Simple Reservoir Computing Capitalizing on the Nonlinear Response of Materials: Theory and Physical Implementations

Shaohua Kan,[1] Kohei Nakajima,[2,3] Yuki Takeshima,[4] Tetsuya Asai,[1] Yuji Kuwahara,[4] and Megumi Akai-Kasaya[1,4,*]

[1] *Graduate School of Information Science and Technology, Hokkaido University, Kita 14, Nishi 9, Kita-ku, Sapporo, Hokkaido, 060-0814, Japan*

[2] *Graduate School of Information Science and Technology, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan*

[3] *AI Center, The University of Tokyo, 7-3-1, Hongo, Bunkyo-ku, Tokyo 113-8656, Japan*

[4] *Graduate School of Engineering, Osaka University, 2-1 Yamadaoka, Suita, Osaka 565- 0871, Japan*

(Received 26 June 2020; revised 21 December 2020; accepted 13 January 2021; published 12 February 2021)

The potential of nonlinear dynamical systems serving as reservoirs has attracted much attention for the physical realization of reservoir computing (RC). Here, we propose a hardware system working as a reservoir with one simple form of nonlinearity that reflects the intrinsic characteristics of the materials. We show that insufficient dynamics in such physical systems can perform like complex dynamical systems with the assistance of external controls. Based on the idea of spatial multiplexing, this dynamical system is studied under two frameworks. The correlation between structural adjustments of the reservoir and system performance in processing various types of task is proposed. Our results are expected to enable the development of material-based devices for RC.

DOI: 10.1103/PhysRevApplied.15.024030

I. INTRODUCTION

Artificial neural networks (ANNs) have been actively developed as the foundation of artificial intelligence (AI) systems. ANNs can be grouped according to their architecture into feedforward [1] and recurrent networks [2], making them suitable for various tasks. Reservoir computing (RC), a framework derived from recurrent neural-network theory, has generated significant interest, owing to its many advantages, such as the relatively small computational burden of the training process, lack of a fading-memory problem, and relatively simple physical implementation. Generally, there are three parts in a RC implementation: an input layer to feed the input signal to the reservoir, a reservoir to process input data, and an output layer to read out the reservoir states and use them for training [3]. Through RC, input data are mapped into a high-dimensional space to facilitate the separation of states [4]. Generally, the output signal, y, at time t can be extracted from a linear combination with adjustable output weights, W^{out}, coupled to reservoir states x at time t, expressed as

$$\begin{aligned} y(t) &= f_{\text{out}}[x(t)], \\ f_{\text{out}}(x) &= W^{\text{out}}x(t). \end{aligned} \tag{1}$$

Different from recurrent neural networks, the RC input and reservoir weights do not need to be adjusted. Hence, the training process can be simplified to the training of output weights, such that it does not affect the node states. Various reservoir-model architectures are proposed, such as multilayer [5,6], parallel time delay [7], single-node structures with time multiplexing [8], and growing echo-state networks (ESNs) [9]. A physical RC scheme (Fig. 1) was recently proposed [10], in which the reservoir part could be any physical dynamical system serving as a computational resource instead of a conventional recurrent network. This concept enables the development of physical reservoir implementation, providing promising candidates for next-generation AI paradigms [11]. Typical examples of dynamical systems serving as the reservoir (Fig. 1) include dynamical memristors [12], neuromorphic materials [13], and a soft body [14].

Criteria for the normal operation of physical dynamical systems as reservoirs are studied [10]. One key criterion is that the system should generate different responses with trends similar to a given input signal, as shown by the different system states in Fig. 1. The nonlinear response of a signal with its complex network or individual but relational dynamical response generates the required dynamics for a RC. Such a large number of product sums of diverse response signals improves the RC performance. Current research on physical RCs relies heavily on software to

*akai@ist.hokudai.ac.jp

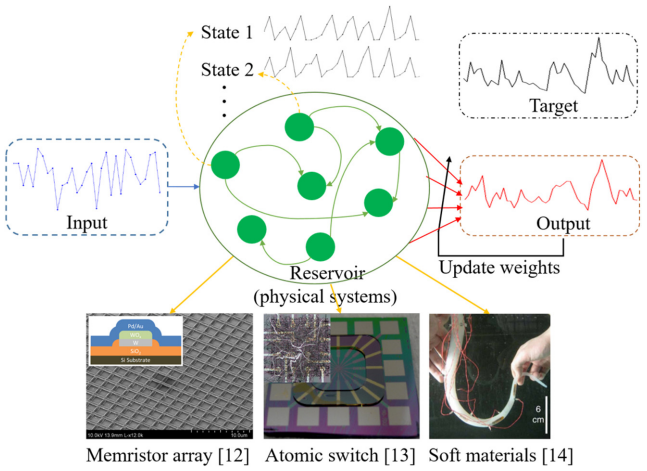

FIG. 1. Concept of a physical RC. Reservoir can be replaced with any physical system that responds differently to a given input signal. Output signal is the linear combination of all system states coupled to adjustable output weights. There can be multiple output signals, depending on the specific task.

replicate nodes [8,15], imposing a burden on performance. Reductions in signal conversion load and memory consumption are required for hardware RCs.

The nonlinearity of the signal response of nanomaterials is proposed for information processing, where their extremely small size allows the creation of a highly dense and complex system for future neuromorphic information processors [16,17]. Because the conductance of a nanomaterial is sensitive to an electrical potential difference, the current-voltage (I-V) characteristics can be used to distinguish nonlinearity. Various nonlinear responses, such as temperature-independent tunneling current through a molecular monolayer that functionalizes nanoparticles [18] and stochastic switching via ion movement dynamics [19], are reported for such systems.

However, nanomaterials with a significant nonlinear conversion capacity do not always show sufficient dynamics, and they are not easy to control or observe. Therefore, to examine the availability of a simple nonlinear I-V response of materials [17,18,20,21], as well as its usage in RC systems, we construct an electronic circuit working as the nonlinear node and introduce an external control to generate a dynamical response. In this study, we use a single physical node constructed from an electronic circuit, and a large number of virtual nodes are generated to form a reservoir by time and spatial multiplexing.

Compared with other physical reservoir schemes, the dynamics in our physical systems are more controllable and simpler. The much lower external memory consumption and signal processing complexity of our control allow the construction of a parallel process for multiple devices, so that a future information processing device can be developed utilizing highly integrated nanomaterials. We investigate the feasibility and effectiveness of the proposed scheme by testing its performance in various tasks and thereby gain insights into the physical realization of a RC. The broader aim of this work is to an alternative RC system, so that appropriate nanomaterials can be used for building them.

II. DESIGN SCHEME AND TESTING METHOD

In this section, we provide an overview of our design scheme to clarify the concept of our physical reservoir. The basic principle of our scheme is to build a dynamical system based on the I-V curves of materials with a simple external control. Two parameters (i.e., input gain, e, and feedback gain, a) are introduced to increase the dynamics and represent individual processing nodes in the reservoir. The I-V curve-based model is defined by the following functions:

$$V_{\text{in}}(t) = aV_{\text{out}}(t-1) + eu_t, \tag{2}$$

$$V_{\text{out}}(t) = \begin{matrix} V_{\text{in}}(t) + V_{\text{th}}[V_{\text{in}}(t) \le -V_{\text{th}}], \\ 0[-V_{\text{th}} < V_{\text{in}}(t) < V_{\text{th}}], \\ V_{\text{in}}(t) - V_{\text{th}}[V_{\text{in}}(t) \ge V_{\text{th}}], \end{matrix} \tag{3}$$

where u_t (random sequences within $[-5,5]$) are the input signals, and V_{th} is the threshold voltage. t represents the sequential number. Figure 2 shows the simulated results of this model, where three I-V curves are generated from three values of e and a, and the value of V_{th} is fixed. In this model, the input gain, e, determines the slope of the curve, and the feedback gain, a, affects the distribution around the curve.

