IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 19 October 2023, accepted 28 November 2023, date of publication 7 December 2023,
date of current version 13 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3340311

RESEARCH ARTICLE

Self-Organizing Multiple Readouts for Reservoir Computing

YUICHIRO TANAKA[1], (Member, IEEE), AND HAKARU TAMUKOH[1,2], (Member, IEEE)
[1]Research Center for Neuromorphic AI Hardware, Kyushu Institute of Technology, Wakamatsu, Kitakyushu 808-0196, Japan
[2]Graduate School of Life Science and Systems Engineering, Kyushu Institute of Technology, Wakamatsu, Kitakyushu 808-0196, Japan

Corresponding author: Yuichiro Tanaka (tanaka-yuichiro@brain.kyutech.ac.jp)

This work was supported in part by the New Energy and Industrial Technology Development Organization (NEDO) under Project JPNP16007; and in part by the Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant 22K17968, Grant 23H03468, and Grant 23K18495.

ABSTRACT With advancements in deep learning (DL), artificial intelligence (AI) technology has become an indispensable tool. However, the application of DL incurs significant computational costs, making it less viable for edge AI scenarios. Consequently, the demand for cost-effective AI solutions, other than DL-based approaches, is increasing. Reservoir computing (RC) has attracted interest owing to its ability to provide low-cost training alternatives, holding great promise for edge AI applications. However, the training capability of RC is constrained by its reliance on a single linear layer, while weight connections in the remaining layers remain static during training. Moreover, accomplishing continuous learning tasks is difficult owing to the catastrophic forgetting in the linear layer. Therefore, we propose the integration of self-organizing multiple readouts to enhance RC's training capability. Our method distributes training data across multiple readouts, which prevents catastrophic forgetting of readouts and empowers each readout to adeptly assimilate new data, thereby elevating the overall training performance. The self-organizing function, which assigns similar data to the same readout, optimizes the memory utilization of these multiple readouts. Experimental results show that an RC equipped with the proposed multiple readouts successfully solved a continuous learning task by mitigating catastrophic forgetting because of the data distribution to the multiple readouts. Additionally, the RC achieved higher accuracy in a sound recognition task compared with the existing RC paradigm because of ensemble learning in the multiple readouts. Multiple readouts are effective in enhancing the training capability of RC and can contribute to the realization of RC applications.

INDEX TERMS Catastrophic forgetting, continuous learning, echo state network, edge computing, ensemble learning, reservoir computing, self-organizing map, sound classification.

I. INTRODUCTION

Deep learning (DL) [1] is a cornerstone of contemporary artificial intelligence (AI) technology and has grown indispensable across diverse fields [2], [3], [4], [5]. The potency of DL lies in its capability to achieve cutting-edge outcomes, facilitated by extensive training data that optimize its intricate architecture. Consequently, the training phase of DL demands substantial computational resources. In general, the computations required for the training and inference of deep neural networks (DNNs) are often accelerated using graphics processing units (GPUs) [6], [7], [8].

Among various applications of AI, its utilization in edge environments, exemplified by service robots [9], [10], [11] operating within domestic settings, has garnered notable attention and aspiration. However, current DL-based methodologies encounter challenges in catering to edge applications owing to constraints imposed by limited training data, computational resources, and energy efficiencies. Certain edge scenarios necessitate AI computation to occur autonomously, devoid of reliance on cloud servers, a prerequisite driven by privacy concerns, particularly pertinent when considering familial data in the context of service robots. This independence from cloud infrastructure, however, imposes intrinsic constraints on available computational resources and processing abilities. Fig. 1 presents a potential solution

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Liu.

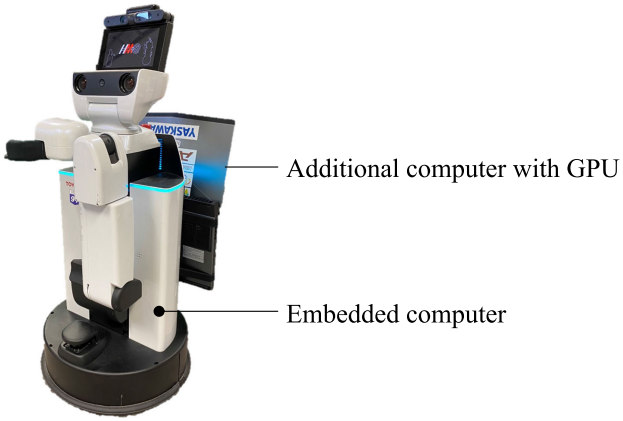

FIGURE 1. One of the solutions for edge AI implementation.

for edge AI implementation, entailing the linking of a robot to an auxiliary laptop outfitted with GPUs. However, it is essential to address the significant power consumption associated with laptop GPUs, as their operation can swiftly deplete the system's battery reserves. Consequently, the drive for AI solutions necessitating minimal training-related computational expenditure, distinct from the conventional DL paradigm, has gained momentum.

Reservoir computing (RC) [12], [13], a type of recurrent neural network, presents an attractive prospect for edge AI applications, characterized by minimal training overhead. Notably, RC exhibits a potential for deployment on energy-efficient hardware platforms through specialized circuits [14], [15], [16], [17], [18], [19], as well as leveraging physical processes for computation [20], [21], [22], [23], [24], [25]. This distinct characteristic positions RC as an optimal candidate for edge AI implementations, aligning with considerations of power consumption.

In general, the RC architecture comprises three fundamental layers: input, reservoir, and readout layers. Within the reservoir layer, an intricate nonlinear transformation transpires, projecting the input data into a high-dimensional space. This transformation captures intricate spatio-temporal patterns inherent in the input data, effectively encoding their complexity. The patterns are then subjected to a linear conversion process within the readout layer. This linear conversion yields outputs, wherein a selection of reservoir layer patterns, pertinent to the tasks at hand, contributes to the generation of these outputs. A key advantage of RC is its establishment of fixed weight connections, with the exception of the readout, as depicted in Fig. 2(a). By contrast, neural networks optimized via backpropagation [26], such as DNNs, are capable of updating all weight connections, as shown in Fig. 2(b). Therefore, RC training demands less data and incurs lower computational burden compared with DNNs.

The readout function of RC, typically realized through a linear model (such as the single-layer perceptron), presents a limitation in its adaptability to diverse training data. Moreover, in cases involving continuous learning [27], the

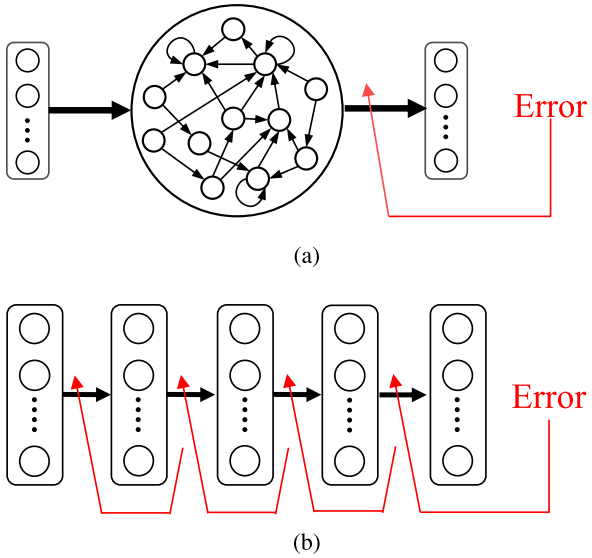

FIGURE 2. Training comparison: Reservoir computing vs. backpropagation-optimized neural network.

optimized readout weight connections can be overwritten by subsequent tasks, often leading to the challenge of catastrophic forgetting [28]. This phenomenon results in the RC effectively accommodating the new task while losing competence in the previous task.

