Training a multilayer dynamical spintronic network with standard machine-learning tools to perform time-series classification

Erwan Plouet, Dédalo Sanz-Hernández, Aymeric Vecchiola, Julie Grollier, and Frank Mizrahi*

Laboratoire Albert Fert, CNRS, Thales, Université Paris-Saclay, 91767 Palaiseau. France

(Received 7 August 2024; revised 17 December 2024; accepted 4 March 2025; published 21 March 2025)

This paper is a contribution to the Physical Review Applied collection titled Physics-Inspired Computing.

The ability to process time series at low energy cost is critical for many applications. Recurrent neural networks, which can perform such tasks, are computationally expensive when implemented in software on conventional computers. Here we propose to implement a recurrent neural network in hardware using spintronic oscillators as dynamical neurons. Using numerical simulations, we build a multilayer network and demonstrate that we can use back-propagation through time and standard machine-learning tools to train this network. Leveraging the transient dynamics of the spintronic oscillators, we solve the sequential digits classification task with 89.83% ± 2.91% accuracy, as good as the equivalent software network. We devise guidelines on how to choose the time constant of the oscillators as well as hyperparameters of the network to adapt to different input time scales.

DOI: 10.1103/PhysRevApplied.23.034051

I. INTRODUCTION

The ability to process time series (classification, prediction, generation, etc.) is important for many applications, from smart sensors in industrial maintenance to personal assistants and medical devices. Using the dynamics of a physical system, leveraging its nonlinearity and memory for such processing has been widely explored with the development of recurrent neural networks, both from a purely mathematical perspective [1–4] as well as from a brain-inspired perspective with spiking recurrent neural networks [5–8]. Chen *et al.* [9] have shown that neural networks based on ordinary differential equations (neural-ODEs) can be seen as residual neural networks where the time dimension acts as depth, thus providing computing power. These findings highlight the potential of dynamical systems to implement deep neural networks. While implementing a recurrent neural network in software on a conventional computer requires computing the evolution of each neuron step by step, a physical network would naturally perform this computation. As there is an increasing demand for learning and processing on the edge, with strong footprint and energy cost constraints, building novel hardware that directly implements dynamic recurrent neural networks is an attractive path. This motivation has led to the realization and training of recurrent neural networks with a wide variety of dynamic systems: analog CMOS [10], photonic systems [11], acoustic resonators [12], mechanical oscillators [13], and wave systems [14] to cite a few.

Spintronic oscillators are promising building blocks for the hardware implementation of neural networks, due to their nonlinear high-speed dynamics as well as potential for miniaturization and CMOS integration [15,16]. Time-series processing using the dynamics of spintronic oscillators has been repeatedly demonstrated experimentally in the context of reservoir computing, thus validating that they have the required nonlinearity and memory properties [17–23]. However, reservoir computing is limited because only the output classifier is trained and the dynamics is fixed. Training of spintronic oscillators for time-series processing has been limited to single-layer networks [24,25]. Ross *et al.* [26] have experimentally demonstrated a multilayer network of spintronic oscillators, but with a feedforward architecture dedicated to static tasks. Rodrigues *et al.* [27] have shown by numerical simulations how to train the transient dynamics of a single-layer network of oscillators with optimal control theory, on a static task.

Here we simulate and train a multilayer network of spintronic oscillators as neurons, using standard machine-learning tools. We leverage the transient dynamics of the oscillators to perform time-series classification of the sequential digits dataset. First, we describe the dynamic spintronic neuron model, the architecture of the network, and how to cascade neuron layers. Second, we describe how to train the network with PyTorch and back-propagation through time (BPTT) and demonstrate 89.83% ± 2.91% accuracy on sequential digits, as good as the equivalent software network. Third, we show that the spintronic network can be trained on input time scales over a fivefold range, centered around a value depending on device parameters. Finally, we derive guidelines to find

*Contact author: frank.mizrahi@cnrs-thales.fr

these device parameters. In particular, the neuron relaxation time must be larger than the input time scale and the cumulative drive must be around one.

II. MULTILAYER NETWORK OF SPINTRONIC DYNAMICAL NEURONS

Spintronic oscillators exhibit magnetization self-sustained oscillations when a dc current drive overcomes the magnetic damping. Magnetoresistive effects translate the magnetization oscillations into voltage oscillations that can be fed as input to other devices. In this work, we use the amplitude of these oscillations as the internal variable of the spintronic neuron. When submitted to a drive, the oscillation amplitude undergoes a transient regime, which provides memory to the device. We use the auto-oscillator model [28] to describe the dynamics of the neurons. We choose this model because it has been repeatedly verified by experiments, in particular in the context of dynamical neurons [27]. Using this model rather than directly solving the Landau-Lifshitz-Gilbert equation or performing micromagnetic simulations has two advantages. First, it reduces simulation length, making it possible to train multilayer neural networks with hyperparameter search, which is critical for physical neural networks, as will be detailed in Secs. III and IV. Second, it provides a general model, which can be applied widely to different types of spintronic oscillators.

Slavin's model [28] describes the dynamics of the magnetization dimensionless normalized amplitude x:

$$\frac{dx}{dt} = -2(\gamma(1+Qx) - \sigma I(t)(1-x))x. \tag{1}$$

Here $\gamma = \alpha \times \omega$ is a damping term, with α the magnetic Gilbert damping and ω the frequency of the device; and Q is the nonlinear damping, which can take various values depending on the exact materials, drives, and devices used for the neurons. We set $Q = 0$ in order to show the intrinsic nonlinearity of a spintronic oscillator sufficient for time-series classification. A nonzero Q provides additional nonlinearity and computing power. In Eq. (1), σ is a material parameter which we set to one, as it simply rescales $I(t)$.

In this work, we simplify the magnetization dynamics to

$$\frac{dx}{dt} = -\gamma x + I(t)x(1-x) \tag{2}$$

where $I(t)$ is the time-dependent drive, a dimensionless variable proportional to the dc current injected into the device. The damping γ pulls the output to zero and $x(x-1)$ is the term that acts as a natural bounding of the output between 0 and 1. The oscillator exhibits nonlinearity as well as memory, making it suitable for processing time series. Furthermore, it follows the shape $x(t+1) = g(x(t), \mathcal{I}(t))$ (where $\mathcal{I}(t)$ is the input at time t), which is the conventional shape of a recurrent neuron. Here, the hidden state variable of the neuron x is the oscillation power.

