Large-Scale Spatiotemporal Photonic Reservoir Computer for Image Classification

Piotr Antonik, Nicolas Marsal, and Damien Rontani

(Invited Paper)

***Abstract*—We propose a scalable photonic architecture for implementation of feedforward and recurrent neural networks to perform the classification of handwritten digits from the MNIST database. Our experiment exploits off-the-shelf optical and electronic components to currently achieve a network size of 16,384 nodes. Both network types are designed within the reservoir computing paradigm with randomly weighted input and hidden layers. Using various feature extraction techniques (*e.g.*, histograms of oriented gradients, zoning, and Gabor filters) and a simple training procedure consisting of linear regression and winner-takes-all decision strategy, we demonstrate numerically and experimentally that a feedforward network allows for classification error rate of 1%, which is at the state-of-the-art for experimental implementations and remains competitive with more advanced algorithmic approaches. We also investigate recurrent networks in numerical simulations by explicitly activating the temporal dynamics, and predict a performance improvement over the feedforward configuration.**

***Index Terms*—Photonics, neuromorphic hardware, reservoir computing, handwritten digit recognition, image classification.**

I. Introduction

THE classification of static images is one of the central problem in Computer Vision in Machine Learning [1], [2] and has a broad range of applications such as in security and authentication with automatic facial recognition [3], [4] or object detection [5], in health sciences with the automatic analysis or segmentation of medical images [6], or intelligent character recognition for handwritten interpretation, such as automatic zip-code identification [7].

Over the last decade, the level of accuracy of automatic image classification techniques has undergone a dramatic improvement coincidentally with to the breakthrough of deep-learning and a particular class of neuro-inspired algorithms, known as convolutional neural networks (CNNs) [8]. CNNs are multihidden-layered, feedforward neural networks trained to extract hierarchical features from images and improve the recognition procedure thanks to their robustness to variability in scale, illumination conditions, and intra-class variations intrinsically present in images.

Deep learning approaches based on CNNs have outclassed simpler classification systems, such as support vector machines (SVM) [9]. As a result, in 2015, a CNN comprised of over 150 layers has surpassed for the first time the accuracy of humans in classifying images from the ImageNet dataset [10], which is made of approximately 14 millions images distributed over more than 21,000 different categories: It reached a classification error rate of 3.57% to be compared to 5.1% for humans. On smaller image datasets, such as the MNIST database made of 70,000 images of handwritten digits distributed over 10 different classes, deep-learning approaches reaches a classification error as low as 0.21% [8]. However, achieving high accuracy requires the training of a large number of hidden layers (*i.e.* the fine tuning of millions of parameters). This is only possible by leveraging (very) large pre-labeled databases and the high-speed, massively parallelizable, computing power from graphical processing units (GPU), thus making the training procedure computationally demanding, usually intractable on regular computers, and energy intensive.

An alternative approach to deep-learning for image classification uses (i) simple feature extraction techniques in combination with (ii) a recurrent neural networks (RNN) and (iii) a simplified training procedure with a limited number of learnable parameters. Reservoir Computing (RC) is a Machine Learning paradigm, which includes echo-state networks (ESN) and liquid-state machines (LSM), to design and train artificial recurrent neural networks [11], [12]. Reservoir Computing exploits the transient response of a randomly-, sparsely-interconnected RNN made of nonlinear dynamical discrete-time nodes, which are subjected to randomly-weighted input signals. The output layer is made of (possibly) several outputs that realize linear combinations of the nodes' transient states. The training procedure for the Reservoir Computer is limited to the optimization of the output layer weights, which is usually obtained by simple linear (or ridge) regression [13]. This leads to a significant alleviation in the computational complexity of the training, because only the final unidirectional connections are optimized by comparison to the entire RNN in typical training procedures for artificial neural networks. Additionally, the reduced number of

Manuscript received April 16, 2019; revised June 15, 2019; accepted June 15, 2019. Date of publication June 24, 2019; date of current version August 20, 2019. This work was supported in part by the AFOSR under Grants FA-9550-15-1-0279 and FA-9550-17-1-0072, and in part by the Région Grand-Est. *(Corresponding authors: Piotr Antonik and Damien Rontani.)*

The authors are with the CentraleSupélec, Metz F-57070, France, and also with the Université de Lorraine, Metz F-57000, France (e-mail: piotr.antonik@centralesupelec.fr; nicolas.marsal@centralesupelec.fr; damien.rontani@centralesupelec.fr).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSTQE.2019.2924138

trainable parameters makes the reservoir computer efficient for classification even on small databases.

Because of its structural simplicity, reservoir computers have motivated many hardware implementations ranging from electronics [14], [15] and spintronics [16], [17], to photonics [18], [19] with the goal of reaching ultra-fast processing speed with an energy consumption at least two order of magnitudes below that of a software-based RC running on a computer. Unprecedented classification speed have been recently achieved on the spoken-digit recognition task from the TI 46 data base with real-time processing speed ranging from 300,000 to 1,000,000 words analyzed per second using laser with optical feedback [19] and optoelectronic oscillators [20], respectively.

The challenge in photonics-based implementations of reservoir computing stems from the experimental difficulty to couple optically a large number (typically greater than 100) of photonic devices together. To overcome this, the time-delay reservoir computer was introduced first with electronic systems [14] and then with various optical/optoelectronic feedback configurations [18]–[23] with applications notably in optical communications [24]. The principle relies on a single nonlinear dynamical node subjected to time-delay feedback, where virtual nodes are located. This approach allows theoretically to increase the size of the network linearly with the length of the feedback loop but at the expense of the processing speed, because data cannot be fed to the systems faster than the time-delay to get all the virtual nodes subjected to the input signal [25].

To overcome this inherent drawback of the time-delay approach, true spatiotemporal photonic reservoir computers have been proposed: (i) using photonic integrated circuits (PICs) limited in scale to a few tens of nodes due to high losses [26]–[28], which is inadequate for tackling complex tasks such as image classification or (ii) using free-space optics with a number of nodes reaching up to 2,500 in the most recent experiments [29], thus making possible for such architecture to solve more complex tasks.

The paper is organized as follows. First, in Section II-A we present the basics of reservoir computing, provide a description of our experimental architecture for a large-scale photonic reservoir computer implementation exceeding 10,000 neurons, and derive the corresponding physical model used in our numerical simulations. Then, in Section III, we introduce the particular task of handwritten digit recognition based on the MNIST database and how our reservoir computer is adapted to solve it. We also describe the various feature extraction strategies (*i.e.* histograms of oriented gradients, zoning, and Gabor filters) to generate the most relevant information from the images prior to their injection in the photonic recurrent neural network. Finally, we devote the Section IV to the discussion of our results, where we compare three conceptually different modes of operation for our reservoir computer to successfully classify handwritten digits: (i) the feedforward mode of operation, (ii) the recurrent mode with echo, and (iii) the recurrent mode with data-splitting. We analyze and compare the performance of these different modes of the reservoir computer when paired with the feature extraction layer. Finally, Section V concludes the paper and summarizes our contributions.

II. Spatiotemporal Photonic Reservoir Computer

In this section, we describe the basics principles of reservoir computing and how we realize and model our experimental spatiotemporal photonic reservoir computer with up to 16,384 nodes.

A. Principles of Reservoir Computing

A reservoir computer consists of a RNN with $n \in \mathbb{N}$ discrete-time nonlinear randomly interconnected dynamical systems, which is mathematically described by the following state equation

$$\mathbf{x}(k+1) = \mathbf{f}_{\mathrm{NL}}\left(W_{res}\mathbf{x}(k) + W_{in}\mathbf{u}(k)\right), \tag{1}$$

where $\mathbf{x}(k) \in \mathbb{R}^n$ is the state vector of the RNN at discrete time $k \in \mathbb{N}$; $\mathbf{f}_{\mathrm{NL}}$ is a nonlinear vector flow mapping the state space $\mathbb{R}^n$ to itself. $W_{res} \in \mathbb{R}^{n\times n}$ is the adjacency matrix of the RNN describing the topology of the weighted interconnections between various neurons. The spectral radius of the matrix W_{res} is usually taken in the range $0.8 - 1.1$ to guarantee the overall stability of the RNN and the fading memory property, one of the necessary condition on the RNN to be used in the framework of reservoir computing [11], [30]. $W_{in} \in \mathbb{R}^{p\times n}$ is the input matrix representing the weighted interconnections between the input layer and the RNN, also known as the *input mask*. Similarly to W_{res}, its weights are randomly distributed with zero mean. Lastly, $\mathbf{u}(n) \in \mathbb{R}^p$ is the input vector.

