

Building a Chaotic Proven Neural Network

Jacques M. Bahi, Christophe Guyeux, and Michel Salomon*
Computer Science Laboratory (LIFC) - University of Franche-Comté
IUT de Belfort-Montbéliard BP 527, 90016 Belfort Cedex, France
e-mail: christophe.guyeux@univ-fcomte.fr

Abstract—Chaotic neural networks have received a great deal of attention these last years. In this paper we establish a precise correspondence between the so-called chaotic iterations and a particular class of artificial neural networks: global recurrent multi-layer perceptrons. We show formally that it is possible to make these iterations behave chaotically, as defined by Devaney, and thus we obtain the first neural networks proven chaotic. Several neural networks with different architectures are trained to exhibit a chaotic behavior.

Devaney's chaos; chaotic iterations; mathematical topology

I. INTRODUCTION

Due to the widespread use of the Internet and new digital technologies in nowadays life, security in computer applications and networks never was such a hot topic. Digital rights managements, e-voting security, anonymity protection, and denial of services are examples of new security concerns appeared this last decade. Tools on which this security is based are, among others: hash functions, pseudo-random number generators, cryptosystems, and digital watermarking schemes. Due to their wide use in security protocols, these tools are targeted everyday by hackers and new threats are frequently revealed. For example, security flaws have been recently identified in the previous standard in hash functions called SHA-1 [1]. As the new standards (SHA-2 variants) are algorithmically similar to SHA-1, stronger hash functions using new concepts are desired.

New approaches based on chaos are frequently proposed as an alternative to solve concerns which recurrently appear in the computer science security field [2]–[4]. The advantage of the use of chaotic dynamics for security problems lies in their unpredictability proven by mathematical theory of chaos. This theory brings many qualitative and quantitative tools, namely ergodicity, entropy, expansivity, and sensitive dependence to initial conditions [5]. These tools allow the study of the randomness of the disorder generated by the considered system [6].

Recently, many researchers have built chaotic neural networks in order to use it as a component of new proposed hash functions [7], pseudo-random number generators, cryptosystems [8], [9], and digital watermarking schemes. Since the first introduction by McCulloch and Pitts in 1959, artificial neural networks have been shown to be efficient nonlinear statistical data modeling tools which can implement complex mapping functions. They may be trained

*Authors in alphabetical order

to learn a chaotic process and also, by construction, exhibit suitable properties: data confusion and diffusion, one-way function and compression. Security is not the only application domain of such new tools: the existence of chaos in our brain has been recently revealed, and the use of a chaotic artificial neural network as a model can serve, for example, neuroscientists in their attempts to understand how the brain works.

However, using an element of chaos as a component of the scheme is not sufficient, in our opinion, to be able to claim that the whole process behaves chaotically. We believe that this claim is not so evident and must be proven. Let us notice that up to now the proposed chaotic neural networks have failed to convince the mathematics community due to a lack of proof. This is why it is explained in this paper how it is possible to build an artificial neural network that behaves chaotically, as it is defined by Devaney [10]. We will establish a correspondence between particular neural networks and chaotic iterations, which leads to the definition of the first artificial neural networks proven chaotic, according to Devaney.

The remainder of this paper is organized as follows. The next section is devoted to some recalls on chaotic iterations and Devaney's chaos followed by a brief description of artificial neural networks (ANNs). Section III presents a review of some works related to chaotic neural networks. Our approach, which consists in building a global recurrent ANN whose iterations are chaotic, is formalized and discussed in Section IV. Concrete examples of chaotic neural networks also show the relevance of our method. Finally in Section V we conclude and outline future work.

II. BASIC RECALLS

In the sequel S^n denotes the n^{th} term of a sequence S and E^i denotes the i^{th} component of a vector E . $f^k = f \circ \dots \circ f$ is for the k^{th} composition of a function f . Finally, the following notation is used: $[[1; N]] = \{1, 2, \dots, N\}$.

A. Chaotic Iterations versus Devaney's Chaos

1) Chaotic Iterations:

Let us consider a finite system with a finite number $N \in \mathbb{N}^*$ of elements (or cells), so that each cell has a boolean state. A sequence of length N of boolean states of the cells corresponds to a particular *state of the system*. A sequence which elements belong to $[[1; N]]$ is called a *strategy*. The set of all strategies is denoted by S .

