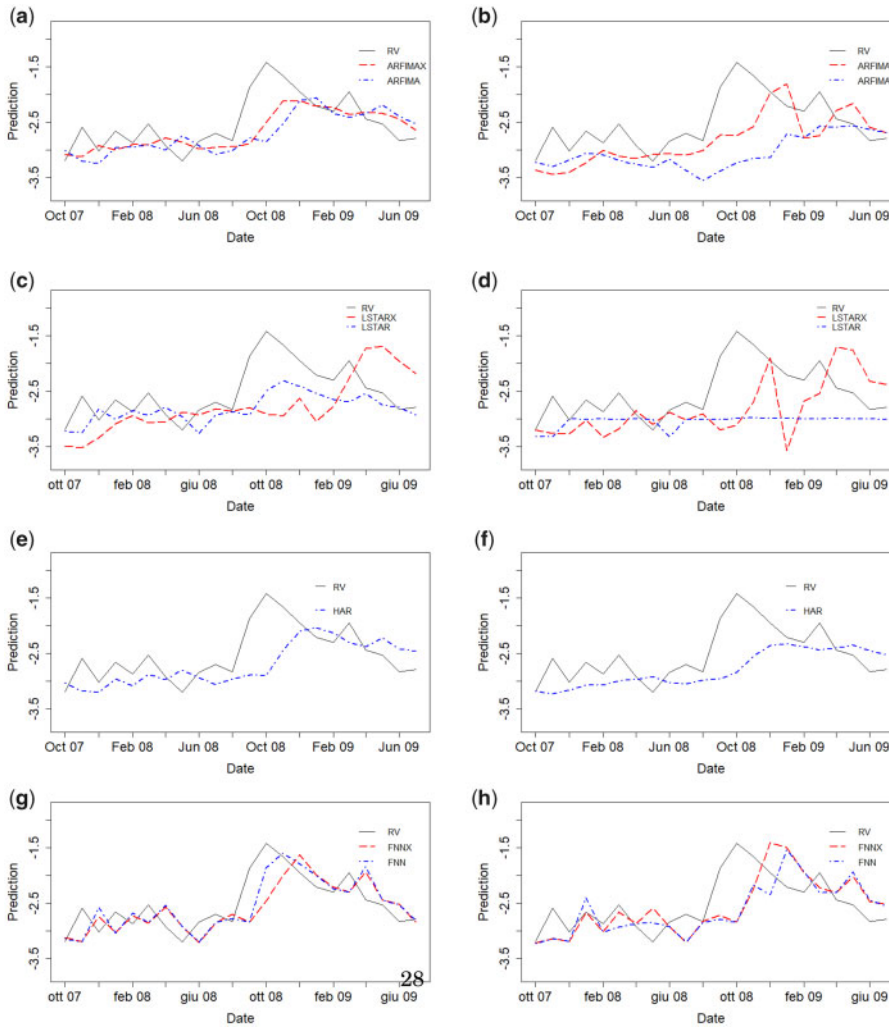



**Figure 8** Continued

Additionally, the test of equal predictive accuracy of Diebold–Mariano (DM) (Diebold and Mariano, 1995) was used as a robustness check. The pairwise comparison test in Table 6 supported our previous findings when $k = 1$ in both the samples, since the null of equal forecast accuracy was rejected only for long-memory detecting models in Panel C and for the HAR model. Thus, HAR, LSTM, and NARX models seemed to be the unique models predicting realized volatility one-step-ahead forecast better than the simple random walk model. A similar result was observed in the subsample when $k = 1$, where the forecast accuracy of the NARX and LSTM models was the unique significantly different from a random walk. When $k = 5$, neural networks model was able to significantly outperform the predictive accuracy of the benchmark method (i.e., $\hat{y}_{t+5} = y_t$), but the test statistics were




28

**Figure 9** Subsample sample forecasts comparison. The black line represents the realized volatility of S&P 500, the red dashed line describes the out-of-sample forecasts with exogenous variables, while the blue dashed line depicts the forecasts from a model without financial and macroeconomic variables. The left panel entails the one-step-ahead forecasts, while five-step-ahead forecasts are showed in the right column. (a) ARFIMA one-step-ahead forecasts. (b) ARFIMA five-step-ahead forecasts. (c) LSTAR one-step-ahead forecasts. (d) LSTAR five-step-ahead forecasts. (e) HAR one-step-ahead forecasts. (f) ) HAR five-step-ahead forecasts. (g) FNN one-step-ahead forecasts. (h) FNN five-step-ahead forecasts. (i) ENN one-step-ahead forecasts. (j) ENN five-step-ahead forecasts. (k) JNN one-step-ahead forecasts. (l) JNN five-step-ahead forecasts. (m) LSTM one-step-ahead forecasts. (n) LSTM five-step-ahead forecasts. (o) NARX one-step-ahead forecasts. (p) NARX five-step-ahead forecasts.

