https://doi.org/10.1038/s44172-024-00227-y

A high-performance deep reservoir computer experimentally demonstrated with ion-gating reservoirs

Check for updates

Daiki Nishioka[1,2], Takashi Tsuchiya[1] ✉, Masataka Imura[3], Yasuo Koide[4], Tohru Higuchi[2] & Kazuya Terabe[1]

While physical reservoir computing is a promising way to achieve low power consumption neuromorphic computing, its computational performance is still insufficient at a practical level. One promising approach to improving its performance is deep reservoir computing, in which the component reservoirs are multi-layered. However, all of the deep-reservoir schemes reported so far have been effective only for simulation reservoirs and limited physical reservoirs, and there have been no reports of nanodevice implementations. Here, as an ionics-based neuromorphic nanodevice implementation of deep-reservoir computing, we report a demonstration of deep physical reservoir computing with maximum of four layers using an ion gating reservoir, which is a small and high-performance physical reservoir. While the previously reported deep-reservoir scheme did not improve the performance of the ion gating reservoir, our deep-ion gating reservoir achieved a normalized mean squared error of 9.08×10^{-3} on a second-order nonlinear autoregressive moving average task, which is the best performance of any physical reservoir so far reported in this task. More importantly, the device outperformed full simulation reservoir computing. The dramatic performance improvement of the ion gating reservoir with our deep-reservoir computing architecture paves the way for high-performance, large-scale, physical neural network devices.

Physical reservoir computing (PRC), which directly utilizes the nonlinear dynamics inherent in physical systems for information processing, has attracted attention in recent years because it can drastically reduce the computational resources required for information processing[1,2]. Nonlinearity, short-term memory, and high dimensionality are required for reservoirs that map input data nonlinearly into a high-dimensional feature space[3,4], and physical devices with these features are promising for PRC[1]. Many types of physical reservoirs have been reported, including memristors, optical devices, spintronics devices, soft bodies, nanowire networks, and ion-gating reservoirs. Further, information processing, including image recognition, spoken digit recognition, and time series prediction tasks, have been demonstrated using such physical reservoir devices[1,2,5–27]. The efficiency of PRC based on material-based computation makes it particularly promising for application to resource-limited edge AI devices. However, the computational performance of PRC is still insufficient for practical information processing tasks performed by such material-based efficient edge AI devices (e.g., dynamic image recognition and time series analysis tasks for in-situ processing of time series output from sensors, such as blood glucose level prediction and speech sentence recognition), an approach is required in order to substantially improve performance. One way to achieve this is deep reservoir computing (deep-RC), in which reservoirs are multilayered. This method is expected to be a promising approach, just as neural networks (NN) have been shown to achieve high expressive power and performance by utilizing deep layering. In full-simulation reservoir computing (RC), multilayering of the reservoir parts has been considered, and it has been reported that layering small reservoirs can improve performance in comparison to single-layered reservoirs with the same total reservoir size[28–33]. Furthermore, it has been reported that not only deepening the reservoir layer but also training the connection weights between layers according to the task improves the expressiveness of the network and the flexibility of the model, as well as the computational performance and efficiency[29,30].

[1]Research Center for Materials Nanoarchitectonics (MANA), National Institute for Materials Science (NIMS), 1-1 Namiki, Tsukuba, Ibaraki 305-0044, Japan. [2]Department of Applied Physics, Faculty of Science, Tokyo University of Science, Katsushika, Tokyo 125-8585, Japan. [3]Research Center for Functional Materials, NIMS, 1-1 Namiki, Tsukuba, Ibaraki 305-0044, Japan. [4]Research Network and Facility Services Division, NIMS, 1-2-1 Sengen, Tsukuba, Ibaraki 305-0047, Japan.
✉e-mail: TSUCHIYA.Takashi@nims.go.jp

In particular, Deep-Echo State Network, with hyperbolic tangents as a nonlinear function, has shown remarkable performance improvements in the prediction tasks of nonlinear autoregressive moving average models and chaotic dynamical systems when the connection weights between the layers are trained by linear regression with the targets[29,30]. On the other hand, few attempts at multilayering in physical reservoirs or physical NN have been reported, and those that have been reported are limited to methods that either do not train the connection weights between reservoirs (i.e., the network is not highly flexible)[34,35] or train the connection weights between reservoirs (or layers of physical NN) using backpropagation algorithms that require complex calculations that rely on external circuits and have large computational costs[33,36]. It is particularly notable that there are no reports of deep-RC with nanodevices that are advantageous for integration to realize practical AI devices, and thus, it is not clear that multilayering is effective in improving the performance of physical reservoirs.

Here, as the first implementation of nanodevice-based Deep-RC, we describe a demonstration of deep physical reservoir computing using an ion-gating reservoir (IGR), which is a compact and high-performance physical reservoir. The IGR is a nanodevice with a transistor structure consisting of a hydrogen-terminated diamond channel and a Li^+ electrolyte (Li-Si-Zr-O)[23,37], and the ion-electron coupled dynamics based on the electric double-layer (EDL) effect in the nanoregion near the Li^+ electrolyte/diamond interface is used as nonlinear dynamics for the reservoir computing.

There are two approaches to achieving high-performance PRC: one is to improve the performance by modifying the physical reservoir itself, and the other is to maximize the information processing capability of the physical system through system approaches such as optimization of the network structure and external data manipulation (e.g., pre-processing such as masking). In this study, we employed the latter approach, which improves PRC performance through deep network architecture, and used the IGR, which has achieved high computational performance in time series data analysis and image recognition tasks[23,26], as a model device for this approach.

First, we verified the computational performance of the deep-RC scheme reported for simulation-RC using IGR[28–30]. In this network, the connection weights between reservoir layers are trained based on a simple linear regression algorithm, which provides a higher network flexibility compared to the scheme in which the connection weights between reservoirs are not trained[34,35], and does not require a back-propagation algorithm. The backpropagation algorithm is an effective method that greatly improves the expressive power of the network, but it is difficult to apply to PRCs based on complex and dynamic nonlinearities (black box functions) originating from physical systems because it requires detailed information on the nonlinearities in the reservoir layer and their derivatives[33,36]. In this respect, the method of learning weights between layers by linear regression using targets is well suited for implementation in physical systems because it does not require detailed information on nonlinearities in physical systems (and their derivatives, etc.) and does not require inverse input of errors to physical systems in order to backpropagate the errors[28–30]. However, the conventional scheme with a simple series structure network reported for the simulation-RC does not improve the computational performance of the IGR. This was found to be due to the inherent characteristic of general physical reservoirs, which are sensitive to input conditions. In addition, this scheme does not provide any improvement in network size limitation, which is one of the main reasons that severely inhibited the performance of conventional PRCs. Whereas in simulation-RC it is easy to increase the reservoir size to the desired performance in exchange for computational cost, in PRCs the number of reservoir states (current and voltage response, mechanical vibration, optical response, etc.) obtained from the physical device is limited by the means of access to the physical system, such as measurement probes. Thus, it is difficult to increase the network size of such PRCs.

On the other hand, a deep-RC scheme with parallel structure overcome such limitations of physical reservoirs in principle, and succeeded in greatly increasing the number of reservoir states by utilizing the reservoir outputs of the previous layer as well as the final outputs among the layered device outputs. As a result, the performance of the IGR applied with the subject deep-RC scheme (deep-IGR) was drastically improved, with a 41% reduction in error compared to single-layer IGR in a second-order nonlinear autoregressive moving average (NARMA2) task. However, the model also showed an increase of unnecessary reservoir states leading to overlearning. Therefore, a modified model was developed to improve this model by evaluating the high dimensionality of the reservoir state in terms of the correlation coefficient between reservoir states, the number of layers was increased by excluding featureless reservoir states (which do not contribute to high dimensionality), which resulted in a 53% reduction in error for the NARMA2 task compared to single-layer IGR, with a normalized mean squared error (NMSE) of 9.08×10^{-3}. This is the best performance of any physical reservoir reported to date[22–25,38–40], outperforming a full-simulation RC[38] for the NARMA2 task.

