Reservoir Computing for ECG Heartbeats Classification

Imane Fadli, Afaf El Kalai, Nadia El Hanine
École Centrale Casablanca, Casablanca, Morocco
*Authors contributed equally to this work

ABSTRACT

Reservoir computation is a new computing paradigm allowing sequential data processing by a non-linear dynamic system. Derived from several models of recurrent neural networks, including echo-state networks and liquid-state machines, reservoir computing allows obtaining very constructive results with a relatively limited implementation and training cost, in several domains such as speech recognition, video processing and time series prediction. With this new approach, learning is performed only on the output layer while the hidden layer, the reservoir, is created randomly. The idea has been expanded by affirming that any large dynamic system, controlled, operating in the correct dynamic regime, can be utilized as a reservoir. This is called physical reservoir computing, which has attracted increasing attention in a variety of research fields. The aim of this review is to describe the theoretical bases of CR, to provide a description of its different variants and to study its applicability for the classification of heartbeats by ECG, to estimate more precisely the efficiency of this approach in the detection of heart rhythm anomalies, which was considered impossible before the appearance of CR due to the temporal characteristics of heartbeats.

Keywords: Reservoir computing, Echo state networks, Liquid machine networks, Backpropagation, electrocardiogram signal, classification.

INTRODUCTION

It has been already fifteen years old since the apparition of the first ideas and models of Artificial neural networks. McCulloch-Pitts threshold neurons,the first generation of artificial neural networks consisted a conceptually very simple model: If the sum of the weighted incoming signals of a neuron exceeds the threshold value ϑ, the neuron sends a "high" binary signal. Even if these neurons can only provide digital output, they have been strongly used in mighty artificial neural networks such as multilayer perceptronsand Hopfield networks.

Second generation neurons do not use threshold function, but rather use other continuous activation function such as Sigmoid $\sigma(y) = \frac{1}{1+e^{-y}}$. and hyperbolic tangent $tanh$. A typical example of a neural network composed of neurons of these type are feed-forward and recurrent neural networks. They are more powerful than their predecessors of the first generation: when the output layer is equipped with a threshold functions, second generation neurons made a universal digital computing model with fewer neurons than the first generation neural networks. Moreover, they can arbitrarily approximate any analog function. Actually, it has been shown that these nets are universal for analog computations. In fact, any continuous function with a compact domain and range can be approximated arbitrarily well (using uniform convergence), e.g.L_∞ with a single hidden layer. In addition, due to the use of a continuous activation function, this second generation of

neural network models does support gradient-descent learning algorithm like backward propagation.

The third-generation neural network has proposed a model approximating the biological reality using spikes [9]. This allows to incorporate spatio-temporal information in communication and computation, as real neurons do. In particular, it has been shown that networks of spiking neurons are, regardless of the number of neurons required, more computationally powerful than other neural network models. Neurons of these networks use the so-called pulse coding, a mechanism in which individual neurons send out erratic sequences of peaks, called spike trains, whose frequency varies considerably over a short period of time. So, neurons are able to extract temporal and spatial information from the incoming spikes and encode their message to other neurons.

Recurrent Neural Networks (RNN) to Reservoir Computing (RC)

Recurrent neural networks has known a significant evolution during this last years. While it is hard to train a RNN, recent researches have shown that training the readout layer is sufficient to obtain good performances. This configuration, called Reservoir Computing, makes it easy to achieve both peak performance and fast learning, resulting in low training cost.

Artificial neural network architectures [3] are generally classified into feedforward networks (Schmidhuber, 2015) and recurrent networks (Mandic,Chambers, et al., 2001), each serving a specific type of computational task. Feedforward neural networks (FNNs) are mainly used for static data processing, because although the individual input data are given sequentially, they are processed independently. In short, FNNs are capable of approximating non-linear input and output functions. In contrast, Recurrent Neural Networks (RNNs) are suitable for dynamic (temporal) data processing because they can integrate the temporal dependence of inputs into their dynamic behavior. Thanks to their feedback connections, RNNs are specifically able to represent dynamic systems driven by sequential inputs.

The first types of RNN were developed in the early 80's: every neuron is directly connected to all other units in the network. Some neurons being designated as input neurons, others as input neurons, others as output neurons, and the rest as hidden neurons. All the following RNNs are specific cases of this architecture.

The networks of Hopfield developed by John Hopfield in 1982 have symmetrical connections and are not used for processing sequences of patterns but independent inputs. The network is driven by a learning Hebbien and the convergence of its dynamics is assured. Elman's networks are another type of RNNs, structured in 3 layers similar to a network in normal layers, but with the addition of external units connected to these three layers, maintaining a memory of the values previously taken by the hidden layer. These networks maintaining a state output can perform some tasks of prediction of sequences that are beyond the reach of conventional layered networks.

The classical method for training an RNN derives from the classical approach of the gradient descent. The weights of the units W(between every two connected neurons) are iteratively adapted according to the gradient $\frac{dE}{dW}$ in oder to minimize an error function $E(y,\hat{y})$ between the output taken by the network y and the desired output $\hat{y}$ We will be giving further detail about another approach: Back Propagation in the following sections.

A new approach was proposed in the Atiya-Parlos Recurrent Learning (APRL). It allows faster convergence than previous methods by updating the network parameters by estimating the gradient $\frac{dE}{dW}$ in relation to the activation of neurons afin to push them in the right direction.

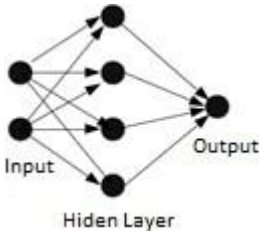


Figure 1 : Layer Network Figure 2 : Recurrent Neural Network

RNNs remain promising and fascinating, as they are able to approximate any dynamic system and resembling the biological neural networks by their possessions of cycles. It is in this context of slow progress and difficiles that a new approach was discovered independently. Herbert Jaeger in the early 2000s, in the field of machine learning under the name of Echo State Network, and by Wolfgang Maass in the field of computational neurosciences under the name Liquid State Machine. It was only later that these two approaches have introduced a new paradigm in artificial recurrent neural networks (RNNs) known as "**reservoir computing**".

Below is a summary of the history of these techniques[3] starting from difficulties to train RNNs to the distinction of the reservoir and the outputs in the Reservoir Computing:

1. **1990-2000:** Methods based on gradient descent and derived from the gradient back-propagation algorithm such as BackPropagation Through Time (BPTT) are used to train of the RNNs but allow only slow, expensive and restricted learning to small networks.
2. **2000.** The Atiya-Parlos Recurrent Learning (APRL) algorithm accelerates significantly the learning speed by allowing a dynamic update much slower for neurons of the reservoir than output units.
3. **2001.** The application of simple linear methods on a recurrent layer of neurons allows a very fast and efficient learning leading to the Echo State Network (ESN) model.
4. **2002**. Neurosciences discover recurrent structures linked to output layers in the micro-columns of the cortex leading to the Liquid State Machine (LSM) model.
5. **2004.** The BackPropagation-DeCorellation (BPDC) algorithm deriving from the previous observation and which allows an acceleration of the learning process by applying APRL only on the weights of the output units.
6. **2007**. The domain is unified under the name Reservoir Computing.

