Realized Volatility Forecasting with Neural Networks

Bucci, Andrea

August 2019

Realized Volatility Forecasting with Neural Networks

Andrea Bucci[*]

Abstract

In the last few decades, a broad strand of literature in finance has implemented artificial neural networks as 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 paper, I compare the predictive performance of feed-forward and recurrent neural networks (RNN), particularly focusing on the recently developed Long short-term memory (LSTM) network and NARX network, with traditional econometric approaches. The results show that recurrent neural networks are able to outperform all the traditional econometric methods. Additionally, capturing long-range dependence through Long short-term memory and NARX models seems to improve the forecasting accuracy also in a highly volatile framework.

1 Introduction

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 et al. (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 et al. (2012).

[*]Department of Economics and Social Sciences, Università Politecnica delle Marche, 60121 Ancona, Italy. email: a.bucci@univpm.it

In this framework, this paper 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 (ANN). 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).

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 et al. (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 networks architecture, showing that the ANNs predict accurately well the targeted variable during crisis periods. Recently, Arnerić et al. (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 a NN model provides superior forecasting accuracy in comparison with other linear and nonlinear models.

Fernandes et al. (2014) extended these studies by specifying a neural-network heterogeneous autoregression (HAR) with exogenous variables to improve implied volatility forecasts. Finally, 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.

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 Jordan neural network (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 paper is organized as follows: Section 2 illustrates the data set, the estimation method of the volatility and the set of macroeconomic and financial predictors. Section 3 introduces the neural network models. The choice of the architecture of the neural networks is presented in Section 4. In Section 5, the performance of the ANNs is assessed in terms of forecasting accuracy, while Section 6 concludes.

2 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:

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

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 in Figure 2, suggesting that a long-memory detecting model should be implemented (see Rossi and Santucci de Magistris (2014)). Since volatility exhibits a highly variable behaviour, 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. Afterwards, both Engle et al. (2009) and Diebold and Yilmaz (2009) showed a strong link between macroeconomic fundamentals and stock return volatility. Recently, Paye (2012) and Christiansen et al. (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.

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 et al. (2012), I include in the analysis many predictive variables from return predictability literature Mele (2007, 2008).

Firstly, 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 return (MKT). As a measure of risk factors, the Fama and French (1993) factors (HML and 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 T-bill rate (T-B), the rate of return on long-term government bond and the term spread difference (TS) of long-term bond yield and 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 et al. (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.

Figure 1: log RV from February 1950 through December 2017

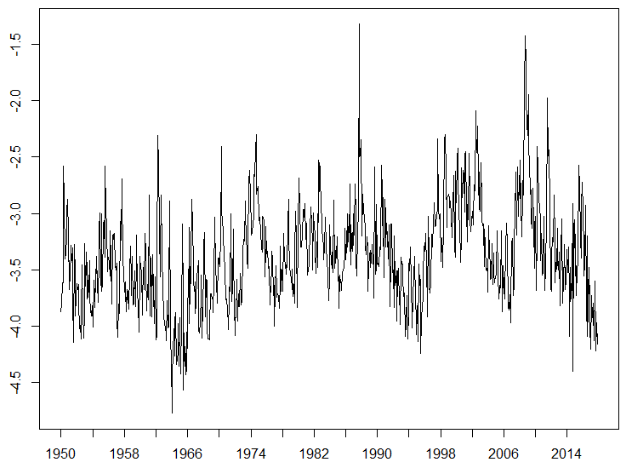


Figure 2: Autocorrelation function (ACF) and partial autocorrelation function (PACF) of RV.

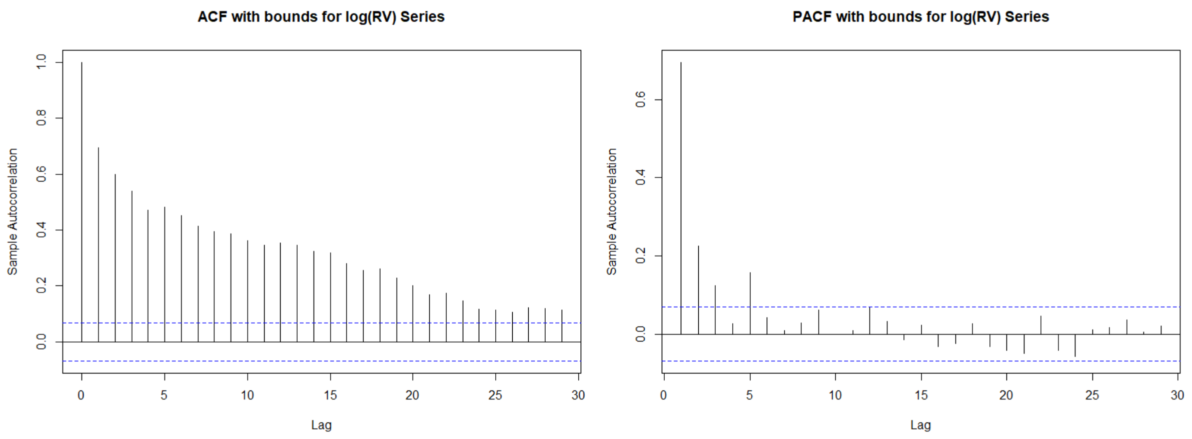


Table 1: Variables description

Symbol	Variable	Data source	
		Description	Source
DP	Dividend Yield Ratio S&P	Dividends over the past year relative to current market prices; S&P500 index	Robert Shiller's website
EP	Earning Price Ratio S&P 500	Earnings over the past year relative to current market prices; S&P500 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

3 Neural Networks

Artificial Neural Networks (ANNs) can be seen as non-parametric tools, inspired by the structure of the human brain, for modelling and predicting the unknown function generating the observed data (Arnerić et al. (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 behaviours, 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*[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 (see Gnana Sheela and Deepa (2013)), there is no commonly agreed solution to this issue. Donaldson and Kamstra (1996b) proved that a single hidden neural network is a *universal approximator*, meaning that the network can approximate a wide range of linear and non-linear functions, 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:

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

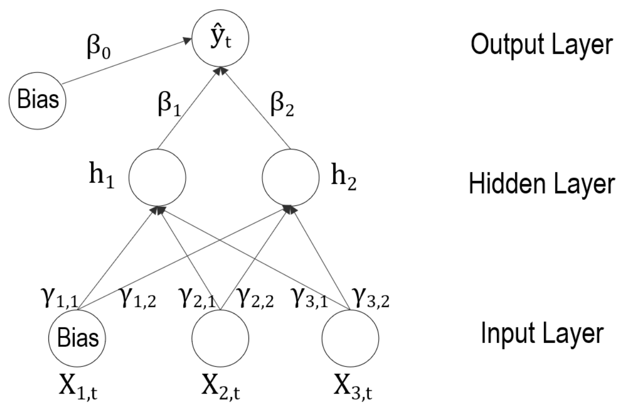
where F is the output activation function, G is the hidden units activation function, β_j, with

[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 backpropagation algorithm.

[2]Other loss functions can be also implemented, such as Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE).

$j = 1,\dots,q$, are the weights from hidden unit j to the output unit, $x_t = \left\{1, x_{1,t},\dots,x_{s,t}\right\}$ is the $1 \times m$ vector of input variables at time t (with $m = s+1$), β_0 is the bias of the final output, $\gamma_j = \left\{\gamma_{1,j},\dots,\gamma_{m,j}\right\}$ 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 $\theta = \left\{\beta_0,\dots,\beta_q,\gamma'_1,\dots,\gamma'_q\right\}$ is the vector of all network weights. This version, with 3 input variables including the bias, and 2 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 *feed-forward neural network* (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, i.e. $F(a) = a$. In this case, Equation (2) can be written as follows

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

A common choice for G is the logistic function, i.e. $G(a) = \frac{1}{1+e^{-a}}$, 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* 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 (see, for example, Schittenkopf et al. (2000); Tino et al. (2001)). This paper focuses on four recurrent architectures: Elman and Jordan recurrent networks, long short-term memory (LSTM) networks and NARX neural networks.

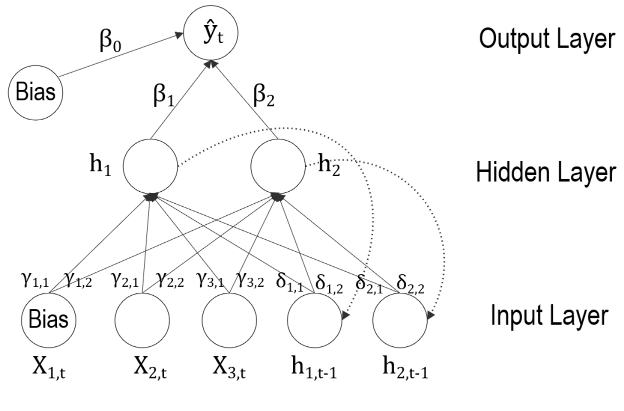
In the Elman neural network (ENN), proposed by Elman (1990), the input layer has additional neurons which are fed back from the hidden layer (see Figure 4). The output of the ENN, with an identity function as output activation function, can be represented as

$$f_t(x_t,\theta) = \beta_0 + \sum_{j=1}^{q} h_{tj}\beta_j \tag{4}$$

$$h_{tj} = G\left(x_t\gamma_j' + h_{t-1}\delta_j'\right) \qquad j = 1,\ldots,q$$

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

Figure 4: ENN with a single hidden layer

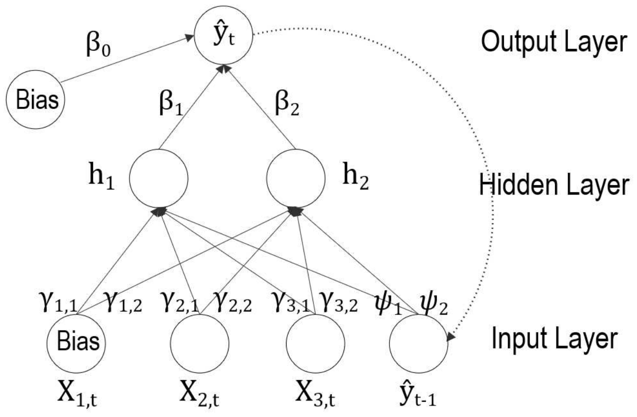

Output Layer
Hidden Layer
Input Layer
$\hat{y}_t$
β_0
β_1
β_2
Bias
h_1
h_2
$\gamma_{1,1}$ $\gamma_{1,2}$ $\gamma_{2,1}$ $\gamma_{2,2}$ $\gamma_{3,1}$ $\gamma_{3,2}$ $\delta_{1,1}$ $\delta_{1,2}$ $\delta_{2,1}$ $\delta_{2,2}$
Bias
$X_{1,t}$ $X_{2,t}$ $X_{3,t}$ $h_{1,t-1}$ $h_{2,t-1}$

Jordan (1986), instead, introduced a recurrent neural network 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 Jordan neural network (JNN) can be specified as follows

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

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

Figure 5: JNN with a single hidden layer



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 AR representation of an ARMA model. Furthermore, differently from FNNs, recurrent neural networks 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, i.e. [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.