Results

Ion-gating reservoirs using electric double-layer transistors

To demonstrate the applicability of the deep-RC architecture, we employed an IGR as the physical reservoir part, implemented with an EDL transistor composed of a Li^+ electrolyte (Li-Si-Zr-O) and hydrogen-terminated diamond, as shown in Fig. 1a. The IGR transistor has 9 channels of different lengths L_{ch} (=5, 10, 25, 35, 50, 100, 250, 350, 500 μm), and the 9 drain current responses can be nonlinearly transformed by the EDL mechanism to the input gate voltage signal. Figure 1b shows the drain current (I_D)-gate voltage (V_G) curves obtained from channels with lengths of $L_{ch} = 50$ μm and $L_{ch} = 500$ μm (upper panel) and a gate current (I_G) – V_G curve (lower panel). With a positive gate voltage applied, Li^+ in the electrolyte accumulates at the electrolyte/diamond interface, and electrons are injected into the diamond, which is a hole conductor, resulting in a high resistance state of the diamond. On the other hand, when a negative gate voltage is applied, negatively charged Li vacancies in the electrolyte accumulate on the diamond surface, forming an EDL at the electrolyte/diamond interface as shown in Fig. 1a. Then, holes are doped into the hydrogen-terminated diamond channel, resulting in a low-resistivity state and the drain current is nonlinearly modulated[37]. As different channel resistances exhibit different relaxation times for different channel lengths, if the gate input is a pulse signal, it will exhibit different drain current responses depending on the channel length, as shown in Fig. 1c, providing a higher dimension as a physical node[23]. The structure of this device, where different channel lengths coexist, provides current responses with different relaxation times. The coexistence in one system of a current response with a short relaxation time, which reflects a short-term experience and has strong nonlinearity, and a current response with a long relaxation time, which reflects a long-term experience and has relatively weak nonlinearity, can be expected to provide high performance[1,23,41]. In addition to these drain currents, the gate currents, obtained from the input gate terminals, show a spiked response, as shown in the bottom panel of Fig. 1c, which provides additional diversity to the IGR[24,26]. In redox-based IGR, the use of gate currents as reservoir states has been reported to improve computational performance[24]. In addition to the 10 physical nodes obtained by adding the gate current response to the 9 drain current responses, 10 nodes per pulse response were obtained as virtual nodes, as shown in the right-hand panel of Fig. 1c. Thus, the number of reservoir states X_i obtained from the IGR (i.e., nodes) is 100. Said reservoir states X_i were normalized from 0 to 1 and subsequently used for the reservoir part in the schematic of reservoir computing shown in Fig. 1d. The reservoir output $y(k)$ at discrete time step k is obtained by the linear combination of reservoir state $X_i(k)$ and readout weight W_i as follows;

$$\begin{aligned} y(k) &= \sum_{i=1}^{N} W_i X_i(k) + b \\ &= \boldsymbol{W}\boldsymbol{X}(k) + b \end{aligned} \tag{1}$$

where b is the bias; N is the reservoir size; $\boldsymbol{W} = (W_1, W_2, \ldots, W_N)$ is the readout weight vector; $\boldsymbol{X}(k)=[X_1(k), X_2(k), \ldots, X_N(k)]^T$ is the reservoir state vector. Figure 1e is a schematic of the deep-IGR which is a physically

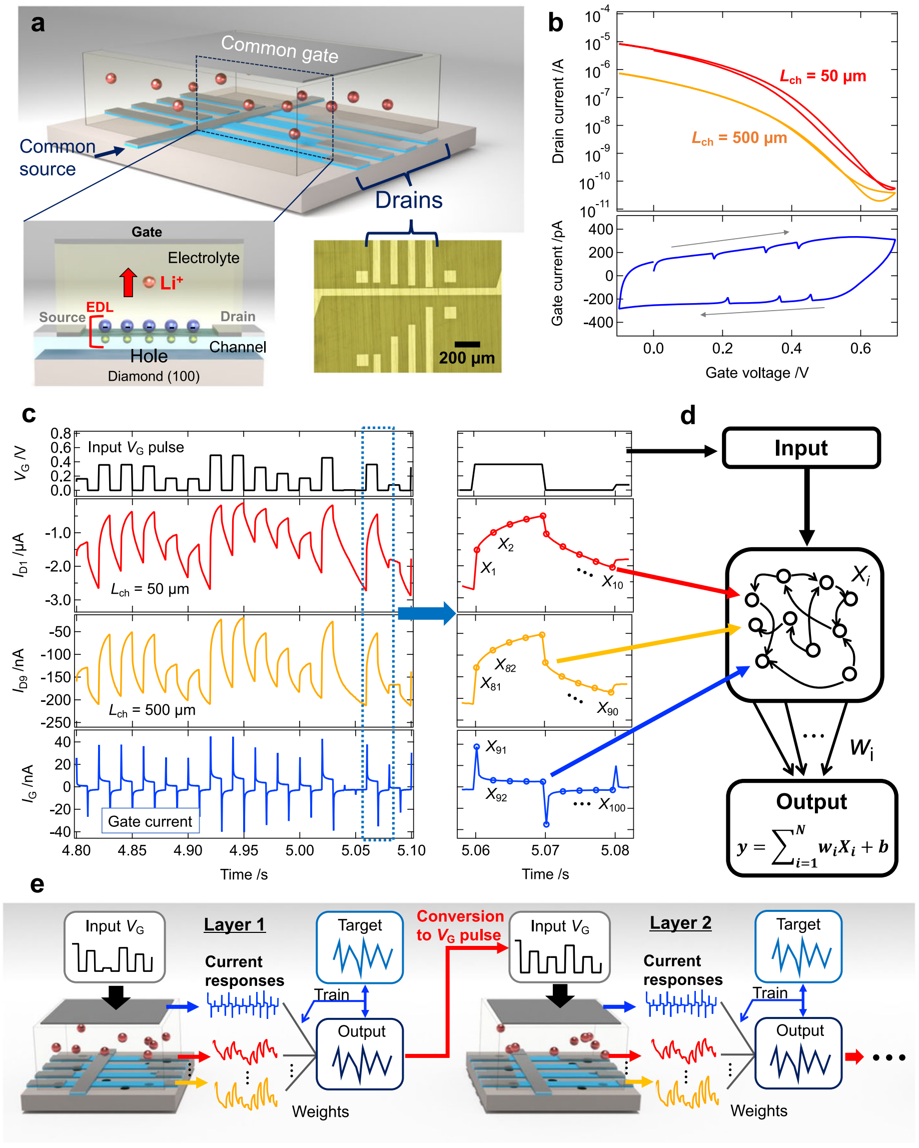

Fig. 1 | Physical reservoir computing using an ion gating reservoir (IGR). **a** Schematic of the IGR transistor. The inset is an optical microscope image of a diamond with source and drain electrodes. **b** I_D-V_G curves (upper panel) for channels with lengths of 50 µm and 500 µm and I_G-V_G curve (lower panel). **c** I_D responses (L_{ch} = 50 µm, 500 µm) and I_G response to pulsed V_G input. The right-hand panel shows how to obtain virtual nodes from such current responses. **d** Schematic of reservoir computing. **e** Schematic of the deep reservoir computing implemented by IGRs.

implemented deep reservoir computing with IGRs. In the first layer, as in conventional IGRs[23], the voltage-transformed input signal was input to the gate of the IGR, and the reservoir output was calculated by a linear sum of the weights and reservoir states (Eq. 1) obtained by acquiring virtual nodes from the obtained current responses, as shown in Fig. 1c. The weights were trained by linear regression so that the target and reservoir outputs matched. In deep-IGR, the reservoir output of the first layer is converted to a voltage pulse and input to the gate of the second layer IGR, and the reservoir output is obtained via weights using the obtained current response as in the first layer. The input that reproduced the target waveform to some degree in the first layer is again nonlinearly transformed by IGR into a higher dimensional feature space for learning, which allows the representation of target features that could not be represented in the first layer. The number of said layers can be increased by inputting the reservoir outputs of the previous layer, in the same way as in the second layer, for the third and subsequent layers. The performance of deep-IGR was evaluated by the error between the target and the reservoir output obtained by the input and forward propagation of a dataset different from the training dataset, with the weights of all layers fixed. The details of deep-IGR are discussed later in this document.

Before evaluating the performance of deep-IGR, we performed a NARMA2 task that predicts the NARMA2 model shown in Eq. 2 following a prior study[42], in order to evaluate the computational performance of the single-layer IGR.

$$y_t(k+1) = 0.4y_t(k) + 0.4y_t(k)y_t(k-1) + 0.6u^3(k) + 0.1 \qquad (2)$$

where $y_t(k)$ are the model outputs; $u(k)$ are the random inputs, ranging from 0 to 0.5. The NARMA2 task, which is widely used as a benchmark task for physical reservoirs, requires reservoirs to exhibit second-order nonlinearities and short-term memory[22–25,38–40]. Figure 2a shows a schematic of the NARMA2 task performed by IGR. The random input signal $u(k)$ was converted into voltage pulse streams with an intensity of 0 V to 0.5 V, a pulse period of T, and a duty cycle of D, then input to the gate terminal of the IGR transistor. The responses of the drain current to the gate voltage pulse streams were measured at a constant drain voltage of −0.5 V. As shown in Fig. 1c, 100 reservoir states X_i were obtained from 10 current responses and virtual nodes (i = 1, 2, …, 100). Furthermore, an additional 100 reservoir states X_i (i = 101, 102, …, 200) were obtained by applying an intensity-reversed input $u_{inv}(k)$ [= 0.5 - $u(k)$] to the IGR (inversion pulse method), resulting in a total of 200 reservoir states, the combination of which was utilized to obtain the reservoir output as shown in Eq. 1. Details on the inversion pulse method and reservoir size of the single IGR are given Supplementary Note 1 and Supplementary Figs. S1-3. In the training phase, the readout weights were trained by linear regression, in order to match the

