Title	Boosting Computational Power through Spatial Multiplexing in Quantum Reservoir Computing
Author(s)	Nakajima, Kohei; Fujii, Keisuke; Negoro, Makoto; Mitarai, Kosuke; Kitagawa, Masahiro
Citation	Physical Review Applied. 11(3) P.034021
Issue Date	2019-03-08
Text Version	publisher
URL	http://hdl.handle.net/11094/77643
DOI	10.1103/PhysRevApplied.11.034021
rights	Copyright (2019) by the American Physical Society
Note	

Boosting Computational Power through Spatial Multiplexing in Quantum Reservoir Computing

Kohei Nakajima,[1,2,*] Keisuke Fujii,[2,3,4] Makoto Negoro,[2,5,6] Kosuke Mitarai,[5] and
Masahiro Kitagawa[5,6]

[1]*Chair for Frontier AI Education, Graduate School of Information Science and Technology,
The University of Tokyo, Bunkyo-ku, 113-8656 Tokyo, Japan*

[2]*JST, PRESTO, 4-1-8 Honcho, Kawaguchi, Saitama 332-0012, Japan*

[3]*The Hakubi Center for Advanced Research, Kyoto University, Yoshida-Ushinomiya-cho,
Sakyo-ku, Kyoto 606-8302, Japan*

[4]*Department of Physics, Graduate School of Science, Kyoto University, Kitashirakawa Oiwake-cho, Sakyo-ku,
Kyoto 606-8502, Japan*

[5]*Graduate School of Engineering Science, Osaka University, 1-3 Machikaneyama, Toyonaka,
Osaka 560-8531, Japan*

[6]*Quantum Information and Quantum Biology Division, Institute for Open and Transdisciplinary Research
Initiatives, Osaka University, Osaka 560-8531, Japan*

Quantum reservoir computing provides a framework for exploiting the natural dynamics of quantum systems as a computational resource. It can implement real-time signal processing and solve temporal machine-learning problems in general, which requires memory and nonlinear mapping of the recent input stream using the quantum dynamics in the computational supremacy region, where the classical simulation of the system is intractable. A nuclear-magnetic-resonance spin-ensemble system is one of the realistic candidates for such physical implementations, which is currently available in laboratories. In this paper, considering these realistic experimental constraints for implementing the framework, we introduce a scheme, which we call a spatial multiplexing technique, to effectively boost the computational power of the platform. This technique exploits disjoint dynamics, which originate from multiple different quantum systems driven by common input streams in parallel. Accordingly, unlike designing a single large quantum system to increase the number of qubits for computational nodes, it is possible to prepare a huge number of qubits from multiple but small quantum systems, which are operationally easy to handle in laboratory experiments. We numerically demonstrate the effectiveness of the technique using several benchmark tasks and quantitatively investigate its specifications, range of validity, and limitations in detail.

I. INTRODUCTION

Recent developments in sensing and Internet-of-things technology follow big data, which consist of a massive amount of complex time series data. Accordingly, a novel information processing technique that can deal with these data efficiently in real time is eagerly required. Conventional computers are, however, based on the von Neumann architecture, where the processor and memory are separately aligned. This structure causes an intrinsic limitation in processing speed, which is called the von Neumann bottleneck. Furthermore, the schemes of the von Neumann–type models stipulate that to handle complex information processing, the computational

system should also be built in a complex manner systematically. While biological systems are complex systems that are constantly exposed to massive sensory data, they perform successful real-time information processing with lower computational costs and energy consumptions. Their way of information processing is a typical non–von Neumann type, capitalizing on its natural and diverse dynamics, and has been a source of inspiration for many researchers [1].

Reservoir computing is a framework for recurrent neural network training inspired by the way the brain processes information [2–5] and it provides a typical example of a non–von Neumann–type computation [6]. A reservoir computing system consists of a high-dimensional dynamical system, called a reservoir, driven by time-varying input streams, which generates transient dynamics with