In this paper, we will first elaborate the Reservoir Computing formalism while elucidating its models. Then, we will shed the light on the construction of the reservoir including its properties, the optimization of its parameters and the training phase. At the end of this paper, we will be in position to develop a Python deep learning program based on RC approach allowing the classification of the heartbeat signals with a high accuracy of nearly 95%.

RESERVOIR COMPUTING FORMALISM

RC system [7] consists of a recurrent neural network made up of three layers, an input layer, a hidden layer based on a non-linear dynamic system called reservoir whose connections do not need to be trained and an output layer. The input layer feeds data into the reservoir layer where it is mapped on a high dimensional state space. The output signal is simply a linear combination of internal states of the reservoir at a given time, which depend on the input data and the system internal dynamics. The concept of training is shown in greater detail in the section Training the Reservoir. In particular, RC fully meets the demands for low training cost and real-time processing that former artificial neural network models couldn't meet.

Recently, a new data processing scheme that exploits physical dynamics as a computational resource instead of RNNs has been proposed. This scheme is called physical reservoir computing (PRC). In principle, any dynamical system has the potential to serve as a reservoir if it can exhibit dynamical responses to inputs. The PRCs have attracted increasing attention in diverse fields of research. And hardware implementation can achieved using different types of physical systems, substrates, and devices.

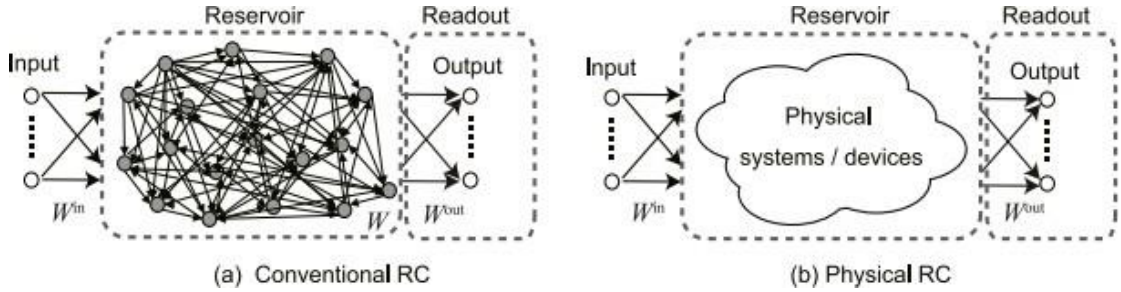


Figure 3 :a) conventional RC system with an RNN-based reservoir as inESNs and LSMs. (b) . (b) A physical RC system in which the reservoir is realized using a physical system or device.

Representative models of Conventional RC

In This section, we will present the major Conventional RC models, namely: Echo State Networks (*ESN*), Leaky-Integrator ESNs (*Li-ESNs*) and Liquid State Machine (*LSM*)

Echo State Networks

The ESN model was proposed by Jaeger [2], it is used as supervised learning algorithms. Consider a given system, consisting of M input units, N internal nodes, and L output units. Consider $u(t)$ the input unit at time t, $u(t) = [u_1(t), u_2(t), \dots, u_M(t)]^T$, $x(t)$ the state of reservoir at t time, $x(t) = [x_1(t), x_2(t), \dots, x_N(t)]^T$,$y(t)$the output units, $y(t) = [y_1(t), y_2(t), \dots, y_L(t)]^T$.

The connexion weights between internal nodes are given by a matrix noted W of size $N \times N$, an $N \times M$ matrix noted $W^{in} = (w_{ij}{}^{in})$ give the connexion weights from the input units to reservoir, an $L \times N$ matrix noted $W^{out} = (w_{ij}{}^{out})$ give the connexion weights from the reservoir to the output units and an $L \times N$ matrix noted $W^{back} = (w_{ij}{}^{back})$ give the connexion resulted from backpropagation training from the output nodes to reservoir nodes.

ESN is based on a property named Echo State Property ESP. This property requires the reservoir states be a function of the previous input sequence in order to assure the response of the system will be the same for a given input at the different units of time [1]. In addition, the $x(t)$ vector can be considered as a spatial transformation of the $u(t)$ signal history [3]. The goal is to minimize the error between the data to be learned $y(t)$ and the $\hat{y}(t)$output learned by the network.

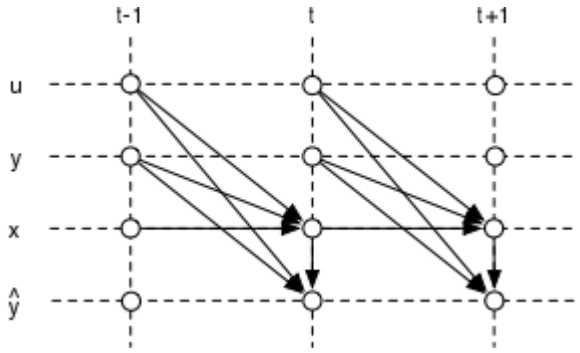


Figure 4: The Figure shows the exact timing setup in case of teacher-forced operation [4].

The network activation vector is governed by the following equation [3]:

$$x(t+1) = f\,(W^{in}u(t+1) + Wx(t) + W^{back}y(t)) \qquad (1)$$

where $f = (f_1, f_2, \ldots, f_N)$ is vector of the reservoir activation functions; they could be sigmoid functions, hyperbolic tangent functions, identity functions or Heaviside function [3].Biasescan be added to the units of the equation (1) to give the full model of the ESN

$$x(t+1) = f(W^{in}u(t+1) + Wx(t) + W^{back}y(t) + W^{bias})$$

Concerning the outputs, optional connections can exist directly between the inputs.
and the outputs of the network, the outputs are then defined such as:

$$\hat{y}(t) = f(W^{out}[x(t); u(t)])$$

Where W^{out} is $L \times (N + M)$ matrix and $[.\,;\,.\,]$ is defined as the vertical concatenation of two vectors [3].

Leaky-Integrator ESNs (Li-ESNs)

This model adds another parameter to the equations of the dynamical reservoir system, this parameter is a leaking rate which represents the leaking into the reservoir and which should be optimised [5].Formally, the dynamics of a continuous time Li-ESN is defined in the following equation [3].

$$\dot{x} = \frac{1}{r}\,(-ax + f\,(W^{in}u(t) + Wx(t) + W^{back}y(t)))$$

$$\hat{y}(t) = f(W^{out}[x(t); u(t)])$$

Where $r > 0$ is a time constant global to the ESN, $a > 0$ is the reservoir neuron's leaking rate (we assume often a uniform leaking rate for simplicity) and the other parameters have the same definition given above.