When subjected to time-varying input data, the state vector of the RNN undergoes complex transient dynamics before returning to a stable quiescent state. These transient dynamics are wire-tapped and weighted to form an output vector $\mathbf{y}(n)$ described by the following linear relation

$$\mathbf{y}(k) = W_{out}\mathbf{x}(k), \tag{2}$$

where $W_{out} \in n \times m$ is the output matrix representing the unidirectional interconnections between the RNN and the ouput layer. The elements of the output matrix, also called *readout weights*, are the only learnable parameters of our reservoir computer architecture. They are trained using a convex optimization problem known as ridge regression (or Tikhonov regularization) [31] that reads

$$W_{out,opt} = \underset{W_{out}\in\mathbb{R}^{m\times n}}{\arg\min} \|\mathbf{Y}_t - W_{out}\mathbf{X}\|^2 + \lambda\|W_{out}\|^2, \tag{3}$$

where $\mathbf{Y}_t \in \mathbb{R}^{m\times k_{tr}}$ is the target matrix containing the k_{tr} targets values for the m outputs of the reservoir computer and $\mathbf{X} \in \mathbb{R}^{n\times k_{tr}}$ is the reservoir states matrix containing the RNN's transient states $\mathbf{x}(k)$ sampled during the training, when inputs vectors are fed to the reservoir during k_{tr} time steps. λ is a positive factor called ridge parameter used to control the norm of the readout weights by penalization; it is usually determined by cross-validation, when $\lambda = 0$ then the ridge regression corresponds to linear regression.

The resolution of the optimization problem to obtain the optimized readout weights in $W_{out,opt}$ is performed either off-line or online [32], *i.e.* either after all the training input data are injected in the reservoir, or in the course of the injection.

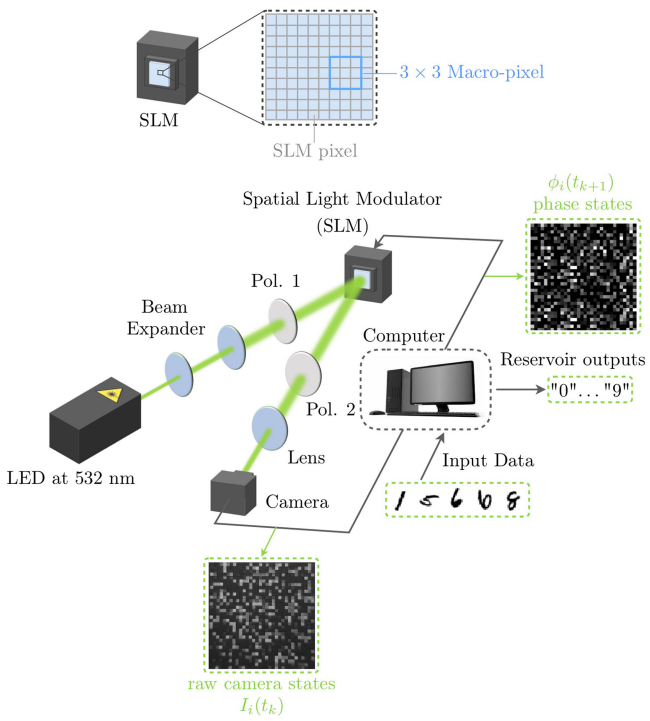


Fig. 1. Scheme of the experimental setup composed of an optical arm linked to a computer. A collimated polarized green (532 nm) LED beam is reflected by the surface of the spatial light modulator (SLM). The SLM is imaged onto a camera through a polarizer (Pol. 2) and an imaging lens. Both the SLM and the camera are driven by a computer running a Matlab script. The latter generates the inputs from the MNIST database images, then computes the pixel values to be loaded on the SLM. Larger macro-pixels made of groups of individual SLM's pixels are used to train the reservoir in such a way to facilitate the separation on the raw camera images. The computer receives the data containing the reservoir states from the camera, computes the outputs and generates the output digits.

B. Experimental Photonic Reservoir Computer

The proposed photonic network is depicted in Fig. 1 and its structure is inspired in parts from recent experiments in Refs. [29], [33]. The setup consists of a photonic arm comprising a collimated incoherent monochromatic light source at 532 nm (Thorlabs M530L3), a set of two polarizers oriented at a 45° angle with respect to the vertical axis, placed before and after a liquid-crystal on Silicon (LCoS), phase-only, spatial-light modulator (SLM) with a 512×512 pixels and a 8-bit resolution (Meadowlark P512 $-$ 0532). This particular combination of polarizing optics and a SLM allows for the phase pattern imprinted in the transverse plan of the optical beam to be nonlinearly converted into an amplitude pattern and hence ensures an all-optical implementation of the nonlinear flow $\mathbf{f}_{\text{NL}}$ from Eq. (1). Finally, the intensity pattern is imaged through a lens on a 1.3-Million-pixels high-speed camera with 10-bit resolution (Allied Vision Mako U130B).

The detected patterns are digitized and processed in the electronic arm of the setup. First, they are linearly combined on a computer (which could potentially be replaced by a DSP board), that effectively realizes the adjacency matrix of the network W_{res} with full control over their choices. Then, input signals are added numerically after the masking operation with the W_{in} matrix. Finally, these signals are fed back to the SLM controller that changes the liquid-crystal orientation of each SLM's pixel at a given discrete time step and hence generates a new phase pattern based on the signals detected at the previous time step. This setup forms a dynamical photonic-based network, which we will be described mathematically as coupled maps in Section II-C.

The size of the reservoir is essentially limited by the lowest resolution between that of the SLM and the camera. In our current setup, this means that, potentially, we could reach a network size of $512 \times 512 = 262,144$ nodes, where each pixel of the SLM, imaged with a $1:1$ ratio on the camera, could be considered as a physical node in our photonic network. However, we only use pixels located at the center of the SLM matrix due to constrains stemming from (i) inhomogeneous intensity distribution of the optical beam and (ii) small misalignment between the SLM and camera optical axes. To mitigate the effects of these experimental imperfections, we regroup neighbouring pixels together to form bigger square-shaped pixels (see Fig. 1), later referred to as *macro-pixels* in this study, and becoming our physical nodes in the photonic network. The smallest macro-pixel effectively usable in our experience is 3×3, which allows for $n = 16{,}384$ macro-pixels available at the center of the SLM. This is currently our experimental upper-bound for the network size, but using a higher-resolution camera and SLM, and possibly smaller macro-pixel sizes with more stringent alignment conditions, our network could easily scale to hundreds thousands nodes.

The speed of the setup is defined by the time needed to compute the next SLM matrix from the raw camera image. This operation is deliberately performed in Matlab for greater flexibility of the experimental scheme. The system is capable of processing 2 images per second with a larger reservoir ($n = 16{,}384$) and up to 7 frames per second with a small reservoir ($n = 1{,}024$). The system's speed limitation can be alleviated by replacing the computer with a dedicated digital signal processing (DSP) board, or a field-programmable gate array (FPGA) chip, capable of performing the matrix-products computations in real time (as in e.g. [32]). Matrix multiplication can also be offloaded to fully parallel optics [29], [34]. Furthermore, a dedicated computing unit (FPGA or DSP board) could also take care of the feature extraction stage, thus allowing real-time processing of the input images.

C. Physical Modelling of the Photonic Reservoir Computer

In this section, we propose a phenomenological modelling of the dynamics of our photonic network.

We first model the optical arm and assume, for simplicity, that the optical field $\mathbf{E}_i(t_k)$ and phase pattern $\phi_i(t_k)$ are homogeneous and constant over the surface of the i-th macro-pixel during $\Delta t = t_{k+1} - t_k$, the sampling time of our electronic feedback loop. Furthermore, using only the macro-pixels close to the optical axis of the SLM, we also assume homogeneity of the optical field over the entire $n = 16{,}384$ macro-pixels and hence $\mathbf{E}_i(t_k) = \mathbf{E}(t_k)$ and its constant value $\mathbf{E}(t_k) = \mathbf{E}_0$ during our experiment.

The incoherent, unpolarized light source is filtered by a linear polarizer oriented at 45° with respect to the vertical axis, hence leading to the following expression $\mathbf{E}_0 = E_0/\sqrt{2}[1,1]^T$.

Each SLM's macro-pixel is composed of liquid-crystal cells on top of a highly-reflective surface. The liquid crystal cells of the i-th macropixel can be considered as a birefringent material with its fast-optical axis along the vertical direction and with a programmable phase-shift $\phi_i(t_k)$. Hence, the Jones Matrix for the SLM's i-th macropixel is given by

$$J_{SLM,i}(t_k) = \begin{pmatrix} e^{i\phi_i(t_k)/2} & 0 \\ 0 & -e^{-i\phi_i(t_k)/2} \end{pmatrix}. \quad (4)$$

After reflexion on the SLM i-th macro-pixel and transmission through the second polarizer oriented at 45° and described by the Jones matrix $J_{p,45}$, the output field is given by $\mathbf{E}_{out,i}(t_k) = J_{p,45}J_{SLM,i}(t_k)\mathbf{E}_0 = \mathbb{i}\frac{E_0}{\sqrt{2}}\sin\left(\frac{\phi_i(t_k)}{2}\right)[1,1]^T$. The intensity detected by the camera and associated to the i-th macro-pixel hence reads

$$I_i(t_k) = \|\mathbf{E}_{out,i}(t_k)\|^2 = I_0\sin^2\left(\frac{\phi_i(t_k)}{2}\right), \quad (5)$$

with $I_0 = \|\mathbf{E}_0\|^2$ the constant and uniform optical intensity emitted by the LED.

