Analog-electronic implementation of a harmonic oscillator recurrent neural network

Pedro Carvalho,[1,*] Bernd Ulmann,[2,†] Wolf Singer,[1,3,‡] and Felix Effenberger[1,§]

[1] *Ernst Strüngmann Institute, Frankfurt am Main, Germany*

[2] *anabrid GmbH, FOM University of Applied Sciences, Frankfurt am Main, Germany*

[3] *Max Planck Institute for Brain Research, Frankfurt am Main, Germany*

(Received 22 August 2025; accepted 24 November 2025; published 22 December 2025)

Oscillatory recurrent networks, such as the harmonic oscillator recurrent network (HORN) model, offer advantages in parameter efficiency, learning speed, and robustness relative to traditional nonoscillating architectures. Yet, while many implementations of physical neural networks exploiting attractor dynamics have been studied, so far implementations of oscillatory models in analog-electronic hardware that utilize the networks' transient dynamics are lacking. This study explores the feasibility of implementing HORNs in analog-electronic hardware while maintaining the computational performance of the digital counterpart. Using a digital-twin approach, we trained a four-node HORN *in silico* for sMNIST (sequential Modified National Institute of Standards and Technology) classification and transferred the trained parameters to an analog-electronic implementation. A set of custom error metrics indicated that the analog system is able to successfully replicate the dynamics of the digital model in most test cases. However, despite the overall well-matching dynamics, when using the readout layer of the digital model on the data generated by the analog system, we observed only 28.39% agreement with the predictions of the digital model. An analysis shows that this mismatch is due to a precision difference between the analog hardware and the floating-point representation exploited by the digital model to perform classification tasks. When the analog system was utilized as a reservoir with a retrained linear readout, its classification performance could be recovered to that of the digital twin, indicating preserved information content within the analog dynamics. This proof of concept establishes that analog-electronic circuits can effectively implement oscillatory neural networks for computation, providing a demonstration of energy-efficient analog systems that exploit brain-inspired transient dynamics for computation.

DOI: 10.1103/bg2h-5tt8

I. INTRODUCTION

Physical neural networks (PNNs) represent an emerging class of computational systems that leverage physical processes for information processing [1–3]. PNNs are also closely related to neuromorphic and reservoir computing [4,5]

In PNNs, nondigital physical systems are engineered such that their natural dynamics serve as the primary computational mechanism to perform machine learning tasks such as classification and regression [6]. Various physical platforms have been investigated for the implementation of PNNs. These include analog-electronic circuits [5,7–10], photonic systems [11,12], spintronic devices [13], hybrid analog-digital architectures [14], and even purely mechanical devices [15]. These systems are designed such that their time-evolving states encode and transform information without relying on digital logic. Two main methods are commonly used to train PNNs on a given task. The first is the "digital-twin" approach, in which the physical system is simulated digitally. The resulting digital model is then trained using a gradient-based learning algorithm (backpropagation) and the learned parameters are transferred to the analog hardware [1]. The second method employs *in situ* techniques that bypass backpropagation, allowing for direct training on the physical substrate [16,17]. A comprehensive discussion on the training of PNNs can be found in Ref. [18], and see Ref. [19] for a discussion about oscillatory neural networks.

*Contact author: pedro.carvalho@esi-frankfurt.de

†Contact author: ulmann@anabrid.com

‡Contact author: wolf.singer@brain.mpg.de

§Contact author: felix.effenberger@esi-frankfurt.de

In parallel to these advances in hardware, recurrent neural networks (RNNs) composed of coupled oscillators have been shown to possess compelling computational advantages over nonoscillating architectures [20,21]. These oscillator-based architectures exhibit enhanced performance and parameter efficiency relative to conventional nonoscillating RNNs. Specifically, the harmonic oscillator recurrent network (HORN) model was shown to reproduce many aspects of cortical dynamics while outperforming nonoscillating RNNs in parameter efficiency, learning speed, and robustness to perturbations [21]. These performance gains can be attributed to the unique dynamical features of oscillatory systems [21]. For example, phase-based encoding and synchronization enable robust information representation [22,23], while resonance and wave interference facilitate selective frequency filtering [24,25]. Moreover, the properties of fading memory in these systems facilitate temporal integration across multiple time scales [21,26,27]. Unlike steady-state, typically attractor-based computational models [28], HORNs utilize transient oscillatory dynamics for computation, mirroring the operational principles observed in biological neural systems [29–32]. HORNs are biologically inspired by the ubiquity of oscillatory activity in the brain [22,33] and are particularly well suited for realization in physical analog hardware [34] where such dynamics can be implemented efficiently.

Although the computational benefits of oscillatory architectures have been demonstrated in digital simulations, their implementation in physical analog substrates remains limited. To address this gap, we present an analog-electronic implementation of a HORN model on a hybrid analog-digital computer (the anabrid Model-1 [35]) that can be programmed digitally, but performs all computations in an analog manner. The proposed implementation employs a "digital-twin" approach [36,37]. Implementing this solution involves addressing key challenges, such as precision limitations, dynamic range constraints, and parameter transfer protocols. Since we have already demonstrated the effectiveness of the HORN model in solving various machine learning tasks across several datasets (see the supplementary information of Ref. [21]), we have selected the sMNIST (sequential Modified National Institute of Standards and Technology) dataset as an exemplar in this study showcasing our proof-of-concept analog-electronic implementation of the HORN model.

To execute this proof of concept, we first trained a HORN *in silico* on an sMNIST [38] pattern-recognition task identifying optimal model parameters through supervised learning. Subsequently, we mapped the optimal parameters determined by training to the analog hardware and inferred the 10 000 MNIST samples from the test dataset using the analog implementation (see Fig. 3 later for an example). To evaluate the fidelity of the analog implementation in replicating the dynamics of the digital model, we defined and computed comprehensive error metrics. To assess the predictive accuracy of the analog implementation, we used two different readout mechanisms. We utilized the trained readout from the digital model within the analog implementation, but also investigated its potential as a reservoir [21,39,40].

The remainder of this paper is organized as follows. In Sec. II, the harmonic oscillator recurrent network model is introduced, including the adaptations necessary for its implementation in analog hardware. Section III outlines the experimental setup and the parameter-scaling procedure essential to transition the parameters from the digital model to the analog implementation. In Sec. IV, we assess the ability of the analog implementation to replicate the dynamics and task performance of its digital counterpart, while also examining its suitability as a reservoir. The paper concludes with a number of appendices.

II. NETWORK MODEL

The harmonic oscillator recurrent network model represents a recurrent neural network in which each node exhibits the dynamics of a damped harmonic oscillator (DHO unit) [21]; see Fig. 1(b). In a HORN consisting of n nodes, the dynamics of each node $1 \leq i \leq n$ is governed by the second-order ordinary differential equation (ODE)

$$\ddot{x}_i(t) + 2\gamma_i \dot{x}_i(t) + \omega_i^2 x_i(t) = F(\mathbf{x}, \dot{\mathbf{x}}, t), \tag{1}$$

where $t \in \mathbb{R}$ represents time, $x_i(t) \in \mathbb{R}$ denotes the state variable (that is, the amplitude) of the oscillator, $\omega_i > 0$ indicates the natural angular frequency, $\gamma_i > 0$ refers to the damping factor, and $F(\mathbf{x}, \dot{\mathbf{x}}, t)$ constitutes a forcing function defined subsequently. Furthermore, $\mathbf{x}(t) = (x_1(t), \ldots, x_n(t))^T$ denotes the state vector of all oscillators in the network. Note that the hyperparameters ω_i and γ_i can vary independently. Together, they determine the relaxation dynamics of each node [21,41].

The nodes are subjected to a time-varying forcing function

$$F(\mathbf{x}, \dot{\mathbf{x}}, t) = \alpha \tanh(V\mathbf{x} + W\dot{\mathbf{x}} + \mathbf{I}s(t)), \tag{2}$$

where $\mathbf{I} \in \mathbb{R}^{1\times n}$ denotes the input weight matrix, that is, I_i is a linear projection of the external input signal $s(t) \in \mathbb{R}$ to the ith node. The matrices $\mathbf{V}, \mathbf{W} \in \mathbb{R}^{n\times n}$ denote recurrent weight matrices. Specifically, the entries V_{ji} (W_{ji}) represent the strength of the amplitude (velocity) coupling from node j to node i; see also Refs. [20,21]. The diagonal entries of $\mathbf{V}$ ($\mathbf{W}$) represent the amplitude (velocity) feedback connection strengths of each node [21].

To accommodate the hardware limitations of the analog computer utilized in this study (see Sec. III), we simplify the model to allow its implementation. First, to

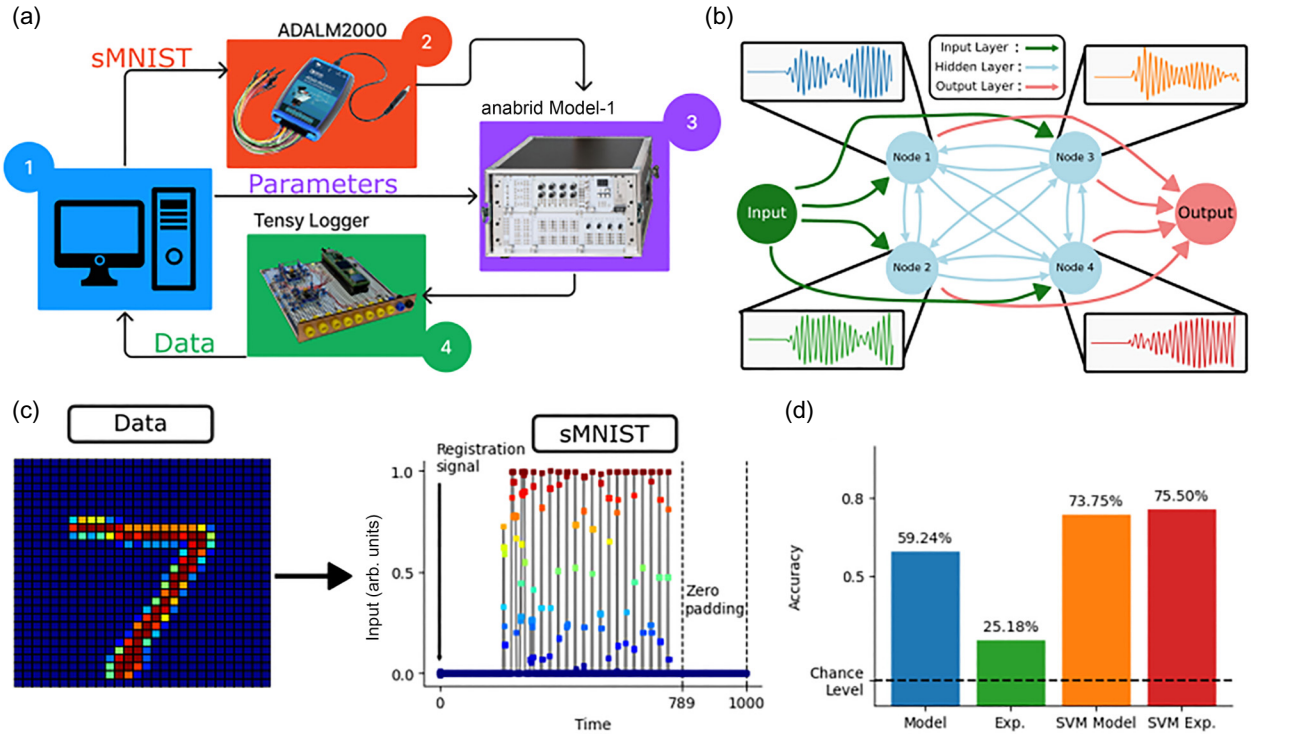

FIG. 1. Experimental setup and performance comparison of digital and analog HORN implementations. (a) Experimental setup with four components: digital computer (1), signal generator (2), analog computer (3), and data logger (4). (b) Schematic of the four-node HORN model with example dynamics for each node (inset plots). (c) MNIST preprocessing: original 28×28 sample (left) and sMNIST version with registration signal and zero padding (right); time is in pixels, and amplitude is in a normalized gray-scale (range $[0, 1]$). (d) Classification accuracy of digital model (blue), analog with digital readout (green), digital with SVM readout (orange), and analog reservoir with SVM readout (red).