Definition 1 The set \mathbf{B} denoting $\{0,1\}$, let $f: \mathbf{B}^N \rightarrow \mathbf{B}^N$ be a function and $S \in \mathcal{S}$ be a strategy. The so-called *chaotic iterations* are defined by $x^0 \in \mathbf{B}^N$ and

$$\forall n \in \mathbf{N}^*, \forall i \in [[1; N]], x_i^n = \begin{cases} x_i^{n-1} & S^n \neq i \\ (f(x^{n-1}))_{S^n} & S^n = i. \end{cases}$$

In other words, at the n^{th} iteration, only the S^n -th cell is “iterated”. Note that in a more general formulation, S^n can be a subset of components and $(f(x^{n-1}))_{S^n}$ can be replaced by, $(f(x^k))_{S^n}$ where $k < n$, describing for example, delays transmission [11]. Finally, let us remark that the term “chaotic”, in the name of these iterations, has a priori no link with the mathematical theory of chaos, recalled below.

2) Devaney’s Chaotic Dynamical Systems:

Consider a topological space (χ, τ) and a continuous function f on χ .

Definition 2 f is said to be *topologically transitive* if, for any pair of open sets $U, V \subset \chi$, there exists $k > 0$ such that $f^k(U) \cap V \neq \emptyset$.

Definition 3 An element (a point) x is a *periodic element* (point) for f of period $n \in \mathbf{N}^*$, if $f^n(x) = x$.

Definition 4 f is said to be *regular* on (χ, τ) if the set of periodic points for f is dense in χ : for any point x in χ , any neighborhood of x contains at least one periodic point.

Definition 5 f is said to be *chaotic* on (χ, τ) if f is regular and topologically transitive.

The chaos property is strongly linked to the notion of “sensitivity”, defined on a metric space (χ, d) by:

Definition 6 f has *sensitive dependence on initial conditions* if there exists $\delta > 0$ such that, for any $x \in \chi$ and any neighborhood V of x , there exists $y \in \chi$ and $n \geq 0$ such that $d(f^n(x), f^n(y)) > \delta$. δ is the *constant of sensitivity* of f .

Indeed, Banks *et al.* have proven in [12] that when f is chaotic and (χ, d) is a metric space, then f has the property of sensitive dependence on initial conditions (this property was formerly an element of the definition of chaos). To sum up, quoting Devaney in [10], a chaotic dynamical system “is unpredictable because of the sensitive dependence on initial conditions. It cannot be broken down or simplified into two subsystems which do not interact because of topological transitivity. And in the midst of this random behavior, we nevertheless have an element of regularity”. Fundamentally different behaviors are consequently possible and occur in an unpredictable way.

3) Chaotic Iterations and Devaney’s Chaos:

In this section we give outline proofs of the properties on which our study of chaotical neural networks is based. The complete theoretical framework is detailed in [13].

Denote by Δ the *discrete boolean metric*, $\Delta(x, y) = 0 \Leftrightarrow x = y$. Given a function $f: \mathbf{B}^N \rightarrow \mathbf{B}^N$, define the function $F_f: [[1; N]] \times \mathbf{B}^N \rightarrow \mathbf{B}^N$ such that

$$F_f(k, E) = (E_j \cdot \Delta(k, j) + f(E)_k \cdot \overline{\Delta(k, j)})_{j \in [[1; N]]},$$

where $+$ and \cdot are the boolean addition and product operations, \overline{x} is the negation of x .

Consider the phase space $\chi = [[1; N]]^N \times \mathbf{B}^N$ and the map

$$G_f(S, E) = (\sigma(S), F_f(i(S), E))$$

where $\sigma: (S^n)_{n \in \mathbf{N}} \in \mathcal{S} \mapsto (S^{n+1})_{n \in \mathbf{N}} \in \mathcal{S}$ is the *shift function*, and the *initial function* i is the map which associates to a sequence, its first term: $i: (S^n)_{n \in \mathbf{N}} \in \mathcal{S} \mapsto (S^0) \in [[1; N]]$.

