http://dx.doi.org/10.4314/bajopas.v9i1.4

Bayero Journal of Pure and Applied Sciences, 9(1): 19 - 24
Received: January, 2016
Accepted: May, 2016
ISSN 2006 – 6996

TIME SERIES PREDICTION WITH SIMPLE RECURRENT NEURAL NETWORKS

Abdulkarim, S. A.
Computer Science Department, Federal University Dutse, Jigawa State, Nigeria
(sakwami@live.com) 08037713775

ABSTRACT
Simple recurrent neural networks are widely used in time series prediction. Most researchers and application developers often choose arbitrarily between Elman or Jordan simple recurrent neural networks for their applications. A hybrid of the two called Elman-Jordan (or Multi-recurrent) neural network is also being used. In this study, we evaluated the performance of these neural networks on three established bench mark time series prediction problems. Results from the experiments showed that Jordan neural network performed significantly better than the others. However, the results indicated satisfactory forecasting performance by the other two neural networks.
Key Words: Time Series Prediction, Artificial Neural Network, Recurrent NN, Resilient Propagation.

INTRODUCTION

The task of predicting future values of a time series is a problem that has applications in many fields such as sales, engineering, epidemiology, etc. For efficient planning, accurate and timely prediction of future events is required. A lot of research efforts have gone into the development of prediction models and improvement of their performances.

Artificial Neural Networks (NN) have been used with success in prediction applications, and outperformed classical statistical models such as Auto Regressive Integrated Moving Average (ARIMA)*(*Zhang *et al.,* 2001) (de Almeida and Fishwick 1991). Their salient features are their non-linearity and ability to handle time series without prior knowledge of how the series was generated. Most of the applications of NN are based on Feed forward architecture (i.e. a structure where information flows in only one direction) (Zhang et al. 1998). However, Feedforward neural networks (FNN) were not designed to handle dynamic systems and are therefore limited to handling stationary data. Since practical time series are often dynamic (non-stationary), a NN structure capable of handling dynamic systems is required for effective modeling. Recurrent neural networks (RNNs) are a type of NN designed with feedback connections that allows information to also flow in a backward direction. These connections serve as internal memory for the network. Introduction of this internal memory enables RNN to remember its previous state during processing, thereby giving it the ability to handle dynamic systems. RNN have been used with success in grammar/language processing (Lawrence *et al.*, 2000), gesture recognition (Murakami and Taguchi 1991) and time series applications (Qi and Zhang 2008). The commonly applied RNNs to forecasting are the Elman and Jordan nets, generally referred to as Simple Recurrent Neural Networks (SRNN). A hybrid of Elman and Jordan nets called Multi-Recurrent Neural Networks (MRNN) has also been used in time series prediction (Dorffner, 1996).

Researchers and developers often arbitrarily choose any of the SRNNs for their forecasting applications. To the knowledge of the authors, there is no study that recommends which one to use based on their performance on time series prediction. In this paper, we aim to do that by carrying out an empirical study comparing their performances and also that of MRNN on well-established bench marks.

Since NN are nothing but structure capable of carrying out nonlinear mapping from a set of input patterns to desired target output values, they need to be trained for accurate target output approximation. Back propagation has been the most widely used NN training algorithm. Its variant, Resilient propagation (RPROP) (Riedmiller, 1994) is employed in our work due to its computational simplicity and fast convergence.

In the subsequent sections, we present the background information necessary for the study, the experimental setup, the results and conclusions.

I. BACKGROUND

A. *Elman Neural Network*

Elman Neural Network (Elman NN) (ELMAN, 1991), is a NN structure designed to allow information flow in both forward and backward direction using feedback links in order to deal with temporal properties of a sequential data. As illustrated in Figure I, it has a set of context units referred to as context set that is annexed to the input layer. All the context units are interconnected fully with all units in the hidden layer. Thus, the input vector;

$$X = \underbrace{x_1, x_2 \ldots, x_{N+1}}_{actual\ input}, \underbrace{x_{N+2}, \ldots, x_{N+1+M}}_{context\ units}$$

The hidden layer units are also connected to their corresponding context layer units with weight values of one, such that their outputs or previous states are stored in the context units.