The camera has a 10-bit resolution and detects a quantified version of the intensity of the i-th macro-pixel on the SLM that we will denote $\lfloor I_i(t_k)\rfloor_{10}$. We define the n-dimensional state vector of the photonic network from the macro-pixel intensities detected by the camera by

$$\mathbf{x}(t_k) = [\lfloor I_1(t_k)\rfloor_{10}, \ldots, \lfloor I_n(t_k)\rfloor_{10}]^T. \quad (6)$$

The 10-bits quantified intensities detected by the camera (corresponding to the SLM's macro-pixels) are linearly combined together and masked input data is added digitally. This operation on the states are then quantified over 8 bits before being fed back to the SLM controller so that phase values of the SLM's macro-pixels $\phi_i(t_{k+1})$ for $i = 1, \ldots, n$ are updated and so are the intensities $I_i(t_{k+1})$. The output of the reservoir are determined by combining linearly the intensities detected by the cameras. This leads to the following state and output equations

$$\mathbf{x}(t_{k+1}) = \left\lfloor I_0\sin^2\left(\lfloor W_{res}\mathbf{x}(t_k) + W_{in}\mathbf{u}(t_k)\rfloor_8\right)\right\rfloor_{10}, \quad (7)$$

$$\mathbf{y}(t_k) = W_{out}\mathbf{x}(t_k). \quad (8)$$

Equation (7) should be understood as the nonlinear function $\lfloor I_0\sin^2(\lfloor\cdot\rfloor_8)\rfloor_{10}$ applied to each coordinate of the n-dimensional phase space. Equations (7)-(8) are the tailored version of Eqs. (1)-(2) specific to our photonic implementation. This model will be later used in our numerical simulation to assess the performance of the photonic reservoir computer.

III. Reservoir Computing With Feature Extraction for Image Classification

Feature extraction is a common approach in computer vision to enhance the classification system, here the photonic reservoir computer, by providing the most relevant information in images. In this section, we first introduce the approach adopted with reservoir computer to solve the MNIST handwritten digit recognition task and then present the five feature extraction techniques that we tested in this study.

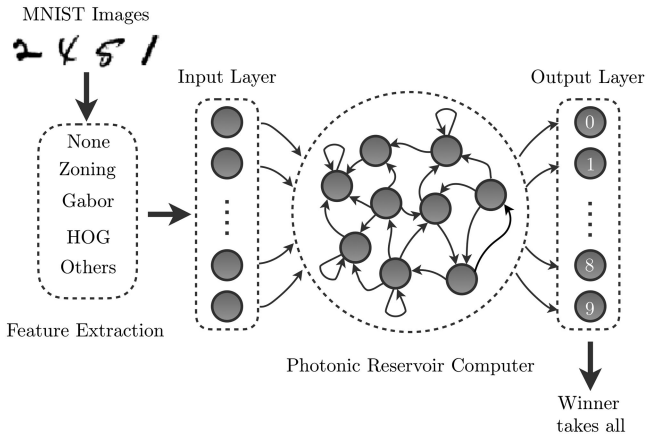


Fig. 2. Handwritten digit recognition in the context of photonic reservoir computing with feature extraction. The images from the MNIST database undergo a feature extraction stage, where different algorithms under investigations are applied (or none, in the case of raw images). The resulting features are fed into the photonic reservoir computer with 10 binary output nodes, one for each digit. The nodes are trained to output 1 for the digit associated with the node and 0 for the other digits. The final classification is obtained by selecting the node with the maximum output, *i.e.* the winner-takes-all decision strategy.

A. Resolution of the Handwritten Digit Recognition Task From the MNIST Database With Reservoir Computing

The principle of the handwritten digit recognition in the context of reservoir computing with feature extraction is illustrated in Fig. 2. In this work, we used the popular MNIST database [35], publicly available online, which contains 70,000 images of handwritten digits from "0" to "9". All images have been normalized to fit into a 28×28 pixels bounding box, anti-aliased, and converted into gray-scale levels.

The input images can be pre-processed in several ways to the input layer of the reservoir computer, depending on the type of feature extraction presented in Sub-section III-B, and it will be shown in Section IV that this choice impacts the classification error significantly.

The output layer is made of 10 binary outputs, introduced to recognize the 10 different class of digits. Hence, each binary output is trained to give a "1" for an image of the digit it is associated to and "0" for all other digits. The winner-takes-all decision strategy is used to classify each image based on the binary output with the highest value.

Furthermore, the photonic reservoir computer can have different modes of operation and processes the masked input data differently depending on how the latter is fed into the reservoir. In this study, we consider three different modes of operation.

Full-image feedforward mode – In this mode, one full MNIST image is sent to the reservoir at each timestep t_k. To avoid cross-talk between consecutive images caused by the internal memory of the system, the reservoir should be exploited as memoryless map with $W_{res} = 0_{n\times n}$. Therefore, the state equation of the reservoir reduces to

$$\mathbf{x}(t_{k+1}) = \left\lfloor I_0\sin^2\left(\lfloor W_{in}\mathbf{u}(t_k)\rfloor_8\right)\right\rfloor_{10}. \quad (9)$$

The results obtained with this approach will be presented in Section IV-A.

TABLE I
FEATURE EXTRACTION TECHNIQUES INVESTIGATED IN THIS WORK

Feature extraction method	Input dimensionality
Raw images	784
Zoning 2	196
Zoning 4	49
Gabor filters	40 – 1152
HOG	324 – 1296

Full-image recurrent mode – A more advanced mode of operation is to present a full MNIST image at a given timestep t_k and let the reservoir process the information during k_e additional timesteps until the system relaxes into a quiescent state. Then, a new image can be sent at t_{k+ke+1}. In this mode of operation, $W_{res} \neq 0_{n\times 0}$ and the state equation of the reservoir is given by Eq. (7). We will discuss this approach further in Section IV-B.

Column-wise recurrent mode – The third mode of operation for the reservoir is inspired by Ref. [36] and consists of dividing each MNIST image into 28 columns. The image is then presented at a rate of one column per time step k, so that it takes 28 timesteps to feed a single image to the reservoir. In this mode of operation, the dynamics of the network is described by Eq. (7) with $W_{res} \neq 0_{n\times 0}$. The underlying idea here is to transform the spatial correlations naturally present in images into temporal correlations in the dynamical states of the reservoir and thus exploit the internal memory of the system created by the recurrence of the network. The disadvantage, however, is a processing speed divided by 28 compared to the first mode of operation (full-image feedforward mode). The results for this mode of operation will be presented in Section IV-C.

B. Feature Extraction Approaches

We implemented and compared five feature extraction techniques, also considered in [37]. The three approaches that gave the best results are presented in this section, and their results will be discussed in Section IV. Table I summarizes these techniques and their respective input dimensionalities (*i.e.* the number of features). The two other approaches that did not yield acceptable classification errors are only briefly discussed at in the end of this section.

1) Raw Images: We used raw images – *i.e.* without any preprocessing – as a benchmark for the feature extraction methods considered below. In this particular case, grey-scale values of the pixels are used as inputs to the reservoir computer, and the dimensionality of the input is $28 \times 28 = 784$. An example of raw image is shown in Fig. 3(a).

2) Zoning: Introduced in Ref. [38], it is the simplest feature extraction technique considered here. It is a statistical region-based approach, where the image is divided into smaller zones and pixel densities are computed in each one. In other terms, this method consists of the combination of a convolution of the image by a filter, followed by pooling a single value from each zone of the image.

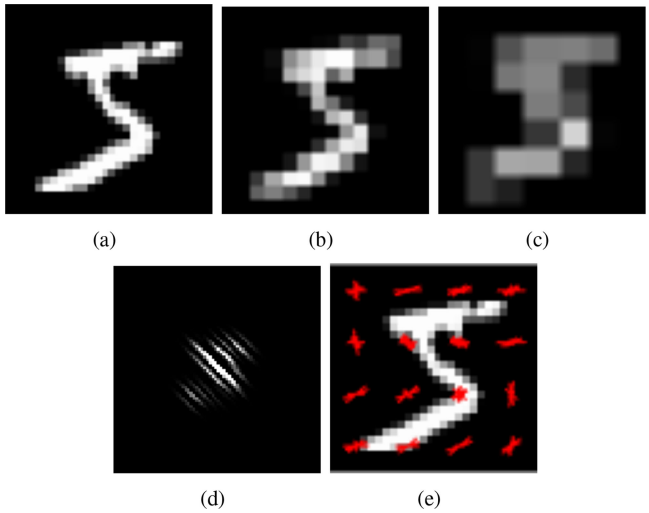


Fig. 3. Illustration of **(a)** raw MNIST image, and the feature extraction methods, applied to it: **(b)** Zoning 2, **(c)** Zoning 4, **(d)** Gabor with $\lambda = 3$ and $\theta = 22.5°$ orientation capturing the middle stroke of the digit, and **(e)** HOG features (red) computed with 7×7 cells and 9 orientation bins, superimposed on top of the source image.