*k_nakajima@mech.t.u-tokyo.ac.jp

034021-1

fading memory property and can perform nonlinear processing on inputs [7]. Its framework can be used for real-time information processing with complex temporal structures, which makes it particularly suited to machine-learning problems requiring memory, such as speech recognition, prediction of stock markets, and autonomous motor controls for robots. Conventionally, this scheme is implemented through randomly coupled artificial neural networks [i.e., echo state networks (ESN) [2]] or through spiking neural networks (i.e., liquid state machines [3]) in the software program running on a PC. As long as it runs on a conventional PC, the resulting computation is inevitably a von Neumann type. On this basis, the physical implementations of the reservoir have been proposed to exploit the dynamics of native physics for information processing. The implementations include the dynamics of the water surface [8], photonics [9,10], spin-tronics [11–13], and the nanomaterials structured in the neuromorphic chip [6]. Even the diverse body dynamics of soft robots have been shown to be used as a successful reservoir [14–17], suggesting that this framework could be applied to physical systems in various scales. Recently, quantum reservoir computing (QRC) has been proposed, which implements reservoir computing powered by quantum physics [18].

Quantum dynamics is difficult to simulate using a conventional or classical computer due to the exponentially large degrees of freedom. This property is generally termed a quantum computational supremacy [19] and the framework of QRC relies heavily on this property of quantum dynamics. Quantum reservoir (QR) dynamics are expressed as transitions of the basis states for quantum bits (qubits) driven by an input stream [Fig. 1(a)], which evolve over time through a unitary operator based on a Hamiltonian. Signals are obtained through projective measurements from the system, called true nodes, which are used as direct reservoir states. An exponential number of degrees of freedom exist behind the measurement called hidden nodes, which affect the time evolution of the true nodes. The framework of QRC naturally takes into account the exponential degrees of freedom of quantum dynamics, which is intractable for the classical computer, for information processing. Furthermore, the framework implements non–von Neumann–type computing through a reservoir computing scheme, suggesting the full exploitation of assets from physical quantum dynamics. It has been shown that the QR system can emulate nonlinear dynamical systems, including classical chaos, and exhibit robust information processing against noise [18]. As candidates for the physical experimental platform of the scheme, nuclear-magnetic-resonance (NMR) spin-ensemble systems [20,21] have been proposed. In these systems, nuclear spins in molecules are used as the ensemble qubit system. Usually, when monitoring a quantum system, its observables are affected by projective measurements, a process
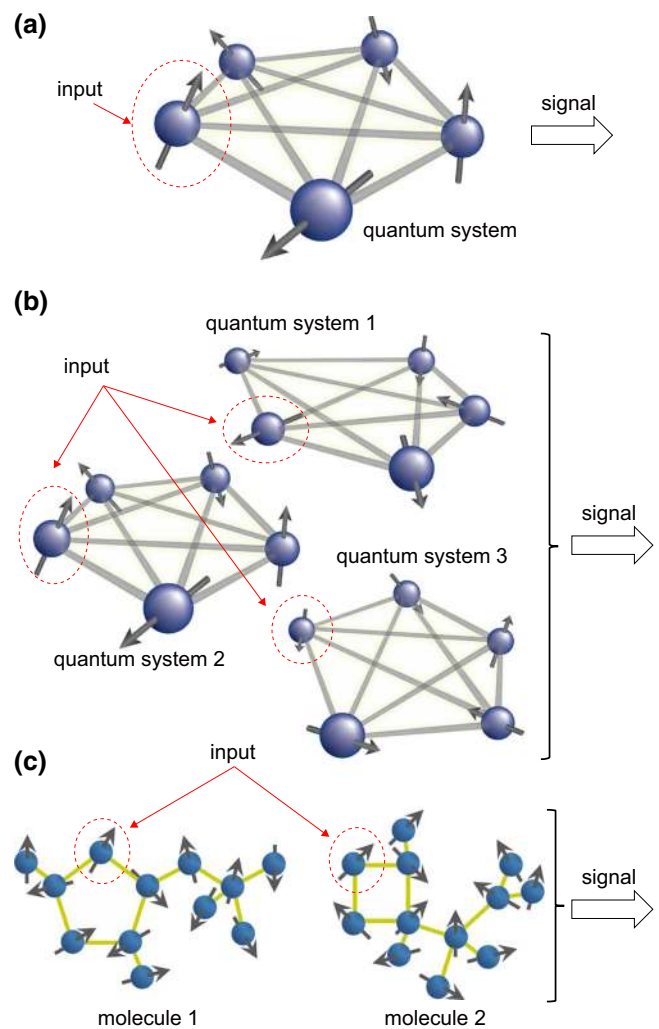

FIG. 1. Schematics explaining a spatial multiplexing technique for QRC. (a) A QRC system without spatial multiplexing for comparison, showing a quantum system with five qubits; the input is injected into the first qubit. (b) A QRC system with spatial multiplexing. It shows multiple disjoint quantum systems, each containing five qubits; the input is injected into the first qubit for each system. (c) Schematic of an experimental implementation of QRC with a spatial multiplexing technique for the NMR system.

called backaction. In the NMR ensemble system, this effect of backaction can be neglected, and the signal can be successfully obtained by averaging the massive amount of copies of molecules existing in the ensemble system. (We here note that the first physical implementation of NMR-based QR has recently been reported; this exploits the controllable dynamics of a nuclear spin ensemble in a molecular solid [22].)

