Embodiment of Learning in Electro-Optical Signal Processors

Michiel Hermans,[1,*] Piotr Antonik,[1,†] Marc Haelterman,[2] and Serge Massar[1]
[1]*Laboratoire d'Information Quantique, Université libre de Bruxelles, 50 Avenue F. D. Roosevelt, CP 224, B-1050 Brussels, Belgium*
[2]*Service OPERA-Photonique, Université libre de Bruxelles, 50 Avenue F. D. Roosevelt, CP 194/5, B-1050 Brussels, Belgium*

(Received 4 March 2016; revised manuscript received 21 May 2016; published 16 September 2016)

Delay-coupled electro-optical systems have received much attention for their dynamical properties and their potential use in signal processing. In particular, it has recently been demonstrated, using the artificial intelligence algorithm known as reservoir computing, that photonic implementations of such systems solve complex tasks such as speech recognition. Here, we show how the backpropagation algorithm can be physically implemented on the same electro-optical delay-coupled architecture used for computation with only minor changes to the original design. We find that, compared to when the backpropagation algorithm is not used, the error rate of the resulting computing device, evaluated on three benchmark tasks, decreases considerably. This demonstrates that electro-optical analog computers can embody a large part of their own training process, allowing them to be applied to new, more difficult tasks.

DOI: 10.1103/PhysRevLett.117.128301

Introduction.—Nonlinear dynamical systems, such as neural networks, can be used to perform highly complex computations, e.g., speech or image recognition. One of the main difficulties when using such systems is to train their internal parameters. The backpropagation (BP) algorithm [1,2] is one of the most important algorithms in this area, and is behind the remarkable successes achieved in the field of deep learning in the last decade [3]. The simple idea behind the BP algorithm is to compute the derivative (or gradient) of a cost function in the parameter space of the system. The gradient is then subtracted from the parameters themselves in order to reduce the cost function. This process is repeated until the cost function no longer reduces.

Such nonlinear dynamical systems can be implemented in hardware. Here, also, the training of internal parameters is key and the use of the BP algorithm is highly beneficial in order to improve performance [4,5]. However, implementing the BP algorithm in hardware systems can be difficult because of the need for an accurate model to compute the gradient and because of the resources necessary to run the BP algorithm. Remarkably, in certain cases, the BP algorithm can be implemented physically on the system it is optimizing [6]. The basic idea behind this advance is to use a slightly modified version of the system for propagating error signals backwards, i.e., for running the BP algorithm. Such self-learning computing systems could be highly advantageous, as any gain in terms of processing speed or limited power consumption will also apply to the training phase. Furthermore, having the same hardware computing the BP algorithm eliminates, to a large extent, the need for an accurate model of the system. This idea may conceivably also have implications for biological neural networks, as these are physical system that—using mechanisms that are not yet well understood—can both compute and carry out their own training process. Reference [6] also reported a proof of concept experiment in which physical BP was tested on a simple task, but left open the question of whether the algorithm, with all the imperfections inherent in an experiment, can provide the same improvement in performance as numerical approaches [4,5].

References [4–6] used, as a computational device, a delay dynamical system (see [7,8]). Such systems can be exploited to realize a form of analog computer based on the reservoir computing (RC) paradigm [9,10] in which unoptimized high-dimensional dynamic systems (termed reservoirs) are used as signal processors. The RC approach is simple, versatile, and can be applied to a wide set of problems (see the review [11]) and experimental implementations [12–20]. Applying the BP algorithm to delay-coupled signal processors allows one to optimize many more parameters than in traditional RC, yielding significant improvements in performance as was shown in simulation in [4], and subsequently, in an experiment [5] in which BP was applied to a numerical model of the system, and the results of the BP algorithm applied to the physical experimental setup.

Here, we implement the BP algorithm physically on an electro-optic delay dynamical system used as a signal processor. Our key innovation is to modify the system used in [16,17] by adding a photonic setup capable of implementing both the nonlinearity and its derivative, so that it can be used both as a signal processor and to perform the BP algorithm. We test our system on several tasks considered hard in the machine learning community,

including a real world phoneme recognition task (the TIMIT task, discussed later in this Letter), obtaining state of the art results when the BP algorithm is used. Thus, the present work demonstrates the full potential of physical BP. It constitutes an important step towards self-learning hardware, with potential applications towards ultrafast, low energy consumption, computing systems.