accommodate the limited number of computational elements (adders, integrators, and coefficients), we chose a model consisting of $n = 4$ nodes. For the same reason, we also eliminate all feedback connections ($\mathbf{V}_{ii} = 0, \mathbf{W}_{ii} = 0$). Second, since an implementation of the tanh nonlinearity is not available, we excluded this function from the forcing function [Eq. (2)]. Third, all recurrent amplitude coupling magnitudes ($\mathbf{V}$) are constrained to the interval $[0, 1]$, prohibiting both inhibitory (negative) and amplifying couplings. The restriction to non-negative values is dictated by circuit limitations, while the prohibition of amplifying couplings aims to prevent runaway dynamics. These constraints enable the implementation of the network on the Model-1 analog computer utilized in this study [35]. For completeness, we also conducted tests on a velocity-coupled network in which we removed all amplitude couplings ($\mathbf{V} = 0$) and velocity feedback connections ($W_{ii} = 0$). Details are provided in Appendix E. We found that the amplitude- and velocity-coupled networks behaved similarly and only report on the findings for the amplitude-coupled network in the main text.

We employ a digital-twin approach to obtain network parameters for an sMNIST digit classification task by simulating the network on a digital computer using a time-discretized version of the HORN model. To simulate the HORN model on a digital computer, we begin by converting the second-order ODE [Eq. (1)] into a family of first-order ODEs through the introduction of an auxiliary variable $\dot{x} = y$. Subsequently, we develop a time-discrete representation of the resulting system using a microscopic time constant h, as outlined in Ref. [21]. Integrating this system through an Euler integration scheme and imposing the experimental constraints leads to the network update equations

$$\begin{aligned} x_{i,t+1} &= x_{i,t} + h y_{i,t+1}, \\ y_{i,t+1} &= y_{i,t} + h \left[\sum_{j \neq i}^{n} (V_{ji} x_{i,t}) + I_i s(t) - 2\gamma y_{i,t} - \omega_i^2 x_{i,t} \right], \end{aligned} \tag{3}$$

where i denotes the node index and t is a discrete-time index; see Ref. [21]. Note that a symplectic Euler integration [42] is needed here to guarantee numerical stability, since the system is stiff [43].

When simulating the discrete-time model [Eq. (3)], we establish a connection between the time scales of the

input and the system by presenting one pixel (intensity value) of an sMNIST stimulus $s(t)$ per iteration step, effectively setting $h = 1$ in Eq. (3). The choice of $h = 1$ allows us to express all time-related quantities in "pixel" time units, improving the interpretability of model quantities and hyperparameters. In particular, this holds for the natural frequency parameters ω_i, which can then be specified in radians per pixel, and the associated period $\tau = 2\pi/(\omega h)$. Note that the system remains invariant under changes in $h \to h'$, provided that the system parameters are appropriately rescaled according to

$$\gamma' = c\gamma, \quad \omega' = c\omega, \quad \alpha' = c^2\alpha, \tag{4}$$

where $c = h/h'$; see Ref. [21]. The frequency $f_N = 1/(2h)$ represents the upper limit of observable frequencies within the system, the Nyquist frequency. In practice, networks should not operate at frequencies near f_N, as the numerical integration error increases when approaching this limit.

The networks were trained on an sMNIST handwritten digit classification task, a common benchmark for RNNs [20,21]. For sMNIST, the 28×28 pixel MNIST samples are serialized into a time series of length $28 \times 28 = 784$ pixels by collecting the intensity values from top left to bottom right in scanline order [Fig. 1(c)]. An affine readout is performed at the last time step of stimulus presentation ($t = 784$ pixel) to perform digit classification, with $\mathbf{M} \in \mathbb{R}^{n\times n}$ denoting the readout matrix, and $b \in \mathbb{R}^n$ as the output bias vector. The model was implemented in PYTORCH [44] and the backpropagation through time (BPTT) algorithm was used to train all model parameters ($\mathbf{I}$, $\mathbf{V}$, b, and $\mathbf{M}$). Following a digital-twin approach [1], the learned weights (except $\mathbf{M}$ and b) are subsequently transferred to the analog implementation of the model.

III. EXPERIMENTAL SETUP

This section outlines the hardware utilized (Sec. III A), the circuit architecture for each DHO unit in the analog implementation (Sec. III B), and the methodology for data preparation and presentation to the analog framework (Sec. III D). Additionally, we also show the process by which model parameters are scaled between the digital simulation and the analog implementation (Sec. III E).

A. Hardware setup

The experimental setup [Fig. 1(a)] consists of four main parts: (1) a digital computer, (2) a signal generator, (3) an analog circuit, and (4) a data logger. The digital computer (1) controls the experiment and performs three different tasks as described below. First, it normalizes the model parameters and configures the analog circuit (3) (see Sec. III E). Second, it normalizes the sMNIST input samples and transmits them to the signal generator (2) (see Sec. III D). Third, it collects the recorded data from the data logger (4) following each inference run for subsequent analysis.

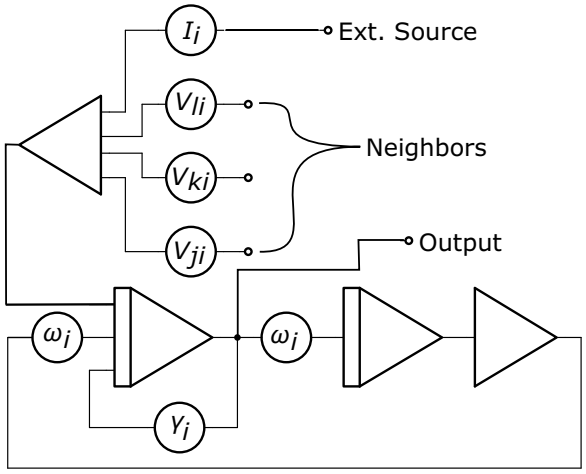

FIG. 2. Analog circuit implementation of a damped harmonic oscillator node. The circuit contains integrators (triangular shapes with rectangles), summers (triangular shapes), and coefficients (circular elements) that can be digitally programmed to values in $[0, 1]$. Note that summers and integrators perform implicit sign inversion in this analog implementation.

The signal generator (2) is an ADALM2000 [45]. This device receives standardized sMNIST data [Fig. 1(c)] from the digital computer (1) through a USB connection. Utilizing its signal generator functionality, it converts the digital data into an analog voltage trace, which is then fed to the analog circuit (3).

The analog circuit (3) was implemented using an anabrid Model-1 hybrid analog-digital computer [35]. Each network node was implemented as a damped harmonic oscillator (see Fig. 2 in Sec. III B).

The Teensy Logger [46] (4) in Fig. 1(a) is custom hardware interfacing with the digital computer (1), the signal generator (2), and the analog circuit (3). It connects to the digital computer via USB and provides five analog input channels that are used to record the voltage traces of all four nodes as well as the input signal (to allow for precise temporal registration of the input and the nodal dynamics). Data are stored internally until retrieved by the digital computer after each inference run.

B. Analog circuit

The HORN model was implemented on an anabrid Model-1 analog computer, a modern, modular analog computer [35]. An analog computer is composed of various computing elements, such as integrators, summers, multipliers, and coefficients, that can be interconnected to model and solve a given problem [47,48]. Analog computers operate nonalgorithmically, lacking explicit memory, instructions, and a central processing unit, and are ideally suited to solve problems that can be described as

systems of coupled differential equations [47]. The "in-memory computation" implemented in analog computers is well suited to simulate recurrent dynamics and can overcome the conceptual slowness of simulating dynamics on digital von Neumann systems for which time has to be discretized, and each dynamics step sequentially requires loading, updating, and writing of system state. This makes such a system an ideal substrate to model a HORN consisting of a recurrently coupled network of damped harmonic oscillator units [Eq. (1)]. Quantities in analog computers are typically represented in abstract machine units confined to the interval $[-1, 1]$; see Ref. [47].

The circuit diagram of one of the four oscillators implemented in this study is presented in Fig. 2. The core oscillator circuit (lower right) consists of two integrators, a summer used for sign inversion, and two coefficients with value ω. This implements the basic differential equation of a harmonic oscillator, $\ddot{x} = -\omega^2 x$. The leftmost integrator has an additional feedback path from its output to one of its inputs by means of an additional coefficient of value γ. Since the integrator performs an implicit change of sign, this feedback represents a damping term γ controlling the decay rate of the oscillator.

Each oscillator has one output and four inputs, which are summed together using the summer shown in the upper left of Fig. 2. This summer serves a dual purpose: First, there are not enough inputs on the integrator to feed all inputs directly to it. Second, the sign of the input signals must be flipped to have the oscillators run in phase when coupled. The top input is the global excitation signal ("Ext. Source"). The three lower inputs are connected to neighboring oscillators.

The complete circuit implements an all-to-all connected HORN on four nodes that are coupled on their amplitudes using 12 coefficients, with an additional four coefficients for controlling the coupling to the external input signal. See Appendix E for the velocity-coupled case.

C. Physical units

Moving from the digital model to an implementation in analog hardware requires proper conversion between the units of the two systems, and between discrete- and continuous-time formulations. This scaling is essential to ensure that the analog implementation replicates the dynamics observed in the digital simulation [47]. In analog hardware, the unit system is established by the characteristics of the component elements within the analog-electronic circuit, such as the properties of integrators. In contrast, the digital model employs a more flexible unit system, as all quantities are represented by floating-point numbers that can be arbitrarily scaled according to Eq. (4). Since the digital model operates in discrete time and the analog implementation operates in continuous time, we first need to establish a relationship between the discrete- and continuous-time parametrizations of the system. Subsequently, we define the scaling relationships between all other model parameters.