Here, we consider two variants of this approach, designed specifically for the MNIST database: *Zoning 2* with 2×2-pixel zones, and *Zoning 4* with larger 4×4-pixel zones. The filters are 2×2 or 4×4 matrices filled with 1s, thus giving a non-normalized average pixel values for each zone. This technique allows to reduce the input-dimensionality by "dropping" 3 out of 4, or 15 out of 16 pixels for Zoning 2 and Zoning 4, respectively. Consequently, the input dimensionality is 196 for Zoning 2, and 49 for Zoning 4. The resulting inputs to the reservoir computer are illustrated in Figs. 3(b) and 3(c).

3) Gabor Filters: They are linear filters mainly used for texture analysis [39] and have been also used to model the receptive field profiles of simple cells in mammalian visual cortex [40]–[43]. Therefore, pre-processing images with Gabor filters could be compared, to a certain extent, to perception in the human visual system. The 2D-impulse response $h(x, y)$ of a Gabor filter is defined by the product of a Gaussian function with a sine/cosine function. There are either complex-valued Gabor filters [40] or pairs of Gabor filters with quadrature-phase relationship [41]–[43]. In this work, we use real-valued, even-symmetric Gabor filters [39], [44], which impulse response is given by

$$h(x, y) = \exp\left(-\frac{x_\theta^2}{2\sigma_x^2} - \frac{y_\theta^2}{2\sigma_y^2}\right)\cos\left(\frac{2\pi}{\lambda}x_\theta\right), \qquad (10)$$

with

$$x_\theta = x\cos(\theta) + y\sin(\theta), \qquad (11)$$

$$y_\theta = y\cos(\theta) - x\sin(\theta), \qquad (12)$$

where λ is the wavelength of the cosine function with orientation θ with respect to the x-axis and $\sigma_{x,y}$ are the space constants of the Gaussian envelope along the x, y axes, respectively. A Gabor filter seeks for a specific frequency content in a direction θ by yielding higher values in locations visually similar

to the filter itself after convolution with the image. The dimensionality of the resulting features depends on (i) the number of filters, differentiated by their lengths and orientations, and (ii) the computation of the local energy, that will be discussed in Section IV-A3.

4) Histograms of Oriented Gradients (HOG): This algorithm is widely used in computer vision for object detection and location [37] in static images. It was originally proposed in Ref. [45] and is based on scale-invariant features transform (SIFT) descriptors [46]. The HOG descriptors are obtained by creating histograms on the magnitude of images gradient $m(i,j) = (D_x(i,j)^2 + D_y(i,j)^2)^{1/2}$ depending on their orientation $\theta(i,j) = \arctan(D_y(i,j)/D_x(i,j))$ with $D_x(i,j)$ and $D_y(i,j)$ the horizontal and vertical approximate gradients evaluated at pixel (i,j). These gradients are computed using the two spatial filters $G_x = [-1,0,1]$ and $G_y = [-1,0,1]^T$.

A typical histogram consists of 9 bins in the $[0°, 180°]$ interval. It is calculated on 8×8-pixel cells within the image. This ensures a dimensional reduction from 64 gray-scale intensity values to 9 cumulative frequencies values corresponding to the so-called HOG descriptors. To make them robust to lighting inhomogeneities, the histograms are block-normalized over larger regions of pixels within the image. Similarly to Zoning, the number of features depends on the size of the image and the cell, the number of bins, and the block-normalization. In the case of the MNIST database, the total number typically ranges between 234 and 1296. This will be further discussed in Section IV-A4. Fig. 3(e) illustrates the resulting gradients superimposed on top of an example image from the MNIST dataset.

C. Other Methods

The following two feature extraction methods were investigated in preliminary simulations, but were not included in the results (Section IV) since they performed worse than the raw images. Moreover, both methods require the distinction between foreground and background pixels, and are thus more complex to implement on real-life images.

1) Projection Histograms: Projection histograms (vertical and horizontal) were considered in [37]. This approach consists in counting the number of white foreground pixels in each column or row, respectively. We used both histograms simultaneously – that is, 56 features for each image – and obtained classification errors of order of 12%.

2) Distance Profiles: This descriptor calculates the pixel-distance between the border of the image and the first pixel of the foreground (*i.e.* the digit) [47]. Based on the four borders of the image, four profiles can be considered: left, right, top, and bottom. In this study, we investigated the combination of all four profiles, since it yields a descriptor with the most information on the image, encoded into 112 features. This approach works better than the projection histograms above, but we could not reduce the classification error below circa 10%.

IV. Results

The results of this study are divided into three sections, based on how the MNIST images are processed by the reservoir computer (see Section III).

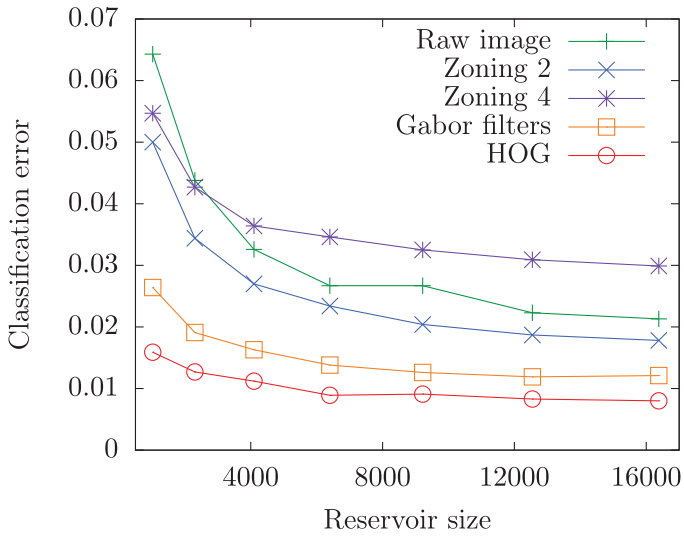


Fig. 4. Numerical results obtained with different feature extraction methods and various reservoir sizes. Raw images (unprocessed pixels) are shown for comparison. Zoning 2 technique performs slightly better, while Zoning 4 yields higher classification errors with larger reservoirs. Gabor filters significantly improve performance in comparison to other techniques, but fall short of the HOG algorithm that produced the best results in this study: 0.80% with the largest reservoir of $n = 16{,}384$ nodes.

A. Full-Image Feedforward Mode

Presenting one full MNIST image per time step k is the simplest approach considered here. Besides the simplicity, it has the advantage of (i) being compatible with feature extraction techniques, presented in Section III-B, and (ii) not increasing the number of processing time steps, since it is equal to the size of the MNIST database. The downside, however, is that it does not exploit the temporal dynamics of the reservoir computer. In fact, since the individual MNIST images are independent from each other, the inputs to the reservoir lack the temporal dependence required to make use of the recurrence of the network. Intuitively, one expects the system to perform better without memory, so that the classification of the current image would not be influenced by the previous input. This assumption was confirmed in both experiments and numerical simulations. Therefore, the recurrent reservoir computer is reduced to a simple feedforward network in this case.

Figure 4 presents a summary of the numerical results obtained with the different feature extraction techniques introduced in Section III-B. We simulated reservoir sizes from 1,024 up to 16,384 nodes. In all cases, the classification error decreases with the number of neurons. At first glance, the HOG features produces the lowest classification errors, while raw images and Zoning 4 perform the worst. We will further discuss each feature extraction methods in the following sections and present the experimental results for comparison.

1) Raw Images: Figure 5 presents the numerical (green) and experimental (black) results obtained with raw images. That is, unprocessed pixels were used as inputs to the reservoir computer. We used these results as benchmark for other feature extraction techniques. In simulations, raw images were classified with a 6.4% error with the smallest reservoir ($n = 1{,}024$ nodes) and 2.1% with the largest reservoir ($n = 16{,}384$ nodes). In

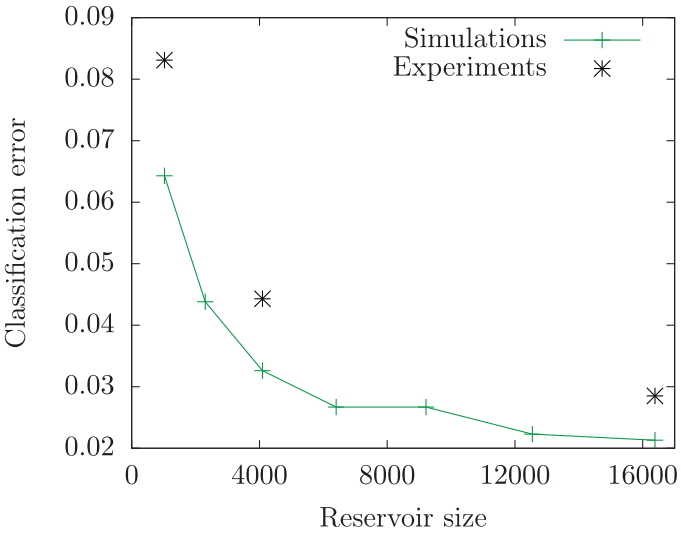


Fig. 5. Numerical (green curve) and experimental (black markers) results obtained with the raw images, without any pre-processing or feature extraction. While the experimental results do not match the simulations exactly, they present a similar trend.

experiments, we obtained a 8.31% and 2.85% error with the smallest and the largest reservoir, respectively.