To improve RC's training capabilities and prevent catastrophic forgetting, we propose an RC model featuring multiple readouts [29]. Our approach distributes the training responsibility among these readouts, thereby enabling each to specialize in specific categories of training data. Conceptually similar to ensemble learning, this method augments RC's capacity for generalization. However, simply increasing the number of readouts is inefficient within edge AI systems, given their limited memory resources. Consequently, this study introduces a self-organizing function that allocates the same readout for similar data and employs different readouts for dissimilar data.

The contributions of this study are as follows. This study introduces a novel RC structure and its associated training method. These enhancements not only boost the performance of RC but also enable it to tackle continuous learning tasks, an area where existing RC models often struggle. Furthermore, the proposed method is applicable to the physical implementations of RC [20], [21], [22], [23], [24], [25] without any modification to the reservoir layer, which is realized by physical phenomena, and therefore the method is also expected to improve the performance of physical implementations of RC, which is often constrained by physical systems. This study can contribute to the realization of RC applications.

The rest of this paper is organized as follows. Section II introduces RC, with a specific focus on echo state networks (ESNs) [12], the RC implementations employed in this study. Section III describes the proposed method, while Section IV delineates the experimental settings and ensuing results.

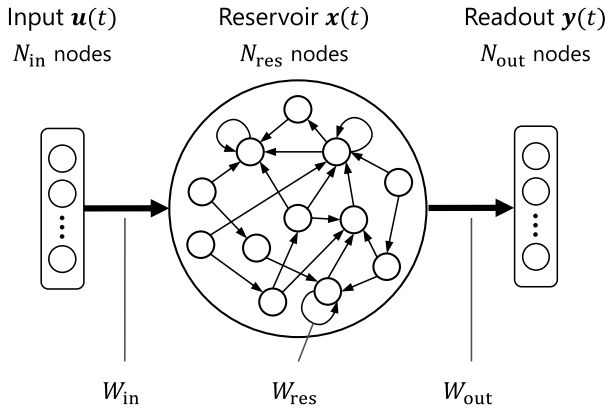

FIGURE 3. Echo state network.

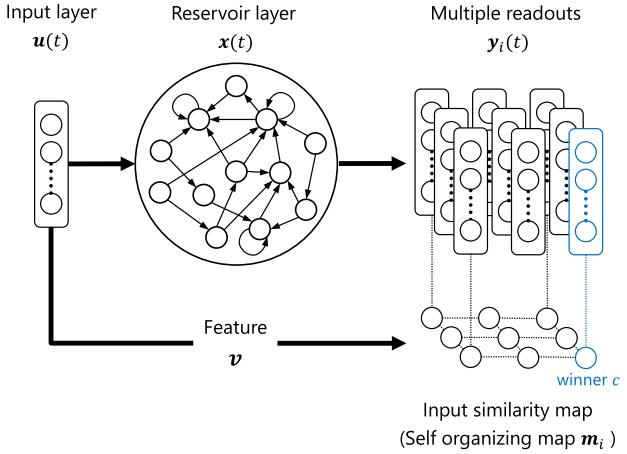

FIGURE 4. Reservoir computing model with multiple readouts (3 × 3 configuration).

Finally, Section V concludes the paper with a discussion that includes a comparative analysis with previous research endeavors.

II. ECHO STATE NETWORK

To implement the proposed RC model, this study utilizes ESNs as RC implementations. This section describes the structure and the information processing of ESNs.

An ESN is one of the RC implementations proposed by Jaeger [12], which has input, reservoir, and readout layers with N_{in}, N_{res}, and N_{out} nodes, respectively, as shown in Fig. 3. The reservoir layer contains weight connections from the input layer and recurrent connections. The readout layer linearly maps the states of the reservoir layer to generate output signals.

When an input signal $\boldsymbol{u}(t) \in \mathbb{R}^{N_{in}}$ at time t is given, a state of the reservoir layer $\boldsymbol{x} \in \mathbb{R}^{N_{res}}$ is updated using Eq. (1):

$$\boldsymbol{x}(t) = f(W_{in}\boldsymbol{u}(t) + W_{res}\boldsymbol{x}(t-1)), \tag{1}$$

where $W_{in} \in \mathbb{R}^{N_{res} \times N_{in}}$ and $W_{res} \in \mathbb{R}^{N_{res} \times N_{res}}$ are weight connections between the input and reservoir layers and recurrent weight connections in the reservoir layer, respectively. f indicates a nonlinear function, where a hyperbolic tangent function is often used.

To control the reservoir state updating speed, an RC model that utilizes leaky integrator models as reservoir nodes was proposed [30], and the reservoir state update process is represented as Eq. (2),

$$\boldsymbol{x}(t) = (1-\alpha)\boldsymbol{x}(t-1) + \alpha f(W_{in}\boldsymbol{u}(t) + W_{res}\boldsymbol{x}(t-1)), \tag{2}$$

where α $(0 < \alpha < 1)$ indicates the leak rate of the leaky integrator models. If α is close to 0, the first-term effect on the reservoir state update process becomes relatively big, slowing the update process. Conversely, if α is close to 1, the reservoir state update process becomes relatively fast.

In general, the reservoir layer is required to satisfy the echo state property [12] that assures the reproducibility of reservoirs. As shown in Eqs. (1) and (2), the reservoir state is uniquely decided by an initial reservoir state $\boldsymbol{x}(0)$ and time-series input $\boldsymbol{u}(t)$. Suppose that an initial reservoir state continues to affect the reservoir state update process. The reservoir state after feeding time-series inputs depends on the randomized initial state, and therefore different outputs are obtained in every execution, even if the same inputs are given. To avoid this, the reservoir state after a significant period must depend only on time-series inputs. If the hyperbolic tangent function is used as the nonlinear function f, the echo state property can be satisfied if the spectral radius of the recurrent weight connections W_{res}, which is the maximum value in the absolute values of eigenvalues of W_{res}, is smaller than 1.

An output of the network $\boldsymbol{y}(t) \in \mathbb{R}^{N_{out}}$ is computed by the readout layer using Eq. (3):

$$\boldsymbol{y}(t) = W_{out}\boldsymbol{x}(t), \tag{3}$$

where $W_{out} \in \mathbb{R}^{N_{out} \times N_{res}}$ is the weight connections between the reservoir and readout layers. Given a target signal $\boldsymbol{z}(t) \in \mathbb{R}^{N_{out}} (1 \leq t \leq T)$, the weight connections W_{out} can be computed by the ridge regression as shown in Eqs. (4), (5), and (6):

$$W_{out} = ZX^{\top}(XX^{\top} + \lambda I)^{-1}, \tag{4}$$

$$X = [\boldsymbol{x}(1), \boldsymbol{x}(2), \ldots, \boldsymbol{x}(T)] \in \mathbb{R}^{N_{res} \times T} \tag{5}$$

$$Z = [\boldsymbol{z}(1), \boldsymbol{z}(2), \ldots, \boldsymbol{z}(T)] \in \mathbb{R}^{N_{out} \times T} \tag{6}$$

where λ is a coefficient of the regularization term of the ridge regression that aims to avoid overfitting, and $I \in \mathbb{R}^{N_{res} \times N_{res}}$ is an identity matrix.

