Towards Adjustable Signal Generation with Photonic Reservoir Computers

Piotr Antonik[1(✉)], Michiel Hermans[1], Marc Haelterman[2], and Serge Massar[1]

[1] Laboratoire d'Information Quantique, Université Libre de Bruxelles, 50 Avenue F. D. Roosevelt, CP 225, 1050 Brussels, Belgium
pantonik@ulb.ac.be

[2] Service OPERA-Photonique, Université Libre de Bruxelles, 50 Avenue F. D. Roosevelt, CP 194/5, 1050 Brussels, Belgium

Abstract. Reservoir Computing is a bio-inspired computing paradigm for processing time dependent signals. We have recently reported the first opto-electronic reservoir computer trained online by an FPGA chip. This setup makes it in principle possible to feed the output signal back into the reservoir, which in turn allows to tackle complex prediction tasks in hardware. In present work, we investigate numerically the performance of an offline-trained opto-electronic reservoir computer with output feedback on four signal generation tasks. We report very good results and show the potential of such setup to be used as a high-speed analog control system.

Keywords: Reservoir Computing · FPGA · Pattern generation · Numerical results · Opto-electronic systems

1 Introduction

Reservoir Computing (RC) is a set of methods for designing and training artificial recurrent neural networks [11,14]. A typical reservoir is a randomly connected fixed network, with random coupling coefficients between the input signal and the nodes. This reduces the training process to solving a system of linear equations [7,13]. The RC algorithm has been successfully applied to channel equalisation [6,16,20], phoneme recognition [18] and won an international competition on prediction of future evolution of financial time series [1].

Reservoir Computing is very well suited for analog implementations: various electronic [4,8], opto-electronic [12,15,16] and all-optical [5,6,19,20] implementations have been reported since 2012. We have recently reported the first online-trained opto-electronic reservoir computer [2]. The key feature of this implementation is the FPGA chip, programmed to generate the input sequence, train the reservoir computer using the simple gradient descent algorithm, and compute the reservoir output signal in real time.

This setup offers the possibility to tackle prediction tasks in hardware by feeding the output signal back into the reservoir. We have shown numerically

DOI: 10.1007/978-3-319-44778-0_44

that such a system could perform well on pattern generation and Mackey-Glass chaotic time series prediction tasks [3]. In this work we improve the experimental setup and focus on the pattern generation task [9], with several additional tasks that have been investigated in the RC community. We adapt the use of the FPGA chip to train the neural network offline for higher precision and more control of the process. The performance of the setup is tested in simulations on four signal generation tasks: simple pattern generation [3], frequency generation, multi-pattern generation [17] and tunable frequency generation [21]. These tasks have various applications in motion generation, robot control and data storage [17]. Solving them in hardware can allow opto-electronic reservoir computers to be applied in fast control applications, for example high-speed robot control [9]. The promising results we report here thus pave the way towards experimental investigations we are planning to carry out in the upcoming months.

2 Reservoir Computing

A general reservoir computer is described in [13]. In our implementation, depicted in Fig. 1, we use a sine function $f = \sin(x)$ and a ring topology to simplify the interconnection matrix, so that only the first neighbour nodes are connected [12,16]. The evolution equations are given by

$$x_0(n+1) = \sin\left(\alpha x_N(n-1) + \beta M_0 u(n)\right), \tag{1a}$$
$$x_i(n+1) = \sin\left(\alpha x_{i-1}(n) + \beta M_i u(n)\right), \tag{1b}$$

where $x_i(n),\ i = 0, \ldots, N-1$ are the internal variables, evolving in discrete time $n \in \mathbb{Z}$, α and β parameters are used to adjust the feedback and the input signals, respectively, $u(n)$ is a time multiplexed input signal, and M_i is the input mask, drawn from a uniform distribution over the interval $[-1, +1]$ [6,16]. The reservoir computer produces an output signal

$$y(n) = \sum_{i=0}^{N} w_i x_i(n), \tag{2}$$

where $x_N = 1$ is a constant neuron used to adjust the bias of the output signal and w_i are the readout weights, trained offline [4–6,12,15,16,19] in order to minimise the Mean Square Error (MSE) between the output signal $y(n)$ and the target signal $d(n)$.