2) Zoning: Figure 4 presents the numerical results obtained with both zoning methods – Zoning 2 with 2×2 windows and Zoning 4 with 4×4 windows. This technique, as presented in Section III-B, is the simplest approach considered here, that consists in averaging pixel values over windows of fixed size. The Zoning 2 technique allows to slightly improve the results in comparison with the raw images. Since its effect can be seen as dimensionality reduction, this result indicates that the default 28×28 MNIST images contain some undesirable information that can be removed to improve performance. Zoning 4 technique, on the other hand, performs worse than the raw images, which indicates that too much information is lost when averaging over 4×4 windows.

Comparing the different curves in Fig. 4, one notes that both Zoning techniques provide the lowest classification error improvement in comparison with the other feature extraction techniques. Therefore, we conclude that those two techniques are not suitable for handwritten digit recognition task with reservoir computing. However, they remain of interest as they bring out characteristic spatial scales of the MNIST images.

3) Gabor Filters: The case of the Gabor filters perfectly illustrates the main difficulty of the feature extraction approach to classification – the user has to find the best set of filters for the task at hand, which often requires good understanding of the task under investigation.

At first, we tried the simplest approach that consists in computing the energy of the convolution of the filter with the image. In this way, one obtain a single number for each filter. Preliminary simulations with 40 Gabor filters of 8 different orientations and 5 differents lengths have shown a classification error of circa 19%, which is significantly worse than with raw images (see Fig. 4). This result suggests that global features extracted from MNIST images do not contain enough information to distinguish the digits. An intuitive example illustrating the inefficiency of global features is the particular case of separating the digits "6" and "9". Since the former is very similar to the latter after a rotation by π, the energy computed from the Gabor filters is approximately the same for both digits, making them indistinguishable from the global features point of view. Others digits, such as "8" and "0", with comparable curves, also yield energies confusing for the classifier. In order to distinguish these digits, the classifiers needs to know where exactly a particular direction is present in the image, and this is where one needs to use the local features.

At the next stage, we computed local energies by dividing images into smaller windows. This adds another tunable parameter to the feature extraction technique: the size of the windows. A sensible choice requires the knowledge of characteristic scales of the MNIST images, and a reasonable first guess can be based on the insights obtained with the zoning techniques, presented in Section IV-A2. Therefore, we computed the energy locally on a 7×7 grid, that is, within 4×4 windows. We fixed 8 directions θ – from 0° to 157.5° by 22.5° steps – and tried different lengths of the filter – from 2 to 8. The lowest classification error of 3.19% was obtained with the filter length $\lambda = 7$ pixels, and the smallest reservoir ($n = 1{,}024$ nodes). Local energy computation increases the number of features by a factor of 49 in this case: that is, with 8 Gabor filters, one obtains 392 features.

At this point, the performance of the reservoir computer is significantly better than with raw images, but the HOG approach remains unmatched (see Fig. 4). Therefore, we tried to improve the features obtained with the Gabor filters by taking inspiration from the HOG algorithm. Dalal and Triggs emphasize the importance of block-normalization: without it performance drop by 27% in comparison [45]. In the next series of numerical simulations, we added 2×2 L2-norm block-normalization. The immediate effect of this operation is the increase of the number of features by a factor of 4 – from 392 to 1,152. Remarkably, we obtain a 26.5%-improvement with 8 Gabor filters of length $\lambda = 3$. Simulating the system with filters of different lengths, we obtain the lowest classification error of 2.67% with filters of length $\lambda = 5$, and the smallest reservoir ($n = 1{,}024$ nodes). This corresponds to a 22.1% improvement gained by adding the block-normalization.

In summary, Gabor filters could only achieve a circa 19% classification error in the preliminary tests. However, hand-tailoring the filters to the MNIST database (*i.e.* adding local-energy computation, block-normalization, and setting the filter length to 5), we managed to reduce the error down to 2.67% with the smallest reservoir ($n = 1{,}024$) and 1.21% with the largest reservoir ($n = 16{,}384$). We did not manage to outperform the HOG algorithm with Gabor filters in this study, but gained additional insights on the MNIST database and how the components of the HOG algorithm contribute to its performance (in particular, the block-normalisation).

4) Histograms of Oriented Gradients: The histograms of oriented gradients is a powerful feature descriptor in computer designed for the purpose of object detection [45]. Similarly to the Gabor filters, the algorithm requires the tuning of a few parameters for better performance. As described in Section III-B4,

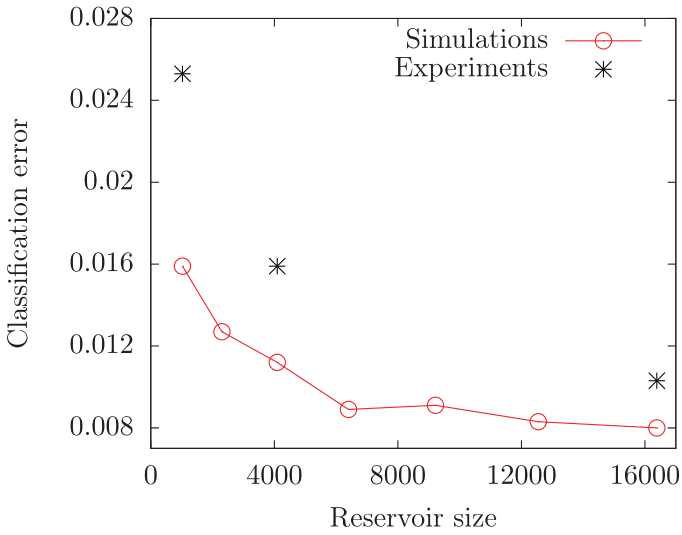


Fig. 6. Numerical (red curve) and experimental (black markers) results obtained with the histograms of oriented gradients. Similarly to the raw images, one observes a certain offset with the small and the midsize reservoir. The numerical and experimental performances obtained with the largest reservoir are fairly close, on the other hand.

TABLE II
PERFORMANCE OF VARIOUS STATE-OF-THE-ART DIGITAL APPROACHES COMPARED TO OUR BEST RESULTS ON THE MNIST DATABASE (WITH $n = 16{,}384$)

Authors	Method	Performance
Wan et al. [48]	CNN	0.21%
Wang et al. [49]	K-Means	0.35%
Keysers et al. [50]	K-NN	0.52%
Decoste et al. [51]	SVM	0.56%
This work (simulations)	**Photonic RC**	**0.8%**
Kégl et al. [52]	Boosted Stumps	0.87%
Schaetti et al. [36]	RC	0.93%
Salakhutdinov et al. [53]	DBM	0.95%
This work (experiment)	**Photonic RC**	**1.03%**
LeCun et al. [35]	LeNet-4	1.1%
LeCun et al. [35]	Non-linear classifier	3.3%
LeCun et al. [35]	Linear classifier	7.6%

the key parameters are the cell size, the number of bins, and the size of the normalisation blocks. In the case of the MNIST classification task, the generally recommended values for the last two parameters – 9 bins and 2×2 block-normalisation – produce the best results. As for the cell size, the intuitive guess that smaller cells capture a better representation of the image does not hold here. In fact, the Zoning approach (see Section IV-A2) demonstrated that more information does not always result in better classification. Preliminary numerical simulations with 4×4-pixel cells produced a classification error of 2.1% with the smallest reservoir ($n = 1{,}024$). Upon increasing the cell size up to 7×7 pixels, we reduced the error down to 1.59% with the same reservoir size. Furthermore, the number of features was cut from 1,296 down to 324. Further decrease of the number of features by limiting the number of bins down to 4 (instead of 9) did not improve performance.

Figure 6 presents the numerical (red) and experimental (black) results obtained with the HOG features and different reservoir sizes. In simulations, the MNIST digits were classified with a 1.59% error with the smallest reservoir ($n = 1{,}024$ nodes) and 0.8% with the largest reservoir ($n = 16{,}384$ nodes). In experiments, we obtained a 2.53% and 1.03% error with the smallest and the largest reservoir, respectively.

These results are of the same order of magnitude as the best performance found in the literature, summarized in Table II. At the moment of writing this article, the best performance on the MNIST task was obtained with CNNs [48] with numerous approaches reporting classification errors below 1% (they will not be exhaustively presented here). We selected and listed several alternative methods to CNN, not relying on neural networks, such as SVM, k-means, k-nearest neighbours (KNN), and boosted stumps. Table II shows that our photonic RC outperforms, in numerical simulation, several deep approaches, such as deep Boltzman machines (DBM). Furthermore, the experiment produces lower error than the earlier versions of CNNs, such as LeNet-4 [35]. One should keep in mind the difference between a shallow neural network, implemented in photonic hardware, and a complex convolutional neural network, implemented on ideal noiseless digital processor. Finally, the reservoir computer significantly improves the performance compared to non-linear and linear classifiers.

