Application of Back Propagation Gradient Based Neural Network Training Algorithms To Volatility Forecasting

S. Suleiman[1]; S.U. Gulumbe[2]

[1] Department of Mathematics, Usmanu Danfodiyo University, Sokoto, Sokoto State, Nigeria.
[2] Department of Mathematics and Statistics, Federal University Birnin Kebbi, Kebbi State, Nigeria.
email: suleman.shamsuddeen@udusok.edu.ng[1]; ugulumbe@gmail.com[2]

***Abstract* — Volatility forecasting has been the subject of recent empirical studies and theoretical investigation both in academia and financial markets because it is one of the primary inputs to a wide range of financial applications from risk measurement to asset and option pricing. GARCH family of models have been extensively applied in volatility forecasting, but one of their limitations is that these models produce better results in relatively stable markets and could not capture violent volatilities and fluctuations. Neural Networks (NNs) trained by Back Propagation gradient descent algorithm are known to have the capability to learn any complex approximate relationships between the inputs and the outputs with a very slow convergence rate for most practical applications. In this study, we proposed two hybrid models based on EGARCH and Recurrent Dynamic Neural Networks trained by dynamic Back Propagation gradient based training algorithms to forecast the volatility of inflation rate returns in Nigeria. The estimates of volatility obtained by an EGARCH model are fed forward to a Neural Network. The input to the first model is complemented by historical values of the other explanatory variable. The second hybrid model takes as inputs both series of the simulated data and explanatory variables. The forecasts obtained by each of those hybrid models have been compared with those of EGARCH model in terms of closeness to standard deviation which is used as a measure of the actual value of volatility. The results show that the second hybrid model trained by Bayesian Regularization algorithms gives better volatility forecasts. This model significantly improves the forecasts over the ones obtained by the EGARCH model.**

Keywords-Forecasting, Volatility, GARCH, Neural Network, Back propagation.

A. INTRODUCTION

Neural Networks (NNs) are non-linear data driven self-adaptive approach as opposed to the traditional model based methods. They are powerful tools for modelling, especially when the underlying data relationship is unknown. These models have the capability to learn the complex approximate relationships between the inputs and the outputs of the system and are not restricted by the size and complexity of the system. The algorithms also learn these approximate relationships on the basis of actual inputs and outputs. Therefore, they are generally more precise compared to the relationships based on assumptions such as regressive and structural models.

The modern view of neural networks began in the 1940s with the work of McCulloch and Pitts (1943), who showed that networks of artificial neurons could, in principle, compute any arithmetic or logical function. Their work is often acknowledged as the origin of the neural network field. McCulloch and Pitts were followed by Hebb (1949), who proposed that classical conditioning is present because of the properties of individual neurons. He proposed a mechanism for learning in biological neurons.

The first practical application of artificial neural networks came in the late 1950s, with the invention of the perceptron network and associated learning rule by Rosenblatt (1958) where he built a perceptron network and demonstrated its ability to perform pattern recognition. This early success generated a great deal of interest in neural network research. At about the same time, Widrow and Hoff (1960) introduced a new learning algorithm called Least Mean Square (LMS) algorithm and used it to train adaptive linear neural networks, which were similar in structure and capability to Rosenblatt's perceptron.

The perceptron learning rules developed by (Rosenblatt, 1958) and Least Mean Square (LMS) algorithm developed by (Widrow and Hoff,1960) were designed to train single-layer perceptron-like networks which can only be used to solve linearly separable classification problems. Both Rosenblatt and Widrow were aware of these limitations and proposed Multilayer networks that could overcome them, but were not able to generalize their algorithms to train these more powerful networks.

One of the key developments in the research of NNs in 1980s was the Back Propagation algorithm for training

Professional Statisticians

multilayer perceptron networks, which was discovered independently by several different researchers such as Rumelhart (*et al*.,1986) and Rumelhart and McClelland (1986). Back Propagation is a generalization of the LMS algorithm that can be used for multilayer networks; it is an approximate steepest descent algorithm, in which the performance index is the mean square error.

The difference between the LMS algorithm and Back Propagation is only in the way in which the derivatives are calculated. For a single-layer linear network the error is an explicit linear function of the network weights, and its derivatives with respect to the weights can be easily computed.

In multilayer networks with nonlinear transfer functions, the relationship between the network weights and the error is more complex. In order to calculate the derivatives, we need to use the chain rule of calculus. The Back Propagation algorithm uses the chain rule in order to compute the derivatives of the squared error with respect to the weights and biases in the hidden layers. It is called Back Propagation because the derivatives are computed first at the last layer of the network, and then propagated backward through the network, using the chain rule, to compute the derivatives in the hidden layers.

The neural network model is an emerging computational technology that provides a new avenue for examining the dynamics of various economic and financial applications. Some of the applications of NNs to model volatility can be seen in Tseng (*et al*., 2008); Chang (*et al*., 2011) and Kristjanpoller (*et al*., 2014).

Volatility, with respect to financial products, can be thought of as a measure of fluctuation in a financial security price around its expected value. It is one of the primary inputs to a wide range of financial applications from risk measurement to asset and option pricing. When discussing the volatility of time series, econometricians refer to the 'conditional variance' of the data and the time-varying volatility typical of asset returns which is otherwise known as 'conditional heteroscedasticity'.