During the training phase, the reservoir computer receives a periodic training sequence as input $u(n)$ and is trained to predict the next value of the sequence from the current one. During the test phase, the reservoir input $u(n)$ is switched from the training sequence to the reservoir output signal $y(n)$, and the system is left running autonomously. In that case, the dynamics of the systems is described by the following equations

$$x_0(n+1) = \sin\left(\alpha x_N(n-1) + \beta M_0 y(n)\right), \tag{3a}$$
$$x_i(n+1) = \sin\left(\alpha x_{i-1}(n) + \beta M_i y(n)\right). \tag{3b}$$

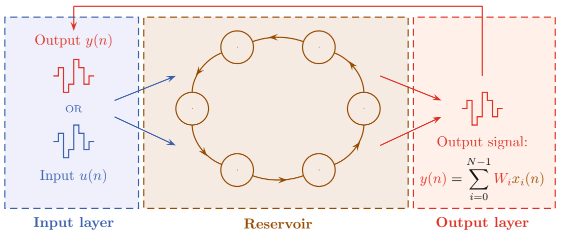

Fig. 1. Schematic representation of our reservoir computer with output feedback. The recurrent neural network with N nodes denoted $x_i(n)$ in ring-like topology (in brown) is driven by either a time multiplexed input signal $u(n)$, or its own output signal $y(n)$, given by a linear combination of the readout weights w_i with the reservoir states $x_i(n)$. (Color figure online)

3 Signal Generation Tasks

Pattern Generation. A pattern is a short sequence of randomly chosen real numbers (here within the interval $[-0.5, 0.5]$) that is repeated periodically to form an infinite time series [3]. The aim is to obtain a stable pattern generator, that reproduces precisely the pattern and doesn't deviate to another periodic behaviour. To evaluate the performance of the generator, we compute the MSE between the reservoir output signal and the target pattern signal during the training phase and the autonomous run.

Frequency Generation. The system is trained to generate a sine wave given by

$$u(n) = \sin(\nu n), \tag{4}$$

where ν is a relative frequency and n is the discrete time. The physical frequency f of the sine wave depends on the experimental roundtrip time T (see Sect. 4) as follows

$$f = \frac{\nu}{2\pi T}. \tag{5}$$

This task allows to measure the bandwidth of the system and investigate different timescales within the neural network.

Multi-pattern Generation. This tasks adds another dimension to the simple pattern generation. The network is trained to generate several different patterns and a second input signal $u_2(n)$ is introduced to select the pattern to generate. Equations (3) thus become

$$x_0(n+1) = \sin\left(\alpha x_N(n-1) + \beta M_0 y(n) + \beta_2 M_0' u_2(n)\right), \tag{6a}$$

$$x_i(n+1) = \sin\left(\alpha x_{i-1}(n) + \beta M_i y(n) + \beta_2 M_i' u_2(n)\right), \tag{6b}$$

where β_2 is a second input gain and M_i' is a second input mask. Both input masks are generated randomly, and both input gains are optimised independently.

During the autonomous run, the second input signal $u_2(n)$ is regularly changed in order to test the performance of the system on all patterns.

Tunable Frequency Generation. Here the frequency generator is upgraded with a second input signal to tune its frequency. The network is trained to generate several sine waves with different frequencies, given by

$$u(n) = \sin\left(\bar{\nu}(n)n\right), \tag{7}$$

where $\bar{\nu}(n)$ is a time-dependent user-defined frequency, that is fed into the system through the second input $u_2(n) = \bar{\nu}(n)$. The physical output frequency f can be computed using Eq. (5). Testing of the performance is similar to the multi-pattern generation task.

4 Numerical Simulations

Figure 2 depicts the experimental setup [2], which is the basis for numerical simulations presented here. The opto-electronic reservoir, a replica of previously reported works [2,16], is driven by a Xilinx ML605 evaluation board, powered by a Virtex 6 FPGA chip and paired with a 4DSP FMC-151 daughter card, used for signal acquisition and generation. The FPGA is programmed to record the reservoir states $x_i(n)$ and send them to the personal computer, running Matlab, through an Ethernet connection. The readout weights w_i are uploaded on the chip for real-time computation of the reservoir output signal $y(n)$ during the autonomous run.

