Artificial Neural Network Modelling of Rössler's And Chua's Chaotic systems

Jobin Sunny, Jesse Schmitz, Dr. Lei Zhang
Faculty of Engineering & Applied Science,
University of Regina, Saskatchewan, Canada

***Abstract*—This paper presents the detailed analysis of various Artificial Neural Network (ANN) modelling techniques for chaotic systems. Specifically, Rössler's system and Chua's system are selected for this study for their practical applications, and the outputs of these two systems are used for the ANN training. The Nonlinear Auto-Regressive(NAR) modelling is used for chaotic time series prediction. The Nonlinear Auto-Regressive with Exogenous Inputs (NARX) modelling is used for generating chaotic time series outputs with varying system parameters as exogenous inputs. The research results show that ANN performs well in modelling chaotic systems. Rössler's attractor is modelled using Radial Basis Function Network(RBFN) and a comparative study between FeedForward Neural Network(FFNN) and RBFN is done. The result shows that RBFN uses more neurons to achieve similar training performance compared to FFNN. The 3-layer ANN architecture with hidden neurons varying from 1 to 16 is designed and trained using MATLAB NN toolbox. In a fixed-point FPGA implementation perspective, a study on the performance of ANN modelling of chaotic systems is very relevant.**

Keywords*—*Artificial Neural Network, Chaos, FPGA, RBFN, Rössler's, Chua's, Auto-Regressive, Fixed-Point, MATLAB.

I. Introduction

There are a lot of chaotic systems around us. Double pendulum, weather systems and electroencephalogram (EEG) are examples of a few chaotic phenomena known to us [5]. Chaotic systems satisfy (i) extreme sensitivity to initial conditions (ii) are deterministic in nature (iii) and show fractal geometry. Modelling is very important for chaotic systems as it gives us the freedom to scrutinize the behaviour of the systems under different conditions. ANN's are used for pattern recognition, classification and are an effective tool for modelling nonlinear systems [1]. Hardware implementation is the end goal for our application of chaotic systems. FPGA implementation of different chaotic systems is described in [2][4]. This equation based implementation is an ideal case when we consider that equations of most of the natural systems (eg: EEG) are unknown. Training the ANN offline and use trained weights in the hardware is a promising solution for modelling wide variety of dynamic systems. A previous attempt at ANN modelling of the Rössler's system is done in [3] using 15 input neurons, 20 hidden neurons and 1 output neuron. With the method proposed in this paper better performance can be achieved using lesser number of neurons. Bifurcation is the changes in the qualitative picture of a given system with respect to parameters. This work is a unique attempt to model the Rössler's bifurcation using NARX ANN and to study the training performance. In section II of this paper, the Rössler's & the Chua's chaotic map and bifurcation details are being discussed. In section III, different ANN's used are examined first, these include FFNN, RBFN, NAR and NARX. In section IV, design and training of ANN, ANN architecture, different training algorithms used and error goal selection are explained. Results of the study and performance comparison are done in section V and a conclusion and future scope of the study is described in section VI.

II. Chaotic Systems

A. The Rössler's Attractor

The Rössler's chaotic system is a continuous nonlinear system expressed by three ordinary differential equations (ODE's). The Rössler's system has 3 parameters namely a, b and c, parameter values (a=0.2,b=0.2,c=5.7) for initial values of x=0.1, y=0.1, z=0.1 are selected to generate chaotic map. The Rössler's system equations are given by (1):

$$\begin{aligned} \frac{dx}{dt} &= -y - z \\ \frac{dy}{dt} &= x + a \cdot y \\ \frac{dz}{dt} &= b + z \cdot (x - c) \end{aligned} \tag{1}$$

The solutions of three dimensional differential equations depend on the initial values which are determined based on critical points. To find the critical points, the 3 Röosslers equations are set to zero and the (x, y, z) coordinates are determined by solving the resulting equations.

Solving this system of equations for chaotic parameter values (a=0.2,b=0.2,c=5.7), two critical points are obtained. Initial points are chosen near the critical point to keep them within the basin of attraction and are subsituted in the ODE to generate attractor maps[4]. The chaotic maps are iterated using MATLAB ODE45 [9] based on explicit Runge-Kutta (4,5) formula which has less error than the Euler method.