In this paper, we present a scheme for boosting the computational power of QRs. (Throughout this study, the term "computational power" implies the system's expressive capability for function approximations, like that found

in neural networks, including reservoir computing systems.) The most prominent and straightforward approach for improving the computational capability of the system is increasing its number of computational nodes. In QRC, this primarily corresponds to increasing the number of qubits. However, when viewed from a physical implementation standpoint (e.g., using a NMR spin-ensemble system), this approach requires a redesign or reconstruction of the sample molecules, which is operationally difficult as well as energy and time consuming.

To overcome this problem, we introduce an effective approach to boost the computational power of the system using readily available small-sample molecules, which are operationally easy to handle in the experiments. Our scheme is called spatial multiplexing, in which we prepare multiple different small-sample molecules, inject common input streams into each system, and use all the signals obtained from these systems as a big single reservoir system [Fig. 1(b)]. This procedure has previously been proposed in the applications of conventional ESNs and many examples have demonstrated its effectiveness (e.g., [2]). Here, we apply the scheme to QRC and present that its procedure is particularly suited to overcome the difficulty in a physically implemented reservoir setting.

In a software-implemented RC, since the scheme of spatial multiplexing exploits multiple disjoint ESNs as a new reservoir, it is operationally equivalent to assuming a single ESN having the same total number of computational nodes in the first place with a specific sparse internal weight matrix. However, when viewing this scheme from physical RC perspectives, the situations are different. In the NMR-implemented QRC, for example, even if the number of computational nodes is the same, the operational cost of preparing one huge sample molecule and that of preparing multiple small-sample molecules are different. By focusing on this operational difference, we can secure the scheme as one of the realistic and practical options to improve the computational power of physical reservoirs, which are often difficult to design freely and easily. In the following sections, we argue the effectiveness of the spatial multiplexing technique for the NMR spin-ensemble-system-based QRC and quantitatively demonstrate how the scheme improves the computational performance in QRC. We also provide a detailed theoretical explanation of the specifications and range of validity of the scheme, which will be useful for evaluating other reservoir systems.

This paper is organized as follows. In the next section, we overview the formalization of QRC [18] and introduce spatial multiplexing into the setting. Subsequently, we theoretically examine the effect of spatial multiplexing in detail from a general standpoint. We then numerically demonstrate the power of the spatial multiplexing technique on QRC using conventional benchmark tasks in a machine-learning context. Several approaches

to engineer QRs through spatial multiplexing are also discussed. Finally, its practical aspects, future application domains in solving real-world machine-learning problems, and its implications for the reservoir computing framework in general are discussed.

II. QUANTUM RESERVOIR COMPUTING THROUGH SPATIAL MULTIPLEXING

A. Quantum reservoir dynamics

Let us consider a quantum state of an N-qubit system, which is described by a density operator ρ. By denoting the Pauli operators to be

$$I = \sigma_{00}, \quad X = \sigma_{10}, \quad Z = \sigma_{01}, \quad Y = \sigma_{11}, \quad (1)$$

an N-qubit Pauli product is defined by $2N$-bit string $\boldsymbol{i}$:

$$P(\boldsymbol{i}) = \bigotimes_{k=1}^{N} \sigma_{i_{2k-1}i_{2k}}. \quad (2)$$

By using the Pauli products $\{P(\boldsymbol{i})\}$ as a basis of the operator space, the quantum state ρ is represented by 4^N real vectors:

$$\rho \rightarrow \boldsymbol{r} = \begin{pmatrix} r_{00\cdots0} \\ \vdots \\ r_{11\cdots1} \end{pmatrix}, \quad (3)$$

where each element $r_{\boldsymbol{i}}$ is given in terms of the Schmidt-Hilbert inner product for the operator space as

$$r_{\boldsymbol{i}} = \text{Tr}[P(\boldsymbol{i})\rho]/2^N. \quad (4)$$

From the properties of the density operators,

$$r_{00\cdots0} = 1/2^N, \quad -1 \le r_{\boldsymbol{i}} \le 1, \quad \sum_{\boldsymbol{i}} r_{\boldsymbol{i}}^2 \le 1. \quad (5)$$