Thus chaotic iterations can be described by the following iterations [13]

$$\begin{cases} X^0 \in \chi \\ X^{k+1} = G_f(X^k) \end{cases}$$

Let us define a new distance between two points $(S, E), (\tilde{S}, \tilde{E}) \in \chi$ by

$$d((S, E); (\tilde{S}, \tilde{E})) = d_e(E, \tilde{E}) + d_s(S, \tilde{S}),$$

where

- $d_e(E, \tilde{E}) = \sum_{k=1}^N \Delta(E_k, \tilde{E}_k) \in [[0; N]]$
- $d_s(S, \tilde{S}) = \frac{9}{N} \sum_{k=1}^{\infty} \frac{|S^k - \tilde{S}^k|}{10^k} \in [0; 1]$.

This new distance has been introduced in [13] to satisfy the following requirements. When the number of different cells between two systems is increasing, then their distance should increase too. In addition, if two systems present the same cells and their respective strategies start with the same terms, then the distance between these two points must be small because the evolution of the two systems will be the same for a while. The distance presented above follows these recommendations. Indeed, if the floor value $\lfloor d(X, Y) \rfloor$ is equal to n , then the systems E, \tilde{E} differ in n cells. In addition, $d(X, Y) - \lfloor d(X, Y) \rfloor$ is a measure of the differences between strategies S and \tilde{S} . More precisely, this floating part is less than 10^{-k} if and only if the first k terms of the two strategies are equal. Moreover, if the k^{th} digit is nonzero, then the k^{th} terms of the two strategies are different.

It is proven in [13] by using the sequential continuity that the vectorial negation $f_0(x_1, \dots, x_N) = (\overline{x_1}, \dots, \overline{x_N})$ satisfies the following proposition:

Proposition 1 G_{f_0} is a continuous function on (χ, d) .

It is then checked in [13], that in the metric space (χ, d) , the vectorial negation fulfills the three conditions for Devaney’s chaos: regularity, transitivity, and sensitivity. This has led to the following result.

Proposition 2 G_{f_0} is a chaotic map on (χ, d) in the sense of Devaney.

B. Neural Networks

An artificial neural network is a set of simple processing elements called neurons that are interconnected, usually with a layer structure. It takes some input values and produces some output ones. Like a biological neural network, the connections between neurons influence the outputs given by the artificial network. Thanks to a training process, an ANN is able to learn complex relationships between inputs and

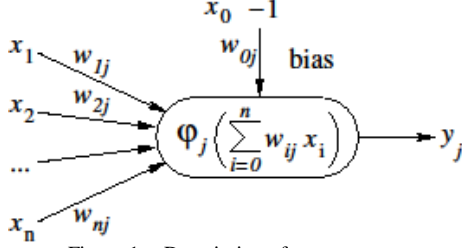


Figure 1. Description of a neuron

outputs. A neuron j computes an output $y = \varphi(x, w)$ where $\varphi()$ is the activation function, x is the input vector, and w the parameter vector. w can be used to parameterize φ or the neuron inputs. In this last case it means that the connections are weighted and a vector w component is then referred to as a synaptic weight. Fig. 1 describes a neuron j with weighted connections. Its output y_j satisfies:

$$y_j = \varphi_j \left(\sum_{i=1}^n w_{ij} x_i + b_j \right) = \varphi_j \left(\sum_{i=0}^n w_{ij} x_i \right)$$

where $x_0 = -1$, $x = (x_1, \dots, x_n)$, and $w_{0j} = -b_j$ defines the bias value.

Neural networks have a layered architecture, but they may differ in the way the output of a neuron affect himself. In fact, based on the connection graph, two kinds of networks can be distinguished: those having a least one loop and those without any one. A neural network which exhibits a loop is called a feedback (or recurrent) network, whereas a network belonging to the second class is said feed-forward. Obviously, a feedback network can be seen as a dynamical system. In the following sections, we use a recurrent version of the multi-layer perceptron (MLP), a well-known ANN architecture for which the universal approximation property has been proven in the feed-forward context [14]. Typically, a MLP consists in layer of input neurons, in one or more layers of hidden neurons, and a layer of output neurons. Since an input neuron is simply used as a channel to dispatch an input to each neuron of the first hidden layer, we will not further consider the input layer. Usually, the neurons of a given layer have similar characteristics and each one is fully connected to the next layer. Finally, note that the number of inputs and output neurons is completely specified by the considered problem, while the number of hidden neurons depends directly on the complexity of the relationships to be learned by the ANN.

