Composite Recurrent Neural Networks for Long-Term Prediction of Highly-DynamicTime Series Supported by Wavelet Decomposition.

Pilar Gomez-Gil[*], Angel Garcia-Pedrero[*] and Juan Manuel Ramirez-Cortes[**]

[*]Department of Computational Science and [**]Department of Electronics. National Institute of Astrophysics, Optics and Electronics. Luis Enrique Erro No. 1 Tonantzintla, Puebla. 72840. MEXICO.
pgomez@acm.org, agarciapedrero@gmail.com, jmramirez@ieee.org

Abstract. Even though it is known that chaotic time series cannot be accurately predicted, there is a need to forecast its behavior in may decision processes. Therefore several non-linear prediction strategies have been developed, many of them based on soft computing. In this chapter we present a new neural network architecutre, called (Hybrid and based-on-Wavelet-Reconstructions Network (HWRN) which is able to perform long-term prediction, using recursive prediction, over highly dynamic and chaotic time series. HWRN is based on recurrent neural networks embedded in a 2 layer neural structure, and uses as a learning aid signals generated by wavelets coefficients obtained from the training time series. In the results reported here, HWRN was able to predict better than a feed-forward neural network and that a fully recurrent neural network with similar number of nodes. Using the benchmark NN5, which contains chaotic time series, HWRN obtained in average a SMAPE = 26% compared to SMAPE = 61% obtained by a fully-connected recurrent neural network and a SMAPE = 49% obtained by a feed forward network.

1. Introduction

The use of long-term prediction as a tool for complex decision processes involving dynamical systems has been of high interest for researchers in the last years. Some current prediction strategies approximate a model of the unknown dynamical system analyzing information contained in a solely time-series, which is supposed to described the system's behavior. A time series may be defined as an ordered sequence of values observed

from a measurable phenomena: $x_1, x_2 ... x_n$; such observations are sensed at uniform time intervals and may be represented as integer or real numbers [24]. Once defined, an approximated model may be used to predict the trend of the system behavior or to predict as much specific values of the time series as desired. As usual, such model will be just as good as the information used to construct it and as the capability of the modeler to represent important information embedded in the time series being analyzed.

Time series prediction consists on estimating future values of a time series $x_{t+1}, x_{t+2}..$ using past time series values $x_1, x_2 ... x_t$. One-step or short-term prediction occurs when several past values are used to predict the next unknown value of the time series. If no exogenous variables are considered, one-step prediction may be defined as [18]:

$$\bar{x}_{t+1} = \phi(x_{t-1}, x_{t-2} ... x_{t-p}) \quad (1)$$

where ϕ is a approximation function used to predict. Similarly, long term prediction may be defined as:

$$\bar{x}_{t+h} ... \bar{x}_{t+2}, \bar{x}_{t+1} = \boldsymbol{\varphi}(x_{t-1}, x_{t-2} ... x_{t-p}) \quad (2)$$

where *h* denotes the prediction time horizon, that is, the number of future values to be obtained by the predictor at once. Long term prediction may also be achieved by recursive prediction, which consists of recursively using equation (1) by feeding back past predicted values to the predictor to calculate the new ones.

The construction of models able to predict highly nonlinear or chaotic time series is of particular interest in this research. A chaotic time series is non-stationary, extremely sensitive to initial conditions of the system and contains positive Lyapunov Exponents [14]. It is claimed that chaotic time series may only be short-term predicted [19]. Even though, in some cases it is possible to approximate a dynamical model with similar characteristics to that found in the non-linear time series and to use it for long-term prediction. There are many techniques used to build predictors; they may be linear or non-linear, statistical or based on computational or artificial intelligence. For example, ARMA, ARIMA and Kalman filters are linear methods [20]; k-nearest neighbors, genetic algorithms and artificial neural networks are examples of non-linear methods. Only non-linear methods are useful to forecast non-linear time series.

The use of fully-connected, recurrent neural networks for long-term prediction of highly-dynamical or chaotic time series has been deeply studied [22]. In spite of the powerful capabilities of these models to represent dynamical systems, their practical use is still limited, due to constraints found in defining an optimal number of hidden nodes for the network and

the long time required to train such networks. As a way to tackle these problems, complex architectures with a reduced number of connections, better learning abilities and special training strategies have been developed [13]. Examples of such works are found at [2,3,4,9,10,12,14] among others.

In this chapter we present a novel neural prediction system called HWRN (**H**ybrid and based-on-**W**avelet-**R**econstructions **N**etwork). HWRN is based on recurrent neural networks, inspired at the Hybrid complex neural network [14] and with a particular kind of architecture and training scheme supported by wavelet decomposition. In the experiments reported here, HWRN was able to learn and predict as far as 56 points of two highly-dynamical time series, obtaining better performance than a fully-connected recurrent neural network and a three-layer, feed-forward neural network with similar number of nodes than the HWRN. This chapter is organized as follows: section two describes the main characteristics, general structure and training scheme of the model. In the same section some details are given related to reconstruction of some signals that are used for supporting training, which is based on discrete wavelet transforms. Criteria used to evaluate the performance of the system are presented at section three. Section four describes the experiments performed and their results; it also includes a description of the time series used to evaluate the model. Last section presents some conclusions and ongoing work.