Output of each unit in the output layer is computed as;

$$O_z = f_{O_z}\left(\sum_{m=1}^{M+1} w_{z,m} f_{y_m}\left(\sum_{N=1}^{N+1+M} v_{m,n} x_n\right)\right) \qquad (1)$$

where $[(x]_{N+2}, \ldots, x_{N+1+M}) = (y_1(t-1), \ldots, y_M(t-1))$.

Introduction of context units makes Elman NN capable of performing sequence prediction that is beyond the power of a standard FNN. However, Elman NN cannot really deal with an arbitrarily long history in the data (Bengio, Simard, and Frasconi 1994). Examples of time series applications with Elman NN are [10 - 13].

B. *Jordan Neural Networks*

Jordan Neural Network (Jordan NN) (Jordan 1986), is a model that realizes functional dependency between sequence elements and estimates on one hand and the to-be forecast value on the other (Dorffner 1996). It is very similar to Elman NN except that the context layer stores a copy of the output layer instead of hidden layer(Engelbrecht 2005). Structure of Jordan NN is shown in Fig II. The input layer becomes

$$X = \underbrace{x_1, x_2 \ldots, x_{N+1}}_{actual\ input}, \underbrace{x_{N+2}, \ldots, x_{N+1+Z}}_{context\ units}$$

by annexing the context set. The output units are calculated as

$$O_z = f_{O_z}\left(\sum_{m=1}^{M+1} w_{z,m} f_{y_m}\left(\sum_{n=1}^{N+1+Z} v_{m,n} x_n\right)\right) \qquad (2)$$

where $[(x]_{N+2}, \ldots, x_{N+1+M}) = (O_1(t-1), \ldots, O_M(t-1))$.

Due to its recurrent nature, it can efficiently be applied to time series processing but cannot capture longer term dependency too like Elman NN (Bengio, Simard, and Frasconi 1994). (Yasdi 1999) (Song 2011)(Song 2011)(Song 2011)[15] are examples of its applications to time series forecasting.

C. *Multi-recurrent Neural Networks*

Multi-recurrent Neural Network (MRNN) is obtained from combination of Elman and Jordan NNs. As depicted in Figure III, it has a feedback connection from both hidden and output layer connecting into the context set. The context set subjoins the input layer and interconnects fully to the hidden layer. The input layer becomes

$$X = \underbrace{x_1, \ldots, x_{N+1}}_{actual\ input}, \underbrace{x_{N+2}, \ldots, x_{N+1+M}}_{context\ units}\ \underbrace{x_{N+2}, \ldots, x_{N+1+Z}}_{context\ units}$$

And each output unit is calculated as

$$O_z = f_{O_z}\left(\sum_{m=1}^{M+1} w_{z,m} f_{y_m}\left(\sum_{n=1}^{N+1+M+K} v_{m,n} x_n\right)\right) \qquad (3)$$

Where

$[(x]_1(N+Z,), \ldots, x_1(N+1+M), \quad x_1(N+1+M+2), \ldots, x_1(N+1+N+K)) = (y_1 1\,(t-1), \ldots, y_1 M\,(t-1), (O_1 1\,(t-1), \ldots, O_1 M\,(t-1))$.

MRNN has a larger number of degree-of-freedoms (weights) compared to SRNNs. According to (Dorffner 1996), a number of empirical studies have some versions of MRNN significantly outperform most other simple forecasting methods in real world applications.