In QRC, each element $r_{\boldsymbol{i}}$ is regarded as a *hidden* node of the network. In quantum mechanics, any physical operation can be written as a linear transformation via a $4^N \times 4^N$ matrix W:

$$\boldsymbol{r}' = W\boldsymbol{r}. \quad (6)$$

The matrix W can be constructed explicitly from the quantum operation $\mathcal{W}$ as follows:

$$W_{\boldsymbol{j}\boldsymbol{i}} = \text{Tr}\{P(\boldsymbol{j})\mathcal{W}[P(\boldsymbol{i})]\}/2^N. \quad (7)$$

For example, a unitary dynamics under the Hamiltonian H with a time interval τ is given by

$$(U_\tau)_{\boldsymbol{j}\boldsymbol{i}} = \text{Tr}[P(\boldsymbol{j})e^{-iH\tau}P(\boldsymbol{i})e^{iH\tau}]/2^N. \quad (8)$$

In order to exploit quantum dynamics for information processing, we have to introduce an input and the signals

of the quantum system [see Fig. 1(a)]. Suppose $\{u_k\}$ is an input sequence, which is a continuous variable ($u_k \in [0, 1]$). (We consider the setting of one-dimensional input for simplicity, but its generalization to a multidimensional case is straightforward.) A temporal learning task here is to find, using the quantum system, a nonlinear function $y_k = f(\{u_l\}_{l=1}^k)$ such that the mean-square error between y_k and a target output $\hat{y}_k$ for a given task becomes minimum. Note that, as we see from Eq. (6), there is no nonlinearity in each quantum operation W. Instead, the time evolution W can be changed according to the external input u_k, namely, W_{u_k}, allowing the quantum reservoir to process the input information $\{u_k\}$ nonlinearly, by repetitively feeding the input. Thus, we can obtain nonlinear terms, for example, with respect to u_k and u_{k-1}, $W_{u_k} W_{u_{k-1}}$ via a multiplication of two linear transformations.

Specifically, as an input in our work, we replace the first qubit to the quantum state [Fig. 1(a)]

$$\rho_{u_k} = \frac{I + (1 - 2u_k)Z}{2}. \tag{9}$$

The corresponding matrix S_{u_k} is given by

$$(S_{u_k})_{ji} = \mathrm{Tr}\left\{P(j)\frac{I + (1 - 2u_k)Z}{2} \otimes \mathrm{Tr}_{1st}[P(i)]\right\}/2^N, \tag{}$$

where Tr_{1st} indicates a partial trace with respect to the first qubit. A unit time step is written as an input-depending linear transformation:

$$r[(k + 1)\tau] = U_\tau S_{u_k} r(k\tau), \tag{10}$$

where $r(k\tau)$ indicates the hidden nodes at time $k\tau$.

A set of observed nodes, which we call *true* nodes, $\{x_l\}_{l=1}^M$ is defined by a $4^N \times M$ matrix R,

$$x_l(k\tau) = \sum_i R_{li} r_i(k\tau). \tag{11}$$

The number of true nodes M has to be a polynomial in the number of qubits N. That is, from exponentially many hidden nodes, a polynomial number of true nodes are obtained. For simplicity, we take the single-qubit Pauli Z operator on each qubit as the true nodes, i.e.,

$$x_1 = r_{010\cdots0}, \quad x_2 = r_{00010\cdots0}, \dots, \quad x_n = x_{0\cdots01}. \tag{12}$$