2. Model Description

HWRN is built using several small, fully-connected, recurrent neural networks (SRNN) attached to a recurrent layer and an output layer. Figure 1 shows the general architecture of HWRN. The SRNN are used to learn signals obtained from the training time series that contain different frequency-time information. Outputs of the SRNN are fed to a recurrent layer, which is able to memorize time information of the dynamical system. The last layer acts as a function approximator builder.

The output of each node *i* at HWRN and SRNN is defined as:

$$\frac{dy_i}{dt} = -y_i + \sigma(x_i) + I_i \tag{3}$$

where:

$$x_i = \sum_{j=1}^{m} y_j w_{ji} \tag{4}$$

represents the inputs to the *i-th* neuron coming from other *m* neurons,

I_i is an external input to *i-th* neuron,

w_{ji} is the weight connecting neuron *i* to neuron *j*,

$\sigma(x)$ is the node's transfer function; it is a sigmoid for all layers except output layer, for which transfer function is linear.

In order to be solved, equation 3 may be approximated as [25]:

$$y_i(t+\Delta t) = (1-\Delta t)y_i(t) + \Delta t\sigma(x_i(t)) + \Delta t I_i(t) \quad (5)$$

for a small Δt, where:

$$x_i(t) = \sum_{j=1}^{m} y_j(t) w_{ji} \quad (6)$$

For the results reported here, initial conditions of each node $y_i(t=0)$, are set as small random values. Indeed, there are no external inputs to nodes, that is $I_i(t)$ for all *i*, all *t*.

Training of a HWRN predictor contains three main phases:

1. Pre-processing of the training time series and generation of reconstructed signals,
2. Training of the SRNN,
3. Training of the HWRN.

After being trained, HWRN receives as input *k* past values of a scaled time series, then recurrent prediction is applied to obtain as many futures values as required. Each training phase is described next.

2.1 Phase 1: Preprocessing

HWRN requires a time series with enough information of the dynamical behavior in order to be trained. Such time series may contain integer or real values and the magnitude of each element must be scaled to the interval [0,1]. This is required in order to use sigmoid transfer functions for the nodes in the network. To achieve this, the time series may be normalized or linearly scaled; in this research a linear scale transformation was applied, as recommended for financial time series by [6]. The linear transformation is defined as:

$$z_t = lb + \frac{x_t - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})}(ub - lb) \quad (7)$$

where:

ub is the desired upper bound; in this case *ub* = 1,
lb is the desired lower bound; in this case *lb* = 0,
max(**x**) is the maximum value found at the time series,
min(**x**) is the minimum value found at the time series.

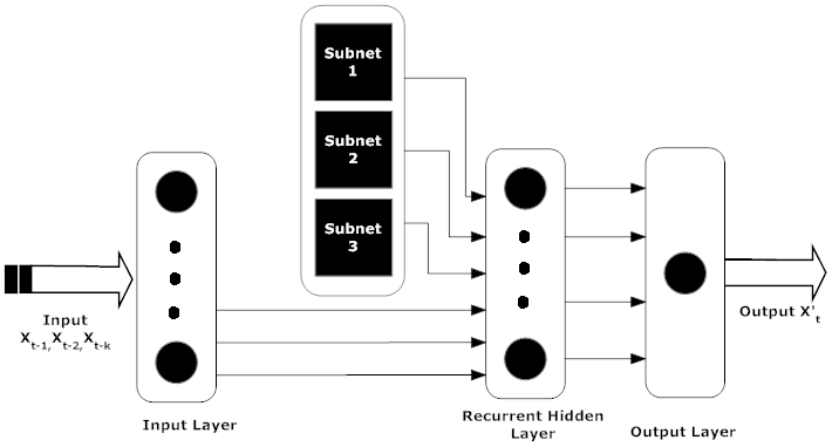

Fig. 1. A **H**ybrid and based-on-**W**avelet-**R**econstructions Network HWRN (adapted from [11])

If the original times series has missing values, they are approximated as the mean of their two nearest neighbors. No further processing is applied.