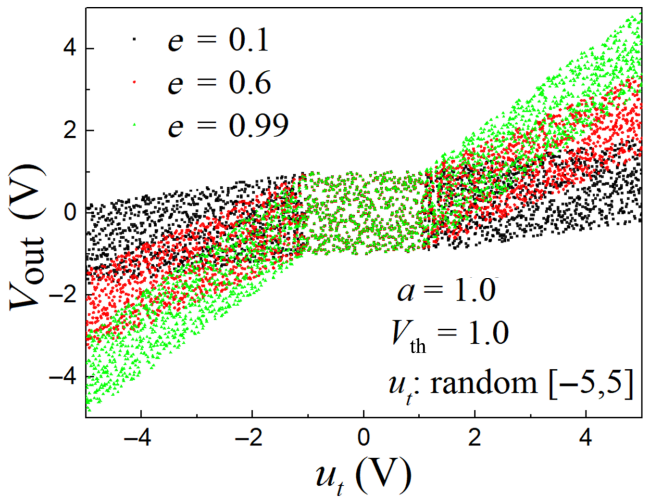

FIG. 2. Response function settings. V_{th} is the threshold voltage and u_t is the random input signal. I-V curves in black, red, and green represent different values of e, which are reflected in the different slopes. Larger e corresponds to a steeper slope. All three curves have the same shape because they have the same value of a.

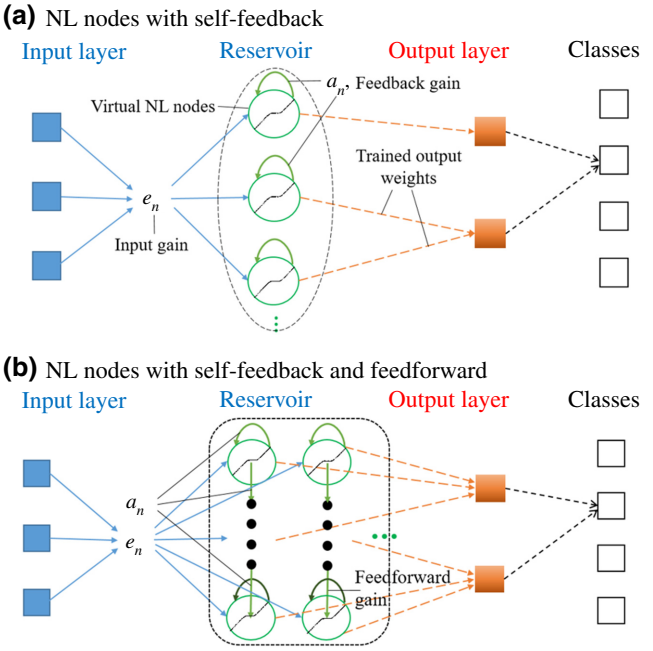

FIG. 3. Diagram of RC scheme. (a) Parallel-node structure. Nodes in this scheme are parallel to each other and independent. Different nodes are defined by different value pairs of a and e. Each node also has self-feedback. Nodes are independent. (b) Parallel-group structure. Nodes are divided into some groups, which are parallel. Six nodes in each group have feedforward from the previous node.

Next, we propose a RC system that offers tunable internal nonlinearity for each node. The nonlinearity needed is shown for each node in Fig. 3. Figure 3(a) displays a diagram of our first scheme: parallel node. Nodes are assigned with different values of input gain, e, and feedback gain, a. Hence, the nth node is defined by e^n and a^n. This separately processes the input stream, $u(t)$. The tth input fed into the nth node can be read as

$$V_{\text{in}}^n(t) = a^n V_{\text{out}}^n(t-1) + e^n u(t). \tag{4}$$

The transient-node state, V_{out}^n, which characterizes the transient response to a certain input, can be expressed as

$$V_{\text{out}}^n(t) = f_{\text{out}}^n[V_{\text{in}}(t)], \tag{5}$$

where f^n denotes the nonlinear transition function of the nth node in the reservoir. During the nth-node process, each input is fed into the node state by coupling it to the same input gain, e^n, and the past-node state is fed by coupling it to the same feedback gain, a^n. The output layer reads out all of these transient states, in which the final output can be obtained as the linear summation of the states of all nodes weighted by an output weight. The output weights are optimized using a training procedure to best fit the output signal to the target signal.

In our second scheme (i.e., parallel group), another external parameter, feedforward gain, is introduced to provide connections between nodes in the reservoir. The node connection structure inside the reservoir is shown in Fig. 3(b), where the nodes are divided into groups of six. These groups are parallel, and we call it the *parallel-group structure*. In each group, e^n and a^n serve as input and feedback gains for the first node, respectively. For the remaining nodes (2–6), a^n serves as the feedforward gain coupled to the output from the previous node at the same time sequence, and λ serves as the feedback gain. The input voltage in each group can be formulated as

$$\begin{aligned} V_{\text{in}}^{6n+1}(t) &= a^{6n+1} V_{\text{out}}^{6n+1}(t-1) \\ &\quad + e^{6n+1} u(t) \quad \text{(first node)}, \\ V_{\text{in}}^{6n+x}(t) &= a^{6n+x} V_{\text{out}}^{6n+x-1}(t) + e^{6n+x} u(t) \\ &\quad + \lambda V_{\text{out}}^{6n+x}(t-1) \quad \text{(rest nodes)}. \end{aligned} \tag{6}$$

Although the input stream is processed sequentially, the central idea of our design includes a solution based on a parallel implementation. Such an implementation derives from a spatial multiplexing research model [22]. Both implementations are related to the ESN framework [23] and satisfy the echo-state property. This system is rather simple and lacks complex nonlinear dynamics, allowing observation of the effects of parameter tuning and architectural changes. Our aim is to verify the effectiveness of such simple dynamics (i.e., single nonlinear response and independent feedback) during information processing. Furthermore, such a parallel framework requires minimum external memory because it needs to store only the previous state. The required external memory will not swell under such parallel processing.

Next, we realize the physical RC system with tunable nonlinearity of a hardware system. Considering the similarity of the I-V characteristic of diodes to the nonlinear response of materials and its easier implementation, we construct our physical system using an electronic circuit in which the nonlinear response is realized with two antiparallel diodes (1S1585). This part is called the physical nonlinear (NL) node. The output-to-input relation we need is plotted in the top panel of Fig. 4, with an input voltage range $[-2.1, 2.1]$, and thus, an output voltage range $[-1.5, 1.5]$. The input range is defined between -2.1 and 2.1 V due to the limitation of our hardware system (i.e., DAC PCF8591), but some inputs will go beyond the range with the effects of external control. To enable the node to return to the normal range, we use a regression setting in our hardware system, that is, if the input voltage exceeds the defined range, the input voltage will return to zero; if not, the input is its original value. The entire procedure of information processing is shown in Fig. 4 under the control and monitoring of the Ardunio Mega 2560 development board.

