



Learning reservoir dynamics with temporal self-modulation

Yusuke Sakemi ^{1,2}✉, Sou Nobukawa^{1,3,4}, Toshitaka Matsuki⁵, Takashi Morie ⁶ & Kazuyuki Aihara^{1,2}

Reservoir computing (RC) can efficiently process time-series data by mapping the input signal into a high-dimensional space via randomly connected recurrent neural networks (RNNs), which are referred to as a reservoir. The high-dimensional representation of time-series data in the reservoir simplifies subsequent learning tasks. Although this simple architecture allows fast learning and facile physical implementation, the learning performance is inferior to that of other state-of-the-art RNN models. In this study, to improve the learning ability of RC, we propose self-modulated RC (SM-RC) that extends RC by adding a self-modulation mechanism. SM-RC can perform attention tasks where input information is retained or discarded depending on the input signal. We find that a chaotic state can emerge as a result of learning in SM-RC. Furthermore, we demonstrate that SM-RC outperforms RC in NARMA and Lorenz model tasks. Because the SM-RC architecture only requires two additional gates, it is physically implementable as RC, thereby providing a direction for realizing edge artificial intelligence.

¹Research Center for Mathematical Engineering, Chiba Institute of Technology, Narashino, Japan. ²International Research Center for Neurointelligence (WPI-IRCN), The University of Tokyo, Tokyo, Japan. ³Department of Computer Science, Chiba Institute of Technology, Narashino, Japan. ⁴Department of Preventive Intervention for Psychiatric Disorders, National Institute of Mental Health, National Center of Neurology and Psychiatry, Tokyo, Japan. ⁵National Defense Academy of Japan, Kanagawa, Japan. ⁶Graduate School of Life Science and Systems Engineering, Kyushu Institute of Technology, Kitakyushu, Japan. ✉email: yusuke.sakemi@p.chibakoudai.jp

Vast amounts of data are generated and observed in the form of time series in the real world. Efficiently processing these time-series data is important in real-world applications such as forecasting the renewable energy supply and monitoring sensor data in factories. In the past decade, data-driven methods based on deep learning have progressed significantly and have successfully linked data prediction and analysis to social values, and they are becoming increasingly important^{1,2}. However, the computational load of data-driven methods results in considerable energy consumption, thus limiting their applicability^{3,4}. In particular, to perform prediction and analysis near the location where the data are generated, which is called edge artificial intelligence (AI), high energy efficiency is required⁵.

Reservoir computing (RC) is attracting attention as a candidate for edge AI because it achieves high prediction performance and high energy efficiency. The RC model consists of an input layer, a reservoir layer, and an output layer^{6,7}. The reservoir layer is typically a recurrent neural network (RNN) with fixed random weights. Because only the output layer is usually trained in RC, the training process is faster than those of other RNN models such as long short-term memory (LSTM)⁸ and gated recurrent units (GRUs)⁹. In addition, because the reservoir layer can be configured with various dynamical systems¹⁰, its high energy efficiency has been demonstrated through physical implementations^{11,12}.

The computational power of RC increases upon using larger-sized reservoirs¹³. However, the reservoir size is often limited by the size of physical systems^{11,12}. Furthermore, as only the output layer is trained in RC, it is unclear whether it can achieve comparable prediction accuracy for real-world applications to other state-of-the-art approaches, given the same energy consumption¹⁴. To improve the computational efficiency of RC, various RC architectures and methods have been proposed¹⁵. Recent proposals include structures that combine convolutional neural networks¹⁶, parallel reservoirs^{17–19}, multilayer (deep) RC²⁰, methods that use information from past reservoir layers²¹, and regularization methods that combine autoencoders²². These studies indicated that the performance can be improved by using an appropriate reservoir structure. However, because only the output layer is trained in these methods, the performance improvement is limited.

One promising approach for improving the flexibility of information processing in RC is to temporally vary the dynamical properties of the reservoir layer to adapt to the input signal. An architecture that feeds the output back to the reservoir can realize this^{23,24}. Sussilo and Abbott proposed the FORCE learning method for stably training an RC model with a feedback architecture and reported that the network can learn various types of autonomous dynamics²³. This architecture has been extended to spiking neural networks²⁵ and applied to reinforcement-learning tasks²⁶. However, although these feedback connections are thought to control the dynamics according to tasks²⁴, their control is limited because the connections are random.

Research has also been conducted to acquire task-dependent dynamics in the reservoir layer by training not only the output layer but also the reservoir layer. Intrinsic plasticity²⁷ is a method for making the outputs of neurons closer to a desired distribution, and a Hebbian rule or anti-Hebbian rule²⁸ allows control of correlations between neurons. These methods increased the prediction accuracy and memory capacity²⁹. Lage and Buonomano proposed innate training, which realizes a long-term memory function by training some connections in the reservoir layer to construct an attractor that is stable for a certain period of time³⁰. Inoue et al. used this method to construct chaotic itinerancy³¹. The aforementioned studies demonstrated that it is

possible to perform tasks that are difficult to achieve with conventional RC models by training the reservoir layer. However, in all the methods used, the properties of the trained reservoir layer were static (e.g., fixed network connections in time), thus limiting the diversity of the dynamics.

Recently, the attention mechanism has been considered as an effective method for realizing information processing adapted to the input signal. In deep learning, the introduction of the attention mechanism was one of the breakthrough techniques proposed in recent years^{32,33}. The introduction of this mechanism allows efficient learning through the selection and processing of important information, and it has impacted various research fields such as natural language processing^{34,35}. In neuroscience, attention is considered an important factor for realizing cognitive functions, and neuromodulation is known as a neural mechanism that is closely related to attention³⁶. Neuromodulation is caused by neurons in brain areas such as the basal forebrain releasing neuromodulators such as acetylcholine into various brain areas to modulate the activity of neurons therein^{36–38}. The attention mechanism can increase the efficiency of information processing. However, in RC, the input signal is uniformly transferred to the reservoir layer and converted into high-dimensional features; thus, there is no attention mechanism.

In this study, we propose the self-modulated RC (SM-RC) architecture that incorporates the advantages of the aforementioned feedback structure, reservoir-layer learning, and attention mechanism. SM-RC has trainable gates that can dynamically modulate the strength of input signals and the dynamical properties of the reservoir layer. Thus, it is possible to learn the reservoir dynamics adapted to the input signal, thereby enabling information processing such as attention and improving the learning performance for a wide range of tasks. Importantly, the gate structure has at most two variables and is controlled by feedback from the reservoir layer; thus, we expect SM-RC to be highly hardware-implementable, similar to conventional RC. The SM-RC architecture provides another option for improving the learning performance of physical RC systems without increasing the size of the reservoir. Below, to promote a new direction of physical reservoir computing research, we investigate the learning performance and model characteristics of SM-RC. In particular, we compare the prediction performance with that of conventional RC through simple attention, NARMA, and Lorenz model tasks. We also discuss prospects for hardware implementation.

Results

Figure 1 shows the SM-RC architecture. In addition to the input layer, reservoir layer, and output layer, the proposed architecture has an input gate g^{in} that modulates the input signal and a reservoir gate g^{res} that changes the dynamical properties of the reservoir layer, both of which are controlled by feedback from the reservoir layer.

These properties extend the functionality of conventional RC, wherein input information is “uniformly” transferred to the reservoir layer. In addition, the information of an input signal stored in the reservoir layer tends to decay over time, which is related to the echo-state property and fading memory^{10,39}. These properties of conventional RC do not pose a significant problem for relatively simple tasks; however, they severely limit the regression performance for tasks that involve capturing long or complex temporal dependencies²¹. SM-RC overcomes the shortcomings of conventional RC by introducing the self-modulation mechanism.