As mentioned above, only the weight connections between the reservoir and readout layers W_{out} have synaptic plasticity and are updated in the training, whereas other weight connections are fixed. Therefore, ESN's capability to adapt training data depends only on the readout layer, which is usually a single linear model, and adapting complex datasets is difficult. Moreover, catastrophic forgetting tends to occur in continuous learning tasks.

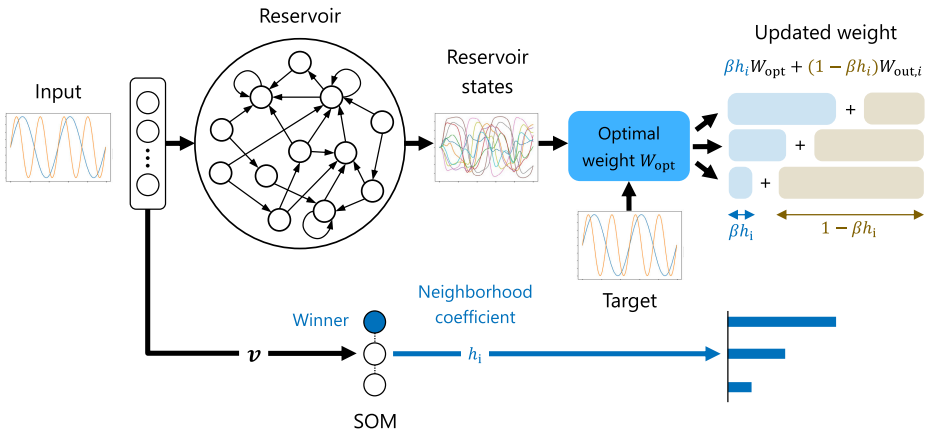

FIGURE 5. Online training with multiple readouts (the number of readouts is three).

III. PROPOSED METHOD

This study proposes an RC model with self-organizing multiple readouts and its two types of training methods. The first one is an online training method, which is effective for continuous learning tasks, and the other is a batch training method. This section includes subsections explaining the RC model structure and the training methods.

A. MODEL STRUCTURE

Fig. 4 shows the structure of the proposed RC model [29], which has an input layer, a reservoir layer, multiple readouts, and an input similarity map that is implemented by a self-organizing map (SOM) [31]. The SOM consists of nodes aligned on a two-dimensional grid (3×3 in the figure). Although the two-dimensional grid is adopted in this study, the grid dimension is not restricted to two. The number of SOM nodes corresponds to the number of multiple readouts, and each SOM node is associated with each readout.

The similarity map is used for input data classification to distribute the readout training. For this data classification, an unsupervised learning method is desired to avoid additional data labeling for the similarity map, which is highly time-consuming. This study adopted the SOM for the similarity map as one of the unsupervised learning methods. Note that other unsupervised learning methods like principal component analysis (PCA) and t-distributed stochastic neighbor embedding (t-SNE) [32] can be utilized (however, we did not adopt it because t-SNE requires considerable computational costs).

The SOM receives an input signal and classifies it to decide a winner node c using Eq. (7):

$$c = arg\min_i \|\boldsymbol{v} - \boldsymbol{m}_i\| \tag{7}$$

where i is an index of the SOM node, and $\boldsymbol{m}_i \in \mathbb{R}^M$ is a reference vector of i-th SOM node. The winner node of the SOM that has the nearest reference vector to $\boldsymbol{v}$ is selected in this process. Here, $\boldsymbol{v} \in \mathbb{R}^M$ is a vector representing the feature of the given task. The simplest way to represent the task feature is concatenating the input vectors of all time steps if they are already known using Eq. (8):

$$\boldsymbol{v} = [\boldsymbol{u}(1)^\top, \boldsymbol{u}(2)^\top, \ldots, \boldsymbol{u}(T)^\top]^\top \in \mathbb{R}^{N_{in}T} \tag{8}$$

In addition to concatenation, an alternative approach involves utilizing a spectrogram obtained through the Fourier transform of the input vectors as $\boldsymbol{v}$. In some cases, employing the spectrogram yields superior outcomes compared to concatenation. This is attributed to the fact that the concatenated vector derived from Eq. (8) lacks robustness against phase shifts present in the time-series inputs. For similar reasons, there is potential in utilizing the reservoir state $\boldsymbol{x}(t)$ as $\boldsymbol{v}$. However, in this setup, the winner node c dynamically changes while the input vector is provided, and the batch training method described in Section III-C cannot be utilized.

The reference vectors of the SOM are optimized via unsupervised competitive learning using Eqs. (9) and (10):

$$\boldsymbol{m}_i^{new} = \boldsymbol{m}_i + \gamma h_i(\boldsymbol{v} - \boldsymbol{m}_i) \tag{9}$$

$$h_i = \exp(-d_i^2/2\sigma^2) \tag{10}$$

where $\boldsymbol{m}_i^{new}$ is the updated reference vector generated after this process. γ is the learning rate, and h_i is a neighborhood coefficient depending on the distance d_i between the i-th node and the winner node on the SOM grid. σ indicates the width of the neighborhood coefficient affection.

The reservoir layer of the model is the same as the reservoir of the ESNs, and therefore the layer receives input signals and updates its state using Eq. (1) or (2). Each multiple readout with weight connections from the reservoir layer $W_{out,i}$ generates an output as shown in Eq. (11):

$$\boldsymbol{y}_i(t) = W_{out,i}\boldsymbol{x}(t) \tag{11}$$

where $\boldsymbol{y}_i(t)$ is the i-th readout output at time t. Finally, a readout output associated with the winner node (winner readout) $\boldsymbol{y}_c(t)$ is adopted as the network output.

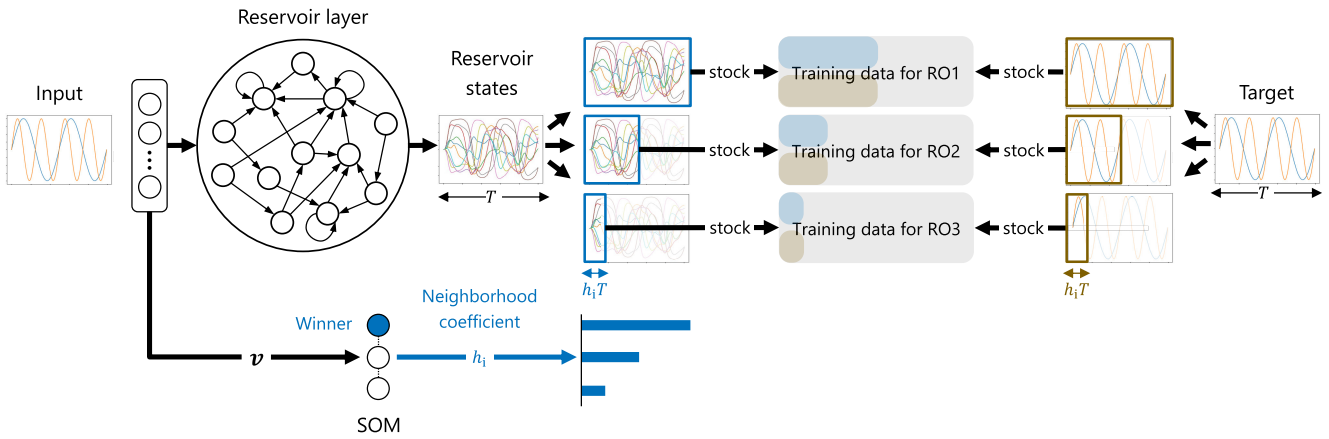

FIGURE 6. Batch training with multiple readouts (the number of readouts is three).