B. The Chua's Attractor

Chua's circuit is an electronic circuit that exhibits nonlinear behaviour like chaos and bifurcation [6]. From the circuit variables we obtain that Chua's equation has 2 parameters, α and β. The Chua's circuit can be modeled mathematically by a system of three dimensions of nonlinear ODE's as in (2):

$$\begin{aligned}\frac{dx}{dt} &= \alpha \cdot (y - f(x)) \\ \frac{dy}{dt} &= x - y + z \\ \frac{dz}{dt} &= -\beta \cdot y\end{aligned} \tag{2}$$

where α, β are real numbers, and f(x) is function of a single variable x which is defined as a piece-wise linear function as shown in (3):

$$f(x) = m_1 x + \frac{1}{2} \cdot (m_0 - m_1)[|x+1| - |x-1|] \tag{3}$$

The Chua's attractor is generated by substituting initial conditions and parameter values in (2).

C. BIFURCATIONS

Bifurcation diagrams are plots of system output with respect to a system parameter. They show how the system navigates through critical points, periodic and chaotic states on variation of a parameter [5]. The bifurcation studied here is a local bifurcation, which occurs when a change in the parameter causes any effect to the stability of a critical point. Bifurcations with respect to parameter α, β and γ are plotted against maximum value of X dimension of the Rössler's output. The bifurcation parameter details, step size and number of training samples used are shown in Table 1 below.

III. ARTIFICIAL NEURAL NETWORKS

Artificial Neural networks are simulated collections of neurons. A general mathematical representation of an individual neuron within an ANN architecture is shown by (4):

$$a_j^i = \sum_{i=1}^{N_l - 1} w_{ji}^l \cdot x_i + b_j^l \qquad y_j^l = f_l(a_j^l) \tag{4}$$

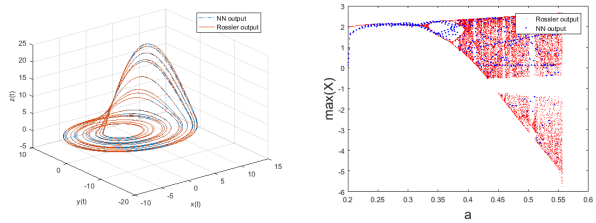
Where N_l is the number of neurons at *l*-layer, *j*=1, 2, . N_l. For a 3 layered network each hidden neuron *j* receives the output of each input neuron *i*, from the input layer multiplied with a weight of w_{ij}^l. The sum of all the weighted inputs are used by an activation function f_l to produce the output of the hidden layer neuron and it is forwarded to an output layer. b_j^l is the bias of the *j*th neuron at the *l*th layer, which are added to randomize the initial weights to get a better chance of convergence.

A. Feed Forward Neural Network(FFNN)

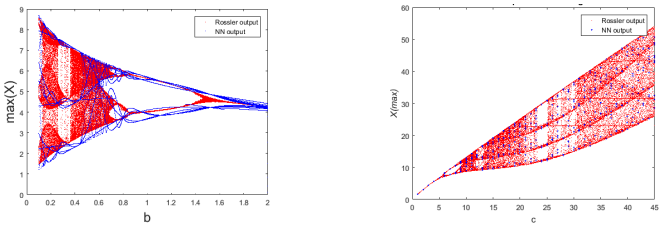
FFNN is a kind of ANN where no feedback connections exist. According to universal approximation theorem for continuous function to approximate successfully only one hidden layer is enough [7]. The ANN works based on back propagation principle and its goal is to minimize the sum of the MSE which is given in (5):

$$\sum MSE = \epsilon = \frac{1}{N} \cdot \sum (t - y)^2 \tag{5}$$

ϵ is Mean Square Error, *t* is the target vector, *y* is the output vector. ANN weights changes using a nonlinear optimization method called gradient descent.