An important challenge forecasting nonlinear and chaotic time series is the complexity found to represent its non-stationary characteristics. To tackle this, the HWRN learns frequency information related to different times using different components. It is known that wavelet analysis has been used to represent local frequency information in a signal. Such analysis calculates the correlation among a signal and a function $\psi(.)$, called wavelet function. Similarity among both functions is calculated for different time intervals, getting a two dimensional representation: time and frequency [1]. In this work, a multi-scale decomposition of the training signal is performed using the sub-band coding algorithm of the Discrete Wavelet Transform [21]. This algorithm uses a filter bank to analyze a discrete signal *x(t)*. This bank is made of low-pass L(z) and high-pass H(z) filters, separating frequency content of the input signal in spectral bands of

equal width. Figure 2 shows a one-level filter bank. After performing a down-sampling with a factor of two, signals cA(t) and cD(t) are obtained. These signals are known as approximation and detail coefficients, respectively. This process may be executed iteratively forming a wavelet decomposition tree up to any desired resolution level. A three-level decomposition wavelet tree, used for the experiments presented in this paper, is shown in Figure 3. The original signal *x(t)* may be reconstructed back using the Inverse Discrete Wavelet Transform (iDWT), adding up the outputs of synthesis filters. Similarly it is possible to reconstruct not only the original signal, but also approximation signals that contain low-frequency information of the original signal and therefore more information about long-term behavior. In the same way, detail signals can be reconstructed; they contain information about short-term changes in the original signal. Using the decomposition wavelet tree at figure 3, four different signals may be reconstructed (one approximation and three detail signals) using the coefficients shown at the leaves of such tree. For the rest of this chapter, these signals are referred as "reconstructed signals."

For example, figure 4(a) shows a chaotic time series called NN5-101 (see section 4); figure 4(b) shows its most general approximation obtained using coefficients cA_3 (see figure 3); figure 4(c) shows the most general detail signal obtained using coefficients cD_3; figure 4(d) shows detail signal at level 2 obtained using coefficients cD_2; figure 4(e) shows detail signal at maximum level obtained using coefficients cD_1.

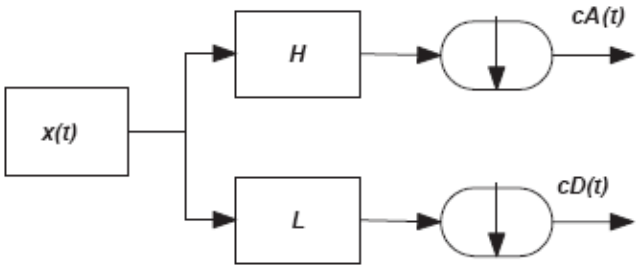

Fig. 2. An analysis filter bank

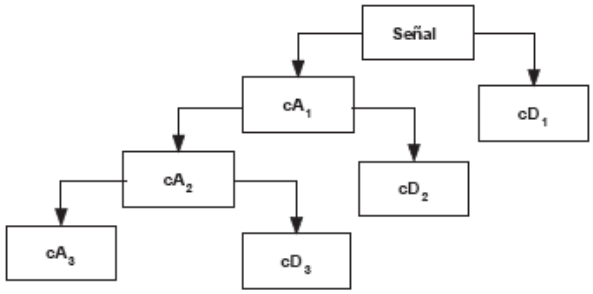

Fig. 3. A three-level decomposition wavelet tree

During the predictor training, a set of these reconstructed signals is selected and independently learned by a set of SRNN. In order to figure out which reconstructed signals contain the most important information, all possible combinations of reconstructed signals are created; next, signals in each combination are added up and the result is compared with the original signal using Mean Square Error (see equation 8). The reconstructed signals in the combination with the smallest MSE are selected to be learnt by the SRNN.

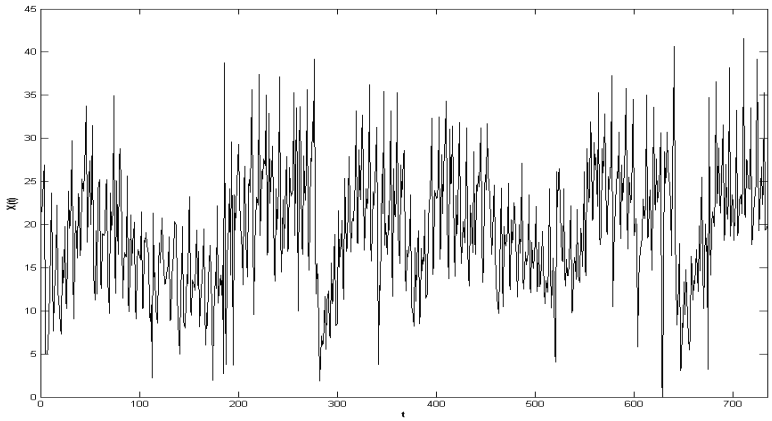

Fig. 4 (a). Original signal NN5-101 (data taken from [7])