As said previously, a neural network is designed to model relationships between inputs and outputs. In order to find a proper modeling, an ANN must be trained so that it provides the desired set of output vectors. The training (or learning) process consists mainly in feeding the network with some input vectors and updating the neurons parameters (weights and bias value) using a learning rule and some information which reflects the quality of the current modeling. When the expected output vectors (D_k) are known in advance, the quality can be expressed through the Mean Square Error [15]:

$$MSE = \frac{1}{2N} \sum_{k=1}^N \|D_k - Y_k\|^2$$

where N is the number of input-output vector pairs used to train the ANN (the pair set is called the training or learning set) and Y_k denotes an output vector produced by the output layer for a given input vector X_k . Consequently, in that case the training process, which is said supervised, results in an optimization algorithm targeted to find the weights and biases that minimize the MSE. Various optimization techniques exist, they have given raise to distinct training algorithms performing iterative parameters update. Gradient-based methods are particularly popular to the backpropagation algorithm, but they are sensitive to local minima. Heuristics like simulated annealing or differential evolution permit to find a global minimum, but they have a slow convergence. To control the training process, two methods are the most commonly used: firstly the number of iterations, also called epochs, reaches an upper bound; secondly the MSE goes below a threshold value.

III. RELATED WORK

Since a while neuroscientists discuss the existence of chaos in the brain. In the context of artificial neural networks, this interest has given raise to various works studying the modeling of chaos in neurons. The chaotic neuron model designed by Aihara *et al.* [16] is particularly used to build chaotic neural networks. For example, in [17] is proposed a feedback ANN architecture which consists of two layers (apart from the input layer) with one of them composed of chaotic neurons. In their experiments, the authors showed that without any input sequence the activation of each chaotic neuron results in a positive average Lyapunov exponent, which means a true chaotic behavior. When an input sequence is given iteratively to the network the chaotic neurons reach unstable periodic orbits with different periods, and thus potentially provide a recognition state. Similarly, the same authors have recently introduced another model of chaotic neuron: the nonlinear dynamic state (NDS) neuron, and used it to build a neural network which is able to recognize learned unstable periodic orbits identifying patterns [18].

Today, another field of research in which chaotic neural networks have received a lot of attention is data security. In fact, chaotic cryptosystems are an appealing alternative to classical ones due to properties such as sensitivity to initial conditions or topological transitivity. Thus chaotic ANNs have been considered to build ciphering methods, hash functions, digital watermarking schemes, pseudo-random number generators, etc. In [8] such a cipher scheme based on the dynamics of Chua's circuit is proposed. More precisely, a feed-forward MLP with two hidden layers is built to learn about 1500 input-output vector pairs, where each pair is obtained from the three nonlinear ordinary differential equations modeling the circuit. Hence, the proposed chaotic neural network is a network which is trained to learn a true chaotic physical system. In the cipher scheme the ANN plays the role of chaos generator with which the plain-text will be merged. Untrained neural networks have also been considered to define block ciphering [9] or hash functions [7]. The background idea is to exploit the inherent properties of the ANNs architecture such as diffusion and confusion.

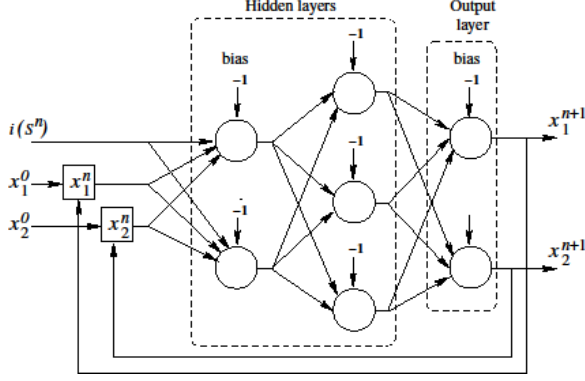


Figure 2. Example of a global recurrent neural network modeling function F_{f_0} such that $x^{n+1} = (x_1^{n+1}, x_2^{n+1}) = F_{f_0}(i(S^n), (x_1^n, x_2^n))$

IV. A FIRST RECURRENT NEURAL NETWORK CHAOTIC ACCORDING TO DEVANEY

A. Defining a First Chaotic Recurrent Neural Network

We will now explain how to build a chaotic neural network using chaotic iterations.