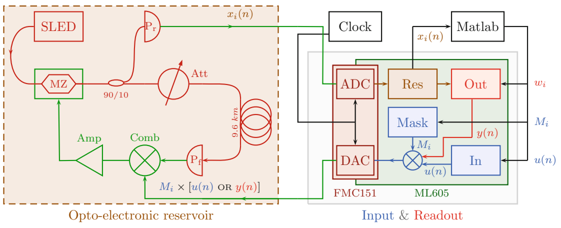

Fig. 2. (a) Schematic representation of the simulated setup, based on the experimental system [2,16]. Optical and electronic components of the opto-electronic reservoir are shown in red and green, respectively. It contains an incoherent light source (SLED), a Mach-Zehnder intensity modulator (MZ), a 90/10 beam splitter, an optical attenuator (Att), a 9.6 km fibre spool, two photodiodes (P$_\text{r}$ and P$_\text{f}$), a resistive combiner (Comb) and an amplifier (Amp). The FPGA board acquires the reservoir states $x_i(n)$ and generates analog input and output signals to the reservoir. A personal computer, running Matlab, computes the readout weights w_i. (Color figure online)

The experiment roundtrip time is defined by the length of the delay loop. We are planning to use 9.6 km of fiber in order to obtain a delay of $T = 32\,\mu\text{s}$.

This would allow sampling the entire loop 8000 times at 250 MS/s (maximum sampling frequency of the FMC-151 Analog-to-Digital Converter) and thus fit up to 1000 neurons into the reservoir, with at least 8 samples per neuron.

All numerical experiments were performed in Matlab, on a standard personal computer. The simulations account for major aspects of the experimental setup and allow to scan the most influential parameters, such as input gains β and β_2, feedback gain α and reservoir size N.

5 Results

Pattern Generation. As we have shown previously [3], this task works well with online learning even on small reservoirs: a 51-neuron network is capable of generating patterns up to 51-element long, where 51 is expected to be a fundamental limit because it is the upper bound on the linear memory of the network [10]. We obtained the same results with offline training here, and found optimal gain parameters. The system works best with a very low input gain $\beta = 0.001$ and high feedback gain $\alpha = 0.9$. The system was trained over $5k$ inputs and then left running autonomously for $50k$ timesteps. We obtained training errors ranging from 10^{-25} (for short patterns with $L = 10$) to 10^{-12} (for long patterns, $L = 51$), and autonomous errors ranging from 10^{-22} to 10^{-8}, respectively.

Frequency Generation. Frequency generation requires a different method for computing the error during the autonomous run, that would focus on the frequency of the generated signal. For this reason we used the Fast Fourier Transform (FFT) algorithm to compute the frequency of the reservoir output signal and compare it to the frequency of the target signal.

We used a slightly larger reservoir with $N = 100$ and trained it over $1k$ input samples. We tried increasing the reservoir size up to $N = 1000$ and the training length up to $10k$ samples without noticeable improvements. The output frequency was measured after an autonomous run of $20k$ timesteps.

With optimal gain parameters $\alpha = 0.9$ and $\beta = 0.1$, we were able to generate relative frequencies within $\nu \in [0.06, 3.14]$ with MSE of order of 10^{-7} and Full Width at Half Maximum (FWHM) of the FFT of about 10^{-3}. The upper limit is given by half of the sampling rate of the system and corresponds to the Nyquist frequency. As for the lower limit, we couldn't obtain stable output signal with frequency lower than 0.06 for most random input mask. The roundtrip time $T = 32\,\mu s$ of the experimental setup gives a sampling frequency of 31.2 kHz. Using Eq. (5), this sets the bandwidth of the generator to 300 Hz–15.6 kHz.

Multi-pattern Generation. This task is significantly more complex than the simple pattern generation, as the network needs to learn to switch between several different patterns. Good performance thus requires a large reservoir and a carefully chosen training sequence which contains all possible transitions between the patterns. We also noted that results depend on the shape of the input mask. Figure 3 shows an example of simulation with 3 different patterns.

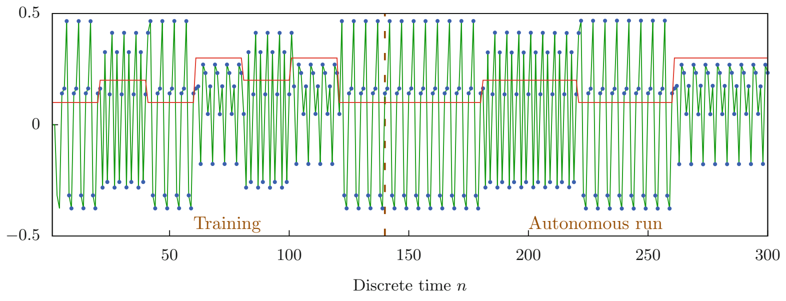

Fig. 3. Example of simulation result for the multi-pattern generation task. The reservoir output signal (blue dots) is almost identical to the target signal (green curve). The second input signal $u_2(n)$, shown in red, switches between 3 values, as the system is trained to generate three short patterns. The training sequence contains all transitions between different patterns. The autonomous run continues beyond the scope of the figure. (Color figure online)