In the following, we first recall the principles of reservoir computing and error back propagation, before introducing our experimental implementation. We then report the results obtained on several benchmark tasks, and conclude with a discussion of the results and their implications.

Reservoir computing.—In typical RC tasks, the goal is to map an input sequence s_i (where $i \in \{1, \ldots, L\}$, with L the total sequence length) to an output sequence y_i, which has target values y_i^*, for example a speech signal to a sequence of labels. In order to use delay-coupled systems as reservoir computers, the discrete time input sequence s_i is encoded into a continuous time function $z(t)$ by the input mask $m(r)$ and bias mask $m_b(r)$, where $r \in [0, T]$, with T the masking period, as follows:

$$z(t) = z(iT + r) = m(r)s_i + m_b(r). \tag{1}$$

In our implementation, we use a delay-coupled system with sine nonlinearity (which stems from the transfer function of the intensity modulator, as will be explained below), which obeys the equation

$$a(t + D) = \mu \sin[a(t) + z(t)], \tag{2}$$

where $a(t)$ is the state variable and D is the delay. The factor μ corresponds to the total loop amplification. Equation (2) can be seen as a special case of the Ikeda delay differential equation [21].

One then needs to map the continuous time state variable $a(t)$ to a discrete time output sequence y_i. This is performed using an output mask $u(r)$ where $r \in [0, T]$ and a bias term u_b as follows:

$$y_i = \int_0^T dr a(iT + r)u(r) + u_b. \tag{3}$$

In the RC paradigm, the input mask is typically chosen randomly, and the output mask $u(r)$ and u_b is determined by solving a linear system of equations which minimizes the mean square error C between the desired and actual output: $C = \langle (y_i - y_i^*)^2 \rangle_i$.

Error backpropagation.—The goal of applying error backpropagation to the above scheme is to optimize both the input and output masks $m(r)$, $m_b(r)$, $u(r)$, and u_b, knowing the output $a(t)$, and the desired output y_i^*. To this end, one needs the gradient of C with respect to the masks, given by (the proof is given in the Supplemental Material [22])

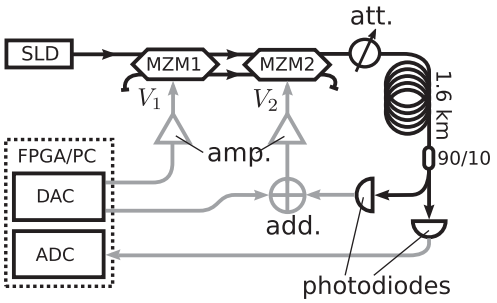

FIG. 1. Schematic representation of the experimental system. SLD: superluminescent diode; MZM1 and MZM2: dual input-dual output Mach-Zehnder Modulators; V_1 and V_2: driving voltages of the MZMs; att.: programmable optical attenuator; add.: electrical combiner; amp.: amplifier.

$$\bar{e}(iT + r) = e_i u(r), \tag{4}$$

$$e(t - D) = J(t)[e(t) + \bar{e}(t)], \tag{5}$$

$$J(t) = \mu \cos[a(t) + z(t)], \tag{6}$$

$$\frac{dC}{dm(r)} = \sum_i e(iT + r)s_i, \tag{7}$$

$$\frac{dC}{dm_b(r)} = \sum_i e(iT + r), \tag{8}$$

where $\bar{e}(t) = \partial C/\partial a(t)$ is a continuous time signal and, as above, $i \in \{1, \ldots, L\}$ and $r \in [0, T]$. One can then iteratively improve the masks so as to lower C.

Physical BP.—In order to use the same hardware for both the signal processing and its own training, one exploits the very close analogy between Eqs. (1) and (4)—both are formed in the same way from a discrete time sequence, multiplied by a periodic mask—as well as the very close analogy between Eqs. (2) and (5)—both are delay systems. However, the equation for $e(t)$ depends on future values, so it needs to be solved backwards in time. In practice one time-inverts $\bar{e}(t)$ and $J(t)$ before computing $e(t)$ to obtain a linear delayed equation

$$e(q + D) = J(q)[e(q) + \bar{e}(q)], \tag{9}$$

where we use q instead of t to remind oneself that we are dealing with time-inverted signals. We also note that $J(t)$, the derivative of the nonlinear function, is a cosine, which can also be implemented using the intensity modulator. Although this property of the sine function is key for this experiment, other types of nonlinearity can be implemented in analogue hardware (see the discussions).