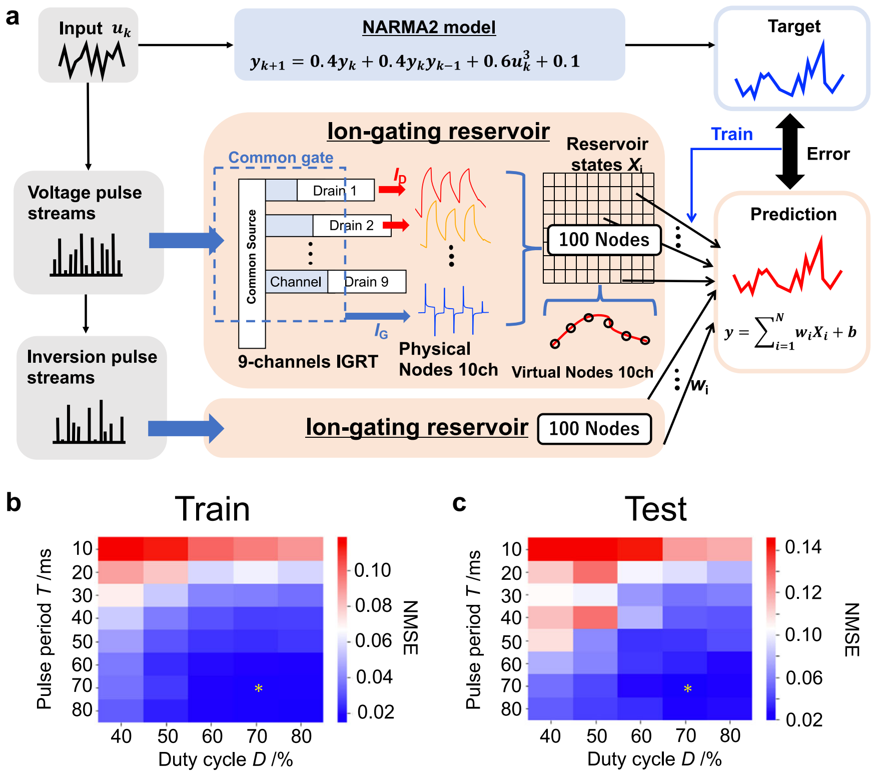

Fig. 2 | Performance evaluation of a single ion gating reservoir (IGR) by the second-order non-linear autoregressive moving average (NARMA2) task. a Schematic of the NARMA2 task performed by IGR, showing the pulse period T and duty cycle D dependence of normalized mean squared errors (NMSEs) for training (**b**) and test (**c**) phases. Optimal conditions are indicated by *.

reservoir output $y(k)$ and the model output $y_t(k)$. Details of the training algorithm are given in the Method section herein. In the test phase, performance was evaluated by NMSE (Eq. 3) of the reservoir output (prediction) $y(k)$ to the model output $y_t(k)$ for a different input $u(k)$ than in the training phase.

$$\mathrm{NMSE} = \frac{1}{M}\frac{\sum_{k=1}^{M}\left[y_t(k) - y(k)\right]^2}{\sigma^2\left[y_t(k)\right]} \tag{3}$$

where M is the data length ($M = 1600$ for the training phase and $M = 700$ for the test phase); $\sigma^2(\cdot)$ is the variance. Figure 2b and c show the pulse period T and duty cycle D dependence of NMSEs in the training and test phases, respectively. The best results, for both training and test phases, were obtained at $T = 70$ ms and $D = 70\%$, where the NMSE was 0.0157 in the training phase and 0.0194 in the test phase. Supplementary Fig. S4a shows the D dependence of NMSE at $T = 70$ ms, and Supplementary Fig. S4b shows the T dependence of NMSE at $D = 70\%$. Both of these are minima at the optimal conditions (indicated by *), indicating that the search for optimal conditions in single-layer IGR was successfully performed. In other words, this is the limit of the computational performance that can be achieved by adjusting the pulse period and duty cycle. Further performance improvements require consideration of voltage conditions (V_G and V_D) and preprocessing of the input signal (feature extraction, masking, etc.). Tuning these so-called 'hyperparameters' is difficult for physical reservoirs that require actual measurements, and the huge variety of parameters (voltage, time, temperature, number of masks, etc.) and their combinations make it extremely difficult to maximize the potential computational performance of the physical system. In all of the deep-IGR experiments discussed below, the voltage pulse conditions were fixed at $T = 70$ ms and $D = 70\%$. In this case, the optimal conditions match between training and testing, but if such is not the case, the optimal conditions in the training data should be adopted as the input conditions for the second and subsequent layers in order to avoid incorrect optimization by the testing data.

Performance evaluation of deep ion-gating reservoir with NARMA2 task

We experimentally demonstrated the deep-IGR shown in Fig. 3a,b as a framework that overcomes the limitations discussed above, and easily maximizes the computational performance of the physical system. Although the deep layered network structure shown in Fig. 3a (Network 1) has been reported to improve performance in a full simulation reservoir[28–30], there are no reports of its application to a physical reservoir. In the first layer of Network 1, the readout weight $W^{(1)}$ is trained with the same procedure as with the single-layer IGR shown in Fig. 2a, so as to obtain the first-layer reservoir output $y^{(1)}(k)$. In the L-th layer($L \geqq 2$), the voltage-transformed reservoir output $y^{(L-1)}(k)$ from the previous layer is input to the IGR instead of random input $u(k)$. The reservoir output of the L-th layer is then calculated by the linear combination of the reservoir state $X^{(L)}(k)$ obtained from the IGR and the readout weight $W^{(L)}$ trained by linear regression. In the test phase, the signal was propagated forward with the weights of all layers fixed, and the computational performance was evaluated by the error between the reservoir output $y^{(L)}(k)$ and the model output $y_t(k)$ (Eq. 2) at the final layer L. For details on the operating time in our Deep-RC scheme, please refer to Supplementary Note 3 and Supplementary Figs. S5 and S6. The black and gray plots in Fig. 3c show the dependence of the NMSE on the number of layers in the test phase and the training phase, respectively, for the NARMA2 task performed on Network 1 to the training data. In this task, the reservoir is used to predict the NARMA2 model output $y_t(k)$ from input $u(k)$. However, in the structure of Network 1 (deep layered), the nature of the problem changes after the second layer, and the task switches to predicting the NARMA2 model output using the reservoir prediction of the previous layer as input (task switching). In a physical reservoir that uses the transient response of a physical system in real time, the properties of the reservoir state (nonlinearity, memory capacity, and diversity) change strongly depending on the operating conditions (in this case, the input V_G pulse stream conditions), so that the accuracy in a given task changes due to sensitivity to the device operating conditions (i.e., as discussed in Fig. 2, the performance varies greatly with the operating conditions of the IGR.). Therefore, in this case, where the nature of the task has changed, different

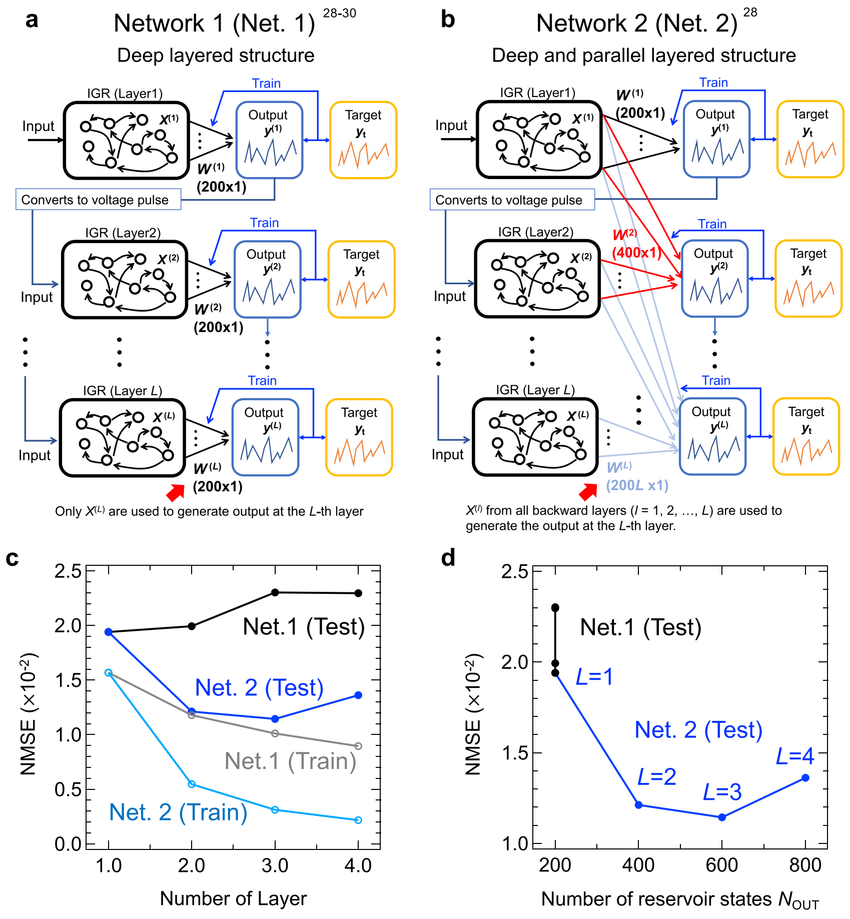

Fig. 3 | Performance evaluation of a deep-ion gating reservoir (deep-IGR) for Network 1 and Network 2 by the second-order nonlinear auto-regressive moving average (NARMA2) task. Schematic diagram of deep-IGR for (**a**) Network 1[28–30] and (**b**) Network 2[28]. **c** NMSEs of the NARMA2 task vs. the number of layers of deep-IGR. The black and blue plots show the results for Network 1 and Network 2, respectively. **d** Normalized mean squared errors (NMSEs) of the NARMA2 task vs the number of reservoir states used to generate output in each layer.

operating conditions for the IGR need to be explored because different characteristics are required for the reservoir. However, the method of searching for optimal operating conditions for each additional layer is not realistically achievable. In addition, although the task does not change considerably after the second layer, the input to the reservoir (i.e., the output of the previous layer) changes according to the learning status of the network, so optimization of the operating conditions is still considered necessary.