Long short-term memory (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 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 $t-1$. 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, i.e. 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 one 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:

$$f_t = \sigma\big(W_f h_{t-1} + U_f x_t + b_f\big) \tag{6}$$

$$i_t = \sigma\big(W_i h_{t-1} + U_i x_t + b_i\big) \tag{7}$$

$$\tilde{c}_t = \tanh\big(W_c h_{t-1} + U_c x_t + b_c\big) \tag{8}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{9}$$

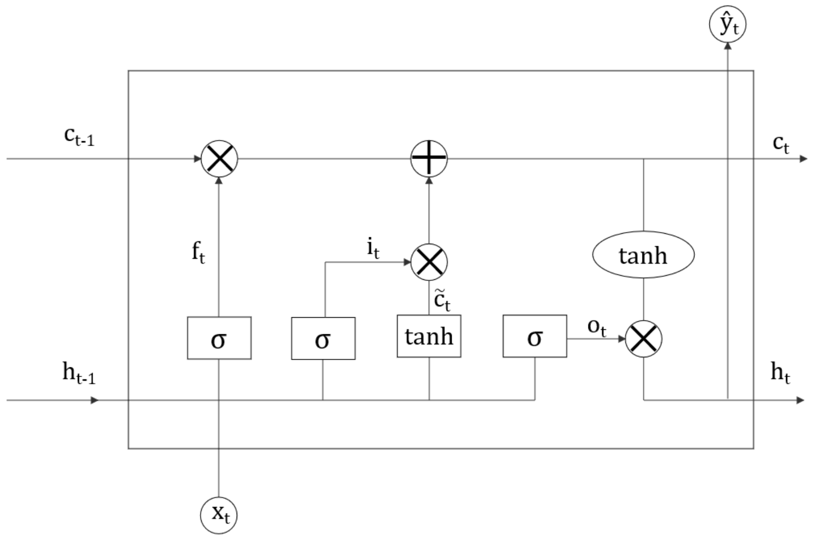
$$o_t = \sigma\big(W_o h_{t-1} + U_o x_t + V_o c_t + b_o\big) \tag{10}$$

$$h_t = o_t \odot \tanh\big(c_t\big) \tag{11}$$

$$\hat{y}_t = h_t \tag{12}$$

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 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, σ is a sigmoid or logistic function and ⊙ is the Hadamard product function.

Figure 6: Basic LSTM memory cell



Note: 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 ⊕, and point-wise multiplication, ⊗. Non-linear interactions are logistic functions, σ.

LSTM can operate where long memory effects are present in the underlying structure of the times series, similarly to HAR or ARFIMA models. Accordingly, there are numerous applications of LSTM models in finance, see, for example, Heaton et al. (2016), Bao et al. (2017), Pichl and Kaizoji (2017), Kim and Won (2018), Di Persio and Honchar (2017) and Xiong et al. (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 twofold way. The first mode is called *parallel (P) architecture*, in which the output is fed back to the input of the feed-forward neural network. The NARX-P architecture behaves like a Jordan neural network 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 Feedforward Neural Network (FNN) with d additional inputs equal to d delays of the real target variable.

In this paper 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

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

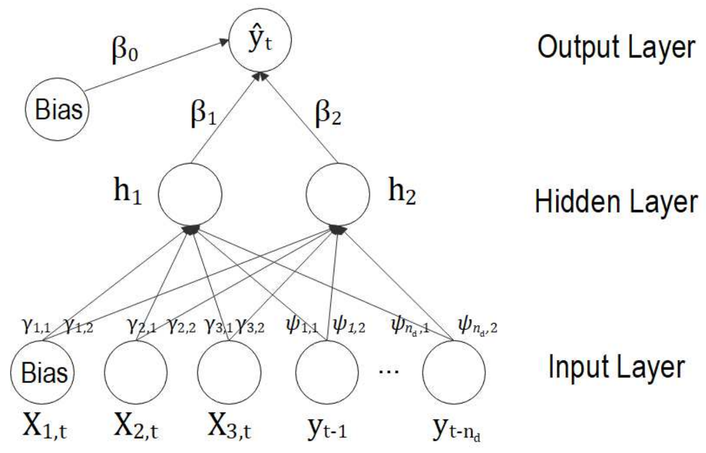
where x_t and y_t are respectively the input and the output of the network at time t, d is the output order and Ψ is a multilayer perceptron as in Figure 7. This architecture can be represented by the following equation

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

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.

Figure 7: Architecture of a NARX network



4 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); 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.

Firstly, 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 2), 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, λ; 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. The final set of variables selected was equal to $X_a = \{RV_{t-1}, RV_{t-2}, RV_{t-3}, DP_{t-1}, MKT_{t-1}, STR_{t-1}, DEF_{t-1}\}$ for the entire sample, and to $X_b = \{RV_{t-1}, RV_{t-2}, RV_{t-3}, MKT_{t-1}, STR_{t-1}\}$ for the subsample. The lack of significance of pure macroeconomic variables, i.e. inflation rate and industrial production growth, is in line with the findings of Schwert (1989) and Christiansen et al. (2012), once again underlying the relevance of premium risk's determinants.

Choosing the set of explanatory variables entails a twofold risk. On the one side, the so-called *look ahead bias*[4] may occur. On the other side, the variables selected through this method, i.e. 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.

[3]This subsample was used to validate the approach in a more volatile framework. 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.

[4]Look ahead bias involves using information not available during the period analysed.

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 paper, 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 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*) backpropagation 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 a NARX architecture is implemented, see Guzman et al. (2017).

[5]LSTM hidden units follow a different setting in comparison with other neural networks, thus the number of hidden units is set to 50, comparably to similar studies.

Table 2: MSE for increasing number of hidden nodes - Entire sample

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.

Model	N. Hidden	Performance	N. weights	Model	N. Hidden	Performance	N. weights
	1*	**0.1029**	10		1	0.1168	6
	2	0.1245	19		2	0.1260	11
	3	0.1062	28		3	0.1177	16
FNNX	4	0.1074	37	FNN	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
	1	0.1027	11		1	0.1177	7
	2	0.1111	23		2	0.1479	15
	3	0.1031	37		3	0.1179	25
ENNX	4*	**0.1006**	53	ENN	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
	1	0.1006	11		1	0.1165	7
	2	0.1236	21		2	0.1315	13
	3*	**0.1004**	31		3	0.1148	19
JNNX	4	0.1044	41	JNN	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
	1	0.0990	10		1	0.1141	6
	2	0.0981	19		2	0.1123	11
	3	0.0978	28		3	0.1160	16
NARX	4	0.0968	37	NAR	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

* denotes the selected number of hidden nodes

Table 3: MSE for increasing number of hidden nodes - Subsample

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 variance.