In this study, for simplicity, we construct SM-RC on the basis of the echo-state network (ESN)^{6,13}, which is the most commonly used form of RC. The ESN is a discrete-time dynamical system,

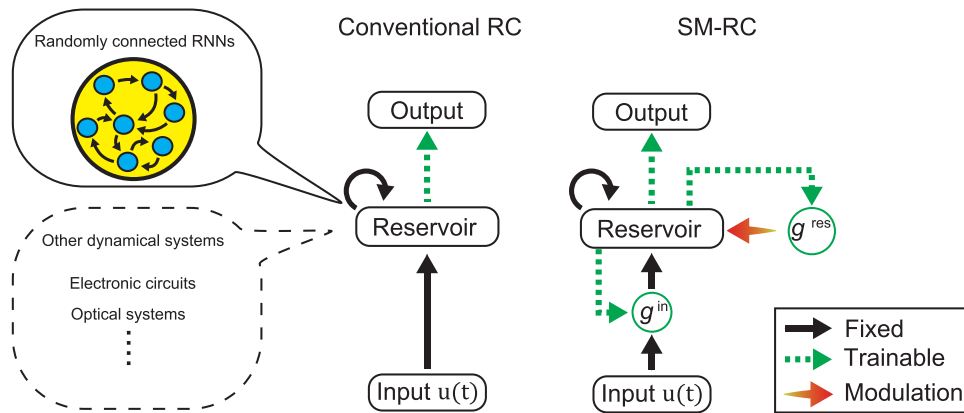


Fig. 1 Comparison of the conventional reservoir computing (RC) and self-modulated RC (SM-RC) architectures. In conventional RC, the input signal is directly transferred to the reservoir layer, and the output is obtained from the reservoir layer. The reservoir layer is typically constructed with a recurrent neural network (RNN) having fixed weights; however, various dynamical systems can be used as reservoirs, and they can be implemented with electronic circuits, optical systems, and other physical systems. SM-RC inherits the basic architecture of conventional RC, including the reservoir layer and output layer. However, the input signal is modulated by the input gate g^{in} before being transferred to the reservoir. Additionally, the reservoir dynamics are modulated with the reservoir gate g^{res} . Both gates are controlled by the feedback from the reservoir layer. The black solid arrows and green dotted arrows represent the randomly initialized fixed connections and trainable connections, respectively. The red gradient arrow represents the function that changes the dynamical properties of the reservoir.

and the reservoir layer consists of an RNN with fixed weights. In addition, we consider an input gate that dynamically changes the input strength and a reservoir gate that dynamically changes the internal connection strength in the reservoir layer. In this case, the input \mathbf{u}^{res} to the reservoir layer and the spectral radius ρ^{res} of the reservoir layer are time-modulated as follows:

$$\mathbf{u}^{\text{res}}(t) = g^{\text{in}}(t-1)\mathbf{u}(t), \quad (1)$$

$$\rho^{\text{res}}(t) = g^{\text{res}}(t-1)\hat{\rho}^{\text{res}}, \quad (2)$$

where $\mathbf{u}(t)$ is the original input vector and $\hat{\rho}^{\text{res}}$ is the spectral radius of the inner connection matrix of the unmodulated reservoir layer. Note that g^{in} and g^{res} are scalar gates trained by nonlinear optimization. From the function of these gates, we can intuitively understand how SM-RC extends RC. For example, the input gate can realize attention by sending important input signals to the reservoir layer and discarding other information, and the reservoir gate can change the memory retention characteristics depending on the input signal.

To evaluate the learning performance of SM-RC from various viewpoints, we compared it with that of conventional RC for three tasks with different characteristics: simple attention, NARMA, and Lorenz model tasks. In all the experiments, the number of neurons in the SM-RC reservoir layer N^{res} was set as 100, and the hyperparameters were fixed to the same values for SM-RC. In contrast, conventional RC used reservoir layers with various numbers of neurons, and the hyperparameters were optimized. The details of the models and their learning procedure are presented in Methods.

Simple attention tasks. To investigate the self-modulation mechanism of SM-RC, we evaluated the learning performance for a simple attention task. In this task, the input signal was $\mathbf{u}(t) = 1$ in the time interval of [250, 259], and Gaussian noise with a mean of 0 and standard deviation of σ^{in} was added at other timesteps. The model was trained to output 1 in the time interval of [290, 291] and output 0 at other timesteps. To cancel the effect of initial values of $x_i(0) = 0$, the first 200 steps were discarded (free run), and common jitter noise was added to the input and output time windows (see Methods for details). This task required a memory retention function that retained

the input signal for a long time while reducing the influence of Gaussian noise when no informative input signals were provided.

Figure 2 shows the learning results for the simple attention task. In the case of conventional RC (Fig. 2a), for $\sigma^{\text{in}} = 0.1$, the regression of the output pulse was poor. Although it has been reported that RC has a long-term memory function⁴⁰, memory retention is difficult when noise is continuously added to the reservoir layer. In contrast, for SM-RC (Fig. 2b), even when the noise was intensified (i.e., $\sigma^{\text{in}} = 0.3$), the regression of the output pulse was performed accurately. Figure 2c presents a comparison of the regression performance between conventional RC and SM-RC. SM-RC outperformed conventional RC for various input noise intensities σ^{in} . For the same input noise intensity, SM-RC with $N^{\text{res}} = 100$ outperformed conventional RC with a reservoir at least 10 times larger.

By examining the time evolution of the input gate $g^{\text{in}}(t)$ and the reservoir gate $g^{\text{res}}(t)$ (the figure shows the modulated spectral radius $\rho^{\text{res}}(t)$) (Fig. 2b), we can intuitively understand the factors contributing to successful regression in the case of SM-RC. The input gate autonomously takes large values during the time period when the input pulse arrives, thus efficiently feeding information into the reservoir layer. After the input pulse disappears, the input gate takes small values to prevent the input noise from corrupting the information stored in the reservoir layer. In contrast, the modulated spectral radius $\rho^{\text{res}}(t)$ takes large values after the input pulse disappears. Because the fading memory condition is not met when the spectral radius is sufficiently large^{10,13}, the reservoir layer can retain the information for a long period.

To further study the dynamics of SM-RC, we performed a sensitivity analysis of the reservoir layer. This was done by adding perturbation to the reservoir states and examining the evolution of the perturbation two steps forward ($t_p = 2$). The results for other values of t_p are presented in Supplementary Note 1. When the perturbation increases after t_p steps, the sensitivity has a positive value; otherwise, it has a negative value (see Methods for details). In Fig. 2a, the sensitivity of the conventional RC model exhibits only negative values during the simulation period. This indicates that the reservoir was in stable states, which is consistent with the fading memory condition that usually holds for