The concept of conditional heteroscedasticity was introduced to economists by Engle (1982), who proposed a model in which the conditional variance of a time series is a function of past shocks; the autoregressive conditional heteroscedastic (ARCH) model. The model provided a rigorous way of empirically investigating issues involving the volatility of economic variables. An example is Friedman's hypothesis that higher inflation is more volatile (Friedman, 1977). In another work, Engle (1982) found that the ARCH model supported Friedman's hypothesis. Engle (1983) applied the ARCH model to US inflation and the converse results emerged, although Cosimano and Jansen (1988) believed that Engle estimates a mis-specified model. The relationship between the level and variance of inflation has continued to interest applied econometricians see for example, Grier and Perry (2000)
Traditional ARCH models were recently used by several authors in Nigeria to forecast volatility (Dauda, 2008; Olowe, 2009; Arowolo, 2013; Dahiru and Joseph, 2013; Yaya, 2013; Amaefula and Asare, 2014).
The Back Propagation algorithm is an extension of the LMS algorithm that can be used to train multilayer networks. Both LMS and Back Propagation are approximate steepest descent algorithms that minimize squared error. It is known to have the capability to learn any complex approximate relationships between the inputs and the outputs. The multilayer perceptron, trained by the Back Propagation algorithm, is currently the most widely used Neural Network (Hagan *et al*., 1996). However, one of the major problems with Back Propagation has been the long training times. It is not feasible to use the basic Back Propagation algorithm on practical problems, because it might take weeks to train a network.

Financial time series forecasting is one of the most challenging applications of modern time series analysis as they are inherently noisy, non-stationary and deterministically chaotic. The desire to forecast volatility of financial markets has motivated a large body of research during the past decades.

The conditional volatility, calculated using ARCH model by Engle (1982), is currently the most popular method of estimating volatility. Although, many financial time series observations have non-linear dependence structure, a linear correlation structure is usually assumed among the time series data. Therefore, ARCH type models may not capture such nonlinear patterns and linear approximation models of those complex problems may not be satisfactory.

Nonparametric models estimated by various methods such as Artificial Intelligence (AI), can be fit on a data set much better than linear models. Recently, (Hajizadeh *et al* 2012) has hybridized EGARH and Neural Networks models to forecast the volatility of S&P 500 index but solely trained with the standard Back Propagation gradient descent algorithm which is known to have slow rate of convergence.

The aim of this paper is to improve the performance of GARCH family models in forecasting Volatility of inflation rate returns in Nigeria.

The data used in this research consists of monthly inflation rates, crude oil price, consumer price index, export, import, money supply, interest rate and premium motor spirit (PMS) price obtained through Central Bank of Nigeria website www.cbn.gov.ng covering the period between January, 1995 and February, 2016.

Professional Statisticians

B. MULTILAYER NEURAL NETWORK

Multilayer networks are universal approximators and can be represented as in figure 1. To train such networks means determining a procedure for selecting the network parameters (weights and biases) which will best approximate a given function. The procedure for selecting the parameters for a given problem is called *training* the network. The Back propagation algorithm for multilayer networks is a gradient descent optimization procedure in which we minimize a mean square error performance index. The algorithm is provided with a set of examples of proper network behaviour:

$$\{p_1 t_1\},\{p_2 t_2\},\ldots,\{p_q t_q\}, \quad (1)$$

Where p_q is an input to the network, and t_q is the corresponding target output. As each input is applied to the network, the network output is compared to the target. The algorithm should adjust the network parameters in order to minimize the sum squared error:

$$F(x) = \sum_{q=1}^{Q} e_q^2 = \sum_{q=1}^{Q} (t_q - a_q)^2 \quad (2)$$

Where x is a vector containing all of network weights and biases. If the network has multiple outputs this generalizes to

$$F(x) = \sum_{q=1}^{Q} e_q^T e_q = \sum_{q=1}^{Q} (t_q - a_q)^T (t_q - a_q). \quad (3)$$

Using a stochastic approximation, we will replace the sum squared error on the latest target:

$$\hat{F}(x) = (t(k) - a(k))^T \big(t(k) - a(k)\big) = e^T(k) e(k), \quad (4)$$

Where the expectation of the squared error is replaced by the squared error at iteration k.

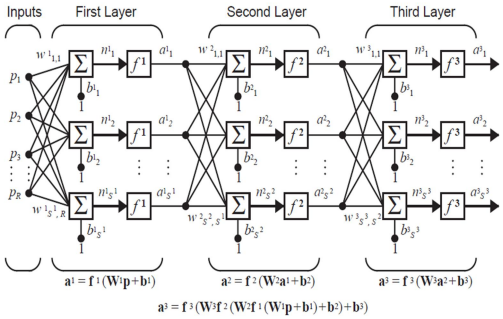

Fig. 1: Multilayer Networks

The Back Propagation algorithm can be summarized as follow:

- *The first step is to propagate the input forward through the network:*

$$\boldsymbol{a}^0 = \boldsymbol{p}, \quad (5)$$

$$\boldsymbol{a}^{m+1} = f^{m+1}(\boldsymbol{W}^{m+1}\boldsymbol{a}^m + \boldsymbol{b}^{m+1}) \text{ for } m = 0,1,\ldots,M-1, \quad (6)$$

$$\boldsymbol{a} = \boldsymbol{a}^M. \quad (7)$$

where M is the number of layers in the network. The neurons in the first layer receive external inputs.