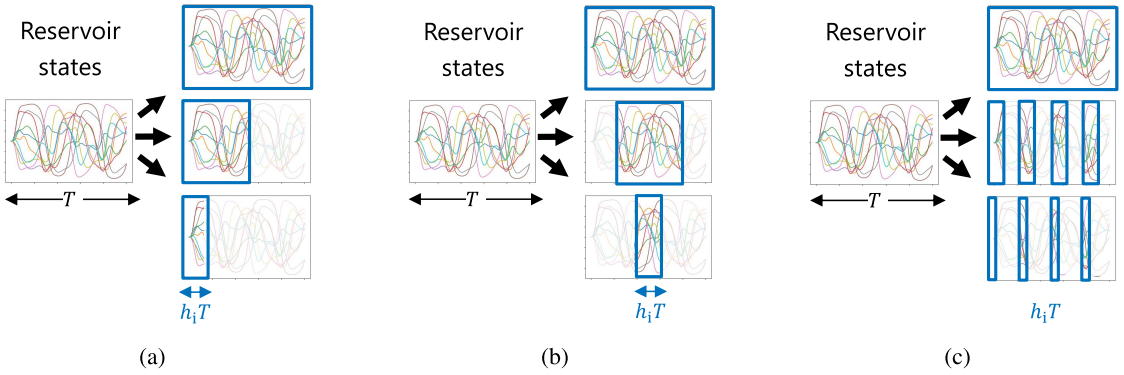

FIGURE 7. Variations in training data quantity during batch training.

B. ONLINE TRAINING

For the self-organizing function of the multiple readouts, we propose a training method where weight connections of the winner readout and its neighboring readouts are updated. This training methodology is strategically devised to enhance the winning readout's adjustment to the data responsible for the activation of the corresponding SOM node. Concurrently, it encourages a coarse alignment of the neighboring readouts with the same data source. As a result, this approach offers a twofold advantage: first, the winning readout becomes adept at specializing in data originating from a particular domain, and second, the neighboring readouts acquire the capability to generalize across data from diverse domains.

To control the readouts' adaptation, the online training algorithm adjusts the learning rate of the weight connection update process as shown in Fig. 5. When input and target signals are given to the model, optimal weight connections for the task W_{opt} can be computed using the ridge regression of Eq. (4). Here, the readout weight connections $W_{out,i}$ are updated to be close to the optimal weight connections. This is achieved by utilizing the neighboring coefficient of the SOM h_i, with the aim of enhancing the extent of updates in the winning readout among the multiple readouts. The update process is conducted using Eq. (12):

$$W_{out,i}^{new} = \beta h_i W_{opt} + (1 - \beta h_i) W_{out,i} \tag{12}$$

where $W_{out,i}^{new}$ is the updated readout weight connections after this process and β is a learning coefficient.

Note that the learning coefficient β and the width of the neighborhood coefficient affection σ are recommended to be monotonously decreased during the training because the updating effect of a newly given task tends to overwrite that of previous ones if the learning coefficient is constant.

C. BATCH TRAINING

The online training algorithm requires the parameters to be scheduled to achieve good results. The batch training algorithm overcomes this problem at the sacrifice of memory usage.

To control the readouts' adaptation, the batch training algorithm adjusts the amount of training data based on the neighborhood coefficient of the SOM as shown in Fig. 6. This method prepares memories associated with readouts and stocks the training data in the memories. When T steps input

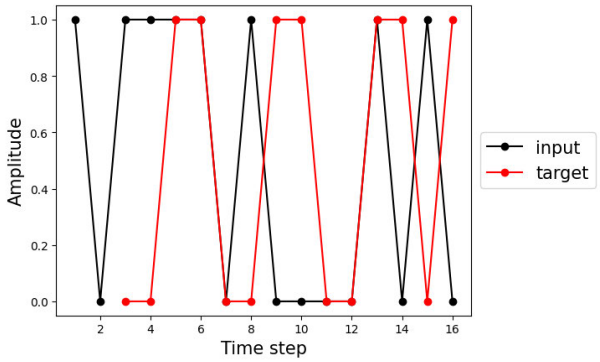

FIGURE 8. Task A: Three-bit parity check.

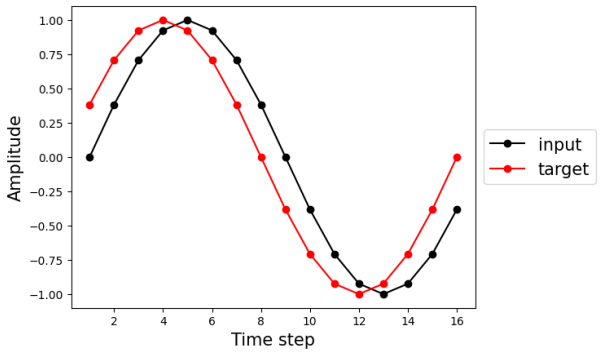

FIGURE 9. Task B: Sinusoidal wave prediction.

signals and T steps target signals are given to the model, T steps reservoir states $[\boldsymbol{x}(1), \boldsymbol{x}(2), \ldots, \boldsymbol{x}(T)]$ are obtained. Here, all reservoir states $[\boldsymbol{x}(1), \boldsymbol{x}(2), \ldots, \boldsymbol{x}(T)]$ and target signals $[z(1), z(2), \ldots, z(T)]$ are stocked in the memory for the winner readout, whereas $\lfloor h_i T \rfloor$ steps reservoir states and the target signals are stocked in the memories for the remaining readouts.

Fig. 7 shows three methods to adjust the amount of training data in the batch training investigated in this study. In the method shown in Fig. 7(a), first $\lfloor h_i T \rfloor$ steps reservoir states $[\boldsymbol{x}(1), \ldots, \boldsymbol{x}(\lfloor h_i T \rfloor)]$ and the target signals $[z(1), \ldots, z(\lfloor h_i T \rfloor)]$ are stocked in the memory. In the method shown in Fig. 7(b), the middle of reservoir states $[\boldsymbol{x}(\lfloor \frac{1}{2}(1 - h_i)T \rfloor), \ldots, \boldsymbol{x}(\lfloor \frac{1}{2}(1 + h_i)T \rfloor)]$ and target signals $[z(\lfloor \frac{1}{2}(1 - h_i)T \rfloor), \ldots, z(\lfloor \frac{1}{2}(1 + h_i)T \rfloor)]$ are cropped and stocked in the memory. In the method shown in Fig. 7(c), the reservoir states and target signals are decimated to get $\lfloor h_i T \rfloor$ step samples. Here, one reservoir state and its target signal in every T_{dec} time step is picked up and stored in the memory. T_{dec} is computed using Eq. (13).

$$T_{dec} = \begin{cases} \lfloor 1/h_i \rfloor & (h_i \geq 1/T) \\ T & (h_i < 1/T) \end{cases} \tag{13}$$

After feeding the training data to the model, each weight connection is computed by the ridge regression shown in Eq. (4) using each stocked data. Because this algorithm updates the readout weight connections once, the parameters need not be scheduled.