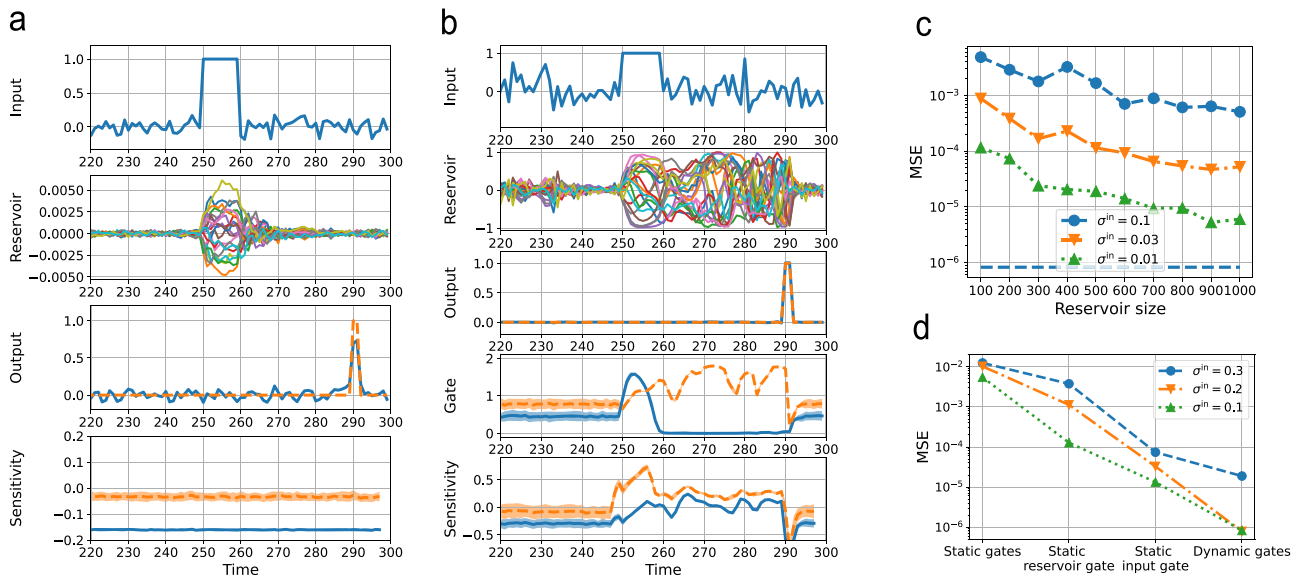


Fig. 2 Simulation results for simple attention tasks. **a** Dynamics of a conventional reservoir computing (RC) model after training with the reservoir size N^{res} of 100 and the input noise intensity σ^{in} of 0.1. From the top, the time evolution of the input signal, reservoir states, output, and sensitivity are shown. **b** Dynamics of the self-modulated RC (SM-RC) model after training with $N^{\text{res}} = 100$ and $\sigma^{\text{in}} = 0.3$. From the top, the time evolution of the input signal, reservoir states, outputs, gates, and sensitivity are shown. In the gate panel, the blue solid line represents the input gate, and the orange dashed line represents the spectral radius. The standard deviations of these gate values were obtained using 100 different input signals without jitter noise. We show the modulated spectral radius $\rho^{\text{res}}(t)$ obtained using Eq. (2). In **a** and **b**, in the sensitivity panel, the blue solid line indicates the mean sensitivity of the reservoir layer, and the orange dashed line indicates the maximum sensitivity of the reservoir layer. The standard deviations of the sensitivity values were obtained using 100 different input signals without jitter noise. For the output layer, the blue solid line indicates the predicted output, and the orange dashed line indicates the teacher signal. In the panels displaying the reservoir states, only 20 reservoir neurons are shown. **c** Comparison of regression errors between conventional RC models and SM-RC models. The regression errors for the conventional RC models are plotted as a function of the reservoir size for different input noise intensities (σ^{in}). The regression errors for the SM-RC models are represented by blue dashed horizontal lines for the input noise intensity of 0.1. For the SM-RC models, the reservoir size was fixed to 100. We present the lowest mean squared errors (MSEs) from 50 random weight initializations as the regression errors for both models. **d** Regression errors for the SM-RC models when one or both gates are made static.

conventional RC¹⁰. In contrast, for SM-RC (Fig. 2b), the sensitivity changed significantly over time and occasionally exhibited positive values. The positive sensitivity indicated that the reservoir was in chaotic states. Usually, chaotic states make regression difficult because the input-output relationship becomes sensitive to the initial conditions. However, in the case of SM-RC, by shutting the input gate, the problem related to the initial conditions was avoided for this attention task. We confirmed that the effects of Gaussian noise were negligible in the reservoir states after the input pulse and before the output pulse. The fact that the positive sensitivity only appeared after the input pulse and before the output pulse indicated that SM-RC successfully learned the complex reservoir dynamics adapted to the input signal.

To investigate the independent characteristics of the gates, we evaluated the regression performance when one or both gates did not evolve over time (see Methods for details). Figure 2d shows the results. For all input noise intensities, the performance was the best when both gates were time-evolved (dynamic gates), followed by the case where the reservoir gate was modulated (static input gate) and then the case where the input gate was modulated (static reservoir gate). The performance was the worst when both gates were static (static gates). This indicated that the observed performance improvement was caused not by the optimization of the spectral radius and input intensity but by their temporal modulation. In addition, the fact that the performance was the best when both gates were time-modulated indicated that the input gate and reservoir gate were cooperatively modulated, as intuitively explained above (see Supplementary Note 2 for a detailed discussion).

Time-series prediction: NARMA and Lorenz model. SM-RC improves the prediction performance even when there is no apparently salient time series to which attention should be paid, as in the case of simple attention tasks. To demonstrate this, we evaluated the performance of time-series prediction using NARMA and the Lorenz model. The NARMA time series is obtained with a nonlinear autoregressive moving average, which is often used to evaluate the learning performance of RC^{21,41,42}. In particular, we use the NARMA5 and NARMA10 time series, which have different internal dynamics. The Lorenz model is a chaotic dynamical system that is also used for RC performance evaluation owing to its difficulty of prediction^{14,22,43–45}. For the Lorenz model tasks, Gaussian noise with different standard deviations σ^{in} was added to the input to approximate the situation of real-world data. The models are trained to predict N^{forward} steps forward of the Lorenz model. The aforementioned tasks involved input signals with different characteristics: a uniform random number for the NARMA tasks and a chaotic time series for the Lorenz model tasks (see Methods).

Figure 3a–b shows the time evolution of the conventional RC and SM-RC models after they were trained on the NARMA10 task. Because the input signal for the task was uniform noise, the reservoir states evolved accordingly in the conventional RC and SM-RC models. In contrast to the case of the simple attention task, the gates in the SM-RC model did not change significantly over time, which reflected the characteristics of the input signal. The prediction performance of SM-RC was better than that of conventional RC, indicating that the self-modulation mechanism is effective even for uniform input signals. Figure 3c–d shows the time evolution of the conventional RC and SM-RC models

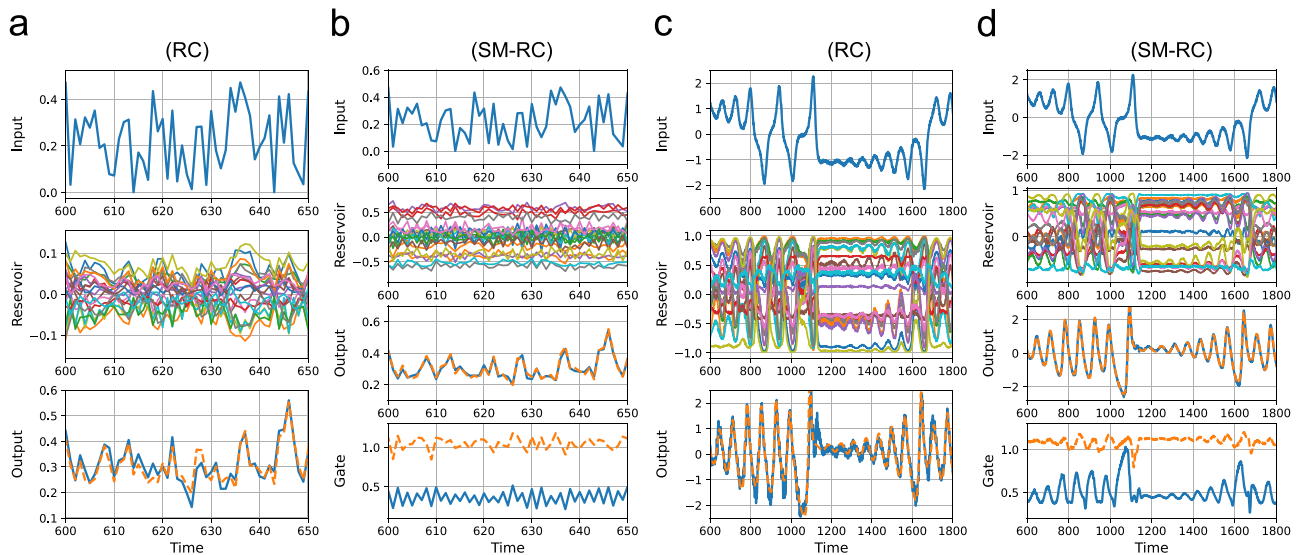


Fig. 3 Reservoir dynamics for NARMA10 and Lorenz model tasks. **a, b** Simulation results for NARMA10 tasks with the reservoir size N^{res} of 100. **c, d** Simulation results for Lorenz model tasks with $N^{\text{res}} = 100$, the input noise intensity σ^{in} of 0.03, and the prediction steps N^{forward} of 20. In **a** and **c**, the figures show the inputs, reservoir states, and outputs (from top to bottom) for the conventional reservoir computing (RC) models. In **b** and **d** the figures show the inputs, reservoir states, outputs, and gates (from top to bottom) for the self-modulated RC (SM-RC) models. For the gates, the solid blue lines represent the input gates, and the dashed orange lines indicate the modulated spectral radius. For the output layer, the solid blue line indicates the predicted output, and the dashed orange line indicates the teacher signal. In the panels displaying the reservoir states, only 20 reservoir neurons are shown.

after they were trained on the Lorenz model task. For both models, the time evolution of the reservoir states reflected the input chaotic time series. Although it was not apparent in the time evolution of the reservoir states, we observed that the gates of the SM-RC model evolved by adapting to the input chaotic time series. The dynamics of SM-RC for various task conditions, such as σ^{in} and N^{forward} , are presented in Supplementary Note 5. Because of the dynamic behavior of the gates, the prediction performance of SM-RC was better than that of conventional RC.

Figure 4 shows a comparison of the prediction performance of conventional RC and SM-RC for NARMA (Fig. 4a–b) and Lorenz model tasks (Fig. 4c–d). In Fig. 4a, c, the prediction error of conventional RC is plotted as a function of the reservoir size. The horizontal lines indicate the SM-RC prediction error. The number of neurons in the reservoir layer of the SM-RC models were all fixed to 100. For the Lorenz model tasks (Fig. 4c), we present the results for different values of the following two parameters: input noise σ^{in} and prediction steps N^{forward} . In both tasks, the best results among 50 random weight initializations for the conventional RC and SM-RC models are shown. For the conventional RC models, the mean and the standard deviation are evaluated in Supplementary Note 6. As shown, SM-RC achieved better prediction performance than conventional RC. For the NARMA tasks, SM-RC exhibited comparable prediction performance to conventional RC with a two times larger reservoir. In contrast, for the Lorenz model tasks, SM-RC exhibited prediction performance comparable to conventional RC with a 10 times larger reservoir.