We consider a network, depicted in Fig. 1, composed of successive layers of dynamical neurons, represented by blue circles. The neurons are connected by interlayer connections (W_{ext}, in purple) and intralayer connections (W_{int}, in green). In the scope of this paper, we consider standard linear connections, performing weighted sums of the rf power outputs of the neurons.

In order to exploit the whole dynamical range of the neurons, their output should be greater than zero (no output) and less than one (saturation). This requires the outputs of the neurons in the absence of external drive to be half of their maximum output power. In principle, the network could achieve this by learning the correct values for the trainable bias applied to each neuron. However, it is more efficient to apply a fixed bias b_{fixed} on top of the trainable biases b_i^n to each neuron. The value of the fixed bias is twice the damping $b_{\text{fixed}} = 2\gamma$. Furthermore, we use a high-pass filter of cut frequency $f_{\text{cut}} = 1$ MHz as well as a tunable amplification factor set to $S = 0.5$ between each layer to keep the drives centered around zero.

The dynamics of the power x_i^{n+1} of neuron i from layer $n+1$ is given by

$$\frac{dx_i^{n+1}}{dt} = -\gamma x_i^{n+1} + I(t)x_i^{n+1}(1-x_i^{n+1}) \tag{3}$$

with the drive

$$I(t) = S(W_{i,j}^{n,\text{ext}} y_j^n + W_{i,j}^{n+1,\text{int}} y_j^n + b_i^{n+1}) + b_{\text{fixed}}, \tag{4}$$

where y_j^n is the output of neuron i from layer n after application of the high-pass filter and follows

$$\frac{dy_i^n}{dt} = -2\pi f_{\text{cut}} y_i^n + \frac{dx_i^n}{dt}. \tag{5}$$

Figures 1(b)–1(d) illustrate the dynamics of the neurons for layers 0, 1 and 2, respectively. Each color is one time-varying input applied to the neuron. We observe varied dynamics around 0.5 (i.e., half of the maximum amplitude of the oscillations). Each layer acts an integrator, slowing down information as it flows into the network.

III. PROCESSING OF A TIME-SERIES TASK

The time-series classification task we chose to evaluate the spintronic network is sequential digits [29]. The dataset is composed of 1797 grayscale 8-pixel-by-8-pixel images of handwritten digits, and the goal is to identify the digit (labels 0 to 9). We split the dataset 50:50 between train and test. The images are presented pixel by pixel to the network, as time series of 64 input points each, as depicted in Fig. 2(a). Here the time interval between input points is

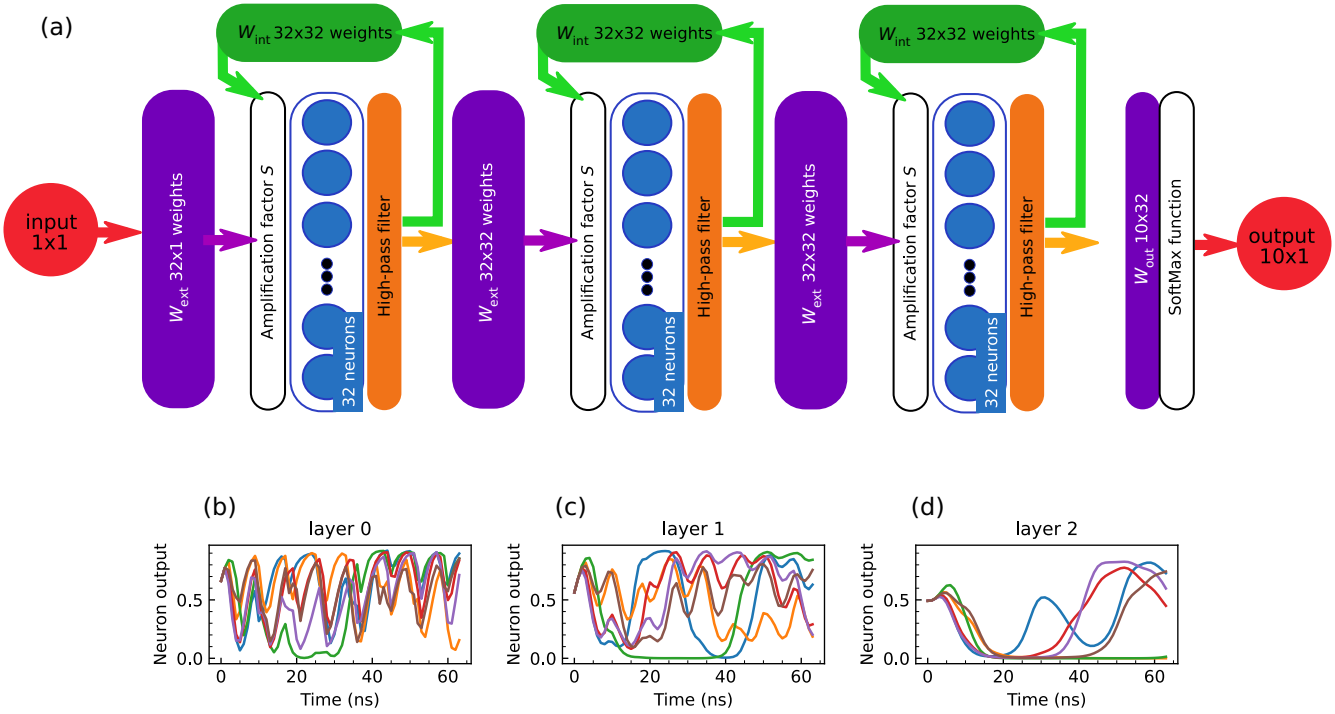

FIG. 1. Architecture of the network. (a) Schematic of the network architecture, with three layers of 32 neurons each. The neurons are represented by blue circles, the interlayer connections (W_{ext}) by purple boxes, the intralayer connections (W_{int}) by green boxes, the amplification factors and SoftMax by white boxes, and the high-pass filters by orange boxes. The dimensions of the connections are indicated in the corresponding boxes. At each time step, the input is of size one. The output is of size 10. For simplicity, we do not represent the fixed and trainable biases on this figure. (b)–(d) Normalized output rf power versus time, emitted respectively by the neuron 0 from layer 0, the neuron 0 from layer 1, and the neuron 1 from layer 2. The colors (blue, orange, green, red, purple, and brown) correspond to different inputs from different classes (respectively 5, 6, 9, 3, 3, and 8).