Therefore, in order to overcome such drawback and improve performance, without the need to search for optimal operating conditions, we considered Network 2 (Fig. 3b). This network is a modified model of Network1 (deep layered) that utilizes all the reservoir states obtained from the reservoirs of the previous layers $[X^{(1)}(k), X^{(2)}(k), \ldots, X^{(L)}(k)]$ so as to obtain the reservoir output of a given L-th layer (deep and parallel structure). In this scheme, higher dimensionality (the number of reservoir states used to obtain output) increases with the number of layers, so that the expressiveness of the reservoir also increases. Another major difference from Network 1 (deep layered) is that task switching does not occur as discussed in Network 1. Since the reservoir state $X^{(1)}(k)$ in the first layer is always used in the computation, regardless of the number of layers, the reservoir states in the second and subsequent layers can be interpreted as additional features that improve the accuracy of predicting the NARMA2 model from the reservoir states in the first layer. The blue and light blue plots show the dependence of the NMSE on the number of layers in the test phase and the training phase, respectively, for the NARMA2 task performed on Network 2 (deep and parallel layered)[28]. The NMSE decreases dramatically as the number of layers increases, and the error decreases by 41% at the third layer compared to the single layer. Figure 3d shows the relationship between NMSEs and the number of reservoir states used to generate output in final layer (N_{OUT}) for the two networks. Network 2 (deep and parallel layered) clearly shows a reduction in errors compared to Network 1(deep layered). This is possibly due to the fact that the number of nodes was effectively increased, while such increase is generally difficult to achieve with physical reservoirs, as mentioned above. However, in Network 2 (deep and parallel layered), the error increased slightly at the fourth layer, which increase effectively halted the performance improvement.

Node Selection in deep-IGR

The decrease in performance with respect to the increase in the number of nodes, which is shown in Fig. 3d, is thought to originate from the effect of overfitting due to the increase in the number of unnecessary nodes[43]. Overfitting in linear regression is determined by the combination of reservoir size and training data length. Therefore, if the training data length is limited, overfitting will occur as the reservoir size increases (especially the number of unnecessary nodes). To verify this hypothesis, we analyzed the high dimensionality in the correlation coefficient r_{AB} between node A and node B shown in Eq. 4 for the reservoir state used at the output of the fourth layer of this Network 2[24].

$$r_{\mathrm{AB}} = \frac{\sum_{k=1}^{M}\left(X_{\mathrm{A}}(k) - \bar{X}_{\mathrm{A}}\right)\left(X_{\mathrm{B}}(k) - \bar{X}_{\mathrm{B}}\right)}{\sqrt{\sum_{k=1}^{M}\left(X_{\mathrm{A}}(k) - \bar{X}_{\mathrm{A}}\right)^{2}\sum_{k=1}^{M}\left(X_{\mathrm{B}}(k) - \bar{X}_{\mathrm{B}}\right)^{2}}} \tag{4}$$

where $X_{\mathrm{A}}(k)$ is the reservoir state of node A and $\bar{X}_{\mathrm{A}}$ is the average value of $X_{\mathrm{A}}(k)$. Figure 4a shows the correlation coefficients $|r_{\mathrm{AB}}|$ between all reservoir states $X_i^{(L)}$ $(i = 1 \sim 200, L = 1 \sim 4)$ used to generate output in the fourth layer of Network 2 (i.e., 800 nodes in total). In the first layer, a random wave $u(k)$ was input to the IGR, whereas in layer L ($\geqq 2$), the output $y^{(L-1)}(k)$ of the

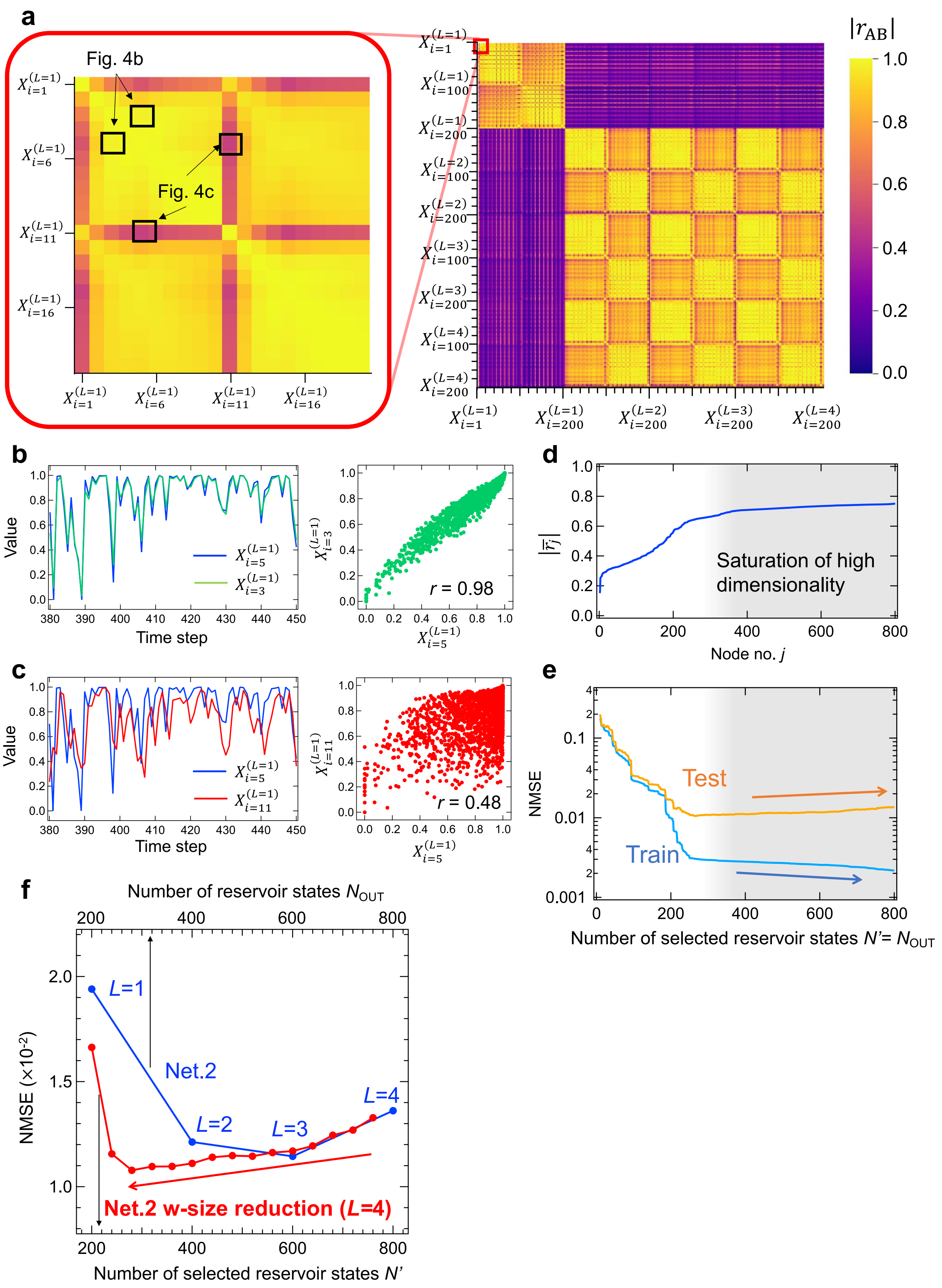

Fig. 4 | Node selection in deep-ion gating reservoir (deep-IGR). **a** Heatmap of the correlation coefficients of the reservoir states used in the output for the 4th layer of Network 2. An example of (**b**) a strongly correlated or (**c**) relatively weakly correlated reservoir state and a scatterplot. **d** The $|\bar{r}_j|$ plots, rearranging the node numbers j in order from the lowest $|\bar{r}_j|$. **e** The relationship between the number of selected nodes N' and the normalized mean squared error (NMSE) of the second-order nonlinear autoregressive moving average (NARMA2) task. **f** N_{OUT} vs NMSEs of NARMA2 task.

previous layer was input to the IGR. Therefore, because these correlation coefficients are small, due to the difference in inputs, the $|r_{AB}|$ in the first layer is very different from that in the rest of the layers, as shown in Fig. 4a. The inset of Fig. 4a shows an expanded part of the reservoir state in the first layer. The other major correlated/uncorrelated node combinations were also achieved by the following node combinations. (1) gate current/drain current, (2) original pulse/inverted pulse, (3) virtual node corresponding to pulse on time/pulse interval, etc. For example, the correlation coefficient $|r|$ between $X_{i=5}^{(L=1)}$ and $X_{i=3}^{(L=1)}$ is close to 1, and they exhibit similar behavior, as shown in Fig. 4b. They correspond to different virtual nodes taken from the same drain current (L_{ch} = 5 μm). On the other hand, $X_{i=5}^{(L=1)}$ and $X_{i=11}^{(L=1)}$ have a relatively low correlation coefficient $|r|$ of 0.48 and exhibit different behavior, as shown in Fig. 4c. These nodes correspond to different virtual nodes taken from different drain currents (L_{ch} = 5 μm and L_{ch} = 10 μm).