Model	N. Hidden	Performance	N. weights	Model	N. Hidden	Performance	N. 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

* denotes the selected number of hidden nodes

5 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 (1) 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 2 regimes) was identified, while lagged realized volatility was used as transition variable.

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 presence of greater persistence.

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, multi-step-ahead forecasts have been considered. The 5-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 5-step ahead is the following:

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

where $z_t = \{DP_{t-1}, MKT_{t-1}, STR_{t-1}, DEF_{t-1}\}$ in the entire sample, and $z_t = \{MKT_{t-1}, STR_{t-1}\}$ in the subsample. The set of input variables used in models ARFIMA, LSTAR, FNN, ENN, JNN, LSTM, and NAR did not include z_t.

The relative performance of the out-of-sample forecasting accuracy was assessed using mean squared error (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 et al. (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 α, a model is discarded when the null hypothesis of equal forecasting ability is rejected. The set of superior models (SSM) is then defined as the set of models not-rejecting the null hypothesis.

As shown by the average of the loss functions in Table 4, all the neural networks were able to outperform the traditional long-memory detecting models in the larger sample, when analysing forecasts for $k = 1$. In most cases, the exclusion of the explanatory variables worsened the forecasting accuracy, confirming that the dynamics of realized volatility are somehow linked to macroeconomic and financial conditions. The best performance in terms of forecasting accuracy measures was exhibited by LSTMX and NARX. These results held regardless of the loss function considered.

The analysis of multi-step-ahead forecasts, in the entire sample, further highlighted the predictive ability of long-term memory detecting recurrent neural networks, which outperformed all the competing models in terms of robust accuracy measures.

In the more volatile framework, classical long-memory detecting model seemed to not forecast accurately well realized volatility. Instead, the superiority of LSTM and NARX models was enhanced. 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 (see Heaton et al. (2016); Pichl and Kaizoji (2017)). Furthermore, the prediction differences between neural networks and linear models may indicate a nonlinear behaviour of the log-realized variance during financially stressed periods; see also Choudhry et al. (2016).

Simultaneously analysing one-step-ahead forecasts via MCS, it may be noted that long-memory and nonlinear relationships in realized volatility were not well approximated by ARFIMA and Logistic Smooth Transition Autoregressive models. Not surprisingly, the SSM contained only the NARX model which provided the best overall performance in terms of accuracy. Similar findings can be deduced from the analysis of multi-step-ahead forecasts.

The scenario was analogous for the one-step-ahead forecasts on the subsample in Table 5. The SSM included all the competing models, only excluding the forecasts from LSTARX models and FNNX. The higher probability of being included in the SSM was exhibited by the NARX and LSTM networks, highlighting the usefulness of a dedicated method to train persistent time series also in an unstable context such as the recent financial crisis. In this volatile framework, the 5-step-ahead forecasts provided mixed results. FNN without exogenous variables exhibited the lowest average losses, which may imply that a simpler model should be implemented when few multi-step-ahead forecasts need to be produced.

Additionally, the test of equal predictive accuracy of Diebold-Mariano (DM) (Diebold and Mariano, 1995) was used as robustness check. The pairwise comparison test in Table 6 supported our previous findings by exhibiting positive and strongly significant rejections in favour of long-memory detecting models, when $k = 1$. Furthermore, neural networks model were able to significantly outperform the predictive accuracy of the benchmark method (i.e. $\hat{y}_{t+5} = y_t$), also for several steps ahead, enhancing once again the ability of NARX models to accurately predict realized volatility.

Finally, Figure 8 and Figure 9 provide a graphical representation of the out-of-sample forecasts, showing that neural networks forecasts accurately approximate observed realized variance in both the samples.

Table 4: MCS with $\alpha = 0.10$ and $10,000$ bootstraps (Entire sample: 1997.08 to 2017.12)

	$k = 1$				$k = 5$			
Model	MSE		QLIKE		MSE		QLIKE	
	Loss	P_{MCS}	*Loss*	P_{MCS}	*Loss*	P_{MCS}	*Loss*	P_{MCS}
ARFIMAX	0.167	0.000	3.317	0.000	0.185	0.455^{**}	3.323	0.268^{*}
ARFIMA	0.205	0.000	3.325	0.000	0.219	0.118^{*}	3.333	0.211^{**}
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	0.145^{*}	3.332	0.137^{*}
FNNX	0.132	0.000	3.313	0.000	0.178	0.174^{*}	3.325	0.068
FNN	0.130	0.000	3.311	0.000	0.176	0.485^{**}	3.325	0.024
ENNX	0.133	0.000	3.313	0.000	0.164	0.663^{**}	3.321	0.549^{**}
ENN	0.138	0.000	3.312	0.000	0.172	0.552^{**}	3.322	0.542^{**}
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^{**}
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	0.150^{*}	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^{**}

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 with MCS p-value larger than 0.1 and 0.3 are identified by one and two asterisks, respectively. Values in boldface represent the lowest average losses.