Therefore, there are $M = N$ true nodes. Figure 2(a) shows the typical reservoir dynamics driven by the input stream $\{u_k\}$ and they consist of signals obtained from the true nodes. Here, we assume that the system is an ensemble quantum system, which consists of a huge number of copies of single quantum systems. Therefore, the signals from the true nodes are obtained without any backaction.
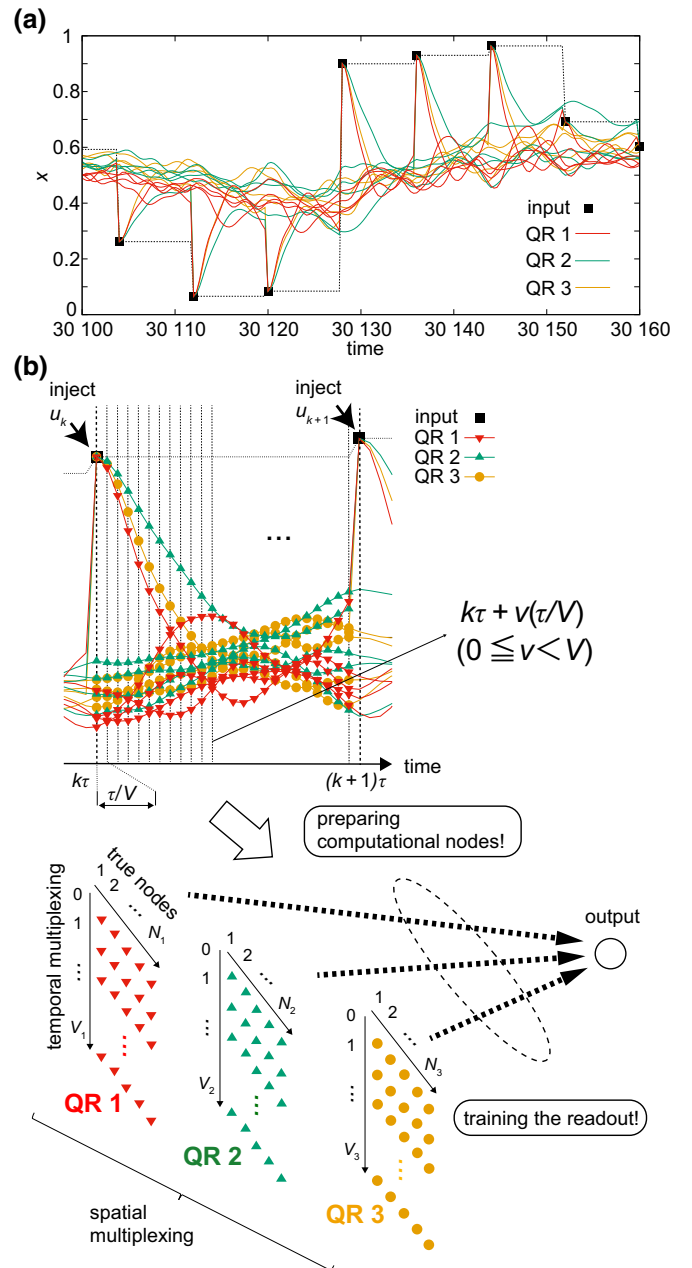

FIG. 2. Preparing computational nodes through spatial multiplexing. (a) A typical example of reservoir dynamics (a time series of signals obtained from true nodes) in multiple disjoint quantum systems driven by a common input stream. The plot overlays reservoir dynamics from three different quantum systems with the number of qubits set to 5 and $\tau\Delta = 8$. (b) The temporal and spatial multiplexing scheme. The upper diagram focuses on the time interval when the input u_k is injected. Signals from the three QR systems are overlaid, where the parameters τ_c, V_c, and N_c are set to be the same among the systems for simplicity. The lower diagram expresses how to prepare the computational nodes in our settings. The linear and static readout weight is attached to each computational node and the learning is performed by training the weights. See text for details.

Actually, the NMR spin-ensemble system is such a system. A sample of a NMR spin-ensemble system contains typically 10^{18-20} copies of the same molecules. The magnetization of 10^{14-16} spins out of the sample can be measured with a rf coil with a sufficient signal-to-noise (SN) ratio, while the remaining is not affected.

The unique feature of QRC in the reservoir computing context is that the exponentially many hidden nodes that originate from the dimensions of the Hilbert space are monitored from a polynomial number of signals defined as the true nodes. Based on this setting, in the next section, two coordinated schemes are introduced to harness QR dynamics in a physically natural setting. The first is called "temporal multiplexing," which was already introduced in Refs. [18,23], and the second is called "spatial multiplexing," which is a procedure applied to the QRC from this study.