In Fig. 4b, d, the prediction performance of SM-RC when one or both gates do not evolve over time is shown. For the NARMA and Lorenz model tasks, the performance was the best when both gates evolved, followed by the case where the reservoir gate was modulated and then the case where the input gate was modulated. The performance was the worst when both gates were static. These results are similar to those for the simple attention task. As before, they indicate that the observed performance improvement was caused not by the optimization of the spectral radius and input intensity but by their temporal modulation.

Finally, we compared the learning performance of SM-RC with that of other state-of-the-art RNN models, namely Elman RNNs (also known as vanilla RNNs), LSTM, and GRUs. Figure 5 shows the learning results for the Lorenz model tasks ($N^{\text{forward}} = 20$ and 30). In the upper panels (Fig. 5a–f.1), MSEs of SM-RC, Elman RNN, LSTM, and GRU models are plotted with different RNN sizes. The RNN size represents the reservoir size N^{res} for the case of SM-RC and the number of output units for the cases of Elman RNNs, LSTM, and GRUs. Although the learning performance of SM-RC was higher than that of the conventional RC models, as seen in Fig. 4, it is worse than that of Elman RNNs, LSTM, and GRUs. For example, when $\sigma^{\text{in}} = 0.01$ and $N^{\text{forward}} = 30$, an Elman RNN model and an LSTM model, both with an RNN size of 100, performed as well as the SM-RC model with an RNN size of 400. Similarly, a GRU model with an RNN size of 30 performed as well as the SM-RC model with an RNN size of 400. The performance of the Elman RNNs declined when the RNN size exceeded 200 or 300. This degradation was due to the instability in the learning process of the Elman RNNs^{46,47}. The performance difference among various RNN models is primarily attributed to the number of trainable parameters. In the lower panels (Fig. 5a–f.2), we compared the MSEs of those RNN models as a function of the number of trainable parameters. We also included the results of the conventional RC models in the comparison. We found that the SM-RC models showed smaller MSEs than the conventional RC models for the same number of trainable parameters. Furthermore, the SM-RC showed MSEs comparable to LSTM and GRUs for the same number of trainable parameters. These results indicate that the learning ability of RC can reach that of state-of-the-art RNN models by employing a temporal self-modulation mechanism.

Discussion

We propose SM-RC that can alter the dynamical properties of the reservoir layer by introducing gate structures. We found that the SM-RC architecture achieved better learning performance than that of conventional RC for NARMA, and especially for the simple attention and Lorenz model tasks. In the cases of

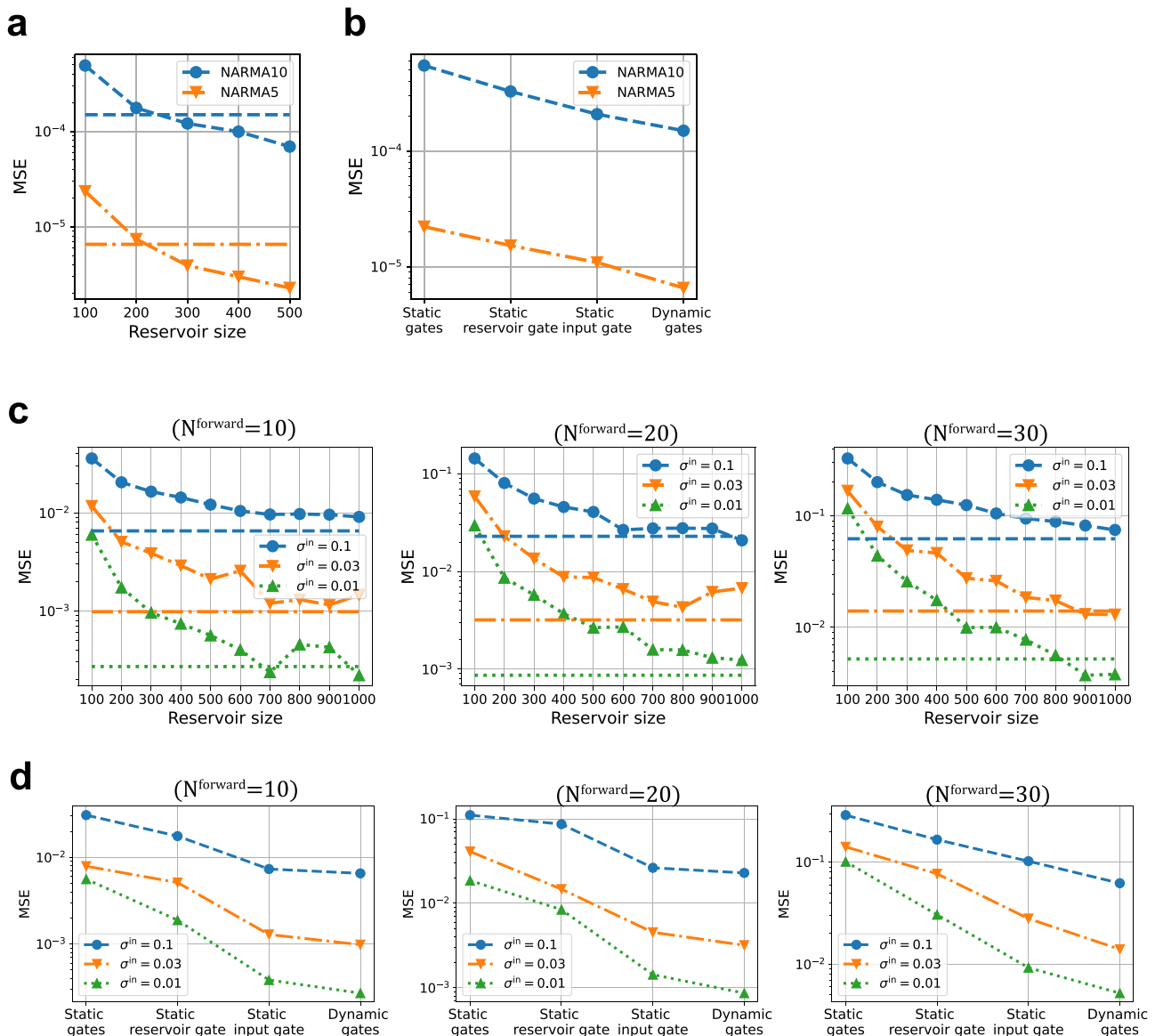


Fig. 4 Performance evaluation for the NARMA and Lorenz model tasks. **a, b** The performance results for the NARMA5 and NARMA10 tasks are shown. **a** The figure shows comparisons of the lowest mean squared errors (MSEs) for the conventional reservoir computing (RC) and self-modulated RC (SM-RC) models among 50 random weight initializations. The horizontal axis indicates the reservoir size of the conventional RC model. The reservoir size of the SM-RC models was fixed to 100. The MSEs for the conventional RC models are plotted, whereas the horizontal lines indicate the MSEs for the SM-RC models. **b** The figure shows comparisons of the MSEs for the SM-RC models when some gates were temporally fixed. **c, d** The performance results for the Lorenz model task are shown. **c** The results for the cases of the prediction steps N^{forward} of 10, 20, and 30 are presented in the left, center, and right figures, respectively. In addition, the results for different values of the input noise intensity σ^{in} are presented. The panels show comparisons of the lowest MSEs for the conventional RC and SM-RC models among 50 random weight initializations. The horizontal axis indicates the reservoir size of the conventional RC model. The reservoir size of the SM-RC models was fixed to 100. In each figure, the MSEs for the conventional RC models are plotted, whereas the horizontal lines indicate the MSEs for the SM-RC models. **d** The figures show comparisons of the MSEs for the SM-RC models when some gates were temporally fixed. Various cases for the values of N^{forward} and σ^{in} are shown as in **c**.

nonuniform input signals, such as those found in simple attention and Lorenz model tasks, the mechanism that adapts the way of information is processed according to the input signal is considered effective. Because most real-world time-series data are nonuniform, the proposed architecture is expected to be effective for real-world data. In the simple attention task, we observed local chaotic dynamics in SM-RC. Chaotic dynamics are of interest in deep-learning research because they increase the expressiveness of learning models^{48,49}. Recently, Inoue et al. observed transient chaos in transformer models⁵⁰. It is expected that SM-RC utilized the high expressivity of chaotic dynamics for encoding the input

signal and outputting the pulse. However, the underlying learning mechanism requires further investigation. We expect that detailed analysis of dynamical systems^{10,24,51,52} will produce ideas for improving SM-RC.