Table 5: MCS with $\alpha = 0.10$ and $10,000$ bootstraps (Subsample - 2007.09 to 2009.06)

	$k = 1$				$k = 5$			
Model	MSE		QLIKE		MSE		QLIKE	
	Loss	P_{MCS}	*Loss*	P_{MCS}	*Loss*	P_{MCS}	*Loss*	P_{MCS}
ARFIMAX	0.166	0.802^{**}	2.857	1.000^{**}	0.295	1.000^{**}	2.890	1.000^{**}
ARFIMA	0.244	0.226^{*}	2.878	0.546^{**}	0.571	0.582^{**}	2.939	0.616^{**}
LSTARX	0.467	0.002	2.957	0.014	0.503	0.122^{*}	2.947	0.083
LSTAR	0.221	0.088	2.872	0.479^{**}	0.488	0.432^{**}	2.930	0.453^{**}
FNNX	0.176	0.069	2.864	0.178^{*}	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	0.220^{*}	2.875	0.318^{**}	**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^{**}
LSTMX	0.151	1.000^{**}	2.853	1.000^{**}	0.533	0.438^{**}	2.936	0.351^{*}
LSTM	**0.080**	1.000^{**}	**2.833**	1.000^{**}	0.460	0.202^{*}	2.952	0.075
NARX	**0.052**	1.000^{**}	**2.824**	1.000^{**}	0.266	1.000^{**}	2.884	1.000^{**}
NAR	0.237	0.111^{*}	2.878	0.260^{*}	0.358	1.000^{**}	2.905	0.740^{**}

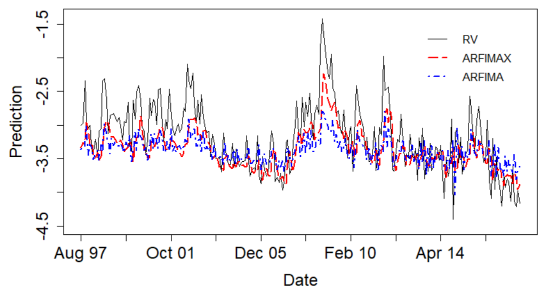
This table reports the average loss over the evaluation sample and the MCS p-values calculated on the basis of the range statistics. The realized volatility forecasts with MCS p-value larger than 0.1 and 0.3 are identified by one and two asterisks, respectively. Values in boldface represent the lowest average losses.

Table 6: Diebold-Mariano test of equal predictive accuracy

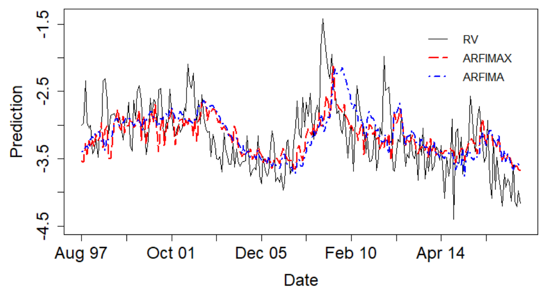
	Entire sample		Subsample	
Model	$k = 1$	$k = 5$	$k = 1$	$k = 5$
ARFIMAX	-1.714*	4.284**	1.648	2.527**
ARFIMA	-3.490***	2.660***	0.130	0.025
LSTARX	-0.182	-1.953*	-2.518**	0.462
LSTAR	1.308	0.812	0.470	0.755
FNNX	1.454	3.552***	1.343	3.602***
FNN	2.457**	3.523***	1.241	3.663***
ENNX	0.778	4.258***	1.191	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**
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*

This table reports the t-statistics for the Diebold-Mariano 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. A positive and statistically significant difference means that the model in the line predicts better than simply using y_t.

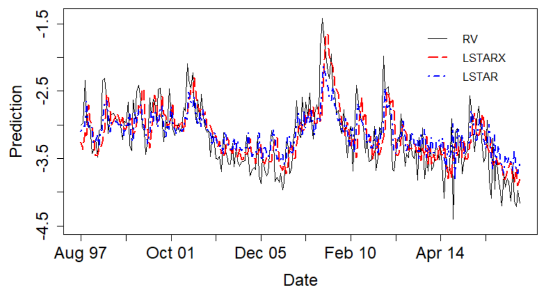
Figure 8: Entire sample forecasts comparison.



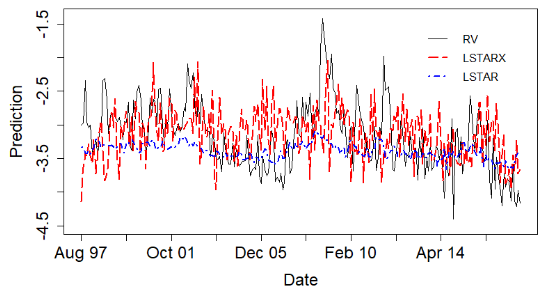
a) ARFIMA One-Step-Ahead.



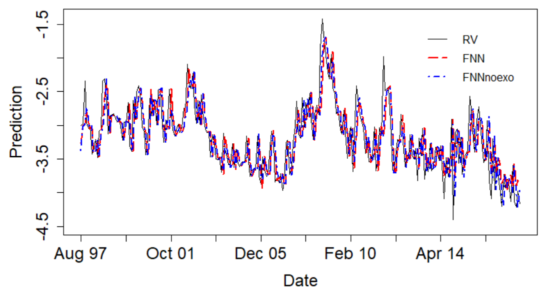
b) ARFIMA 5-Step-Ahead.



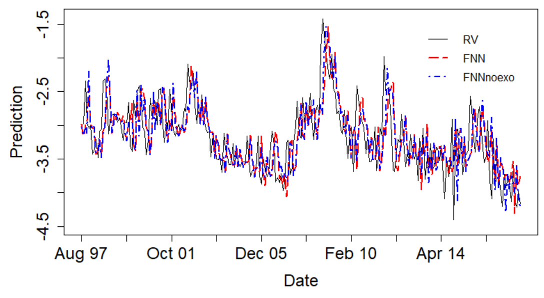
c) LSTAR One-Step-Ahead.



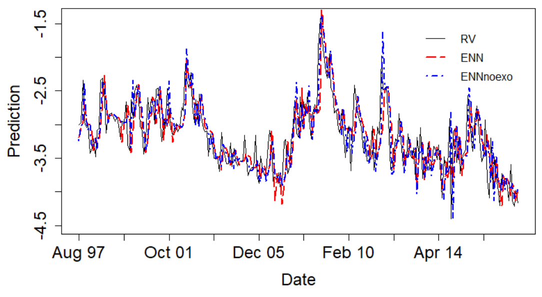
d) LSTAR 5-Step-Ahead.