Experimental implementation.—In the present Letter, we show how Eqs. (2) and (9) can be realized using the same physical setup. Our fibre optics experiment is depicted in Fig. 1. Light is generated by a superluminescent diode (SLD)

emitting in the telecommunications band (1550 nm, with a 33 nm FWHM), modulated by two dual input and dual output Mach-Zehnder modulators (MZM), and attenuated using a programmable optical attenuator used to control the total loop amplification of the system, i.e., μ in Eq. (2). It then propagates through an approximately 1.6 km long spool of optical fibre which provides a total loop delay of 7.93 μs. The light is split and enters two photodiodes, one of which provides the feedback signal. The signals are produced and recorded by digital-to-analog converters (DAC) and analog-to-digital converters (ADC), controlled by a Xilinx Virtex 6 field-programmable gate array (FPGA) chip. The FPGA simultaneously generates the input voltage signals and records the output signals. The FPGA communicates with a computer that controls the whole experiment. (Further details on the experimental setup are given in the Supplemental Material [22]).

The key innovation with respect to the earlier experiments [16,17] is the use of two dual input and dual output MZMs, see Fig. 1, which allows us to implement both Eqs. (2) and (9) using the same physical system. Taking into account the incoherence of light in the two branches between the modulators (see Supplemental Material [22] for details), the output of the upper branch of MZM2 (see Fig. 1) can be found to be

$$I_2^+ = \frac{I_0}{2}[1 + \sin(V_1/V_0)\sin(V_2/V_0)], \tag{10}$$

where I_0 is the input intensity in the upper branch of MZM1, V_1 and V_2 are the driving voltages, and V_0 a constant depending on the MZM. The computational details are presented in the Supplemental Material [22]. In the forward mode, we choose $V_1/V_0 = \pi/2$. Thus, the transfer function acts as a sinusoidal function for the input argument $V_2/V_0 = a(t) + z(t)$. The constant offset $I_0/2$ is removed by the high-pass filter of the amplifier, that drives the MZM. Therefore, once the loop is closed, we end up with Eq. (2). In the backward mode, we drive MZM1 with a voltage $V_1/V_0 = a(q) + z(q) + \pi/2$, and MZM2 with a signal proportional to $\bar{e}(q) + e(q)$, but scaled down sufficiently such that $\sin(V_2/V_0) \approx V_2/V_0 = \bar{e}(q) + e(q)$, which gives the desired functionality for the adjoint system Eq. (9).

In order to train our reservoir computer, we first choose a value of μ close to the threshold for instability. We then iterate the following three steps for (typically) several thousands of iterations, during which performance slowly improves until it converges: (1) We take the training data (typically a small subsequence of the complete set), and convert it to $z(t)$ using the input masks. We feed this signal to the experimental setup, physically implementing Eq. (2). Next, we measure and record the signal $a(t)$, and generate an output sequence y_i using the output masks. (2) From the output and the desired target values we compute the sequence $e_i = \partial C/\partial y_i$ at the output, and convert it to $\bar{e}(t)$, now using the output mask as an input mask. Next, we time-invert it and feed it back into the experimental setup. Simultaneously, we drive the first MZM with the (time-inverted) signal $a(q) + z(q)$ in order to implement the online multiplication with $J(q)$. We record the response signal $e(q)$. (3) From the recorded signals $a(t)$ and $e(t)$, we obtain the gradients for the masking signals, which we use to update the input and output masks

$$\begin{aligned} m(r) &\leftarrow m(r) - \eta dC/dm(r), \\ m_b(r) &\leftarrow m_b(r) - \eta dC/dm_b(r), \\ u(r) &\leftarrow u(r) - \eta dC/du(r), \\ u_b &\leftarrow u_b - \eta dC/du_b, \end{aligned} \tag{11}$$

where η is a (typically small) learning rate. In order to speed up convergence, we applied a slightly more advanced variant of these update rules known as the Nesterov momentum [24,25] (details are given in the Supplemental Material [22]).