To identify which of the 800 nodes shown in Fig. 4a has the lower correlation coefficient (i.e., contributes to the higher dimensionality), the average value of the correlation coefficient for node j (=1, …, 800) was calculated as shown in Eq. 5;

$$\left|\bar{r}_j\right| = \frac{\sum_{i\neq j}^{N_{all}} \left|r_{ji}\right|}{N_{all} - 1} \tag{5}$$

where N_{all} is the number of all nodes (in this case, N_{all} = 800). The $|\bar{r}_j|$ plots, rearranging the node numbers j in order from the lowest $|\bar{r}_j|$, are shown in Fig. 4d. The lowest $|\bar{r}_1|$ is 0.3, whereas $|\bar{r}_j|$ increases with increasing j, indicating that $|\bar{r}_j|$ saturates at about j = 300. This suggests that about 500 of the total 800 node reservoir states are nodes that do not contribute to high dimensionality (i.e., the nodes are less effective for performing the given task). To evaluate the effect of node correlation coefficients and high dimensionality on computational performance, the NARMA2 task was performed by increasing number of selected reservoir states N' in the order of lower $|\bar{r}_j|$ (for example, when N' = 100, X_j(j = 1 ~ 100) was used for the calculation). Note that N' = N_{OUT} in this case, where node selection is made only on the final layer (L = 4). Figure 4e shows the relationship between N' and NMSE; in the region where N' is approximately 260 or less, N' increases while NMSE decreases for both training and testing. On the other hand, in the region where N' is above approximately 260, N' increases and the training error continues to decrease, while the test error increases. Also, as shown in Supplementary Fig. S7, the derivative of the test error converges to a slightly positive value in the region where N' is above about 270. This indicates the effect of overfitting with increasing reservoir size, in which reservoir states in regions of saturated diversity have a negative impact on the computation. Figure 4f shows the results obtained with size reduction (Fig. 4e) and the relationship between the NMSEs of the NARMA2 task (test phase) and N_{OUT} for Network 2. The red plots show the results of the computation when gradually excluding nodes, in order of largest $|\bar{r}_j|$, from the 800 nodes which were used to generate output in the fourth layer of Network 2. It was found that the NMSE decreased despite any reduction in the number of reservoir states used to generate the output. Reducing to N' = 600 (i.e., excluding from the calculation the 200 nodes with large $|\bar{r}_j|$), the improvement in computational performance was only about the same as in the third layer without size reduction, but at N' = 460 (i.e., excluding from the calculation the 340 nodes with large $|\bar{r}_j|$), it rather outperformed an interpolated version of Network 2 without size reduction between L = 2 and L = 3. The performance continued to improve as the number of nodes was reduced to about N' = 300, but when N' was further reduced, the performance rapidly degraded due to the loss of the nodes necessary for the computation being performed. These results indicate that the adverse effects of multilayering, such as unnecessary node growth and overfitting, were successfully suppressed without sacrificing the advantages of improved computational performance due to multilayering. Here, network pruning was performed by selectively incorporating uncorrelated nodes in the network through correlation coefficient analysis. This pruning method effectively exploits the feature of RC that its computational performance is achieved by high dimensionality. Therefore, the pruning method based on the correlation coefficient is useful for identifying nodes that lead to higher dimension and identifying performance improvement mechanisms that lead to higher dimension and performance improvement, although other methods, such as network pruning based on network structure analysis can also be used in our Deep-RC.

Deep-IGR architecture, which utilizes node selection at each layer

The node selection (reduction) by $|\bar{r}_j|$ was adapted to only one layer in the configuration shown in Fig. 4. Here, in order to maximize computational performance, we consider Network 3, in which node selection is adapted to all layers from the second layer onward by modifying Network 2 (deep and parallel layered). The number of selected reservoir states N' used for the output of each layer was set to 400. Therefore, the first (single) and second layers, where the number of reservoir states N_{OUT} is 200 and 400, respectively, do not perform node selection and are therefore identical to Network 2, represented in Fig. 3b. The calculation method used for the third and subsequent layers of Network 3 is explained below. The output in the third layer of Network 2 used 600 reservoir states of $X^{(L=1)}$, $X^{(L=2)}$ and $X^{(L=3)}$ [here, these reservoir states are rewritten as X_j (j = 1, 2,…, 600)]. In Network 3, of these 600 reservoir states, N' nodes (here 400 nodes) selected as $X'^{(L=3)}$ in order of smallest $|\bar{r}_j|$ are used to generate the output of the third layer $y^{(L=3)}$. Furthermore, by voltage converting $y^{(L=3)}$ and inputting it to IGR again, $X^{(L=4)}$ (200 nodes) is obtained, and together with the reservoir states $X'^{(L=3)}$ (400 nodes) obtained in the previous layers, there are a total of 600 reservoir states [here, these reservoir states are rewritten as X_j (j = 1, 2,…, 600)]. As above, N' (=400) of these 600 nodes with small $|\bar{r}_j|$ were selected $X'^{(L=4)}$ and used to generate output in the fourth layer $y^{(L=4)}$. Figure 5a and b show schematic diagrams of how the reservoir outputs $y^{(L)}$ of Network 2 and Network 3 in layer L are computed, respectively. In Network 2, shown in Fig. 5a, the number of reservoir states used for output increases by 200 as the number of layers increases, because all reservoir states $X^{(L=1)}$, $X^{(L=2)}$, …, $X^{(L)}$ in layers 1 to L are used to generate the reservoir output in layer L. On the other hand, in Network 3, shown in Fig. 5b, $X'^{(L)}$ selected by N' nodes in order of smallest $|\bar{r}_j|$ of the reservoir state $X'^{(L-1)}$ selected up to the previous layer and the reservoir state $X^{(L)}$ in Layer L, as described above. Therefore, the number of reservoir states used to generate output is fixed at N', regardless of the number of layers L (>1). Figure 5c shows a plot of NMSE vs. the number of layers in the NARMA2 task for Network 3 (deep and parallel layered with node selection), with the number of selected nodes N' = 400. The NMSE continued to decrease as the number of layers increased, for both training and testing errors, with the fourth layer achieving better computational performance than Network 2 (deep and parallel layered), shown by the dotted blue line, with an NMSE of 0.00326 in the training phase and 0.00921 in the testing phase. These values are 79% and 53% lower in the training and testing phases, respectively, compared to a single-IGR. Figure 5d shows the relationship between the number of nodes used for output N_{OUT} and NMSE (test phase). Network 3 showed improved computational performance over the conventional Network 2, despite N_{OUT} after the second layer being fixed at 400 (= N'). This performance improvement is explained by the network being trained by increasing the number of layers while excluding unnecessary nodes that cause overfitting and nodes that do not contribute to diversity, thereby effectively incorporating the features obtained in each layer. The predicted and target waveforms at layers 1 and 4 are shown in Fig. 5e and f, respectively. Although the predicted waveform in the first layer captured the trend of the target waveform, there was a divergence between the predicted waveform and the target waveform in some areas. On the other hand, the predicted waveforms at the fourth layer, shown in Fig. 5f, are in almost perfect agreement. This shows that the deep-IGR based on Network3 is able to successfully solve the NARMA2 model shown in Eq. 2, and that this architecture drastically improved the computing performance of the IGR. Supplementary Fig. S8 shows the N' dependence of NMSE during the test phase of the 3-layer Deep-IGR (Network3). For the deep layered scheme, the