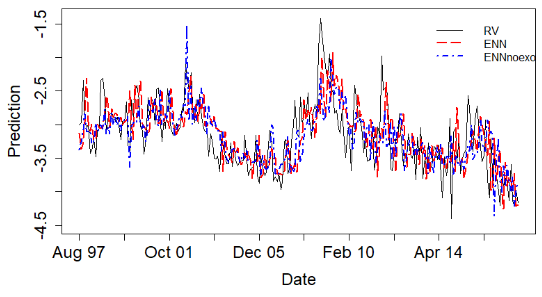
e) FNN One-Step-Ahead.



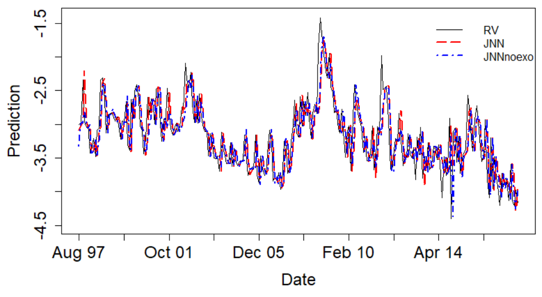
f) FNN 5-Step-Ahead.



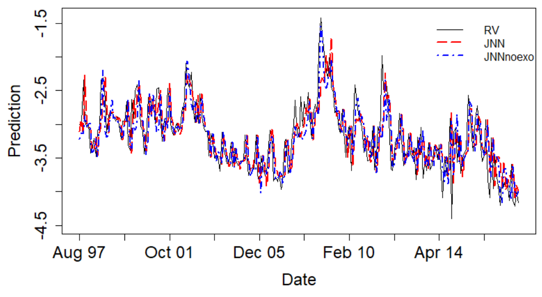
g) ENN One-Step-Ahead.



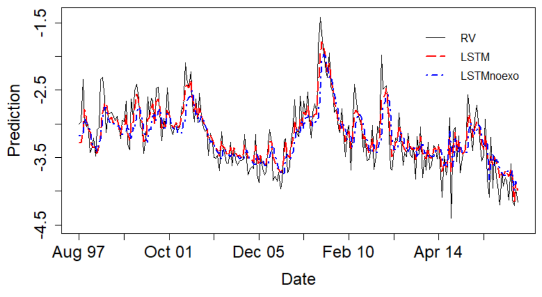
h) ENN 5-Step-Ahead.



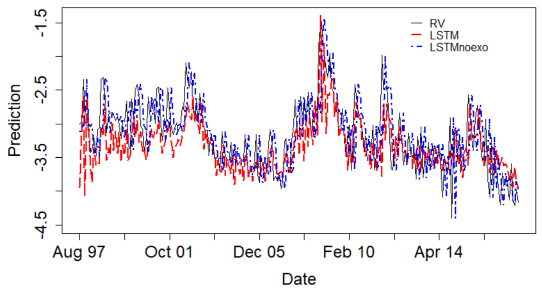
i) JNN One-Step-Ahead.



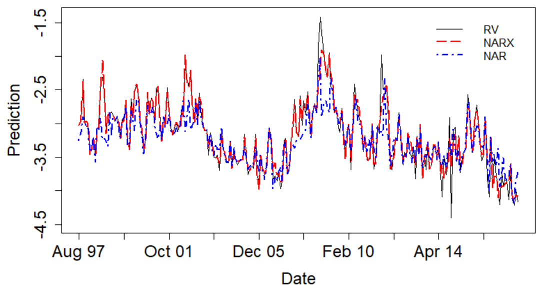
j) JNN 5-Step-Ahead.



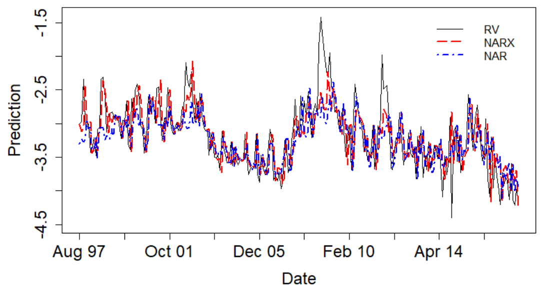
k) LSTM One-Step-Ahead.



l) LSTM 5-Step-Ahead.

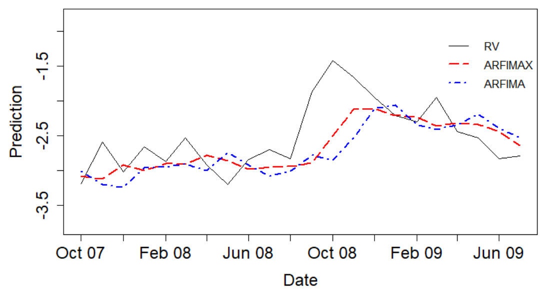


k) NARX One-Step-Ahead.

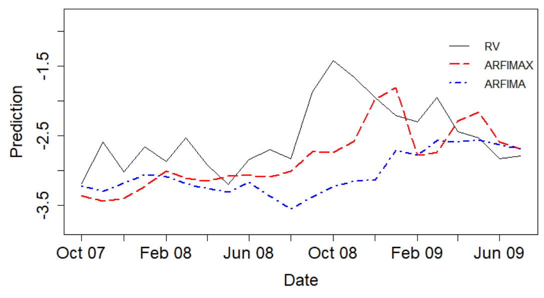


l) NARX 5-Step-Ahead.

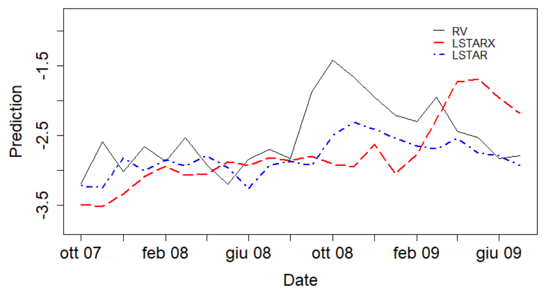
Figure 9: Subsample sample forecasts comparison.