1 ns. In consequence, the input of the network is of size 1 and the output of size 10.

We chose this task because the digits dataset is well known and freely available, which will simplify future replication studies and benchmarking. We present the pixels sequentially in order to evaluate the ability of the network to process time series, which makes the task harder than the standard static image classification task. The network is simulated using the PyTorch library as well as the differential equation (2).

After the final time step of the series, we compute the log-likelihood loss of the output. We reset the internal state of the neurons after processing each image, as there is no time correlation between successive images. We update the trainable parameters (weights and biases) using BPTT and batches of 120 images [30–32]. We clip the amplitude of each gradient element to 1 to prevent gradient explosion. We set the connectivity density at 0.5 for both the interlayer and intralayer connections ($d_{\text{inter}} = 0.5$ and $d_{\text{intra}} = 0.5$). The effect of connection density is studied in Sec. IV.

Figure 1(d) shows the behavior of neuron 1 of the output neuron. We observe different evolutions for the different classes, i.e., colors, illustrating the separation property of the network, as well as similar evolutions for the purple and red curves, which correspond to the same class. We use a hyperparameter optimization using the library Optuna [33], in order to maximize the accuracy, and obtain a relaxation time of $\tau = 14.12$ ns for the neurons. We use a learning-rate decay method $\mathcal{R}_l = \mathcal{R}_{l,0}/(n_{\text{epoch}}/\mathcal{R}_{l,\text{decay}} + 1)$ with $\mathcal{R}_{l,0} = 0.149$ and $\mathcal{R}_{l,\text{decay}} = 8.077$, as well as the Adam optimizer [34]. We achieve an accuracy of $89.83\% \pm 2.91\%$. The relaxation time corresponds to $\tau = 1/\gamma = 1/\alpha\omega$. Typical magnetic damping values in spintronic oscillators are around $\alpha = 0.01$. In consequence, the optimal frequency for the oscillators would be around $\omega \approx 7.1$ GHz, which is achieved with current devices [35].

We benchmark this result by comparing the simulated spintronic network to a standard software-based standard continuous-time recurrent neural network (CTRNN) [36]. We consider a CTRNN with the same architecture (three layers of 32 neurons, inter- and intralayer connections with density of 0.5 each).