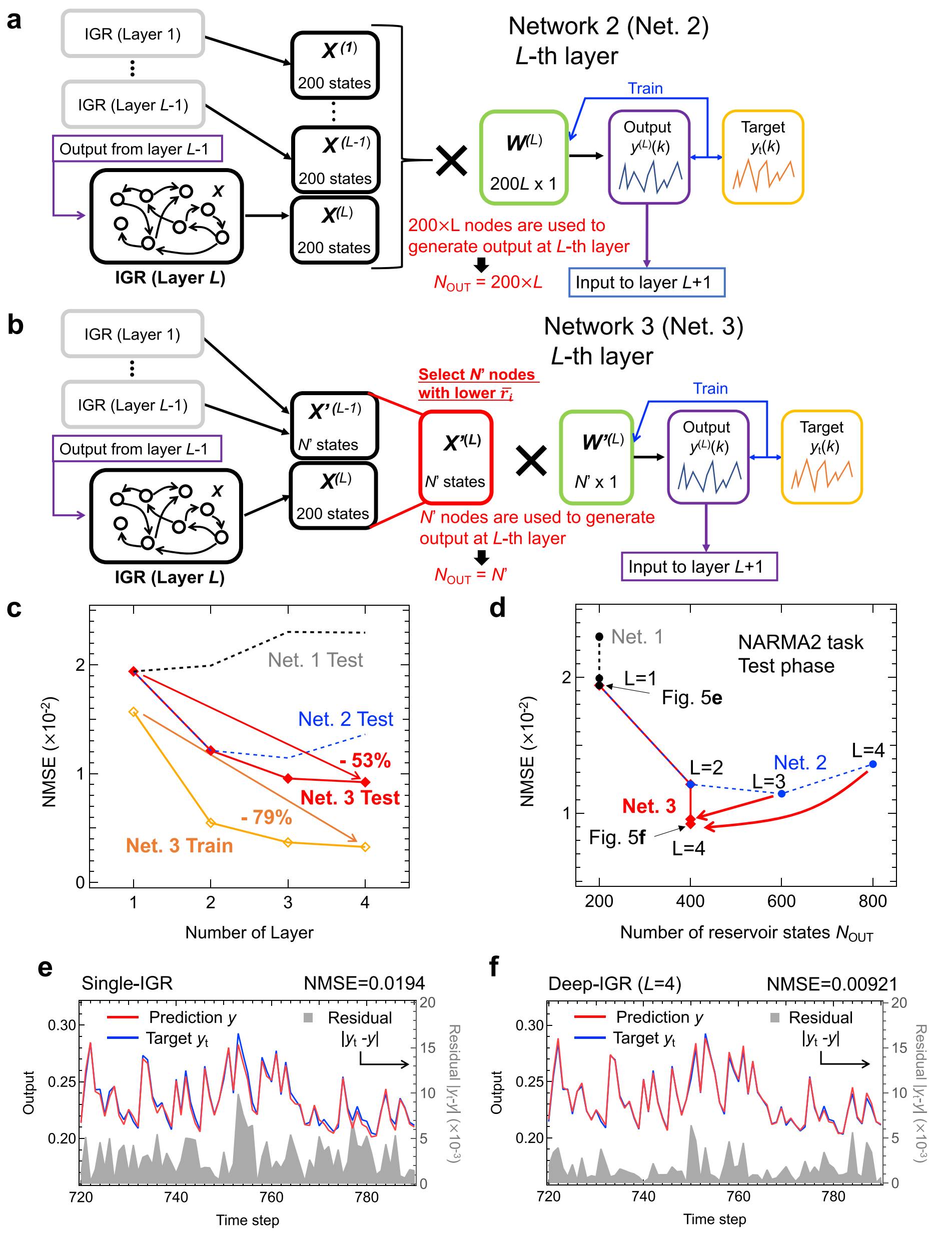

Fig. 5 | Performance evaluation of a deep-ion gating reservoir (deep-IGR) for Network 3 by the second-order nonlinear autoregressive moving average (NARMA2) task. Schematic diagram of layer L of a deep-IGR for (**a**) Network 2 and (**b**) Network 3 with node selection. **c** Normalized mean squared errors (NMSEs) of the NARMA2 task vs. the number of layers of deep-IGR. **d** NMSEs of the NARMA2 task vs N_{OUT}. Target waveform and predicted waveform by IGR at (**e**) $L = 1$ and (**f**) $L = 4$.

test error was minimal when $N' = 400$, therefore, $N' = 400$ is set here. Figure 6a shows a comparison of the computational performance evaluated in the NARMA2 task of this study with other physical reservoirs reported so far[22–25,38–40]. Here, in order to maximize the performance of the deep-IGR, the output weights of the final layer ($L = 4$) of Network 3 were trained with ridge regression. Our deep-IGR (Network 3, $L = 4$) achieved NMSE = 9.08×10^{-3} in the test phase for the NARMA2 task (NMSE = 3.37×10^{-3} in the training phase), and achieved the highest computational performance of any physical reservoir reported to date for the NARMA2 task[22–25,38–40], even outperforming the NMSE = 0.016 of fully simulated reservoir computing [Standard Echo State Network (ESN)][38]. This standard ESN is a single-reservoir with 300 nodes of a hyperbolic tangent as the activation function[38], and differs from Deep-IGR (400 nodes and four multilayers) in its network architecture and reservoir size. Although they are not a perfect like-for-like comparison, ESN, which is a typical example of a simulation reservoir, has been widely compared in performance with many physical reservoirs[1,38,44–46]. Simulation reservoirs that can adjust the network structure and reservoir size according to the purpose generally perform better than PRCs, and actually no PRCs that outperform the reported standard ESNs have been reported for the NARMA2 task, as shown in Fig. 6a. Therefore, this comparison has a certain value because our Deep-IGR is the first physical system to outperform this reported simulation reservoir in the NARMA2 task[22–25,38–40]. In particular, the NARMA2 task is one of the best tasks for evaluating and comparing the basic information processing capabilities of PRC, since the task of analyzing second-order dynamic systems is widely performed by PRC[1,10,13,22–25,38–40,47,48]. On the other hand, the NARMA10 task is considered as a more challenging benchmark task, but Deep-IGR was unable to solve the NARMA10 task. This is due to the relatively small memory capacity (MC) of IGR (MC ~ 4). It is expected that if our Deep-RC architecture is implemented in a dynamical system with high memory capacity (e.g., optical elements and spin-wave interference)[5,6,8,25], the NARMA10 task can be solved with high accuracy. We would like to demonstrate this as future work.

The deep-RC architecture proposed in this study has very few hyperparameters, and can easily enhance the computational performance of physical reservoirs. Usually, it is extremely difficult to maximize the information processing capability of a physical system in physical reservoir computing. There is a wide variety of parameters to be considered if the best computational performance is to be obtained from a physical reservoir;

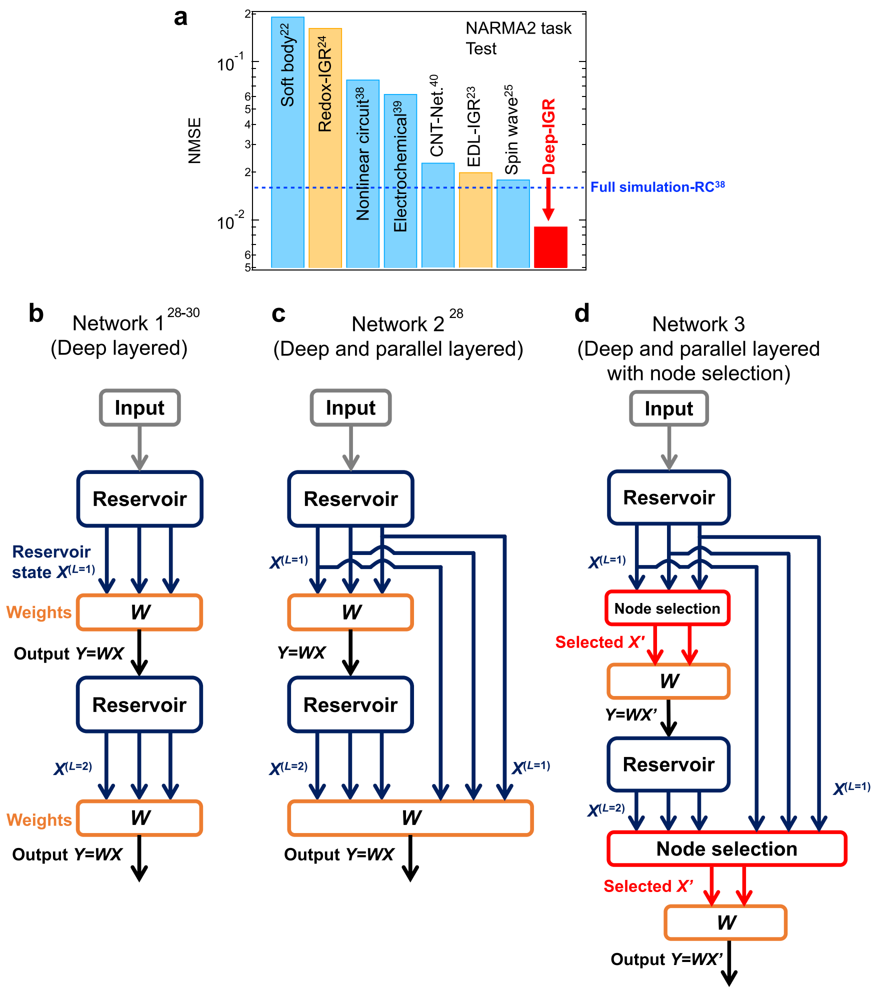

Fig. 6 | Performance comparison with other physical reservoirs and summary of deep-reservoir computing (deep-RC) structures in this study. **a** Performance comparison with other physical reservoirs by the second-order nonlinear auto-regressive moving average (NARMA2) task[22–25,38–40]. The result for soft body[22] was converted to the normalized mean squared error (NMSE) in Eq. 3 for comparison. Schematic diagram of the Deep-RC architecture for physical reservoir computing (PRC) investigated in this study for (**b**) Network 1[28–30], (**c**) Network 2[28] and (**d**) Network 3.

these include the delay time of the feedback loop, the mask matrix used for input signal preprocessing (mask format, number of masks, mask length, etc.), in addition to the intensity and frequency of the signal input to the physical system. Such complexity makes it almost impossible to experimentally search for the best combination of driving conditions for a physical system. Hence, the deep-layered physical reservoir architecture proposed in this study can dramatically improve the computational performance, and can be realized in a small parameter space that is possible to realistically explore. We experimentally examined three different deep reservoir architectures, which are shown in Fig. 6b-d, and found that Network 1 (deep layered, shown in Fig. 6b), which was previously reported for simulated reservoirs, does not necessarily contribute to improved performance in physical reservoirs[28–30]. Said scheme is suitable only as a component part of physical reservoirs, whose characteristics are not so sensitive to operating conditions. Further, it does not increase the network size, which is a serious problem faced by physical reservoirs. On the other hand, we have succeeded in increasing the network size and in improving the computational performance of the physical reservoir drastically with Network 3 (deep and parallel layered with node selection), which is a modified version of Network 2 (deep and parallel layered, shown in Fig. 6c) that selectively uses useful nodes for information processing, as shown in Fig. 6d. These deep-RC schemes should be practically implemented by integrating an FPGA and IGRs, as shown in Supplementary Fig. S9, which we would like to address as future work. Our deep-IGR is the first nanodevice implementation of deep-RC, and the dramatic improvement of the performance of IGR by our architecture provides the possibility of realizing large-scale, brain-like physical devices.