D. Temporal scale

In contrast to digital simulations, where time is an abstract quantity, analog-electronic implementations necessitate running system dynamics for a defined duration T, measured in seconds. Although the physical parameters of the analog circuit are fundamentally constrained only by the properties of its components (see Sec. III B), accurately mapping the digital model to the analog implementation requires the selection of a parameter (for example, γ, ω, or h) to serve as the basis for this mapping. The selection of this parameter is arbitrary; however, it has to be made with the objective of replicating the dynamics of the chosen digital model. Consequently, once a parameter is established, all remaining parameters are derived from this selection. For example, if ω is designated as a basis, then γ will be determined by the definition of ω.

To achieve this, we fix the duration T for presenting an sMNIST digit to the system in a way that meets several constraints, as described below. This choice is equivalent to setting the value of h in the digital model, as establishing the total time is effectively the same as setting the pixel input duration in real time. The critical quantity in this context is the machine integration factor k_0 of the integrators [47]. We opted to set the time scale of the analog computer by setting $k_0 = 10$, such that a value of $\omega_{\max} = 1$ (the maximal possible natural frequency of a node, in machine units; see Sec. III B) corresponds to a frequency of 16 Hz. Since HORNs exploit resonance for feature extraction, optimal natural frequencies for sMNIST processing are near $\omega^* = 2\pi/28$ (in rad/pixel units), as previously demonstrated [21]. Considering an analog implementation in which ω^* corresponds to a maximal value of the frequency coefficient ($\omega_{\max} = 1$), the duration of the experiment must allow at least 28 cycles of a node, which corresponds to a minimal duration of $T_{\min} = 28/16 = 1.75$ s. To avoid inaccuracies of the analog circuit occurring at extreme parameter values, we set the experiment length to $T = 6$ s, and scale system parameters as described in Sec. III E.

Our experimental setup incorporates components operating at three distinct sampling frequencies [Fig. 1(a)]. As a result, each system component yields a different total number of data points given a duration T of an experimental run. For instance, the number of stimulus samples for the sMNIST digit is L, the number of samples from the signal generator is N_I, and the number of samples recorded by the data logger is N_O [Fig. 1(a)]. To ensure that the different components of the experiment have a coherent representation of time, these quantities should be commensurate—preferably they should satisfy $N_I/L, N_O/L \in \mathbb{N}$.

TABLE I. Experimental parameters for the analog HORN implementation.

Symbol	Description	Value
L	sMNIST sample size	1000 pixels
T	Total time of the experiment	6000 ms
f_I	Sampling frequency of the input	75 000 000 Hz
S_r	Sample repetition	450 000
N_I	Number of input data points	450×10^6
τ_O	Output sample interval	3 ms
N_O	Number of output data points	2000
k_0	Computational integration factor	10

For example, given L and N_O, let $N_O/L = a$. Thus, a data points recorded by the data logger correspond to the time period associated with one pixel of the digital model.

To allow for a registration of the signal generator and the data logger in the recorded data, a registration signal is added at the beginning of each sMNIST sample [Fig. 1(c)], as the data logger records the input generated by the signal generator [Fig. 1(a)]. Moreover, we zero-pad the sMNIST samples at the end to obtain samples of length $L = 1000$ pixels, as this simplifies the calculations of all time-related quantities [Fig. 1(c)].

The signal generator internally operates at a frequency of $f_I = 75$ MHz [45]. To present the standardized input consisting of $L = 1000$ data points [Fig. 1(c)] within the time window T, the signal generator repeats each input data point $S_r = 450\,000$ times, resulting in a total of $N_I = 450 \times 10^6$ input data points.

The data logger has the capacity to store up to 16 535 samples across five recording channels [46]. Since we utilize all five recording channels (input and amplitudes of four oscillators), recording a single time step requires storing five samples (we call this a *data point*). This limits the maximum number of data points to $16\,535/5 = 3307$. To ensure that we can record an entire run, we configure the data logger with a sampling interval of $\tau_O = 3$ ms, resulting in $N_O = 2000$ data points per run. This choice also ensures that the temporal sampling is not too coarse, resulting in two samples within the duration of each input sMNIST pixel. The parameters are summarized in Table I.

E. Scaling

This section describes the process of mapping all model quantities to machine units, which is essential for the operation of the analog computer. Analog computers typically code variables in the range $[-1, 1]$ and parameters in the range $[0, 1]$, and physical dimensions are expressed in "machine units"; see Ref. [47]. Values of variables that exceed these limits result in clipping, which should be avoided, to prevent this leading to invalid results. In what follows, let $\Delta S = T/L$ denote the sample duration (Table I), let k_0 denote the machine integration factor [47] (see Sec. III B), let ω denote the natural frequency, let γ denote the damping factor, and let $V_{(i,j)}$ denote the coupling strength from node i to j. To differentiate between the parameters of the digital model (defined in discrete time) and those of the analog implementation (defined in continuous time), we use subscripts "M" and "E", respectively. The scaling relationship between the parameters of the digital model and their corresponding experimental values is expressed by

$$\begin{aligned} c &= \Delta S k_0, \\ I_E &= \frac{1}{c}\frac{I_M}{\omega_M}, \\ \gamma_E &= \frac{\gamma_M}{c}, \\ \omega_E &= \frac{\omega_M}{c}, \\ V_{(i,j),E} &= \frac{1}{c}\frac{V_{(i,j),M}}{\omega_{j,M}}. \end{aligned} \tag{5}$$

The final quantity that requires scaling is the input matrix I. This matrix governs the magnitude of the input drive applied to each node, as described in Eq. (2), and consequently determines the dynamic range of the network. To prevent clipping, it is necessary to scale the entries of I, ensuring that the dynamic range of all nodes in the network remains within the valid range ± 1 (in machine units) for each experimental run. Theoretically, a single scaling of the entries of I would suffice, given the maximal dynamic range across all runs. However, this approach may lead to very small dynamic ranges for some experimental runs. Because of limitations in machine precision (in our setup, it is $\Delta = \pm 0.03$ [35]), the analog computer may not accurately reproduce the dynamics of experimental runs characterized by a limited dynamic range. Consequently, set-and-forget scaling may compromise the model performance of the analog implementation.

To achieve optimal model performance in the analog implementation, we distinctly scale the entries of I for each experimental run. This approach guarantees that the effective dynamic range of the analog computer is maximized given the sample-specific dynamic range of the digital model.

The scaling of I is performed through a sequence of simulations utilizing the digital model. In this process, we modify the scaling of the input matrix using a scalar multiplier to achieve maximal nodal amplitudes in the analog implementation, while ensuring that these amplitudes remain within the valid range $[-1, 1]$ (see Fig. 7 in Appendix B). This scaling procedure exemplifies a hybrid computing approach [48].

In the rescaled system, the variables, specifically the amplitudes x [Eq. (3)], will be within the dynamic range

of the analog machine. This observation is illustrated in Fig. 3 and later in Fig 6.

IV. RESULTS

Given the limited number of computational elements of the anabrid Model-1 analog computer used for this study, we evaluated the proposed setup utilizing a HORN model comprising four nodes (Fig. 3). We constructed a homogeneous HORN in which all nodes have the same values for the natural frequency, $\omega_M = 0.22$, and the damping, $\gamma_M = 0.01$. The model was trained *in silico* using BPTT [21], achieving a classification accuracy of 59.24% on sMNIST [Fig. 1(d)]—a notable result given the small size of the network. Following training, we use these parameters in our analog implementation, referred to as the analog twin (see Sec. III E). We then performed inference on the 10 000 sMNIST test samples using the analog twin (Fig. 3).

To perform the inference, we processed the recorded time-series data from all nodes through an affine readout layer. This layer maps the configuration of nodal amplitudes at pixel $T = 789$ to predictions of MNIST digits (0–9) [see Fig. 1(c)]. Applying the readout weights from the digital model to the data produced by the analog implementation resulted in a 28.39% correspondence with the predictions of the digital model. Among the 2839 samples for which both the digital and the analog twin predicted the same label, 1973 corresponded to the ground-truth MNIST label (see Fig. 8 later). We will show that this limited agreement between the predictions of the digital and the analog twin when using the readout layer of the digital model results from the digital model's reliance on high-precision floating-point representations. Furthermore, we demonstrate that the analog twin preserves the dynamics of its digital counterpart. Using alternative readout strategies, we can implement a readout on the analog-generated data using a newly trained linear support vector machine

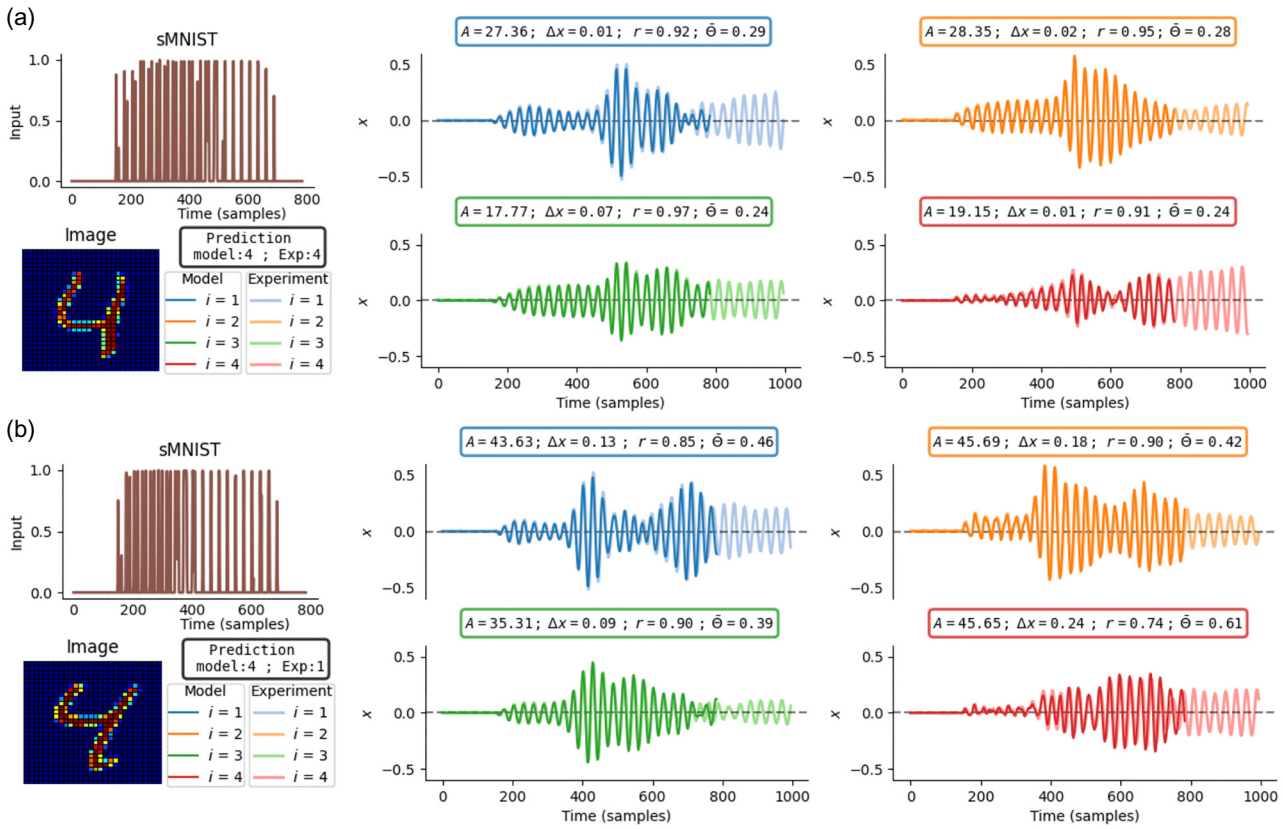

FIG. 3. Comparison of analog and digital HORN dynamics for representative sMNIST samples demonstrating different reproduction outcomes: (a) case with matching predictions; and (b) case with mismatched predictions. The left-hand column shows input sMNIST sample (top) and corresponding MNIST image (bottom). Subsequent columns display the dynamics of each of the four nodes, with light traces showing analog implementation dynamics and dark traces showing digital-twin dynamics. Time is measured in pixels for the digital twin and mapped to seconds in the analog implementation ($T = 6$ s). Amplitudes are in machine units (with range $[-1, 1]$). Error metrics quantify reproduction fidelity: A, area between traces; Δx, amplitude mismatch at $T = 789$ pixel; r, correlation; and $\overline{\Theta}$, average phase difference.

