Prediction of Chaotic Time-Series with Different MLE Values using FPGA-based ANNs

[1]A.D. Pano-Azucena, [1,2]E. Tlelo-Cuautle, [2]L.G. de la Fraga, [3]C. Sanchez-Lopez
[1]J.J. Rangel-Magdaleno, [4]Sheldon X.-D. Tan
[1]INAOE, Puebla, Mexico
[2]CINVESTAV, Mexico City, Mexico
[3]Universidad Autónoma de Tlaxcala, Mexico
[4]University of California at Riverside, USA
Email: ana.dalia.p.a@gmail.com, etlelo@inaoep.mx

***Abstract*—Chaotic time series can be generated from different kinds of chaotic oscillators in different dimensions and directions. However, their prediction becomes a challenge when they have different values of their maximum Lyapunov exponent (MLE), which is associated to the degree of unpredictability of a chaotic system. In this manner, we highlight how an artificial neural network (ANN) can be used to predict chaotic time series with different MLE value. The cases of study are three chaotic time series with different MLE values, which are predicted by an ANN that is validated through measuring the mean square error (MSE) and root MSE (RMSE). We provide design and training details of a 5-layers ANN, and its implementation using field-programmable gate arrays. Finally, the ANN is validated with the prediction results that are compared to the original chaotic time series according to their MSE and RMSE.**

I. Introduction

Chaotic oscillators generate chaotic time series that can be highly unpredictable, and which unpredictability can be quantified by evaluating their maximum Lyapunov exponent (MLE) [1]. Chaotic time series having high MLE values are more complex than those ones having low MLE values and therefore, when performing time series prediction of this kind of signals, it becomes quite difficult to perform a long-term prediction. That way, we show how to perform time series prediction of chaotic signals with different MLE by using an artificial neural network (ANN) that is implemented in a field-programmable gate array (FPGA).

Time series prediction has the challenge of determining the number of data samples to perform a long-term prediction [2], with low mean square error (MSE) and root MSE (RMSE). On the one hand, some approaches to tackle time series prediction are for example: ANNs, evolutionary algorithms, fuzzy logic, expert systems, support vector machines, hybrid method, and so on [3, 4]. On the other hand, among those approaches we use ANNs taking advantage of their ability to approximate any nonlinear function universally. In addition, ANNs are a new generation of information processing systems that are deliberately constructed to make use of some of the organizational principles that characterize the human brain [5]. ANNs are characterized by fault tolerance, adaptivity, generalization, etc. Due to these characteristics, an ANN is suitable for time series prediction [3], and we implement it by using an FPGA due to its advantages for fast prototyping, configurability, reprogrammability, low development and acquisition costs.

In this work we highlight general aspects of the ANN to perform the prediction of three chaotic time series with different MLE values. Besides, one must be aware that at the present time, no one provided a general solution for implementing an ANN yet, so that the following problems remain open: selection of hidden neurons, training algorithm, and architecture. For instance, hidden neurons influence the error on the nodes to which their output is connected, and the accuracy of training is determined by the architecture, number of hidden neurons in hidden layer, kind of activation function, inputs, and updating of weights. In this manner: Sect. II describes the generation of chaotic time series with different MLE. Sect. III describes the training of the ANN topology from [6], which is implemented herein into an FPGA [7], we also describe the activation functions, learning rules, and updating of weights. The experimental results are listed in Sect. IV. Finally, Sect. V concludes the paper.

II. Generating chaotic time series

A time series represents a measure of a physical variable x_t registered at a time t, and it is discrete. The observed data can be used to model or to predict future values to determine the behavior of a time series [8]. Herein, we generate chaotic time series by implementing into an FPGA the chaotic oscillator described by (1) [7], where x, y and z are state variables, and a, b, c, d_1 are real and positive coefficients that can have values between the interval [0,1]. The nonlinear function $g(x)$ can be approached by piecewise-linear (PWL) functions [1].

$$\begin{aligned} \dot{x} &= y \\ \dot{y} &= z \\ \dot{z} &= -ax - by - cz + d_1 g(x) \end{aligned} \tag{1}$$

The solution of (1) is obtained by numerical methods that require different number of FPGA resources. An important issue is that by varying the characteristics of the PWL function $g(x)$ in (1), and/or the coefficient values, the time series have different MLE values, and in addition, one can generate as many number of scrolls as they can be implemented into an FPGA [7]. From this issue, MLE provides a measure to determine unpredictability in a dynamical system. For instance, the MLE can be optimized by applying evolutionary algorithms [1], and by varying a, b, c, d_1 in (1), where the search space for each coefficient can be between 0.0001 to 1.0000, and if $g(x)$

978-1-5090-5052-9/17/$31.00 ©2017 IEEE

is modeled by break points α, saturation levels k, and slopes $s(x)$, then Table I lists three different MLE values for the same double-scroll chaotic attractor. We obtained the experimental data of those chaotic time series with different MLE values from an FPGA-based implementation and used the data to train the ANN shown in the next Section, and then to perform chaotic time series prediction.