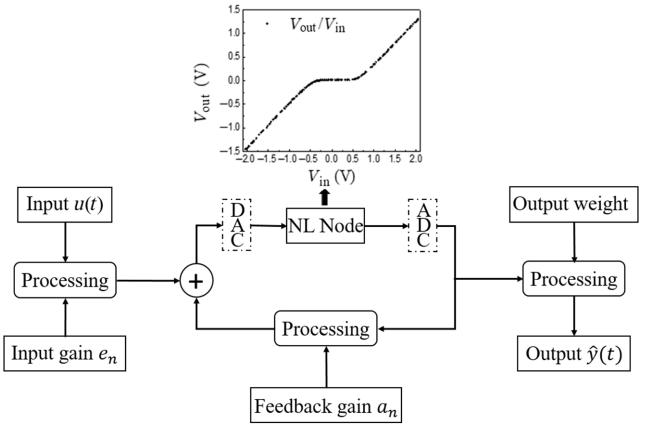

FIG. 4. Schematic of physical realization. NL node with output-to-input characteristics generates N virtual nodes via time multiplexing, so that the entire input signal is processed N times. Each input data item passing a node is coupled with the same input gain. Next, it is fed into the node with the previous node state coupled with the same feedback gain. Inputs are transformed using a digital-to-analog converter (DAC) when fed into the NL node and by an analog-to-digital converter (ADC) to obtain output voltage from nodes. Training process is then conducted using all collected digital data.

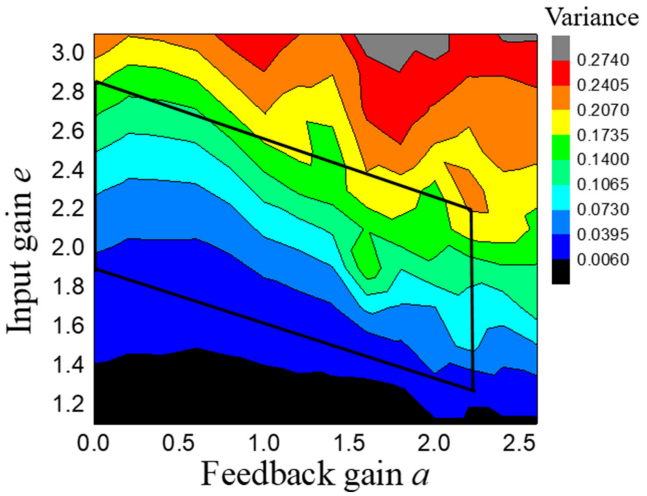

FIG. 5. Variance mapping. We calculate 14×11 groups of variance of node states for this mapping graph; 14 because parameter a ranges from 0 to 2.6 in increments of 0.2 and 11 because parameter e ranges from 1.1 to 3.1. After around 30 tests under different ranges in the NARMA2 task, we determine the parameter ranges that give the best performance (marked with black parallelogram).

To save space, we draw on the concept of time multiplexing [8], so that only one single NL node is used to create our RC system. We define N individual virtual nodes by generating N value pairs of a and e, which means that the input signal is processed N times through the physical NL node, whereas the e and a values are updated each time. Thus, a single physical NL node serves as different virtual nodes at different times. Owing to implementation under one single physical node, compared with the parallel-node scheme, the parallel-group scheme requires external memory for all output states from the ante nodes, $V_{\text{out}}^{6n+x}(t-1)$, to create the connection mentioned in Eq. (6).

Because the input and feedback gains have a direct impact on the processing results, an optimization of their values is necessary. Even if a and e are randomly selected, an optimal range should be determined. We leverage a mapping method that simultaneously displays the changes in dynamics along both e and a. Parameter a ranges from 0 to 2.6, and parameter e ranges from 1.1 to 3.1, both having a step of 0.2. For each pair of e and a, we feed an input signal of 300 sequences randomly generated from -0.5 to 0.5 to our reservoir and calculate the variances of the node states (300 output voltage values). A variance mapping graph (Fig. 5) is then plotted for various parameter values. A greater variance indicates richer dynamic characteristics of the nodes because the output voltage distribution is more scattered. Based on the distribution of variance, different ranges (i.e., shapes, colors, and their combinations) are selected for testing in the benchmark nonlinear autoregressive moving average (NARMA2) task, introduced in Sec. III, to help us adjust the range selection. After 30 tests under different ranges, we determine the parameter ranges that give the best performance, as marked with a black parallelogram in Fig. 5.

This method determines the values of input gain e and feedback gain a under the parallel-node scheme. In the parallel-group structure, parameters a and e have the same ranges as those determined for the parallel-node structure. Parameter λ is 0.05, 0.1, 0.12, 0.12, and 0.12 for the remaining five nodes in each group. Gradually increasing the feedback parameter λ is helpful for remembering previous input information. These values are selected according to several tests under the benchmark NARMA2 task. Owing to the limitation of physical realization for finding the precise range, we conduct limited groups of tests to determine all parameters. However, we believe that the ranges can be further improved.

III. PERFORMANCE ANALYSIS

After parameter ranges are determined, we test the performance of our diode-based physical system. The results of the following tasks become experimental data from our hardware reservoir. The values of the parameters in all tests, if not specified, are taken from the parameter ranges discussed in Sec. II.

A. Features of output of node state

We first test the output features of our physical RC to the input signal, $u(t)$, comprising sequential random data in the interval [0, 0.5]. This range is used for the input

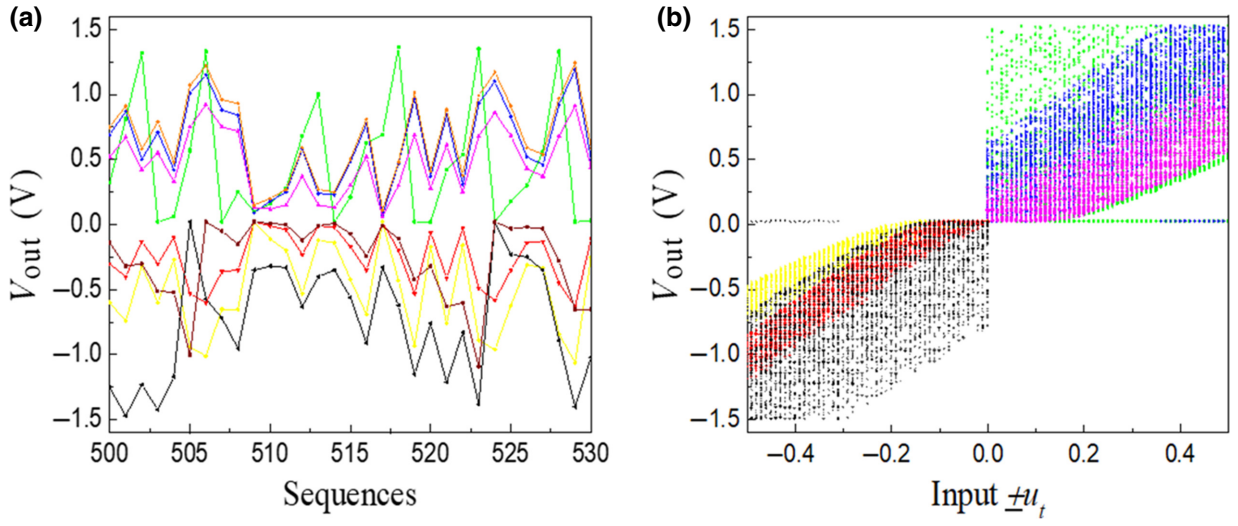

FIG. 6. Output features. (a) Sequential node states. This graph plots 10 node states out of 300 nodes along the time sequences from our physical reservoir based on the scheme in Fig. 3(a). Input signal to this reservoir includes 4000 random data items, $u(t)$, in the interval [0, 0.5]. Output data are divided into either all positive or all negative based on the sign of the input gain, e. (b) Typical input-to-output characteristics. Outputs of six nodes selected from 300 nodes. Negative part, caused by a negative sign of input gain e, is supposed to be on the right side of the x axis, because $u(t)$ is in the interval [0, 0.5]. However, it is flipped along the vertical axis for a clear I-V curve.

signal because it is a common range used in NARMA tasks, although it goes against our design principles, which state that both positive and negative values should be used, as in Fig. 2. To use both positive and negative sides of this nonlinear response, we randomly set some of the e values to be negative.