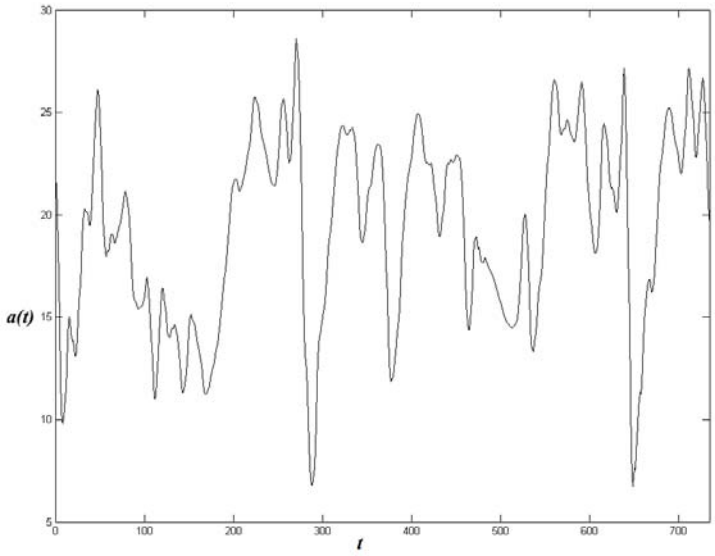

Fig. 4 (b). Most general approximation signal obtained from NN5-101

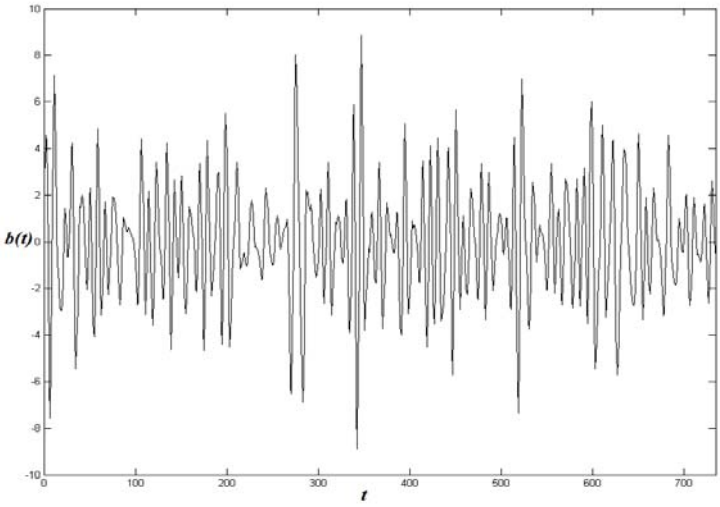

Fig. 4 (c). Most general detail signal obtained from NN5-101

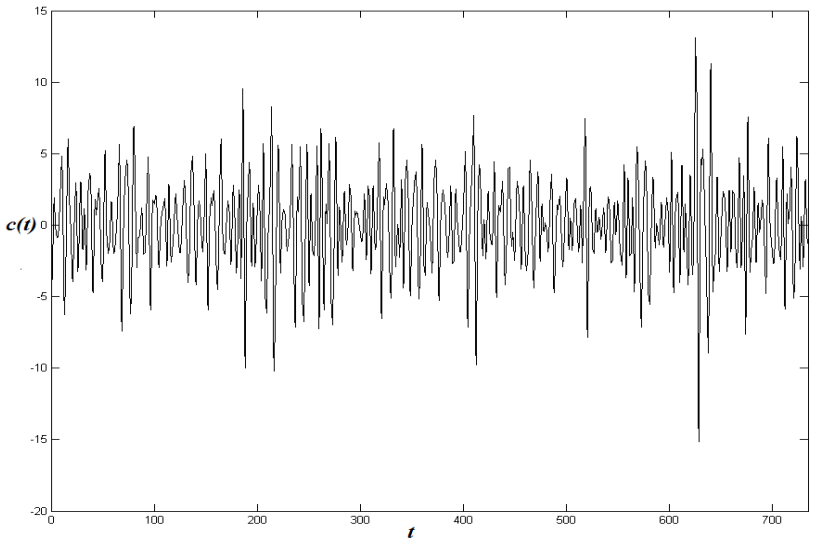

Fig. 4 (d). Detail signal at level 2 obtained from NN5-101

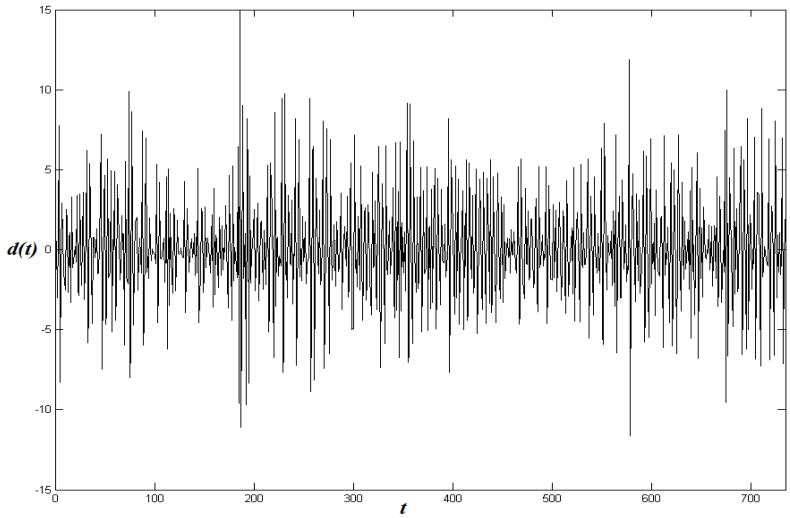

Fig. 4 (e). Detail signal at maximum level obtained from NN5-101

2.2 Phase 2: Training the SRNN

SRNN are trained to predict one point in each selected reconstructed signal; they receive as input *k* values of the corresponding reconstructed signal and predict the next one. Once trained, the SRNN require only the first *k* values of the reconstructed signal; the rest values are generated using recursive prediction as long as the predictor works. These *k* values are stored as free parameters of the system, to use them when prediction of the time series is taking place.

Training of all SRNN is performed using the algorithm *"Real-time real-learning based on extended Kalman filter (RTRL-EKF)"* [15]. This algorithm contains 2 parts: gradient estimation and weights adjustment. The first part is done using the Real-Time, Real-Learning Algorithm proposed by Williams and Zipser [27]; second part is done using an extended Kalman Filter. RTRL-EKF has a complexity of $O(n^4)$, where *n* is the number of neurons in the neural network [11].

2.3 Phase 3: Training the HWRN

After training all SRNN, their weights are imbedded in the architecture of the HWRN (see figure 1) which also contains a hidden layer with recurrent connections and an output layer with feed-forward connections. The complete architecture is trained to predict one point of the original signal, keeping fixed the weights of sub-networks SRNN. As in the case of SRNN, training is performed using *"Real-time real-learning based on extended Kalman filter (RTRLEKF) algorithm"* [15]

3. Metrics for Performance Evaluation

The prediction ability of the proposed architecture and comparative models was measured using three metrics: Mean Square Error (MSE), Symmetrical-Mean Absolute Percentage Error (SMAPE) and Mean Absolute Scaled Error (MASE). Next each metric is explained.

"Mean Square Error" is defined as:

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(x_t - \hat{x}_t)^2 \tag{8}$$