(a) Initial values of x,y,z = 0, 0.1, 0.2(b) a=(0.2 to 0.556), b=2.0,c=4.0 parameter values a=0.2, b=0.2, c=5.7



(c) b= (0.1 to 2), a=0.2,c=5.7

(d) c= (0 to 45) where, a=0.1 and b=0.1

Fig. 1: (a) State diagram of NN model superimposed on state diagram of actual Rösslers attractor. (b) (c) (d)The Rössler's bifurcation for parameter a, b and c vs Xmax superimposed on NN Rössler's bifurcation

B. Radial Basis Function Network(RBFN)

RBFN is another member of the FFNN and has Radial basis (Radbas) as activation function. The input is divided into clusters and function approximation is done by manipulating free parameters: cluster center(c_i), spread and weight. Input to the RBFN is a vector of real numbers X ϵR^n and the output of the network is a scalar function of the given input vectors as in (6)

$$y(x) = \sum_{i=1}^{N} a_i \cdot \rho(||x - c_i||) \tag{6}$$

where, *N*-number of neurons, $a_i$weight of neuron *i*, c_i-center vector of neuron *i* & ρ is the Radbas function. Given the number of hidden neurons flexible, an RBFN can approximate any continuous functions with arbitrary precision [8] and this property makes it ideal for function approximation.

C. Nonlinear Auto Regressive Artificial Neural Network (NAR)

NAR ANN can train and predict a time series from that series past values. No external input is given here, input is delayed target values itself. The output *y(t)* given *d* past values of *y(t)* is:

$$y(t) = f(y(t-1) \cdots y(t-d)) + \epsilon_t \tag{7}$$

where ϵ_t is an error term which is the difference between target and predicted input fed into the system.

D. Nonlinear Auto-Regressive with Exogenous Inputs (NARX)

NARX ANN can learn to predict a time series given past values of the same time series, plus another time series called external input. The NARX predict time series *y(t)* given d past values of *y(t)* and another external series *x(t)*:

$$y(t) = f(x(t-1)\cdots.x(t-d), y(t-1)\cdots.y(t-d)) + \epsilon_t \quad (8)$$

IV. Design Training of Neural Network

A. ANN Architecture & Error Goal

Throughout this study MATLAB ANN Toolbox is used [9]. Different ANN topologies with hidden layers ranging from 1 to 16 neurons are studied. MLP with input layer, one hidden layer and output layer having 3 inputs and 3 outputs are employed for system modelling. For training RBFN single function layered ANN is used. MATLAB iteratively creates a Radial Basis Network with one neuron at a time and neurons are added to the network until the training MSE falls beneath the goal or a maximum number of neurons is achieved. A single hidden layer based NAR network is used for bifurcation modelling. For feedback delays, different values are experimented and a delay of 1 time step is found to be the optimal. Also this information is used in NARX network creation. MATLAB ANN toolbox has a default training performance goal of zero. Although minimum performance goal is best, it may result in invalid training and over fitting [9]. For FFNN an MSE goal of (1.0E-06) is selected and for RBFN resource usage comparison is done at a training performance similar to that of FFNN. For Chua's system error goal is set as (1.0E-06) and for NAR and NARX implementation the error goal is set as (1.000E-03). MSE goals are set based on previous study done [1][3] and whether the results are practically achievable in a real world system [2].

B. Training Data & Algorithms Used

Three dimensions of chaotic systems are used for both input & target vector during ANN training. From a single dimensional time series x, input vector is created as the one step delayed time series while target vector is the one step ahead time series. This is achieved by windowing method where consecutive values of time series are assigned one value as input and another as target. To avoid improper starting point off the trajectory the first 10000 samples are discarded. The Tan-Sigmoid activation function is used in the hidden layer with each trial running at most a 1000 epochs. Three trials are done for each architecture to avoid the gradient hitting local minima. 3 different algorithms used here, namely the Levenberg-Marquat(LM), the Bayesian Regulation(BR)(slower but better convergence) and the Scaled Conjugate Gradient(SCG) algorithm respectively [1]. Mean Square Error (MSE) is used to evaluate training performance. 6 validation stops are used to check the consecutive increment in gradient. For RBFN, MSE is calculated for different values of spread and the value which has minimum training MSE performance is chosen. NAR network with a single hidden layer is used to model the Rössler's chaotic system for system parameters of a=0.2, b=0.2, c=5.7. Total 10000 set of samples are used along with 1000 Epochs is tested for LM algorithm. The single input to NARX is bifurcation parameter of the Rössler's map. Bifurcation range of b=0.1 to 2 (step size=0.01) & a=0.2, to 0.5 (step size=0.005) & c=1 to 45 (step size=0.1) is experimented. 800 samples/parameter is used for training bifurcation. Input has same delay defined as target, because we are training the system with data corresponding to each parameter and we want that correlation between data and output to stay during training.