- The next step is to propagate the sensitivities backward through the network:

$$s^M = -2\dot{F}^M(\boldsymbol{n}^M)(t - \boldsymbol{a}), \quad (8)$$

$$s^m = \dot{F}^m(\boldsymbol{n}^m)(\omega^{m+1})^T s^{m+1}, m = M-1,\ldots,2,1 \quad (9)$$

where

$$\dot{F}^m(n^m) = \begin{bmatrix} \dot{f}^m(n_1^m) & 0 \ldots 0 \\ 0 & \dot{f}^m(n_2^m) \ \ldots 0 \\ 0 & 0 \ \ldots \dot{f}^m(n_{s^m}^m) \end{bmatrix} \quad (10)$$

- Finally, the weights and biases are updated using the approximate steepest descent rule:

$$\boldsymbol{W}^m(k+1) = \boldsymbol{W}^m(k) - \alpha s^m (\boldsymbol{a}^{m-1})^T, \quad (11)$$

$$\boldsymbol{b}^m(k+1) = \boldsymbol{b}^m(k) - \alpha s^m \quad (12)$$

where

$$s^m = \frac{\partial \hat{F}}{\partial \boldsymbol{n}^m} = \begin{bmatrix} \frac{\partial \hat{F}}{\partial \boldsymbol{n}_1^m} \\ \frac{\partial \hat{F}}{\partial \boldsymbol{n}_2^m} \\ \vdots \\ \frac{\partial \hat{F}}{\partial \boldsymbol{n}_{s^m}^m} \end{bmatrix} \quad (13)$$

Since the performance index F in Eq. (2) is a function of weights and biases, $x = [x_1 x_2 \ \ldots x_n]$ and can be given by

$$F(x) = \frac{1}{N}\sum_{q=1}^{N} e_q^2(x) \quad (14)$$

The performance of the neural network can be improved by modifying x till the desired level of the performance index, $F(x)$ is achieved. This is achieved by minimizing $F(x)$ with respect to x and the gradient required for this is given by

$$\nabla F(x) = J^T(x) e(x) \quad (15)$$

where, $J(x)$ is the Jacobian matrix given by

$$J(x) = \begin{bmatrix} \frac{\partial e_1(x)}{\partial x_1} \cdots \frac{\partial e_1(x)}{\partial x_n} \\ \vdots \quad \ddots \quad \vdots \\ \frac{\partial e_N(x)}{\partial x_1} \cdots \frac{\partial e_N(x)}{\partial x_n} \end{bmatrix} \quad (16)$$

and $e(x)$ is the error for all the inputs. The gradient in (15) is determined using back propagation algorithms discussed above, which involves performing computations backward through the network. The process stops if a pre-specified criterion is fulfilled, i.e. if the values of the gradient are smaller than a given threshold.

This gradient is then used by different algorithms to update the weights of the network. These algorithms differ in the way they use the gradient to update the weights of the network and are known as the variants of the back propagation algorithms.

Gradient descent algorithm with other variants is discussed in what follows.

Professional Statisticians

i. **Gradient Descent**

The network weights and biases, x is modified in a direction that reduces the performance `function in (14) most rapidly i.e. the negative of the gradient of the performance function (Hagan *et al*., 1996). The updated weights and biases in this algorithm are given by

$$x_{k+1} = x_k - \alpha_k \nabla F_k \quad (17)$$

where, x_k is the vector of the current weights and biases, ∇F_k is the current gradient of the performance function and α_k is the learning rate.

ii. **Scaled Conjugate Gradient Descent Algorithm (SCGDA)**

The gradient descent algorithm updates the weights and biases along the steepest descent direction but is usually associated with poor convergence rate as compared to the Conjugate Gradient Descent algorithms, which generally result in faster convergence (Moller, 1993). In the Conjugate Gradient Descent algorithms, a search is made along the conjugate gradient direction to determine the step size that minimizes the performance function along that line.

This time consuming line search is required during all the iterations of the weight update. However, the Scaled Conjugate Gradient Descent algorithm does not require the computationally expensive line search and at the same time has the advantage of the Conjugate Gradient Descent algorithms (Moller, 1993). The step size in the conjugate direction in this case is determined using the Levenberg-Marquardt approach. The algorithm starts in the direction of the steepest descent given by the negative of the gradient as

$$p_f = -\nabla F_v \quad (18)$$

The updated and weights and biases are then given by

$$x_{k+1} = x_k + \alpha_k p_k \quad (19)$$

where α_k is the step size determined by the Levenberg-Marquardt algorithm (Hagan and Menhaj., 1994). The next search direction that is conjugate to the previous search directions is determined by combining the new steepest descent direction with the previous search direction and is given by

$$p_k = -\nabla F_k + \beta_k p_{k-1} \quad (20)$$

The value of β_k is given by

$$\beta_k = \frac{|\nabla F_{k+1}|^2 - \nabla V_{k+1} \nabla F_k}{\mu_k} \quad (21)$$