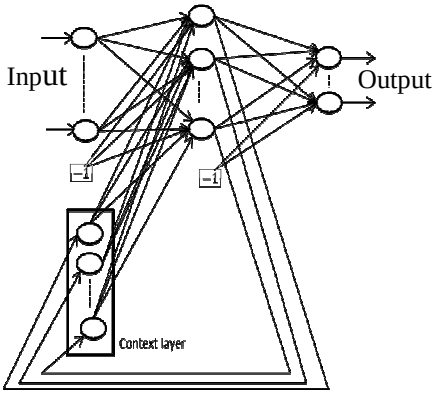

Figure I Elman NN

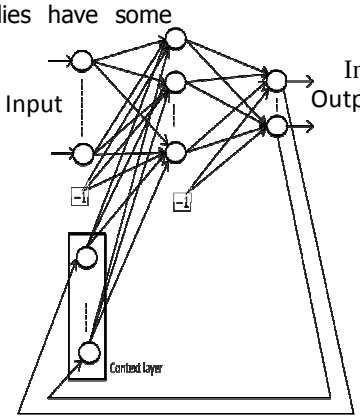

Figure II Jordan NN

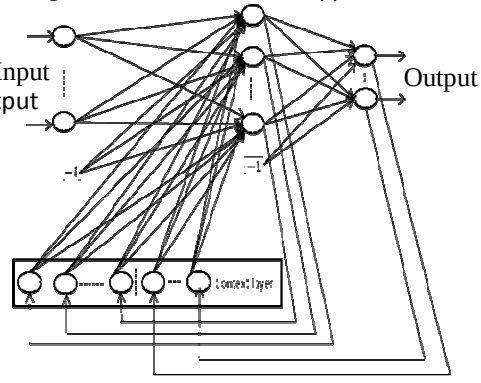

Figure III Multi-Recurrent NN

MATERIALS AND METHODS

In this study, three well-known established benchmark time-series were used to evaluate the performances of the prediction models investigated. The first two series were obtained from online repository at https://datamarket.com/data/list/?q=provider:dstl and the last artificially generated;

1) *International airline time series*; This series has a total of 144 observations of monthly totals of passengers from January, 1949 to December 1960. It follows a multiplicative seasonal pattern with upward trend as shown in Fig I. The dataset is non-stationary due to the presence of strong seasonal variation.
2) *The Quarterly Standard & Poor's 500 (S&P 500) indexes (1900 to 1996)*; It has 388 data points. Plot of the dataset as shown in Figure II revealed a constant trend with long-run cycles.
3) *Mackey Glass chaotic time series*; This data set is a solution of the Mackey-Glass delay-differential equation (Lapedes and Farber 1987) ;

$$\frac{dx}{dt} = \frac{ax(t-\tau)}{1 + x^c(t-\tau)} - bx(t) \qquad (4)$$

using $\tau = 30$, a = 0.2, b = 0.1, c = 10, initial condition x(t) = 0.9 for $0 \leq t \leq \tau$, a 500 points dataset was generated for this study, where 480 data points after the initial transients were used for training and testing. Plot of the series is shown in Figure III;

All datasets were scaled to [-1, 1] and normalized using (Engelbrecht, 2005);

$$x_n' = \frac{x_n}{\sqrt{N}}$$

where N is the number of observations in the dataset. Each of the datasets was divided into two independent subsets in a chronological order, where the first 80% of the dataset was used for training and the last 20% for testing. Note that for parameter optimization purpose only, the training dataset was further partitioned where the first 70% was used for training & the outstanding 30% for validation.