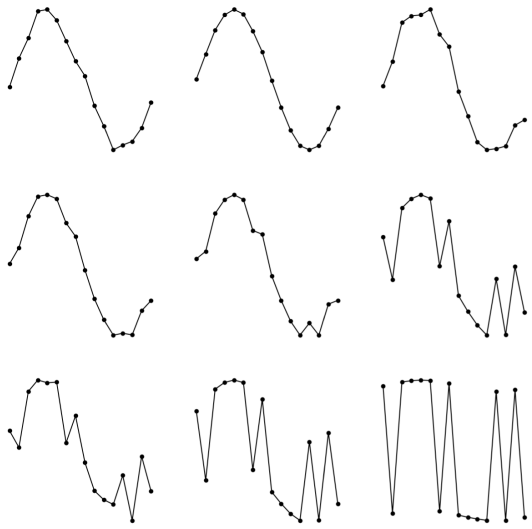

FIGURE 10. SOM reference vectors after training for Tasks A and B.

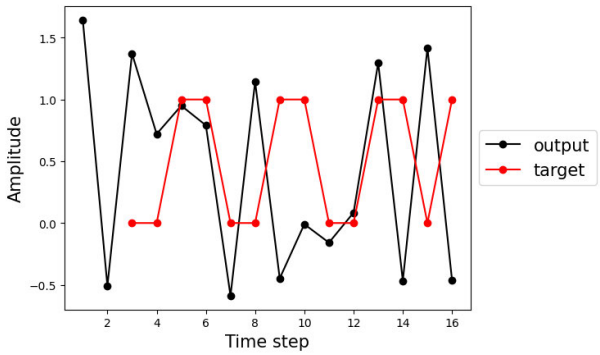

FIGURE 11. Output of Task A using the normal ESN.

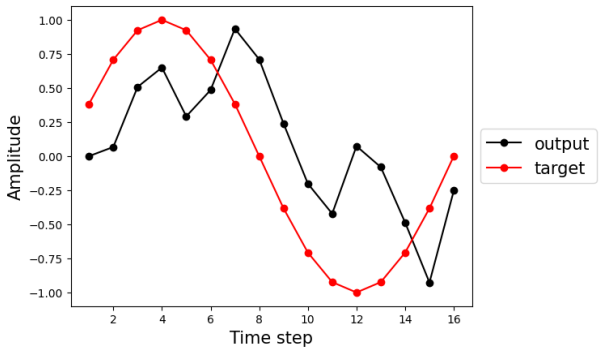

FIGURE 12. Output of Task B using the normal ESN.

IV. EXPERIMENT

A. EVALUATION OF ONLINE TRAINING

We conducted an experiment to evaluate the proposed RC model's performance with the online training algorithm.

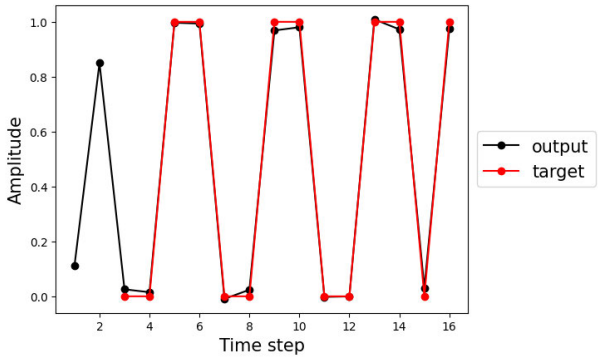

FIGURE 13. Output of Task A using the proposed method.

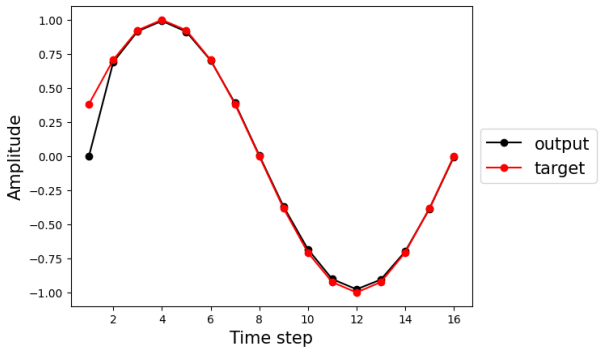

FIGURE 14. Output of Task B using the proposed method.

Here, we prepared two sets of input and target data that were inspired by a continuous learning task in [33], in which 16-step time-series inputs were given to the RC model.

One of them was a three-bit parity check task (named Task A), as shown in Fig. 8, where the input value $u(t)$ (drawn as a black line) was randomly selected from $\{0, 1\}$ and the target value $z(t)$ (drawn as a red line) was computed using Eq. (14):

$$z(t) = \{\sum_{i=0}^{2} u(t-i)\} \bmod 2 \tag{14}$$

such that $N_{in} = 1$, $N_{out} = 1$, and $T = 16$. Note that the target signals at $t = 1, 2$ is not plotted in the figure because they cannot be defined. The other task was a sinusoidal wave prediction task (named Task B), as shown in Fig. 9, where the input value $u(t)$ (drawn as a black line) and the target value $z(t)$ (drawn as a red line) were computed using Eqs. (15) and (16):

$$u(t) = \sin(\pi t/17) \tag{15}$$

$$z(t) = u(t+1) \tag{16}$$

where N_{in}, N_{out}, and T were the same as Task A.

In this experiment, we first fed the time-series data of Task A into the RC model. Here, as a vector representing the task feature, a concatenated vector $\boldsymbol{v} = [u(1), u(2), \ldots, u(16)]^\top$ was fed to the SOM such that $M =$

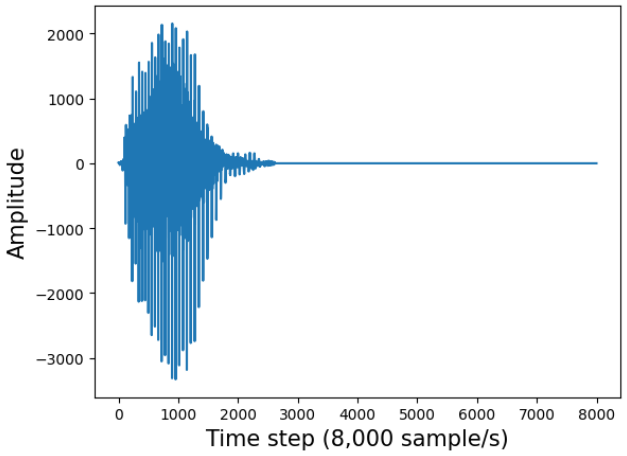

FIGURE 15. Input signal example from FSDD.

16, and the winner node and readout were determined by Eq. (7). Based on the determined winner, we executed the online training algorithm of Eq. (12). We also executed the unsupervised competitive learning of the SOM of Eq. (9). After the training of Task A, we fed the data for Task B and executed the training algorithm in the same way as Task A. Here, the number of nodes in the reservoir layer N_{res} was 100 and the grid size of the SOM was 3×3. The weight connections between the input and reservoir layers W_{in} were randomly generated from a uniform distribution of $[-1, 1]$. The recurrent connections in the reservoir layer W_{res} were generated from a uniform distribution of $[-1.5, 1.5]$, and then 50% of the connections were randomly selected and set to zero. The hyperbolic tangent function was used as the nonlinear function f. The leak rate of the reservoir layer α was 0.95, and the regularization term coefficient of the ridge regression λ was 0.01. The learning rate of the SOM γ was 0.5, the width of the neighborhood coefficient σ was 0.8, and the learning coefficient of the multiple readouts β was 0.95, which were fixed (not scheduled) at this time because the tasks were simple. We repeated this procedure ten times. Additionally, we conducted the same evaluation using a normal ESN with 100 reservoir nodes.