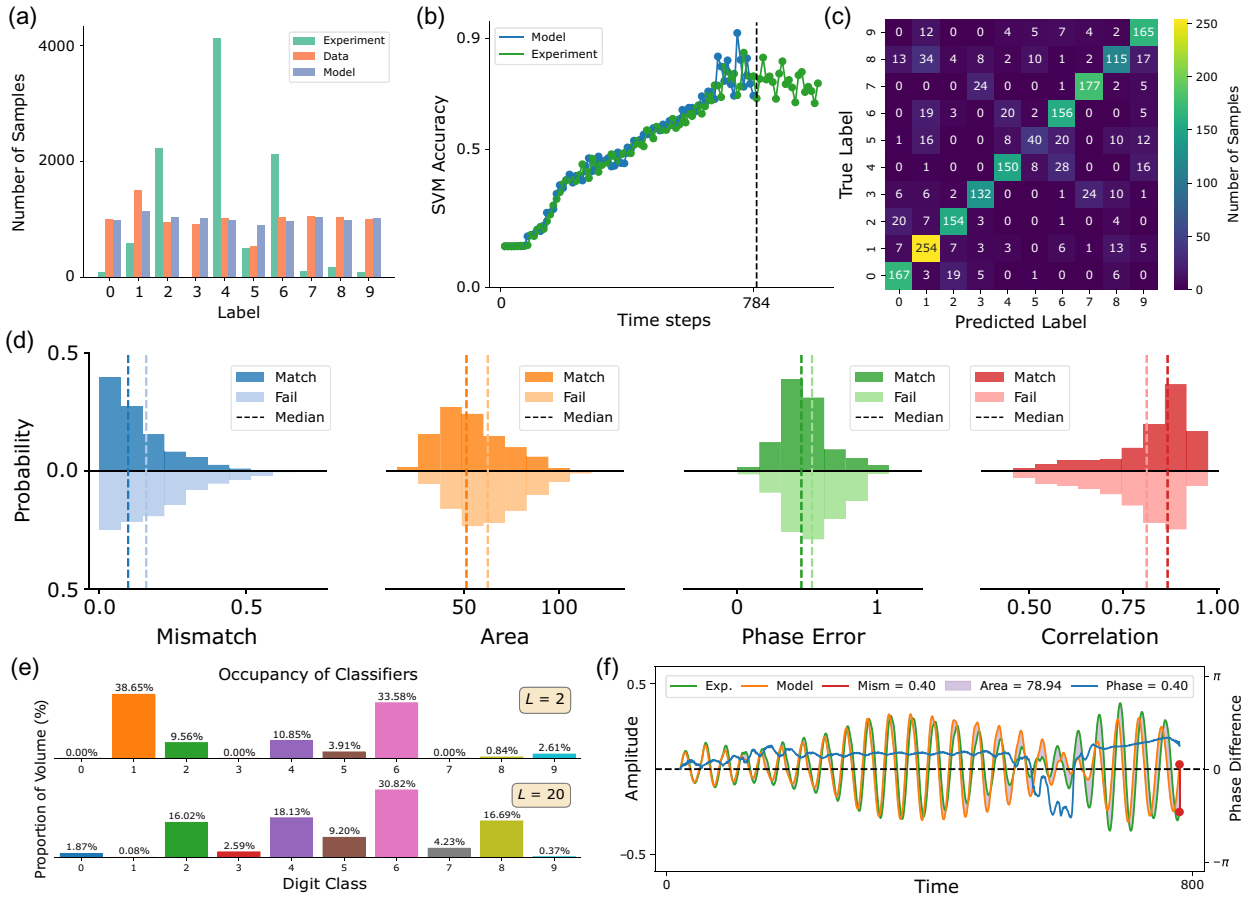

FIG. 4. Analysis of the analog implementation of the HORN model. (a) Distribution of predicted labels showing class-specific differences: analog implementation (Experiment), digital twin (Model), and true MNIST labels (Data). Digits 0, 3, 7, and 9 are underrepresented in the analog implementation due to precision limitations. (b) SVM readout accuracy over time demonstrating that both digital and analog systems achieve similar reservoir performance (≈74%) when appropriate readouts are used, confirming preserved information content. (c) Confusion matrix for the analog implementation with an SVM readout at $T = 789$ pixels, showing recovery of all digit classification capabilities when trained directly on analog dynamics. (d) Error metric distributions comparing cases where analog and digital predictions match (top, low error) versus fail to match (bottom, higher error): mismatch (Δx), area (A), phase ($\overline{\Theta}$), and correlation (r) metrics. Dashed lines indicate medians. (e) Volume occupancy analysis in four-dimensional amplitude space (x_1, x_2, x_3, x_4) illustrates precision limitations. Small label volumes (digits 0, 3, 7, and 9) require higher precision than analog hardware provides. Top: $L = 2$ (dense grid near origin). Bottom: $L = 20$ (wide dynamic range). (f) Error metrics visualization showing analog (green) versus digital (orange) node dynamics in the $[5, 789]$ pixel time interval. Displayed error metrics are: area difference (purple), amplitude mismatch (red), and instantaneous phase difference (blue trace, right axis).

(SVM), which recovers the performance level of the analog twin to the same extent as that of its digital counterpart.

To systematically evaluate the degree to which the analog model replicates the dynamics of the digital twin (Fig. 3), we introduce several error metrics calculated on a per-node basis [see Fig. 4(f)]: (a) "mismatch," the difference in nodal amplitudes between digital and analog implementations at $T = 789$ pixel [see Fig. 1(c)]; (b) "area," the difference in total area under time series of nodal amplitude; (c) "phase," average instantaneous phase difference between the time series; and (d) "correlation," the temporal correlation between the time series. Here, (b), (c), and (d) are computed on the interval $[5, 789]$ pixel [see Fig. 1(c)].

After computing the error metrics for all nodes across 10 000 test samples, we analyzed the distributions in two scenarios: where the label predictions of the analog and digital twins align, termed "Match," and where they do not, termed "Fail" (Fig. 5). Network-scale metrics were derived by averaging nodal metrics [Fig. 4(d)]. As expected, lower values of the error metrics were associated with greater agreement between analog and digital predictions [Fig. 4(d)]. Notably, even when the digital twin produces an incorrect prediction, the dynamics of

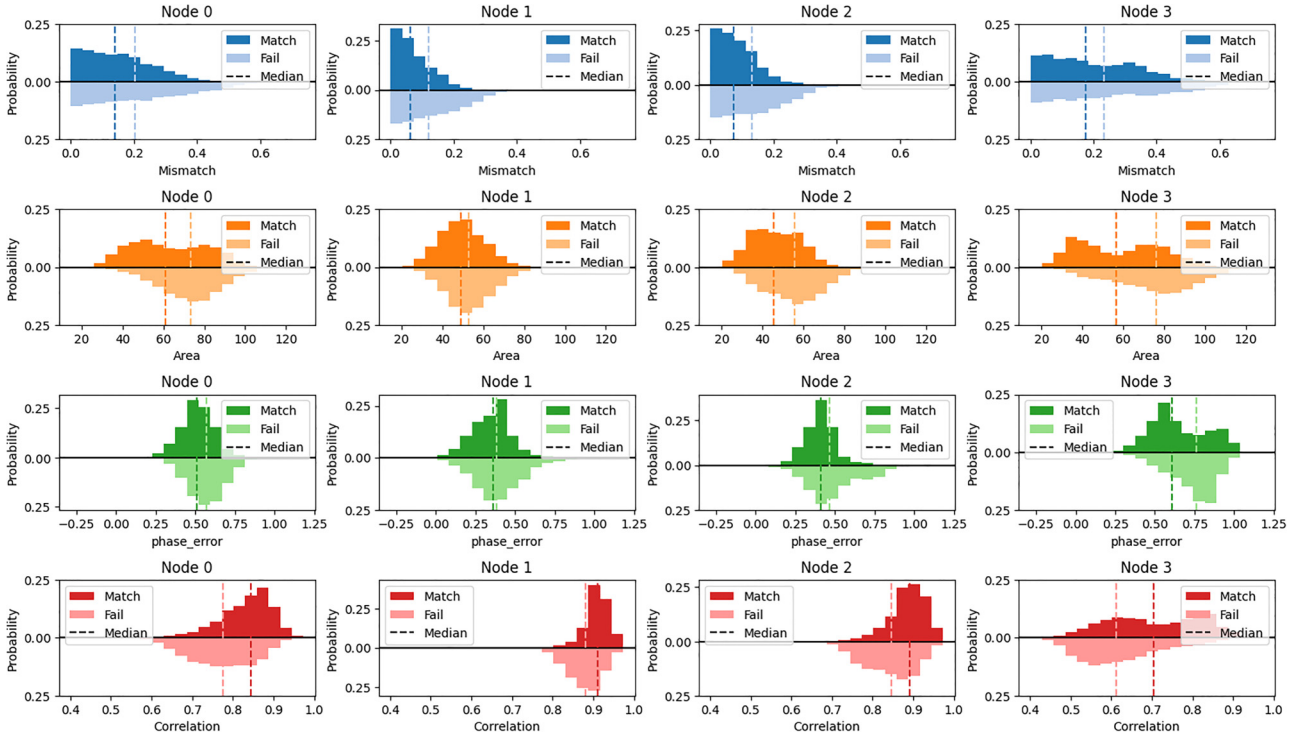

FIG. 5. Node-wise error metric distributions for prediction agreement analysis. Probability densities of error metrics (mismatch, area, phase, and correlation) for each of the four nodes, separated by cases where analog and digital predictions match versus fail to match.

the analog implementation frequently replicate those of the digital twin (Fig. 3). Visualization of the error metrics [Fig. 4(f)] reveals that inference runs with error values near the median of these metrics [Fig. 4(d)] correspond to cases where the analog implementation successfully replicates the dynamics of its digital counterpart. Overall, these findings indicate that the analog twin is capable of reproducing the dynamics of the digital model in most cases.