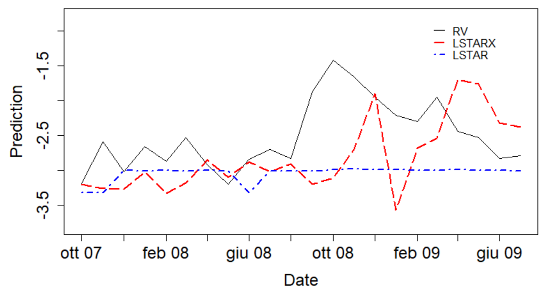
a) ARFIMA One-Step-Ahead.



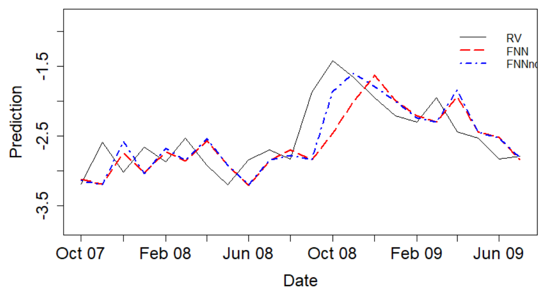
b) ARFIMA 5-Step-Ahead.



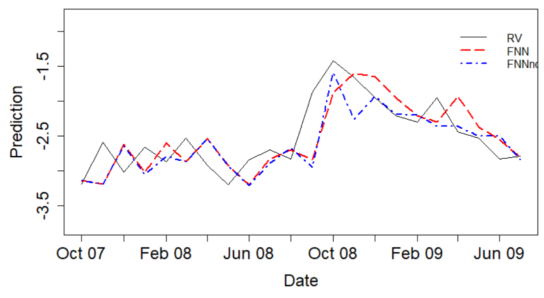
c) LSTAR One-Step-Ahead.



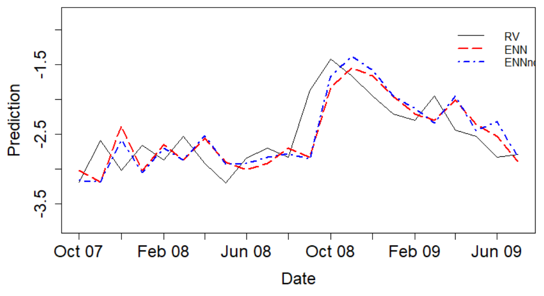
d) LSTAR 5-Step-Ahead.



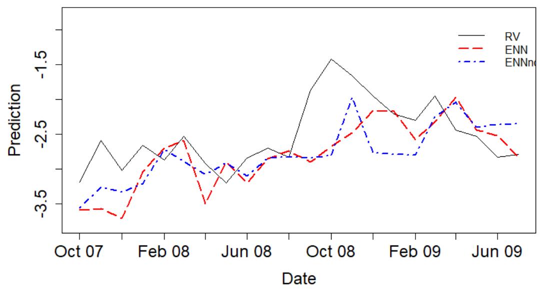
e) FNN One-Step-Ahead.



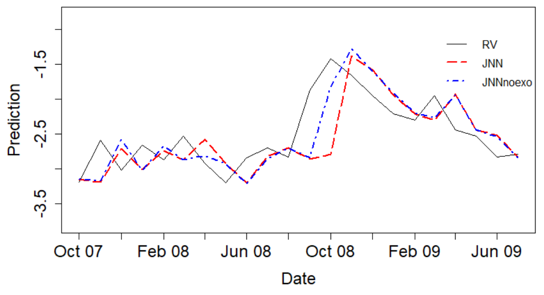
f) FNN 5-Step-Ahead.



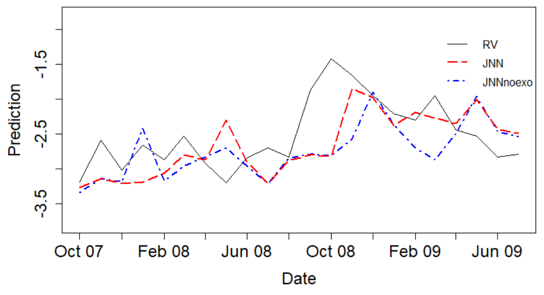
g) ENN One-Step-Ahead.



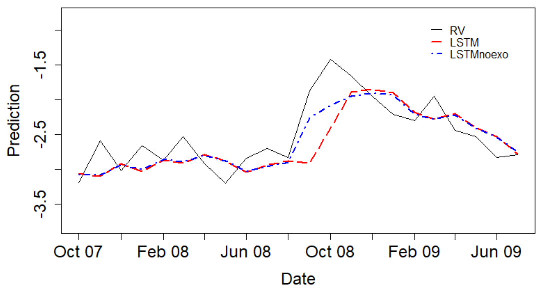
h) ENN 5-Step-Ahead.



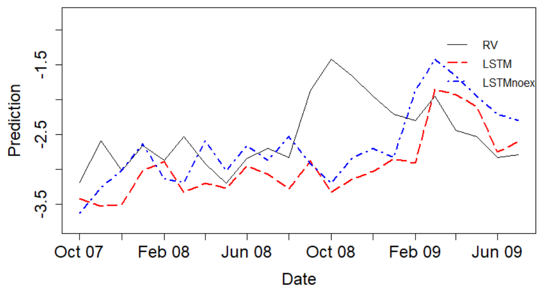
i) JNN One-Step-Ahead.



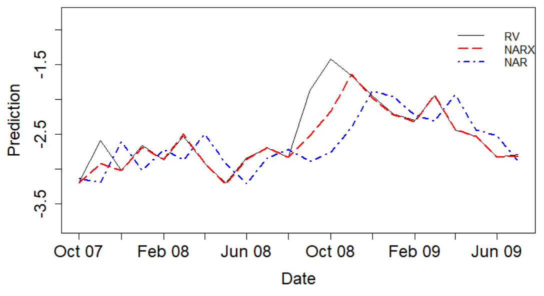
j) JNN 5-Step-Ahead.