Fig. 10 shows the reference vectors of the SOM after training, which are aligned on the 3×3 grid. Figs. 11 and 12 show the outputs (drawn as a black line) and targets (drawn as a red line) of Tasks A and B for the normal ESN, respectively. Because of the catastrophic forgetting, the outputs did not follow the target signals in both tasks. Conversely, the proposed RC model predicted the outputs with smaller loss compared with the normal ESN in both tasks, as shown in Figs. 13 and 14. This result reveals that the proposed model and algorithm can avoid catastrophic forgetting to complete the continuous learning task.

B. EVALUATION OF BATCH TRAINING

We also conducted an experiment to evaluate the proposed RC model's performance with the batch training algorithm

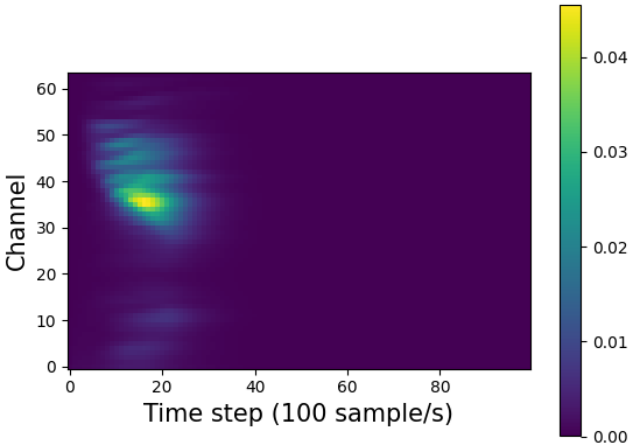

FIGURE 16. Cochleagram of the signal depicted in Fig. 15.

TABLE 1. Accuracy comparison: ESN with a single readout vs. ESN with multiple readouts in a speaker classification task.

	ESN with single readout	ESN with multiple readouts		
		(a)	(b)	(c)
$N_{\mathrm{res}} = 100$	86.6%	94.7%	95.9%	**96.0%**
$N_{\mathrm{res}} = 250$	93.5%	96.4%	**97.6%**	97.3%
$N_{\mathrm{res}} = 500$	94.5%	97.8%	97.9%	**98.2%**

TABLE 2. Accuracy comparison: ESN with a single readout vs. ESN with multiple readouts in a digit classification task.

	ESN with single readout	ESN with multiple readouts		
		(a)	(b)	(c)
$N_{\mathrm{res}} = 100$	78.7%	83.0%	85.3%	**87.5%**
$N_{\mathrm{res}} = 250$	85.3%	90.1%	90.7%	**92.3%**
$N_{\mathrm{res}} = 500$	89.8%	92.5%	93.6%	**94.8%**

using the free-spoken digit dataset (FSDD) [34]. This dataset comprises 3,000 audio files of English digits (“zero” to “nine”) pronounced by six persons and recorded at 8 kHz (50 audio files of each digit per person is included). We divided the dataset into training and test data, with 90% of the data being used for training and 10% for testing. Although the audio data time step length of FSDD varies (its mean is 3499.5 samples and its standard deviation is 1180.9 samples), all utterance sections are within the first 8,000 samples, and therefore the first 8,000 samples of audio data were used in this experiment. It should be noted that if the length of the audio data was less than 8,000 samples, the remaining sections were zero-filled to contain 8,000 samples.

Before feeding audio data from the FSDD to the model, we used Lyon’s auditory model [35] to convert it into a cochleagram, which is a time series of intensities of quantized frequency channels. In this experiment, each cochleagram had 64 frequency channels and 100 time steps. The sampling rate of the cochleagram was 100.0 sample/s (with every 79 samples decimated out of every 80 samples), resulting in $N_{in} = 64$, $T = 100$. Figs. 15 and 16 show an example of a FSDD input signal and its cochleagram, respectively. Note that because the utterance section of the input signal ends approximately at the 2,500 th step, the remaining section is zero-filled, as shown in Fig. 15.

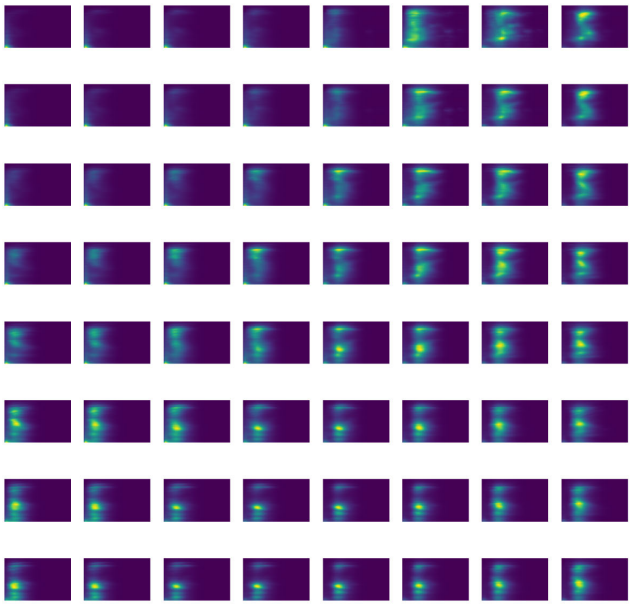

FIGURE 17. Trained SOM reference vectors for cochleagrams.

Here, we constructed a SOM with a grid size of 8×8 and fed training data from the FSDD to the SOM for pretraining. The input signals obtained from the FSDD were concatenated using Eq. (8), with the specific purpose of aligning the dimensionality, resulting in $M = N_{in}T$, equaling 6,400. The SOM’s learning rate γ and neighborhood coefficient width σ varied in time during the SOM training: the learning rate γ started from 0.1 and decreased monotonously to 0.001, and the width σ started from 3.0 and decreased monotonously to 0.1. Fig. 17 shows the SOM’s reference vectors after training, which are aligned on the 8×8 grid.

We conducted two sound classification tasks using the FSDD: a speaker and a digit classification. We fed training data from the FSDD to the reservoir layer and stocked the reservoir states with the target signals based on the neighborhood coefficient h_i in the memories. Here, we used the pretrained SOM shown in Fig. 17 to determine h_i and set the parameter σ as 1.0. The weight connections between the input and reservoir layers W_{in} were randomly generated from a uniform distribution of $[-30, 30]$. The recurrent connections in the reservoir layer W_{res} were generated from a uniform distribution of $[-0.999, 0.999]$, and then 50% of the connections were randomly selected and set to zero. The hyperbolic tangent function was used as the nonlinear function f. The reservoir’s leak rate α was 0.5. After feeding the training data, we computed the weight connections of the multiple readouts by the ridge regression setting λ as 0.1. In this experiment, we investigated the test accuracy when the number of reservoir nodes N_{res} was set to 100, 250, and 500, and compared the accuracy between an ESN with a single readout and an ESN with multiple readouts. We also investigated and compared the effectiveness of the

TABLE 3. Accuracy rates of the SVM and the LSTM-based network in the speaker and digit classification tasks.

	Speaker	Digit
SVM	95.3%	64.7%
LSTM-based network (100 nodes)	**99.0%**	89.5%
LSTM-based network (250 nodes)	97.9%	**91.3%**
LSTM-based network (500 nodes)	95.3%	**91.3%**

three methods to adjust the amount of training data, shown in Fig. 7.