The "Symmetrical-Mean Absolute Percentage Error" is scale-independent; therefore it is frequently used to compare performances when

different time series are involved [16]. This is the official metric used by the "NN5 forecasting competition for artificial neural networks & computational Intelligence" [7]. SMAPE is defined as:

$$SMAPE = \frac{1}{n}\sum_{t=1}^{n}\left(\frac{|x_t - \hat{x}_t|}{(x_t + \hat{x}_t)/2}\right)(100\%) \tag{9}$$

It is important to point out that SMAPE cannot be applied over time series with negative values.

Other popular metric is the "Mean Absolute Scaled Error," defined as:

$$MASE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{x_t - \hat{x}_t}{\frac{1}{n-1}\sum_{i=2}^{n}|x_i - x_{i-1}|}\right| \tag{10}$$

where x_t is the original time series and $\hat{x}_t$ is the predicted time series.

4. Experiments and Results

The proposed architecture and training scheme were tested using two benchmark time series; they are:

a) The time series generated by Matlab function *sumsin()*, available at version 7.4 and commonly used in Matlab demos [23]. It is defined as:

$$s(t) = \sin(3t) + \sin(0.3t) + \sin(0.03t) \tag{11}$$

Figure 5 shows an example of 735 points of *sumsin()* time series.

b) Eleven of the time series found in the database of the "NN5 Forecasting Competition for Artificial Neural Networks and Computational Intelligence" [7]. These time-series correspond to cash drawbacks occurred daily in teller machines at England from 1996 to 1998; these series may be stationary, have local tendencies or contain zeroes or missing values. Figure 4 (a) shows the first time-series of such database, identified as "NN5-101". The eleven time-series used here correspond to what is called the "reduced set" in such competition. In order to determine if these series were chaotic, the maximum Lyapunov Exponent (LE) of each one was calculated using the method proposed by Sano and Sawada [17]. Table 1 shows the maximum LE of each time series; notice that all are positive, an indication of chaos.

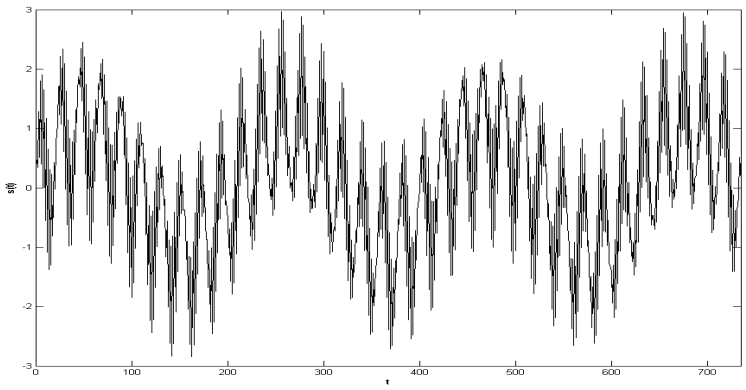

Fig. 5. 735 points of the time series *sumsin()*

Table 1. Maximum LE of reduced set series NN5

Series ID	Maximum LE
NN5-101	0.0267
NN5-102	0.6007
NN5-103	0.0378
NN5-104	0.0565
NN5-105	0.0486
NN5-106	0.0612
NN5-107	0.0678
NN5-108	0.0384
NN5-109	0.8405
NN5-110	0.0621
NN5-111	0.0220