For this task the reservoir size was increased to $N = 800$ neurons. We were able to generate up to 4 different patterns of length 10, with training error of 2×10^{-7}, and 4×10^{-4} for the autonomous run. The system was trained over 850 inputs and then ran autonomously for 4250 timesteps. All transitions occured synchronously, that is, from the last element of a pattern to the first element of another. We also tried generating shorter patterns and could store 8 patterns of length 5, with training and autonomous run errors of 2×10^{-6} and 4×10^{-4}, respectively. This required much longer simulations, with $5.6k$ inputs for the training and $28k$ timesteps for the autonomous run.

Tunable Frequency Generation. Similar to multi-pattern generation, this task requires a large reservoir capable of containing many smaller clusters oscillating at different frequencies (see [21] for a more in-depth overview). The reservoir computer was trained to generate different sine waves given by Eq. (7), FFT algorithm was used to evaluate the performance.

We used a large reservoir with $N = 1000$ neurons and the following parameters: $\alpha = 0.7$, $\beta = 0.03$ and $\beta_2 = 0.9$. The network was trained over $6.6k$ samples and was taught to generate 40 frequencies equally spaced between 0.1 and 1.1. Each frequency was learned for 10 periods, to ensure smooth transitions. For the autonomous run, we investigated different scenarios. At first, we decreased the frequency back to 0.1 and then increased it to 1.1 again. This was done by large steps of 0.05 every $5k$ timesteps to test the stability of the generator. The system produces very good results, with frequency MSE of 1.5×10^{-6}. As another test case, the second input signal $u_2(n)$ was first decreased to 0.6, and then followed a random walk. The reservoir computer generated the desired frequencies very well, with frequency MSE of 1.4×10^{-6}. The FWHM of the FFT for these two cases is about 0.005. Faster control modulation, with $u_2(n)$ changing every 200 timesteps, results in higher frequency MSE (1.2×10^{-3}, with FFT

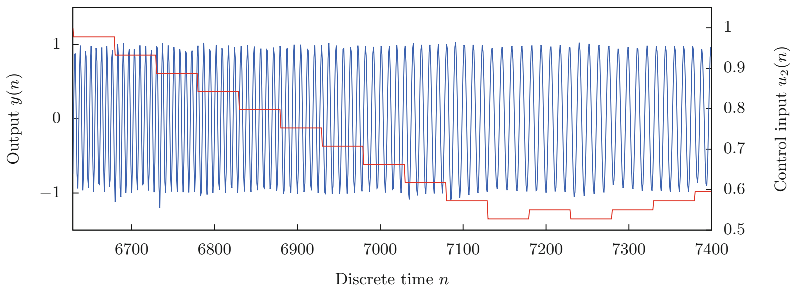

Fig. 4. Example of autonomous run for the tunable frequency generation task. The control signal, shown in red, is decreased down to 0.6 every 50 timesteps, and then follows a random walk, that continues beyond the scope of the figure. Although u_2 switches asynchronously, the RC shifts smoothly from one frequency to another. (Color figure online)

FWHM of order of 0.1), but still the RC follows the desired frequency reasonably well. Figure 4 shows an example of simulation with fast modulation (every 50 timesteps).

6 Conclusion

We investigated numerically how an opto-electronic reservoir computer with output feedback performs on various signal generation tasks. We evaluated optimal gain parameters, reservoir sizes and elaborated specific training sequencies for each tasks. We obtained very good results, showing that the upcoming experimental setup could in principle be employed as a fast analog control system. Coupled with the recently implemented online learning [2] this system could possibly be used as an analog "brain" for high-speed self-learning robots.

Acknowledgements. We acknowledge financial support by Interuniversity Attraction Poles program of the Belgian Science Policy Office under grant IAP P7-35 "photonics@be", by the Fonds de la Recherche Scientifique FRS-FNRS and by the Action de la Recherche Concertée of the Académie Universitaire Wallonie-Bruxelles under grant AUWB-2012-12/17-ULB9.