Figure 6(a) displays the sequential states of eight selected nodes from 300. All states having the same abscissa use the same input value, but each node responds differently. Figure 6(b) shows the relation between output and input, which is similar to that of the I-V curves (Fig. 2), where the slopes in both the graphs reflect feedback gain a. However, unlike Fig. 2, the input gain, e, is not fixed in this case.

B. Results of NARMA2 task

Here, we report the performance of our physical reservoir on the NARMA task [24], which is a widely used and effective benchmark for RC, when evaluating the reservoir's ability to duplicate a higher-order dynamical model constructed from current and previous inputs. Generally, the NARMA2 and NARMA10 tasks require different optimal parameter settings [8,25]. NARMA10, as a higher-order dynamical system, is a fairly difficult target to train for. Hence, we use the NARMA2 task as our benchmark. The parameters in this paper are set by observing the performance for the NARMA2 task. NARMA2 is a second-order nonlinear model defined as

$$y(t+1) = 0.4y(t) + 0.4y(t)y(t-1) + 0.6u(t)^3 + 0.1. \tag{7}$$

The input signal, $u(t)$, is a sequence of random digits in the interval [0, 0.5] generated by a random function, and the model output, $y(t)$, serves as the target. We train the system's output signal by selecting the output weights to render the signal as close as possible to the target one. Output weights, W_{out}, are determined from training data using ridge regression. The predicted signal is calculated as $\hat{y}(t) = X_{\text{test}} W_{\text{out}}$ Details of the training and testing process are given in the Appendix.

We introduce the normalized-mean-square error (NMSE) to quantify the deviation between the target and predicted signals, expressed as

$$\Delta E = \frac{1}{N} \frac{\sum_t [\hat{y}(t) - y(t)]^2}{\sigma^2(y)}. \tag{8}$$

A total of 300 nodes are used for all schemes to perform this task. Figure 7 shows the testing results for the parallel-node and -group structure reservoirs. Black lines reflect the NARMA2 model serving as the target signal in this prediction task. We train the output weights during the training step and use them to calculate the prediction signal from the testing data (shown in red). Green lines in the inset are the testing results for the standard ESN (spectral radius: 0.99; input weight interval: [−1,1]; activation function: tanh function); see the settings in the Appendix. Performance is reflected by the reproducibility of the prediction to the target. To assess the performance, the ΔE, defined by Eq. (7), is introduced to evaluate the deviation between the target and calculated values. The parallel-group structure [Fig. 7(b)] has a smaller ΔE value, and

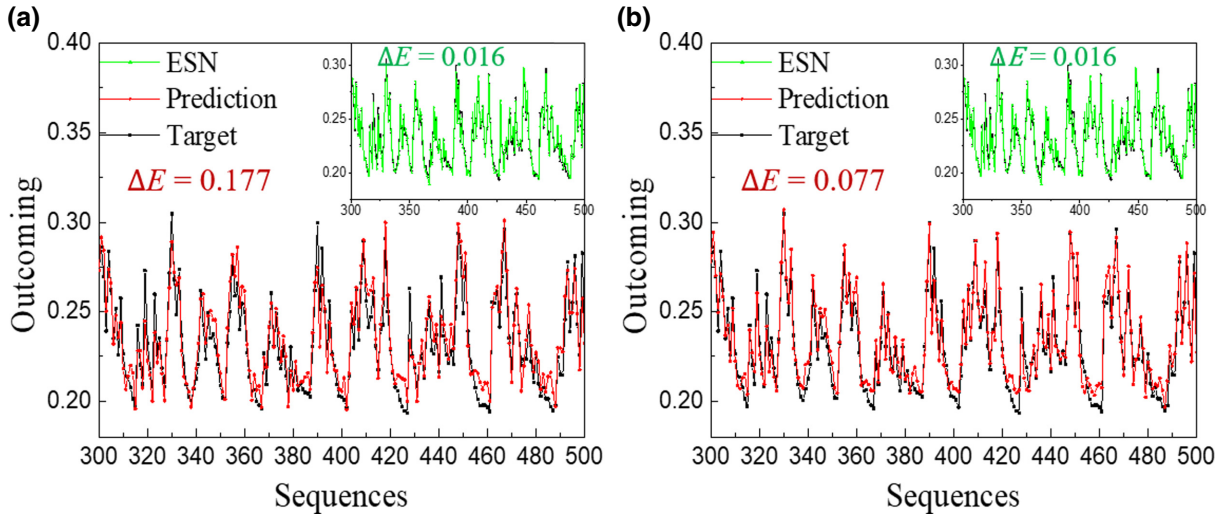

FIG. 7. Performance in NARMA2 task. Results for (a) parallel-node and (b) parallel-group structures. Red line is the trained prediction signal, and black line is the supervisor signal (NARMA2 model). Green lines in the inset are testing results from ESN code. ΔE is used to reflect the extent of deviation between predicted and target values.

thus, outperforms the parallel-node structure in this prediction task. Not surprisingly, the standard ESN obtains an obviously smaller ΔE. However, if we were to replace the activation function in this programmed ESN with the *I*-*V* response used in our physical schemes, the ΔE would increase to 0.121.

C. Short-term memory capacity

A well-known feature of RC is its fading (short-term) memory, meaning that the current reservoir state contains information from recent past inputs, but it is unrelated to older ones. The short-term memory capacity (STMC) is introduced to reflect the correlation between the current reservoir states (at time t) and past inputs [at time $(t-k)$], ranging from zero to one. The STMC in RC networks is defined to be represented by the squared correlation coefficient of the target testing signal, y_k, and the fitted signal, $\widehat{y_k}$, as proposed by Jeager [26]. Based on this definition, we obtain STMCs for higher-order target signals [27], where the nonlinear target function is a Legendre polynomial of a time-delayed input. Specifically, target signal y_k^q is constructed from the kth delayed original input signal $[u(t-k)]$, according to the qth order of Legendre polynomials (see the Appendix). The fitted signal, $\widehat{y_k^q}$, is trained using ridge regression by reservoir-state X, and the procedure is the same as that used in the NARMA2 task. Under the same nonlinear order, q, $r(y,\hat{y})_k$ is used to calculate the degree of correlation between the optimally trained output