Results.—We experimentally validate the above scheme using the system described in Fig. 1 by testing it on three time series processing tasks. We consider first of all the NARMA10 task [26], an academic task often used in the RC community. Here, the input sequence s_i consists of a series of independent and identically distributed random numbers drawn uniformly from the interval [0, 0.5]. The desired output sequence is given by

$$y_i^* = 0.3y_{i-1}^* + 0.05y_{i-1}^* \sum_{n=1}^{10} y_{i-n}^* + 1.5s_i s_{i-9} + 0.1. \tag{12}$$

The second task we will call VARDEL5 (from variable delay). Here, the input sequence consists of independent and identically distributed. digits drawn from the set $\{1, 2, 3, 4, 5\}$. The desired output is then given by $y_i^* = s_{i-s_i}$; i.e., the goal is to retrieve the input instance delayed with the number of time steps given by the current input.

As a performance metric for NARMA10 and VARDEL5 we use the normalized root mean square error (NRMSE), which is given by

$$\mathrm{NRMSE} = \sqrt{\frac{\langle (y_i - y_i^*)^2 \rangle_i}{\langle (y_i^*)^2 \rangle_i}}. \tag{13}$$

The NRMSE varies between 0 (perfect match), and 1 (no relation between output and target).

The third task is a frame-wise phoneme labeling task. We use the TIMIT dataset [27], a speech dataset in which each time step has been labeled with one of 39 phonemes. The input data are high-dimensional (consisting of 39 frequency channels), and the desired output is one of (coincidentally) 39 possible output classes. The goal is to label each frame

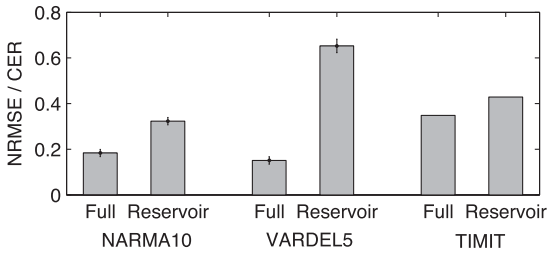

FIG. 2. Comparison of performances for the three tasks under consideration. We show either NRMSE (for NARMA10 and VARDEL5) or the classification error rate (CER) for TIMIT. For each task, we show performance for fully trained systems (Full) vs those trained using the RC paradigm (Reservoir). Error bars indicate standard deviations if available.

in a separate test set. Consequently, the performance metric is now the classification error rate, i.e., the fraction of misclassified phonemes in the test set. Note that the masking scheme and BP algorithm are easily extended to multidimensional input and output sequences (more details are provided in the Supplemental Material [22]). The TIMIT task has been studied before in the context of RC, which has shown it to be challenging, typically requiring extremely large reservoirs to obtain competitive performance [28,29].

For all these tasks, we compared performance of the fully trained system to traditional RC, where we kept the input and bias masks fixed and random, and only optimized their global scaling and the feedback strength parameter μ. Full experimental details for each task may be found in the Supplemental Material [22], together with an example of optimized input masks and of convergence of NRMSE during training. The results are shown in Fig. 2. The experimental setup is successful in performing both useful computations, and implementing its own training process. The fully trained system consistently outperforms the RC approach in all tasks considered.

For the NARMA10 task, we improve over all previous experimental results. The previous best was published in [20], which reported a NRMSE of 0.249 for 50 virtual nodes, and 0.22 for 300 virtual nodes, whereas here, we obtain a NRMSE of 0.185 for 80 nodes (note that, in [20], they report normalized mean square error, which is the square of the NRMSE). That result was obtained on an experimental setup that was specially designed to produce a minimal amount of noise (using a passive cavity as a reservoir). The lowest reported experimental NRMSE on a setup equivalent to ours was 0.41 [17]. Note that we obtain a better average performance for the RC setup (NRMSE $= 0.32$), which is most likely due to the higher number of virtual nodes (80 as opposed to 50 in [17]).

For the VARDEL5 task, we cannot directly compare to literature; however, as pointed out in chapter 5 of [30], this task is an important example of a task that is so nonlinear that it is nearly impossible to solve it with RC. This is confirmed here; the NRMSE of RC is 0.66, indicating that the reservoir has only captured the task on a very rudimentary level. The fully trained system shows a drastically better performance (NRMSE $= 0.15$). This shows that training the input masks not only allows for better performance on existing tasks, but also allows one to tackle tasks that are so intricate that they are considered beyond the reach of traditional RC.