where μ_k is given by

$$\mu_k = p_k{}^{\mathrm{T}} \nabla F_k \quad (22)$$

iii. **Levenberg-Marquardt (LM) Algorithm**

Since the performance index in (14) is sum of squares of nonlinear function, the numerical optimization techniques for nonlinear least squares can be used to minimize this cost function. The Levenberg-Marquardt algorithm, which is an approximation to the Newton's method is said to be more efficient in comparison to other methods for convergence of the Back Propagation algorithm for training a moderate-sized feed forward neural network (Hagan and Menhaj., 1994). As the cost function is a sum of squares of nonlinear function, the Hessian matrix required for updating the weights and biases need not be calculated and can be approximated as

$$H = J^T(x)J(x) \quad (22)$$

The updated weights and biases are given by

$$x_{k+1} = x_k - [J^T(x)J(x) + \mu I]^{-1} J^T(x)e(x) \quad (23)$$

where μ is a scalar and I is the identity matrix.

iv. **Automated Bayesian Regularization (BR)**

Regularization as a mean of improving network generalization is used within the Levenberg Marquardt algorithm. Regularization involves modification in the performance function. The performance function for this is the sum of the squares of the errors and it is modified to include a term that consists of the sum of squares of the network weights and biases. The modified performance function is given by

$$F_{reg} = \beta SSE + \alpha SSW \quad (24)$$

Where SSE and SSW are given by

$$SSE = \sum_{q=1}^{N} e_q^2(x) \quad (25)$$

$$SSW = \sum_{j=1}^{n} w_j^2 \quad (26)$$

where n is the total number of weights and biases, w_j in the network. The performance index in (24) forces the weights and biases to be small, which produces a smoother network response and avoids over fitting. The values α and β are determined using Bayesian regularization in an automated manner (Foresee and Hagan., 1997) and (Mackay, 1992)

The Baye's theorem relates two variables (or events), A and B, based on their prior (or marginal) probabilities and posterior (or conditional) probabilities as in (27):

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (27)$$

Professional Statisticians

where $P(A|B)$ is the posterior probability of A conditional on B. $P(B|A)$ is the prior of B conditional on A, and $P(B)$ the non-zero prior probability of event B, which functions as a normalizing constant. In order to find the optimal weight space, objective function (24) needs to be minimized, which the equivalent of maximizing the posterior probability function is given as in (28) (Hagan *et al.*,1996):

$$P(\alpha,\beta|D,M) = \frac{P(D|\alpha,\beta,M)P(\alpha,\beta|M)}{P(D|M)} \quad (28)$$

where α and β are the factors needed to be optimized, D is the weight distribution, M is the particular neural network architecture, $P(D|M)$ is the normalization factor, $P(\alpha,\beta|M)$ is the uniform prior density for the regularization parameters and $P(D|\alpha,\beta,M)$ is the likelihood function of D for given α,β,M.

Maximizing the posterior function $P(\alpha,\beta|D,M)$ is equivalent of maximizing the likelihood function $P(D|\alpha,\beta,M)$. As a result of this process, optimum values for α and β for a given weight space are found. Afterwards, algorithm moves in to LM phase where Hessian matrix calculations take place and updates the weight space in order to minimize the objective function. Then, if the convergence is not met, algorithm estimates new values for α and β and the whole procedure repeats itself until convergence is reached.

V. Neural Network Autoregressive with Exogenous Input(NNARX)

The Neural Network Autoregressive with Exogenous Input(NNARX) (NNARX) is a recurrent dynamic network, with feedback connections enclosing several layers of the network. The NNARX model is based on the linear ARX model, which is commonly used in time-series modelling and forecasting. The NNARX model can be represented as follows:

$$y(t) = f(y(t-1), y(t-2) \dots, y(t-n), u(t-1), u(t-2), \dots, u(t-n)) \quad (29)$$

where the next value of the dependent output signal $y(t)$ is regressed on previous values of the output signal and previous values of an independent (exogenous) input signal. The output is feed backed to the input of the feed-forward neural network as part of the standard NNARX architecture as shown in Fig 2.

Since the true output is available during the training, one could create a series parallel architecture in which the true output is used instead of feeding back the estimated output as shown in Fig 3. This has two advantages, first is that the input to the feed-forward network is more accurate, second is that the resulting network has purely feed-forward architecture and static back propagation can be used for training.

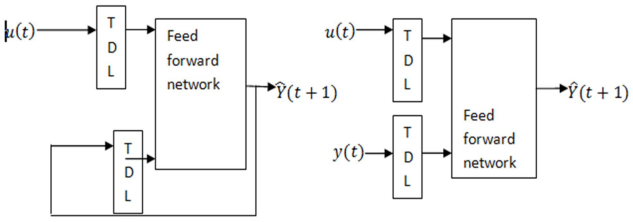

Fig 2: Parallel architecture of NNARX

Fig 3 Series Parallel architecture of NNARX

Dynamic networks can be trained in the same gradient-based algorithm that is applied in back propagation. Although the method of training is same with static networks but the performance of this algorithm in dynamic networks is different from static networks because the gradient is computed in a more complex way.