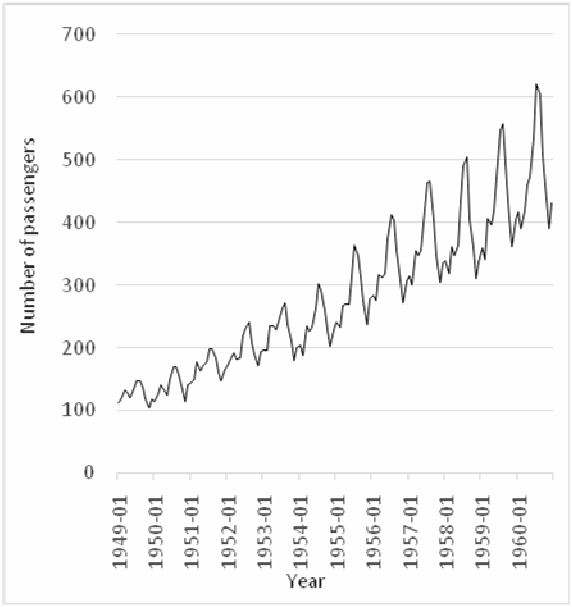

Figure iv: Airline Passengers Time Series

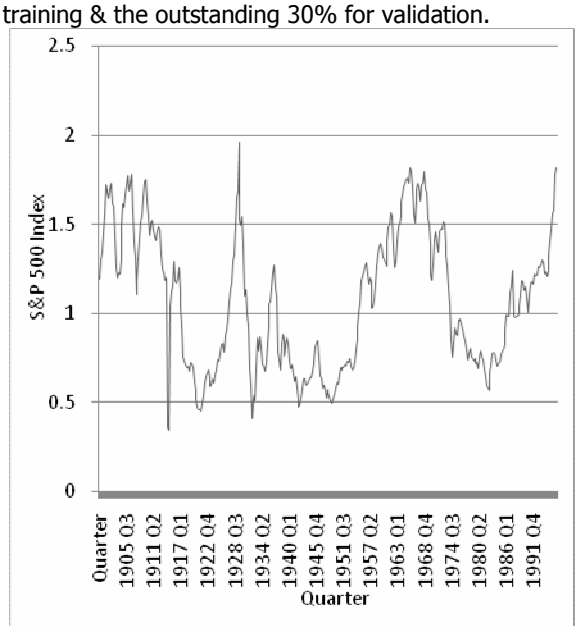

Figure v: Airline Passengers Time Series

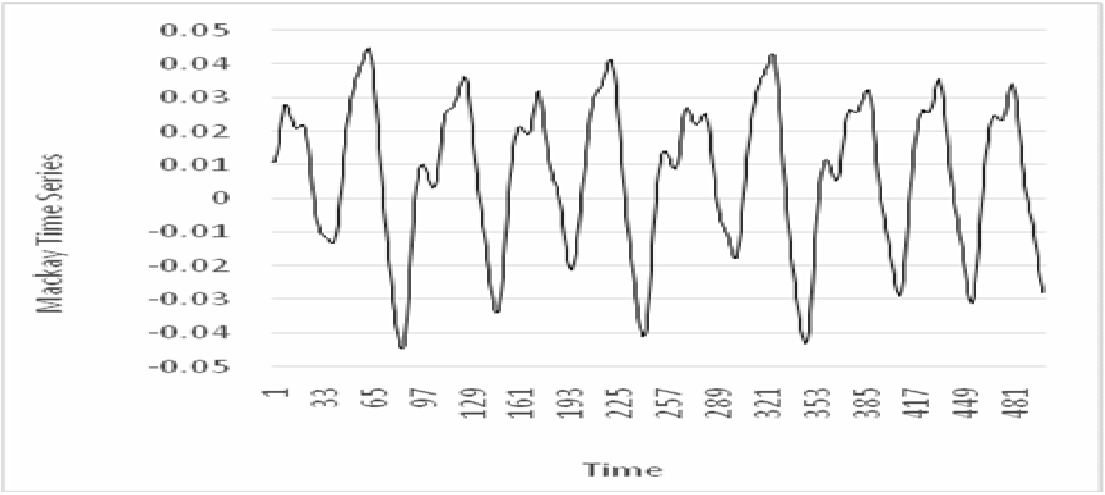

Figure vi: Mackay-Glass Time Series

In all the experiments, one step-ahead prediction horizon was considered. This translates to a single output neuron in all the NN architectures used. For Airline time series, we used 12 input neurons each representing month of a year since the data was collected monthly. For the S&P, we used 4 input neurons each representing quarter of a year since it is quarterly based. Four input neurons were used in Mackay time series prediction, as adopted from previous work of (Larsen *et al.,* 1998). Criteria proposed by (Sheela and Deepa 2013) was used in fixing the number of hidden neurons;

$$N_h = \frac{(4n^2 + 3)}{(n^2 - 8)} \qquad (6)$$

where N_h and n are the number of hidden and input units respectively. The criteria satisfy convergence theorem and has proven to be an effective method as evident in their empirical studies.
For all the NNs, linear activation functions were used in the input layer units. In the hidden and output layer units, modified hyperbolic tangent functions were employed as recommended in (LeCun et al. 2012). It is defined as

$$f(net) = 1.7159 tanh\left(\frac{2}{3} net\right) \qquad (7)$$

Starting values for weights were chosen randomly. To use optimal weight initialization range, we considered ranges {(-0.01, 0.01), (-0.02, 0.02), (-0.03, 0.03), (-0.04, 0.04), (-0.05, 0.05)}. The range that gave us minimum average training and validation error after 30 runs was chosen as optimal.
In the experiments, we used MSE as the performance measure. All experiments were carried out using version 1 of Computational Intelligence Library (CILib) (Cloete *et al.,* 2008)and results reported are averages over 30 simulations, where 1000 iterations was the stopping condition for each algorithm.
Two-tailed non-parametric Mann-Whitney U test was used to statistically determine if the difference in performance were significant. The null hypothesis $H0$: $\mu1 = \mu2$, where $\mu1$ and $\mu2$ are the means of the two samples being compared, were evaluated at a significance level of 95%. $H1$: $\mu1 \neq \mu2$ defined the alternative hypothesis. Thus, any p-value less than 0.05 corresponded to rejection of the null hypothesis that there is no statistically significant difference between the sample means. For the sake of convenience, all p-values were bounded below by 0.0001.

