Reservoir computing: a photonic neural network for information processing

Yvan Paquot[1*], Joni Dambre[2], Benjamin Schrauwen[2], Marc Haelterman[1], Serge Massar[3]

[1]Service OPERA-Photonique, CP 194/5, Université Libre de Bruxelles (U.L.B.), avenue F.D. Roosevelt 50, 1050 Brussels, Belgium.

[2]Department of Electronics and Information Systems, Ghent University ; Sint-Pietersnieuwstraat 41; 9000 Ghent, Belgium.

[3]Laboratoire d'Information Quantique, CP 225, Université Libre de Bruxelles (U.L.B.), Boulevard du Triomphe, B-1050 Bruxelles, Belgium.

[*]Corresponding author: ypaquot@ulb.ac.be

Abstract

At the boundaries between photonics and dynamic systems theory, we combine recent advances in neural networks with opto-electronic nonlinearities to demonstrate a new way to perform optical information processing.

The concept of reservoir computing arose at the beginning of the XXIst century as a powerful solution to the issue of training recurrent neural networks. Indeed, it is comparable to, or even outperforms, other state of the art solutions for tasks such as speech recognition, handwriting recognition or time series prediction. As it is based on a static topology, it allows making the most of very simple physical architectures having complex nonlinear dynamics. The method is inherently robust to noise and does not require explicit programming operations thanks to its simple training procedure. It is therefore particularly well adapted for analog realizations. Among the various implementations of the concept that have been proposed, we focus on the promising field of optics.

Our experimental reservoir computer is based on opto-electronic technology, and can be viewed as an intermediate step towards an all optical device. Our fiber optics system is based on a nonlinear feedback loop operating at the threshold of chaos. In its present preliminary stage it is already capable of simple classification tasks as well as more complicated tasks like modeling nonlinear systems with memory. In the future we will increase the performance of the system by improving the interconnection topology.

Our aim is to demonstrate that such an analog reservoir can have performances comparable to state of the art digital implementations of Neural Networks. Furthermore, our system can in principle be operated at very high frequencies thanks to the high speed of photonic devices. Thus one could envisage targeting applications such as online information processing in broadband telecommunications.

1 Introduction

The neural networks have by now definitely a place in the field of complex information processing and are already used in the industry. However, they are usually implemented on classical binary processors in the form of codes simulating the networks. We study here a new way to implement a neural network architecture directly in an analog framework. Our system is based on an opto-electronic oscillator finely tuned at the edge of chaotic behaviour. Its application to standard benchmark tests shows that it is able to process non trivial information.
On the technological side, the use of optical components is a first step towards an all optical neural network, which could outperform the speed of electronics in the future. In particular, broadband telecommunications need high speed data processors to follow the bitrate of the optical fibers for basic signal processing tasks like routing or error detection. The study of analog computing is also of high interest on the theoretical point of view, as it can potentially overcome the discrete logic for various applications [15].
The principle underlying our neural network is Reservoir Computing which we present below. We then describe our experimental setup and present the results obtained so far.

2 Reservoir Computing

2.1 The concept of Reservoir

Reservoir Computing can be seen as a particular example of neural networks. It arose from the issue of training those networks. The idea was developed independently by Maas et al. [1] and by Jaeger [2] in 2002.
The basic principle uses a randomly fixed Recurrent Neural Network[1] (RNN) excited with an external stimulus (input signal). Since the dynamics of the system is modified by the input (the trajectory in the phase space changes), a measure of the instantaneous state of the system can provide information about this input. This measure ("read-out") is optimized (trained) with a sequence of known inputs to bring out a given property of the input. One can hope that the system then returns a relevant response for any similar input sequence [2] [4].
Such systems can be used to perform various *online input*[2] tasks with high generalisation capability and robustness to noise.