B. Temporal multiplexing

In Ref. [18], temporal multiplexing has been found to be useful to extract complex dynamics on the exponentially large hidden nodes through the restricted number of true nodes. In temporal multiplexing, the signals are sampled from the QR not only at the time $k\tau$, but also at each of the subdivided V time intervals during the unitary evolution U_τ to construct V virtual nodes, as shown in Fig. 2(b) (the upper diagram). After each input by S_{u_k}, the signals are obtained for each subdivided interval after the time evolution by $U_{v\tau/V}$ ($v = 1, 2, \ldots V$), i.e.,

$$\boldsymbol{r}[k\tau + (v/V)\tau] \equiv U_{(v/V)\tau} S_{u_k} \boldsymbol{r}(k\tau). \quad (13)$$

Accordingly, as the QR system has N true nodes, we have NV corresponding computational nodes at each input timestep k in total, and the virtual nodes are defined by

$$x_l[k\tau + (v/V)\tau] = \sum_i R_{li} r_i[k\tau + (v/V)\tau]. \quad (14)$$

This procedure allows us to make full use of input-driven transient dynamics, which can potentially include the influence of hidden nodes. Using this technique, it is possible to effectively increase the total number of computational nodes employed in the learning process. A similar technique can also be found, for example, in Ref. [9] under the same motivations.

It is important to note that, as is obvious from the setting, the parameter τ modulates directly the dynamics of QR, while the parameter V defines how we observe the dynamics. In Ref. [18], the relevance of these parameters to the computational capability of the QR system was investigated. It was observed that, according to the choice of the parameter τ, the type of computation that can be performed well has changed and the increase in the parameter V essentially contributes to an improved computational performance.

C. Spatial multiplexing

Now, we consider boosting the computational power in QRC further. The most straightforward and promising approach that comes to mind is increasing the number of computational nodes. This naturally leads to an increase in the number of qubits in the QR system. (The approach of temporal multiplexing, which secures virtual nodes from the signals, is also reasonable in terms of increasing the number of computational nodes.) Considering the physical implementations of QRC to the NMR system, however, as we explained previously, this procedure of increasing the number of qubits corresponds to the enlargement and redesign of sample molecules, and it is not always easy in practice. In the NMR system, the local control and measurement of a qubit is accomplished with the difference of the resonant frequency. The resonant frequency differs from the nuclear species. Among many species, only a few species such as ^{1}H, ^{13}C, ^{15}N, and ^{19}F are easy to handle and thus used as qubits before. The resonant frequency is also slightly shifted due to the difference of the chemical environment even with the same species, which enables us to have local control of them. However, it is not easy to design and synthesize a molecule that includes many addressable spins with the different species and environment. Since a 12-addressable-spin system in a liquid has been developed in 2006 [24], the record still remains unbroken.

In this study, based on these physical constraints of the experimental settings, we introduce an effective and practical procedure to increase the computational resource, which is relatively easy to implement under the practical condition. The procedure is called spatial multiplexing. We prepare multiple disjoint QR systems, which are spatially distant or uncoupled, and we drive them with a common input stream in parallel [Fig. 1(b)]. Subsequently, we collect the signals from each QR system in the previously explained manner and we use all of these signals from different QR systems as one entire set of reservoir dynamics. (Note that, because of the experimental constraints, it would be difficult to inject inputs in a perfectly synchronized manner into each separate system. Even in these cases, if I/O timesteps could be coordinated among each system, then the spatial multiplexing technique could be applied.) For the NMR-implemented QRC, this approach enables the exploitation of readily available sample molecules, which already exist in the laboratory, to increase the number of computational nodes. Compared to redesigning the sample molecules as a computational resource, this approach should be relatively handy and practical for experimenters. For example, the aforementioned 12-qubit molecule and another one developed in 2017 [25] can be potentially utilized for spatial multiplexed reservoirs [Fig. 1(c)]. To synthesize other 12-qubit molecules based on the developed molecules with

chemical modifications may be easier than a molecule with more qubits. It is important to note that this procedure of spatial multiplexing combines true nodes from disjoint quantum systems and this does not generate an exponential number of hidden nodes against the number of true nodes. This implies that, even if the number of true nodes is the same, the expressive power of QR with spatial multiplexing is different from that of QR prepared from a single quantum system in principle.

Let us consider C different QRs driven by a common input stream u_k in parallel. For each QR system c, which has N_c qubits with a corresponding number of true nodes, the time interval to inject input τ_c and the corresponding unitary evolution U_{τ_c} can be set differently [Fig. 2(b)]. Each QR system is also equipped with temporal multiplexing having V_c virtual nodes [Fig. 2(b)]. As a result, the spatial multiplexing induces $\sum_{c=1}^{C} N_c V_c$ nodes in total and these computational nodes are exploited as a single reservoir. We investigate systematically whether the procedure of spatial multiplexing really boosts the computational power of the QRC or not and, furthermore, to what extent it improves the performance in detail in the later sections.