SM-RC provides insights into the mechanism of the nervous system. Neural networks with random connections, as used in RC, are attracting attention as models of the brain^{7,51,53,54}. The proposed SM-RC model can be thought of as incorporating the mechanism of neuromodulation into RC. This is because both systems globally and temporally modulate the activity levels of neurons^{36–38}. For example, the neuromodulator acetylcholine is

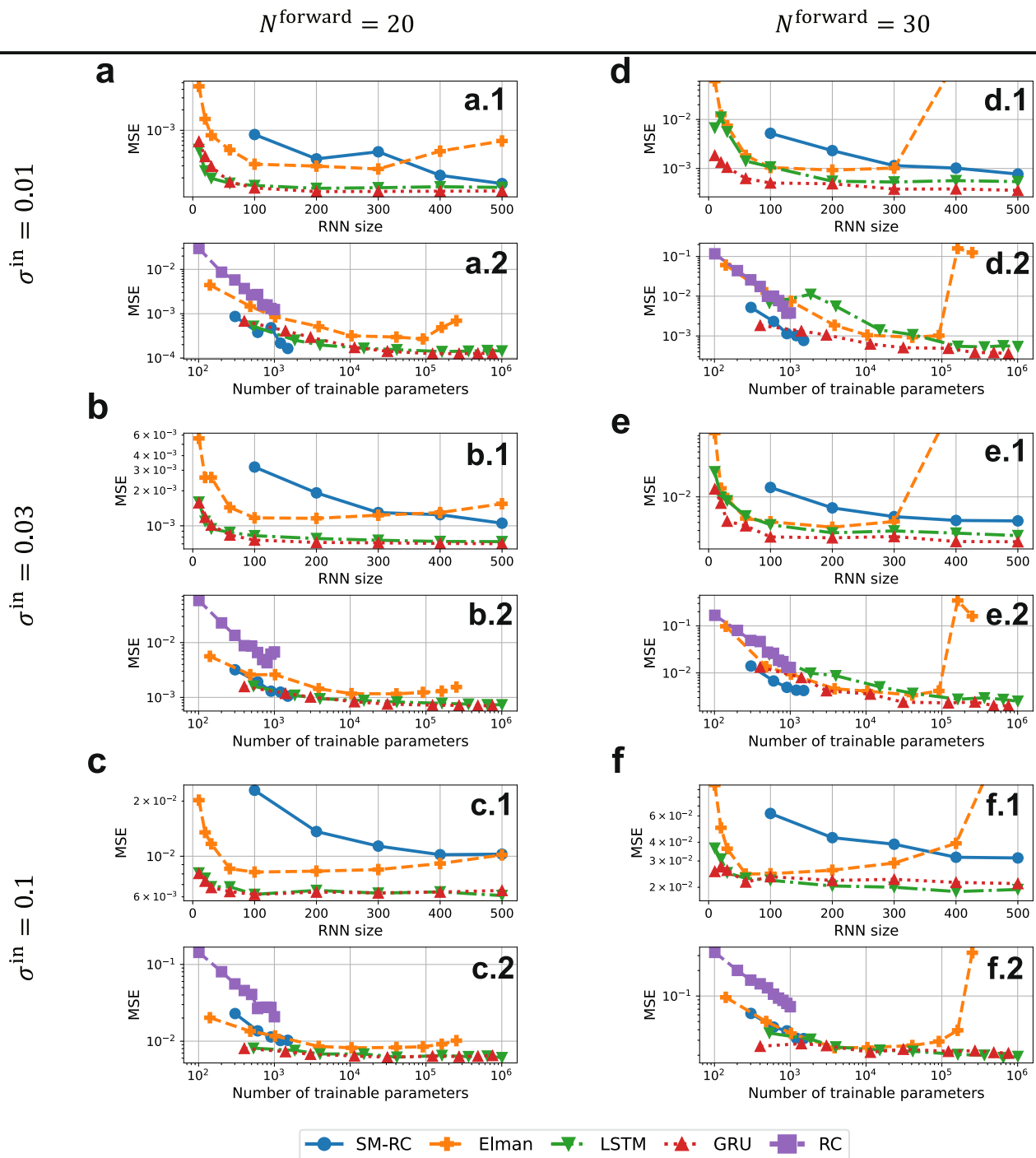


Fig. 5 Performance comparison of RNN models for the Lorenz model tasks. **a–c** and **d–f** represent the results for the cases of prediction steps N^{forward} of 20 and 30, respectively. The results for the cases of input noise intensity σ^{in} of 0.01 (**a** and **d**), 0.03 (**b** and **e**), and 0.1 (**c** and **f**) are presented. In the upper panels (**a–f.1**), mean squared errors (MSEs) of self-modulated reservoir computing (SM-RC), Elman recurrent neural network (RNN), long short-term memory (LSTM), and gated recurrent unit (GRU) models are plotted with different RNN sizes. In the lower panels (**a–f.2**), MSEs are plotted as a function of the number of trainable parameters, where the results of conventional reservoir computing (RC) models are also plotted. Each panel presents the lowest MSEs among 50 random weight initializations for SM-RC, Elman RNNs, LSTM, GRU, and conventional RC models.

delivered from the basal forebrain to brain areas, thus increasing the neuronal activity therein³⁶. SM-RC realizes an attention mechanism that can be related to neuromodulation³⁶. We expect that the resemblance between SM-RC and the neuromodulation mechanism can be studied by adopting biologically plausible characteristics such as spike-based communications, synaptic weights and time constants, and neural geometries.

Efficient hardware implementation is important for the real-world application of SM-RC^{11,12}. Because the input gate g^{in} only directly modulates the input intensity, it is not difficult to implement. Additionally, we believe that the reservoir gate g^{res} can be implemented according to the hardware mechanism. For example, in analog circuit implementations, connections between neurons are represented by current magnitudes^{55,56}. If the

connection weights are implemented with digital memory, it is sufficient to provide a mechanism for modulating the digital values. In the case of analog memory, if it is implemented with three-terminal memory devices such as floating-gate devices, by modulating the gate voltage of all or part of the field-effect transistor of the memory, modulation of reservoir-layer dynamics can be achieved. In addition, delay feedback systems⁵⁷ can build a reservoir layer using a single nonlinear node, which is often used in photonic systems⁵⁸ and electronic integrated circuits⁵⁹. In these systems, the entire reservoir system can be modulated simply by changing the single node; thus, it is relatively easy to implement the reservoir gate. This hardware-friendly aspect of SM-RC is what motivates us to use SM-RC for edge AI rather than other state-of-the-art RNN models, such as LSTM⁸ and GRUs⁹. Because the RNN connections in LSTM and GRUs must be trained, it is difficult to implement these models with physical systems where the internal connections are not finely tunable. Such physical systems include spintronics systems⁶⁰ and photonic systems⁶¹. We note that the gate structure of SM-RC is simpler than those of LSTM and GRUs. In LSTM and GRUs, the dimension of the gate is the same as that of the output units. In other words, RNN neurons are modulated differently. By contrast, in SM-RC, the gates are scalar (one-dimensional). Therefore, RNN neurons (reservoir in this case) are modulated as a whole. This simple structure enables physical systems whose dynamic properties can only be globally modulated to be used as reservoirs.

We believe that it is possible to train SM-RC implemented in physical systems by using various learning techniques. In this study, we trained the SM-RC model via backpropagation through time (BPTT)⁶². In BPTT, detailed information regarding the reservoir-layer dynamics is required; however, in physical and analog systems, it is often difficult to capture the internal dynamics in detail. Efforts to train such hardware have progressed in recent years. For example, Wright et al. successfully trained a black-boxed system by capturing the internal dynamics using deep learning⁶³. In addition, error backpropagation approximations, such as feedback alignment⁶⁴ and its variants^{65–68}, are suitable for training models implemented in physical systems. It has been reported that BPTT can be approximated so that RNNs can be trained in a biologically plausible manner^{67,69}. Nakajima et al. extended these methods and successfully trained multilayer physical RC⁷⁰. By applying these methods to SM-RC, it is expected that SM-RC can be trained in analog and physical systems. We believe that SM-RC broadens the design space of physical RC.