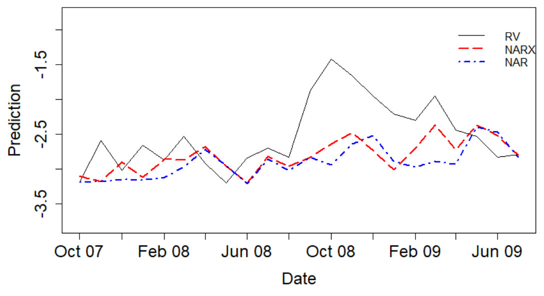
k) LSTM One-Step-Ahead.



l) LSTM 5-Step-Ahead.



k) NARX One-Step-Ahead.



l) NARX 5-Step-Ahead.

6 Conclusions

In this paper, a flexible nonlinear tool for forecasting volatility has been applied. The purpose of the article was to understand whether artificial neural networks 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 recurrent neural networks 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 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, multi-step-ahead recursive forecasts were further compared. Among the main results, it emerged that long-term memory detecting neural networks have the best performance when a large sample is analysed, and provide comparable performance with other methods when a smaller and more volatile sample is evaluated.

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. 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 easy interpretation of explanatory variables due to the structure of the layers. On this purpose, the author acknowledges that this paper was mainly focused on providing superior forecasting accuracy rather than interpreting causal relationships.

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

Moreover, in this paper 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 paper to the multivariate setting.

References

Anders, U., Korn, O., 1996. Model selection in neural networks. ZEW Discussion Papers, No. 96-21.

Andersen, T.G., Bollerslev, T., Diebold, F.X., Ebens, H., 2001. The distribution of realized stock return volatility. Journal of Financial Economics 61, 43–76.

Arnerić, J., Poklepovic, T., Aljinović, Z., 2014. GARCH based artificial neural networks in forecasting conditional variance of stock returns. Croatian Operational Research Review 5, 329–343.

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

Bao, W., Yue, H., Rao, Y., 2017. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PLoS One 12.

Barndorff-Nielsen, O.E., Shephard, N., 2002. Econometric analysis of realised volatility and its use in estimating stochastic volatility models. Journal of Royal Statistical Society 64, 253–280.

Bianchi, F.M., Kampffmeyer, M.C., Rizzi, A., Jenssen, R., 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.

Choudhry, T., Papadimitriou, F.I., Shabi, S., 2016. Stock market volatility and business cycle: Evidence from linear and nonlinear causality tests. Journal of Banking & Finance 66, 89–101.

Christiansen, C., Schmeling, M., Schrimpf, A., 2012. A comprehensive look at financial volatility prediction by economic variables. Journal of Applied Econometrics 27, 956–977.

Clements, M.P., Krolzig, H.M., 1998. A comparison of the forecast performance of markov-switching and threshold autoregressive models of US GNP. The Econometrics Journal 1, 47–75.

De Pooter, M., Martens, M., Van Dijk, D., 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., Honchar, O., 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., Mariano, R.S., 1995. Comparing predictive accuracy. Journal of Business and Economic Statistics 13, 253–263.

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

Donaldson, G.R., Kamstra, M., 1996a. A new dividend forecasting procedure that rejects bubbles in asset prices. Review of Financial Studies 8, 333–383.

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

Donaldson, G.R., Kamstra, M., 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., Ghysels, E., Sohn, B., 2009. On the Economic Sources of Stock Market Volatility. Working paper.

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

Fernandes, M., Medeiros, M.C., Scharth, M., 2014. Modeling and predicting the CBOE market volatility index. Journal of Banking & Finance 40, 1–10.

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

Guzman, S.M., Paz, J.O., Tagert, M.L.M., 2017. The Use of NARX Neural Networks to Forecast Daily Groundwater Levels. Water Resour Manage 31, 1591–1603.

Hajizadeh, E., Seifi, A., Fazel Zarandi, M., Turksen, I., 2012. A hybrid modeling approach for forecasting the volatility of S&P 500 index return. Expert Systems with Applications 39, 431–436.

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

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

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Computation 9, 1735–1780.

Hu, Y.M., Tsoukalas, C., 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. Technical Report. Institute for Cognitive Science Report 8604, UC San Diego.

Kamijo, K., Tanigawa, T., 1990. Stock price pattern recognition - a recurrent neural network approach. 1990 IJCNN International Joint Conference on Neural Networks.

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., 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.

Lin, T., Horne, B.G., Tino, P., Giles, C.L., 1996. Learning long-term dependencies in NARX recurrent neural networks. IEEE Transactions on Neural Networks 7, 1329–1338.

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

Maheu, J.M., McCurdy, T.H., 2002. Nonlinear features of realized volatility. Review of Economics and Statistics 84, 668–681.

McAleer, M., Medeiros, M., 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., Barreto, G., 2006. A New Look at Nonlinear Time Series Prediction with NARX Recurrent Neural Network, in: 2006 Ninth Brazilian Symposium on Neural Networks (SBRN'06), pp. 160–165.

Nelson, D.B., 1990. Stationarity and persistence in the GARCH (1,1) model. Econometric Theory 6, 318–334.

Panchal, G., Ganatra, A., Kosta, Y., Panchal, D., 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., Paya, I., Peel, D.A., 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., Kaizoji, T., 2017. Volatility Analysis of Bitcoin Price Time Series. Quantitative Finance and Economics 1, 474–485.

Rossi, E., Santucci de Magistris, P., 2014. Estimation of long memory in integrated variance. Econometric Reviews 33, 785–814.

Schittenkopf, C., Dorffner, G., Dockner, E.J., 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. Journal of Finance 44, 1115–1153.

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

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

Tino, P., Schittenkopf, C., Dorffner, G., 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., Goyal, A., 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.

Xiong, R., Nichols, E.P., Shen, Y., 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.