The HWRN contains 3 SRNN; the number of nodes at each SRNN was from 6 to 10, determined experimentally depending upon the reconstructed signal being learnt; the hidden layer has 10 nodes. The performance of HWRN was compared with a three layer, feed-forward neural network (5-26-1) and a fully-connected recurrent neural network with 5 input nodes, 26 hidden nodes and one output node. These architectures have a similar number of nodes as the HWRN. All architectures receive as input 5 values ($k = 5$) of the time series and predict next value. Recurrent prediction is used to generate 56 futures values, following rules of the "NN5 forecasting

competition for artificial neural networks & computational Intelligence" [7]. The architecture was implemented using Matlab V7.4, C++, and public libraries for the training algorithm available at [5]. For both cases, four reconstructed signals were generated using DWT with wavelet function Daubechies '*db10*' available at Matlab. Three of the reconstructed signals were selected using the strategy described at section 2.1.

Twelve experiments were executed for each time series and each neural model. For each experiment, a different random initial set of weights was used. All trainings were made of 300 epochs. The first 635 values of each series were employed to train all models and the next 56 values were used as a testing set to compare the performance of the proposed architecture with respect to the other two models. The last 56 values of the series were used as a validation set in order to compare the performance of this architecture with respect to the competition results published by [8].

Table 2 shows the results obtained using recursive prediction of 56 values (validation set) by the 12 experiments over series *sumsim();* the metric MAPE is not shown because it is not valid for negative values, as is with *sumsin()*. Figure 6 plots 56 predicted values (validation set) of series NN5-109, which was the series at NN5 dataset that obtained the best prediction results, with a SMAPE = 20.6%. Figure 7 plots 56 predicted values (validation set) of series NN5-107, which was the worst case obtained with series NN5, with a SMAPE = 40.5%.

Table 3 summarizes the average results obtained for the two cases, all experiments, all architectures predicting the validation set. For the three metrics in the two tested cases, HWRN got, in average, better results than the feed-forward and the fully-connected recurrent architectures., HWRN got a average SMAPE of 54% for the *sumsinn()* time series and 27% for the NN5 time series. It is important to point out that, with respect to contest results published by [8] using NN5 reduced test, HWRN could be located between the 16th and 17th place in the category of "neural networks and computational intelligence methods."

Notice at table 3 the high Standard Deviation found in the performance measured by MASE for the three architectures. This may be due to the facts that these series are chaotic (see table 1), and that the ability of the learning algorithm RTRLEKF to find the best solution space depend, among other factors, upon the initial set of weights randomly generated. However, it may be noticed that HWRN got the smallest Standard Deviation for these cases.

Table 2. Twelve experiments predicting validation set over series *sumsin().* For a definition of MSA and MASE see equations (8) and (10)

Experiment Number	Feed-forward network		Recurrent Network		HWRN	
	MSE	**MASE**	**MSE**	**MASE**	**MSE**	**MASE**
1	0.112	60.827	0.236	100.004	0.110	73.184
2	0.089	49.823	0.083	56.638	0.092	66.314
3	0.070	47.931	0.036	40.038	0.370	41.687
4	0.099	58.184	0.191	86.501	0.034	41.807
5	0.061	48.613	0.023	34.78	0.029	37.863
6	0.165	80.531	0.090	53.566	0.040	43.689
7	0.096	59.478	0.003	12.399	0.132	78.450
8	0.049	49.816	0.521	107.336	0.052	49.518
9	0.104	74.631	0.146	78.909	0.054	49.677
10	0.063	55.017	0.064	53.904	0.116	67.361
11	0.063	50.018	0.087	73.610	0.099	68.531
12	0.160	84.994	0.086	61.378	0.021	32.832
Mean	0.094	59.989	0.131	63.255	0.068	54.243
St. deviation	0.038	13.044	0.140	27.553	0.039	15.551