RESULTS AND DISCUSSION

Table 1 presents average training errors (T_E) and generalization errors (G_E) with their confidence interval obtained from predicting the three-time series considered, where minimum values obtained are displayed in italics.
For Airline time series, clearly Jordan NN yields the minimum T_E and G_E, outperforming Elman NN and MRNNS. Elman performed worst compared to all the three RNNs investigated. Mann Whitney test result in table 2 & 3 showed that Jordan NN performance was statistically significant in both training and generalization compared to Elman NN and to MRNN. Also, from table 2 & 3, the p-values suggested significant difference in Elman NN Vs. MRNN performance, with table 1 indicating superiority of Elman NN performance over MRNN.

Table 1: Mean Errors of the Forecasting Models

Prediction problem		Elman	Jordan	Multi-recurrent
Airline	T_E	0.000321±0.000203	*0.000375±0.000108*	0.000503±0.000076
	G_E	0.001066±0.001012	*0.000707±0.000120*	0.001007±0.000187
S&P	T_E	0.000201±0.000097	*0.000134±0.000025*	0.000229±0.000053
	G_E	0.000132±0.000084	*0.000084±0.000027*	0.000139±0.000041
Mackey Glass	T_E	0.000094±0.000073	*0.000104±0.000024*	0.000160±0.000038
	G_E	0.000119±0.000100	*0.000106±0.000028*	0.000199±0.000053

In predicting the S&P time series, Jordan NN outperformed both Elman NN and MRNN by yielding the lowest average errors as shown in table 1. However, Mann Whitney (in Table 2 & 3) indicated that difference in performance between Jordan and Elman NN in predicting the S&P was not statistically significant. Compared to MRNN, Jordan NN performed better at significant level of 95%.

Table 2: Mann-Whitney U p-values obtained for the average training error comparisons on the time series with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95%.

Time Series	Elman Vs Jordan	Elman Vs MRNN	Jordan Vs MRNN
Airline	0.0309	0.0003	0.0444
S&P 500	0.8476	0.0565	0.0136
Mackay-Glass	0.0024	0.0001	0.0493

Table 3: Mann-Whitney U p-values obtained for the average generalization error comparisons on the time series with reference to the null hypothesis that the means of the compared samples are equal at the significance level of 95%.

Time Series	Elman Vs Jordan	Elman Vs MRNN	Jordan Vs MRNN
Airline	0.0289	0.0021	0.0399
S&P 500	0.9411	0.0690	0.0332
Mackay-Glass	0.0082	0.0002	0.0136

In predicting Mackay time series, Jordan NN yielded the lowest average errors (T_E and G_E), outperforming the other RNNS as shown in table 1. MRNN had the worst performance. Mann Whitney (in table 2 & 3) showed that Jordan NN performance was statistically significant compared to both Elman NN and MRNN. Elman NN also performed significantly better than MRNN.

CONCLUSION

The study evaluated the performance of simple recurrent neural networks and the so called Elman-Jordan (multi-recurrent) architectures in time series prediction. Resilient propagation was used to train the RNNs on three established bench mark time series prediction problems. Results from the experiments showed that Jordan NN yielded the lowest average training and generalization errors, outperforming both Elman NN and MRNN in all the three prediction problems. Mann Whitney U test showed that Jordan NN performed significantly better in two out of the three problems. The results also indicated that combining Elman and Jordan architectures (MRNN) does not necessarily improve performance. Based on the datasets studied, we therefore recommend using Jordan NN in time series prediction against Elman NN and MRNN. However, further investigation using more dataset is required.

REFERENCES

de Almeida, C., and P. A. Fishwick. (1991). Time Series Forecasting Using Neural Networks vs. Box- Jenkins Methodology. *Simulation* 57(5): 303–10. http://sim.sagepub.com/content/57/5/303.short

Bengio, Y, P Simard, and P Frasconi. (1994). Learning Long-Term Dependencies with Gradient Descent Is Difficult. *IEEE Transactions On Neural Networks* 5(2): 157–66. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=279181 (November 16, 2015).