We obtain the following discrete network update equation for dealing with a given discrete time sampled input $u(n\delta)$ by using an Euler discretization with stepsize δ of this equation [5]

$$x(n+1) = (1 - a\frac{\delta}{r})\, x(n) + \frac{\delta}{r} f\,(W^{in}u((n+1)\delta) + Wx(n) + W^{back}y(n))$$

$$y(n) = f(W^{out}[x(n); u(n\delta)]$$

This model is slightly complicated than the first one. However, it is more adequate to use when timescale phenomena are considered. In addition, caution should be taken in choosing the step time of Euler discretization.

Liquid Machine State

A liquid state machine (LSM) is a kind of reservoir computer using a spiking neural network. An LSM is composed of a large number of units (called nodes or neurons). Each node gets a time-variable input from external sources (the inputs) as well as from other nodes. The nodes are connected to each other at random. The recurring nature of the connections converts the time-variable input into a spatio-temporal scheme of activations in the nodes of the network. The spatio-temporal activation models are read by linear discriminant units.

The soup of recursively connected nodes will eventually compute a wide variety of non-linear functions on the input. Given a sufficiently large variety of these nonlinear functions, it is in theory possible to get linear combinations (using the read units) to accomplish any mathematical operation needed to perform a certain task, such as speech recognition or computer vision.
LSMs have been suggested as a way to describe how the brain works. LSMs are considered advancement over the theory of artificial neural networks because:

- Circuits are not hard-coded to accomplish a specific task.
- Continuous time inputs are treated "naturally".
- Calculations on various time scales can be performed using the same network.
- The same network is able to perform multiple calculations.

The LSM model comes from the neurosciences and is based on multiple loops that form a large complex network comprising more than 80% of all synapses in a neocortical column and shows that multiple outputs can be trained to perform different tasks on the same sequence of states. The computational model no longer requires convergence to stable internal states or attractors,but is based on information about the input signals automatically captured in the perturbations. of the dynamic system. This series of perturbations cause the excitation of a medium in the same way that wavelets produced by perturbations on the surface of a liquid form a representation of these perturbations that fades with time, hence the name Liquid State Machine. Formally, an LSM is defined by the state of the liquid at time t:

$$x_t^M = (L^M u)_t$$

where $L^M$is the set of liquid filters, u_s is the input signal at time s ≤ t. The output of the network can then be computed using an output function $f^M(.)$ for the target function *y(.)*[3].

$$\hat{y}_t = f^M(x_t{}^M)$$

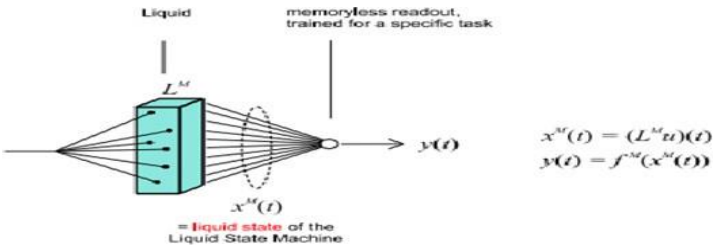


Figure 5: Properties of liquid state machine

LSM must satisfy two properties:

- ✓ Separation property for a basic filter class β: if we have two entries (.) et v(.) with$u$(.) ≠v(.) then whatever$b\in\beta$ with a basis filter β we have $bu(t) \neq bv(t)$.
- ✓ Property Approximation: the ability of the reading units to discern the trajectories of an input stream and relate them to the target results.

CONSTRUCTION OF THE RESERVOIR

Reservoir dynamics and properties

Reasoning property:

A reservoir is a dynamic system, therefore it has the potential to adopt chaotic behaviors, in this case the input signal loses all influence in front of the initial conditions, leading to a drop in its performance. The reasoning property or Echo State Property [10] provides that the effect of previous states and previous entries on future states should gradually disappear as time passes (k →∞) and not persist or be amplified. Thus, the sequential outputs are represented with a short-term dependencies.

The frontier of Chaos:

The chaos frontier is a region of parameters of a dynamic system in which it operates at the boundary between chaotic and non-chaotic behavior. It is commonly accepted in the literature that a reservoir at this boundary has a high computing power and a maximum memory. This point has also been the subject of some confusion in the application of Reservoir Computing, which led to the widespread application of dimensioning the spectral radius so that it is close to 1, but lower, so that the dynamics of the system is close to the border of chaos. Indeed, results in the field of LSMs seem to indicate that reservoirs operating in a regime between stable and chaotic behavior have optimal

computational performances [11]. Moreover, in [12], it is suggested that an ESN is optimal in a stable regime but close to chaos, which would allow it to display rich dynamics without drowning the information. The term "chaos frontier" originated in early work analyzing the chaotic behaviors and phase transitions of cellular automata whose transition tables are modified to make them more or less homogeneous [13]. By their nature, systems such as cortical networks or gene regulation networks operate at a dynamic close to chaos.

Exhibitor of Lyapunov:

Lyapunov's exponent [14] is a way to measure the stability (or chaoticity) of an orbit across the state space of a dynamic system.

The Lyapunov exponent measures the exponential deviation of a trajectory in a certain direction resulting from an infinitesimal perturbation in the state of the system.

If the Lyapunov exponent is less than 0, then the effect of the perturbations if it is greater than zero then the system will deviate exponentially from the orbit. If the system evolves in a k-dimensional space, then it will have k Lyapunov exponent(s). For an orbit to be chaotic, no exponent must be equal to zero and at least one must be greater than zero. A method for analytically calculating the Lyapunov exponent for a reservoir of sigmoid neurons is presented in [15], it is based on the Jacobian matrix of the reservoir for a sequence of xt states. Since this measure cannot be computed for $t \rightarrow \infty$, it is only an approximation and is therefore called Lyapunov pseudo-exponent by the authors. In [11], the Lyapunov exponent is empirically measured for a LSM by calculating the mean Euclidean distance of the reservoir states resulting from a time shifted pulse sequence. In [16], the authors introduce the local Lyapunov exponent which allows to estimate the predictability and excitability of a dynamic system around a given point in the state space and apply it to some tasks. Their results show that this measure accurately predicts the performance of a reservoir, and that a trade-off is made between the maximum gain of the system (its excitability) and its degree of freedom in the state space. It should be noted that it has been empirically shown that the best performance for a given task corresponds to a Lyapunov exponent specific to that task [15].