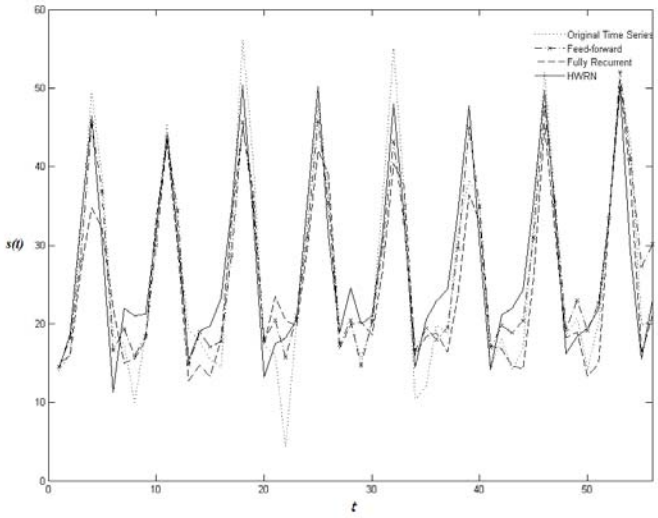

Fig. 6. Best Prediction Case using NN5, SMAPE = 20.6%, series NN5-109

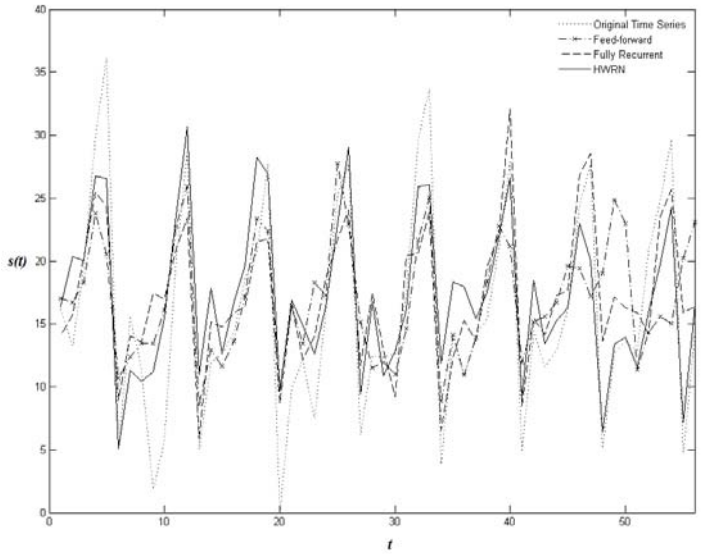

Fig. 7. Worst prediction case using NN5, SMAPE = 40.5%, series NN5-109

Table 3. Prediction errors obtained by the proposed architecture and two other architectures using 56 values ahead.

Time Series	Metric	Neural Architecture		
		Feed-forward	**Recurrent**	**HWRN**
sumsin()	MSE	0.09 ± 0.04	0.13 ± 0.14	**0.07 ± 0.04**
	MASE	59.99 ± 13.04	63.25 ± 27.55	**54.24 ± 15.55**
Eleven examples of NN5 time series	MSE	250.12 ± 226.05	198.69 ± 131.12	**34.05 ± 20.12**
	SMAPE	49.28% ± 12.36	60.75% ± 13.05	**27.22% ± 8.27**
	MASE	517.50 ± 1,079.68	546.31 ± 1,218.95	**194.99 ± 387.22**

5. Conclusions

We presented a novel neural network predictor, called HWRN, based on a combination of small, fully-connected recurrent sub-networks, called SRNN, that are embedded in a composite neural system. This system is

able to generate as many future values as desired using recursive prediction. HWRN was able to predict up to 56 points ahead of several non-linear time series, as shown by experiments done using the time series generated by Matlab's function *sumsin()* and the time series found at the reduced set of the "NN5 Forecasting Competition for Artificial Neural Networks and Computational Intelligence" [7]. The SRNN's are trained to reproduce selected reconstructed signals that represent different frequencies at different times of the original one. The reconstructed signals are obtained using the Discrete Wavelet Transform and the Inverse Discrete Wavelet Transform [1]. In average, the HWRN obtained a Symmetrical-Mean Absolute Percentage Error (SMAPE) of 27% when predicting in a recursive way 56 points ahead of 11 chaotic NN5 time series. This performance was better than the obtained with a fully-connected recurrent neural network (SMAPE= 61%) and a feed-forward network (SMAPE = 49%), both with similar number of nodes and weights. The main drawback of this system is the time required to train it. Currently our research group is looking for ways to train this system faster and for a efficient method to select the reconstructed signals generated by the iDWT.

References

1. Addison P.S. : The Illustrated Wavelet Transform Handbook: Introductory Theory and Applications in Science, Engineering, Medicine and Finance. IOP Publishing, UK (2002)
2. Alarcon-Aquino V., Garcia-Treviño E. S., Rosas-Romero R., y Ramirez-Cruz J. F.: Learning and approximation of chaotic time series using wavelet networks. In: Proceedings of the Sixth Mexican International Conference on Computer Science ENC 2005, pp. 182–188 (2005)
3. Beliaev I. y Kozma R.: Time series prediction using chaotic neural networks on the cats benchmark. Neurocomputing, 70(13-15), 2426–2439 (2007).
4. Cai X., Zhang N., Venayagamoorthy G. K., y Wunsch II D. C. : Time series prediction with recurrent neural networks trained by a hybrid PSO-EA algorithm. Neurocomputing, 70(13-15), 2342–2353 (2007)
5. Cernansky M. Michal Cernansky's homepage downloads. Available at http://www2.fiit.stuba.sk/~cernans/main/download.html. Last accessed January 2009 (2008)
6. Crone S.F., Guajardo J., and Weber R.: The impact of preprocessing on support vector regression and neural networks in time series prediction. In: Proceedings of the 2006 International Conference on Data Mining, DMIN 2006, pp. 37-44 (2006)