Let us reconsider the vectorial negation function denoted by $f_0: \mathbf{B}^N \rightarrow \mathbf{B}^N$ and its associated map $F_{f_0}: [[1; N]] \times \mathbf{B}^N \rightarrow \mathbf{B}^N$. Firstly, it is possible to define a MLP which recognizes F_{f_0} . That means, for all $(k, x) \in [[1; N]] \times \mathbf{B}^N$, the response of the output layer to the input (k, x) is $F_{f_0}(k, x)$. Secondly, the output layer can be connected to the input layer as it is depicted in Fig. 2 leading to a global recurrent neural network working as follows:

- At the initialization stage, the ANN receives a boolean vector $x^0 \in \mathbf{B}^N$ as input state, and $S^0 \in [[1; N]]$ in its input integer channel $i()$. Thus, $x^1 = F_{f_0}(S^0, x^0) \in \mathbf{B}^N$ is computed by the network.
- This state x^1 is published as an output. Additionally, x^1 is sent back to the input layer, to act as Boolean state in the next iteration.
- At iteration number n , the recurrent neural network receives the state $x^n \in \mathbf{B}^N$ from its output layer and $i(S^n) \in [[1; N]]$ from its input integer channel $i()$. It can thus calculate $x^{n+1} = F_{f_0}(i(S^n), x^n) \in \mathbf{B}^N$, which will be the new output of the network.

In this way, if the initial state $x^0 \in \mathbf{B}^N$ is sent to the network with a sequence $S \in [[1; N]]^N$ applied in the input channel $i()$, then the sequence $(x^n)_{n \in \mathbb{N}^*}$ of the outputs is exactly the same than the sequence obtained from the following chaotic iterations: $x^0 \in \mathbf{B}^N$ and

$$\forall n \in \mathbb{N}^*, \forall i \in [[1; N]], x_i^n = \begin{cases} x_i^{n-1} & S^n \neq i \\ (f_0(x^{n-1}))_{S^n} & S^n = i. \end{cases}$$

From a mathematical viewpoint, the MLP defined in this subsection and chaotic iterations recalled above have the

same behavior. In particular, given the same input vector $((S^n)_{n \in \mathbb{N}^*}, x^0)$, they produce the same output vector $(x^n)_{n \in \mathbb{N}^*}$: they are two equivalent reformulations of the iterations of G_{f_0} in χ . As a consequence, the behavior of our MLP faithfully reflects the behavior of G_{f_0} which is chaotic according to Devaney.

B. Improving the Variety of Chaotic Recurrent Neural Networks

The approach proposed to build chaotic neural networks, explained in the previous subsection, is not restricted to an adhoc function $f_0: \mathbf{B}^N \rightarrow \mathbf{B}^N$, it can be generalized as follows. The function F_{f_0} associated to the vectorial negation f_0 , which has been recognized by the neural network, can be replaced by any functions $F_f: [[1; N]] \times \mathbf{B}^N \rightarrow \mathbf{B}^N$ such that the chaotic iterations G_f are chaotic, as defined by Devaney.

To be able to define functions that can be used in this situation, we must firstly introduce the graph of iterations of a given function $f: \mathbf{B}^N \rightarrow \mathbf{B}^N$, $x \mapsto (f_1(x), \dots, f_n(x))$.

Let be given a configuration x . In what follows the configuration $N(i, x) = (x_1, \dots, x_i, \dots, x_n)$ is obtained by switching the i -th component of x . Intuitively, x and $N(i, x)$ are neighbors. The chaotic iterations of the function f can be represented by the graph $\Gamma(f)$ defined below.

Definition 7 (Graph of iterations) In the oriented graph of iterations $\Gamma(f)$, vertices are configurations of \mathbf{B}^N , and there is an arc labeled i from x to $N(i, x)$ iff $F_f(i, x)$ is $N(i, x)$.