TABLE I: Optimized coefficient values for generating a double-scroll chaotic attractor from (1).

Slope s(x)	Break points α	Coefficient (a, b, c, d_1)	MLE
60.606	-0.0165, 0.0165	1.0000, 1.0000, 0.4997, 1.0000	0.3761
60.606	-0.0165, 0.0165	1.0000, 0.7000, 0.7000, 0.2542	0.3425
60.606	-0.0165, 0.0165	0.7000, 0.7000, 0.7000, 0.7000	0.2658

III. ANN DESIGN ISSUES AND TRAINING

An ANN has elementary processing units called neurons or nodes whose processing capability is stored in the connections by synaptic weights, and whose adaptation depends on learning [9]. Basically, there are three kinds of neurons: input (that are used to allocate input values), hidden (that are used to perform operations and consists of one or more layers), and output ones (that are used to perform operations and to compare the values with target or reference ones). Figure 1 shows the basic neuron structure, where x_j represents the input signals, w represents the synaptic weights (if w is positive it is associated to an excitation, if it is negative to an inhibition.), b is the bias and $f(.)$ denotes the activation function that defines the output of the neuron [7, 9].

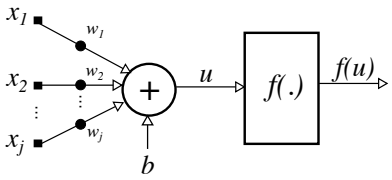

Fig. 1: Functional structure of a neuron.

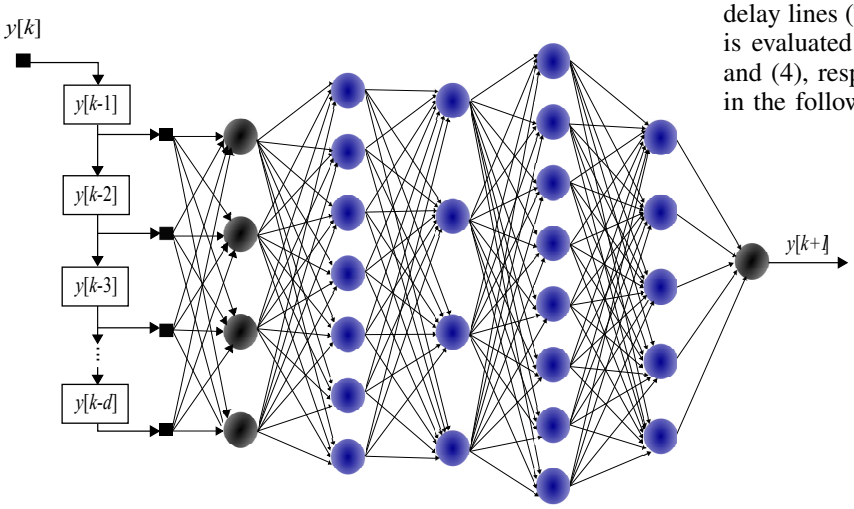

Fig. 2: ANN consisting of five layers taken from [6], and including the description of the delay line at the input layer.

The final state is evaluated by (2), where the function f can be unitary step, lineal, sigmoid, hyperbolic tangent or gaussian function [5, 10]. These and other issues are selected heuristically according to the application [7].

Among all types of ANNs, one can identify two major types of network topology, namely: feedforward and recurrent one. Feedforward ANNs, processes information in one direction, so that the prediction datum in $y[k+1]$ depends only on the input values $y[t]$ and cannot capture possible dependencies of $y[k+1]$ on earlier values of y, as shown in [11]. Recurrent ANNs allow the neurons to depend not only on the input variables $y[t]$, but also on their own lagged values, therefore, this kind of network builds a memory in the evolution of the neurons [11]. However, there is not a recipe on how many delay lines (see Fig. 2) are required at the input layer to perform a good prediction.

Before implementing an ANN into an FPGA [7], one needs to train the topology for which a learning technique must be selected to adjust the parameters during the train processes. The learning technique can be supervised or unsupervised. The former does not involve competition, but it uses an external agent (actual output) for each input pattern that guides the learning process [12], in this case there is a teacher to provide feedback information. The unsupervised learning does not need a teacher to provide feedback for the information, in this case the network discovers by itself patterns, features, regularities, correlations, or categories in the input data and code for them in the output [5].