7. Crone S.F.: NN5 forecasting competition for artificial neural networks & computational Intelligence." Available at: http://www.neural-corecasting-competition.com/. Last consulted at March 2009 (2008)
8. Crone S. F. : NN5 forecasting competition results. Available at: http://www.neural-forecasting-competition.com/NN5/results.htm. Last consulted at July 2009 (2009)
9. Espinoza M., Suykens J., De Moor B.: Short term chaotic time series prediction using symmetric LS-SVM regression. In: Proceedings of the 2005 International Symposium on Nonlinear Theory and Applications (NOLTA), pp. 606–609 (2005)
10. Gao H., Sollacher R., y Kriegel H.P.: Spiral recurrent neural network for online learning. In: Proceedings of the European Symposium on Artificial Neural Networks, pp. 483–488 (2007)
11. García-Pedrero, A. *Arquitectura Neuronal Apoyada en Señales Reconstruidas con Wavelets para predicción de Series de Tiempo Caóticas* (A neural architecture supported by wavelet's reconstructed signals for chaotic time series prediction). Master Thesis (in Spanish), Computational Department, National Institute of Astrophysics, Optics and Electronics (2009)
12. García-Treviño E.S., Alarcon-Aquino V.: Chaotic time series approximation using iterative wavelet-networks. In: Proceedings of the 16th International Conference on Electronics, Communications and Computers CONIELECOMP 2006, IEEE Computer Society pp. 19-24 (2006)
13. Gomez-Gil, P.: Long Term Prediction, Chaos and Artificial Neural Networks. Where is the meeting point? Engineering Letters 15(1), available at: http://www.engineeringletters.com/issues_v15/issue_1/EL_15_1_10.pdf (2007)
14. Gomez-Gil, P., Ramirez-Cortes, M.: Experiments with a Hybrid-Complex Neural Networks for Long Term Prediction of Electrocardiograms. In: Proceedings of the IEEE 2006 International World Congress of Computational Intelligence, IJCNN 2006. DOI: 10.1109/IJCNN.2006.246952 (2006)
15. Haykin S.: Neural Networks: A Comprehensive Foundation. Macmillan College Publishing Company, New York (1994)
16. Hyndman R. J.: Another look at forecast accuracy metrics for intermittent demand. Foresight: The International Journal of Applied Forecasting, 4, 43–46 (2006)
17. Kantz H., Schreiber T.: Nonlinear Time Series Analysis. Cambridge University Press New York (2003)
18. Lendasse A., Wertz V., Simon G., Verleysen M.: Fast Bootstrap applied to LS-SVM for Long Term Prediction of Time Series. In: Proceedings of the 2004 International Joint Conference on Neural Networks, IJCNN 2004, pp. 705–710 (2004)
19. Lillekjendlie B., Kugiumtzis D., y Christophersen N.: Chaotic time series part II: System identification and prediction. *Modeling, Identification and Control*, 15(4) 225–243 (1994).

20. Makridakis S. G.,Wheelwright S. C., McGee V. E.: Forecasting: Methods and Applications. John Wiley & Sons, Inc., New York (1983)
21. Mallat S. G.: A theory for multiresolution signal decomposition: The wavelet representation. IEEE Transactions on Pattern Analysis and Machine Intelligence 11(7), 674– 693 (1989)
22. Mandic D. P., Chambers J. Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability. John Wiley & Sons NewYork (2001)
23. Misiti M, Misii Y., Oppenheim G., Poggi J. M.: Wavelet Toolbox 4 User's Guide. The MathWorks, Edition 4.3 (2008)
24. Palit A. K. Popovic D.: Computational Intelligence in Time Series Forecasting: Theory and Engineering Applications. Advances in Industrial Control, Springer-Verlag New York, Inc., Secaucus (2005)
25. Pearlmutter, B. A.: Dynamic Recurrent Neural Networks. Technical Report CMU-CS-90-196, School of Computer Science, Carnegie Mellon University, Pittsburgh (1990)
26. Sarmiento H. O., Villa W. M.: Artificial intelligence in forecasting demands for electricity: an application in optimization of energy resources. *Revista Colombiana de Tecnologías de Avanzada* 2(12), 94–100 (2008)
27. Williams R. J. y Zipser D.: A learning algorithm for continually running fully recurrent neural networks. Neural Computing 1(2), 270–280 (1989)