VI. Proposed Hybrid Models

In this study, two hybrid models were proposed for forecasting conditional volatility of inflation Rate in Nigeria. Initially, in each of the proposed models, a preferred GARCH model is identified upon which the hybrid model is built. For this purpose, optimum lags for each GARCH model was estimated using AIC and BIC indices. Then, each model is used for predicting some forecasts of the price returns volatility of inflation rate and the preferred model was selected according to pre-defined measures.

a. Hybrid model I

The underlying concept for the first hybrid model is that there are some explanatory factors other than historical prices that affect the future price returns volatility in the Market. We forecast volatility of price returns for inflation rate in Nigeria with a number of market variables which affects its price returns.

Selection of the input variables depends on the knowledge of which ones affect volatility significantly. Some endogenous variables related to the historical performance of the returns such as price returns, squared price returns, price, price squared, etc (based on the preferred model) are used. The exogenous variables which likely influence price returns were also considered. Both the endogenous and exogenous variables were used as the input variables to the Artificial Neural Network (ANN) model and the standard deviation was considered to be target output for the training the network.

Professional Statistics

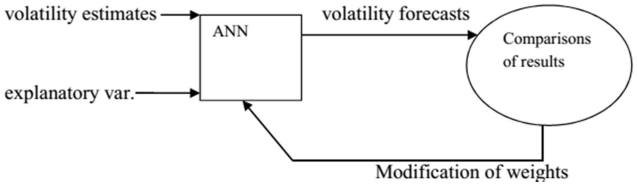

Fig 6: Schematic representation of hybrid model I

b. Hybrid model II

In order to keep the properties of the best fitted GARCH model while enhancing it with an ANN model, we have to somehow introduce the autocorrelation structure of data (captured by GARCH model) to the network. Otherwise, the hybrid model could not recognize the underlying autocorrelation from a single set of estimated time series. Therefore, one has to generate synthetic series with the same statistical properties as the estimated volatility. Simulation is a widely used technique to generate synthetic series. Hybrid model II is constructed using several simulated GARCH series instead of a single estimated series.

The number of simulated series for training the Neural Network depends on nature of the problem and type of data. It also takes exactly the same market variables as hybrid model I as input. The standard deviation is also be used as the output target of the network. It is expected that this model better captures the characteristics of the GARCH model as well as the impacts of the market variables.

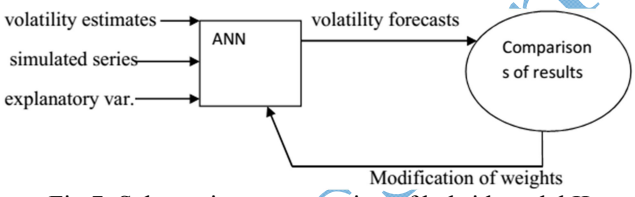

Fig 7: Schematic representation of hybrid model II

VI. Training of the Proposed Hybrid Models

The neural network proposed in section 3.3 and sub-sections 3.3.1 and 3.3.2 were trained with training algorithms described in section 3.1 and using the weights and biases updates algorithms in subsection 3.1.1, but the gradient computation is based on the principle of dynamic learning described in section 3.2.

An Intel (R) Core (TM) i3-2310M CPU @ 2.10GHz processor was used to train the proposed neural network models. Figures 8 and 9 are examples of the trained NARX configuration which are used for training purposes which are generally referred to as (Open loop), multi-step ahead prediction also known as (Closed loop). After the training is done in an open loop (also called series-parallel architecture, including the validation and testing steps.

The typical workflow is to fully create the network in open loop, and only when it has been trained (which includes validation and testing steps) it is then transformed to closed loop for multi-step ahead prediction. We can now use the closed-loop (parallel) configuration to perform an iterated prediction of nth time steps.

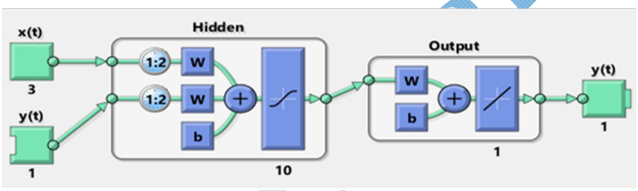

Fig 8: Open-loop Hybrid Model I Architecture for INFRT Volatility Forecaster

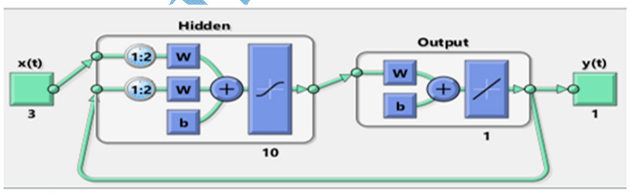

Fig 9: Closed-loop Hybrid Model I Architecture for INFRT Volatility Forecaster

Moreover, Eviews statistical software was used to calibrate GARCH models as well as preliminary time series analysis. Econometric toolbox in the MATLAB programming language was used in simulating the synthetic series for the preferred GARCH models. Having selected EGARCH (2, 1) as the best for predicting inflation rate returns volatilities, it was used in simulating some synthetics series as part of the building blocks for hybrid model II which is depicted by figure 10. The figure shows the log returns of hundred synthetic series from EGARCH (2,1) and its confidence bounds.