Optimization of reservoir parameters

The construction of an optimal reservoir requires making the right choice among a set of parameters. This choice is most of the time made manually and requires a certain amount of know-how. This section gives an overview and some basic rules to create a reservoir with good performance.

The following parameters will be discussed:

- The size of the reservoir.

- Connectivity of the reservoir(sparsity).

- The spectral radius.

- Input scaling.

- The leaking rate α.

Using an exhaustive search is the most efficient method to search in the parameter space. Reservoir performance should be evaluated empirically by performing the learning phase and then by measuring his error on the task. If you optimize the parameters by hand, change only one parameter at the same time. [18] proposes a guide and a set of rules to be applied in order to achieve good performance. Anyone interested in more information on optimizing reservoir parameters can refer to it.

Reservoir size

The size of the reservoir determines the capacity of the reservoir, whether it is the computing or memory capacity. As with neural networks, too many units can lead to overfitting and a time-waste. To avoid overfitting, it is advisable to use Tikhonov regularization. For more details about Tikhonov regularization refer to section : Training of the reservoir.

The dimensionality of a reservoir is related to its size. High dimensionality is one important prerequirement to be satified. In classification tasks, the reservoir should content a huge number of nodes such that guarantee the nonlinearity transformation of sequential inputs into a high-dimensional space and facilitate the separation of inseparable inputs.

Experiences show that, for a given task, progress is becoming more and more minimal as units are added. Passing a certain number of units, increasing the size of the reservoir increases the calculation time by allowing only small progress. On the other hand, the performances of large reservoirs show less variation.

Concerning the optimization of a reservoir, the parameters vary only slightly with the size, it is recommended to optimize them on a small reservoir and then create larger ones with the same parameters. At last, remember that the memory capacity of a reservoir cannot exceed its number of units, this is to be taken into account in order to integrate enough unitsto respond to the needs of the task at hand.

Input Scaling

As mentioned in previous sections, the inputs participate in the dynamic of the reservoir. Thus, a reservoir with a spectral radius greater than1.0 can remain in a non-chaotic mode with appropriate inputs or push the reservoir in regions far from the state trajectories for which the parameters have been optimized and for which the outputs were trained. It is therefore advisable to normalize the data in order to keep the input signal in controlled terminals and avoid outliers. This also avoids saturation of the units of the reservoir, unpredictable outputs and possible loss of memory.

More importantly, the scale of magnificationW^{in} in which is applied to the inputs **regulates the non-linearity** of the reservoir. A small-scale matrix W^{in}will push the activations of the units in the central part of the tanh(x) function, whereas too large-scale matrix may saturate the units.

Spectral Radius

Concerning the spectral radius, it is advisable to follow the method proposed in section 3.1 in order to obtain a W matrix guaranteeing the reasoning property. It is also advisable to take into account the dynamics of the system at the edge of the chaos and to try to optimize the spectral radius to create a dynamic corresponding to the needs of the task. Once again, the optimal spectral radius is

not always located at the border of chaos. Indeed, the spectral radius is linked to the computing and memory capacity of the reservoir which seems to arrive at a compromise on the edge of chaos. But some tasks need more memory than computational or conversely, the spectral radius must therefore be higher for tasks requiring an important memory.

Leaking rate

The leaking rate changes the dynamics of the system and can be interpreted as a range of time of a continuous system between two time steps in a discrete system. In practice, changing the leaking rate is comparable to resembling the input signal. It is also important to note that this parameter can significantly increase the short-term memory of the reservoir. This parameter must push the system into a dynamic similar to that of y. For this purpose, the empirical approachremains the best. If our reservoir has to be to adapted to several time scales, it is recommended to use a different leaking rate for each unit in the reservoir.

Sparsity or inputDensity

Sparsity is a parameter that specifies the number of connections between neurons of the reservoir. High sparsity indicates that the units are all connected to each other, while low sparsity indicates that each unit is connected to only a small number of other neurons. Connectivity indicates that each unit is connected to only a small number of other neurons. A reservoir with low connectivity shows slightly better performance but affects only the capacity of the reservoir. However, a weakly connected reservoir has a matrix which most of the elements are null. Thus, with a representation of the sparse matrices, a weakly connected reservoir allows to calculate updates to its states more quickly. It is therefore advisable to use a reservoir (10%), regardless of its size, in order to save calculation time.

Training of the reservoir

The learning phase consists of solving a system of equations that aims to minimise the error $E(Y,\ W^{out}x)$.This training can be done in different ways, using either a back propagation approach, linear regression algorithm, or Tikhonov regularization.

Gradient backpropagation is a method for calculating the error gradient for each neuron in a neural network, from the last layer to the first. It consists in calculating the error between the output
provided by the network and the one we want to have. We try to vary the input signal for each neuron, so as to have a minimal error. In particular, the purpose of the gradient algorithm is to iteratively converge towards an optimized configuration of synaptic weights. In the case of neural networks, synaptic weights that contribute to generating a large error will be modified more significantly than weights that generated a marginal error.

The weights in the neural network are initialized beforehand with random values. Then we consider a data set to be used for learning.. Each sample has its target values, which are those that the neural network must eventually predict when presented with the same sample.

The algorithm follows the following steps:

- Let a sample be presented by $\vec{x}$at the input of the neural network and $\vec{y}$the desired output for this sample.

- The signal is propagated forward in the neural network layers: $x_k^{n-1} \rightarrow x_i^n$, with n the layer number. x_i^n is the output of neuron i of layer n, which is also the input to the neurons of the next layer.
- Forward propagation is calculated using the activation function g, the aggregation function h (often a scalar product between weights and neuron inputs, h_i^n is actually the input signal to neuron i of the layer n) and synaptic weights w_{ik} between neuron k of layer $n-1$ and neuron i of layer n.

$$x_i^n = g(h_i^n) = g(\sum_k x_k^{n-1} w_{ik})$$

- When forward propagation is completed, the output provided by the algorithm is $\vec{y}$.
- We then compute the error between the network output $\vec{y}$ and the desired output for this sample $\hat{\vec{y}}$. For each neuron i in the output layer, we calculate (g' being the derivative of g) :

$$e_i^{output} = (y_i - \hat{y}) g'(h_i^{output})$$

- The error is propagated backwards $e_i^n \rightarrow e_k^{n-1}$ using the following formula:

$$e_j^{n-1} = (\sum_i e_i^n w_{ij}) \times g'^{(n-1)}(h_j^{n-1})$$

Note that we can associate to each layer a specific activation function, hence the notation $g^{(n-1)}$.

- Finally, We update the weights in all layers :

$$w_{ij}^l = w_{ij}^l - \delta e_i^l x_j^{l-1}$$

δ : Learning rate ; how much of update we want to apply to the weights during training, often in the range between 0 and 1.