In this work the feedforward ANN shown in Fig. 2 is trained by using experimental data of the three chaotic time series with different MLE listed in Table I. For the first five layers from the input, we applied an hyperbolic tangent activation function that can be implemented in digital hardware as shown in [7], and for the output layer we applied a linear function. The learning technique is supervised and the weights for the input neurons have associated a time delay line ($y[k-1] \ldots y[k-d]$) to provide a finite dynamic response for the time series data. In this manner, in our experiments, chaotic time series prediction are performed by adding tapped delay lines (TDL) to perform better predictions. The prediction is evaluated by measuring the MSE and RMSE by using (3) and (4), respectively. Details on the experiments are provided in the following Section.

$$y = f(u) = f\left(\sum_{j=1} x_j w_j + b\right) \tag{2}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}[y_i - \hat{y}_i]^2 \tag{3}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}[y_i - \hat{y}_i]^2} \tag{4}$$

IV. EXPERIMENTAL RESULTS

Before training the ANN topology shown in Fig. 2, the three chaotic time series data are normalized by MATLAB *mapminmax* within the range $[-1, 1]$. Among the training algorithms available into MATLAB, the *levenberg-marquardt*

algorithm (trainlm) is applied herein. The training is executed using 3 subsets of data: The first one (training) computes the gradient, weights and bias updating. The second subset (validation) monitors the error during the training. Finally, the third subset (test) adjust the error during the validation process. Such processes are executed into MATLAB *dividerand* using: training ratio (trainRatio)=0.8, validation ratio (valRatio)=0.1, and test ratio (testRatio)= 0.1.

As already known, ANNs are majorly used to perform short-term prediction, and in this case feedforward ANN topologies are the good option. As mentioned previously, in feedforward ANNs the information flows in one direction and the prediction depends on input signals, only. That way, the network shown in Fig. 2 performs only one-step ahead prediction after it has been trained. The state variable x of the chaotic time series is used for prediction and the time series consists of 21650 data that was generated by using an FPGA. The first 15650 samples were used as training data, while the remaining 6000 were used to test the prediction model. We added tapped delay lines (TDL), with $d = 10$ delays. The chaotic data was saved into a computer and then transmitted to the FPGA-based ANN by implementing a serial communication, as shown in Fig. 4, where at the output of the block ANN-Rx-Tx a register is located and it is enabled after 10 clock cycles, which are required by the ANN to process the data from the chaotic time series to the output prediction block. This data is concatenated with 6 bits (to have a word of 32 bits), so that four packages of 8 bits are created, and which are selected by a multiplexer when they are transmitted.

A state machine is designed to control the whole FPGA-based ANN prediction system. In the FPGA realization, multipliers are improved by implementing single constant multiplication blocks as already highlighted in [7], and which improve the maximum operating frequency (FMax) of the ANN. Taking as reference the clock of the FPGA, we implemented a frequency divider to obtain the required clock frequencies in all the ANN blocks, therefore we obtained Fmax = 26.32 MHz.

The determination of the model structure substantially affects the performance of learning and prediction of the ANN topology. For this reason, the MSE and RMSE are used to measure the prediction performance of the feedforward ANN topology, as described by (3) and (4).

TABLE II: The average of statistical metrics

	Feedforward artificial neural network			
MLE	Epoch	Performance	MSE	RMSE
0.3761	842.50	8.97×10^{-3}	2.45×10^{-11}	4.95×10^{-6}
0.3425	783.25	1.02×10^{-5}	9.21×10^{-11}	9.59×10^{-6}
0.2658	846.45	1.29×10^{-6}	2.63×10^{-11}	5.13×10^{-6}