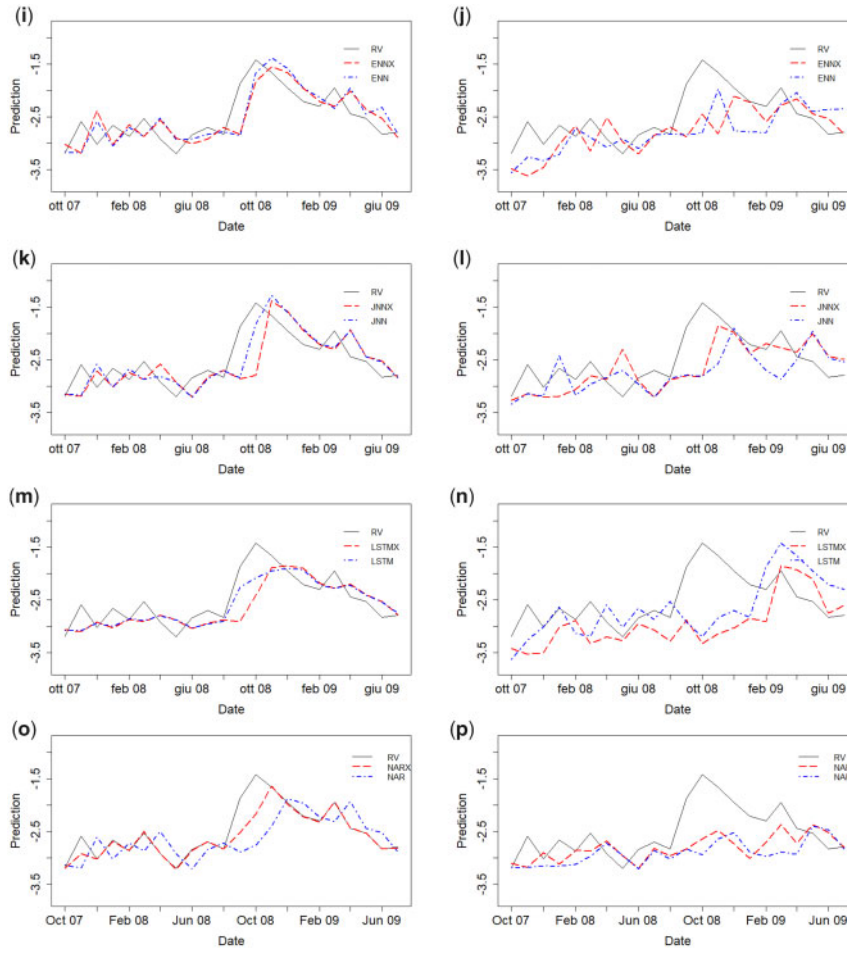



**Figure 9** Continued

comparable among the neural networks. Still, in the entire sample, the greatest significant difference between the forecasts of the benchmark and the models compared was found with the HAR (with a DM statistics of 7.428) and the NARX models, which provided a test statistic of 4.772 and 4.758. In the subsample, as already assessed by the MCS in Table 5, the test statistics were mostly comparable among the models.

Finally, the forecasts were compared pairwise via encompassing tests, discussed in Fair and Shiller (1989) and Chong and Hendry (1986). The idea behind encompassing tests is that, given the realized variable, $y_{t+k}$, and two sets of forecasts of the variable, $\hat{y}_{1,t+k}$ and $\hat{y}_{2,t+k}$, $y_{t+k}$ can be explained by pooling the forecasts as follows:

$$y_{t+k} = \alpha_0 + \alpha_1 \hat{y}_{1,t+k} + \alpha_2 \hat{y}_{2,t+k} + u_t.$$

**Table 5** MCS with 10,000 bootstraps (subsample: 2007.09–2009.06)

| Model | $k=1$ | | | | $k=5$ | | | |
|---|---|---|---|---|---|---|---|---|
| | MSE | | QLIKE | | MSE | | QLIKE | |
| | Loss | $P_{MCS}$ | Loss | $P_{MCS}$ | Loss | $P_{MCS}$ | Loss | $P_{MCS}$ |
| Panel A: time series models | | | | | | | | |
| ARFIMAX | 0.166 | 0.998** | 2.857 | 1.000** | 0.295 | 1.000** | 2.890 | 1.000** |
| ARFIMA | 0.244 | 0.306** | 2.878 | 0.546** | 0.571 | 0.593** | 2.939 | 0.890** |
| LSTARX | 0.467 | 0.000 | 2.957 | 0.000 | 0.503 | 0.121* | 2.947 | 0.134* |
| LSTAR | 0.221 | 0.000 | 2.872 | 0.449** | 0.488 | 0.440** | 2.930 | 0.813** |
| HAR | 0.247 | 0.453** | 2.878 | 0.653** | 0.263 | 1.000** | 2.885 | 1.000** |
| Panel B: FFN, JNN, and ENN | | | | | | | | |
| FNNX | 0.176 | 0.103 | 2.864 | 0.147* | 0.259 | 1.000** | 2.890 | 1.000** |
| FNN | 0.138 | 1.000** | 2.849 | 1.000** | **0.234** | 1.000** | **2.874** | 1.000** |
| ENNX | 0.142 | 1.000** | 2.850 | 1.000** | 0.286 | 1.000** | 2.883 | 1.000** |
| ENN | 0.143 | 1.000** | 2.853 | 1.000** | 0.283 | 1.000** | 2.887 | 1.000** |
| JNNX | 0.216 | 0.309** | 2.875 | 0.295** | **0.244** | 1.000** | **2.879** | 1.000** |
| JNN | 0.137 | 1.000** | 2.855 | 1.000** | 0.290 | 1.000** | 2.892 | 1.000** |
| Panel C: long-term dependence detecting neural networks | | | | | | | | |
| LSTMX | 0.151 | 1.000** | 2.853 | 1.000** | 0.533 | 0.426** | 2.936 | 0.688** |
| LSTM | **0.080** | 1.000** | **2.833** | 1.000** | 0.460 | 0.148* | 2.952 | 0.137* |
| NARX | **0.052** | 1.000** | **2.824** | 1.000** | 0.266 | 1.000** | 2.884 | 1.000** |
| NAR | 0.237 | 0.170* | 2.878 | 0.260** | 0.358 | 1.000** | 2.905 | 0.740** |

*Notes*: This table reports the average loss over the evaluation sample and the MCS $p$-values calculated on the basis of the range statistics. The realized volatility forecasts in $\hat{\mathcal{M}}^{*}_{90\%}$ and $\hat{\mathcal{M}}^{*}_{75\%}$ are identified by one and two asterisks, respectively. Values in boldface represent the lowest average losses.