For the TIMIT task, we obtain a classification error rate of 34.8% for fully trained systems, vs 42.9% for the standard RC approach. These results are only slightly worse than similar experimental results presented in [5], (33.2% for the fully trained systems and 40.5% for the RC approach) where 600 virtual nodes were used as opposed to 200 in our case.

Discussion.—The present work confirms the results anticipated in [4,5]: the performance of delay-based reservoir computers can be drastically improved by optimizing both input and output masks. Furthermore, following the proposal of [6], we showed that the underlying hardware is capable of running a large part of its own optimization process. We performed our demonstrations on a fast electro-optical system (whose speed could be readily improved by several orders of magnitude, see, e.g., [15]), and on tasks considered hard in the RC community. Importantly, our work has revealed that the BP algorithm is robust against various experimental imperfections (see the Supplemental Material [22] for details), as the performance gains we obtained on all three tasks were similar to those predicted by numerical simulations.

Although our experiment relies on the sine nonlinearity and its cosine derivative, other nonlinear functions can also be successfully realized in hardware with their derivatives. For instance, the so-called linear rectifier function, which truncates the input signal below a certain threshold, is a popular activation function in neural architectures [31]. Its derivative is a simple binary function which can be easily implemented using an analogue switch, as in [6]. In [32], it is shown how to implement a sigmoid nonlinearity and its derivative, and in [18,20], the nonlinearity is quadratic, and therefore, the derivative, which is linear, should also be easy to implement. Furthermore, the BP algorithm is robust against imperfect implementation of the derivative, as shown in section 4.3 of the Supplemental Material [22], and in the Supplemental Material of [6] (Supplemental Note 4). Therefore, we expect that physical implementation of the BP algorithm will be possible in a wide variety of physical systems.

The current setup still requires some slow digital processing to perform the masking and to compute gradients from the recorded signals. Performing masking operations in analog hardware, however, is actively being researched [33], and these approaches could be used to speed up the present setup. Another limitation is the

relatively slow data transfer between the FPGA and the computer. Implementing the full training algorithm on the FPGA would drastically increase the speed of the experiment. FPGAs have already been demonstrated to be useful for controlling and training electro-optical signal processors [34,35].

Nowadays, there is an increased interest in unconventional, neuromorphic computing, as this could allow for energy efficient computing, and may provide a solution to the predicted end of Moore's law [36]. These novel approaches to computing will likely be made with components that exhibit strong element-to-element variability, or whose characteristics evolve slowly with time. Self-learning hardware may be the solution that enables these systems to fulfil their potential. The results in [6] and in this Letter, therefore, constitute an important step towards this goal.

The authors acknowledge financial support by the Interuniversity Attraction Poles Program (Belgian Science Policy) Project Photonics@be IAP P7-35, by the Fonds de la Recherche Scientifique (FRS-FNRS), by the Action de Recherche Concertée of the Fédération Wallonie-Bruxelles through Grant No. AUWB-2012-12/17-ULB9.