Tables 1 and 2 show the accuracy achieved by the ESN with a single readout and the ESN with multiple readouts in the speaker and digit classification tasks, respectively. Each accuracy is an average of five trials. The ESN with multiple readouts outperformed the ESN with a single readout.

We also investigated the performance of a support vector machine (SVM) [36] and a long short-term memory (LSTM)-based neural network [4] for both tasks. The LSTM-based network consisted of an LSTM layer and a linear layer and we set the number of nodes in the LSTM layer as 100, 250, or 500. The training of the LSTM-based network was conducted for 5,000 epochs and the optimization algorithm was Adam. The accuracy rates of the SVM and the LSTM-based network, which are averages of five trials, are shown in Table 3.

V. CONCLUSION

This study presents an innovative approach to enhance the training capabilities of an RC model through the integration of multiple readouts for distributed readout training. The effectiveness of this approach was demonstrated through a continuous learning task in which the proposed model successfully avoided catastrophic forgetting, a challenge that conventional ESNs struggled with. Experiment A showed that the proposed model distributed tasks using the SOM, as shown in Fig. 10, to avoid catastrophic forgetting and accomplished the task as shown in Figs. 13 and 14, while the conventional ESN was unable to accomplish the task as shown in Figs. 11 and 12. Additionally, a sound classification task utilizing the FSDD was performed. Experiment B showed that the proposed model distributed training data from FSDD using the SOM, as shown in Fig. 17, to make readouts easily adapt data and outperformed conventional ESN as shown in Tables 1 and 2.

A key advantage of the proposed method is its low training cost. While Kawai et al. also proposed a reservoir-based approach [33] for continuous learning, their method employed a genetic algorithm for network optimization, necessitating extensive training repetition. Conversely, the proposed method required ten iterations for training in the case of the task shown in Figs. 8 and 9.

In our experiment, we employed a 3×3 SOM configuration for both Tasks A and B, which may seem redundant for solving two distinct tasks. However, it led to the observation of an intriguing property within the neighborhoods of the winner readouts. Despite the absence of explicit data input,

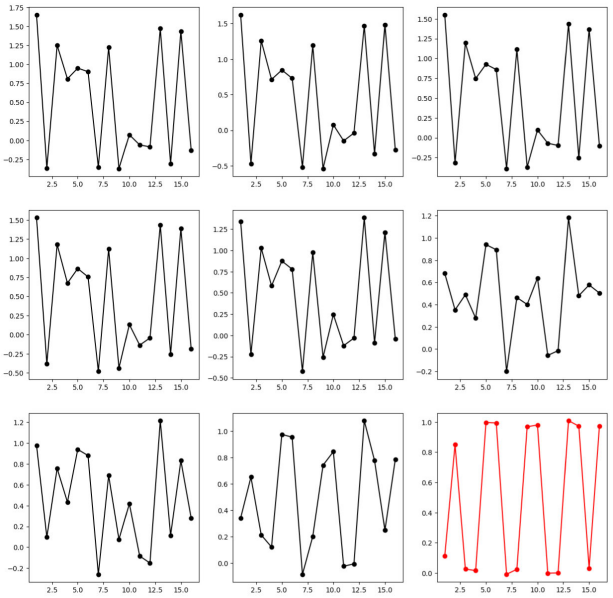

FIGURE 18. Task A readout responses.

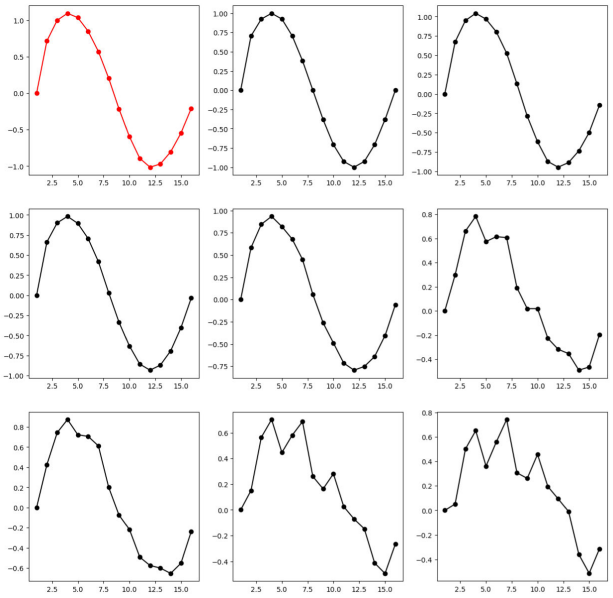

FIGURE 19. Task B readout responses.

the SOM nodes displayed intermediate reference vectors between the inputs of the given tasks, as depicted in Fig. 10. Moreover, the responses of the neighborhood readouts demonstrated an intermediate nature between the outputs of the given tasks. Figs. 18 and 19 show responses from all readouts aligned on the 3×3 grid during the execution of Tasks A and B, respectively. The graph, delineated by a red line, serves to indicate the response generated by the winner readout. Examining these figures reveals that the neighboring readouts displayed a discernible degree of alignment with the characteristics of the given tasks. These results can be regarded as the neighboring readouts virtually learned tasks, except for Tasks A and B. Therefore, transfer learning may be

possible with a few adjustments using the weight connections of the neighborhood readouts as initial values.

Tables 1 and 2 show that Method (c) showed the best performance among the three methods in adjusting the amount of training data during batch training. By using Methods (b) and (c), the readouts could learn various time steps of reservoir states, whereas Method (a) focused only on the first parts of reservoir states. Thus, the performances of Methods (b) and (c) were superior to that of Method (a). Interestingly, the performance improvements achieved through the introduction of the proposed method further improved when the number of reservoir nodes N_{res} reduced. We therefore conclude that the proposed method is effective for RCs with few reservoir nodes.

Tables 1, 2, and 3 show that the proposed network was inferior to the LSTM-based network in the speaker classification task. However, the LSTM-based network required at least 900 epochs to achieve more than an accuracy rate of 95% in the experiment, whereas the proposed network was optimized with one epoch, and therefore, the proposed network is more suitable for training in edge computing than the LSTM-based network. A possible reason for the performance of the LSTM-based network with 100 nodes was better than that with 250 and 500 nodes in the speaker classification task is overfitting which happened as the complexity of the network increased.

The versatility of the proposed RC model extends beyond sound recognition tasks, making it applicable to other domains, including reinforcement learning [37]. Integrating the proposed structure in a reservoir-based reinforcement learning model [38] could enhance its performance by avoiding catastrophic forgetting and enabling transfer learning from previously acquired knowledge.

Although our implementation of the proposed RC model was based on ESNs, the method is not exclusive to this architecture and can be adapted to other reservoir implementations. This versatility is particularly promising for enhancing the training capabilities of physical RCs, such as nanomaterial-based RCs [25].

REFERENCES

[1] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.

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

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, vol. 1, 2012, pp. 1097–1105.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," in *Proc. NIPS Deep Learning Workshop*, 2013.

[6] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8024–8035.

[7] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Oper. Syst. Design Implement. (OSDI)*, 2016, pp. 265–283.

[8] R. Okuta, Y. Unno, D. Nishino, S. Hido, and C. Loomis, "CuPy: A numpy-compatible library for NVIDIA GPU calculations," in *Proc. ML Syst. Workshop NIPS*, 2017.

[9] L. Iocchi, D. Holz, J. Ruiz-del-Solar, K. Sugiura, and T. van der Zant, "RoboCup@home: Analysis and results of evolving competitions for domestic and service robots," *Artif. Intell.*, vol. 229, pp. 258–281, Dec. 2015.