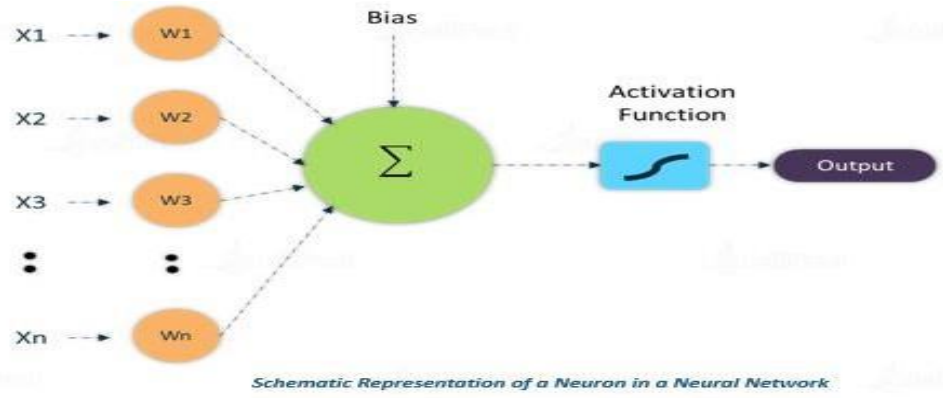


Figure 6: The connections between different units of a RNN and concept of feed forward training

In linear regression, the matrix of the output weights $W^{out}$is defined by the following equation:

$$W^{out}x = y$$

Where X is the $IR^{N_x \times T}$matrix that contains the states of the reservoir during the learning phase, and Y is the $IR^{N_y \times T}$matrix that contains the expected outputs for each output over time, with$N_x < T$.

Expressing the problem as a normal equation, we get:

$$W^{out}xx^T = yx^T$$

A naive solution to find $W^{out}$is then

$$W^{out} = yx^T(xx^T)^{-1}(27)$$

As referred in paper [20] , the complexity of this solution does not depend on the size of the training set. To reduce sensitivity to noise and overlearning, we can add a regularisation parameter λ; an additional cost to least-square optimization in order to minimise the amplitude of the matrix weights outW, to equation (27):

$$W^{out} = yx^T(xx^T + \lambda\, \mathrm{I})^{-1}$$

where I is the identity matrix $IR^{Nx\times N_x}$.

In [17], **Tikhonov's methods** of noise injection and regularisation are compared on signal generation and time series prediction tasks. The results show the superiority of Tikhonov's regularisation of noise injection. Indeed, an improvement in the results of the order of 30% is obtained with the latter in terms of precision and stability. In addition, the regularisation of Tikhonov does not need to recalculate the reservoir states for each configuration, contrary to the addition of noises, in order to find the optimal parameter but only to recalculate the weights for different regulation parameters λ.

APPLICATION : ECG HEARTBEATS CLASSIFICATION

In this part, we highlight one of the crucial applications of the RC techniques. The detection of abnormalities in the heart rhythm is under development shift. The challenge lies in their sensitivity to variations in morphology and the temporal characteristics of heartbeats. Thus, the adaptation of RC for the processing of cardiac signals proves to be useful in view of their characteristics mentioned above.

In our study, we have used **MIT-BIH Arrhythmia Database** [19] composed of electrocardiogram ECG signals shaped form normal and affected patients with arrythmia diseases. Each row is a series of 187 heartbeats and which the 188 element indicates the class to which this signal belongs. There are five classes: 0: **'normal beat'**, 1: **'unknown Beats'**, 2: **'Ventricular ectopic beats'**, 3: **'Supraventricular ectopic beats'**, 4: **'Fusion Beats'**. (Figure 6)

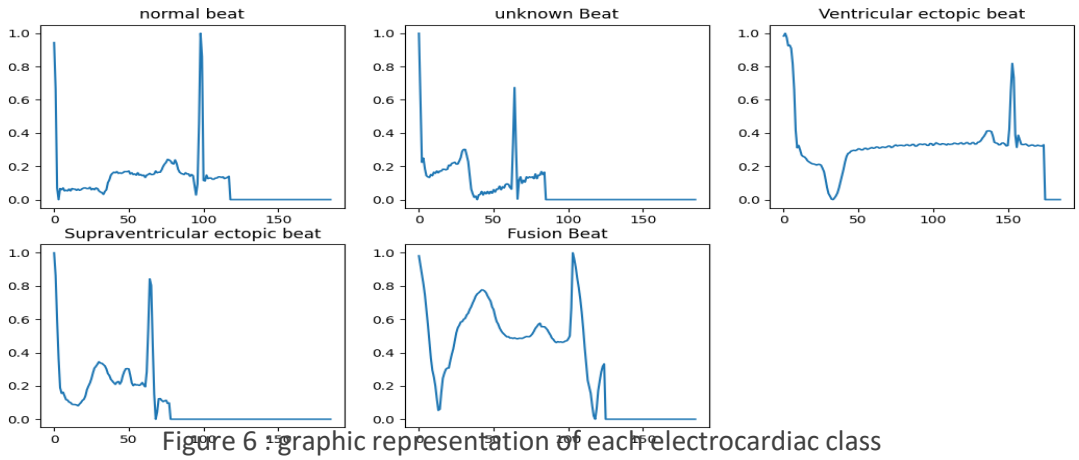


Figure 6 : graphic representation of each electrocardiac class

The main toolbox used in this study is :

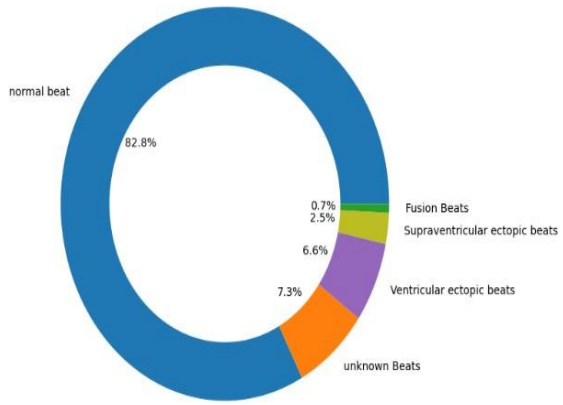
-easyESN: Python library for recurrent neural networks utilizing Echo State Networks (ESNs, also called "reservoir computing") with an easy-to-use high-level API that is closely "sklearn" oriented. It is designed to make the use of ESNs simpler by offering algorithms for the automatic adjustment of the hyperparameters as a function of gradient (peak regression penalty, spectral radius, leakage rate and feedback scale), as well as transient time estimation. In addition, it permits the use of spatio-temporal NSEs for the forecast and classification of geometrically extended input signals.