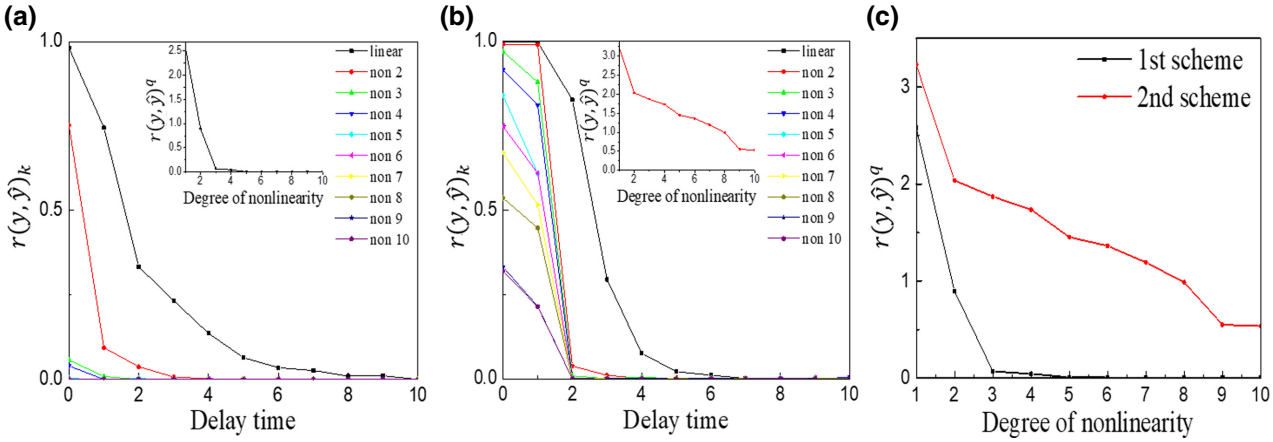

FIG. 8. Memory-capacity performance of hardware schemes. Results for (a) parallel-node and (b) parallel-group structures. $r(y,\hat{y})_k$ represents the fraction of variance explainable in one signal by another. $r(y,\hat{y})^q$ sums all delayed $r(y,\hat{y})_k$ (k from 0 to 10). (c) $r(y,\hat{y})^q$ results for two schemes. Degree q (1 to 10) represents the target signal given by qth-order Legendre polynomials.

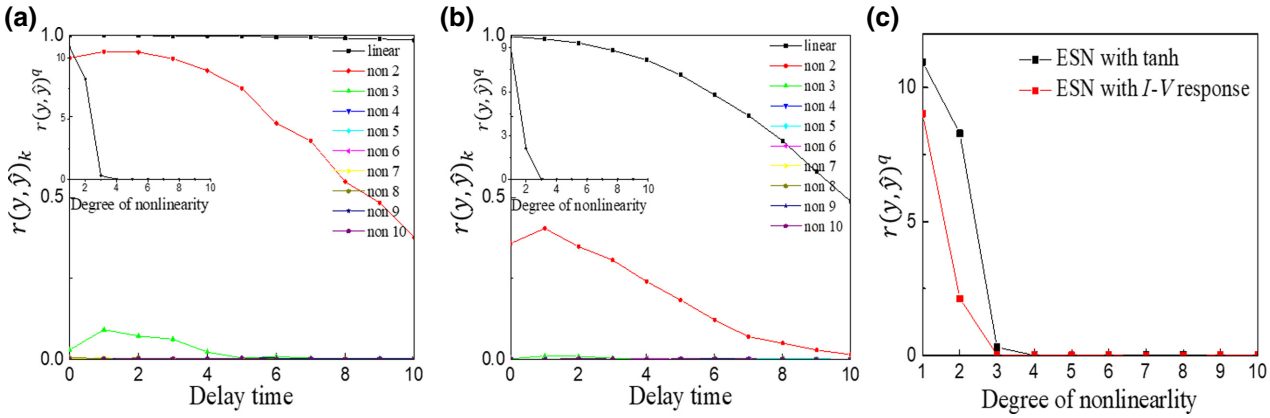

FIG. 9. MC performance of programmed ESN (a) with hyperbolic tangent function and (b) with I-V response as the activation function. (c) MC to different orders of target signal (i.e., $r(y,\hat{y})^q$) for these two ESNs.

signal, $\widehat{y_k}$, and the target signal, y_k. MC^q, summed over all MC_k with the same q, signifies how much of the delayed input signal can be recovered from $\widehat{y_k}$. The input signal, $ut(t)$, here, is the same random sequence as that used in the NARMA2 task, but mapped from its original range [0, 0.5] to [−1, 1], which is the standard input range for a memory-capacity task. Under the same order of nonlinearity, a STMC ranging between zero and one represents the correlation coefficient of the trained output and target signals. This is the k-delay STMC and is expressed as

$$r(y,\hat{y})_k^q = \frac{cov^2(y_k^q, \widehat{y_k})}{\sigma^2(y_k^q)\sigma^2(\widehat{y_k})}. \tag{9}$$

The STMC of the same order of the nonlinear target is the sum of all delayed STMCs:

$$r(y,\hat{y})^q = \sum_{k=1}^{\infty} r(y,\hat{y})_k^q. \tag{10}$$

Figures 8(a) and 8(b) show the STMCs for the parallel-node and -group structures, respectively, both with 300 nodes. Although their total memory capacities are not high, the parallel-group structure suppresses memory decline, especially against forgetting the two prior memory spans. The curves in Fig. 8(c) are the $r(y,\hat{y})^q$ results of the two structures. The MC $r(y,\hat{y})$ of the second scheme is improved, and the higher-order target is better remembered.

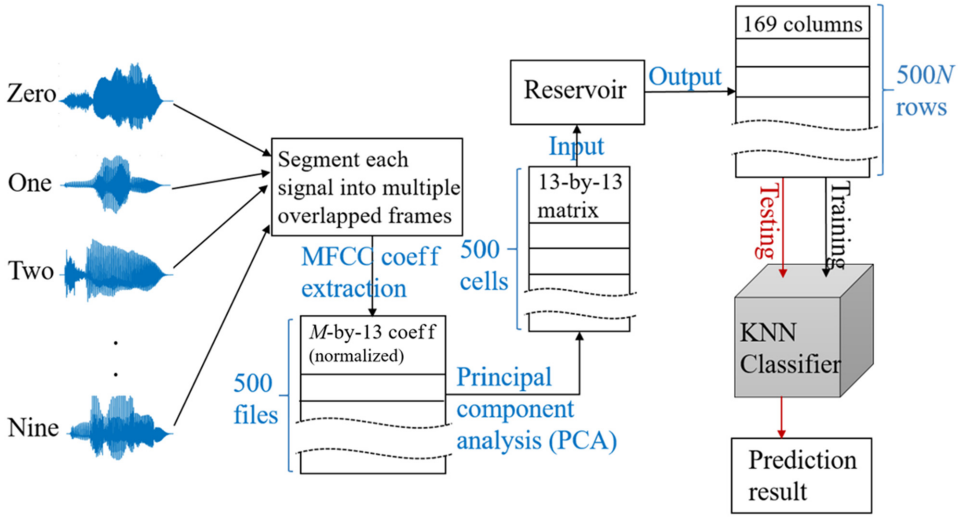

FIG. 10. Schematic of the isolated-word recognition process. $M \times 13$ features are extracted separately for each of the 500 audio signals, where M is the number of segmented frames. After principal component analysis, all coefficient matrixes with different sizes are organized into the same size of 13×13 (i.e., 169 coefficients) for one speech sample. Reservoir of N nodes would give a reservoir state of $500 \times 169 \times N$ data items. We rearrange the reservoir state into 500 cells (each given the same label) and use them for training and testing sets. Detailed description of this classification process is given in Fig. 11.