The hidden state x_i^{n+1} of the ith neuron of layer $n+1$ is driven by

$$\frac{dx_i^{n+1}}{dt} = -\gamma x_i^{n+1} + I(t) \tag{6}$$

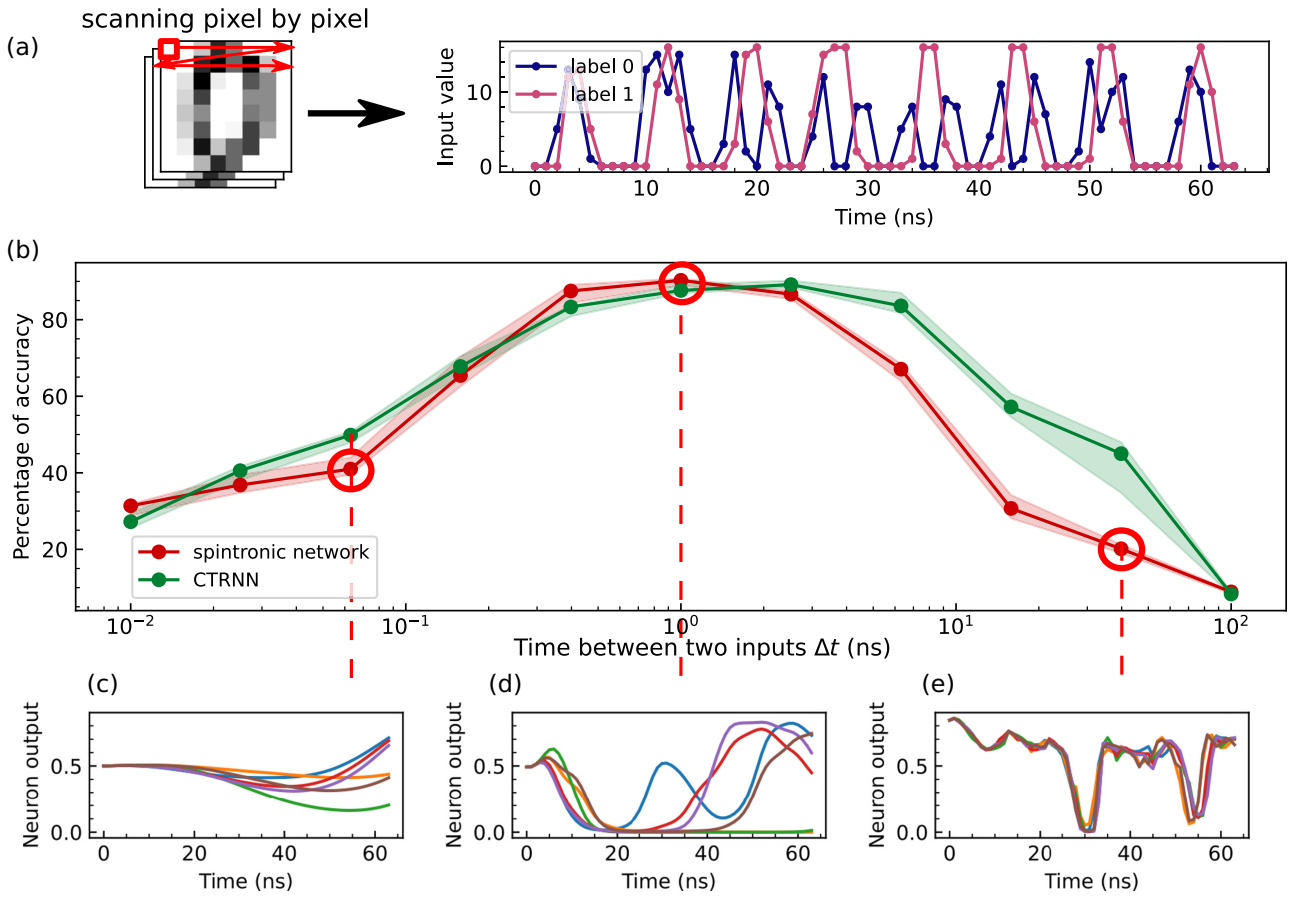

FIG. 2. Performance on the sequential digits classification task. (a) Schematic of the task. The images are scanned row by row and the pixel intensity in used as input to the network. The violet and pink curves represent the input values versus time for an image of label 0 and label 1, respectively. (b) Accuracy of the network versus the time interval between two input points. The red curve corresponds to the simulated spintronic network, while the green curve corresponds to a standard continuous-time recurrent neural network. Each accuracy point corresponds to the median accuracy on a series of 10 random initializations; the shaded areas correspond to the range between the 75th and 25th percentiles. (c)–(e) Normalized output versus time for, respectively, neuron 3, neuron 1, and neuron 2 of layer 2. The colors (blue, orange, green, red, purple, and brown) correspond to different inputs from different classes (respectively 5, 6, 9, 3, 3, and 8). Each of the three panels corresponds to a different time interval between input points, shown by red circles and red dashed lines in panel (b). The neuron dynamics is respectively too slow, adapted, and too fast for the input time scale.

with the drive

$$I(t) = S(W_{i,j}^{n,\mathrm{ext}} y_j^n + W_{i,j}^{n+1,\mathrm{int}} y_j^n + b_i^{n+1}), \tag{7}$$

where y_j^n is the output of the jth neuron of layer n with application of a nonlinear activation function

$$y_j^n = \tanh(x_j^n). \tag{8}$$

We see here that the spintronic network is a natural implementation of a CTRNN by a physical system. The main difference is that the internal state of the spintronic neuron is intrinsically bounded, while it is not for the software neurons, which rely on their activation function to bound their outputs. This constraint is common to all physical implementations of neurons. We observe that the CTRNN achieves a top accuracy of $89.00\% \pm 3.48\%$. The uncertainty corresponds to the standard deviation over 10 runs, each with a different random initialization of the trainable parameters. We conclude that the spintronic network performs as well as a standard CTRNN, despite not being isomorphic to the CTRNN equations, and in particular having the strong constraint of having neurons with bound internal state.

IV. ADAPTATION OF THE NETWORK TO DIFFERENT TIME SCALES

When processing time series, a critical feature is the time scale of the input. Because the memory of the network comes from the dynamics of the physical system, the system needs to be designed to match the time scales relevant to the task. We derive design rules by studying how the network behaves at different input time scales.

We set the hyperparameters of the network to the values found by optimization for 1 ns between input points, then train the network on series with different input time scales. Figure 2(b) shows how the accuracy depends on the time interval between two input points. We observe that the maximum accuracy is achieved for 1 ns, which is expected, as this is the optimization point. This maximal accuracy is maintained over a wide window of time scales, about a fivefold factor, from 0.4 to 2.5 ns. When the time scale is well below or well above 1 ns, the accuracy is significantly degraded.

Here again, the spintronic network performs as well as the CTRNN. Figures 2(c)–(e) show the evolution of a neuron from layer 2 (i.e., output layer) for different inputs (colors) of different classes, in three regimes. In Fig. 2(d), the dynamics of the network is optimized for the input time scale. We observe that the different inputs produce rich varied dynamics, enabling separation of the inputs into classes. In Fig. 2(c), the input is faster and the neuron is too slow to follow its changes. We observe that the dynamics of the input is completely lost in the smoothed trajectories of the neuron, preventing accurate classification. In Fig. 2(e), the input is slower and the neuron is too fast. The memory of the network is too short to remember enough input points to achieve accurate classification.

Building on these results, we devise guidelines to design networks adapted to learn a task with a given time scale. The dynamics of the spintronic neuron relies on a competition between the current drive and the damping. The damping pulls the output amplitude to zero while the current drive pushes the neuron away from this rest position.

Figure 3(a) shows the accuracy versus the time between two input points, for three different relaxation times. We observe that the optimal time scale shifts with the relaxation time. The dependence of the accuracy is always a bell-shaped curve, centered around the optimal time. In particular, slower inputs require longer relaxation times so that the network has a long enough memory to remember the input points necessary for classification.

Figure 3(b) shows the accuracy versus the time between two input points, for three different learning rates. We observe that, here as well, learning requires a matching between learning rate and input time scale. Furthermore, we observe that faster inputs require higher learning rates. Indeed, the neuron receives a cumulative drive $I\,\Delta t$ for each input point. The strength of this drive depends on the weights and the scaling factors,

$$I = S\langle|W|\rangle\sqrt{N_{\text{neurons}}}, \tag{9}$$

where $N_{\text{neurons}} = 32d_{\text{inter}} + 32d_{\text{intra}} = 32$ is the fan-in for each neuron and $\langle|W|\rangle$ is the mean of the absolute value of the weight elements. When the input is fast (low Δt), the network needs to reach higher weights to provide a cumulative drive high enough to counterbalance the damping. Higher learning rates allow the weights to reach higher values.