- Sklearn: Python module that includes classical machine learning algorithms in the closely connected world of Python scientific software packages (numpy, scipy, matplotlib). It is developed to offer simple and effective solutions to learning problems, available to all and reusable in a variety of contexts: machine learning as a versatile tool for science and engineering.

- Pandas: Python software that gives fast, flexible and expressive data structures that are designed to make working with structured data and time series both easy and intuitive. It is designated to be the high-level foundation for performing practical, real-world data analysis in Python.

- Matplotlib: a library for drawing and visualizing graphs.

After reading the CSV files into our idle, we've defined the features and targets and determinate the data's composition (figure 7). It seems that the data is imbalanced in terms of class. Thus, we've chosen randomly 2000 signals of each class (figure 8). Then, we constructed our model with the predefined function named PredictionESNusing as a first attempt**187 input nodes**, **750 reservoir nodes** and **1 output node**. We reset the **spectral radius to 0.98** and the **leaking rate to 1**.the first performance of the model obtained is **accuracy=0.662** and **MSE=0.44**.



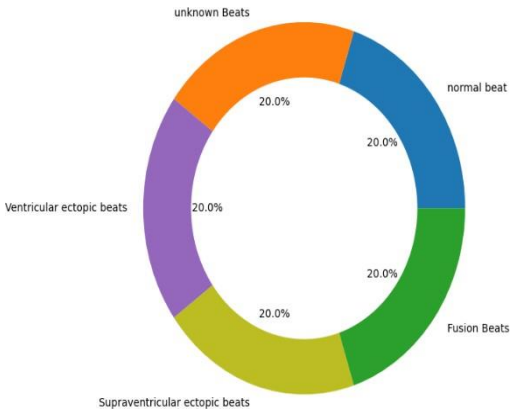


Figure 7: graphical representation of the initial data Figure 8: graphical representation of the new data composition

Once our model is defined, the next step is to choose the best combination of our model parameters that lead to the best performance. As we already said, If you optimize the parameters by hand it is highly recommended to change only one parameter at the same time.Therefore, we opted for a parameter optimization method which consists in visualizing the variation in the performance of our model as a function of a few parameters and then finding the values of these parameters that maximize the score (accuracy) or minimize the root mean square error (MSE).

The first graph, figure 9, validates the theory of parameter optimization explained in the section **Optimization of reservoir parameters;** increasing the reservoir size increases the model accuracy, but from a certain range the accuracy becomes constant even if the reservoir size is expanded.

The second and third graphs, figure 9 and 10, shows that the model accuracy increases (respectively the MSE decreases) with the spectral radius, but once it approaches 1 the accuracy starts to decrease. This can be explained by reaching chaos frontier. As we have stated in the section **Reservoir dynamics and properties,** we should set the spectral radius close to 1 but lower so that we don't exceed the border of chaos where the dynamics of the system will have a chaotic behavior.

The third graph, figure 11, shows that the optimum leaking rate to use is around 0.1.

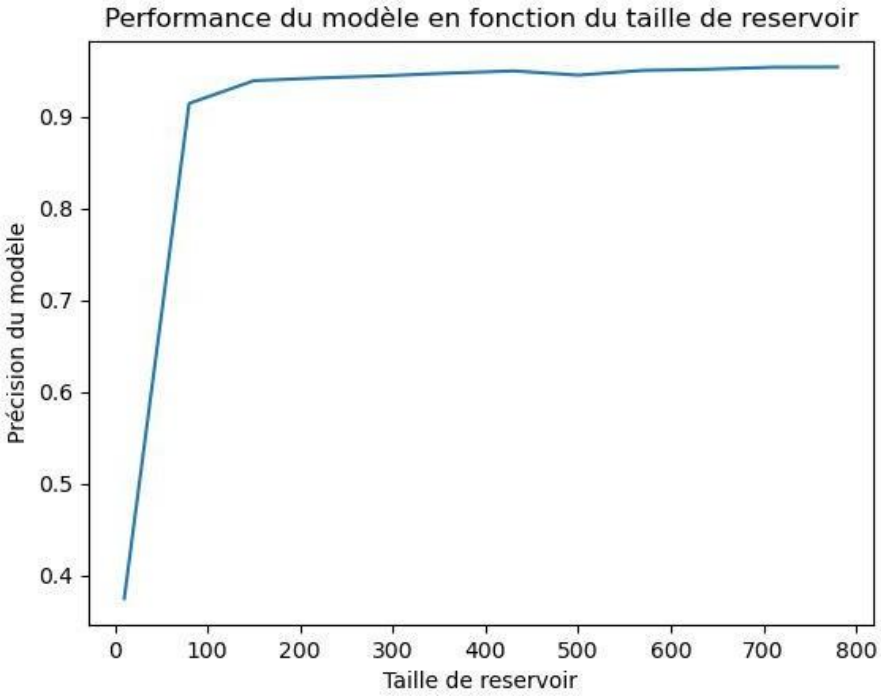


Figure 9 : model accuracy as a function of reservoir size.