As a first step, we describe this concept in the common case of a discrete time Neural Network. In mathematical terms, a Reservoir is a dynamical system whose state can be defined at each discrete time t as a set of N scalar variables $x_{i,[t]}$ $(i = 1...N)$ called "neurons". The time behaviour is governed by an evolution equation. Given a state at a time t, the next timestep will be a non linear combination of the state at t.
The most general evolution equation governing the dynamics of a Reservoir of N neurons is the ex-

[1] Discrete nonlinear system of interconnected elements with possibility for the information of cycling back into the system.
[2] The system is out of equilibrium and processes a time varying signal.

pression [2]

$$x_{i,[t]} = \mathcal{F}_{NL}\left(\sum_{j=1}^{N} a_{ij} x_{j,[t-1]} + input_{i,[t]}\right) \tag{1}$$

where a_{ij} is the *connection matrix* defining the network and $\mathcal{F}_{NL}$ is a non linear function whose main properties are:

1. monotonously increasing,
2. saturating function reaching an asymptotic value for large arguments.

Typically, the hyperbolic tangent is chosen. The exact choice of this function is not critical, but influences slightly the performance of the RNN.
The last term, $input_{i,[t]}$, is the external stimulus exciting the Reservoir. It can be decomposed as a product of two contributions:

$$input_{i,[t]} = Mask_i \ . Inp_{[t]} \tag{2}$$

- $Inp_{[t]}$ $(t = 1 \dots T)$ is the sequence of values given by the task to solve. It is the real input signal of the system, changing arbitrarily from one time t to another,
- $Mask_i$ $(i = 1 \dots N)$ is an *Input Mask*. It is a random time-invariant set of N weights fixed during all the evolution of the Reservoir. This mask applies the input values $Inp_{[t]}$ to each neuron with a different weight, in order to break the symmetry of the system regarding the input values.

The successive states $x_{i,[t]}$ of the neurons indexed by i at the times t are defined recursively from an initial condition. Most of the time, the latter will be the trivial state.

$$x_{j,[0]} = 0 \quad \forall j \in \{1...N\} \tag{3}$$

In (1), the connection matrix a_{ij} contains all the information about the dynamics of the system. It characterises the topology of the links between the different neurons. Two important facts are that

- only a fraction of the elements of a_{ij} are non zero,
- the values of the non zero elements are assigned randomly.

The last statement ensures that the entropy of the system is maximum, so as to "enrich" its dynamics. In order to obtain an efficient Reservoir based data processor, the parameters (mainly: the norm of a_{ij} and the mean amplitude of the input signal) have to be tuned so as to be at the "Edge of chaos" [3].

2.2 Training a Reservoir Computer

The goal is to teach the dynamical system to solve a given task.
For this purpose, the concept of Reservoir Computing proposes a way to analyze the states of the Reservoir. At each time t, a linear functional assigns a scalar value depending on the values of the N neurons. Hence the read-out function is defined by a set of N scalar values (called "read-out weights"). Training the system consists in optimizing those N weights Ω_i. This optimization is done by a linear regression algorithm, based on a long enough sequence of successive Reservoir states.
The major innovations provided by the Reservoir Computing are

- the fact that the system (the Reservoir) itself is not trained,
- the use of a linear regression to optimize a linear read-out function.

Thus the response of the Reservoir Computer resulting from the application of the Read-out function comes down to:

$$R_{[t]} = \sum_{i=1}^{N} \Omega_i \ .x_{i,[t]} \tag{4}$$

2.3 What does a Reservoir look like?

Historically, the name Reservoir comes from the idea of processing data by observing the transient state of a bucket of water perturbed by a mechanical excitation [5].
Up to now, the Reservoirs were mainly studied by the computer simulation of their evolution equation. However, some attempts of physical implementations were done:

- a bucket water was used to perform speech recognition [5],
- an experiment made use of cat's brains [6]
- a VLSI implementation [7]

The possibility of developing a Reservoir based on a grid of Semiconductor Optical Amplifiers (SOA) has also been studied ([8], [9], [10] and [11]). However, this challenging idea requires further technological developments.
In this work, we implement a Reservoir in a different way. We identified a physical system which can be modelled by the equations of a Reservoir, eventually slightly modified.
The following sections describe how we developed an opto-electronic device behaving like a Reservoir. This work is to be considered as an intermediate step before a future all optical implementation.