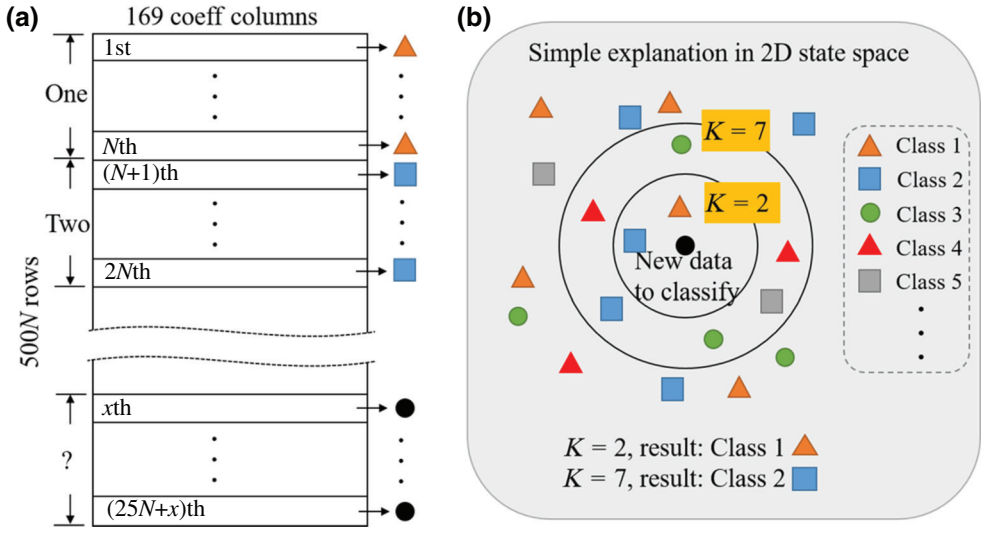

FIG. 11. Classification process using KNN model. (a) 500 output cells with N rows and 169 columns. After identifying training subsets, each row with 169 variables is projected to a point in 169-dimensional space. Rows or observations with the same label are projected to points of the same color and shape. (b) Simplified schematic of the KNN model. This illustration represents two-dimensional (2D) space for easier understanding, but the actual space is 169-dimensional, and the coordinates come from 169 variables.

For comparison, we measure the memory capacity of our programmed ESNs with 300 nodes. The activation of one is a hyperbolic tangent function [Fig. 9(a)], and the other one uses the I-V response we use in our physical schemes [Fig. 9(b)]. Both ESNs demonstrate significant abilities to remember inputs after a longer time, compared with our hardware systems. However, they perform poorly in the memory-capacity-to-higher-order target.

D. Testing the accuracy of speech recognition

An isolated-word recognition task is performed to further analyze their performances. Five hundred audio files comprising ten English spoken words ("zero" to "nine") are downloaded from the internet [28]. These audio files are recorded from five people (three men and two women), and each word is repeated ten times by the same speaker. Preprocessing, feature extraction, and training procedures are performed using MATLAB 2019a. See the Appendix for a complete description.

Figure 10 shows a general flowchart of this task. Five hundred feature vectors are extracted from 500 audio files input into the reservoir. Five hundred output cells are collected from the reservoir for training and testing, where each cell consists of N vectors processed by N nodes ($N = 50$ for this task). A k-nearest-neighbor (KNN) classifier combined with 20-fold stratified cross validation realizes the training and testing of samples. The KNN classifier is trained by specifying the distance metric as Euclidean, the distance weighting function as squared inverse, and the number of neighbors as k. The best performance is obtained when $k = 58$ for the parallel-node structure, 36 for the parallel-group structure, and nine for the ESN, as applied to the *fitcknn* function in MATLAB. All samples are divided into 20 disjointed subsamples (folds) chosen randomly, but with the same size. The training and testing process is repeated 20 times, each with a different assignment of subsamples as the testing set, and the remaining subsamples use the training set. Training data are mapped into 169-dimensional space based on their 169 predictor variables represented by the colored points in Fig. 11(b), and testing data, represented by black points, are used for classification. According to the calculated distances between each testing point and all other training points, the class with the largest weighted value is specified as the predicted class. In this case, each speech cell has 50 predicted labels; the most frequent label is the final class for this speech cell. The final accuracy rate is calculated as the average of these 20 cross validations.

The results for three RC systems are listed in Table I. The accuracies for the parallel-node and -group structures (both having 50 nodes) are 81% and 83.6%, respectively, and those for the standard ESN programming are 66.8% with 50 nodes and 72.6% with 200 nodes. When we change the spectral radius (SP, an eigenvalue of the network weight matrix with the largest absolute value) in the defined ESN, the accuracy under 50 nodes reaches 81.6%. To evaluate the contribution of the reservoir to performance, we test it in accordance with the procedure in Fig. 9, but without the reservoir. The testing accuracy is 74% (68%–80%), which represents the contribution from the preprocessing and postprocessing of data.

TABLE I. Cross-validation results of three different reservoir structures.

System	ESN ($\rho(W) = 0.99$)	ESN ($\rho(W) = 0.19$)	Parallel-node RC	Parallel-group RC
Accuracy	66.8% (52%–80%)	81.6% (68%–92%)	81% (64%–92%)	83.6% (76%–92%)

IV. DISCUSSION AND CONCLUSION

The motivation for this work is to find a suitable method for using the inherent nonlinear responses of materials to construct suitable dynamical systems for physical RCs. We demonstrate the use of a simple nonlinear I-V response and a simple external control to increase its internal dynamics. This opens the door to using molecular devices as processors.

Conventional RCs have several important properties, such as varied dynamical responses, fading memory, and nonlinear transformations, which are reflected by parameters such as input scaling, spectral radius, and sparsity. In our scheme, we reduce the RC to only two parameters: feedback gain a and input gain e. These two parameters can be used to adjust nonlinearity, control the dynamics of nodes, and generate virtual nodes. Their values have a significant effect on the final performance. Owing to the limitations of physical implementation, we find the optimal range for these values based on a limited number of trials. However, this does not affect our understanding of the relationship between dynamic properties and final performance. We hope to develop a practical method for finding the optimal range of parameters in future studies.

Effective reservoirs are generally considered to be necessary for achieving richly varied signals, which are ensured by a sparse interconnectivity of all nodes. However, in this work, we show that a scheme that uses independent nodes in the reservoir can work well for the isolated-word recognition task. An accuracy of 83.6% is achieved with only 50 nodes, indicating the potential of using basic nonlinearity. Moreover, we show that this kind of nonlinear response is more suitable for a structure with regular connections between nodes. The parallel-group RC outperforms the standard ESN with an activation function of the I-V response in a NARMA task, and it has a larger memory capacity for higher-order target signals. The standard ESN has a high STMC and, therefore, has a low ΔE in the NARMA2 task. However, its improvement to STMC for a higher-order nonlinear target ($q \geq 3$) is quite limited. The performance of these schemes in the speech-recognition task is inconsistent with that of previous tasks. Although the pre- and postprocessing of speech information plays a positive role in this task, our physical schemes need only a small number of nodes to improve the accuracy rate. Our physical reservoir does not require an increased number of nodes to show higher accuracy in the recognition task, whereas the standard ESN does. Nevertheless, the highest accuracy for this ESN (obtained with 200 nodes) is still lower than that of our physical reservoir with 50 nodes. Notably, the standard ESN that performs well in the NARMA task performs poorly in our isolated-word recognition task. When the ESN reaches an accuracy of 81.6% (under a spectral radius of 0.19), the performance in the NARMA task becomes worse ($\Delta E = 0.0778$). This demonstrates that, for tasks such as the isolated-word recognition task, the extraction of current information is important, and information from the previous state may undermine the characteristics of the current information. Therefore, although increasing the number of nodes in the ESN facilitates the extraction of information features, it can improve only the recognition accuracy to a level similar to the contribution of pre- and postprocessing. These results can be used to develop heuristics for building specific physical reservoirs and selecting the training and testing method for a specific task.