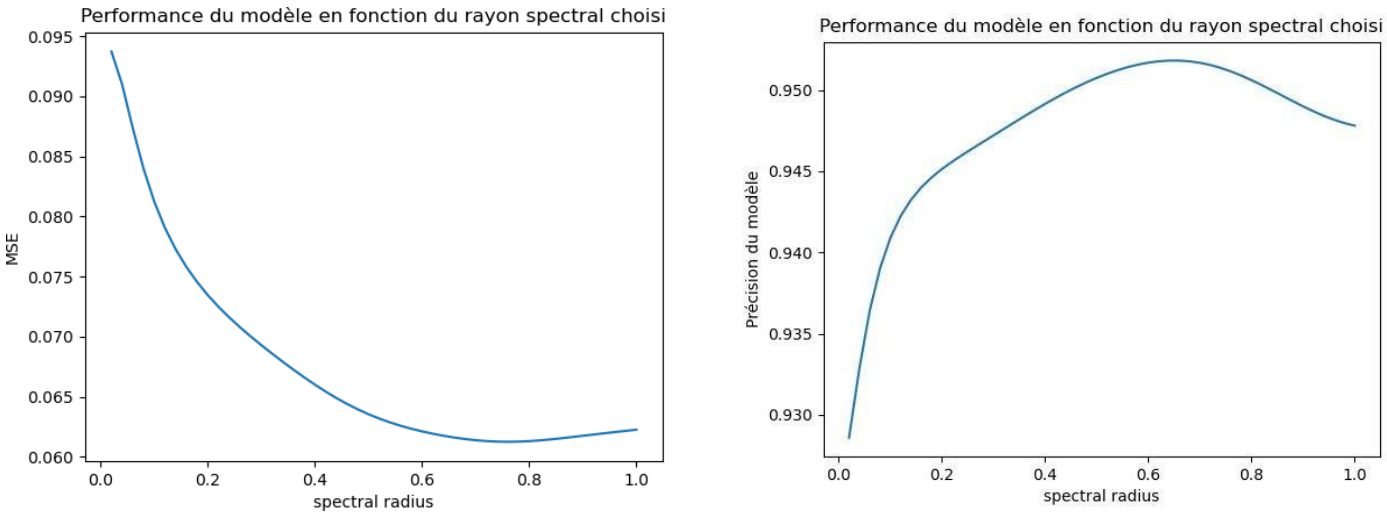


Figure 10: the error and the accuracy as a function of the spectral radius

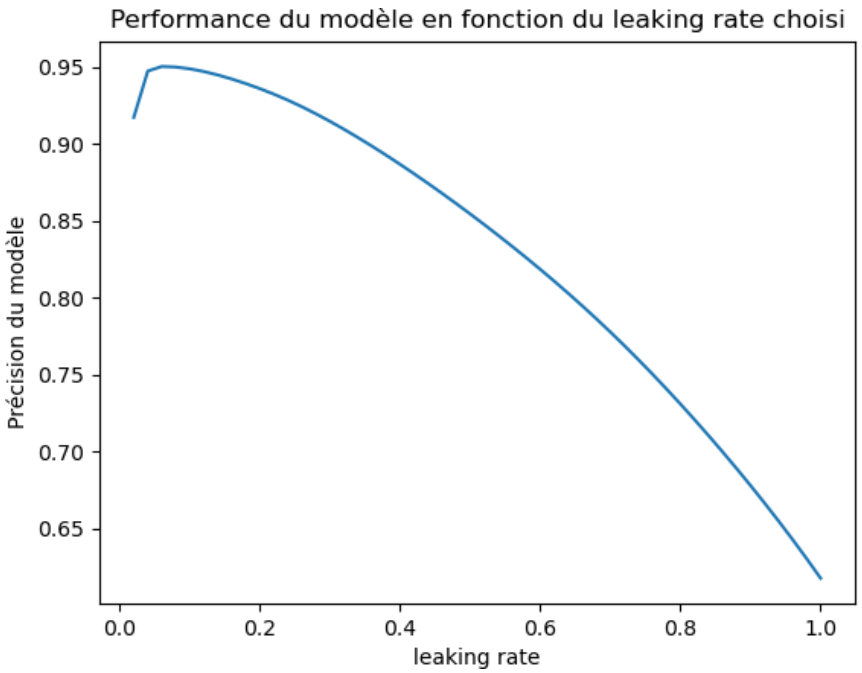


Figure 11: model accuracy as a function of leaking rate.

Resultat :

After proceeding as stated above, we achieveda **high accuracy : 95%** and **low MSE :0.06.**by setting the parameters in :

input nodes = 187

output nodes = 1

Reservoir size = 300

Spectral radius = 0.7

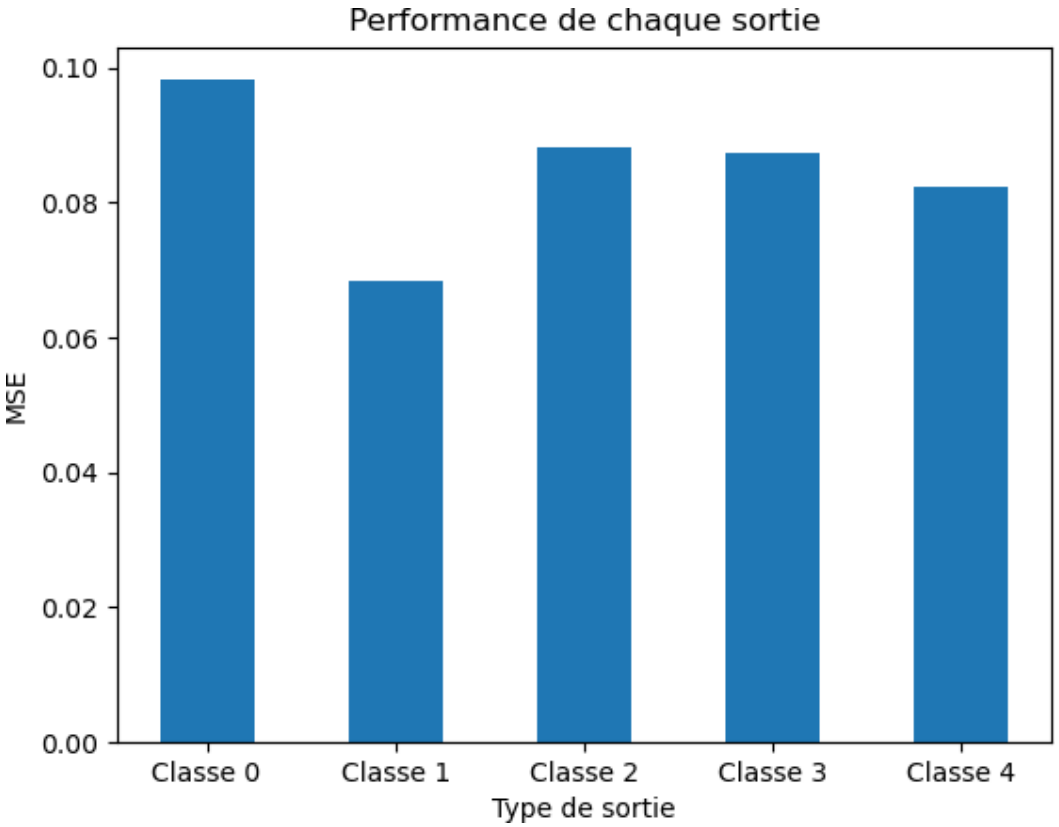
Leaking rate = 0.08

We therefore obtain a very constructive result with a relatively limited training cost and real-time processing (20 secondes).

Comparative Analysis with Related Works:

In [20], a comparative study which recapitulates the accuracy obtained by other classifiers, who apply ANN as a machine learning model, are done. This study shows these classifiers are effective to the tune of about 98.7% - 93.8 % accuracy. The classification made by the learning algorithms Bayesian-Regularization (trainbr) on the same data that we used, has 93.8% as the most accurate result .

It is well to mention that several working conditions must be taken into account for a fair comparison, such as the choice of database, the duration of ECG recordings, the number of classes to be classified, and its number of decompositions and many others.