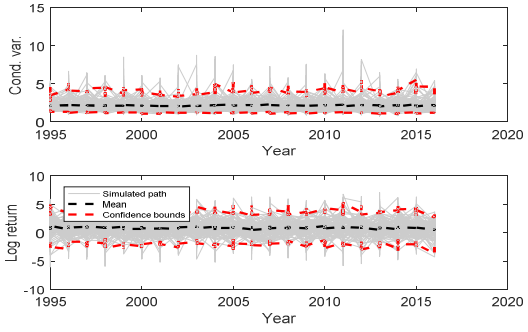

Fig 10: Confidence intervals for the simulated synthetic series for the inflation rate.

VII. RESULTS

Neural Network toolbox was adopted for the implementation and training of the hybrid models.

Table 1: Descriptive statistics of different training algorithms for Hybrid models I and II

Model	Training Algorithms	Average Time(s)	Maximum Time(s)	Minimum Time(s)	Standard Deviation
Hybrid model I	Scaled Conjugate Gradient Descent	3.7	9	1	2.66875
	Levenberg Marquardt	**3.6**	9	1	1.27058
	Bayesian Regularization	7	9	5	2.44949
Hybrid model II	Scaled Conjugate Gradient Descent	16.1	18	15	1.28668
	Levenberg Marquardt	**15.8**	21	12	2.93636
	Bayesian Regularization	slow in convergence			

Table 1 summarizes the results of training the proposed networks using the four training algorithms discussed above. Each entry in the table represents 100 different trials, with random weights taken for each trial to rule out the weight sensitivity of the performance of the different training algorithms. The network was trained in each case till the value of the performance index in equation (14) was 0.0000. The average time required for training the hybrid models using the Levenberg-Marquardt algorithm was generally the least, whereas, maximum time is required for training the network using Scaled Conjugate Gradient Descent algorithm for Hybrid Model I. The training algorithm employing Bayesian Regularization continuously modifies its performance function and hence, takes more time on average to train compared to the Levenberg-Marquardt and Scaled Conjugate Gradient Descent algorithms. In the hybrid model II, when more simulated synthetic series are included, thereby expanding the size and complexity of the network, Bayesian Regularization takes far more time to train than even Hybrid Model I. At that level, it was very slow in convergence.

Similarly some tools were used to confirm/validate the network performance in relation to the performance index defined in equation (14). These tools are based on some basic properties of accurate predictions in terms of the validated mean squared errors and autocorrelation of errors at different time steps. In the first place an average best validation of a mean squared error from each of the three training algorithms were obtained after hundred trials are shown in table 2 for hybrid models I and II in which Bayesian Regularization (BR) achieved the best result(minimum error) but with largest number of iterations.

Table 2: Performance of the trained Hybrid model I and II

Model	Training Methods	Average Absolute Error	Minimum	Maximum	Average no. of Epoch
Hybrid model I	SCGD	0.002283863	0.00020325	0.052454	21
	LM	0.001412116	0.000208118	0.036147	27
	BR	$\mathbf{1.58e^{-12}}$	$1.0598e^{-12}$	$3.0166e^{-12}$	324
Hybrid model II	SCGD	0.001551366	0.00043678	0.0031612	43
	LM	0.00033551	0.00014711	0.000444890	319
	BR	**0.0000050193**	$1.3741e^{-13}$	$8.6788e^{-14}$	307

The neural networks trained using the training algorithms listed in table 1 was tested on the same Central Processing Unit (CPU). The test datasets consisted of data points not included in the training sets. The actual (target) and predicted (output) values of the volatilities according to the hybrid models I and II using the training algorithms with least validation prediction error as shown in table 2 were measured. This is compared in a regression line as shown in Figs 11 and 12 for hybrid I and II respectively. The color line represents the linear regression, the dash line represents the perfect match $Y = T$, and the circles represent data points. Attached to these graphs are the correlations between the targets and network outputs.

In Figs 13 and 14, the response outputs are compared to response targets by the hybrid model I and II respectively across training and testing categories. These graphs, just like the previous ones indicate promising performance because the two values compares very well.

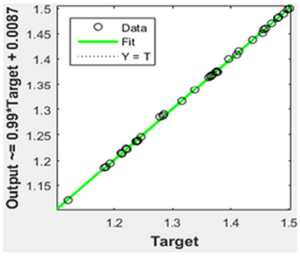

Figure 11: Scatter Plots of Network Outputs vs. Targets for the Hybrid Model I

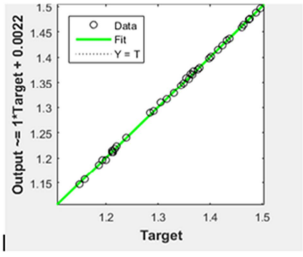

Figure 12: Scatter plots of Network Output vs. Targets for the hybrid model II

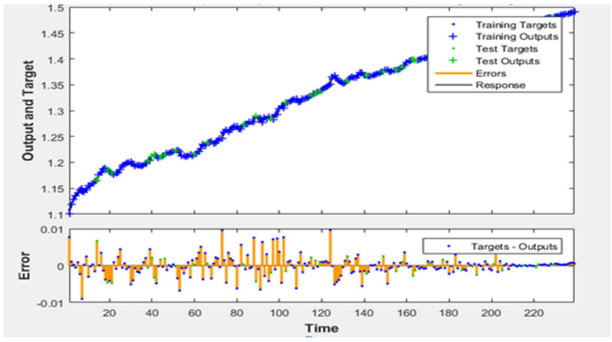

Fig.13: Response of output of Hybrid model I for Inflation Rate Volatility Forecaster using BR Algorithm