Conclusion

In this study, we demonstrated the implementation of Deep-RC by nanodevices and the improvement of computational performance achieved by optimizing the deep network structure, whereas Deep-RC has only been implemented by simulated RCs and limited physical reservoirs. We implemented IGR that uses ion-electron coupling dynamics as a computational resource[23] in the reservoir part, so as to create a multi-layered framework, and found that a simple serial network structure did not improve the performance (Network 1). On the other hand, we determined that a network structure that uses the reservoir state of the previous layer for outputs shows a remarkable improvement in performance (Network 2). This is because, in addition to succeeding in further increasing the dimension, which is generally difficult in physical systems, the predicted output of the reservoir is again input to the IGR and transferred to the high-dimensional feature space, where it is learned so that the deviation from the target output is reduced. In order to suppress overlearning due to increase of unnecessary nodes, which was confirmed in this network structure, node selection based on correlation coefficients was used in Network 3 to effectively extract nodes that are effective for information processing. As a result, an NMSE of 9.08×10^{-3} was achieved in the NARMA2 task for deep-IGR. This is the best performance of all the physical reservoirs reported so far for the NARMA2 prediction task which is a typical benchmark task for RCs and PRCs, and even outperforms full-simulation RC[22–25,38–40]. The easy and dramatic performance improvement of IGRs with our deep-RC architecture, and opens the way to the implementation of practical, large-scale, brain-based physical devices.

Method

Device fabrication and electrical measurements

Hydrogen-terminated diamond homoepitaxial film was deposited, as a channel, on a IIa-type high-pressure high-temperature single-crystal diamond substrate (100) (EDP) by the microwave-plasma chemical vapor deposition (MPCVD) method. During the deposition process, 500 and 0.5 standard cubic centimeters per minute of H_2 and CH_4, respectively, were supplied, and the hydrogen-terminated diamond was grown at a radio frequency of 950 W. The IGR were fabricated with nine different channel lengths (5, 10, 25, 35, 50, 100, 250, 350, and 500 μm), all with a channel width of 50 μm. Pd/Pt electrodes (10 and 35 nm, respectively) were deposited as source and drain electrodes by electron beam evaporation with maskless lithography after oxygen termination of the diamond surfaces (other than channels) by oxygen plasma asher. A 3.5-μm LSZO thin film, used as an electrolyte, was deposited by pulsed laser deposition (PLD) with an ArF excimer laser. A 100-nm Au thin film was deposited by electron beam deposition as a gate electrode.

Electrical measurements of IGR were performed by the source measure unit and pulse measure unit of a semiconductor parameter analyzer (4200A-SCS, Keithley), which measurements were carried out at room temperature inside a vacuum chamber that had been evacuated by a turbo molecular pump. Probers were used to connect the IGR inside the chamber.

Linear regression algorithm for learning readout networks

In the NARMA2 task, which was used in this study to evaluate performance, the readout network was trained with linear regression using the algorithm described below. The reservoir output shown in Eq. 1 can also be expressed as;

$$y(k) = \boldsymbol{W} \cdot \boldsymbol{x}(k) \tag{6}$$

where $\boldsymbol{W} = (b, W_1, W_2, \ldots, W_N)$ and $\boldsymbol{x}(k) = [1, X_1(k), X_2(k), \ldots, X_N(k)]^{\mathrm{T}}$ are the weight vector and the reservoir state vector, respectively. The reservoir output $\mathbf{Y}$ for all training periods ($k = 1, 2, \ldots, M$) is described by;

$$\mathbf{Y} = \mathbf{W}\mathbf{X} \tag{7}$$

where $\mathbf{X} = (\boldsymbol{x}(1), \boldsymbol{x}(2), \ldots, \boldsymbol{x}(M))$ and $M = 1600$ are the reservoir state matrix and the data length for the training phase, respectively. The weight matrix that minimizes the squared error is given by;

$$\mathbf{W} = \mathbf{Y}_{\mathrm{t}}\mathbf{X}^{\dagger} \tag{8}$$

where $\mathbf{X}^{\dagger} = [\mathbf{X}^{\mathrm{T}}(\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}]$ and $\mathbf{Y}_{\mathrm{t}}$ are the Moore-Penrose pseudo-inverse matrix and the target matrix, respectively. In the results shown in Fig. 6a, the readout weights of the final layer of Deep-IGR (Network 3) were trained with the ridge regression as follows.

$$\mathbf{W} = \mathbf{Y}_{\mathrm{t}}\mathbf{X}^{\mathrm{T}}(\mathbf{X}\mathbf{X}^{\mathrm{T}} + \lambda\mathbf{I})^{-1} \tag{9}$$

where λ (=2×10^{-3}) and $\mathbf{I}$ ($\subseteq \mathbb{R}^{N\times N}$) are the ridge parameters and identity matrix, respectively. The learning and the sum-of-product calculation performed in the readout network were done on a personal computer using current data obtained from the IGR. However, by utilizing artificial synaptic devices that reproduce weights by conductance, the sum-of-product calculation performed in the readout can also be calculated in a physical process, which is expected to further improve efficiency[49–58].

Data availability

The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Code availability

The codes used in the current study are available from the corresponding author on reasonable request.

Received: 6 October 2023; Accepted: 11 June 2024;
Published online: 19 June 2024

References

1. Tanaka, G. et al. Recent advances in physical reservoir computing: A review. *Neural Netw.* **115**, 100–123 (2019).
2. Nakajima, K. Physical reservoir computing—an introductory perspective. *Jpn. J. Appl. Phys.* **59**, 060501 (2020).
3. Jaeger, H. *The 'echo state' approach to analysing and training recurrent neural networks-with an Erratum note 1*. *Germany*. *Ger. Natl Res. Cent. Inf. Technol. GMD Tech. Rep.* **148**, 13 (2001).
4. Jaeger, H. & Haas, H. Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication. *Science* **304**, 78–80 (2004).
5. Paquot, Y. et al. Optoelectronic reservoir computing. *Sci. Rep.* **2**, 287 (2012).
6. Van der Sande, G., Brunner, D. & Soriano, M. C. Advances in photonic reservoir computing. *Nanophotonics* **6**, 561–576 (2017).
7. Torrejon, J. et al. Neuromorphic computing with nanoscale spintronic oscillators. *Nature* **547**, 428–431 (2017).
8. Nakane, R., Tanaka, G. & Hirose, A. Reservoir computing with spin waves excited in a garnet film. *IEEE A ccess* **6**, 4462–4469 (2018).
9. Tsunegi, S. et al. Physical reservoir computing based on spin torque oscillator with forced synchronization. *Appl. Phys. Lett.* **114**, 164101 (2019).
10. Jiang, W. et al. Physical reservoir computing using magnetic skyrmion memristor and spin torque nano-oscillator. *Appl. Phys. Lett.* **115**, 192403 (2019).
11. Akashi, N. et al. Input-driven bifurcations and information processing capacity in spintronics reservoirs. *Phys. Rev. Res.* **2**, 043303 (2020).
12. Sillin, H. O. et al. A theoretical and experimental study of neuromorphic atomic switch networks for reservoir computing. *Nanotechnology* **24**, 384004 (2013).
13. Du, C. et al. Reservoir computing using dynamic memristors for temporal information processing. *Nat. Commun.* **8**, 2204 (2017).
14. Moon, J. et al. Temporal data classification and forecasting using a memristor-based reservoir computing system. *Nat. Electron.* **2**, 480–487 (2019).
15. Midya, R. et al. Reservoir computing using diffusive memristors. *Adv. Intell. Syst.* **1**, 1900084 (2019).
16. Zhu, X., Wang, Q. & Lu, W. D. Memristor networks for real-time neural activity analysis. *Nat. Commun.* **11**, 2439 (2020).
17. Sun, L. et al. In-sensor reservoir computing for language learning via two-dimensional memristors. *Sci. Adv.* **7**, eabg1455 (2021).
18. Hochstetter, J. et al. Avalanches and edge-of-chaos learning in neuromorphic nanowire networks. *Nat. Commun.* **12**, 4008 (2021).
19. Zhong, Y. et al. Dynamic memristor-based reservoir computing for high-efficiency temporal signal processing. *Nat. Commun.* **12**, 408 (2021).
20. Milano, G. et al. In materia reservoir computing with a fully memristive architecture based on self-organizing nanowire networks. *Nat. Mater.* **21**, 195–202 (2022).
21. Nakayama, J., Kanno, K. & Uchida, A. Laser dynamical reservoir computing with consistency: an approach of a chaos mask signal. *Opt. Express* **24**, 8679–8692 (2016).
22. Nakajima, K., Hauser, H., Li, T. & Pfeifer, R. Information processing via physical soft body. *Sci. Rep.* **5**, 10487 (2015).
23. Nishioka, D. et al. Edge-of-chaos learning achieved by ion-electron–coupled dynamics in an ion-gating reservoir. *Sci. Adv.* **8**, eade1156 (2022).
24. Wada, T. et al. A Redox-based Ion-Gating Reservoir, Utilizing Double Reservoir States in Drain and Gate Nonlinear Responses. *Adv. Intell. Syst*. **5**, 2300123 (2023).