We identify two conditions. First, the relaxation time of the neurons should be larger than the time between two input points $\Delta t < \tau$. Second, the cumulative drive divided by the relaxation time should be roughly equal to the damping:

$$I\,\frac{\Delta t}{\tau} \approx \gamma. \tag{10}$$

As the damping is the inverse of the relaxation time, this simplifies as

$$I\,\Delta t \approx 1. \tag{11}$$

Figure 3(c) shows the accuracy versus the cumulative drive for a broad range of conditions. We observe that the best accuracies are indeed obtained when the cumulative drive is around 1, provided that the input scale is larger than the relaxation time (green dots). Furthermore, we recommend to set the learning rate in the range of one-hundredth of the optimal weights values $1/(\Delta t\,S\sqrt{N_{\text{neurons}}})$.

Note that the cumulated drive must take device and material parameters into account. For instance, if σ is different from 1, the factor S should be rescaled. We observe that the design rule regarding the cumulative drive does not require fine tuning. In Fig. 2(b) as well as Fig. 3, we observe that the accuracy is not significantly degraded when the input time scale varies within a wide window, typically fivefold centered around the optimal value, which is promising for practical realizations. In this work, we have provided an example of how to perform hyperparameter tuning for a simplified set of device and material parameters. For designing and training a physical network, it will be necessary to perform hyperparameter tuning with the actual device and material parameters.

V. ARCHITECTURE DESIGN

We evaluate the impact of the connection density. Figure 4 shows the accuracy of the task versus the density of connections. Here the network is randomly sparsified before training. We observe that the intralayer connectivity (red curve) can be reduced to 10% without losing accuracy. When removing the intralayer connections altogether, the accuracy is degraded but remains high. The interlayer connectivity (green curve) can be reduced to 20% without losing accuracy (87.22%). We also observe that the performance exhibits higher run-to-run variability at higher densities of connection. This is due to the fact that higher connectivity increases the risk of neuron saturation and thus failure of learning. The global connectivity (both interlayer and intralayer) is optimal around 50% (90.28% accuracy) and the accuracy is still very high

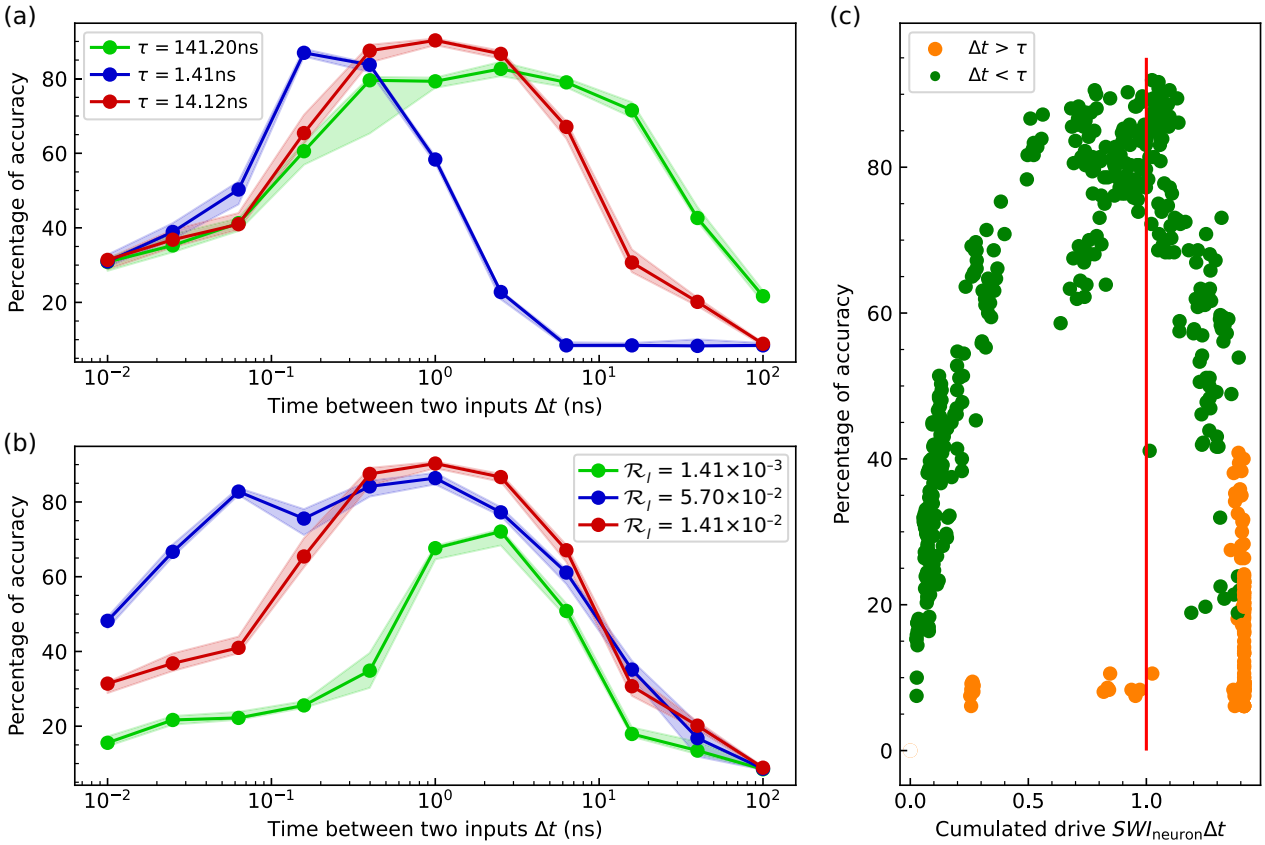

FIG. 3. Time-scale adaptation. (a) Accuracy versus the time between two input points Δt for three neuron relaxation times τ (colors). (b) Accuracy versus the time between two input points Δt for three learning rates (colors). For panels (a) and (b), each point corresponds to the median accuracy on a series of 10 random initializations; the shaded areas correspond to the range between the 75th and 25th percentiles. (c) Accuracy versus the cumulative drive, for a wide range of parameters. Each point corresponds to the median accuracy on a series of 10 random initializations. The points in green (orange) have a neuron relaxation time τ larger (smaller) than the input time scale Δt. The vertical red line marks where the cumulative drive is equal to 1.

down to 30% density (85.42%). At 20% global connectivity, the accuracy is still high (77.78%) but has dropped significantly. Removing connections helps to prevent overfitting, as shown in the Appendix. Removing the interlayer connections altogether prevents any learning, as expected, as the information cannot flow from input to output.