To better understand the differences between the predictions of the digital model and its analog twin, we analyzed the distribution of predicted labels for each digit class [Fig. 4(a)]. We find that, at the digit class level, the analog implementation systematically fails to predict some labels (namely 9, 7, and 3), whereas these labels are predicted correctly by the digital model. This phenomenon warrants further analysis. Comparing the predictions by the digital and the analog twin results in four possible scenarios: both twins accurately predicting the digit, both failing to predict correctly, or one twin making a correct prediction while the other fails (see Fig. 8 later). This study primarily investigates the capacity of the analog twin to replicate the dynamics of the digital twin. Consequently, we concentrate on instances where both twins produce identical predictions, irrespective of their correctness with respect to the ground truth.

To better understand the reasons for the limited agreement between the predictions of the digital and the analog twin (28.39%, Fig. 8 later), we analyzed the volume that each label occupies in the four-dimensional amplitude space (x_1, x_2, x_3, x_4) at the time step used for readout (referred to as "decision space"). The volume associated with each label represents the subspace where the network predicts a specific digit class. To facilitate this analysis, we created a four-dimensional grid by discretizing each dimension x_1, x_2, x_3, and x_4 into $N = 75$ evenly spaced bins, thereby covering the interval $[-L/2, L/2]$ in each dimension. We considered two lattice sizes: $L = 10$, which spans a cube corresponding to the maximum amplitude of any node in the analog system computed across all samples; and $L = 2$, which produces a smaller lattice with a higher density of points near the origin $(0, 0, 0, 0)$.

We observed that certain digit classes (0, 3, 7, and 9) occupy significantly smaller volumes within the decision space compared to other digits [Fig. 4(e)]. Accessing these regions with small volumes necessitates high numerical precision. This indicates that the digitally trained network exploits floating-point precision capabilities for making its

predictions. However, the degree of precision of the digital twin is orders of magnitude higher than the precision of the analog implementation. Therefore, some labels with small volumes in the decision space are not accessible to the analog twin [see Fig. 9(b) later]. For this reason, we can conclude that the difference in precision between the digital and the analog twin is one of the factors that results in incorrect predictions, which explains the discrepancy in performance between the two twins.

However, this precision mismatch represents a solvable limitation rather than a fundamental constraint. Appropriate readout strategies can recover the performance of the analog implementation to the same degree as that of its digital counterpart, as demonstrated below.

To test whether the analog system preserves the same information that is present in the digital system in its dynamics, we discarded the affine readout layer trained as part of the digital model and replaced it with a linear support vector machine trained on the data generated by the analog implementation. Thus, the analog twin was utilized as a reservoir [39]. For comparison and to avoid creating an unfair advantage for the analog twin, the same procedure was applied to its digital counterpart [Fig. 4(b)]. The new setup with an SVM-based readout [Fig. 1(d)] allows the analog twin to achieve the same level of performance as its digital counterpart [Fig. 4(b)]. In particular, it also recovers prediction accuracy across all digit classes [Fig. 4(c)]. These results indicate that the analog twin preserves the information present in the digital model. Furthermore, this observation indicates that analog neural networks can attain performance comparable to digital systems when utilized effectively.

V. DISCUSSION

This study showcases an implementation of the harmonic oscillator recurrent neural network model [21] in analog-electronic hardware, demonstrating that the analog implementation is able to reproduce the essential dynamics underlying model performance. In particular, we employed a digital-twin approach [1] to train a small-scale HORN on a sequential MNIST (sMNIST) classification task. The learned parameters were subsequently transferred to an analog system for inference. For this proof of concept, we utilized a HORN model in its simplest configuration, characterized by a homogeneous network in which all nodes possess identical natural frequencies and damping coefficients. Additionally, we eliminated all feedback mechanisms present in the original HORN model. Two network architectures were considered and both were found to faithfully reproduce the dynamics of its digital twin, an amplitude-coupled one and a velocity-coupled one (see Appendix E).

Quantitative analyses indicated that, despite hardware constraints such as a finite number of circuit components, a restricted dynamic range, and limited precision, the analog implementation was able to replicate the transient oscillatory dynamics of the digital model with high fidelity. Despite the small size of the model and the additional simplifications imposed by the limitations of the analog computer, the digital model achieved an accuracy of 59.24% on the sMNIST digit classification task using the low number of 66 trainable parameters (of which only 16 were used in the analog twin, and 50 belong to the readout layer). This parameter efficiency of HORN models over other RNN architectures was previously demonstrated [21], which makes this model particularly well suited for implementation in resource-constrained analog systems where parameter efficiency is crucial. Dropping these requirements on model size and architecture necessitated by constraints of the analog hardware allows HORN models to achieve state-of-the-art performance on various tasks for various datasets; see Ref. [21].

Error analysis indicates that the primary limitation of the analog model stems from the limited precision and dynamic range of its hardware, which results in reduced performance when applying the original digital readout weights to the analog model directly. This mismatch particularly affected MNIST digit classes whose decision boundaries required fine-grained amplitude discrimination (namely, the digits 0, 3, 7, and 9). Additional limitations stem from the simplified network used here, including the small number of nodes, the absence of nonlinear activation functions, and the restricted connection weights, which were imposed by the limited number of circuit elements of the analog computer used in this study.

Importantly, these effects are not inherent to oscillatory analog computation, but reflect constraints of the readout strategy and hardware resolution. When used as a reservoir with a separately trained readout provided by a linear SVM, the analog system achieved classification accuracy equivalent to the digital model (75.50% versus 73.75%, respectively), indicating that the dynamics of the analog system preserves sufficient information for accurate decoding. To address the limitations associated with readout, we propose that effective analog computation using oscillator networks may necessitate task-specific decoding strategies that exploit the full temporal structure of the network, rather than relying on fixed-time-point linear readouts [49]. Furthermore, we hypothesize that training with perturbed inputs or with intrinsic noise could enhance robustness and improve generalization. However, it is important to note that the implementation of such strategies may require larger networks to maintain performance. Finally, another approach involves training the analog implementation directly. Prior work has demonstrated the success of such physical training methods [50],

which we plan to investigate in the context of HORNs in a future study.

Our results have practical implications for the design of analog neural networks. (i) We found that readout mechanisms may be more critical than internal dynamics for achieving target performance, suggesting that reservoir computing approaches may be particularly suited for analog implementations. (ii) Our 16-parameter analog implementation demonstrates the feasibility of deploying neural networks in strongly resource-constrained environments where digital alternatives are impractical, such as edge computing applications [51] or energy-limited sensors. (iii) The implementation of the HORN model in this setting supports the broader proposition that oscillatory transient dynamics [32] offers a viable substrate for robust, efficient, and real-time physical computation, in contrast to networks based on the principles of steady-state attractors [52].

The current work opens up several avenues for designing energy-efficient, brain-inspired computing architectures and hardware [53]. In particular, analog oscillator networks with parallelized information processing might benefit from neuromorphic computing approaches [4], potentially enabling fully analog learning systems [3,50]. Future work may explore larger networks [54], the inclusion of nonlinearities, spiking networks that implement similar principles [43,55], and incorporating *in situ* learning methods to bypass the need for parameter transfer from a digital twin and thereby potentially enabling the construction of entirely analog systems capable of real-time, low-power machine learning.

In summary, the proof of concept presented here demonstrates that brain-inspired analog-electronic oscillator networks and their transient dynamics can serve as physically instantiated recurrent neural networks capable of performing machine learning tasks with extreme power efficiency [6]. With continued advances in programmable analog hardware, oscillator-based recurrent networks could become a practical building block for real-time, energy-efficient, and fully analog neuromorphic computing systems.

DATA AVAILABILITY

The data that support the findings of this article are openly available [56,57].

APPENDIX A: ERROR METRICS

To assess the difference between the dynamics of the analog implementation and its digital counterpart, we defined the following error metrics, calculated node-wise between the analog implementation and the digital twin:

(a) *Mismatch*, the difference in amplitudes at the last time step.

(b) *Area*, the total area between the time series.

(c) *Phase*, the average instantaneous phase difference.

(d) *Correlation*, the correlation between the two time series.

These metrics can be visualized as in Fig. 4(f), and network samples (Fig. 6) provide an intuitive qualitative understanding of how these values relate to dynamics reproduction.

Comparative histograms of these error metrics for cases where labels match versus fail to match between analog and digital implementations are shown for each node individually in Fig. 5. The concatenated data are presented in a condensed format in Fig. 4(d).

APPENDIX B: SCALING ALGORITHM

Prior to any run of the analog implementation, the digital computer [Fig. 1(a), item 1] loads the network parameters and the input sample. With the network parameters, the digital computer undergoes a simulation loop (Fig. 7) to obtain a rescaling factor s. This rescaling factor is applied to the input matrix I [Eq. (2)] to ensure that the network dynamic range can fit into the analog implementation nonclipping amplitude interval.

To illustrate the rescaling algorithm operation (Fig. 7), consider the case where the digital computer loads the network data and simulates the model for the first time. After simulating the network for the first time, the digital computer obtains the network dynamic range ($\max(|x|)$). This value should lie within a predefined range $[\mathit{floor}, \mathit{ceil}]$. If $\max(|x|) < \mathit{floor}$, the amplitude is insufficient, meaning that the network dynamic range is comparable to the machine precision limit. A rescaling correction, r, is then calculated to alter the scaling factor s. In the case of insufficient amplitude, s is increased and a new simulation is run. If $\max(|x|) > \mathit{ceil}$, the experiment might face clipping problems; in this case s is reduced and a new simulation is run. This process is repeated until $\max(|x|)$ lies inside the desired experiment range (Fig. 7).

Since resonance effects are expected, we opted to set $\mathit{ceil} = 0.6$, leaving a margin for the system before reaching its limits, and set $\mathit{floor} = 0.1$.

APPENDIX C: ORIGINAL DECODER

The straightforward approach for decoding the analog circuit output uses the same decoder as the digital twin: an affine readout layer with readout weights determined by the digital BPTT training procedure.

The decoding performance can be visualized using a Venn diagram showing the coincidence of the ground-truth labels, the label predictions by the digital twin, and the label predictions by the analog twin on a per-class basis (Fig. 8).