Forecasts encompassing can be tested by analyzing the significance of the coefficients. For example, if the hypothesis $\alpha_1 = 0$ is rejected, while $\alpha_2 = 0$ not, the first model encompasses the second one. Moreover, if the null hypothesis $\alpha_2 = 0$ and $\alpha_1 = 1$ cannot be rejected, the forecasts from model 1 are also unbiased. Thus, ideally a model should have a coefficient close to or greater than 1 to significantly encompass the compared model.

Since the number of out-of-sample forecasts is limited in the subsample, only the forecasts from the entire sample were compared via encompassing tests in Tables 7 and 8.

The comparison of one-step-ahead forecasts highlighted that the coefficient of the HAR in encompassing regressions is close to 1 when included in a regression with forecasts from models in Panels A and B; thus, the HAR forecast encompasses those forecasts. For instance, if we regress RV on the HAR and the FNN forecasts, we get a (statistically significant) coefficient on HAR of 0.845 (from the FNN column and HAR row), while we get a coefficient on the FNN forecast of 0.146 (from the HAR column and FNN row), suggesting that the HAR forecast encompasses the FNN forecast. In contrast, the HAR forecast is encompassed by the LSTM, LSTMX, and NARX forecasts. From Table 7, NARX emerges as the unique method able to encompass the other forecasts and with a (statistically significant) coefficient always near to 1. All the models from Panel C encompassed the forecasts from models in Panels A and B, HAR excluded.

**Table 6** DM test of equal predictive accuracy

| Model | Entire sample | | Subsample | |
|---|---|---|---|---|
| | $k = 1$ | $k = 5$ | $k = 1$ | $k = 5$ |
| Panel A: time series models | | | | |
| ARFIMAX | −1.714* | 4.284** | 1.648 | 2.527** |
| ARFIMA | −3.490*** | 2.660*** | 0.130 | 0.025 |
| LSTARX | −0.182 | −1.953* | −2.518** | 0.462 |
| LSTAR | 1.308 | 0.812 | 0.470 | 0.755 |
| HAR | 3.100*** | 7.428*** | 0.064 | 2.731** |
| Panel B: FFN, JNN, and ENN | | | | |
| FNNX | 1.454 | 3.552*** | 1.343 | 3.602*** |
| FNN | 2.457** | 3.523*** | 1.241 | 3.663*** |
| ENNX | 0.778 | 4.258*** | 1.191 | 2.542** |
| ENN | −0.232 | 4.015*** | 1.129 | 2.557** |
| JNNX | 0.203 | 4.009*** | 0.904 | 2.466** |
| JNN | 0.496 | 4.044*** | 1.251 | 2.480** |
| Panel C: long-term dependence detecting neural networks | | | | |
| LSTMX | 7.647*** | 4.457*** | 1.761* | 0.271 |
| LSTM | 2.713*** | 3.075*** | 2.264** | 0.867 |
| NARX | 9.179*** | 4.772*** | 3.057*** | 3.010*** |
| NAR | 7.707*** | 4.758*** | 0.276 | 1.788* |

*Notes*: This table reports the *t*-statistics for the DM test where the null hypothesis is the equivalence of the predictive accuracy of the compared models with the information available at time $t$ (i.e., $\hat{y}_{t+h} = y_t$). *, **, and *** indicate a significant difference between the forecasting abilities at 1%, 5%, and 10% level, respectively. A positive and statistically significant difference means that the model in the line predicts better than simply using $y_t$.

When $k = 5$, the results were mixed. There was less clear discrimination between methods, with more evidence of some value to combining forecasts. If we had to pick a single model from Table 8, it would be the HAR model.

The graphical representation of the out-of-sample forecasts in the two samples was provided in Figures 8 and 9.

In Figure 8, the ARFIMA models both underestimated the RV during the sub-prime crises in sub-figure (a), while the five-step ahead forecasts were in line with the trend of RV. LSTAR models were more precise than ARFIMA when $k = 1$, especially during the financial crisis. This was not true for the five-step-ahead forecasts that were inaccurate both with and without exogenous variables. HAR model seemed to slightly underestimate RV both for $k = 1$ and $k = 5$. The graphical representations of the one-step-ahead forecasts from FNN, ENN, and JNN almost replicated the observed RV, both in the model with and without determinants. Furthermore, FNNs seemed to be precise also in five-step-ahead forecasts, while just JNNX's representation was in line with the observed RV. LSTM one-step-ahead forecasts seemed to be smoother than the observed RV, while the same forecasts from the NARX model (i.e., the red line in Figure 8, letter o) were almost overlapped in the period 1997–2010. In the five-step-ahead forecasts, FNN and JNN models provided precise forecasts, while ENNs were less accurate. Finally, LSTM multistep-ahead forecasts were in