25. Namiki, W. et al. Experimental Demonstration of High-Performance Physical Reservoir Computing with Nonlinear Interfered Spin Wave Multi-Detection. *Adv. Intell. Syst*. **5**, 2300228 (2023).
26. Takayanagi, M. et al. Ultrafast-switching of an all-solid-state electric double layer transistor with a porous yttria-stabilized zirconia proton conductor and the application to neuromorphic computing. *Mater. Today Adv.* **18**, 100393 (2023).
27. Appeltant, L. et al. Information processing using a single dynamical node as complex system. *Nat. Commun.* **2**, 468 (2011).
28. Hasegawa, H., Kanno, K. & Uchida, A. Parallel and deep reservoir computing using semiconductor lasers with optical feedback. *Nanophotonics* **12**, 869–881 (2023).
29. Akiyama, T. & Tanaka, G. Computational Efficiency of Multi-Step Learning Echo State Networks for Nonlinear Time Series Prediction. *IEEE Access* **10**, 28535–28544 (2022).
30. Akiyama, T. & Tanaka, G. *Analysis on Characteristics of Multi-Step Learning Echo State Networks for Nonlinear Time Series Prediction; Analysis on Characteristics of Multi-Step Learning Echo State Networks for Nonlinear Time Series Prediction*. *2019 International Joint Conference on Neural Networks (IJCNN)* 1–8 (2019).
31. Gallicchio, C. & Micheli, A. Echo State Property of Deep Reservoir Computing Networks. *Cogn. Comput.* **9**, 337–350 (2017).
32. Goldmann, M., Köster, F., Lüdge, K. & Yanchuk, S. Deep time-delay reservoir computing: Dynamics and memory capacity. *Chaos* **30**, 093124 (2020).
33. Nakajima, M. et al. Physical deep learning with biologically inspired training method: gradient-free approach for physical hardware. *Nat. Commun.* **13**, 7847 (2022).
34. Liu, K. et al. Multilayer Reservoir Computing Based on Ferroelectric α-In2Se3 for Hierarchical Information Processing. *Adv. Mater.* **34**, 2108826 (2022).
35. Lin, B.-D. et al. Deep time-delay reservoir computing with cascading injection-locked lasers. *IEEE J. Sel. Top. Quantum Electron.* **29**, 1–8 (2022).
36. Wright, L. G. et al. Deep physical neural networks trained with backpropagation. *Nature* **601**, 549–555 (2022).
37. Tsuchiya, T. et al. The electric double layer effect and its strong suppression at Li+ solid electrolyte/hydrogenated diamond interfaces. *Commun. Chem.* **4**, 117 (2021).
38. Kan, S. et al. Simple reservoir computing capitalizing on the nonlinear response of materials: theory and physical implementations. *Phys. Rev. Appl.* **15**, 024030 (2021).
39. Kan, S., Nakajima, K., Asai, T. & Akai-Kasaya, M. Physical implementation of reservoir computing through electrochemical reaction. *Adv. Sci.* **9**, 2104076 (2022).
40. Akai-Kasaya, M. et al. Performance of reservoir computing in a random network of single-walled carbon nanotubes complexed with polyoxometalate. *Neuro. Comput. Eng.* **2**, 014003 (2022).
41. Inubushi, M. & Yoshimura, K. Reservoir computing beyond memory-nonlinearity trade-off. *Sci. Rep.* **7**, 10199 (2017).
42. Atiya, A. F. & Parlos, A. G. New results on recurrent network training: unifying the algorithms and accelerating convergence. *IEEE Trans. Neural Netw.* **11**, 697–709 (2000).
43. Dambre, J., Verstraeten, D., Schrauwen, B. & Massar, S. Information processing capacity of dynamical systems. *Sci. Rep.* **2**, 514 (2012).
44. Maksymov, I. S. Analogue and physical reservoir computing using water waves: Applications in power engineering and beyond. *Energies* **16**, 5366 (2023).
45. Maksymov, I. S. & Pototsky, A. Reservoir computing based on solitary-like waves dynamics of liquid film flows: A proof of concept. *Europhys. Lett.* **142**, 43001 (2023).
46. Maksymov, I. S., Pototsky, A. & Suslov, S. A. Neural echo state network using oscillations of gas bubbles in water. *Phys. Rev. E* **105**, 044206 (2022).
47. Nishioka, D., Shingaya, Y., Tsuchiya, T., Higuchi, T. & Terabe, K. Few- and single-molecule reservoir computing experimentally demonstrated with surface enhanced Raman scattering and ion-gating. *Sci. Adv.* **10**, eadk6438 (2024).
48. Shibata, K. et al. Redox-based ion-gating reservoir consisting of (104) oriented $LiCoO_2$ film, assisted by physical masking. *Sci. Rep.* **13**, 21060 (2023).
49. Arnold, A. J. et al. Mimicking neurotransmitter release in chemical synapses via hysteresis engineering in MoS2 transistors. *ACS Nano* **11**, 3110–3118 (2017).
50. Ielmini, D. & Wong, H. S. P. In-memory computing with resistive switching devices. *Nat. Electron.* **1**, 333–343 (2018).
51. Ielmini, D. Brain-inspired computing with resistive switching memory (RRAM): Devices, synapses and neural networks. *Microelectron. Eng.* **190**, 44–53 (2018).
52. Zhu, J. et al. Ion gated synaptic transistors based on 2D van der Waals crystals with tunable diffusive dynamics. *Adv. Mater.* **30**, 1800195 (2018).
53. Kumar, S., Williams, R. S. & Wang, Z. Third-order nanocircuit elements for neuromorphic engineering. *Nature* **585**, 518–523 (2020).
54. Schranghamer, T. F., Oberoi, A. & Das, S. Graphene memristive synapses for high precision neuromorphic computing. *Nat. Commun.* **11**, 5474 (2020).
55. Sebastian, A. et al. Two-dimensional materials-based probabilistic synapses and reconfigurable neurons for measuring inference uncertainty using Bayesian neural networks. *Nat. Commun.* **13**, 6139 (2022).
56. Wu, X., Dang, B., Wang, H., Wu, X. & Yang, Y. Spike-Enabled Audio Learning in Multilevel Synaptic Memristor Array-Based Spiking Neural Network. *Adv. Intell. Syst.* **4**, 2100151 (2022).
57. Kumar, S., Wang, X., Strachan, J. P., Yang, Y. & Lu, W. D. Dynamical memristors for higher-complexity neuromorphic computing. *Nat. Rev. Mater.* **7**, 575–591 (2022).
58. Nishioka, D., Tsuchiya, T., Higuchi, T. & Terabe, K. Enhanced synaptic characteristics of H_xWO_3-based neuromorphic devices, achieved by current pulse control, for artificial neural networks. *Neuromorph. Comput. Eng.* **3**, 034008 (2023).

Acknowledgements

This work was in part supported by Japan Society for the Promotion of Science (JSPS) KAKENHI Grant Number JP22H04625 (Grant-in-Aid for Scientific Research on Innovative Areas “Interface Ionics”), and JP22KJ2799 (Grant-in-Aid for JSPS Fellows). A part of this work was supported by JST PRESTO Grant number, JPMJPR23H4. A part of this work was supported by “Advanced Research Infrastructure for Materials and Nanotechnology in Japan (ARIM)” of the Ministry of Education, Culture, Sports, Science and Technology (MEXT). Proposal Number JPMXP1223NM5072. A part of this work was supported by the Iketani Science and Technology Foundation.

Author contributions

D.N., T.T., and K.T. conceived the idea for the study. D.N. and T.T. designed the experiments. D.N. and T.T. wrote the paper. D.N. carried out the experiments. D.N. prepared the samples. D.N. and T.T. analyzed the data. All authors discussed the results and commented on the manuscript. K.T. directed the projects.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s44172-024-00227-y.

Correspondence and requests for materials should be addressed to Takashi Tsuchiya.

Peer review information *Communications Engineering* thanks Flavio Abreu Araujo, Anatole Moureaux, Fabien Alibart, and the other, anonymous, reviewer(s) for their contribution to the peer review of this work. Primary Handling Editors: Damien Querlioz, Anastasiia Vasylchenkova, Rosamund Daw.