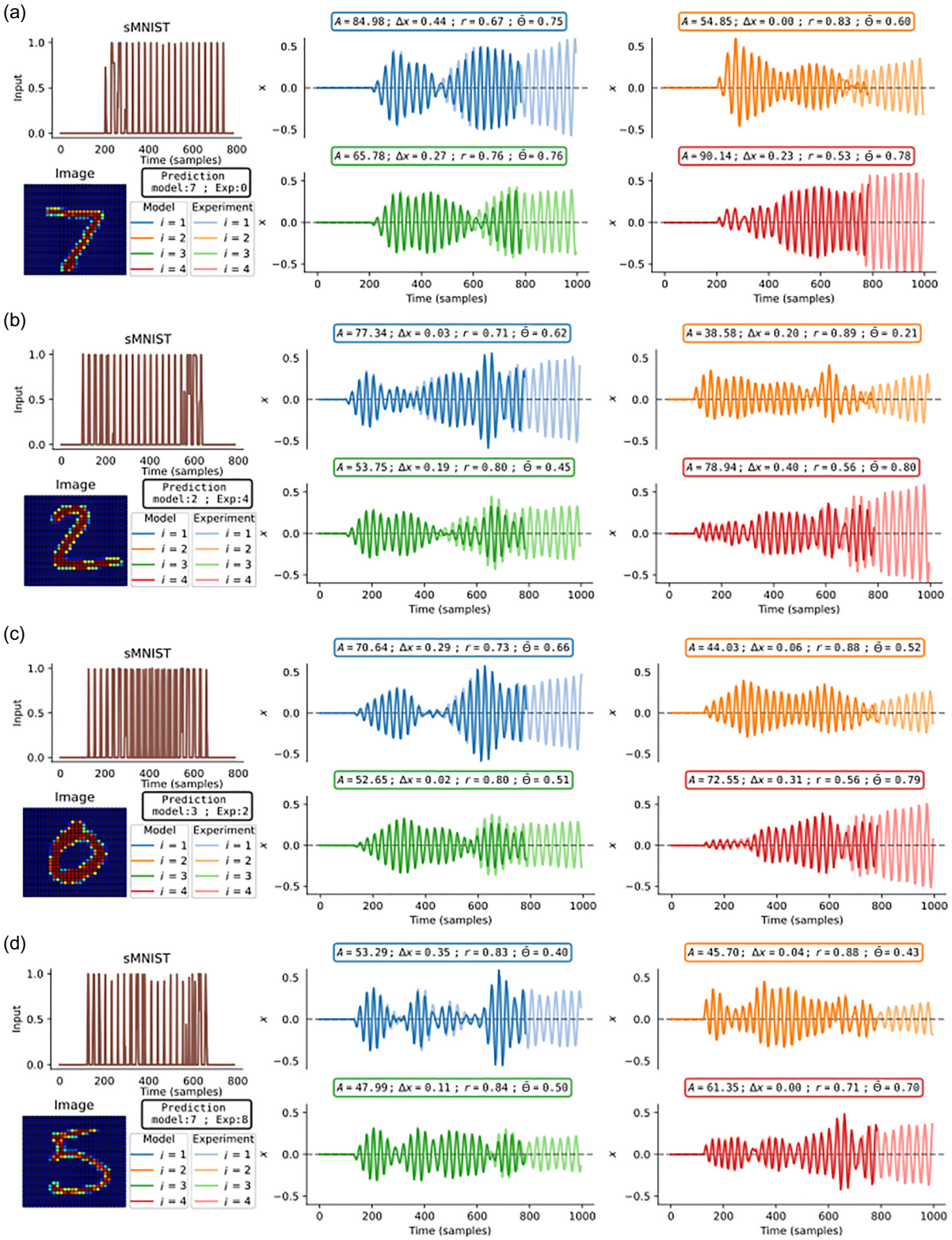

FIG. 6. Additional examples of the analog and digital dynamics comparison across different input samples. The left-hand column displays input in both sMNIST (serialized) and MNIST (image) formats. The central and right-hand columns show the corresponding node dynamics, with light traces representing analog implementation and dark traces representing digital-twin dynamics.

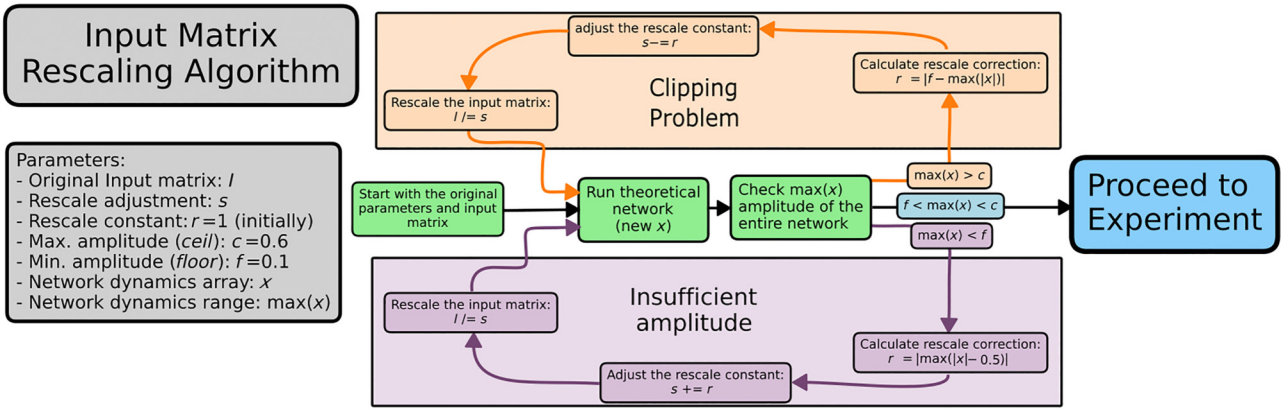

FIG. 7. Flowchart of the input matrix rescaling algorithm. This iterative procedure optimizes the dynamic range usage of the analog implementation, avoiding both clipping and machine-precision artifacts.

We next examine whether decoding performance is consistent across all digit classes or if certain digits are decoded with superior or inferior accuracy (Fig. 9). The analog twin performs best for the digit classes 1, 4, 6, and 8.

APPENDIX D: SVM DECODING

An alternative method for assessing the networks' performance involves utilizing the analog HORN as a reservoir. In this approach, an independent readout (linear SVM) is trained on the output of a pretrained network while keeping the network parameters fixed. The linear SVM was implemented using the Scikit-learn PYTHON package [58].

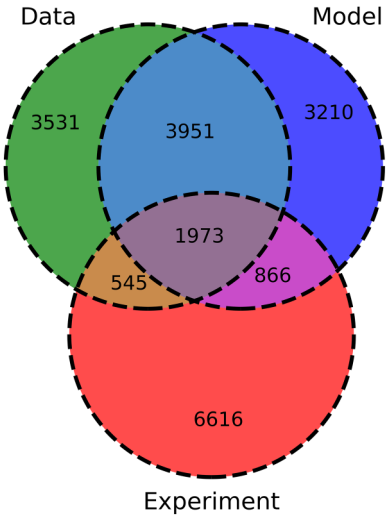

FIG. 8. Prediction agreement analysis using Venn diagram representation. Overlap regions show the intersection between predictions from the analog implementation, digital twin, and true MNIST labels across the test dataset.

APPENDIX E: VELOCITY-COUPLED NETWORK

Here, we present the obtained results for the analysis of the physical harmonic oscillator recurrent network with its nodes velocity-coupled. See Fig. 10 for the circuit diagram.

In this network, the nodes are coupled through their velocity terms. This coupling leads to a modified form of the update equations [Eq. (3)], expressed as follows:

$$x_{i,t+1} = x_{i,t} + h y_{i,t+1},$$
$$y_{i,t+1} = y_{i,t} + h\left[\sum_{j\neq i}^{n}\left(W_{ji}y_{i,t}\right) + I_i s(t) - 2\gamma y_{i,t} - \omega_i^2 x_{i,t}\right]. \tag{E1}$$

For the readout, we used a "Hilbert decoder." The Hilbert decoder computes the analytic signal by applying the discrete Hilbert transform to each node's amplitude time series $x_i(t)$ to obtain the complex-valued analytic signal $\phi_i(t)$ on a time window δT (default window length 100 time steps) around the readout time t^*. We then construct an eight-dimensional feature vector by splitting $\phi_i(t^*)$ into its real and complex parts. This feature vector is read out by an affine readout layer as previously to perform a digit class prediction. This readout exploits phase information that is directly available in the velocity-coupled network.

Once implemented, we can visualize some sample network runs (Fig. 11). We then proceed with the analysis as in the main text. First, we look at the performance of the network (Fig. 12). Similar to the amplitude-coupled case, we find that when used as a reservoir, the network performs better. However, the performance of the velocity-coupled network is better than that of the amplitude-coupled network.

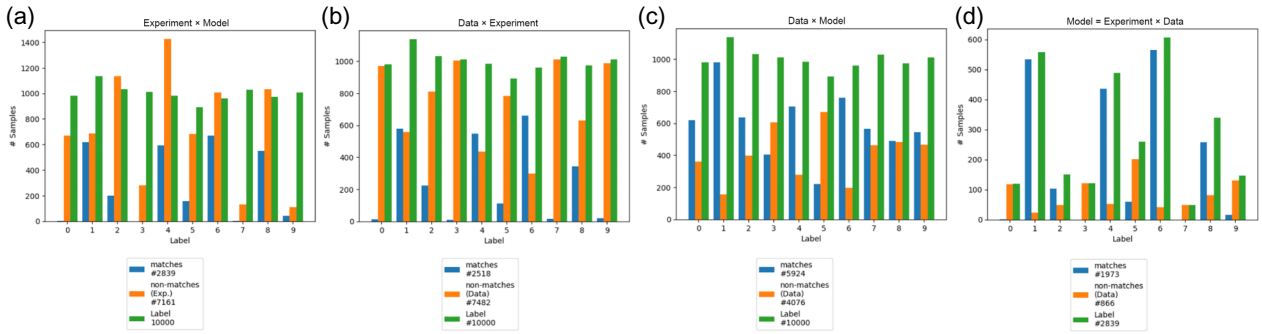

FIG. 9. Digit-wise classification performance comparison across different evaluation scenarios. (a) Analog implementation performance versus true dataset labels. (b) Digital-twin performance versus true dataset labels. (c) Agreement between analog implementation and digital-twin predictions. (d) Accuracy of the analog-digital agreement subset when compared to true labels, showing how well cases where both systems agree actually correspond to correct classifications.

A Venn diagram was computed using the same procedure as in Fig. 8. The velocity-coupled network achieves a better reproduction of the label of its digital twin (58.9%, Fig. 13). The error metrics analysis (Fig. 14) shows that the velocity-coupled network exhibits smaller errors than the amplitude-coupled network [Fig. 4(d)].