To build such a network, we need physical implementations of the synaptic layer. These synapses need to respond fast enough to preserve the neuron dynamics. Achieving the large number of connections required for implementing state-of-the-art artificial neural networks in hardware is challenging. However, there has been significant recent progress regarding spintronic connections. In particular, several teams have demonstrated crossbars of magnetic tunnel junctions monolithically co-integrated with standard CMOS [37,38]. Alternatively, Ross *et al.* [26] have proposed to use spintronic resonators to convert the oscillator outputs into DC signals, which provides a simpler architecture than direct coupling of oscillators.

Several technologies have been proposed to implement oscillator neurons. For instance, a CMOS oscillator at 8 GHz can be implemented with a circuit about 10-μm wide and consuming 200 μW [39]. Resistive switching oscillators have a smaller footprint (down to a few nanometers) but their frequencies are subgigahertz and their endurance is limited compared to spintronic devices [40]. Photonic oscillators can reach hundred gigahertz frequencies but require micron-sized devices. Spintronic oscillators combine small size (down to a few nanometers) [41], high frequencies (above 10 GHz), low power consumption (below 10 μW), and high endurance (more than 10^{12} cycles).

In the network described here, there are 96 neurons and 2896 synapses. If the image is presented for about 100 ns, according to the estimations in Ref. [26], each neuron and each synapse would consume 100 and 10 fJ, respectively, leading to a total consumption of about 40 pJ. These estimations take into account the tunable amplifiers between layers.

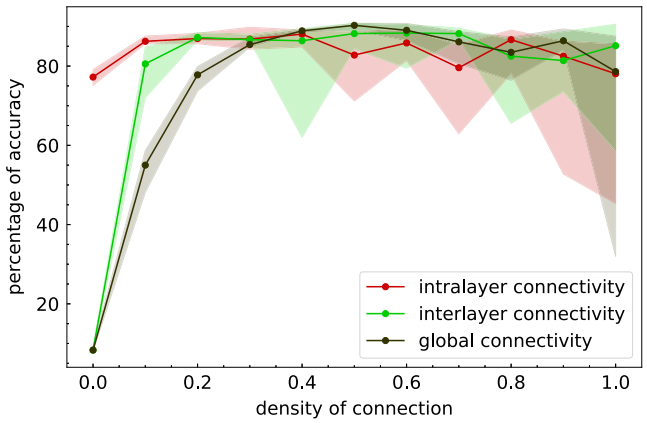

FIG. 4. Effect of the density of connections. The red curve shows the accuracy versus the density of the intralayer connections (W_{int}), for an interlayer density of 1. The green curve shows the accuracy versus the density of the interlayer connections (W_{ext}), for an intralayer density of 1. The black curve shows the accuracy versus the density of all connections. Each point corresponds to the median accuracy on a series of 10 random initializations; the shaded areas correspond to the range between the 75th and 25th percentiles.

In this work, we have shown the importance of hyperparameter tuning. We propose the following methodology for designing and training future hardware implementations. A first hyperparameter search is performed with software simulations for various tasks of the targeted applications, with detailed models of the specific spintronic devices to be used as neurons and synapses and taking technology constraints into account. From this search, values for the oscillators' frequency, fixed bias, and amplifier factor are found, in order to design and build the hardware network. The oscillators' frequency is set by material and geometry parameters, but can be tuned postfabrication over wide ranges, as shown in Ref. [35] for instance. Tunable biases, filters, and amplifiers can be made with standard analog CMOS circuits. While these are bulkier than nanodevices, only one per layer is required.

Once fabricated, the hardware network can be trained using either off-chip training or chip-in-the-loop training. Chip-in-the-loop training is a method where the forward pass is computed by the physical network, and the backward pass is computed with a model on a computer, using the loss from the physical network. This method has the advantage of been robust to mismatch between the model and the physical network up to about 20% [42]. However, it requires one to train separately each instance of the physical network. Furthermore, the hyperparameter tuning will be cumbersome to implement. Off-chip training makes it possible to train the network model on a computer once and then upload the trained weights on each instance of the physical network. This method enables a thorough hyperparameter tuning, as well as a single training for all nominally identical networks. Recent advances, using measurement-driven methods [37,43] as well as loss sharpness analysis [44] have made off-chip training robust enough to model mismatch to train physical networks.

It is likely that a combination of chip-in-the-loop and off-chip training will be optimal. This methodology supposes that the hardware network is only used for inference, which is satisfactory for a wide range of applications. Online training leveraging the physics of the network would make it possible for the network to keep learning new tasks during its use. However, both the hyperparameter tuning and the sensitivity to errors make this goal challenging. More research is required to develop novel learning algorithms that are less sensitive than backpropagation to small variations of the parameters.

VI. CONCLUSION

We have simulated a multilayer neural network where the neurons are spintronic oscillators. Leveraging the transient dynamics of the neurons for memory and nonlinearity, we have performed a time-series classification task, sequential digits. We demonstrate that this multilayer dynamical spintronic network can be trained with BPTT and standard machine-learning tools, which is key for scaling to more complex tasks. Despite having a bounded internal variable, the spintronic network achieves 89.83% accuracy, as good as a standard recurrent neural network. These results demonstrate the potential of spintronic oscillators for time-series processing.

We have devised guidelines for the design of such networks and their training: the relaxation time must be longer than the time scale of the input, and the cumulative drive must be close to one. Once built, a dynamical spintronic network can be trained to learn tasks over a fivefold range of input time scales. We have shown that we can sparsify the network down to 50% connectivity before training, further reducing the footprint and energy consumption of the system. For the sequential digits tasks and the considered architecture, we estimate that the network would consume about 40 pJ per image.