**Table 7** Encompassing tests—entire sample ($k = 1$)

| Model | Panel A: time series models | | | | | Panel B: FFN, JNN, and ENN | | | | | | Panel C: long-term dependence detecting neural networks | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ARFIMAX | ARFIMA | LSTARX | LSTAR | HAR | FNNX | FNN | ENNX | ENN | JNNX | JNN | LSTMX | LSTM | NARX | NAR |
| Panel A: time series models | | | | | | | | | | | | | | | |
| ARFIMAX | – | 0.921*** | 0.911*** | 0.480** | –0.137 | 0.555*** | 0.396* | 0.453** | 0.378* | 0.600*** | 0.547*** | –0.786*** | –0.162 | –0.096** | 0.403*** |
| ARFIMA | 0.587*** | – | 0.988*** | –0.603* | 0.073 | 0.115 | –0.137 | 0.409** | 0.097 | 0.102 | 0.069 | –0.394*** | 0.317* | –0.131** | 0.484*** |
| LSTARX | 0.275** | 0.576*** | – | 0.333*** | 0.017 | 0.297*** | 0.270*** | 0.192 | 0.244** | 0.324*** | 0.304*** | –0.543*** | –0.332** | –0.079** | 0.241*** |
| LSTAR | 0.808*** | 1.638*** | 0.908*** | – | 0.154 | 0.801** | 0.372 | 0.604*** | 0.454 | 0.940*** | 0.805** | –0.641*** | 0.292 | –0.125*** | 0.426*** |
| HAR | 1.028*** | 0.984*** | 1.001*** | 0.901*** | – | 0.971*** | 0.845*** | 0.785*** | 0.743*** | 0.976*** | 0.959*** | –0.775*** | 0.138 | –0.111*** | 0.322*** |
| Panel B: FFN, JNN, and ENN | | | | | | | | | | | | | | | |
| FNNX | 0.471*** | 0.760*** | 0.592*** | 0.303 | 0.039 | – | 0.053 | 0.334** | 0.312** | 0.526* | 0.398* | –0.434*** | 0.108 | –0.097*** | 0.253*** |
| FNN | 0.556*** | 0.840*** | 0.603*** | 0.561** | 0.141 | 0.738*** | – | 0.419*** | 0.371** | 0.759*** | 0.739*** | –0.455*** | 0.168 | –0.084*** | 0.254*** |
| ENNX | 0.510*** | 0.632*** | 0.632*** | 0.426*** | 0.199 | 0.472*** | 0.384*** | – | 0.352** | 0.502*** | 0.468*** | –0.400*** | 0.132 | –0.083*** | 0.270*** |
| ENN | 0.522*** | 0.691*** | 0.569*** | 0.476*** | 0.208 | 0.468*** | 0.402*** | 0.412*** | – | 0.495*** | 0.479*** | –0.385*** | 0.184 | –0.051* | 0.247*** |
| JNNX | 0.428*** | 0.739*** | 0.556*** | 0.205 | 0.035 | 0.275 | 0.030 | 0.291** | 0.271** | – | 0.293 | –0.389*** | 0.099 | –0.108*** | 0.241*** |
| JNN | 0.461*** | 0.754*** | 0.571*** | 0.291 | 0.048 | 0.404* | 0.051 | 0.328** | 0.286** | 0.495** | – | –0.401*** | 0.102 | –0.092*** | 0.255*** |
| Panel C: long-term dependence detecting neural networks | | | | | | | | | | | | | | | |
| LSTMX | 1.734*** | 1.369*** | 1.675*** | 1.633*** | 1.885*** | 1.654*** | 1.697*** | 1.648*** | 1.662*** | 1.618*** | 1.632*** | – | –1.973*** | 0.293*** | 0.899*** |
| LSTM | 1.219*** | 0.955*** | 1.431*** | 0.873*** | 0.951*** | 0.965*** | 0.886*** | 0.924*** | 0.845*** | 0.971*** | 0.967*** | –0.908*** | – | –0.070* | 0.399*** |
| NARX | 1.097*** | 1.089*** | 1.098*** | 1.111*** | 1.121*** | 1.124*** | 1.117*** | 1.117*** | 1.095*** | 1.134*** | 1.122*** | 0.826*** | 1.094*** | – | 0.918*** |
| NAR | 1.034*** | 1.095*** | 1.086*** | 1.019*** | 1.010*** | 1.035*** | 1.022*** | 1.010*** | 1.009*** | 1.039*** | 1.028*** | 0.424*** | 0.970*** | 0.210*** | – |

*Notes*: The table presents the test statistics on the null hypothesis that the coefficient of the model in the row is equal to zero in the regression $y_{t+1} = \alpha_0 + \alpha_1 \hat{y}_{i,t+1} + \alpha_2 \hat{y}_{j,t+1} + u_{t+1}$, where $\hat{y}_{i,t+1}$ is the forecast from the $i$th model in the row and $\hat{y}_{j,t+1}$ is the forecast from the model in the $j$th column. The forecast evaluation period covers August 1997–December 2017 ($N = 245$). *, **, and *** indicate the significativity at 1%, 5%, and 10% level, respectively, respectively.