References

1. The 2006, 07 forecasting competition for neural networks & computational intelligence (2006). http://www.neural-forecasting-competition.com/NN3/. Accessed 21 Feb 2014
2. Antonik, P., Duport, F., Smerieri, A., Hermans, M., Haelterman, M., Massar, S.: Online training of an opto-electronic reservoir computer. In: Arik, S., Huang, T., Lai, W.K., Liu, Q. (eds.) ICONIP 2015. LNCS, vol. 9490, pp. 233–240. Springer, Heidelberg (2015). doi:10.1007/978-3-319-26535-3_27

3. Antonik, P., Hermans, M., Duport, F., Haelterman, M., Massar, S.: Towards pattern generation and chaotic series prediction with photonic reservoir computers. In: SPIE's 2016 Laser Technology and Industrial Laser Conference, vol. 9732 (2016)
4. Appeltant, L., Soriano, M.C., Van der Sande, G., Danckaert, J., Massar, S., Dambre, J., Schrauwen, B., Mirasso, C.R., Fischer, I.: Information processing using a single dynamical node as complex system. Nat. Commun. **2**, 468 (2011)
5. Brunner, D., Soriano, M.C., Mirasso, C.R., Fischer, I.: Parallel photonic information processing at gigabyte per second data rates using transient states. Nat. Commun. **4**, 1364 (2012)
6. Duport, F., Schneider, B., Smerieri, A., Haelterman, M., Massar, S.: All-optical reservoir computing. Opt. Express **20**, 22783–22795 (2012)
7. Hammer, B., Schrauwen, B., Steil, J.J.: Recent advances in efficient learning of recurrent networks. In: Proceedings of the European Symposium on Artificial Neural Networks, pp. 213–216, Bruges, Belgium, April 2009
8. Haynes, N.D., Soriano, M.C., Rosin, D.P., Fischer, I., Gauthier, D.J.: Reservoir computing with a single time-delay autonomous Boolean node. Phys. Rev. E **91**(2), 020801 (2015)
9. Ijspeert, A.J.: Central pattern generators for locomotion control in animals and robots: a review. Neural Netw. **21**(4), 642–653 (2008)
10. Jaeger, H.: Short term memory in echo state networks. Technical GMD report 152 (2001)
11. Jaeger, H., Haas, H.: Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science **304**, 78–80 (2004)
12. Larger, L., Soriano, M., Brunner, D., Appeltant, L., Gutiérrez, J.M., Pesquera, L., Mirasso, C.R., Fischer, I.: Photonic information processing beyond turing: an optoelectronic implementation of reservoir computing. Opt. Express **20**, 3241–3249 (2012)
13. Lukoševičius, M., Jaeger, H.: Reservoir computing approaches to recurrent neural network training. Comp. Sci. Rev. **3**, 127–149 (2009)
14. Maass, W., Natschläger, T., Markram, H.: Real-time computing without stable states: a new framework for neural computation based on perturbations. Neural Comput. **14**, 2531–2560 (2002)
15. Martinenghi, R., Rybalko, S., Jacquot, M., Chembo, Y.K., Larger, L.: Photonic nonlinear transient computing with multiple-delay wavelength dynamics. Phys. Rev. Let. **108**, 244101 (2012)
16. Paquot, Y., Duport, F., Smerieri, A., Dambre, J., Schrauwen, B., Haelterman, M., Massar, S.: Optoelectronic reservoir computing. Sci. Rep. **2**, 287 (2012)
17. Sussillo, D., Abbott, L.: Generating coherent patterns of activity from chaotic neural networks. Neuron **63**(4), 544–557 (2009)
18. Triefenbach, F., Jalalvand, A., Schrauwen, B., Martens, J.P.: Phoneme recognition with large hierarchical reservoirs. Adv. Neural Inf. Process. Syst. **23**, 2307–2315 (2010)
19. Vandoorne, K., Mechet, P., Van Vaerenbergh, T., Fiers, M., Morthier, G., Verstraeten, D., Schrauwen, B., Dambre, J., Bienstman, P.: Experimental demonstration of reservoir computing on a silicon photonics chip. Nat. Commun. **5**, 3541 (2014)
20. Vinckier, Q., Duport, F., Smerieri, A., Vandoorne, K., Bienstman, P., Haelterman, M., Massar, S.: High-performance photonic reservoir computer based on a coherently driven passive cavity. Optica **2**(5), 438–446 (2015)
21. Wyffels, F., Li, J., Waegeman, T., Schrauwen, B., Jaeger, H.: Frequency modulation of large oscillatory neural networks. Biol. Cybern. **108**(2), 145–157 (2014)