[10] T. Yamamoto, K. Terada, A. Ochiai, F. Saito, Y. Asahara, and K. Murase, "Development of human support robot as the research platform of a domestic mobile manipulator," *ROBOMECH J.*, vol. 6, no. 1, Dec. 2019.

[11] T. Ono, D. Kanaoka, T. Shiba, S. Tokuno, Y. Yano, A. Mizutani, I. Matsumoto, H. Amano, and H. Tamukoh, "Solution of world robot challenge 2020 partner robot challenge (real space)," *Adv. Robot.*, vol. 36, nos. 17–18, pp. 870–889, Sep. 2022.

[12] H. Jaeger, "The 'echo state' approach to analysing and training recurrent neural networks-with an erratum note," German Nat. Res. Center Inf. Technol. GMD, Bonn, Germany, Tech. Rep., 2001, vol. 148, no. 34.

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

[14] M. L. Alomar, V. Canals, N. Perez-Mora, V. Martínez-Moll, and J. L. Rosselló, "FPGA-based stochastic echo state networks for time-series forecasting," *Comput. Intell. Neurosci.*, vol. 2016, pp. 1–14, Jan. 2016.

[15] L. Loomis, N. McDonald, and C. Merkel, "An FPGA implementation of a time delay reservoir using stochastic logic," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 14, no. 4, pp. 1–15, Oct. 2018.

[16] N.-S. Huang, J.-M. Braun, J. C. Larsen, and P. Manoonpong, "A scalable echo state networks hardware generator for embedded systems using high-level synthesis," in *Proc. 8th Medit. Conf. Embedded Comput. (MECO)*, Jun. 2019, pp. 1–6.

[17] M. L. Alomar, E. S. Skibinsky-Gitlin, C. F. Frasser, V. Canals, E. Isern, M. Roca, and J. L. Rosselló, "Efficient parallel implementation of reservoir computing systems," *Neural Comput. Appl.*, vol. 32, no. 7, pp. 2299–2313, Apr. 2020.

[18] K. Honda and H. Tamukoh, "A hardware-oriented echo state network and its FPGA implementation," *J. Robot., Netw. Artif. Life*, vol. 7, no. 1, pp. 58–62, 2020.

[19] I. Kawashima, Y. Katori, T. Morie, and H. Tamukoh, "An area-efficient multiply-accumulation architecture and implementations for time-domain neural processing," in *Proc. Int. Conf. Field-Program. Technol. (ICFPT)*, Dec. 2021, pp. 1–4.

[20] K. Nakajima, H. Hauser, T. Li, and R. Pfeifer, "Information processing via physical soft body," *Sci. Rep.*, vol. 5, no. 1, pp. 1–11, May 2015.

[21] G. Van der Sande, D. Brunner, and M. C. Soriano, "Advances in photonic reservoir computing," *Nanophotonics*, vol. 6, no. 3, pp. 561–576, May 2017.

[22] J. Torrejon, M. Riou, F. A. Araujo, S. Tsunegi, G. Khalsa, D. Querlioz, P. Bortolotti, V. Cros, K. Yakushiji, A. Fukushima, H. Kubota, S. Yuasa, M. D. Stiles, and J. Grollier, "Neuromorphic computing with nanoscale spintronic oscillators," *Nature*, vol. 547, no. 7664, pp. 428–431, Jul. 2017.

[23] 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.*, vol. 115, pp. 100–123, Jul. 2019.

[24] K. Nakajima, "Physical reservoir computing—An introductory perspective," *Jpn. J. Appl. Phys.*, vol. 59, no. 6, May 2020, Art. no. 060501.

[25] Y. Usami, B. van de Ven, D. G. Mathew, T. Chen, T. Kotooka, Y. Kawashima, Y. Tanaka, Y. Otsuka, H. Ohoyama, H. Tamukoh, H. Tanaka, W. G. van der Wiel, and T. Matsumoto, "In-materio reservoir computing in a sulfonated polyaniline network," *Adv. Mater.*, vol. 33, no. 48, Dec. 2021, Art. no. 2102688.

[26] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[27] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Netw.*, vol. 113, pp. 54–71, May 2019.

[28] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of Learning and Motivation*, vol. 24, G. H. Bower, Ed. New York, NY, USA: Academic, 1989, pp. 109–165.
[29] Y. Tanaka and H. Tamukoh, "Ensemble learning of multiple readouts for reservoir computing," in *Proc. Int. Symp. Nonlinear Theory Appl. (NOLTA)*, 2023, pp. 509–512. [Online]. Available: https://www.ieice.org/publications/proceedings/summary.php?expandable=13&iconf=NOLTA&session_num=C3L-1&number=C3L-11&year=2023
[30] H. Jaeger, M. Lukoševičius, D. Popovici, and U. Siewert, "Optimization and applications of echo state networks with leaky- integrator neurons," *Neural Netw.*, vol. 20, no. 3, pp. 335–352, Apr. 2007.
[31] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biol. Cybern.*, vol. 43, no. 1, pp. 59–69, 1982.
[32] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 86, pp. 2579–2605, 2008.
[33] Y. Kawai, Y. Ozasa, J. Park, and M. Asada, "Avoiding catastrophic forgetting in echo state networks by minimizing the connection cost," in *Proc. 33rd Annu. Conf. Jpn. Sociery Artif. Intell.*, 2019.
[34] Z. Jackson, C. Souza, J. Flaks, Y. Pan, H. Nicolas, and A. Thite, "Jakobovski/free-spoken-digit-dataset," Tech. Rep., 2018, doi: 10.5281/zenodo.1342401.
[35] R. Lyon, "A computational model of filtering, detection, and compression in the cochlea," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 7, May 1982, pp. 1282–1285.
[36] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Jan. 1995.
[37] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 2018.
[38] M. Inada, Y. Tanaka, H. Tamukoh, K. Tateno, T. Morie, and Y. Katori, "A reservoir based Q-learning model for autonomous mobile robots," in *Proc. Int. Symp. Nonlinear Theory Appl. (NOLTA)*, 2020, pp. 213–216.

YUICHIRO TANAKA (Member, IEEE) received the B.E., M.E., and Ph.D. degrees from the Kyushu Institute of Technology, in 2016, 2018, and 2021, respectively. He was a Research Fellow with the Japan Society for the Promotion of Science (JSPS), from 2019 to 2021. He has been an Assistant Professor with the Research Center for Neuromorphic AI Hardware, Kyushu Institute of Technology, since 2021. His research interests include neural networks, digital hardware implementation, and home service robots. He is a member of IEICE and JNNS.

HAKARU TAMUKOH (Member, IEEE) received the B.Eng. degree from Miyazaki University, Japan, in 2001, and the M.Eng. and Ph.D. degrees from the Kyushu Institute of Technology, Japan, in 2003 and 2006, respectively. He was a Post-doctoral Research Fellow with the 21st Century Center of Excellent Program, Kyushu Institute of Technology, from April 2006 to September 2007. He was an Assistant Professor with the Tokyo University of Agriculture and Technology, from October 2007 to January 2013. He is currently a Professor with the Graduate School of Life Science and Systems Engineering, Kyushu Institute of Technology. His research interests include hardware/software complex systems, digital hardware design, neural networks, soft computing, and home service robots. He is a member of IEICE, SOFT, JNNS, JSAI, and RSJ.