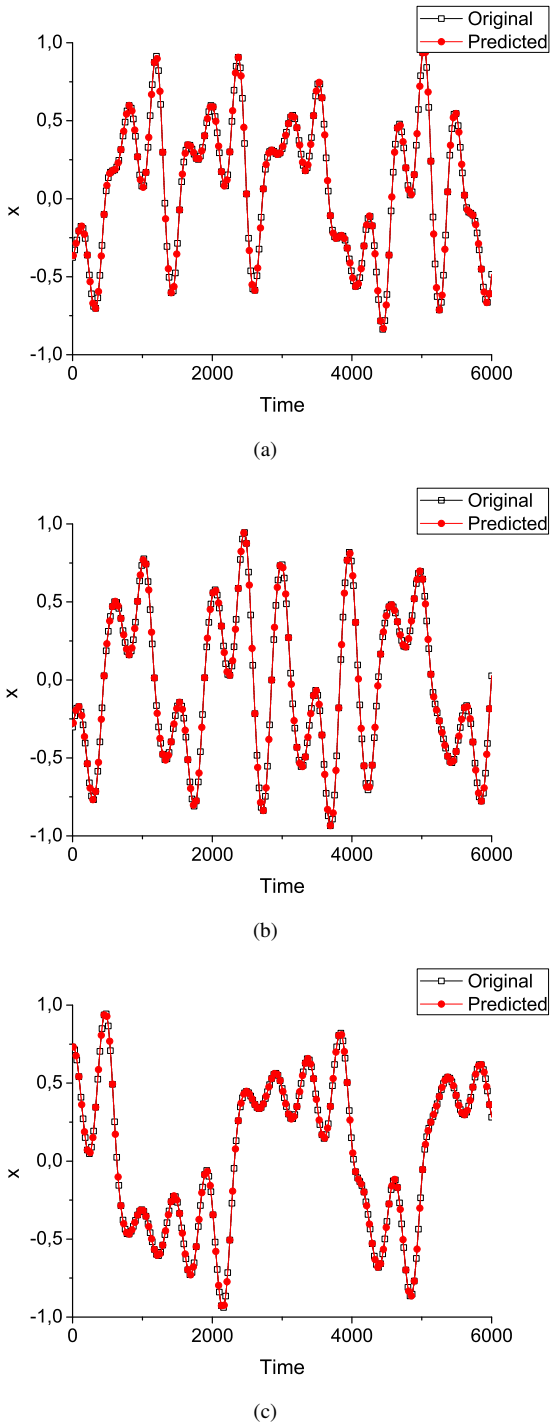

Fig. 3: Prediction of the time series generated by a chaotic oscillator based on saturated nonlinear function series with: (a) MLE=0.3761, (b) MLE=0.3425 and (c) MLE=0.2658

TABLE III: Results of statistical metrics

MLE=0.3761				MLE=0.3425				MLE=0.2658			
Epoch	Perform	RMSE	MSE	Epoch	Perform	RMSE	MSE	Epoch	Perform	RMSE	MSE
1000	4.3292e-6	1.4209e-6	2.0190e-12	452	3.5200e-6	3.2435e-6	1.0520e-11	659	1.8931e-6	9.1185e-7	8.3149e-13
1000	9.7057e-7	1.7215e-6	2.9636e-12	259	4.0658e-6	3.5167e-6	1.2368e-11	1000	6.7778e-7	1.0833e-6	1.1735e-12
1000	1.0726e-6	2.1815e-6	4.7588e-12	1000	3.7602e-5	4.0839e-6	1.6678e-11	1000	2.3744e-6	2.6479e-6	7.0111e-12
562	3.6240e-7	2.8254e-6	7.9829e-12	854	8.5960e-6	4.7094e-6	2.2179e-11	532	5.0810e-7	2.9305e-6	5.8611e-6
645	4.3223e-7	3.6286e-6	1.3167e-11	1000	1.3535e-6	4.7531e-6	2.2592e-11	1000	4.9100e-7	3.1895e-6	1.0173e-11

Thus, after the training process, we selected the ANN topology that provided the smallest error value in the validation set given from 20 experimental runs for a double scroll attractor with different MLE values. Figure 3 shows the prediction of the three experimental chaotic time series with different MLE as listed in Table I. Table III lists the mean of the epochs, performance, and statistical metrics obtained from 20 experimental runs for a double scroll attractor and for the three different MLE values. Finally, Table II lists the five best values of MSE and RMSE for the three different MLEs.

V. Conclusion

Chaotic time series prediction has not been taking into account when they have different MLE values. Among all the available time series prediction techniques, we selected ANNs for their properties on modeling nonlinear behavior. In the state of the art, there are not guidelines to estimate the number of layers for implementing an ANN, all research works mention that they depend on the problem being solved. The ANN shown in Fig. 2 is capable of predicting chaotic time series with different MLE values, it was implemented into an FPGA and three experimental results are shown in Fig. 3. It leads us concluding that feedforward ANNs are quite useful for the short-term prediction. In this case we obtained one-step ahead prediction by using TDLs with 10 delays. Finally, the simulation results, the FPGA implementation and the statistical MSE and RMSE metrics, demonstrated the suitability of the feedforward ANN to predict the three experimental chaotic time series listed in Table I.