Cloete, T, A P Engelbrecht, and Pampará G. (2008). CIlib : A Collaborative Framework for Computational Intelligence Algorithms – Part I. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on IEEE*, , 1750–57.

Dorffner, Georg. (1996). Neural Networks for Time Series Processing. *Neural Network World*.

ELMAN, JEFFREY L. (1991). Distributed Representations, Simple Recurrent Networks, and Grammatical Structure. *Machine learning* 7(2-3): 195–225. http://link.springer.com/article/10.1007/BF00114844 (November 15, 2015).

Engelbrecht, AP. (2005). 2nd Ed. Hoboken, US: John Wiley & Sons, Ltd *Fundamentals of Computational Swarm Intelligence.* http://scholar.google.com/scholar?q=A.P.+Engelbrecht.+Fundamentals+of+Computational+Swarm+Intelligence.+Wiley%2C+2005&btnG=&hl=en&as_sdt=0%2C5#0 (July 21, 2014).

Larsen, J. Svarer, C. Andersen, L. N.and Hansen. L. K. (1998). Adaptive Regularization in Neural Network Modeling." *Neural Networks: Tricks of the Trade. Springer*: 113–32.

Jordan, MI. (1986). "Attractor Dynamics and Parallellism in a Connectionist Sequential Machine." : 531–46. http://www.citeulike.org/group/1778/article/932207 (May 17, 2015).

Lapedes, AS, and RM Farber. 1987. How Neural Nets Work." *Neural information processing systems*. http://papers.nips.cc/paper/59-how-neural-nets-work (May 15, 2014).

Lawrence, S, CL Giles, and S Fong. (2000). "Natural Language Grammatical Inference with Recurrent Neural Networks." *IEEE Transaction on Knowledge and Data Engineering* 12(1): 126–40. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=842255 (December 18, 2015).

Lawrence, Steve (2001). "Noisy Time Series Prediction Using Recurrent Neural Networks and Grammatical Inference." *Machine Learning* 44(1): 161–83. http://link.springer.com/article/10.1023/A:1010884214864 (November 16, 2015).

LeCun, YA, L Bottou, GB Orr, and KR Müller. (2012). "Efficient Backprop. *Neural networks: Tricks of the trade. Springer Berlin Heidelberg*: 9–48. http://link.springer.com/chapter/10.1007/978-3-642-35289-8_3 (October 18, 2015).

Murakami, K, and H Taguchi. (1991). Gesture Recognition Using Recurrent Neural Networks." *In Proceedings of the SIGCHI conference on Human factors in computing systems. ACM*: 237–42. http://dl.acm.org/citation.cfm?id=108900 (December 18, 2015).

Qi, Min, and G Peter Zhang. (2008). Trend Time-Series Modeling and Forecasting with Neural Networks." *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council* 19(5): 808–16.

Riedmiller, Martin. (1994). *Rprop - Description and Implementation Details*.

Sheela, K. Gnana, and S. N. Deepa. (2013). Review on Methods to Fix Number of Hidden Neurons in Neural Networks." *Mathematical Problems in Engineering* 2013: 1–11. http://www.hindawi.com/journals/mpe/2013/425740/.

Song, Qing. 2011. Robust Jordan Network for Nonlinear Time Series Prediction. In *Neural Networks (IJCNN), The 2011 International Joint Conference on IEEE*, , 2542–49. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6033550.

Yasdi, R. (1999). Prediction of Road Traffic Using a Neural Network Approach." *Neural computing & applications* 8(2): 135–42. http://link.springer.com/article/10.1007/s005210050015 (November 16, 2015).

Zhang, G.Peter, B.Eddy Patuwo, and Michael Y. Hu. (2001). A Simulation Study of Artificial Neural Networks for Nonlinear Time-Series Forecasting." *Computers & Operations Research* 28(4): 381–96.

Zhang, G. (1998). Forecasting with Artificial Neural Networks:: The State of the Art." *International journal of forecasting* 14(1): 35–62. http://www.sciencedirect.com/science/article/pii/S0169207097000447 (October 28, 2013).