We have proven in [5] that:

Theorem 1 Functions $f: \mathbf{B}^N \rightarrow \mathbf{B}^N$ such that G_f is chaotic according to Devaney, are functions such that the graph $\Gamma(f)$ is strongly connected.

Since it is easy to check whether a graph is strongly connected, we can use this theorem to discover new functions $f: \mathbf{B}^N \rightarrow \mathbf{B}^N$ such that the neural network associated to G_f behaves chaotically, as defined by Devaney.

C. The Discovery of New Chaotic Neural Networks

Considering Theorem 1, it is easy to check that $f_1(x_1, \dots, x_n) = (x_1, x_1, x_2, \dots, x_{N-1})$ is such that G_{f_1} behaves chaotically, as defined by Devaney. Consequently, we can now obtain two chaotic neural networks by learning either F_{f_0} or F_{f_1} .

To support our approach, a set of illustrative examples composed of five neural networks is given. The three first networks are respectively defined by:

- $f_{0,1}(x_1, x_2, x_3, x_4) = (\overline{x_1}, \overline{x_2}, \overline{x_3}, x_4)$,
- $f_{0,2}(x_1, x_2, x_3) = (\overline{x_1}, \overline{x_2}, \overline{x_3})$,
- $f_{1,1}(x_1, x_2, x_3) = (\overline{x_1}, x_1, x_2)$,

while the last ones are defined by:

- $g_0(x_1, x_2, x_3) = (\overline{x_1}, x_2, x_3)$,
- $g_1(x_1, x_2, x_3) = (\overline{x_1}, \overline{x_2}, x_3)$.

TABLE I. OUTLINE OF THE RESULTS FROM SEVERAL ITERATION FUNCTIONS LEARNING USING DIFFERENT RECURRENT MLP ARCHITECTURES

Function	One hidden layer			
	8 neurons		10 neurons	
	Mean epoch	Success rate	Mean epoch	Success rate
$f_{0,2}$	82.21	100%	73.44	100%
$f_{1,1}$	76.88	100%	59.84	100%
g_1	36.24	100%	37.04	100%
	Two hidden layers: 8 and 4 neurons			
	Mean epoch number		Success rate	
$f_{0,2}$	203.68		76%	
$f_{1,1}$	135.54		96%	
g_1	72.56		100%	

Due to Theorem 1, the ANNs associated to $f_{0,1}$, $f_{0,2}$ and $f_{1,1}$ behave chaotically, as defined by Devaney. Whereas it is not the case for the networks based on the boolean function g_0 and g_1 , since $\Gamma(g_0)$ and $\Gamma(g_1)$ are not strongly connected.

D. Experimental Results

Among the five neural networks evoked in the previous subsection we decided to study the training process of three of them. Note also that for each neural network we have considered MLP architectures with one and two hidden layers, with in the first case different numbers of hidden neurons (sigmoidal activation). Thus we will have different versions of a neural network modeling the same iteration function. Only the size and number of hidden layer may change, since the numbers of inputs and output neurons (linear activation) are fully specified by the function. The neural networks are trained using the quasi-Newton L-BFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno) algorithm in combination with the Wolfe linear search. The training is performed until the learning error (MSE) is lower than a chosen threshold value (10^{-2}).

Table I gives for each considered neural network the mean number of epochs needed to train it and a success rate which reflects a successful training in less than 1000 epochs. Both values are computed considering 25 trainings with random weights and biases initialization. These results highlight several points. Firstly, various MLP architectures can learn a same iteration function, with obviously a best suited one (a hidden layer composed of ten sigmoidal neurons). In particular the two hidden layer structure seems to be too complex for the functions to be learned. Secondly, training networks so that they behave chaotically seems to be more difficult, since they need in average more epochs to be correctly trained. However, the relevance of this point needs to be further investigated. Similarly, there may be a link between the training difficulty and the disorder (evaluation of their constants of sensitivity, expansivity, etc.) induced by a chaotic iteration function.