**Table 8** Encompassing tests—entire sample ($k = 5$)

| Model | Panel A: time series models | | | | | Panel B: FFN, JNN, and ENN | | | | | | Panel C: long-term dependence detecting neural networks | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ARFIMAX | ARFIMA | LSTARX | LSTAR | HAR | FNNX | FNN | ENNX | ENN | JNNX | JNN | LSTMX | LSTM | NARX | NAR |
| Panel A: time series models | | | | | | | | | | | | | | | |
| ARFIMAX | – | 1.036*** | 0.917*** | 0.649*** | –0.405*** | 0.453*** | 0.435*** | 0.302** | 0.324** | 0.271** | 0.232** | 0.325*** | 0.430*** | 0.361*** | 0.377*** |
| ARFIMA | –0.086 | – | 0.604*** | 0.400*** | –0.385*** | 0.248*** | 0.238*** | 0.028 | –0.012 | 0.164** | 0.132* | 0.201*** | 0.242*** | 0.201** | 0.167* |
| LSTARX | 0.069 | 0.100 | – | 0.142** | 0.009 | 0.132* | 0.118* | 0.109 | 0.117* | 0.122** | 0.131** | 0.048 | 0.116* | 0.148** | 0.115* |
| LSTAR | 1.746*** | 1.944*** | 2.286*** | – | 0.532*** | 1.371*** | 1.331*** | 1.316*** | 1.494*** | 1.139*** | 1.028*** | 1.351*** | 1.313*** | 1.143*** | 1.279*** |
| HAR | 1.654*** | 1.678*** | 1.371*** | 1.256*** | – | 1.584*** | 1.528*** | 0.152*** | 1.688*** | 1.640*** | 1.449*** | 1.115*** | 1.537*** | 1.546*** | 1.605*** |
| Panel B: FFN, JNN, and ENN | | | | | | | | | | | | | | | |
| FNNX | 0.524*** | 0.587*** | 0.662*** | 0.504*** | –0.168** | – | 0.071 | 0.307*** | 0.438*** | 0.255*** | 0.263*** | 0.337*** | 0.015 | 0.332*** | 0.399*** |
| FNN | 0.529*** | 0.586*** | 0.653*** | 0.508*** | –0.118 | 0.603*** | – | 0.358*** | 0.450*** | 0.285*** | 0.276*** | 0.351*** | 0.354 | 0.357*** | 0.413*** |
| ENNX | 0.613*** | 0.728*** | 0.726*** | 0.564*** | –0.124 | 0.465*** | 0.401*** | – | 0.497*** | 0.345*** | 0.309*** | 0.413*** | 0.403*** | 0.390*** | 0.443*** |
| ENN | 0.596*** | 0.766*** | 0.739*** | 0.558*** | –0.289*** | 0.438*** | 0.414*** | 0.378*** | – | 0.121 | 0.084 | 0.326*** | 0.413*** | 0.263*** | 0.346*** |
| JNNX | 0.705*** | 0.745*** | 0.793*** | 0.657*** | –0.204** | 0.615*** | 0.584*** | 0.577*** | 0.720*** | – | 0.287** | 0.440*** | 0.587*** | 0.687*** | 0.681*** |
| JNN | 0.736*** | 0.772*** | 0.810*** | 0.683*** | –0.057 | 0.638*** | 0.616*** | 0.615*** | 0.764*** | 0.560*** | – | 0.476*** | 0.620*** | 0.708*** | 0.754*** |
| Panel C: long-term dependence detecting neural networks | | | | | | | | | | | | | | | |
| LSTMX | 0.832*** | 0.880*** | 0.952*** | 0.797*** | 0.264*** | 0.727*** | 0.707*** | 0.710*** | 0.753*** | 0.592*** | 0.549*** | – | 0.708*** | 0.656*** | 0.721*** |
| LSTM | 0.498*** | 0.551*** | 0.615*** | 0.477*** | –0.117* | 0.617*** | 0.304 | 0.334*** | 0.423*** | 0.263*** | 0.252*** | 0.333*** | – | 0.331*** | 0.386*** |
| NARX | 0.695*** | 0.754*** | 0.821*** | 0.660*** | –0.150 | 0.581*** | 0.554*** | 0.552*** | 0.639*** | 0.141 | 0.142 | 0.409*** | 0.554*** | – | 0.605*** |
| NAR | 0.769*** | 0.866*** | 0.940*** | 0.726*** | –0.251** | 0.616*** | 0.589*** | 0.566*** | 0.645*** | 0.197 | 0.109 | 0.391*** | 0.587*** | 0.326** | – |

*Notes*: The table presents the coefficient of the first model in the regression $y_{t+5} = \alpha_0 + \alpha_1 \hat{y}_{i,t+5} + \alpha_2 \hat{y}_{j,t+5} + u_t$, where $\hat{y}_{i,t+5}$ is the multistep ahead forecast from the $i$th model in the row and $\hat{y}_{j,t+5}$ is the forecast from the model in the $j$th column. The forecast evaluation period covers August 1997–December 2017 ($N = 245$). *, **, and *** indicate the significativity at 1%, 5%, and 10% level, respectively.

line with the forecasts from other neural networks, while NARX models forecasts, despite a little underestimation during volatility peaks, almost replicated the real trend of RV.

In Figure 9, the representations of the out-of-sample forecasts in the subsample were heterogeneous. One-step-ahead forecasts from ARFIMA, LSTAR, and HAR models appeared to be smoother than the observed RV in the subsample, while the five-step-ahead forecasts from LSTAR models were highly inaccurate. When $k = 1$, the forecasts from FNN, ENN, JNN, and NARX seemed to almost overlap with the observed RV. Excluding FNNs and JNNX, the neural networks seemed instead to be less precise when $k = 5$, especially in the volatility peak in October 2008.

The analysis here presented has a two-fold implication. On the one side, the neural networks outperformed the compared classical methods in both the sample analyzed. In particular, the forecasts from long-term detecting networks proved to be the most accurate among the compared methods. This is not surprising given the well-known long memory of the realized volatility and the need for a model which accounts for that feature. On the other side, the use of macroeconomic and financial variables to make predictions increased the forecasting accuracy, even in the more complex models, although a causal effect cannot be assessed with neural networks.