2.4 Continuous Reservoir Computer

Up to now, we described an idealized RNN in discrete time (timesteps) and discrete topology (neurons). However, the physical processes enumerated in section 2.3 generally evolve in time continuously.
The continuous analog of equation (1) ($N \to \infty$ and with continuous temporal evolution) is [12]

$$x_{[t]}(\xi) = \mathcal{F}_{NL}\left(x_{[t]}^{input}(\xi) + \int_{t'}\int_{\xi'} x_{[t']}(\xi')K(\xi,\xi',t,t')d\xi' dt'\right) \tag{5}$$

Where the kernel $K(\xi,\xi',t,t')$ is a distribution representing the density of topological and temporal interaction between the "continuous neurons".
As well, the read-out function turns into

$$R_{[t]} = \int_{\xi'} \Omega_{\xi'} \ .x_{[t]}(\xi') \ d\xi' \tag{6}$$

In practice, with a view to simplify the problem, our implementation is half way between the continuous and discrete domains. The physical process is actually continuous (governed by equation (5)), however the input and the read-out operations are processed in discrete time and topology. Indeed, the input signal is applied by timesteps and the effective output states of the Reservoir are discretized in a postprocessing stage.

3 Experimental Reservoir Computer

The physical framework that we designed to emulate the behaviour of equation (1) is based on an enhanced version of an opto-electronic oscillator proposed by L. Larger et al. [16] [17]. It was first studied by L. Larger et al. as a chaotic device potentially applicable to chaos based cryptography.
After having reproduced some of the existing results on our device, we adapted it to be able to process information as a particular case of a Reservoir Computer.
The diagram of our experimental setup is shown in figure 1. A Superradiance LED injects continuous incoherent light of central wavelength 1550nm in an optical fiber. The continuous signal is modulated by an electro-optic Mach-Zehnder (M-Z), delayed by a loop of optical fiber (eventually a network of different delays) and converted into an electronic signal by a fast photodiode. An input signal is then added by using a radiofrequency coupler and the resulting signal drives the Mach-Zehnder modulator as a feedback effect. The whole system is controlled and synchronized by a computer.
An auxilliary device has been designed to perform the hardware read-out operation, equivalent to the equation (6), see inset in Fig. 1. However, this last part has not been tested yet and is presently done as a digital post-processing operation.