Variable Para	**Range**	**Fixed Para1**	**Fixed Para2**	**Step size**	**Sample per Para**
a	0.2 to 0.5	b=2.0	c=4.0	0.005	800
b	0.1 to 2.0	a=0.2	c=5.7	0.01	800
c	1 to 45	a=0.1	b=0.1	0.1	800

Table1: Bifurcation parameter and training sample details

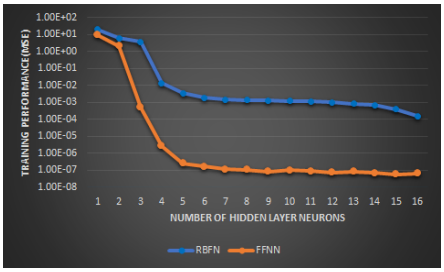
Quantitative analysis of the computational complexity of the ANN model versus Runge-Kutta based ODE approximation in generating chaotic time series is indicated using run time comparison in table 2.

Chaotic system	**Sample points**	**Runtime (sec)**	**Method**
Rössler's map	10000	0.433	ODE45
Rössler's map	10000	1.014	ANN
Chua's map	10000	0.393	ODE45
Chua's map	10000	1.365	ANN
Rössler's Bifurcation a	48000	2.563	ODE45
Rössler's Bifurcation	48000	12.531	ANN

Table2: Runtime comparison between ODE method and NN generated chaotic models.

V. PERFORMANCE EVALUATION

For all architectures used for modelling a general trend is that training performance improves as we increase the number of neurons. Output time series and bifurcation diagrams w.r.t parameter a, b, c are reconstructed using the dynamic NN model is shown in figure 1. FFNN for Rössler's system achieved a training performance of order of 8.00E-08, for LM and BR training algorithms (at 5 neuron) but SCG gave a poor performance of the order of 3.00E-03. Considering the rate of convergence and training time LM is the best algorithm. An interesting result is that after 8 neurons the training performance does not improve significantly. The Chua's system training performance followed the Rössler's system and an MSE of order of 3E-04 is achieved for LM algorithm. Modelling of chaotic system using RBFN takes more neurons than FFNN to



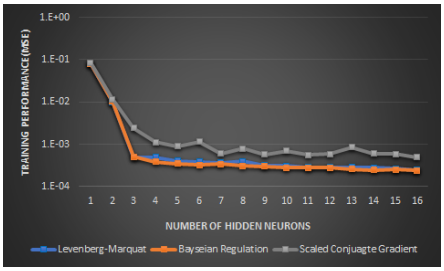
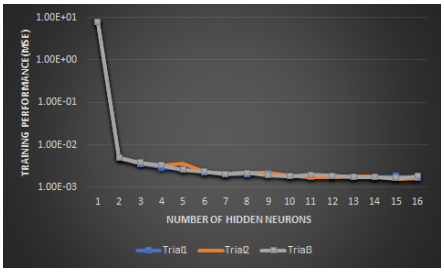


Fig. 2: (a) Training performance comparison plot of RBFN vs FFNN for the Rössler's system.(b) Training performance of FFNN over different learning algorithms for the Chua's map