## 5 Conclusions

In this article, a flexible nonlinear tool for forecasting volatility has been applied. The purpose of the article was to understand whether ANNs were able to capture linear and nonlinear relations and provide more accurate forecasts than traditional econometric methods. The target variable to be forecast was the logarithm of realized volatility, while the models included also macroeconomic and financial variables as determinants.

The most attractive feature of ANNs is that, by modifying the structure of the network, any linear and nonlinear function can be approximated. Moreover, in comparison with traditionally employed nonlinear time series model, such as smooth transition autoregressive model and threshold autoregressive model, they do not necessitate the knowledge of the number of regimes to be trained and require a minor computational effort in the estimation of the parameters.

Out-of-sample comparisons indicated that neural networks provide significant benefits in predicting relations expected to be nonlinear, such as between realized volatility and its determinants.

In a comparison of feed-forward and RNNs with traditional econometric methods, the best performing models appeared to be LSTM and NARX neural networks. The results further showed that these long-term dependence detecting models consistently outperformed competing for neural networks, like FNN, ENN, and JNN. The superior forecasting ability of LSTM and NARX was also assessed in a period where the stock market volatility was particularly high, like the recent financial crisis.

Since a researcher is often interested in producing forecasts for a horizon greater than one, multistep-ahead recursive forecasts were further compared. Among the main results, it emerged that long-term memory detecting neural networks had good performance when a large sample is analyzed, and provided comparable performance with other methods when a smaller and more volatile sample was evaluated.

Interestingly, in this article, realized volatility was predicted accurately well in a nonlinear framework. Several papers (McAleer and Medeiros, 2008; Hillebrand and Medeiros, 2010) showed that nonlinear models were not able to improve forecasting accuracy of realized volatility when compared with linear models. There could be several reasons that explain the findings of this manuscript. First, the analysis of out-of-sample forecasts proved that the models significantly able to outperform the linear models were the long-term dependence detecting models, that is, LSTM and NARX. So far, this seems to be the first attempt to implement such approaches to predict realized volatility. Moreover, the results highlighted the usefulness of the macroeconomic and financial variables in forecasting the realized volatility of S&P 500 index. Although there are many papers on forecasting realized volatility through a linear model, the literature on analyzing the role of volatility determinants in a nonlinear framework is still scarce.

Although appealing, there are still some issues concerning neural networks. The number of parameters to be trained can be extremely high even with a limited number of input variables. This is a stark contrast to the number of parameters of an ARFIMA or an HAR model. However, the number of trained weights does not differ excessively from the number of parameters in a smooth transition autoregressive model with multiple regimes. Furthermore, the network models do not lend themselves to the easy interpretation of explanatory variables due to the structure of the layers. On this purpose, the author acknowledges that this article was mainly focused on providing superior forecasting accuracy rather than interpreting causal relationships.

In future works, the performance of RNNs should be tested with different architectures, for example, by modifying the activation function or enlarging the number of hidden layers.

Moreover, in this article, I have exclusively focused on univariate time series while, in practice, multivariate forecasting problems require to forecast a set of possibly dependent time series. An important future direction is to extend the strategies developed in this article to the multivariate setting.

## References

Anders, U., and O. Korn. 1996. "Model Selection in Neural Networks." ZEW Discussion Paper No. 96–21.

Andersen, T. G., T. Bollerslev, and F. X. Diebold. 2007. "Roughing It Up: Including Jump Components in the Measurement, Modeling and Forecasting of Return Volatility." CREATES Research Paper No. 2007–18.

Andersen, T. G., T. Bollerslev, F. X. Diebold, and H. Ebens. 2001. The Distribution of Realized Stock Return Volatility. *Journal of Financial Economics* 61: 43–76.

Arnerić, J., T. Poklepovic, and Z. Aljinović. 2014. GARCH Based Artificial Neural Networks in Forecasting Conditional Variance of Stock Returns. *Croatian Operational Research Review* 5: 329–343.

Bai, J., and P. Perron. 2003. Computation and Analysis of Multiple Structural Change Models. *Journal of Applied Econometrics* 18: 1–22.

Bao, W., H. Yue, and Y. Rao. 2017. A Deep Learning Framework for Financial Time Series Using Stacked Autoencoders and Long-Short Term Memory. *PLoS One* 12: e0180944.

Barndorff-Nielsen, O. E., and N. Shephard. 2002. Econometric Analysis of Realised Volatility and Its Use in Estimating Stochastic Volatility Models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 64: 253–280.

Bianchi, F. M., M. C. Kampffmeyer, A. Rizzi, and R. Jenssen. 2017. "An Overview and Comparative Analysis of Recurrent Neural Networks for Short Term Load Forecasting." *CoRR*, abs/1705.04378.

Black, F. 1976. Noise. *Journal of Finance* 41: 529–543.

Bollerslev, T. 1986. Generalized Autoregressive Conditional Heteroskedasticity. *Journal of Econometrics* 31: 307–327.

Bucci, A., G. Palomba, and E. Rossi. 2019. "Does macroeconomics help in predicting stock markets volatility comovements? A nonlinear approach." Working Paper No. 440, Dipartimento di Scienze Economiche e Sociali, Università Politecnica delle Marche.

Chong, Y. Y., and D. F. Hendry. 1986. Econometric Evaluation of Linear Macro-Economic Models. *The Review of Economic Studies* 53: 671–690.