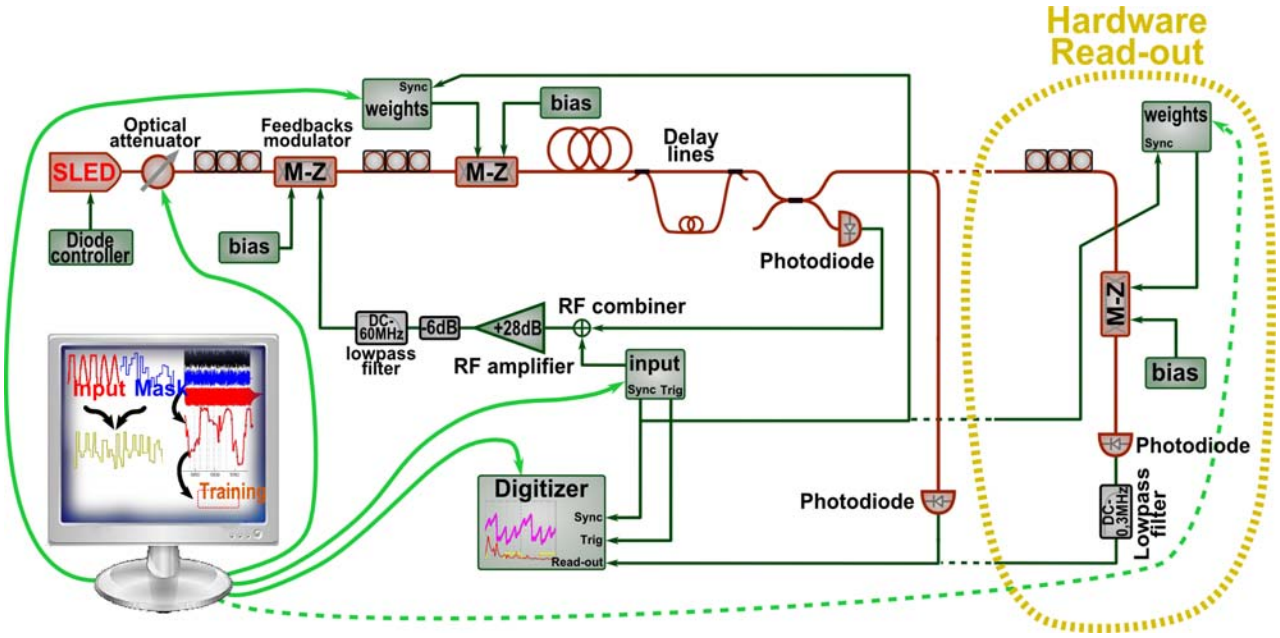

Figure 1: Experimental set-up. The red and green parts correspond respectively to the optical and electronic components.
"**SLED**" : Superradiance Light Emitting Diode source. "**M-Z**" : Mach-Zehnder modulators
"**Sync**" signal : common clock shared between the waveform generators "**input**" and "**weights**". "**Trig**" : trigger signal. "**RF combiner**" : electronic coupler adding the feedback and input signals. The resulting signal is then amplified ("**RF amplifier**").

The fundamental part of this scheme is the nonlinear delayed feedback loop appearing in the center of the diagram. It can be simplified as in figure 2. Each neuron is materialised by the light intensity

over a time window in the delay line. The length of the delay must equal the sum of the lengths of all the neurons. Hence, a neuron going through the Feedback modulator is modified by the value of the neuron reaching the end of the delay at the same time. As the Mach-Zehnder modulator's characteristics is of the form $sin^2(x)$, the loop induces a nonlinear feedback on the optical signal.
The total delay in our loop is $8,53\mu s$, so that for 100 neurons the neuron passage frequency is $11,96MHz$. The bandwidth of the system has been limited to 60MHz in order to avoid the propagation of higher frequency noise components.

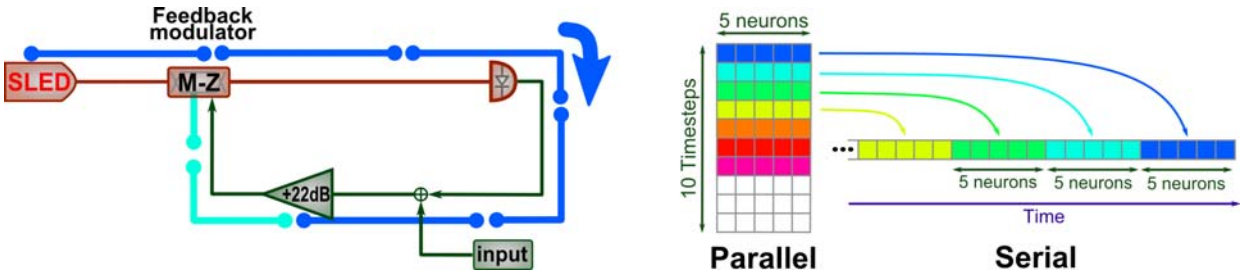

Figure 2: **Left**: Simplified diagram. The blue segments represent 5 neurons going through the loop. **Right**: Transition from the usual parallel vision of the network to the serial one.

4 Experimental results

4.1 Route to Chaos

The first property we studied was the response of our experimental device without signal. Indeed when the feedback and modulator gains reach a threshold value, the feedback loop amplifies any perturbation of the signal and the random fluctuations of the components can lead to the apparition of a non trivial signal in the system. The figure 3 shows a succession of stable modes followed by a chaotic behaviour, when increasing the feedback gain.
As well, the study of the effect of a periodic input on the system showed frequency doubling phenomenons (highlighted by the FFT of the signal in figure 3), followed by chaos.