Professional Statistician

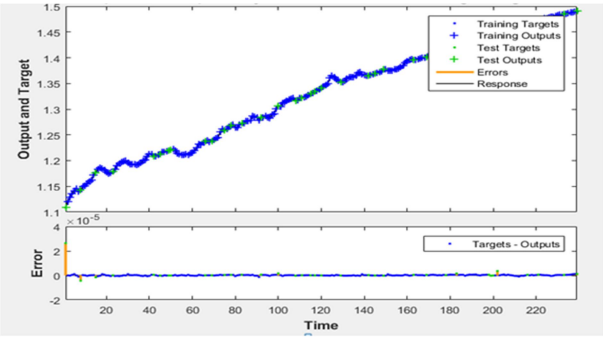

Fig. 14: Response of output of Hybrid model II for Inflation Rate Volatility

VIII. Forecasting Profiles of the Trained Neural Networks

The datasets used to train, validate and test the proposed models covered the periods from January, 1995 to February, 2015. The remaining datasets covering the periods from March,2015 to February, 2016, which were not utilized in the calibration phase but used in this section to evaluate the forecasting performance of the hybrid models.

The trained hybrid models were also used to forecast the volatilities of the various assets returns within these periods (March, 2015 to February, 2016) and compared with the standard deviation as a measure of the actual volatility in order to compute some fitness measures to evaluate the models performances.

Table 3 presents the result of the applications of the proposed hybrid models for forecasting inflation rate returns volatilities in 6 and 12 months ahead.
In this table are the forecasts according to the hybrid models, EGARCH (2, 1) and Hajizadeh *et al* hybrid model based on EGARCH (3, 3).

The computational results show that EGARCH model outperforms hybrid model I and Hajizadeh *et al,* (2012) but could not perform as good as Hybrid model II. It is therefore observed that Hybrid model II perform better than all three other models. That is likely due to inclusion of simulated series as extra inputs to hybrid model II. The forecasts of these models were also depicted in the figures 18 and 19 for 6 and 12 Months ahead forecast respectively.

Table 3: Hybrid models performance to volatility forecasting

Forecasting Frequency	Measures	EGARCH (2,1)	Hajizadeh NN	Hybrid model I	Hybrid model II
6 months ahead forecast	RMSE	0.001737	0.003537	0.003288	0.000774
	MAE	0.001583	0.00315	0.002883	0.000683
	MAPE	0.00106	0.003537	0.001932	0.000457
12 months ahead forecast	RMSE	0.001912	0.002692	0.002446	0.001924
	MFE	0.001517	0.0022	0.002008	0.0015
	MAPE	0.001013	0.001471	0.001343	0.001002

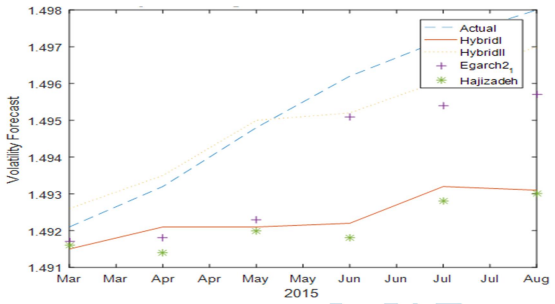

Fig 18: Hybrid Models performance to Volatility Forecasting of Inflation Rate for 6 Months ahead

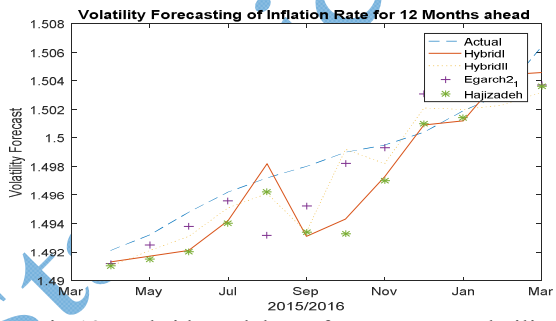

Fig 19: Hybrid Models performance to Volatility Forecasting of Inflation Rate for 12 Months ahead.

IX. Conclusion

This research is based on the application of GARCH models. One of the limitations is that these models produce better results in relatively stable markets and could not capture violent volatilities and fluctuations.

In this research, GARCH family of models have been integrated in to the recurrent dynamic neural networks to form hybrid models which were trained using variants of back propagation training algorithms as opposed to recent work of (Hajizadeh *et al.,* 2012) that also used hybrid models to forecast volatility of S & P 500 index return but, was solely trained with standard steepest descent Back Propagation.

The accuracy-wise comparison of three different gradient based Back Propagation training algorithms, i.e., Levenberg-Marquardt (LM), Scaled Conjugate Gradient Descent (SCGD) and Bayesian Regularization (BR) is investigated. Four types of GARCH family of models have been calibrated and used for forecasting the Inflation Rate based on some macro-economic variables. Then, their performances have been compared to pre-defined measures.