Choudhry, T., F. I. Papadimitriou, and S. Shabi. 2016. Stock Market Volatility and Business Cycle: Evidence from Linear and Nonlinear Causality Tests. *Journal of Banking & Finance* 66: 89–101.

Christiansen, C., M. Schmeling, and A. Schrimpf. 2012. A Comprehensive Look at Financial Volatility Prediction by Economic Variables. *Journal of Applied Econometrics* 27: 956–977.

Clements, M. P., and H.-M. Krolzig. 1998. A Comparison of the Forecast Performance of Markov-Switching and Threshold Autoregressive Models of US GNP. *The Econometrics Journal* 1: 47–75.

Corsi, F. 2009. A Simple Approximate Long-Memory Model of Realized Volatility. *Journal of Financial Econometrics* 7: 174–196.

De Pooter, M., M. Martens, and D. Van Dijk. 2008. Predicting the Daily Covariance Matrix for S&P 100 Stocks Using Intraday Data - But Which Frequency to Use?. *Econometric Reviews* 27: 199–229.

Di Persio, L., and O. Honchar. 2017. Recurrent Neural Networks Approach to the Financial Forecast of Google Assets. *International Journal of Mathematics and Computers in Simulation* 11: 7–13.

Diebold, F. X., and R. S. Mariano. 1995. Comparing Predictive Accuracy. *Journal of Business & Economic Statistics* 13: 253–263.

Diebold, F. X., and K. Yilmaz. 2009. Measuring Financial Asset Return and Volatility Spillovers, with Application to Global Equity Markets. *The Economic Journal* 119: 158–171.

Donaldson, G. R., and M. Kamstra. 1996a. A New Dividend Forecasting Procedure That Rejects Bubbles in Asset Prices. *Review of Financial Studies* 8: 333–383.

Donaldson, G. R., and M. Kamstra. 1996b. Forecast Combining with Neural Networks. *Journal of Forecasting* 15: 49–61.

Donaldson, G. R., and M. Kamstra. 1997. An Artificial Neural network-GARCH Model for International Stock Return Volatility. *Journal of Empirical Finance* 4: 17–46.

Elman, J. L. 1990. Finding Structure in Time. *Cognitive Science* 14: 179–211.

Engle, R. F. 1982. Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica* 50: 987–1007.

Engle, R. F., E. Ghysels, and B. Sohn. 2009. "On the Economic Sources of Stock Market Volatility." NYU Working Paper No. FIN-08-043

Fair, R., and R. J. Shiller. 1989. The Informational Context of Ex Ante Forecasts. *The Review of Economics and Statistics* 71: 325–331.

Fama, E., and K. French. 1993. Common Risk Factors in the Returns on Stocks and Bonds. *Journal of Financial Economics* 33: 3–56.

Fernandes, M., M. C. Medeiros, and M. Scharth. 2014. Modeling and Predicting the CBOE Market Volatility Index. *Journal of Banking & Finance* 40: 1–10.

Gnana Sheela, K., and S. Deepa. 2013. Review on Methods to Fix Number of Hidden Neurons in Neural Networks. *Mathematical Problems in Engineering* 2013: 11.

Guzman, S. M., J. O. Paz, and M. L. M. Tagert. 2017. The Use of NARX Neural Networks to Forecast Daily Groundwater Levels. *Water Resources Management* 31: 1591–1603.

Hajizadeh, E., A. Seifi, M. Fazel Zarandi, and I. Turksen. 2012. A Hybrid Modeling Approach for Forecasting the Volatility of S&P 500 Index Return. *Expert Systems with Applications* 39: 431–436.

Hamid, S. A., and Z. Iqbal. 2004. Using Neural Networks for Forecasting Volatility of S&P 500 Index Futures Prices. *Journal of Business Research* 57: 1116–1125.

Hansen, P. R., A. Lunde, and J. M. Nason. 2011. The Model Confidence Set. *Econometrica* 79: 435–497.

Heaton, J., N. Polson, and J. Witte. 2016. "Deep Learning in Finance." *CoRR*, abs/1602.06561.

Hillebrand, E., and M. C. Medeiros. 2010. The Benefits of Bagging for Forecast Models of Realized Volatility. *Econometric Reviews* 29: 571–593.

Hochreiter, S., and J. Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9: 1735–1780.

Hu, Y. M., and C. Tsoukalas. 1999. "Combining Conditional Volatility Forecasts Using Neural Networks: An Application to the EMS Exchange Rates. *Journal of International Financial Markets, Institutions & Money* 9: 407–422.

Jordan, M. I. 1986. "Serial Order: A Parallel Distributed Processing Approach." Discussion Paper, Institute for Cognitive Science Report 8604, University of California San Diego.

Kamijo, K., and T. Tanigawa. 1990. "Stock Price Pattern Recognition - A Recurrent Neural Network Approach." *1990 IJCNN International Joint Conference on Neural Networks*. San Diego, CA, USA: IEEE, pp. 215–221.

Keenan, D. M. 1985. A Tukey Nonadditivity-Type Test for Time Series Nonlinearity. *Biometrika* 72: 39–44.

Khan, A. I. 2011. Financial Volatility Forecasting by Nonlinear Support Vector Machine Heterogeneous Autoregressive Model: Evidence from Nikkei 225 Stock Index. *International Journal of Economics and Finance* 3: 138–150.

Kim, H. Y., and C. H. Won. 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.