B. *Full-Image Recurrent Network*

This second approach is similar to the first one (see Section IV-A) in the way that the reservoir computer is presented with the full MNIST image and the features extracted from it. The difference lies in the fact that each subsequent image is input with a fixed delay, so that the temporal dynamics of the reservoir can be exploited to process the information. The transients, *i.e.* the states of the nodes induced by the input, are recorded and concatenated together into a single reservoir state, used for training. This approach effectively increases the number of trainable parameters, since the size of W_{out} is now proportional to the duration of the transients. Consequently, it also lengthens the overall processing and makes the training process more computationally expensive, especially the matrix inversion operation.

The main downside of this technique is the increased number of hyper-parameters of the reservoir. That is, on top of the input gain, the properties of the W_{res} matrix (such as the spectral radius, the amplitude of the diagonal elements, and others) have to be optimized to improve the results. As experimental optimization of the hyper-parameters requires long runs in practice, we only investigated this method in numerical simulations.

Figure 7 presents the numerical results obtained with this approach. The classification error is plotted against the number of transients recorded for training using the same feature extraction techniques as in Section IV-A. All results were obtained with the smallest reservoir ($n = 1{,}024$) and optimized hyper-parameters. The left-most point of each curve corresponds to the feedforward network approach, that is, no transient is recorded and the images are input at each time step k. The graph reflects the two major observations produced by this study: (i) the additional transients improve the classification performance for all feature extraction

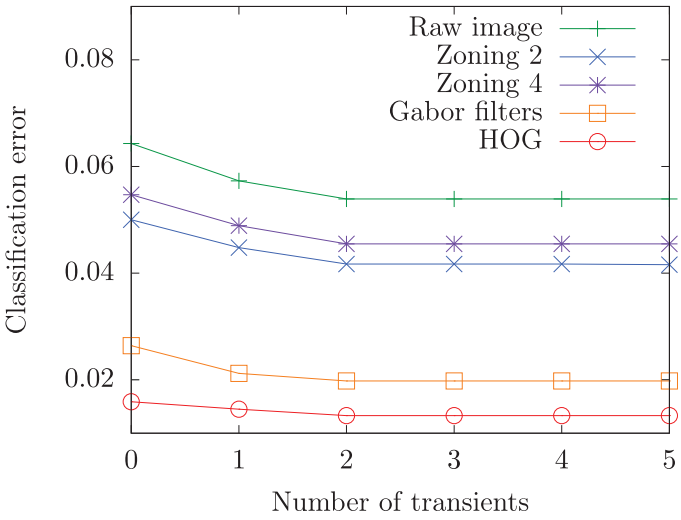


Fig. 7. Numerical results obtained with the recurrent network ($n = 1{,}024$) approach and various feature extraction methods. Different lengths of transients have been investigated: zero transients corresponds to the feedforward network case, discussed in Section IV-A. All curves decrease until two transients and then remain constant.

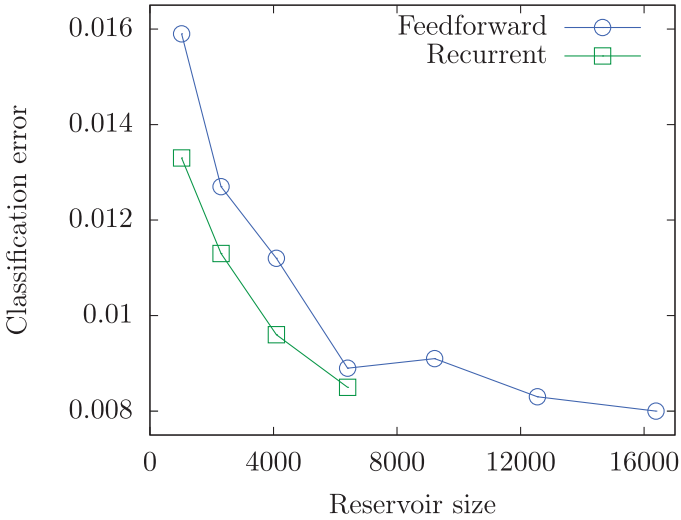


Fig. 8. Numerical results obtained with the histograms of oriented gradients, processed by a feedforward network (blue) and a recurrent network (green) with 2 transients. Due to computational complexity, we could only train reservoirs up to $n = 6{,}400$ nodes.

TABLE III
PERFORMANCE IMPROVEMENT OFFERED BY THE RECURRENT APPROACH WITH THE SMALLEST RESERVOIR ($n = 1{,}024$)

Feature extraction method	Feedforward	Recurrent	Gain
Raw images	0.0643	0.0539	16.2%
Zoning 2	0.0500	0.0417	16.6%
Zoning 4	0.0547	0.0455	16.8%
Gabor filters	0.0264	0.0198	25.0%
HOG	0.0159	0.0133	16.4%

techniques (including raw images), and (ii) two transients is the optimal setting for all cases. Table III contains the performance improvements obtained with 2 transients as compared to the feedforward network.

Since the additional transients improve the classification error for the smallest reservoir ($n = 1{,}024$), we investigated its effects on larger reservoirs. For simplicity, we only considered the HOG features, since they yield the lowest error rate with the feedforward network (see Section IV-A4). At this stage, we could perceive the increased complexity of training a reservoir with transients, and could only perform numerical simulations up to $n = 6400$ nodes on our desktop computers with 32 GB of RAM. Figure 8 compares the classification errors obtained with feedforward and recurrent networks. The optimal number of transients was found to be 2, similarly to the previous observations. Table IV lists the performance gains for different reservoir sizes.

In summary, although the recurrent approach increases the number of trainable parameters, lengthens the training process and makes it more computationally expensive, it offers interesting results that have to be taken into account for improving the performance of our reservoir.

TABLE IV
PERFORMANCE IMPROVEMENT OFFERED BY THE RECURRENT APPROACH WITH THE HISTOGRAMS OF ORIENTED GRADIENTS AND DIFFERENT RESERVOIR SIZES

Reservoir size (n)	Feedforward	Recurrent	Gain
1024	0.0159	0.0133	16.4%
2304	0.0127	0.0113	11.0%
4096	0.0112	0.0096	14.3%
6400	0.0089	0.0085	4.5%

C. Column-Wise Recurrent Mode

The third approach, inspired by [36], uses an explicit temporal encoding of the spatial visual information from the MNIST images in order to activate the recurrent dynamics of the reservoir computer. The idea here is to split the full image into smaller portions and feed them sequentially into the classifier. The separation can be done in various ways: columns or rows (overlapping or adjacent), sliding windows, sub-images, among others. Similarly to [36], we consider non-overlapping columns, which transforms each image into 28 inputs of 28 dimensions. Therefore, such temporal encoding reduces the input dimensionality, but increases the processing time proportionally.

After defining the encoding procedure, the specifics of the reservoir training have to be considered. That is, as each image corresponds to 28 inputs, which in turn give birth to 28 reservoir states, one needs to define how to use those states to train the output weights. We chose to investigate two approaches here:

- The first training procedure consists of considering each column as an individual input and train the reservoir computer to output the target digit on every input column forming the corresponding image. Intuitively, one does not expect the system to produce the correct class at the first input column, but assumes the output to converge to the target digit with the last columns of the image. Therefore, while

all 28 reservoir states are used for training for each image, the evaluation should only be performed on the basis of the last columns. In this work, we chose to select the last 7 columns and assign the final decision to the most frequent class among the 7 outputs.

- The second training method is inspired by [36] and consists in combining a selection of reservoir states at specific time steps into one, and computing the output weights based on the aggregate reservoir state. This idea is fairly similar to the recurrent network approach (see Section IV-B), as it effectively increases the network size by combining several reservoir states into one, and thus expanding the number of trainable parameters.

The results of these two approaches are presented in Sections IV-C1 and IV-C2, respectively. We used a small reservoir computer ($n = 1{,}024$) in all simulations for practical reasons.

1) Training With Individual Reservoir States: The first training option appears more challenging for the reservoir computer, since it requires a system with memory long enough to keep all relevant columns of the image. In fact, we obtained a relatively high classification error of 21.2%. Upon investigation of the reservoir signals we noticed that while the outputs converge, as expected, they often converge to a incorrect class. This observation suggests that individual columns do not contain enough relevant information to allow the classifier to make the correct decision.

2) Training With Aggregation of Reservoir States: The second training option alleviates the issue encountered with single columns, as the reservoir computer is trained on an aggregation of several reservoir states activated by their corresponding input columns. This methods also relaxes the memory requirements, as the explicit aggregation of reservoir states no longer forces the network to keep the input information in memory.