The best model turns out to be EGARCH (2, 1). To enhance the forecasting power of the selected model, two hybrid models have been constructed using Artificial

Professional Statisticians Society

Neural Networks. The inputs to the proposed hybrid models include the volatility estimates obtained by the fitted EGARCH models as well as other explanatory variables. Furthermore, the second hybrid model takes simulated volatility series as extra inputs. Such inputs have been intended to characterize the statistical properties of the volatility series when fed in to Neural Networks.

The results show that though it took longer time and number epochs to train the hybrid models by Bayesian Regularization Algorithms, it gives more accurate predictions than both the Levenberg-Marquadrt and Scaled Conjugate Gradient Descent Algorithms.

The results also demonstrate that the second hybrid model, using simulated volatility series, provides better volatility forecasts. This model significantly improves the forecasts over the ones obtained by the best EGARCH models and Hajizadeh *et al,* (2012).

References

Amaefula, C.G., and Asare,B.K. (2014). The Impacts of Inflation Dynamics and Global Financial Crises on Stock Market Returns and Volatility. Evidence from Nigeria, *Asian Economic and Financial Review*, **4**(5):641-650

Arowolo, W.B., (2013) Predicting Stock Prices Returns using GARCH Model .*The International Journal of Engineering And Science (IJES)*, (2)5: 32-37.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroscedasticity. *Journal of Econometrics*, 31, 307–327.

Chang J.R., Wei L.Y., and Cheng C.H., (2011), A hybrid ANFIS model based on AR and volatility for TAIEX forecasting, *Applied Soft Computing*, 11, 1388–1395.

Cosimano, T.F. and D.W. Jansen (1988) Estimates of the variance of U.S. inflation based upon the ARCH model: comment, *Journal of Money, Credit and Banking,* 20, 409–421.

Dahiru A.B. and Joseph O. A.,(2013), Exchange–Rates Volatility in Nigeria: Application of GARCH Models with Exogenous Break *CBN Journal of Applied Statistics (4)1,89-116*

Dauda O. Y. (2008). Between dollarization and exchange rate volatility: Nigeria's portfolio diversification option, *Journal of Policy Modelling 30, 811–826*

Engle, R.F., (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation, *Econometrica,* 50, 987-1007.

Engle, R.F., (1983). Estimates of the variance of U.S. inflation based on the ARCH model, *Journal of Money, Credit and Banking,* 15, 286–301.

Foresee, F.D. & Hagan, M.T., (1997) "Gauss-Newton approximation to Bayesian regularization", *International Joint Conference on Neural Networks*.

Friedman, M., (1977). Nobel lecture: inflation and unemployment, *Journal of Political Economy,* 85, 451–472

Grier, K.B. and Perry M.J., (2000). The effects of real and nominal uncertainty on inflation and output growth: Some GARCH-M evidence. *Journal of Applied Econometrics*, 15, 45–58.

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

Hagan, M.T., & Menhaj, M., (1994) "Training feed-forward networks with the Marquardt algorithm", IEEE Trans. *Neural Networks*, Vol. 5, No. 6, pp989-993.

Hagan, M. T., Demuth, H. B., & Beale, M. H (1996) *Neural Network Design*, MA: PWS Publishing Boston.

Hebb D. O.(1949), *The Organization of Behavior*, New York: Wiley.

Jesús O. D. and Hagan M. T. (2007). Back Propagation Algorithms for a Broad Class of Dynamic Networks, *IEEE Transactions on Neural Networks,* 18(1), 14-27.

Kristjanpoller W., Fadic A. and Minutolo M. C. (2014). Volatility forecast using hybrid Neural Network models, *Expert Systems with Applications*, 41, 2437–2442.

Mackay, D.J.C., (1992) "Bayesian interpolation", *Neural Computation*, Vol. 4, No. 3, 415- 447.

McCulloch W. and Pitts W., (1943). A logical calculus of the ideas immanent in nervous activity, *Bulletin of Mathematical Biophysics.*, 5, 115–133.

Moller, M.F., (1993) "A scaled conjugate gradient algorithm for fast supervised learning", *Neural Networks,* Vol. 6, pp525-533.

Olowe R. A., (2009). Modelling Naira/Dollar Exchange Rate Volatility: Application of GARCH and Asymmetric Models, *International Review of Business Research Papers 5(3), 377- 398.*

Rosenblatt F.(1958). The perceptron: A probabilistic model for information storage and organization in the brain, *Psychological Review*, 65,386–408.*tructure of Cognition*, Vol. 1, Cambridge, MA: MIT Press.

Professional Statisticians

Rumelhart D. E., Hinton G. E. and. Williams R. J.(1986.). Learning representations by back-propagating errors, *Nature*, 323, 533–536.

Rumelhart D. E. and McClelland J. L.,(1986) eds., *Parallel Distributed Processing: Explorations in the Micros*

Tseng C.H., Cheng S.T., Wang Y.H. and Peng J.T. (2008). Artificial neural network model of the hybrid EGARCH volatility of the Taiwan stock index option prices, *Physica A,* 387, 3192–3200.

Widrow B. and Hoff M. E. (1960) Adaptive switching circuits, *1960 IRE WESCON Convention Record*, New York, 496–104.

Yaya O. S., (2013), Nigerian Stock Index: A Search for Optimal GARCH, *CBN Journal of Applied Statistics,* 4, 69-85.

Professional Statisticians