Finally, the velocity-coupled network was used as a reservoir with an SVM trained to decode the network dynamics (Fig. 15). The SVM was trained using the variables x and y, so that the readout has the same dimensionality as the Hilbert decoder ("Exp. full" and "Model full" curves in Fig. 15). The SVM was also trained using only the x variable ("Exp." and "Model" curves in Fig. 15) for comparison with the amplitude-coupled network. Similarly to the amplitude-coupled network, the SVM readout enables the analog and digital networks to achieve

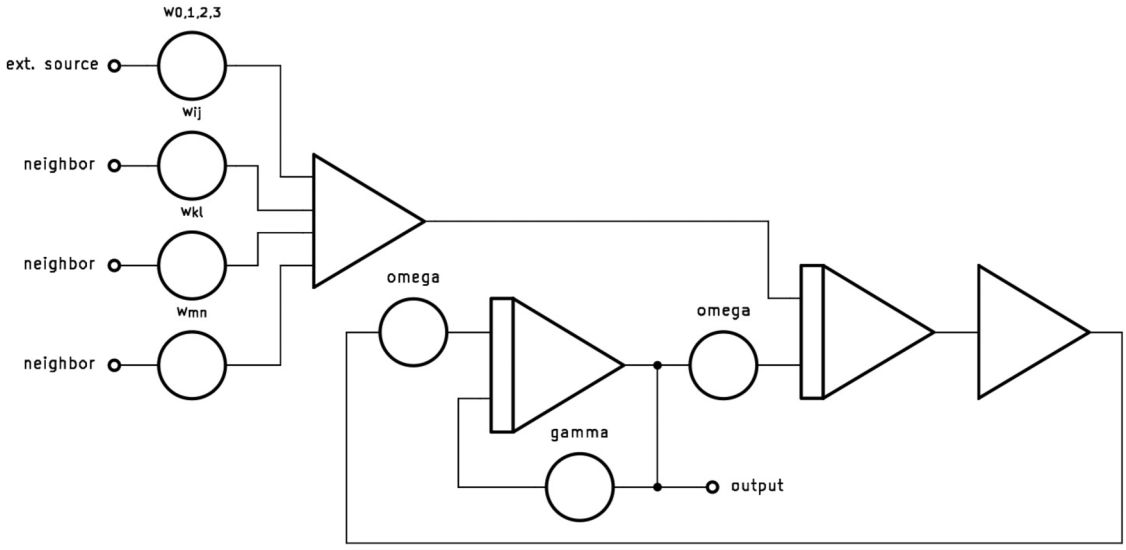

FIG. 10. Circuit diagram for the oscillatory unit of the velocity-coupled network.

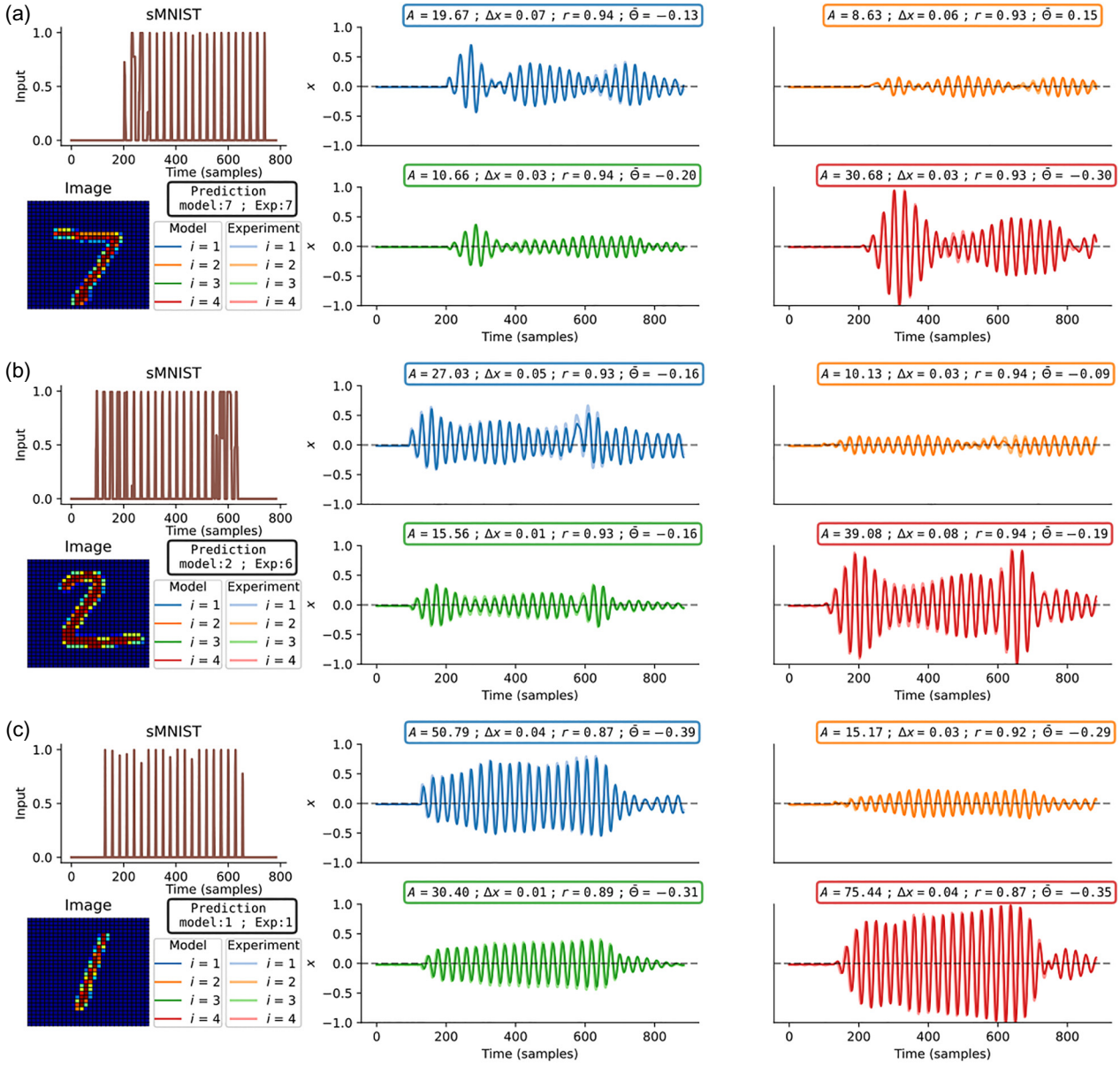

FIG. 11. Representative dynamics samples from runs of velocity-coupled network. The left-hand column displays input in both sMNIST (serialized) and MNIST (image) formats. The central and right-hand columns show the corresponding node dynamics, with light traces representing analog implementation and dark traces representing digital-twin dynamics.

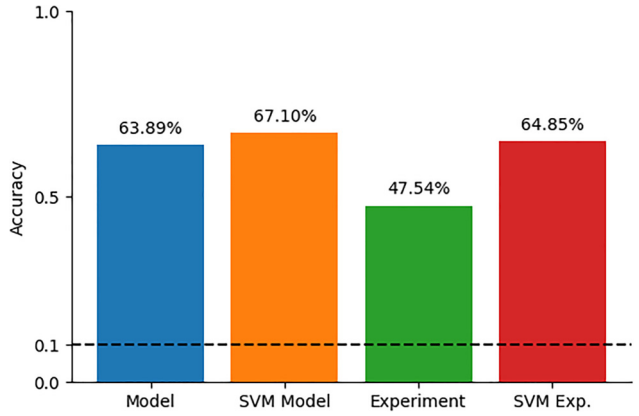

FIG. 12. Classification performance comparison for velocity-coupled HORN implementation, showing the accuracy metrics for different readout strategies applied to the velocity-coupled network variant.

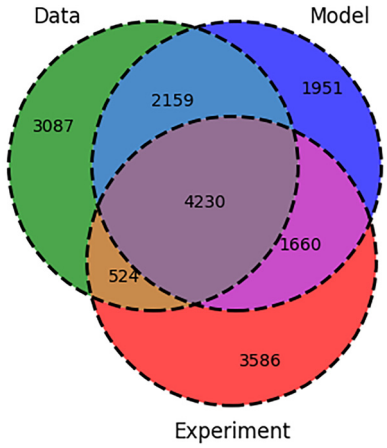

FIG. 13. Prediction agreement analysis for velocity-coupled network implementation. Venn diagram showing overlaps between analog implementation, digital twin, and true label predictions.

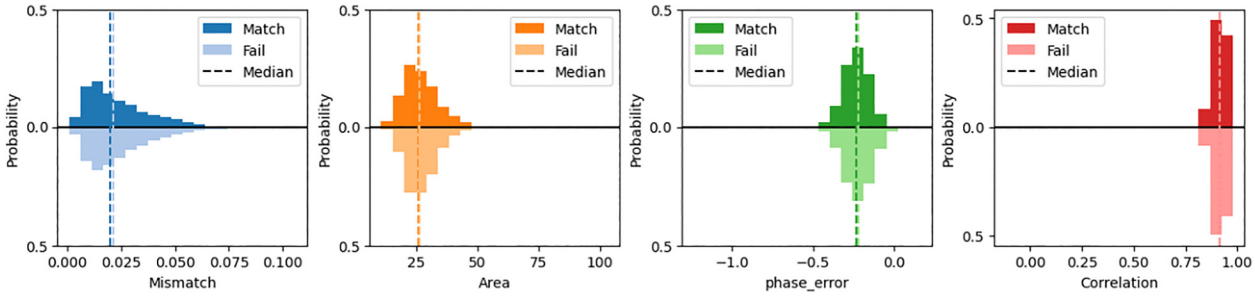

FIG. 14. Error metric distributions for velocity-coupled network. Comparison of error metrics between analog implementation and digital twin for the velocity-coupled variant.

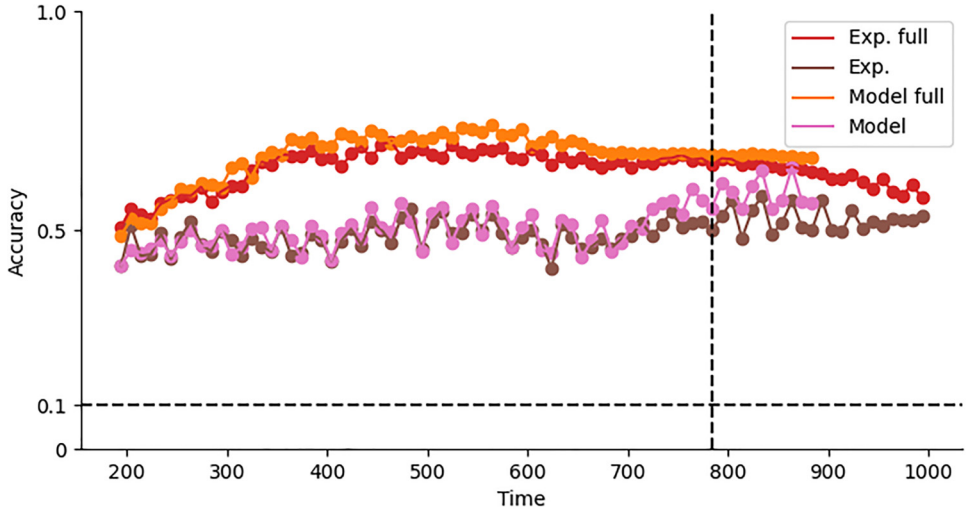

FIG. 15. SVM readout performance analysis for velocity-coupled network. Temporal evolution of classification accuracy using both amplitude and velocity variables for decoding.

comparable performance (Fig. 15), demonstrating that the analog implementation preserves the information of its digital twin.

[1] 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).