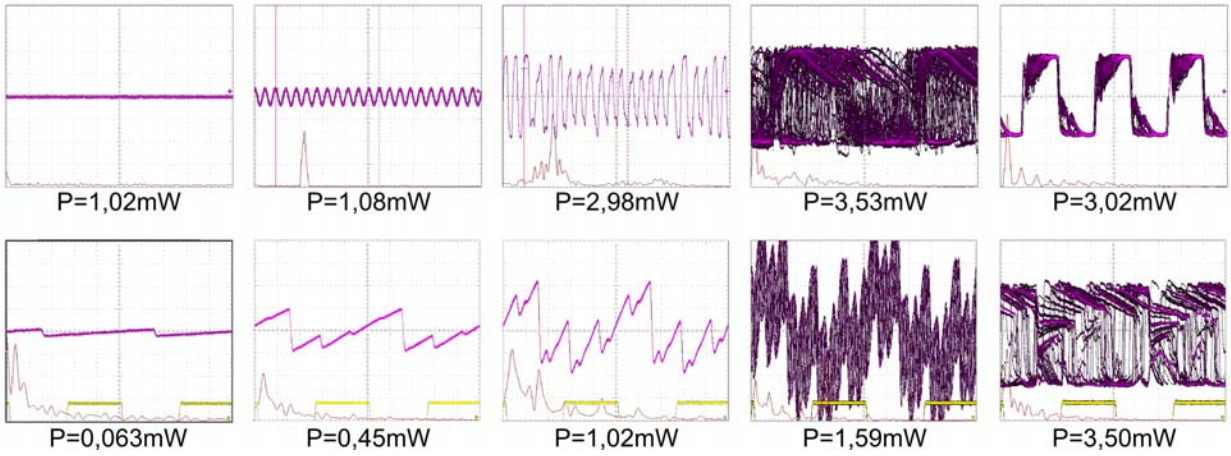

Figure 3: Route to Chaos as a function of the feedback gain (optical power).
Up: Without input signal. An hysteresis effect is observed when decreasing the gain.
Down: Excitation by a periodic sawtooth signal. Frequency doubling effects are observed.
Magenta: System output ; *Red:* FFT of the output

4.2 Benchmarks

We used three classical Reservoir Computing benchmark tasks to evaluate the performance of our system. A task provides the Reservoir Computer with an input signal and the expected answer for this input. The evaluation of the Reservoir performance is done using a metric that measures the error between the experimental and expected output.

NRMSE - Error Metric
The Normalized Root Mean Square Error is comprised between 0 and 1. A NRMSE score of 0 means that the Reservoir reproduces exactly the expected answer. A score of 1 means that no correlation can be found; the system is unable to process the task.
Let $s(t)$ be the output signal of the Reservoir and $y(t)$ the expected output over a sequence of n timesteps [13] :

$$NRMSE = \sqrt{\frac{\langle (s(t) - y(t))^2 \rangle_t}{\langle (y(t) - \langle y(t) \rangle_t)^2 \rangle_t}} = \sqrt{\frac{\frac{1}{n}\sum_{t=1}^{n}(s(t) - y(t))^2}{\frac{1}{n}\sum_{t=1}^{n}\left(y(t) - \frac{1}{n}\sum_{t=1}^{n} y(t)\right)^2}} \tag{7}$$

Signal classification task
Although very basic, this test provides an intuitive picture of the way a Reservoir processes information. The input signal is made of randomly arranged sine and square waves periods. The goal of the Reservoir Computer is to differentiate one from another by answering $'1'$ for a square wave and $'0'$ for a sine wave.