Acknowledgment

The authors want to thank to UC-MEXUS-CONACyT collaborative research grants for the funding support under project CN-61-161, and to CONACyT-Mexico under grant 237991.

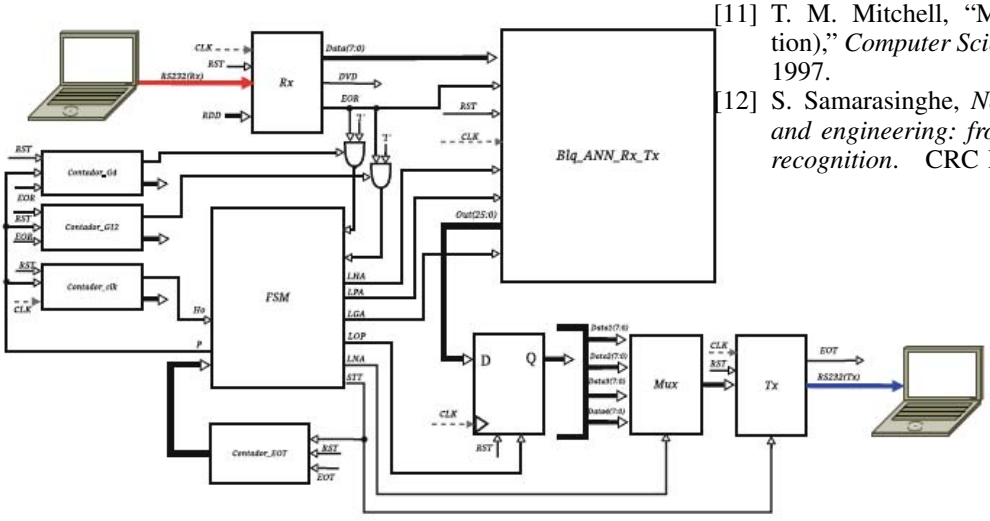

Fig. 4: Diagram of the FPGA-based ANN and the serial interface to feed chaotic time series data from a personal computer.

References

[1] L. G. de la Fraga and E. Tlelo-Cuautle, "Optimizing the maximum lyapunov exponent and phase space portraits in multi-scroll chaotic oscillators," *Nonlinear Dynamics*, vol. 76, no. 2, pp. 1503–1515, 2014.

[2] R. Chandra and M. Zhang, "Cooperative coevolution of elman recurrent neural networks for chaotic time series prediction," *Neurocomputing*, vol. 86, pp. 116–123, 2012.

[3] E. Parras-Gutierrez, V. M. Rivas, M. Garcia-Arenas, and M. Del Jesus, "Short, medium and long term forecasting of time series using the l-co-r algorithm," *Neurocomputing*, vol. 128, pp. 433–446, 2014.

[4] H. J. Sadaei, R. Enayatifar, F. G. Guimarães, M. Mahmud, and Z. A. Alzamil, "Combining arfima models and fuzzy time series for the forecast of long memory time series," *Neurocomputing*, vol. 175, pp. 782–796, 2016.

[5] C.-T. Lin and C. Lee, *Neural fuzzy systems: a neuro-fuzzy synergism to intelligent systems*. Prentice-Hall, Inc., 1996.

[6] M. Molaie, R. Falahian, S. Gharibzadeh, S. Jafari, and J. C. Sprott, "Artificial neural networks: powerful tools for modeling chaotic behavior in the nervous system," *Frontiers in computational neuroscience*, vol. 8, p. 40, 2014.

[7] E. Tlelo-Cuautle, L. G. de la Fraga, and J. Rangel-Magdaleno, *Engineering Applications of FPGAs*. Springer, 2016.

[8] E. Molino-Minero-Re, J. G. Cardoso-Mohedano, A. C. Ruiz-Fernandez, and J.-A. Sanchez-Cabeza, "Comparison of artificial neural networks and harmonic analysis for sea level forecasting (urias coastal lagoon, mazatlan, mexico)," *Ciencias Marinas*, vol. 40, no. 4, pp. 251–261, 2014.

[9] J. D. Rairán Antolines, "Reconstruction of periodic signals using neural networks," *Tecnura*, vol. 18, no. 39, pp. 34–46, 2014.

[10] T. Masters, *Practical neural network recipes in C++*. Morgan Kaufmann, 1993.

[11] T. M. Mitchell, "Machine learning (international edition)," *Computer Science Series. McGraw-Hill, New York*, 1997.

[12] S. Samarasinghe, *Neural networks for applied sciences and engineering: from fundamentals to complex pattern recognition*. CRC Press, 2006.