Despite the large interest for time-series processing with physical oscillators—whether spintronic, photonic, resistive, or CMOS-based—there is little work on systematic scalable training methods for oscillators. Here, we show how to use standard machine-learning tools to train multilayer spintronic oscillators, a method that is scalable. We also point out important design rules regarding the matching of the network dynamics with the input characteristics. Our use of a simplified model that is general enough makes our work easily adaptable to other types of physical oscillators. In this work we have demonstrated the concept with a simple task. Future studies will tackle larger networks and state-of-the-art machine-learning tasks. These results open

the path to the scaling up of spintronic neural networks for time-series processing at low energy cost.

ACKNOWLEDGMENTS

This work was supported by the European Union's Horizon 2020 Research and Innovation Programme under Grant RadioSpin No. 101017098.

APPENDIX

We study the impact of the connection density on the overfitting. Figure 5 shows the evolution of the accuracy and loss versus the number of epochs, for the train and test datasets at different global connection densities. The conditions are the same as described in Sec. III. We observe that the gap between the train and test datasets decreases with the connection density. This shows that reducing the density helps to prevent overfitting and explains why the optimal density is 50% rather than 100%.

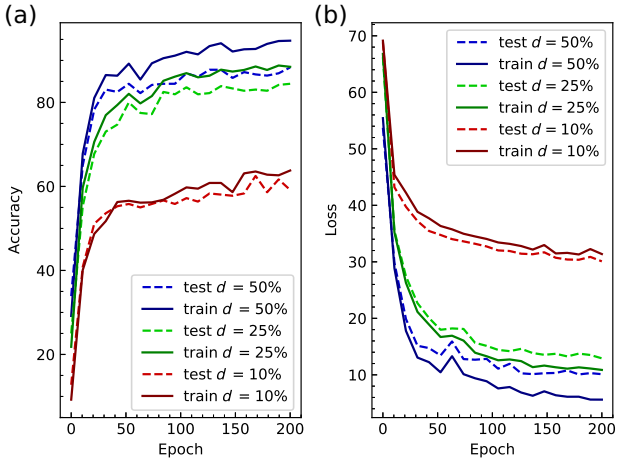

FIG. 5. Evolution of (a) the accuracy and (b) the loss versus the number of epochs (in percentage of correctly classified inputs). The train dataset is represented with solid lines and the test dataset with dashed lines. The colors blue, green, and red correspond to global connection densities of 50%, 25%, and 10%, respectively.

[1] F. Pineda, in *Neural Information Processing Systems* (American Institute of Physics, 1987).
[2] Y. Lecun, in *Artificial Neural Networks: Concepts and Theory* (Morgan Kaufmann, Pittsburgh, 1992).
[3] M. Hermans and B. Schrauwen, in *Advances in Neural Information Processing Systems 26* (Curran Associates, 2013).
[4] R. Hasani, M. Lechner, A. Amini, D. Rus, and R. Grosu, in *Proceedings of the AAAI Conference on Artificial Intelligence* (AAAI (Association for the Advancement of Artificial Intelligence) Press, Palo Alto, California, USA, 2021), Vol. 35, pp. 7657–7666.
[5] J. K. Eshraghian, M. Ward, E. O. Neftci, X. Wang, G. Lenz, G. Dwivedi, M. Bennamoun, D. S. Jeong, and W. D. Lu, Training spiking neural networks using lessons from deep learning, Proc. IEEE **111** (2023).
[6] A. Taherkhani, A. Belatreche, Y. Li, G. Cosma, L. P. Maguire, and T. M. McGinnity, A review of learning in biologically plausible spiking neural networks, Neural Netw. **122**, 253 (2020).
[7] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, Deep learning in spiking neural networks, Neural Netw. **111**, 47 (2019).
[8] B. Yin, F. Corradi, and S. M. Bohté, Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks, Nat. Mach. Intell. **3**, 905 (2021).
[9] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, in *Advances in Neural Information Processing Systems 31* (Curran Associates, 2018).
[10] C. Pehle, S. Billaudelle, B. Cramer, J. Kaiser, K. Schreiber, Y. Stradmann, J. Weis, A. Leibfried, E. Müller, and J. Schemmel, The BrainScales-2 accelerated neuromorphic system with hybrid plasticity, Front. Neurosci. **16**, 795876 (2022).
[11] J. Bueno, S. Maktoobi, L. Froehly, I. Fischer, M. Jacquot, L. Larger, and D. Brunner, Reinforcement learning in a large-scale photonic recurrent neural network, Optica **5**, 756 (2018).
[12] Y. Qu, M. Zhou, E. Khoram, N. Yu, and Z. Yu, Resonance for analog recurrent neural network, ACS Photonics **9**, 1647 (2022).
[13] J. C. Coulombe, M. C. York, and J. Sylvestre, Computing with networks of nonlinear mechanical oscillators, PLoS One **12**, e0178663 (2017).
[14] T. W. Hughes, I. A. Williamson, M. Minkov, and S. Fan, Wave physics as an analog recurrent neural network, Sci. Adv. **5**, eaay6946 (2019).
[15] G. Finocchio, M. Di Ventra, K. Y. Camsari, K. Everschor-Sitte, P. K. Amiri, and Z. Zeng, The promise of spintronics for unconventional computing, J. Magn. Magn. Mater. **521**, 167506 (2021).
[16] J. Grollier, D. Querlioz, K. Camsari, K. Everschor-Sitte, S. Fukami, and M. D. Stiles, Neuromorphic spintronics, Nat. Electron. **3**, 360 (2020).
[17] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, Recent advances in physical reservoir computing: A review, Neural Netw. **115**, 100 (2019).
[18] J. Torrejon, M. Riou, F. A. Araujo, S. Tsunegi, G. Khalsa, D. Querlioz, P. Bortolotti, V. Cros, K. Yakushiji, A. Fukushima *et al.*, Neuromorphic computing with nanoscale spintronic oscillators, Nature **547**, 428 (2017).
[19] D. Markovic, N. Leroux, M. Riou, F. Abreu Araujo, J. Torrejon, D. Querlioz, A. Fukushima, S. Yuasa, J. Trastoy, P. Bortolotti *et al.*, Reservoir computing with the frequency, phase, and amplitude of spin-torque nano-oscillators, Appl. Phys. Lett. **114**, 012409 (2019).
[20] S. Tsunegi, T. Taniguchi, K. Nakajima, S. Miwa, K. Yakushiji, A. Fukushima, S. Yuasa, and H. Kubota, Physical reservoir computing based on spin torque oscillator with forced synchronization, Appl. Phys. Lett. **114**, 164101 (2019).