The choice of the reservoir states to aggregate for the training is crucial in this approach. That is, training the system on the combination of the first three columns, for instance, is unlikely to produce low classification errors. For this reasons, we started by fixing the hyperparameters of the reservoir computer to reasonably good values (yet not the most optimal), and tested, in numerical simulations, all possible combinations of training with two reservoir states (that is, 378 trials). We found that the combination of the 17-th and 24-th reservoir states yielded the lowest error. Furthermore, we observed that no states below 10-th were chosen in the top ten results. We thus conclude that the earlier reservoir states (from the first to circa the 10-th) do not contain enough information on the digit to allow precise classification, and the later states (from 11-th to the last) should be privileged. Then, we performed the same series of trials with 3 reservoir states. In order to reduce the number of trials from $n = 3{,}276$ down to 969, we ignored the reservoir states from 1 to 10, based on the observation above. We found the best combination of states to be the 14-th, the 18-th, and the 26-th. Finally, we tried the combinations of 4 states, and found the optimal one to be the 14-th, the 16-th, the 20-th, and the 24-th.

After defining the optimal combinations of reservoir states to aggregate for the training, we kept them fixed, and performed the optimisation of the hyperparameters. Figure 9 presents the classification errors with optimal training and reservoir settings. For completeness, we also evaluated the scenario with one column, and obtained a 11.7% error with the 17-th column. The best performance of 4.42% was produced with the combination of 4 reservoir states. Compared to the results obtained with raw images, discussed in Sections IV-A1 and IV-B, this method yields lower errors and thus presents the best approach on raw images among those considered in this study. Nevertheless, the 4.42% classification error is no match for the HOG technique (see Section IV-A4) or the state-of-the-art.

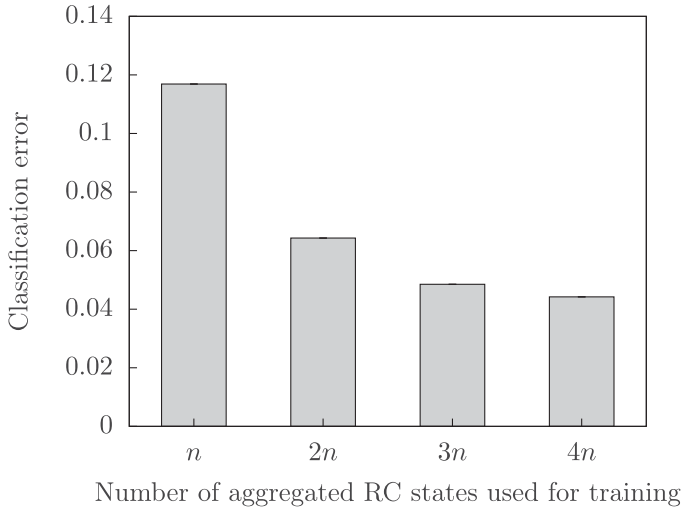


Fig. 9. Numerical results obtained with column-wise input data encoding and aggregation of several reservoir states for the training process, with $n = 1{,}024$. In particular, training with n corresponds to optimising the readouts weights on the basis of the sole reservoir state, produced after the processing of the 17th column of the image. Conversely, training with $4n$ consists of aggregating four reservoir states, obtained by processing the columns 14, 16, 20, and 24 of the image, for the training process.

V. Conclusion

In summary, we report two major results in this paper. First, we present a large-scale experimental photonic reservoir computer, based on off-the-shelf components, capable of implementing neural networks up to 16,384 nodes. Second, the computational power offered by our large dynamical system allows us to tackle an important task in computer vision: the recognition of handwritten digits. To this end, we study three conceptually different approaches of applying our reservoir computer to this task. First, we exploit the reservoir in a memoryless, feedforward mode of operation paired with popular handcrafted feature-extraction techniques (Zoning, Gabor filters, and HOG) prior to feeding the input layer of the photonic reservoir. The second mode of operation consisted in using the temporal dynamics of the network and hence benefit from its internal memory enabled by recurrent connections, still paired to feature extraction. Third, we used the temporal dynamics of our reservoir computer with a time-dependent signal, created by splitting the raw MNIST images into individual columns, thus creating temporal dependence and stimulating the recurrence of the system. We report

a 0.8% classification error in numerical simulations and 1.03% error in experiments, obtained with the feedforward mode of operation using the HOG feature extraction technique. We also demonstrated in numerical simulations the benefit of the recurrence mode of operation allowing for relative gain improvement ranging from 4.5% to 16.4% for the smallest reservoir in our study and the superiority of preserving spatial correlation and feature extraction compared to transforming the 2D MNIST images into multiple 1D signals. In this paper, we have shown that large-scale photonic reservoir computer can perform with level of performance comparable to the best digitally-implemented neuro-inspired algorithms currently available and hence represent a meaningful alternative to these methods, hence opening more in-depth research endeavours on exploiting photonic reservoir computer for advanced image processing.

In perspective, our work opens several directions for future research. The handcrafted features could be replaced by transfer learning, *i.e.* by automatically synthetized features from the first layers of a convolutional neural network trained on the MNIST dataset [54]. Furthermore, since the reservoir computer yields competitive results on handwritten digit recognition, one could increase the complexity of the task and apply the system to visual object recognition using e.g. the CIFAR-10 dataset [55]. The present work is thus the first step in what we hope should be a very fruitful line of investigations.

ACKNOWLEDGMENT

The authors would like to thank Dr. D. Brunner for the insightful scientific exchanges and discussions related to spatiotemporal photonic reservoir computers.

REFERENCES

[1] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York, NY, USA: Wiley, 2000.

[2] R. Cipolla, S. Battiato, and G. M. Farinella, Eds. *Machine Learning for Computer Vision (Studies in Computational Intelligence)*, 2nd ed. New York, NY, USA: Springer, 2013.

[3] C. Liu and H. Wechsler, "Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition," *IEEE Trans. Image Process.*, vol. 11, no. 4, pp. 467–476, Apr. 2002.

[4] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 12, pp. 2037–2041, Dec. 2006.

[5] P. F. Felzenszwalb, R. B. Girshick, D. Mcallester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.

[6] A. Miller, B. Blott, and T. Hames, "Review of neural network applications in medical imaging and signal processing," *Med. Biol. Eng. Comput.*, vol. 30, pp. 449–464, 1992.

[7] Y. LeCun *et al.*"Handwritten zip code recognition with multilayer networks," in *Proc. 10th Int. Conf. Pattern Recognit.*, 1990, pp. 35–40.

[8] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[9] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[10] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[11] H. Jaeger, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *Science*, vol. 304, no. 5667, pp. 78–80, Apr. 2004.

[12] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Comput.*, vol. 14, no. 11, pp. 2531–2560, Nov. 2002.

[13] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Comput. Sci. Rev.*, vol. 3, no. 3, pp. 127–149, Aug. 2009.

[14] L. Appeltant *et al.*, "Information processing using a single dynamical node as complex system," *Nature Commun.*, vol. 2, no. 1, Sep. 2011, Art. no. 468.

[15] N. Haynes, M. Soriano, I. Fischer, and D. Gauthier, "Reservoir computing with a single time-delay autonomous boolean node," *Phys. Rev. E*, vol. 91, 2014, Art. no. 020801.

[16] J. Torrejon *et al.*, "Neuromorphic computing with nanoscale spintronic oscillators," *Nature*, vol. 547, pp. 428–431, 2017.

[17] D. Marković *et al.*, "Reservoir computing with the frequency, phase, and amplitude of spin-torque nano-oscillators," *Appl. Phys. Lett.*, vol. 114, 2019, Art. no. 012409.

[18] L. Larger *et al.*, "Photonic information processing beyond turing: An optoelectronic implementation of reservoir computing," *Opt. Express*, vol. 20, no. 3, pp. 3241–3249, Jan. 2012.

[19] D. Brunner, M. C. Soriano, C. R. Mirasso, and I. Fischer, "Parallel photonic information processing at gigabyte per second data rates using transient states," *Nature Commun.*, vol. 4, no. 1, Jan. 2013, Art. no. 1364.

[20] L. Larger *et al.*, "High-speed photonic reservoir computing using a time-delay-based architecture: Million words per second classification," *Phys. Rev. X*, vol. 7, no. 1, Feb. 2017, Art. no. 011015.

[21] Y. Paquot *et al.*, "Optoelectronic reservoir computing," *Scientific Rep.*, vol. 2, no. 1, Feb. 2012, Art. no. 287.

[22] K. Takano *et al.*, "Compact reservoir computing with a photonic integrated circuit," *Opt. Express*, vol. 26, pp. 29 424–29 439, 2018.

[23] F. Duport, B. Schneider, A. Smerieri, M. Haelterman, and S. Massar, "All-optical reservoir computing," *Opt. Express*, vol. 20, no. 20, pp. 22783–22795, sep 2012.

[24] A. Argyris, J. Bueno, and I. Fischer, "Photonic machine learning implementation for signal recovery in optical communications," *Scientific Rep.*, vol. 8, 2018, Art. no. 8487.