Kristjanpoller, W., A. Fadic, and M. C. Minutolo. 2014. Volatility Forecast Using Hybrid Neural Network Models. *Expert Systems with Applications* 41: 2437–2442.

Lin, T., B. G. Horne, P. Tino, and C. L. Giles. 1996. Learning Long-Term Dependencies in NARX Recurrent Neural Networks. *IEEE Transactions on Neural Networks* 7: 1329–1338.

Maciel, L., F. Gomide, and R. Ballini. 2016. Evolving Fuzzy-GARCH Approach for Financial Volatility Modeling and Forecasting. *Computational Economics* 48: 379–398.

Maheu, J. M., and T. H. McCurdy. 2002. Nonlinear Features of Realized Volatility. *Review of Economics and Statistics* 84: 668–681.

McAleer, M., and M. Medeiros. 2008. A Multiple Regime Smooth Transition Heterogeneous Autoregressive Model for Long Memory and Asymmetries. *Journal of Econometrics* 147: 104–119.

Mele, A. 2007. Asymmetric Stock Market Volatility and the Cyclical Behavior of Expected Returns. *Journal of Financial Economics* 86: 446–478.

Mele, A. 2008. "Understanding Stock Market Volatility - A Business Cycle Perspective." Working Paper.

Menezes, J., and G. Barreto. 2006. "A New Look at Nonlinear Time Series Prediction with NARX Recurrent Neural Network." *2006 Ninth Brazilian Symposium on Neural Networks (SBRN'06)*. Ribeirao Preto, Brazil: IEEE, pp. 160–165.

Miura, R., L. Pichl, and T. Kaizoji. 2019. "Artificial Neural Networks for Realized Volatility Prediction in Cryptocurrency Time Series," In H. Lu, H. Tang, and Z. Wang *(eds.), Advances in Neural Networks – ISNN 2019*. Cham: Springer International Publishing, pp. 165–172.

Nelson, D. B. 1990. Stationarity and Persistence in the GARCH (1,1) Model. *Econometric Theory* 6: 318–334.

Panchal, G., A. Ganatra, Y. Kosta, and D. Panchal. 2010. Searching Most Efficient Neural Network Architecture Using Akaikes Information Criterion (AIC). *Inernational Journal of Computer Applications* 5: 41–44.

Patton, A. J. 2011. Volatility Forecast Comparison Using Imperfect Volatility Proxies. *Journal of Econometrics* 160: 246–256.

Pavlidis, E. G., I. Paya, and D. A. Peel. 2012. Forecast Evaluation of Nonlinear Models: The Case of Long-Span Real Exchange Rates. *Journal of Forecasting* 31: 580–595.

Paye, B. S. 2012. Dèjà Vol: Predictive Regressions for Aggregate Stock Market Volatility Using Macroeconomic Variables. *Journal of Financial Economics* 106: 527–546.

Pichl, L., and T. Kaizoji. 2017. Volatility Analysis of Bitcoin Price Time Series. *Quantitative Finance and Economics* 1: 474–485.

Rosa R., L. Maciel, F. Gomide, and R. Ballini. 2014. "Evolving Hybrid Neural Fuzzy Network for Realized Volatility Forecasting with Jumps." *2014 IEEE Conference on Computational Intelligence for Financial Engineering Economics (CIFEr)*. London: IEEE, pp. 481–488.

Rossi, E., and P. Santucci de Magistris. 2014. Estimation of Long Memory in Integrated Variance. *Econometric Reviews* 33: 785–814.

Schittenkopf, C., G. Dorffner, and E. J. Dockner. 2000. Forecasting Time-Dependent Conditional Densities: A Semi Non-parametric Neural Network Approach. *Journal of Forecasting* 19: 355–374.

Schwert, W. G. 1989. Why Does Stock Market Volatility Change over Time. *The Journal of Finance* 44: 1115–1153.

Stinchcombe, M., and H. White. 1992. "Using Feedforward Networks to Distinguish Multivariate Populations." *Proceedings of the International Joint Conference on Neural Networks*. Baltimore, MD, USA: IEEE, pp. 788–793.

Tang, Z., and P. A. Fishwick. 1993. Feed-Forward Neural Nets as Models for Time Series Forecasting. *ORSA Journal on Computing* 5: 374–385.

Terasvirta, T. 1994. Specification, Estimation, and Evaluation of Smooth Transition Autoregressive Models. *Journal of the American Statistical Association* 89: 208–218.

Tibshirani, R. 1996. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)* 58: 267–288.

Tino, P., C. Schittenkopf, and G. Dorffner. 2001. Financial Volatility Trading Using Recurrent Neural Networks. *IEEE Transactions on Neural Networks* 12: 865–874.

Vortelinos, D. I. 2017. Forecasting Realized Volatility: HAR against Principal Components Combining, Neural Networks and GARCH. *Research in International Business and Finance* 39: 824–839.

Welch, I., and A. Goyal. 2008. A Comprehensive Look at the Empirical Performance of Equity Premium Prediction. *Review of Financial Studies* 21: 1455–1508.

White, H. 1988. "Economic Prediction Using Neural Networks: The Case of IBM Daily Stock Returns." *IEEE 1988 International Conference on Neural Networks*. San Diego, CA, USA: IEEE, pp. 451–458.

Xiong, R., E. P. Nichols, and Y. Shen. 2016. "Deep Learning Stock Volatility with Google Domestic Trends." *CoRR*, abs/1512.04916.

Zou, H. 2006. The Adaptive Lasso and Its Oracle Properties. *Journal of the American Statistical Association* 101: 1418–1429.