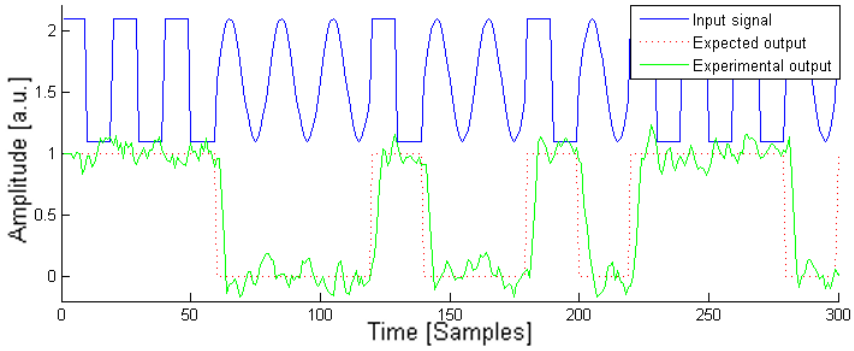

Figure 4: Example of the input and output for the signal classification task.

The smallest experimental NRMSE obtained reached $0,07$, which is excellent. However, this task is usually considered as "too simple" for characterizing neural networks.

Narma10 task

A much more complicated task asks the system to model a nonlinear nonpermanent system of order 10. The output y of the Nonlinear Auto-Regressive Moving Average (Narma) task [14] to a random input u (white noise) is

$$y(k+1) = 0,3y(k) + 0,05y(k).\left(\sum_{i=0}^{9} y(k-i)\right) + 1,5u(k-9)u(k) + 0,1 \tag{8}$$

The aim of the NARMA10 task is for the system to produce y(k) when it receives as input u(k). The NARMA10 task requires the reservoir to memorize at least the 10 past inputs, because of the fading memory property of the equation (8).

The best experimental NRMSE scored by a neural network of 100 neurons was $0,50$, which is at the limit of the usual threshold of an "interesting" system. The figure below shows a map of the NRMSE as a function of the two main parameters of our device: the input gain (= electronic gain) and the feedback gain (= −optical attenuation).

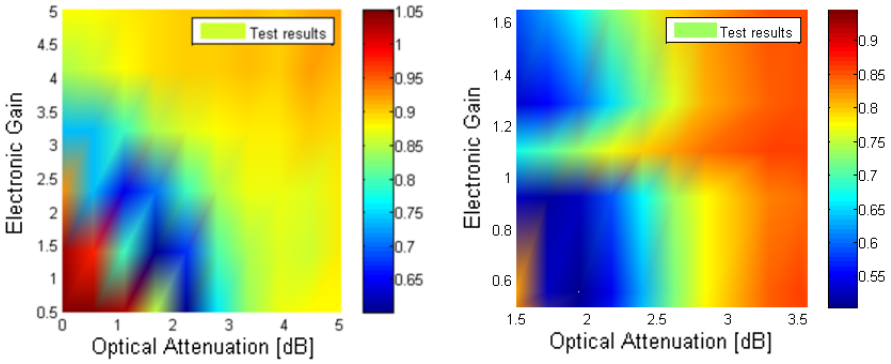

Figure 5: **Left:** Large range performance map for the Narma10 task; **Right:** Focus on the valley.

Memory capacity task

This test quantifies the memory of the system regarding the past events. Indeed the delayed feedback loop makes the instantaneous state depend on the previous inputs. We used the NRMSE metric to estimate the accuracy of the system to reconstruct the past input values. In Fig. 6 we show the evolution of the reconstruction error for an increasing delay of n timesteps regarding the input signal (in this case, a random sequence). Obviously, the neural network "forgets" the past events as the delay increases, due to its *fading memory* property.

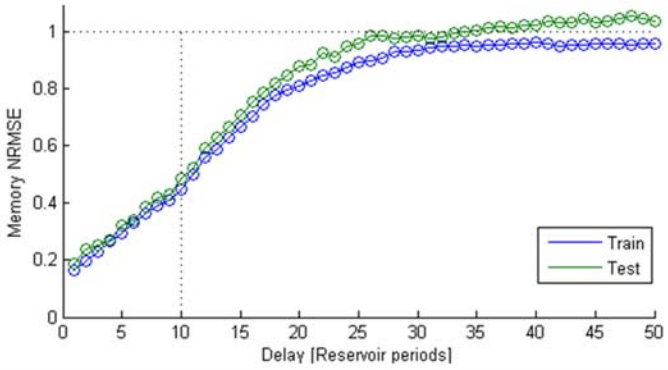

Figure 6: Memory capacity test for a neural network of 100 neurons in optimal conditions.