[2] D. Marković, A. Mizrahi, D. Querlioz, and J. Grollier, Physics for neuromorphic computing, Nat. Rev. Phys. **2**, 499 (2020).

[3] W. Yu, H. Guo, J. Xiao, and J. Shen, Physical neural networks with self-learning capabilities, Sci. China Phys. Mech. Astron. **67**, 287501 (2024).

[4] C. D. Schuman, S. R. Kulkarni, M. Parsa, J. P. Mitchell, P. Date, and B. Kay, Opportunities for neuromorphic computing algorithms and applications, Nat. Comput. Sci. **2**, 10 (2022).

[5] T. Dalgaty, F. Moro, Y. Demirağ, A. De Pra, G. Indiveri, E. Vianello, and M. Payvand, Mosaic: In-memory computing and routing for small-world spike-based neuromorphic systems, Nat. Commun. **15**, 142 (2024).

[6] M. Aifer, Z. Belateche, S. Bramhavar, K. Y. Camsari, P. J. Coles, G. Crooks, D. J. Durian, A. J. Liu, A. Marchenkova, A. J. Martinez, P. L. McMahon, F. Sbahi, B. Weiner, and L. G. Wright, Solving the compute crisis with physics-based ASICs, ArXiv:2507.10463.

[7] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, A million spiking-neuron integrated circuit with a scalable communication network and interface, Science **345**, 668 (2014).

[8] Y. Zhong, J. Tang, X. Li, X. Liang, Z. Liu, Y. Li, Y. Xi, P. Yao, Z. Hao, B. Gao, H. Qian, and H. Wu, A memristor-based analogue reservoir computing system for real-time and power-efficient signal processing, Nat. Electron. **5**, 672 (2022).

[9] K. Vandoorne, P. Mechet, T. Van Vaerenbergh, M. Fiers, G. Morthier, D. Verstraeten, B. Schrauwen, J. Dambre, and P. Bienstman, Experimental demonstration of reservoir computing on a silicon photonics chip, Nat. Commun. **5**, 3541 (2014).

[10] H. Cılasun, W. Moy, Z. Zeng, T. Islam, H. Lo, A. Vanasse, M. Tan, M. Anees, R. S, A. Kumar, S. S. Sapatnekar, C. H. Kim, and U. R. Karpuzcu, A coupled-oscillator-based Ising chip for combinatorial optimization, Nat. Electron. **8**, 537 (2025).

[11] S. Abreu, *et al.*, A photonics perspective on computing with physical substrates, Rev. Phys. **12**, 100093 (2024).

[12] G. Wetzstein, A. Ozcan, S. Gigan, S. Fan, D. Englund, M. Soljačić, C. Denz, D. A. B. Miller, and D. Psaltis, Inference in artificial intelligence with deep optics and photonics, Nature **588**, 39 (2020).

[13] M. Romera, P. Talatchian, S. Tsunegi, F. Abreu Araujo, V. Cros, P. Bortolotti, J. Trastoy, K. Yakushiji, A. Fukushima, H. Kubota, S. Yuasa, M. Ernoult, D. Vodenicarevic, T. Hirtzlin, N. Locatelli, D. Querlioz, and J. Grollier, Vowel recognition with four coupled spin-torque nano-oscillators, Nature **563**, 230 (2018).

[14] Y. Chen, M. Nazhamaiti, H. Xu, Y. Meng, T. Zhou, G. Li, J. Fan, Q. Wei, J. Wu, F. Qiao, L. Fang, and Q. Dai, All-analog photoelectronic chip for high-speed vision tasks, Nature **623**, 48 (2023).

[15] R. H. Lee, E. A. B. Mulder, and J. B. Hopkins, Mechanical neural networks: Architected materials that learn behaviors, Sci. Robot. **7**, eabq7278 (2022).

[16] A. Momeni, B. Rahmani, M. Malléjac, P. del Hougne, and R. Fleury, Backpropagation-free training of deep physical neural networks, Science **382**, 1297 (2023).

[17] M. Stern, D. Hexner, J. W. Rocks, and A. J. Liu, Supervised learning in physical networks: From machine learning to learning machines, Phys. Rev. X **11**, 021045 (2021).

[18] A. Momeni, *et al.*, Training of physical neural networks, ArXiv:2406.03372.

[19] A. Todri-Sanial, C. Delacour, M. Abernot, and F. Sabo, Computing with oscillators from theoretical underpinnings to applications and demonstrators, npj Unconv. Comput. **1**, 14 (2024).

[20] T. K. Rusch and S. Mishra, Coupled oscillatory recurrent neural network (coRNN): An accurate and (gradient) stable architecture for learning long time dependencies, in *International Conference on Learning Representations* (2021), https://openreview.net/forum?id=F3s69XzWOia.

[21] F. Effenberger, P. Carvalho, I. Dubinin, and W. Singer, The functional role of oscillatory dynamics in neocortical circuits: A computational perspective, Proc. Natl. Acad. Sci. **122**, e2412830122 (2025).

[22] P. Fries, Rhythms for cognition: Communication through coherence, Neuron **88**, 220 (2015).

[23] M. Rosenblum, A. Pikovsky, J. Kurths, C. Schäfer, and P. A. Tass, Chapter 9 Phase synchronization: From theory to data analysis, in *Neuro-Informatics and Neural Modelling*, edited by F. Moss and S. Gielen, Handbook of Biological Physics (North-Holland, 2001), Vol. 4, pp. 279–321.

[24] T. W. Hughes, I. A. D. Williamson, M. Minkov, and S. Fan, Wave physics as an analog recurrent neural network, Sci. Adv. **5**, eaay6946 (2019).

[25] G. Buzsáki and X.-J. Wang, Mechanisms of gamma oscillations, Annu. Rev. Neurosci. **35**, 203 (2012).

[26] A. V. Goltsev, M. A. Lopes, K.-E. Lee, and J. F. F. Mendes, Critical and resonance phenomena in neural networks, AIP Conf. Proc. **1510**, 28 (2013).

[27] I. Dubinin and F. Effenberger, Fading memory as inductive bias in residual recurrent networks, Neural Netw. **173**, 106179 (2024).

[28] P. Ashwin and C. Postlethwaite, Excitable networks for finite state computation with continuous time recurrent neural networks, Biol. Cybern. **115**, 519 (2021).

[29] D. Nikolić, S. Häusler, W. Singer, W. Maass, and J. D. Victor, Distributed fading memory for stimulus properties in the primary visual cortex, PLoS Biol. **7**, e1000260 (2009).

[30] K. B. Doelling and D. Poeppel, Cortical entrainment to music and its modulation by expertise, Proc. Natl. Acad. Sci. **112**, E6233 (2015).

[31] R. Schmidt, J. Rose, and V. Muralidharan, Transient oscillations as computations for cognition: Analysis,

modeling and function, Curr. Opin. Neurobiol. **83**, 102796 (2023).
[32] A. Palmigiano, T. Geisel, F. Wolf, and D. Battaglia, Flexible information routing by transient synchrony, Nat. Neurosci. **20**, 1014 (2017).
[33] R. F. Helfrich, I. C. Fiebelkorn, S. M. Szczepanski, J. J. Lin, J. Parvizi, R. T. Knight, and S. Kastner, Neural mechanisms of sustained attention are rhythmic, Neuron **99**, 854 (2018).
[34] G. Csaba and W. Porod, Coupled oscillators for computing: A review and perspective, Appl. Phys. Rev. **7**, 011302 (2020).
[35] B. Ulmann, anabrid Model-1 analog computer user manual, http://analogparadigm.com/downloads/handbook.pdf.
[36] H. Chen, *et al.*, Continuous-time digital twin with analog memristive neural ordinary differential equation solver, Sci. Adv. **11**, eadr7571 (2025).
[37] T. Rudner, W. Porod, and G. Csaba, Design of oscillatory neural networks by machine learning, Front. Neurosci. **18**, 1307525 (2024).
[38] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE **86**, 2278 (1998).
[39] 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).
[40] C. Du, F. Cai, M. A. Zidan, W. Ma, S. H. Lee, and W. D. Lu, Reservoir computing using dynamic memristors for temporal information processing, Nat. Commun. **8**, 2204 (2017).
[41] F. K. Kneubühl, *Oscillations and Waves* (Springer, Berlin, Heidelberg, 1997).
[42] E. Hairer, C. Lubich, and G. Wanner, Geometric numerical integration illustrated by the Störmer–Verlet method, Acta Numer. **12**, 399 (2003).
[43] M. Baronig, R. Ferrand, S. Sabathiel, and R. Legenstein, Advancing spatio-temporal processing through adaptation in spiking neural networks, Nat. Commun. **16**, 5776 (2025).
[44] A. Paszke, *et al.*, PyTorch: An Imperative Style, High-Performance Deep Learning Library, in *Advances in Neural Information Processing Systems*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. D. Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc., 2019), Vol. 32, Vol. 32, https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.
[45] ADALM2000 Website, https://www.analog.com/en/resources/evaluation-hardware-and-software/evaluation-boards-kits/adalm2000.html.
[46] B. Ulmann and S. Köppel, TeensyLogger: Open-source analog data logger based on Teensy, 2025. https://github.com/anabrid/TeensyLogger
[47] B. Ulmann, *Analog Computing* (De Gruyter Oldenbourg, Berlin, Boston, 2022).
[48] B. Ulmann, *Analog and Hybrid Computer Programming* (De Gruyter Oldenbourg, Berlin, Boston, 2020).
[49] A. Ohkubo and M. Inubushi, Reservoir computing with generalized readout based on generalized synchronization, Sci. Rep. **14**, 30918 (2024).
[50] J. D. Kendall, R. D. Pantone, K. Manickavasagam, Y. Bengio, and B. Scellier, Training end-to-end analog neural networks with equilibrium propagation, ArXiv:2006.01981.
[51] M. Satyanarayanan, The emergence of edge computing, Computer **50**, 30 (2017).
[52] D. J. Amit, *Modeling Brain Function: The World of Attractor Neural Networks* (Cambridge University Press, Cambridge, 1989).
[53] G. Indiveri and S.-C. Liu, Memory and information processing in neuromorphic systems, Proc. IEEE **103**, 1379 (2015).
[54] D. Kudithipudi, *et al.*, Neuromorphic computing at scale, Nature **637**, 801 (2025).
[55] S. Higuchi, S. Kairat, S. Bohté, and S. Otte, Balanced Resonate-and-Fire Neurons, in *Proceedings of the 41st International Conference on Machine Learning*, edited by R. Salakhutdinov, Z. Kolter, K. Heller, A. Weller, N. Oliver, J. Scarlett, and F. Berkenkamp, Proceedings of Machine Learning Research (PMLR, 2024), Vol. 235, pp. 18305–18323, https://proceedings.mlr.press/v235/higuchi24a.html.
[56] P. Carvalho, Digital twin data, Figshare https://doi.org/10.6084/m9.figshare.29959970 (2025).
[57] P. Carvalho, Analog twin data, Figshare https://doi.org/10.6084/m9.figshare.29959967 (2025).
[58] 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).