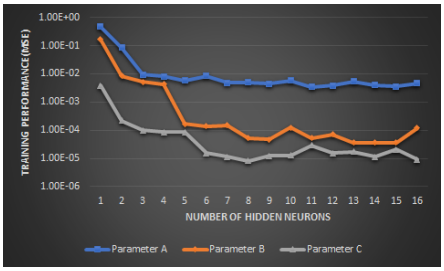


Fig. 3: (a) Training performance of NAR ANN over different trials for Rössler's chaotic map .(b) Training performance comparison of Rössler's bifurcation for parameters a,b,c vs number of neurons for NARX ANN

achieve same training performance. FFNN achieves an training performance of 1E-06 with 5 neurons while RBFN achieves the same with 20 neurons. The training performance comparison plot of RBFN vs FFNN for different ANN architectures for the Rössler's system and the FFNN performance for the Chua's map is shown in figure 2(a). The NAR ANN for Rössler's map achieved a training performance of 1E-02 and the plot is shown in the figure 3(a). Even though training performance improves with increase in number of neurons, the change is negligible after 12 neurons. For NARX ANN the training performance goal of bifurcation MSE $<$ (1.000E-03) is achieved in all cases. LM algorithm performs best for parameter c when comparing 3 bifurcations of the Rössler's system. A graph representing trials vs training performance for different topologies for 3 bifurcations is shown in figure 3(b). It is reported in [10] that for FPGA implementation of ANN a 6 fractional bit precision is sufficient and based on this the training target for ANN is computed as 1.5625E-02 [1]. In table 2 the run time of trained ANN is computed for 8 neurons in the hidden layer with above performance goal and found to be close to Runge-Kutta based ODE approximation method.

VI. Conclusion

This paper is an attempt to study about ANN modelling of chaotic system. Different ANN topologies are experimented on Rössler's and Chua's systems. FFNN, RBFN are compared for their resource usage and performance. Bifurcation modelling using ANN is first attempted using NARX ANN to verify the performance. FFNN modelling is reproduced on the Chua's attractor and the claims are confirmed. Each architecture is implemented on 16 different ANN topologies to find minimum training performance. NARX or NAR ANN has the accumulated feedback error causing degradation of performance. This comprehensive study on the resource usage, algorithms, NN models, training performance and behavior of trained model aids in future development of optimized hardware models. Future work will be carried out how to reduce the accumulation of feedback errors in NAR/NARX ANN and use the system for modeling specific EEG patterns.

References

[1] Lei Zhang, *Artificial Neural Network Model Design and Topology Analysis for FPGA Implementation of Lorenz Chaotic Generator* :30th Canadian Conference on Electrical and Computer Engineering (CCECE),IEEE. 2017.

[2] Jesse Schmitz, Lei Zhang, *Rossler based Chaotic communication system implemented on FPGA* :30th Canadian Conference on Electrical and Computer Engineering (CCECE),IEEE. 2017.

[3] Archana R, A. Unnikrishnan R. Gopikakumari, M.V Rajesh, *An Intelligent Computational Algorithm based on Neural Networks for the Identification of Chaotic systems* :Recent Advances in Intelligent Computational Systems (RAICS),IEEE. 2011.

[4] Lei Zhang, *Henon map chaotic system analysis and VHDL-based fixed-point FPGA implementation for brain stimulation* :30th Canadian Conference on Electrical and Computer Engineering (CCECE),IEEE. 2017.

[5] Razieh Falahian,Maryam Mehdizadeh Dastjerdi,et.al, *Artificial neural network-based modeling of brain response to flicker light* :Nonlinear Dyn,IEEE. 2015.

[6] T. Matsumoto, *A chaotic attractor from Chua's circuit, IEEE Transactions on Circuits and Systems* : ,IEEE. 1984.

[7] S. Mcmillan, *Neural Networks: A comprehensive foundation* :N.Y. 1994.

[8] Bishop C, *Neural Networks for pattern recognition* : Clarendon Press,Oxford, 1995.

[9] MATLAB 16.0, *The MathWorks, Inc., Natick,Massachusetts, United States.*

[10] M. Alcin, I. Pehlivan, and smail Koyuncu, *Hardware design and implementation of a novel ann-based chaotic generator in fpga* :Optik International Journal for Light and Electron Optics, vol. 127, 2016.