4.3 Optimization of the device

The working point of the experimental Reservoir Computer is a function of numerous parameters. Given the complexity of its behaviour, their optimization cannot be done analytically nor in simulation. We automated the system with a view to generate the performance maps according to a set of parameters. Usually, we studied the two main gain parameters (electronic gain of the input and optical attenuation of the feedback) plus a third parameter:

- **The period of the input:** varying this parameter comes to desynchronize the input sequence with the length of the delay loop. We found out that the best performance was obtained by desynchronizing the input by 1 neuron time, allowing consecutive neurons to interact.
- **The addition of a second delay line:** varying the attenuation in a second delay line of different length realises links between more neurons (it makes the interconnection matrix a_{ij} of eq. (1) richer). We showed that the performance increased slightly at a given attenuation value.
- **The working point of the feedback Mach-Zehnder modulator:** a bias voltage controls the working point on the nonlinear characteristics of the modulator. This study confirms the bias voltage $V = V_{\frac{\pi}{2}}$ (the transparency of the modulator is $\frac{1}{2}$ when the feedback signal equals zero) to be the optimal working point.

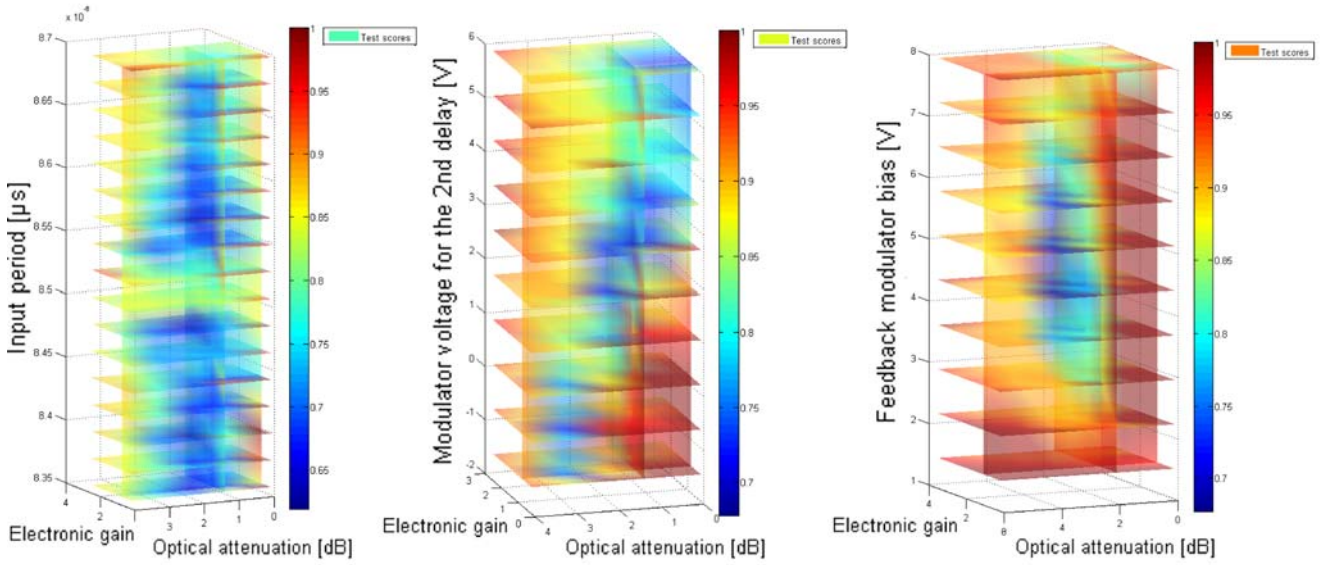

Figure 7: Impact of the period of the input (**left**), second delay line (**center**) and Mach-Zehnder bias (**right**). The three diagrams show the performance of the reservoir computer (the blue areas correspond to the highest performances) as a function of the electronic and optical gains (horizontal slices) and of a third parameter (vertical axis).