CONCLUSION

Reservoir Computing remains an exciting and promising field that still requires a great deal of research in order to improve performance beyond the creation of simple reservoirs. The main purpose of this article is the exploration of reservoir computing theory, computational technique derived from Recurrent Neural Networks. It consists of creation of reservoir containing nodes which are randomly connected and allowing the inputs mapping into a high dimensional space while responding the limitations of the RNN Model, notably the low precision when it comes to data that vary very quickly in time and space. Therefore, we have exploited this technique to develop a training model for the classification of cardiac signals and succeeded in reaching an accuracy of 95%

FURTHER APPLICATIONS:

- Since a robot is a dynamic system, Reservoir Computing [3] has been widely applied to the field of robotics in order to solve certain tasks such as motor control or the interaction of the

robot with its environment. An Echo State Network type network has been used to control a robot, without a rigid structure and inspired by an octopus, and to exploit to the maximum the flexible and dynamic nature of its body. Moreover, the control of the direction of a robot's wheels as well as its acceleration is studied from the angle of Reservoir Computing. Then, Reservoir Computing is able to determine the location of a robot from the sensory data alone, which is studied so that it can create an implicit map of its environment.

- In the medical field, the capabilities of Reservoir Computing have been evaluated to detect epileptic seizures in laboratory rats, to transform the activity of a neuron in a space of high altitude, and to detect the presence of epileptic seizures in a space of high altitude dimension in order to determine the position of the hand and to filter the data from an MRI.
- The Reservoir Computing paradigm has been studied in the field of chaotic series prediction and more particularly in the financial field. In a comparative study evaluates the efficiency of different strategies and different types of reservoirs in order to model financial time series. Different strategies include the use of a single reservoir or multiple reservoirs averaging outputs, and whether or not to decompose the series into seasonal periods.
- Reservoir Computing has shown very good capabilities in various engineering problems including automatic fuel cell prognosis, pedestrian counting by a network of sensors, adaptive filtering, direct control of a tactile sensor, channel equalization problems, and to model the thermal behavior of a conductor monitored by a network of sensors placed on a high-voltage line in order to optimize its use.
- Reservoir Computing has been applied in various other fields such as astrophysics where an ESN network has been used to model the point spread function and apply it to the optical adaptive .Finally, Reservoir Computing has been applied in various other fields such as astrophysics where an ESN has been used to model the point spread function and apply it to adaptive optics. But also in the optimization of network architecture by predicting the popularity of a Web object, or in home automation in order to model user habits.

ACKNOWLEDGMENT

We would like to express our deepest gratitude toward our tutor Mr. Muftah Al-Mahdawi for his support, guidance and the benevolence that he showed us during several phases of this project.

REFERENCES

[1] Kohei, Nakajima, Physical reservoir computing—an introductory perspective. Published 14 May 2020. Available online on: https://iopscience.iop.org/article/10.35848/1347-4065/ab8d4f

[2] GouheiTanaka ,Toshiyuki Yamane ,Jean Benoit Héroux, RyoshoNakane, Naoki Kanazawa ,Seiji Takeda , Hidetoshi Numata ,Daiju Nakano , Akira Hirose, Recent advances in physical reservoir computing: A review. Published July 2019. Available online on:https://www.sciencedirect.com/science/article/pii/S0893608019300784

[3] Raphaël Couturier, Michel Salomon, Réservoir Computing : Étude théorique et pratique en reconnaissance de chiffres manuscrits.Published September 2015. Available

online on https://www.researchgate.net/publication/282705516_Reservoir_Computing_Etude_theorique_et_pratique_en_reconnaissance_de_chiffres_manuscrits

[4] Benjamin Schrauwen, David Verstraeten, Jan Van Campenhout , An overview of reservoir computing: theory, applications and implementations. Published April 2007.Available online on :https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2007-8.pdf

[5] Herbert Jaeger, Mantas Lukoševičius, Dan Popovici, Udo Siewert, Optimization and applications of echo state networks with leaky- integrator neurons . Published April 2007. Available online on:https://www.sciencedirect.com/science/article/abs/pii/S089360800700041

[6] Wolfgang Maass, Liquid State Machines: Motivation, Theory, and Applications, Published March 25, 2010.Available online on:https://igi-web.tugraz.at/PDF/189.pdf

[7] DawidPrzyczyna, Sébastien Pecqueur, Dominique Vuillaume, Konrad Szaciłowski, Reservoir computing for sensing: an experimental approach. Published 10 Jan 2020. Available online on:https://arxiv.org/abs/2001.04342

[8] ChrisanthaFernando ,SampsaSojakka , Pattern Recognition in a Bucket (2005). Published 2005. Available online on:http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.97.3902

[9] JillesVreeken, Spiking neural networks, an introduction. Published in 2003. Available online on: https://scholar.google.com/scholar?q=Spiking+neural+networks,an+introduction&hl=fr&as_sdt=0&as_vis=1&oi=scholart

[10] M. LukošEvičIus and H. Jaeger, Reservoir computing approaches to recurrent neural network training. Published in 2009. Available online on: https://www.sciencedirect.com/science/article/abs/pii/S1574013709000173

[11] R. Legenstein and W. Maass, What makes a dynamical system computationally powerful, New directions in statistical signal processing : From systems to brain. Published in 2007. Available online on:https://igi-web.tugraz.at/people/maass/psfiles/165.pdf

[12] H. Jaeger, Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach. GMD-ForschungszentrumInformationstechnik. Published in 2002. Available online on:https://igi-web.tugraz.at/people/maass/psfiles/165.pdf

[13] C. G. Langton, Computation at the edge of chaos : phase transitions and emergent computation. Published in 1990. Available online on:https://www.sciencedirect.com/science/article/abs/pii/016727899090064V

[14] A. M. Lyapunov, The general problem of the stability of motion. Published in 1992. Available online on: https://www.tandfonline.com/doi/abs/10.1080/00207179208934253

[15] D. Verstraeten, B. Schrauwen, M. d'Haene, and D. Stroobandt, An experimental unification of reservoir computing methods. Published in 2007. Available online on: https://pubmed.ncbi.nlm.nih.gov/17517492/

[16] D. Verstraeten and B. Schrauwen, On the quantification of dynamics in reservoir computing. Published in 2009. Available online on: https://link.springer.com/chapter/10.1007/978-3-642-04274-4_101

[17] F. Wyffels, B. Schrauwen, and D. Stroobandt, Stable output feedback in reservoir computing using ridge regression. Published in 2008.Available online on: https://www.researchgate.net/publication/221166114_Reservoir_regularization_stabilizes_learning_of_Echo_State_Networks_with_output_feedback

[18] M. Lukoševičius, A practical guide to applying echo state networks. Published in 2012. Available online on: https://link.springer.com/book/10.1007%2F978-3-642-35289-8

[19] MIT-BIH Database. Available online on: https://www.kaggle.com/shayanfazli/heartbeatmitbihtrain.csvA

[20] H. Lassoued, K.Raouf, ECG multi-class classification using neural network as machine learning model. Published in 2018. Available online on: https://www.researchgate.net/publication/325772788_ECG_multi-class_classification_using_neural_network_as_machine_learning_model