We use a single electronic circuit to generate numerous nodes as the reservoir, where the signals are serially fed into it. Processing under the sequential feeding of input is slow. However, in our schemes, nodes or node groups have an independent parallel architecture, which facilitates the fabrication of parallel hardware of physical nodes to compensate for the scaling of the total computation time with the number of virtual nodes. For the parallel-node structure, because the response is instantaneous and determined only by the most recent output and the current input, there is no need to access memory to extract the collected output states. In the parallel-group scheme, the memory is accessed only once for each current input calculation to read the state of the previous node.

Although real molecular devices will have nonidealities, such as noise and variability of characteristics, we think that the inherent properties of molecules can correspond to the randomness in our parameter selection, which would make the reservoir states more separable. A similar point of view is also proposed for a physical RC comprising a memristor array [12]. Noise that occurs during signal processing may degrade its performance by violating the rules of processing, but it may still be possible to utilize. For example, we have previously investigated a nanomaterial device with a noise-assisted nonlinear signal response, which is a promising approach for decreasing energy consumption through informational signal processing and transference [21]. However, based on current research, we believe that materials with stable and adjustable I-V properties, such as a nanoparticle bridge with temperature-dependent properties [18], are a more appropriate choice. The results of this work can be used for the development of single-molecule devices for informational processing and the application of nanomaterials to physical RCs.

ACKNOWLEDGMENTS

This work is partially based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO). K.N. is supported by JSPS KAKENHI Grant No. JP18H05472.

APPENDIX

Training procedure for NARMA2 task

This section describes the training process for the NARMA2 task. The input signal, $u(t)$, is a sequence of $T = 4000$ digits randomly generated in the interval [0, 0.5]. By substituting the input sequence into the NARMA2 model [Eq. (6)], we can obtain the target signal, $y(t)$. If the input signal is fed into the reservoir with $N = 300$ nodes, they will respond to the input signal, producing 300 vectors of node states. Each vector contains 4000 data items, which include the output voltages of each physical node recorded at 4000 time steps. This 4000×300 matrix is the reservoir state, X_{out}. A total of 100 data items in each vector are used to initialize the reservoir, 2000 data items are used for training, and the remaining 1900 are used for testing.

With a 300×1 output weight matrix, the output signal can be calculated using matrix multiplication. During the training process, the optimal output weights (i.e., those that render the output signal as close as possible to the target signal) are determined. We use ridge regression, which is essentially an improved least-squares estimation method, to train the output signal. The aim of least-squares estimation is to minimize the objective function, $\sum \left[y_{\text{train}} - \left(\beta_0 + \sum_{i=1}^{N} \beta_i X_{\text{train}i} \right) \right]^2$, so the coefficients can be obtained by choosing $\beta = (X_{\text{train}}^T X_{\text{train}})^{-1} X_{\text{train}}^T y_{\text{train}}$. Ridge regression addresses the problem of multicollinearity by estimating regression coefficients as

$$\beta = (X_{\text{train}}^T X_{\text{train}} + kI)^{-1} X_{\text{train}}^T y_{\text{train}}, \tag{A1}$$

where k is the ridge parameter and I is the identity matrix. k is 22, 5, 0.6, and 1, respectively, for the parallel-node structure, parallel-group structure, standard ESN with hyperbolic tangent function, and ESN with I-V response. The output weight matrix, W_{out}, is $\beta_{1:N} \times 1$. To test the performance of this system, we apply the optimal output weights determined during the training stage to the testing data and obtain the predicted signal, $\hat{y}(t) = \beta_0 + X_{\text{test}} W_{\text{out}}$.

Supplementary information for memory-capacity task

The process of memory-capacity measurement is similar to the NARMA2 task, except that the target signal is obtained by Legendre polynomials instead of the NARMA2 model. The expressions of the Legendre polynomials of orders 1–10 are listed in Table II.

For a given delay, k, the input variable, x, in the Legendre polynomials is replaced by $u(t-k)$. The reservoir still receives $u(t)$ as its input signal. This means that the reservoir state, X, is the response under input sequence $u(t)$. The aim is to fit the target signal given by the sequence $u(t-k)$. The training and testing process is exactly the same as that for the NARMA2 task, but the ridge parameter in ridge regression training is taken as one for all schemes in this task.

TABLE II. Legendre polynomials.

q	$P_q(x)$
1	x
2	$\frac{1}{2}(3x^2 - 1)$
3	$\frac{1}{2}(5x^3 - 3x)$
4	$\frac{1}{8}(35x^4 - 30x^2 + 3)$
5	$\frac{1}{8}(36x^5 - 70x^3 + 15x)$
6	$\frac{1}{16}(231x^6 - 315x^4 + 105x^2 - 5)$
7	$\frac{1}{16}(429x^7 - 693x^5 + 315x^3 - 35x)$
8	$\frac{1}{128}(6435x^8 - 12012x^6 + 6930x^4 - 1260x^2 + 35)$
9	$\frac{1}{128}(12155x^9 - 25740x^7 + 18018x^5 - 4620x^3 + 315x)$
10	$\frac{1}{256}(46189x^{10} - 109395x^8 + 90090x^6 - 30030x^4 + 3465x^2 - 63)$

Procedure for speech-recognition task

This task uses the extracted features of Mel-frequency cepstral coefficients (MFCCs) and linear classification using KNN, as described below.

We download our dataset from a website [28,29] that hosts a large number of audio files with words (i.e., digits "zero" to "nine") spoken by various speakers. We randomly pick 500 audio files (three male speakers and two female speakers) as our dataset. Each spoken digit is recorded 10 times in the dataset (i.e., 100 samples for each speaker and 500 samples in total).

We first use the *audioread* function in MATLAB to read data from each audio file and return the sampled data obtained at a sampling rate of 8 kHz. Then, we use the MFCC function to extract the log energy and MFCCs of each signal. The extraction process is performed frame by frame, every 30 *ms* with an overlap of 75%. The log energy and MFCCs are calculated in each frame. The first coefficient in the coefficient vector is replaced with the log energy value, so that each coefficient vector includes 13 coefficients. This method is widely used to extract parameters in speech-recognition tasks, including delta and delta-delta coefficients. We perform three trials using our reservoir system under the same settings. One is done using MFCCs and log energy; one is done using MFCCs, delta, and log energy; and one is done using MFCCs, delta, delta-delta, and log energy. The results of these trials show that the accuracy decreases with an increasing number of

parameters. We believe that isolated words are not complicated enough to extract many parameters. Hence, we extract only MFCCs and log energy as the features in this task.

We notice that log energy is not on the same scale as MFCC, which may bias the classifier. Thus, we normalize the features by subtracting the mean and dividing the standard deviation of each column. After MFCC extraction, we have $M \times 13$ coefficients for each audio file. Because the length of each sampled speech is different, M is not fixed. To reduce dimensionality and normalize the size of coefficients of each speech signal, principal component analysis is used to organize the coefficients extracted from each audio file into a matrix of the same size (i.e., 13×13). Before inputting these data into the reservoir, we rearrange each 13×13 matrix into a row vector of 169 elements, labeled with its spoken word (digits "zero" to "nine"). The input data are hence a 500×169 matrix, which is fed into the reservoir row by row after the row minimum and maximum values are mapped to $[-0.5, 0.5]$. This normalization is necessary, because, in this task, all parameters take positives values. Recall that, when we choose parameters based on the NARMA2 task, the range of the input signal for the NARMA task is [0, 0.5], and the input gain, e, is randomly assigned a plus or minus sign within the selected range. Next, the output from the reservoir with 50 nodes is rearranged into 500 50×169 submatrices, where the rows represent the observations, and the columns represent the predicted variables. After the features and their labels are collected, they are used to train the classifiers.