V. CONCLUSION AND FUTURE WORK

Many chaotic neural networks have been developed for different fields of application, in particular for data security purpose where they are used to define ciphering methods, hash functions and so on. Unfortunately, the proposed networks are usually claimed to be chaotic without any

proof. In this paper we have presented a rigorous mathematical framework which allows us to construct artificial neural networks proven chaotic, according to Devaney. More precisely, a correspondence between chaotic iterations, which are a particular case of topological chaos in sense of Devaney, and MLP neural networks with a global feedback is established. In fact, we have shown that an iteration function is chaotic if its graph of iteration is strongly connected (a property easily checked), and that a global recurrent MLP can learn such a function. Future research will study more carefully the performance of the training process and alternative neural network architectures.

REFERENCES

- [1] V. Rijmen and E. Oswald, "Update on sha-1," Cryptology ePrint Archive, Report 2005/010, 2005, <http://eprint.iacr.org/>.
- [2] J. M. Bahi and C. Guyeux, "Topological chaos and chaotic iterations. Application to hash functions," Proc. IEEE World Congress on Computational Intelligence (WCCI 10), Jul. 2010, pp. 1–7.
- [3] —, "An improved watermarking algorithm for internet applications," Proc. 2nd Int. Conf. on Evolving Internet (INTERNET 2010), Spain: IEEE seccion Espania, Sep. 2010, pp. 119–124.
- [4] Q. Wang, J. M. Bahi, C. Guyeux, and X. Fang, "Randomness quality of CI chaotic generators. Application to internet security," Proc. 2nd Int. Conf. on Evolving Internet (INTERNET 2010), Spain: IEEE seccion Espania, Sep. 2010, pp. 125–130.
- [5] C. Guyeux, "Le désordre des itérations chaotiques et leur utilité en sécurité informatique," Ph.D. dissertation, Université de Franche-Comté, 2010.
- [6] J. M. Bahi, C. Guyeux, and Q. Wang, "Improving random number generators by chaotic iterations. Application in data hiding," Proc. Int. Conf. on Computer Application and System Modeling (ICCSAM 2010), Taiyuan, China, Oct. 2010, vol. 13, pp. 643–647.
- [7] Y. Li, S. Deng, and D. Xiao, "A novel hash algorithm construction based on chaotic neural network," Neural Computing and Applications, pp. 1–9, 2010.
- [8] I. Dalkiran and K. Danisman, "Artificial neural network based chaotic generator for cryptology," Turk. J. Elec. Eng. & Comp. Sci., vol. 18, no. 2, pp. 225–240, 2010.
- [9] S. Lian, "A block cipher based on chaotic neural networks," Neurocomputing, vol. 72, no. 4–6, pp. 1296–1301, 2009.
- [10] R. L. Devaney, An Introduction to Chaotic Dynamical Systems, 2nd Edition, Westview Pr., March 2003.
- [11] F. Robert, Discrete Iterations: A Metric Study, Spring Series in Computational Mathematics, Springer-Verlag, Ed., vol. 6, 1986.
- [12] J. Banks, J. Brooks, G. Cairns, and P. Stacey, "On devaney's definition of chaos," Amer. Math. Monthly, vol. 99, pp. 332–334, 1992.
- [13] C. Guyeux and J. M. Bahi, "Hash functions using chaotic iterations," Journal of Algorithms & Computational Technology, vol. 4, no. 2, pp. 167–182, 2010.
- [14] G. Cybenko, "Dynamic load balancing for distributed memory multi-processors," J. Parallel Distrib. Comput., vol. 7, no. 2, pp. 279–301, 1989.
- [15] Y. LeCun, L. Bottou, G. Orr, and K. Muller, "Efficient backprop," Neural Networks: Tricks of the trade, G. Orr and M. K., Eds. Springer, 1998.
- [16] K. Aihara, T. Takabe, and M. Toyoda, "Chaotic neural networks," Physics Letters A, vol. 144, no. 6–7, pp. 333–340, 1990.
- [17] N. Crook and T. O. Scheper, "A novel chaotic neural network architecture," Proc. (ESANN 2001), 2001, pp. 295–300.
- [18] N. Crook, W. J. Goh, and M. Harawat, "Pattern recall in networks of chaotic neurons," Biosystems, vol. 87, no. 2–3, pp. 267–274, 2007.