*michiel.hermans@ulb.ac.be
†piotr.antonik@ulb.ac.be

[1] D. Rumelhart, G. Hinton, and R. Williams, *Learning Internal Representations by Error Propagation* (MIT Press, Cambridge, MA, 1986).
[2] P. Werbos, Neural Netw. **1**, 339 (1988).
[3] Y. LeCun, Y. Bengio, and G. Hinton, Nature (London) **521**, 436 (2015).
[4] M. Hermans, J. Dambre, and P. Bienstman, IEEE Transactions on Neural Networks and Learning Systems **26**, 1545 (2015).
[5] M. Hermans, M. C. Soriano, J. Dambre, P. Bienstman, and I. Fischer, J. Mach. Learn. Res. **16**, 2081 (2015).
[6] M. Hermans, M. Burm, T. Van Vaerenbergh, J. Dambre, and P. Bienstman, Nat. Commun. **6**, 6729 (2015).
[7] T. Erneux, *Applied Delay Differential Equations* (Springer Science & Business Media, New York, 2009).
[8] V. Flunkert, I. Fischer, and E. Schöll, Phil. Trans. R. Soc. A **371**, 20120465 (2013).
[9] H. Jaeger, Tech. Rep. GMD Report 152, German National Research Center for Information Technology, 2001.
[10] D. Verstraeten, B. Schrauwen, M. d'Haene, and D. Stroobandt, Neural Netw. **20**, 391 (2007).
[11] M. Lukoševičius and H. Jäger, Computer Science Review **3**, 127 (2009).
[12] C. Fernando and S. Sojakka, in *Advances in Artificial Life: 7th European Conference, ECAL 2003, Dortmund, Germany, 2003: Proceedings*, edited by Wolfgang Banzhaf (Springer, New York, 2003), pp. 588–597.
[13] K. Caluwaerts and B. Schrauwen, in *Conference Proceedings: 2nd International Conference on Morphological Computation, Venice, Italy, 2011* (Ghent University, Ghent, Belgium, 2011).
[14] L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer, Nat. Commun. **2**, 468 (2011).
[15] D. Brunner, M. C. Soriano, C. R. Mirasso, and I. Fischer, Nat. Commun. **4**, 1364 (2013).
[16] L. Larger, M. Soriano, D. Brunner, L. Appeltant, J. Gutierrez, L. Pesquera, C. Mirasso, and I. Fischer, Opt. Express **20**, 3241 (2012).
[17] Y. Paquot, F. Duport, A. Smerieri, J. Dambre, B. Schrauwen, M. Haelterman, and S. Massar, Sci. Rep. **2**, 1 (2012).
[18] K. Vandoorne, P. Mechet, T. Van Vaerenbergh, M. Fiers, G. Morthier, D. Verstraeten, B. Schrauwen, J. Dambre, and P. Bienstman, Nat. Commun. **5**, 3541 (2014).
[19] N. D. Haynes, M. C. Soriano, D. P. Rosin, I. Fischer, and D. J. Gauthier, Phys. Rev. E **91**, 020801 (2015).
[20] Q. Vinckier, F. Duport, A. Smerieri, K. Vandoorne, P. Bienstman, M. Haelterman, and S. Massar, Optica **2**, 438 (2015).
[21] K. Ikeda and K. Matsumoto, Physica D (Amsterdam) **29D**, 223 (1987).
[22] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevLett.117.128301 for mathematical derivations and experimental details, which includes Ref. [23].
[23] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)* (Springer, New York, 2006).
[24] Y. Nesterov, Sov. Math. Dokl. **27**, 372 (1983).
[25] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, in J. Mach. Learn. Res. **28**, 1139 (2013).
[26] A. Atiya and A. Parlos, IEEE Trans. Neural Networks **11**, 697 (2000).
[27] J. Garofolo, *TIMIT Acoustic-phonetic Continuous Speech Corpus.* (Philadelphia, Linguistic Data Consortium, 1993).
[28] F. Triefenbach, A. Jalalvand, B. Schrauwen, and J.-P. Martens, in *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010, Vancouver, B.C., Canada, 2010* (Neural Information Processing Systems, La Jolla, CA, 2010), pp. 2307–2315.
[29] F. Triefenbach, K. Demuynck, and J.-P. Martens, IEEE Signal Process. Lett. **21**, 311 (2014).
[30] M. Hermans, Ph.D. thesis, Ghent University, 2012, http://hdl.handle.net/1854/LU-3171075.
[31] X. Glorot, A. Bordes, and Y. Bengio, in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, FL, 2011*, Vol. 15, p. 275.
[32] B. Shi and C. Lu, U.S. Patent No. 6,429,699 (2002).
[33] F. Duport, A. Smerieri, A. Akrout, M. Haelterman, and S. Massar, Sci. Rep. **6**, 22381 (2016).
[34] P. Antonik, F. Duport, A. Smerieri, M. Hermans, M. Haelterman, and S. Massar, Lect. Notes Comput. Sci. **9490**, 233 (2015).
[35] P. Antonik, M. Hermans, F. Duport, M. Haelterman, and S. Massar, Proc. SPIE Int. Soc. Opt. Eng. **9732**, 97320B (2016).
[36] M. M. Waldrop, Nature (London) **530**, 144 (2016).