Ten classes (labels), representing ten speech targets (digits "zero" to "nine") are trained using the KNN algorithm (*fitcknn* function in MATLAB). The hyperparameters for the KNN classifier include the number of nearest neighbors, the distance metric used to compute the distance to the neighbors, and the weight of the distance metric. In this task, the number of neighbors is set to k ($k = 58$ for the parallel-node structure, 36 for the parallel-group structure, and 11 for the ESN, optimized for the best performance), and the metric for the selected distance is the squared-inverse-weighted Euclidean distance. An n-fold ($n = 20$) stratified cross validation is used in the KNN classifier to reduce the error caused by the selection of training data and testing data.

To decrease the sensitivity of model performance to data partitioning, 500 output submatrices are divided randomly into 20 disjointed subsamples, each consisting of 25 submatrices. The training and testing process is repeated 20 times, with a different set of subsamples as the testing set each time. The remaining subsamples are used as the training set. This training of 20 classifiers under different combinations of training subsets continues until each subset is predicted. All 25 000 rows of observations are mapped onto 169-dimensional space based on their 169 predictor variables as the relative coordinate value. We calculate the squared-inverse-weighted Euclidean distances between each testing point and all other training points and rank them in ascending order by the KNN model. Next, for the KNN, the class with the largest weighted value is specified as the predicted label. The most-frequent label in one submatrix with 50 labels is used as the final class for this speech cell. The final accuracy rate is the average of prediction results of these 20 trained classifiers. The accuracies of the parallel-node and -group structures are 81% (64%–92%) and 83.6% (76%–92%), respectively, and that of the ESN, which performs much better in the NARMA2 task, is only 66.8% (52%–80%) for this task.

Parameter settings in ESN systems

Most parameters for our ESN are fixed (activation function, tanh function; spectral radius, 0.99; input weight interval, $[-1,1]$; node connection weight interval, $[-0.99, 0.99]$). Input scaling is introduced in the speech-recognition task to normalize the input signal within the interval $[-0.5, 0.5]$, which is consistent with the input scale in our physical RC systems. Input scaling is not used in the NARMA2 task because the input signal, $u(t)$, is generated within the range of [0, 0.5], and the same signal, $u(t)$, is used for all physical systems and the ESN system.

[1] J. Schmidhuber, Deep learning in neural networks: An overview, Neural Netw. **61**, 85 (2015).

[2] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, How to Construct Deep Recurrent Neural Networks, *ArXiv:1312.6026* (2014).

[3] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt, An experimental unification of reservoir computing methods, Neural Netw. **20**, 391 (2007).

[4] T. Cover, Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition, IEEE Trans. Electron **14**, 326 (1965).

[5] C. Gallicchio, A. Micheli, and L. Pedrelli, Deep reservoir computing: A critical analysis, Neurocomputing **268**, 87 (2017).

[6] Z. K. Malik, A. Hussain, and Q. J. Wu, Multilayered echo state machine: A novel architecture and algorithm, IEEE Transactions on Cybernetics **47**, 946 (2017).

[7] L. K. Grigoryeva, J. Henriques, L. Larger, and J.-P. Ortega, Nonlinear memory capacity of parallel time-delay reservoir computers in the processing of multidimensional signals, Neural Comput. **28**, 1411 (2016).

[8] L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer, Information processing using a single dynamical node as complex system, Nat. Commun. **2**, 468 (2011).

[9] J. Qiao, F. Li, H. Han, and W. Li, Growing echo-state network with multiple subreservoirs, IEEE **28**, 391 (2017).

[10] K. Nakajima, Physical reservoir computing—an introductory perspective, Jpn. J. Appl. Phys. **59**, 060501 (2020).

[11] G. Tanaka, T. Yamane, J. B. Heroux, R. Nakaneab, N. Kanazawac, S. Takedac, H. Numatac, D. Nakanoc, and A. Hiroseab, Recent advances in physical reservoir computing: A review, Neural Netw. **115**, 100 (2019).

[12] 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).

[13] H. Sillin, R. Aguilera, H. H. Shieh, A. V. Avizienis, M. Aono, A. Z. Stieg, and J. K. Gimzewski, A theoretical and experimental study of neuromorphic atomic switch networks for reservoir computing, Nanotechnology **24**, 38 (2013).

[14] K. Nakajima, H. Hauser, T. Li, and R. Pfeifer, Information processing via physical soft body, Sci. Rep. **5**, 10487 (2015).

[15] J. Torrejon, M. Riou, F. A. Araujo, 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 (2015).

[16] S. K. Bose, C. P. Lawrence, Z. Liu, K. S. Makarenko, R. M. J. van Damme, H. J. Broersma, and W. G. van der Wiel, Evolution of a designless nanoparticle network into reconfigurable Boolean logic, Nat. Nanotechnol. **10**, 1048 (2015).

[17] H. Tanaka, M. Akai-Kasaya, A. TermehYousefi, L. Hong, L. Fu, H. Tamukoh, D. Tanaka, T. Asai, and T. Ogawa, A molecular neuromorphic network device consisting of single-walled carbon nanotubes complexed with polyoxometalate, Nat. Commun. **9**, 2693 (2018).

[18] S. Nishijima, Y. Otsuka, H. Ohoyama, K. Kajimoto, K. Araki, and T. Matsumoto, Resonant tunneling via a Ru-dye complex using a nanoparticle bridge junction, Nanotechnology **29**, 245205 (2018).

[19] Z. Wang, S. Joshi, S. E. Savel'ev, H. Jiang, R. Midya, P. Lin, M. Hu, N. Ge, J. P. Strachan, Z. Li *et al.*, Memristors with diffusive dynamics as synaptic emulators for neuromorphic computing, Nat. Mater. **16**, 101 (2017).

[20] Z. Crljen, A. Grigoriev, G. Wendin, and K. Stokbro, Nonlinear conductance in molecular devices: Molecular length dependence, Phys. Rev. B **71**, 165316 (2005).

[21] H. Fuji, A. Setiadi, Y. Kuwahara, and M. Akai-Kasaya, Single walled carbon nanotube-based stochastic resonance device with molecular self-noise source, Appl. Phys. Lett. **111**, 133501 (2017).

[22] K. Nakajima, K. Fujii, M. Negoro, K. Mitarai, and M. Kitagawa, Boosting Computational Power Through Spatial Multiplexing in Quantum Reservoir Computing, Phys. Rev. Appl. **11**, 034021 (2019).

[23] H. Jaeger, The "Echo State" approach to analyzing and training recurrent neural network, in GMD-German National Research Institute for Computer Science, *Bremen: GMD report 148* (2001).

[24] A. F. Atiya and A. G. Parlos, New results on recurrent network training: Unifying the algorithms and accelerating convergence, IEEE Trans. Neural Netw. **11**, 697 (2000).

[25] Y. Yamanaka, T. Yaguchi, K. Nakajima, and H. Hauser, Mass-spring damper array as a mechanical medium for computation, Lect. Notes Comput. Sci. **11141**, 781 (2018).

[26] H. Jeager, Short term memory in echo state networks, GMD Rep. **152**, 245205 (2001).

[27] J. Dambre, D. Verstraeten, B. Schrauwen, and S. Massar, Information processing capacity of dynamical systems, Sci. Rep. **2**, 514 (2012).

[28] B. Cramer, Y. Stradmann, J. Schemmel, and F. Zenke, The Heidelberg spiking datasets for the systematic evaluation of spiking neural networks, *ArXiv*:1910.07407 [Cs, q-Bio] (2019).

[29] https://compneuro.net.