[25] G. Van der Sande, D. Brunner, and M. C. Soriano, "Advances in photonic reservoir computing," *Nanophotonics*, vol. 6, pp. 561–576, 2017.

[26] K. Vandoorne *et al.*, "Experimental demonstration of reservoir computing on a silicon photonics chip," *Nature Commun.*, vol. 5, no. 1, Mar. 2014, Art. no. 3541.

[27] A. Katumba *et al.*, "Low-loss photonic reservoir computing with multimode photonic integrated circuits," *Scientific Rep.*, vol. 8, 2018, Art. no. 2653.

[28] A. Katumba, X. Yin, J. Dambre, and P. Bienstman, "A neuromorphic silicon photonics nonlinear equalizer for optical communications with intensity modulation and direct detection," *J. Lightw. Technol.*, vol. 37, no. 10, pp. 2232–2239, 2019.

[29] J. Bueno *et al.*, "Reinforcement learning in a large-scale photonic recurrent neural network," *Optica*, vol. 5, no. 6, pp. 756–760, Jun. 2018.

[30] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks – with an Erratum note," German National Research Center for Information Technology GMD, Bonn, Germany, Tech. Rep. 148, 2001.

[31] A. N. Tikhonov, A. Goncharsky, V. Stepanov, and A. G. Yagola, *Numerical methods for the solution of ill-posed problems*.New York, NY, USA: Springer, vol. 328, 1995.

[32] P. Antonik *et al.*, "Online training of an opto-electronic reservoir computer applied to real-time channel equalization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 11, pp. 2686–2698, Nov. 2017.

[33] A. M. Hagerstrom *et al.*, "Experimental observation of chimeras in coupled-map lattices," *Nature Phys.*, vol. 8, no. 9, pp. 658–661, 2012.

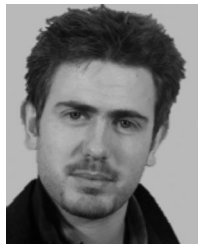
[34] D. Psaltis and N. Farhat, "Optical information processing based on an associative-memory model of neural nets with thresholding and feedback," *Opt. Lett.*, vol. 10, no. 2, pp. 98–100, Feb. 1985.

[35] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[36] N. Schaetti, M. Salomon, and R. Couturier, "Echo State Networks-Based Reservoir Computing for MNIST Handwritten Digits Recognition," in *Proc. 19th IEEE Int. Conf. Comput. Sci. Eng., 14th IEEE Int. Conf. Embedded Ubiquitous Comput. 15th Int. Symp. Distrib. Comput. Appl. Bus., Eng. Sci.*, 2016, pp. 484–491.

[37] H. E. Bahi, Z. Mahani, A. Zatni, and S. Saoud, "A robust system for printed and handwritten character recognition of images obtained by camera phone," *WSEAS Trans. Signal Process.*, vol. 11. pp. 9–22, 2015. [Online]. Available: http://www.wseas.org/multimedia/journals/signal/2015/a045714-403.pdf
[38] A. B. S. Hussain, G. T. Toussaint, and R. W. Donaldson, "Results obtained using a simple character recognition procedure on munson's handprinted data," *IEEE Trans. Comput.*, vol. C-21, no. 2, pp. 201–205, Feb. 1972.
[39] A. K. Jain and F. Farrokhnia, "Unsupervised texture segmentation using Gabor filters," *Pattern Recognit.*, vol. 24, no. 12, pp. 1167–1186, Jan. 1991.
[40] A. Bovik, M. Clark, and W. Geisler, "Multichannel texture analysis using localized spatial filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 1, pp. 55–73, 1990.
[41] M. R. Turner, "Texture discrimination by Gabor functions," *Biol. Cybern.*, vol. 55, no. 2–3, pp. 71–82, 1986.
[42] A. Perry and D. Lowe, "Segmentation of textured images," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit.*, 1989, pp. 319–325.
[43] T. Tan and A. Constantinides, "Texture analysis based on a human visual model," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 1990, pp. 2137–2140.
[44] J. Malik and P. Perona, "Preattentive texture discrimination with early vision mechanisms," *J. Opt. Soc. Amer. A*, vol. 7, no. 5, pp. 923–932, May 1990.
[45] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit.*, 2005, pp. 886–893.
[46] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.
[47] K. S. Siddharth, M. Jangid, R. Dhir, and R. Rani, "Handwritten Gurmukhi character recognition using statistical and background directional distribution," *Int. J. Comput. Sci. Eng.*, vol. 3, no. 06, pp. 2332–2345, 2011.
[48] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proc. 30th Int. Conf. Mach. Learn.*, (Proceedings of Machine Learning Research), S. Dasgupta and D. McAllester, Eds., vol. 28, no. 3, Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1058–1066.
[49] D. Wang and X. Tan, "Unsupervised feature learning with c-SVDDNet," *Pattern Recognit.*, vol. 60, pp. 473–485, Dec. 2016.
[50] D. Keysers, T. Deselaers, C. Gollan, and H. Ney, "Deformation models for image recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 8, pp. 1422–1435, Aug. 2007.
[51] D. Decoste and B. Schölkopf, "Training invariant support vector machines," *Mach. Learn.*, vol. 46, no. 1/3, pp. 161–190, 2002.
[52] B. Kégl and R. Busa-Fekete, "Boosting products of base classifiers," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 497–504.
[53] R. Salakhutdinov and G. Hinton, "Deep Boltzmann machines," in *Proc. 12th Int. Conf. Artif. Intell. Statist.*, 2009, pp. 448–455.
[54] E. S. Olivas, J. D. M. Guerrero, M. M. Sober, J. R. M. Benedito, and A. J. S. Lopez, *Handbook Of Research On Machine Learning Applications and Trends: Algorithms, Methods and Techniques - 2 Volumes*, Hershey, PA: IGI Publishing, 2009.
[55] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Princeton, New Jersey, USA, Tech. Rep., 2009.

Piotr Antonik was born in Minsk, Belarus, in 1989. He received the Master's degree in physics from Université libre de Bruxelles, Brussels, Belgium, in 2013 and the Ph.D. degree, in 2017, under the direction of Prof. S. Massar. During his Ph.D., he studied implementations of machine learning methods in photonic systems. From 2017 to 2018, he was a Post-Doctoral Researcher with the LMOPS EA-4423 and the Chair in Photonics with CentraleSupélec, Metz, France. In October 2018, he became an Associate Professor with CentraleSupélec, Metz Campus, with the LMOPS EA-4423 and the Chair in Photonics. He continues his research activities in machine learning and photonics, with applications in e.g., biomedical imaging and telecommunications. To this day, He authored/coauthored six articles in peer-reviewed journals and presented 16 contributions (talks and posters) to international conferences. His Ph.D. thesis won the Springer Theses Award and was published in the Springer Theses collection.



Nicolas Marsal was born in Woippy, France. He received the Master of Physics degree from the Université de Lorraine, Metz, France, in 2007 and the Ph.D. degree in physics from CentraleSupélec, Metz, France, in 2010. From 2011 to 2012, he was a Postdoctoral Fellow with the Nonlinear Optics Department, Georgia Tech Lorraine, Metz, France. Since December 2012, he has been an Assistant Professor with CentraleSupélec, Metz, France. His research activities are mainly focused on spatio-temporal dynamics of nonlinear photonic systems, propagation and interaction of nonconventional optical beams and machine learning at the physical layer for innovative applications in neuro-inspired information processing and cognitive computing. His research is conducted within the framework of the LMOPS EA-4423 Laboratory, a joint unit between CentraleSupélec and the Université de Lorraine, Metz, France. He has authored and coauthored 23 publications in international peer-reviewed journals and presented more than 50 contributions in international conferences.

Damien Rontani was born in Clamart, France. He received the Master of Science degree in electrical engineering (Supélec Curriculum) and the master's degree in information science, energy and systems from CentraleSupélec, Université Paris-Saclay, Metz, France, in 2004 and 2006, respectively, the Master of Science degree in electrical and computer engineering from the Georgia Institute of Technology (Georgia Tech), Atlanta, GA, USA, in 2005, and in 2011, he received the Ph.D. degree in photonics from CentraleSupélec and the Ph.D. degree in electrical and computer engineering from Georgia Tech. From 2011 to 2013, he was a Postdoctoral Fellow with the Physics Department, Duke University, Durham, NC, USA. Since December 2013, he has been an Assistant Professor with CentraleSupélec, Metz, France, where he conducts leading-edge research in nonlinear dynamics of photonics systems, nonlinear optics, and machine learning at the physical layer for innovative applications in neuro-inspired information processing and cognitive computing. His research is conducted within the framework of the LMOPS EA-4423 Laboratory, a joint unit between CentraleSupélec and the Université de Lorraine, Metz, France. He has authored and coauthored 24 publications in international peer-reviewed journals and more than 50 contributions in international conferences. In his young academic career, he has been the recipient of an IBM Faculty Award in Cognitive Computing in 2015 and a JSPS Fellowship for Overseas Researchers from the Japanese Society for the Promotion of Science in 2016.