Methods

Model. The time evolution of the ESN-type SM-RC model can be expressed as

$$\mathbf{x}(t) = (1 - \alpha)\mathbf{x}(t - 1) + \alpha \tanh(g^{\text{res}}(t - 1)W^{\text{res}}\mathbf{x}(t - 1) + g^{\text{in}}(t - 1)W^{\text{in}}\mathbf{u}(t) + \xi\mathbf{1}), \quad (3)$$

$$g^{\text{res}}(t) = f(W_{\text{fb}}^{\text{res}}\mathbf{x}(t) + b_{\text{fb}}^{\text{res}}), \quad (4)$$

$$g^{\text{in}}(t) = f(W_{\text{fb}}^{\text{in}}\mathbf{x}(t) + b_{\text{fb}}^{\text{in}}), \quad (5)$$

$$y(t) = W^{\text{out}}\mathbf{x}(t) + b^{\text{out}}, \quad (6)$$

where $\mathbf{x}(t) \in \mathbb{R}^{N^{\text{res}}}$, $\mathbf{u}(t) \in \mathbb{R}^{N^{\text{in}}}$, and $y(t) \in \mathbb{R}$ represent the internal state, input signal, and output of the reservoir layer, respectively. α is the leakage constant. For simplicity, α is set to 1 as in^{21,43,45}. $W^{\text{in}} \in \mathbb{R}^{N^{\text{res}} \times N^{\text{in}}}$ and $W^{\text{res}} \in \mathbb{R}^{N^{\text{res}} \times N^{\text{res}}}$ are matrices representing the connection weights from the input to the reservoir and the inner connection weights of the reservoir,

respectively. N^{in} and N^{res} represent the input dimension and reservoir dimension, respectively. Each element of W^{in} is initialized by randomly sampling from the uniform distribution of $[-\rho^{\text{in}}, \rho^{\text{in}}]$. Each element of W^{res} is randomly sampled from the uniform distribution of $[-1, 1]$; then, it is multiplied by a constant so that the “initial” spectral radius becomes $\hat{\rho}^{\text{res}}$. $\mathbf{1}$ is a vector of 1s, and $\xi \in \mathbb{R}$ controls the magnitude of the bias term of neurons in the reservoir layer⁴³. $g^{\text{in}}(t) \in \mathbb{R}$ is the input gate that modulates the input intensity and is controlled by feedback from the reservoir layer via weights $W_{\text{fb}}^{\text{in}}$. Similarly, $g^{\text{res}}(t) \in \mathbb{R}$ is the reservoir gate that modulates the connection strength of the reservoir and is controlled by feedback from the reservoir layer via weights $W_{\text{fb}}^{\text{res}}$. The output function f of the gate is a non-negative function. In this study, the output value was limited to (0, 2) to stabilize the learning process:

$$f(x) = \frac{2}{1 + e^{-x}}. \quad (7)$$

The input gate temporally modulates the intensity of the input by multiplying it by a value within the range of (0, 2), thus allowing the selection of useful input signals. In contrast, the reservoir gate temporally modulates the spectral radius ρ^{res} within $(0, 2\hat{\rho}^{\text{res}})$, thus allowing the information stored in the reservoir layer to be retained or discarded. The time evolution of conventional RC is obtained by replacing both g^{in} and g^{res} with 1. Specifically, the time evolution of the conventional RC model is expressed as

$$\mathbf{x}(t) = (1 - \alpha)\mathbf{x}(t - 1) + \alpha \tanh(W^{\text{res}}\mathbf{x}(t - 1) + W^{\text{in}}\mathbf{u}(t) + \xi\mathbf{1}) \quad (8)$$

$$y(t) = W^{\text{out}}\mathbf{x}(t) + b^{\text{out}}. \quad (9)$$

W^{in} and W^{res} are fixed weights, as in the conventional RC framework. In SM-RC, in addition to the output weights (W^{out} , b^{out}), feedback weights ($W_{\text{fb}}^{\text{res}}$, $b_{\text{fb}}^{\text{res}}$, $W_{\text{fb}}^{\text{in}}$, $b_{\text{fb}}^{\text{in}}$) are trained. In this study, the output weights were trained using the pseudo-inverse method, and the feedback weights were trained using BPTT⁶² (see below).

Learning procedure. All the tasks performed in this study involved predicting the target signal $y^t(t)$ using the input time series $\{u(0), u(1), \dots, u(t)\}$ obtained by time t . RC usually inputs only $u(t)$ to the reservoir layer at time t (Eq. (3)). In the case of conventional RC, the output weights were trained using the pseudo-inverse method to minimize the squared error between the output and the target signal¹³:

$$\sum_{t=1}^T \|y(t) - y^t(t)\|^2, \quad (10)$$

where T represents the number of timesteps of the training data. When multiple training data exist, T is the product of the number of timesteps of the training data and the number of training data. To avoid the influence of the initial state, the first 200 steps in the simulation were excluded from the training data (free run). In this study, adding the regularization term $\|W^{\text{out}}\|$ (ridge regression) did not improve the performance.

For SM-RC, Elman RNNs, LSTM, and GRUs, we used the same loss function (Eq. (10)) and trained weights via the gradient descent method. In each epoch, the weights were updated once using the whole training dataset (full-batch training), and this was repeated for 10^4 epochs. We found that while LSTM and GRUs perform well, the learning process of SM-RC using BPTT tends to be unstable. To address this, we adopted the following strategy. During each full-batch training session, we first trained the output layer using the pseudo-inverse method, similar to conventional RC techniques. Subsequently, the gates were trained

using BPTT. By learning the output layer in advance, the backpropagating error signals are reduced, which may lead to stable learning (see Supplementary Note 4). In BPTT, the weights were updated using the Adam optimizer with a learning rate of 10^{-3} ⁷¹. All implementations and simulations were performed using the PyTorch framework⁷².

The hyperparameters of the conventional RC model (ρ^{in} , ρ^{res} , and ξ) were optimized via Bayesian optimization^{73,74}. We set $\xi=0$ for the simple attention tasks and NARMA tasks. The hyperparameters of the SM-RC model were set as $\rho^{\text{in}}=0.12$, $\rho^{\text{res}}=0.9$. The bias term was set as $\xi=0$ for the simple attention tasks and NARMA tasks and $\xi=0.2$ for the Lorenz model tasks. We observed that SM-RC training was somewhat unstable (see Supplementary Note 4). Therefore, in the performance evaluation of SM-RC and other models, we considered the best results for training with 50 different initial weights.

In this study, we investigated the case where a single gate or both gates did not change over time. This was done by fixing all the weight elements ($W_{\text{fb}}^{\text{in(res)}}$) to 0 and training only the bias term $b_{\text{fb}}^{\text{in(res)}}$.

Datasets. In the simple attention task, the i th input training datum had a value of 1 in the input time interval $[250 + t_i^{\text{jitter}}, 259 + t_i^{\text{jitter}}]$ and a value sampled from a Gaussian distribution with a mean of 0 and a standard deviation of $\sigma^{\text{in}} \in \{0.1, 0.2, 0.3\}$ at other timesteps. The i th output training datum had a value of 1 in the time interval of $[290 + t_i^{\text{jitter}}, 291 + t_i^{\text{jitter}}]$ and 0 at other timesteps. t_i^{jitter} represents the jitter noise; one of $\{-2, -1, 0, 1, 2\} \in Z$ was randomly sampled uniformly. The jitter noise was introduced to prevent the learning models from generating output pulses using the initial value ($x_i(0) = 0$) instead of using the input signal (see Supplementary Note 3). We used 100 data obtained as described above as training data and 100 other data as test data.

NARMA is a nonlinear autoregressive moving average model given by

$$y^{\text{tc}}(t) = 0.3y^{\text{tc}}(t-1) + 0.05y^{\text{tc}}(t-1) \sum_{i=1}^m y^{\text{tc}}(t-i) + 1.5s(t-m+1)s(t) + 0.1, \quad (11)$$

where $s(t) \in \mathbb{R}$ is uniformly sampled from the interval $[0, 0.5]$. The NARMA5 and NARMA10 time series were obtained with $m=5$ and $m=10$, respectively. The task was to predict $y^{\text{tc}}(t)$ using $s(t)$ as the input. To eliminate the influence of the initial value, the first 200 steps were discarded, and 2000 steps were generated as training data. We also generated test data for 2000 steps using the same method. We only used a single data for the training dataset and a single data for the test dataset.