[21] T. Kanao, H. Suto, K. Mizushima, H. Goto, T. Tanamoto, and T. Nagasawa, Reservoir computing on spin-torque oscillator array, Phys. Rev. Appl. **12**, 024052 (2019).

[22] S. Shreya, A. Jenkins, Y. Rezaeiyan, R. Li, T. Böhnert, L. Benetti, R. Ferreira, F. Moradi, and H. Farkhani, Granular vortex spin-torque nano oscillator for reservoir computing, Sci. Rep. **13**, 16722 (2023).

[23] S. Tsunegi, T. Taniguchi, S. Miwa, K. Nakajima, K. Yakushiji, A. Fukushima, S. Yuasa, and H. Kubota, Evaluation of memory capacity of spin torque oscillator for recurrent neural networks, Jpn. J. Appl. Phys. **57**, 120307 (2018).

[24] M. Romera, P. Talatchian, S. Tsunegi, F. Abreu Araujo, V. Cros, P. Bortolotti, J. Trastoy, K. Yakushiji, A. Fukushima, H. Kubota *et al.*, Vowel recognition with four coupled spin-torque nano-oscillators, Nature **563**, 230 (2018).

[25] M. Zahedinejad, A. A. Awad, S. Muralidhar, R. Khymyn, H. Fulara, H. Mazraati, M. Dvornik, and J. Åkerman, Two-dimensional mutually synchronized spin Hall nano-oscillator arrays for neuromorphic computing, Nat. Nanotechnol. **15**, 47 (2020).

[26] A. Ross, N. Leroux, A. De Riz, D. Marković, D. Sanz-Hernández, J. Trastoy, P. Bortolotti, D. Querlioz, L. Martins, L. Benetti *et al.*, Multilayer spintronic neural networks with radiofrequency connections, Nat. Nanotechnol. **18**, 1273 (2023).

[27] D. Rodrigues, E. Raimondo, V. Puliafito, R. Moukhader, B. Azzerboni, A. Hamadeh, P. Pirro, M. Carpentieri, and G. Finocchio, Dynamical neural network based on spin transfer nano-oscillators, IEEE Trans. Nanotechnol. **22**, 800 (2023).

[28] A. Slavin and V. Tiberkevich, Nonlinear auto-oscillator theory of microwave generation by spin-polarized current, IEEE Trans. Magn. **45**, 1875 (2009).

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. **12**, 2825 (2011).

[30] A. J. Robinson and F. Fallside, *The Utility Driven Dynamic Error Propagation Network* (University of Cambridge, Department of Engineering, Cambridge, 1987), Vol. 11.

[31] P. J. Werbos, Generalization of backpropagation with application to a recurrent gas market model, Neural Netw. **1**, 339 (1988).

[32] M. C. Mozer, in *Backpropagation* (Psychology Press, 2013), pp. 137–169.

[33] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Association for Computing Machinery (ACM), 2019).

[34] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.

[35] J.-G. Choi, J. Park, M.-G. Kang, D. Kim, J.-S. Rieh, K.-J. Lee, K.-J. Kim, and B.-G. Park, Voltage-driven gigahertz frequency tuning of spin Hall nano-oscillators, Nat. Commun. **13**, 3783 (2022).

[36] K.-I. Funahashi and Y. Nakamura, Approximation of dynamical systems by continuous time recurrent neural networks, Neural Netw. **6**, 801 (1993).

[37] W. A. Borders, A. Madhavan, M. W. Daniels, V. Georgiou, M. Lueker-Boden, T. S. Santos, P. M. Braganca, M. D. Stiles, J. J. McClelland, and B. D. Hoskins, Measurement-driven neural-network training for integrated magnetic tunnel junction arrays, Phys. Rev. Appl. **21**, 054028 (2024).

[38] S. Jung, H. Lee, S. Myung, H. Kim, S. K. Yoon, S.-W. Kwon, Y. Ju, M. Kim, W. Yi, S. Han *et al.*, A crossbar array of magnetoresistive memory devices for in-memory computing, Nature **601**, 211 (2022).

[39] D. E. Nikonov, P. Kurahashi, J. S. Ayers, H. Li, T. Kamgaing, G. C. Dogiamis, H.-J. Lee, Y. Fan, and I. Young, Convolution inference via synchronization of a coupled CMOS oscillator array, IEEE J. Explor. Solid-State Comput. Devices Circuits **6**, 170 (2020).

[40] S. K. Nath, X. Sun, S. K. Nandi, X. Chen, Z. Wang, S. K. Das, W. Lei, L. Faraone, W. D. Rickard, and R. G. Elliman, Harnessing metal/oxide interlayer to engineer the memristive response and oscillation dynamics of two-terminal memristors, Adv. Funct. Mater. **33**, 2306428 (2023).

[41] J. Feldmann, N. Youngblood, C. D. Wright, H. Bhaskaran, and W. H. Pernice, All-optical spiking neurosynaptic networks with self-learning capabilities, Nature **569**, 208 (2019).

[42] L. G. Wright, T. Onodera, M. M. Stein, T. Wang, D. T. Schachter, Z. Hu, and P. L. McMahon, Deep physical neural networks trained with backpropagation, Nature **601**, 549 (2022).

[43] M. J. Rasch, C. Mackin, M. Le Gallo, A. Chen, A. Fasoli, F. Odermatt, N. Li, S. Nandakumar, P. Narayanan, H. Tsai *et al.*, Hardware-aware training for large-scale and diverse deep learning inference workloads using in-memory computing-based accelerators, Nat. Commun. **14**, 5282 (2023).

[44] T. Xu, Z. Luo, S. Liu, L. Fan, Q. Xiao, B. Wang, D. Wang, and C. Huang, Perfecting imperfect physical neural networks with transferable robustness using sharpness-aware training, arXiv:2411.12352.