5 Conclusion

The study of our prototype shows that this choice of architecture is relevant for implementing a neural network. However, the performance of the system is still limited, most probably due to the noise level of the components and the discretization of the response from the continuous neural network. As well, our device is governed by a simplification of the equations (1) and (5) due to the limited interconnection between the neurons. Further work will focus on better understanding the continuous dynamics of the system and improving the experimental conditions in order to solve more complicated tasks.

Acknowledgments

We acknowledge the support of the *Fonds pour la formation à la Recherche dans l'Industrie et dans l'Agriculture* (FRIA, Belgium), of the *Interuniversity Attraction Poles Photonics@be Programme* (Belgian Science Policy) under grant IAP6-10.

References

[1] W. Maass, T. Natschläger, and H. Markram., *Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations*, Neural Computation, 14(11):2531-2560, 2002.

[2] H. Jaeger, *The "echo state" approach to analysing and training recurrent neural networks*, Fraunhofer Institute for Autonomous Intelligent Systems, Technical report: GMD Report 148, 2001.

[3] R. Legenstein and W. Maas, *What makes a dynamical system computationally powerful?*, in *New Directions in Statistical Signal Processing: From Systems to Brain*, MIT Press, Cambridge, 2005.

[4] H. Jaeger, H. Haas *Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication*, Science Vol.304, pp. 78 - 80, 2004.

[5] C. Fernando, S. Sojakka, *Pattern recognition in a bucket*, Springer, Lecture notes in computer science, Berlin, pp.588-597, 2003.

[6] D. Nikolić, S. Häusler, W. Singer, W. Maass. *Temporal dynamics of information content carried by neurons in the primary visual cortex.*, Proc. of NIPS, 2007.

[7] F. Schürmann, K. Meier, J. Schemmel, *Edge of chaos computation in mixed-mode vlsi - "a hard liquid"*, Proc. of NIPS, 2004.

[8] K. Vandoorne et al. *Toward optical signal processing using Photonic Reservoir Computing*, Optics Express, Vol. 16, 2008.

[9] J. Callant, *Conception et modélisation d'un réseau de neurones optique de type réservoir*, Master thesis, Université Libre de Bruxelles, 2008.

[10] A. Rahim, *Coupled Semiconductor Optical Amplifier Network for Reservoir Computing*, Master thesis, Gent University, 2008.

[11] W. Dierckx, *Analyse van een nanophotonische implementatie van reservoir computing*, Master thesis, Gent University, 2008.

[12] Y.Paquot, *Opto-electronic implementation of an Artificial Intelligence system based on the concept of Reservoir Computing*, Master thesis, Université Libre de Bruxelles, 2009.

[13] M. Lukosevicius, H. Jaeger, *Overview of Reservoir Recipes*, Jacobs University, Technical Report No. 11, 2007.

[14] D. Verstraeten et al., *An experimental unification of reservoir computing methods*, Neural Networks 20, 391-403, Elsevier, 2007.

[15] H.T. Siegelmann, *Neural networks and analog computation: beyond the Turing limit*, Birkäuser Boston, 1999.

[16] L. Larger et al., *From Flow to Map in an Experimental High-Dimensional Electro-Optic Nonlinear Delay Oscillator* , Physical Review Letters 95 - 043903, July 2005.

[17] Y. Chembo Kouomou et al., *Chaotic Breathers in Delayed Electro-Optical Systems* , Physical Review Letters 95 - 203903, 2005.

[18] R.C. Hilborn, *Chaos and Nonlinear Dynamics - An Introduction for Scientists and Engineers*, 2^{nd} edition, Oxford University Press, New York, 2006.