The Lorenz model evolves over time according to the following differential equation:

$$\frac{dx}{dt} = 10(y-x), \frac{dy}{dt} = x(28-z) - y, \frac{dz}{dt} = xy - \frac{8}{3}z. \quad (12)$$

Discrete time-series data were obtained from the above differential equation by using the Euler method with a timestep of $\Delta t = 0.01$. To eliminate the influence of the initial value, the first 1000 steps were discarded, and the time series of the subsequent 2000 steps was obtained. Further, 100 training data and 100 different test data were generated in the same way from different initial values. The task was to predict $z(t + N^{\text{forward}}\Delta t)$ with $x(t)$ as the input. $N^{\text{forward}} \in \{10, 20, 30\}$ represents the number of steps forward to predict. $x(t)$ and $z(t)$ were normalized to a mean of 0 and variance of 1 for the training and test data, respectively. We also added Gaussian noise with a mean of 0 and

variance of $\sigma^{\text{in}} \in \{0.01, 0.03, 0.1\}$ to the input $x(t)$ in the training data. Note that the noise was not updated in the training epoch.

Sensitivity analysis. We evaluated the sensitivity of the learning models in the simple attention task. The sensitivity indicates how a perturbation to the reservoir state is magnified in t_p timesteps. For the input data without jitter noise, SM-RC was operated, and the states of the reservoir layer are taken as the reference states $\{\mathbf{x}^{\text{base}}(0), \mathbf{x}^{\text{base}}(1), \dots, \mathbf{x}^{\text{base}}(t), \dots\}$. Then, a perturbation $p_j \in \mathbb{R}^{N^{\text{res}}}$ ($j = 1, 2, \dots, N_p$) is applied to the reservoir state $\mathbf{x}^{\text{base}}(t) \in \mathbb{R}^{N^{\text{res}}}$ at time t , and the perturbed reservoir state $\mathbf{x}^{\text{ptb}}(t+t_p) \in \mathbb{R}^{N^{\text{res}}}$ at time $t+t_p$ is obtained using Eq. (3). The perturbation vector p_j was obtained by sampling a vector contained in the unit circle using the Metropolis–Hastings algorithm, followed by rescaling so that the obtained vector satisfied $\|p_j\| = \epsilon$. We estimated the sensitivity using the reference and perturbed states as follows:

$$\lambda(t) = \frac{1}{t_p N_p} \sum_{j=1}^{N_p} \ln \left(\frac{\|\mathbf{x}^{\text{base}}(t+t_p) - \mathbf{x}^{\text{ptb}}(t+t_p)\|}{\epsilon} \right), \quad (13)$$

Similarly, we estimated the maximum sensitivity:

$$\lambda_{\text{max}}(t) = \frac{1}{t_p} \max_j \left[\ln \left(\frac{\|\mathbf{x}^{\text{base}}(t+t_p) - \mathbf{x}^{\text{ptb}}(t+t_p)\|}{\epsilon} \right) \right]. \quad (14)$$

In the simulation, we used $t_p = 2$, $\epsilon = 10^{-8}$, and $N_p = 200$. The simulation results when other values of t_p were used are presented elsewhere (see Supplementary Note 1). The perturbation to the gates g^{res} , g^{in} was applied indirectly in accordance with Eq. (4) because they are not independent (state) variables. The gates g^{res} , g^{in} are dependent variables because they are determined by the state of the reservoir layer. Therefore, in the case of SM-RC, the perturbation is applied only to the state of the neurons in the reservoir layer. Note that the maximum sensitivity is also known as the maximum local Lyapunov exponent.

Data availability

All data used in the paper are numerically reproducible according to the procedures described in Methods.

Code availability

Computer codes are available from the authors upon request.

Received: 18 April 2023; Accepted: 8 December 2023;

Published online: 12 January 2024

References

1. Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L. & Muller, P.-A. Deep learning for time series classification: a review. *Data Min. Knowl. Discov.* **33**, 917–963 (2019).
2. Dong, S., Wang, P. & Abbas, K. A survey on deep learning and its applications. *Comput. Sci. Rev.* **40**, 100379 (2021).
3. Thompson, N. C., Greenewald, K., Lee, K. & Manso, G. F. The computational limits of deep learning. *arXiv:2007.05558* (2020).
4. Patterson, D. et al. Carbon emissions and large neural network training. *arXiv:2104.10350* (2021).
5. Murshed, M. G. S. et al. Machine learning at the network edge: A survey. *ACM Comput. Surv.* **54**, 1–37 (2021).
6. Jaeger, H. The “echo state” approach to analysing and training recurrent neural networks. *Technical Report GMD Report 148, German National Research Center for Information Technology* (2001).
7. Maass, W., Natschläger, T. & Markram, H. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Comput.* **14**, 2531–2560 (2002).

8. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**, 1735–1780 (1997).
9. Cho, K. et al. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734 (2014).
10. Dambre, J., Verstraeten, D., Schrauwen, B. & Massar, S. Information processing capacity of dynamical systems. *Sci. Rep.* **2**, 514 (2012).
11. Tanaka, G. et al. Recent advances in physical reservoir computing: A review. *Neural Netw.* **115**, 100–123 (2019).
12. Nakajima, K. Physical reservoir computing—an introductory perspective. *Jpn. J. Appl. Phys.* **59**, 060501 (2020).
13. Lukoševičius, M. A practical guide to applying echo state networks. *Neural Networks: Tricks of the Trade, Lecture Notes in Computer Science*, vol 7700, 659–686 (2012).
14. Vlachas, P. et al. Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. *Neural Netw.* **126**, 191–217 (2020).
15. Sun, C. et al. A systematic review of echo state networks from design to application. *IEEE Trans. Artif. Intell. (Early Access)*, <https://doi.org/10.1109/TAI.2022.3225780> (2022).
16. Tong, Z. & Tanaka, G. Reservoir computing with untrained convolutional neural networks for image recognition. In *2018 24th International Conference on Pattern Recognition (ICPR)*, 1289–1294 (2018).
17. Pathak, J., Hunt, B., Girvan, M., Lu, Z. & Ott, E. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Phys. Rev. Lett.* **120**, 024102 (2018).
18. Kawai, Y., Park, J., Tsuda, I. & Asada, M. Learning long-term motor timing/patterns on an orthogonal basis in random neural networks. *Neural Netw.* **163**, 298–311 (2023).
19. Iinuma, T., Nobukawa, S. & Yamaguchi, S. Assembly of echo state networks driven by segregated low dimensional signals. In *2022 International Joint Conference on Neural Networks (IJCNN)*, 1–8, <https://doi.org/10.1109/IJCNN55064.2022.9892881> (2022).
20. Gallicchio, C., Micheli, A. & Pedrelli, L. Deep reservoir computing: A critical experimental analysis. *Neurocomputing* **268**, 87–99 (2017).
21. Sakemi, Y., Morino, K., Leleu, T. & Aihara, K. Model-size reduction for reservoir computing by concatenating internal states through time. *Sci. Rep.* **10**, 21794 (2020).
22. Chen, P., Liu, R., Aihara, K. & Chen, L. Autoreervoir computing for multistep ahead prediction based on the spatiotemporal information transformation. *Nat. Commun.* **11**, 4568 (2020).
23. Sussillo, D. & Abbott, L. Generating coherent patterns of activity from chaotic neural networks. *Neuron* **63**, 544–557 (2009).
24. Rivkind, A. & Barak, O. Local dynamics in trained recurrent neural networks. *Phys. Rev. Lett.* **118**, 258101 (2017).
25. Nicola, W. & Clopath, C. Supervised learning in spiking neural networks with force training. *Nat. Commun.* **8**, 2208 (2017).
26. Matsuki, T. & Shibata, K. Adaptive balancing of exploration and exploitation around the edge of chaos in internal-chaos-based learning. *Neural Netw.* **132**, 19–29 (2020).
27. Schrauwen, B., Wardermann, M., Verstraeten, D., Steil, J. J. & Stroobandt, D. Improving reservoirs using intrinsic plasticity. *Neurocomputing* **71**, 1159–1171 (2008).
28. Yusoff, M.-H., Chrol-Cannon, J. & Jin, Y. Modeling neural plasticity in echo state networks for classification and regression. *Inf. Sci.* **364–365**, 184–196 (2016).
29. Morales, G. B., Mirasso, C. R. & Soriano, M. C. Unveiling the role of plasticity rules in reservoir computing. *Neurocomputing* **461**, 705–715 (2021).
30. Laje, R. & Buonomano, D. V. Robust timing and motor patterns by taming chaos in recurrent neural networks. *Nat. Neurosci.* **16**, 925–933 (2013).
31. Inoue, K., Nakajima, K. & Kuniyoshi, Y. Designing spontaneous behavioral switching via chaotic itinerancy. *Sci. Adv.* **6**, eabb3989 (2020).
32. Bahdanau, D., Cho, K. & Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473* (2014).
33. Vaswani, A. et al. Attention is all you need. In *Advances in Neural Information Processing Systems*, vol. 30 (2017).
34. Han, K. et al. A survey on vision transformer. *IEEE Trans. Pattern Anal. Mach. Intell.* **45**, 87–110 (2023).
35. Jumper, J. et al. Highly accurate protein structure prediction with AlphaFold. *Nature* **596**, 583–589 (2021).
36. Thiele, A. & Bellgrove, M. A. Neuromodulation of attention. *Neuron* **97**, 769–785 (2018).
37. Edelman, E. & Lessmann, V. Dopaminergic innervation and modulation of hippocampal networks. *Cell Tissue Res.* **373**, 711–727 (2018).
38. Palacios-Filardo, J. & Mellor, J. R. Neuromodulation of hippocampal long-term synaptic plasticity. *Curr. Opin. Neurobiol.* **54**, 37–43 (2019).
39. Yildiz, I. B., Jaeger, H. & Kiebel, S. J. Re-visiting the echo state property. *Neural Netw.* **35**, 1–9 (2012).
40. Jaeger, H. Long short-term memory in echo state networks: Details of a simulation study. *Jacobs University Technical Reports* **27** (2012).
41. Inubushi, M. & Yoshimura, K. Reservoir computing beyond memory–nonlinearity trade-off. *Sci. Rep.* **7**, 10199 (2017).
42. Jordanou, J. P., Aislan Antonelo, E., Camponogara, E. & Gildin, E. Investigation of proper orthogonal decomposition for echo state networks. *Neurocomputing* **548**, 126395 (2023).
43. Lu, Z. et al. Reservoir observers: Model-free inference of unmeasured variables in chaotic systems. *Chaos: Interdiscip. J. Nonlinear Sci.* **27**, 041102 (2017).
44. Katori, Y., Tamukoh, H. & Morie, T. Reservoir computing based on dynamics of pseudo-billiard system in hypercube. In *2019 International Joint Conference on Neural Networks (IJCNN)*, 1–8, <https://doi.org/10.1109/IJCNN.2019.8852329> (2019).
45. Inubushi, M. & Goto, S. Transfer learning for nonlinear dynamics and its application to fluid turbulence. *Phys. Rev. E* **102**, 043301 (2020).
46. Doya, K. Bifurcations in the learning of recurrent neural networks. In *[Proceedings] 1992 IEEE International Symposium on Circuits and Systems*, vol. 6, 2777–2780 (1992).
47. Pascanu, R., Mikolov, T. & Bengio, Y. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, PMLR vol. 28, 1310–1318 (2013).
48. Poole, B., Lahiri, S., Raghun, M., Sohl-Dickstein, J. & Ganguli, S. Exponential expressivity in deep neural networks through transient chaos. In *Advances in Neural Information Processing Systems*, vol. 29 (2016).
49. Zhang, G., Li, G., Shen, W. & Zhang, W. The expressivity and training of deep neural networks: Toward the edge of chaos? *Neurocomputing* **386**, 8–17 (2020).
50. Inoue, K., Ohara, S., Kuniyoshi, Y. & Nakajima, K. Transient chaos in bidirectional encoder representations from transformers. *Phys. Rev. Res.* **4**, 013204 (2022).
51. Barak, O., Sussillo, D., Romo, R., Tsodyks, M. & Abbott, L. From fixed points to chaos: Three models of delayed discrimination. *Prog. Neurobiol.* **103**, 214–222 (2013).
52. Sussillo, D. & Barak, O. Opening the Black Box: Low-Dimensional Dynamics in High-Dimensional Recurrent Neural Networks. *Neural Comput.* **25**, 626–649 (2013).
53. Seoane, L. F. Evolutionary aspects of reservoir computing. *Philos. Trans. R. Soc. B: Biol. Sci.* **374**, 20180377 (2019).
54. Suárez, L. E., Richards, B. A., Lajoie, G. & Misis, B. Learning function from structure in neuromorphic networks. *Nat. Mach. Intell.* **3**, 771–786 (2021).
55. Bauer, F. C., Muir, D. R. & Indiveri, G. Real-time ultra-low power ECG anomaly detection using an event-driven neuromorphic processor. *IEEE Trans. Biomed. Circuits Syst.* **13**, 1575–1582 (2019).
56. Yamaguchi, M., Katori, Y., Kamimura, D., Tamukoh, H. & Morie, T. A chaotic Boltzmann machine working as a reservoir and its analog VLSI implementation. In *2019 International Joint Conference on Neural Networks (IJCNN)*, 1–7, <https://doi.org/10.1109/IJCNN.2019.8852325> (2019).
57. Appeltant, L. et al. Information processing using a single dynamical node as complex system. *Nat. Commun.* **2**, 468 (2011).
58. Brunner, D. et al. Tutorial: Photonic neural networks in delay systems. *J. Appl. Phys.* **124**, 152004 (2018).
59. Penkovsky, B., Larger, L. & Brunner, D. Efficient design of hardware-enabled reservoir computing in FPGAs. *J. Appl. Phys.* **124**, 162101 (2018).
60. Finocchio, G. et al. The promise of spintronics for unconventional computing. *J. Magn. Magn. Mater.* **521**, 167506 (2021).
61. Lugnan, A. et al. Photonic neuromorphic information processing and reservoir computing. *APL Photonics* **5**, 020901 (2020).
62. Werbos, P. J. Backpropagation through time: what it does and how to do it. *Proc. IEEE* **78**, 1550–1560 (1990).
63. Wright, L. G. et al. Deep physical neural networks trained with backpropagation. *Nature* **601**, 549–555 (2022).
64. Lillicrap, T. P., Cownden, D., Tweed, D. B. & Akerman, C. J. Random synaptic feedback weights support error backpropagation for deep learning. *Nat. Commun.* **7**, 13276 (2016).
65. Nøkland, A. Direct feedback alignment provides learning in deep neural networks. In *Advances in Neural Information Processing Systems*, vol. 29 (2016).
66. Guerguiev, J., Lillicrap, T. P. & Richards, B. A. Towards deep learning with segregated dendrites. *eLife* **6**, e22901 (2017).
67. Murray, J. M. Local online learning in recurrent networks with random feedback. *eLife* **8**, e43299 (2019).
68. Frenkel, C., Lefebvre, M. & Bol, D. Learning without feedback: Fixed random learning signals allow for feedforward training of deep neural networks. *Front. Neurosci.* **15**, 629892 (2021).
69. Bellec, G. et al. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nat. Commun.* **11**, 3625 (2020).
70. Nakajima, M. et al. Physical deep learning with biologically inspired training method: gradient-free approach for physical hardware. *Nat. Commun.* **13**, 7847 (2022).

71. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv:1412.6980* (2014).
72. Paszke, A. et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, vol. 32 (2019).
73. Snoek, J., Larochelle, H. & Adams, R. P. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, vol. 25 (2012).
74. Frazier, P. I. A tutorial on bayesian optimization. *arXiv:1807.02811* (2018).

Acknowledgements

This work was partially supported by SECOM Science and Technology Foundation, JST PRESTO Grant Number JPMJPR22C5, JST Moonshot R&D Grant Number JPMJMS2021, AMED under Grant Number JP23dm0307009, Institute of AI and Beyond of UTokyo, the International Research Center for Neurointelligence (WPI-IRCN) at The University of Tokyo Institutes for Advanced Study (UTIAS), JSPS KAKENHI Grant Number JP20H05921. Computational resource of AI Bridging Cloud Infrastructure (ABCI) provided by National Institute of Advanced Industrial Science and Technology (AIST) was used.

Author contributions

Y.S. invented the method, performed all the simulations, and wrote the first draft of the paper. Y.S., S.N., T.M., and K.A. designed the sensitivity analysis. Y.S. and S.N. discussed the relationship with neuromodulation. Y.S. and T.M. discussed the future prospects for hardware implementation. All the authors discussed the results and wrote the paper.

Competing interests

Y.S. and K.A. have applied for a patent related to the proposed methods (Japanese Patent Application No.2022-138099). The remaining authors have no conflict of interests to declare.

Additional information

Supplementary information The online version contains supplementary material available at <https://doi.org/10.1038/s42005-023-01500-w>.

Correspondence and requests for materials should be addressed to Yusuke Sakemi.

Peer review information *Communications Physics* thanks the anonymous reviewers for their contribution to the peer review of this work. A peer review file is available.

Reprints and permission information is available at <http://www.nature.com/reprints>

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this license, visit <http://creativecommons.org/licenses/by/4.0/>.

© The Author(s) 2024