D. Output settings and learning procedure

In the reservoir computing approach, the output is obtained as a weighted sum of the reservoir states and the learning of a target function is executed by training linear and static readout weights attached to the reservoir nodes in a supervised manner. Here, we explain how to train the readout weights from the observed signals of QR after the procedures of temporal and spatial multiplexing. According to the previous sections, temporal and spatial multiplexing introduces $N_{\text{total}} = \sum_{c=1}^{C} N_c V_c$ computational nodes in total [Fig. 2(b)]. The state of the computational node i at timestep k is expressed as x'_{ki} by rearranging the subscript from the original and we introduce a constant bias term $x'_{k0} = 1.0$. The system output of the system is then expressed as

$$y_k = \sum_{i=0}^{N_{\text{total}}} x'_{ki} w_i, \tag{15}$$

where w_i is a linear and static weight attached to node i. Let $\{\hat{y}_k\}_{k=1}^{L}$ be the target sequence for learning, where L is the length of the training phase that is assumed much greater than $N_{\text{total}} + 1$ and the training of the readout weights $\{w_i\}_{i=0}^{N_{\text{total}}}$ is to minimize $\sum_{k=1}^{L} (y_k - \hat{y}_k)^2$. By collecting the target output $\hat{y} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_L]^T$ and the corresponding $N_{\text{total}} + 1$ reservoir states in the learning phase as the training data matrix X, which is an $L \times (N_{\text{total}} + 1)$ matrix, the optimal weight $\hat{w} = [\hat{w}_0, \hat{w}_1, \ldots, \hat{w}_{N_{\text{total}}}]^T$ can be obtained as a least-squares solution $\hat{w} = (X^T X)^{-1} X^T \hat{y}$.

As we see later in detail, when we actually let the QR system perform the computational tasks in this study, the experimental trial consists of a washout phase, training phase, and evaluation phase. The washout phase is to eliminate the influence of initial transients of the reservoir states, and the trained readout weights in the training phase are exploited to generate outputs in the evaluation phase.

E. Theoretical insights into the effect of spatial multiplexing

In this section, we investigate theoretically the effect of spatial multiplexing and we show its range of validity and limitations. The argument in this section is not limited to a quantum system but is generally applicable to any reservoir system. Initially, we prove concisely that the procedure of spatial multiplexing always improves computational performance (or, at worst, will not change the performance).

Let us assume that we have two reservoirs, reservoirs A and B, which have N_A and N_B computational nodes, respectively. Consider the corresponding regression equations, $y = X_A w_A + r_A$ and $y = X_B w_B + r_B$, where X_A is a $T \times N_A$ matrix and X_B is a $T \times N_B$ matrix, with realizations T satisfying $N_A + N_B \leq T$, and r_A and r_B are residuals. We assume that X_A and X_B are full rank, and w_A and w_B are least-squares solutions expressed as $w_A = (X_A^T X_A)^{-1} X_A^T y$ and $w_B = (X_B^T X_B)^{-1} X_B^T y$, respectively. With projectors $P_A = X_A (X_A^T X_A)^{-1} X_A^T$ and $P_B = X_B (X_B^T X_B)^{-1} X_B^T$, $X_A w_A = P_A y$ and $X_B w_B = P_B y$. Accordingly, the residuals can be expressed as $r_A^2 = \|y - X_A w_A\|^2 = \|(I - P_A)y\|^2$ and $r_B^2 = \|y - X_B w_B\|^2 = \|(I - P_B)y\|^2$. We consider combining reservoirs A and B and constructing a new reservoir "$A + B$." Similarly, for $X_{A+B} = (X_A \ X_B)$, $y = X_{A+B} w_{A+B} + r_{A+B}$, where w_{A+B} is a least-squares solution expressed as $w_{A+B} = (X_{A+B}^T X_{A+B})^{-1} X_{A+B}^T y$ and r_{A+B} is a residual. We assume that X_{A+B} is full rank. With projectors $P_{A+B} = X_{A+B}(X_{A+B}^T X_{A+B})^{-1} X_{A+B}^T$, $X_{A+B} w_{A+B} = P_{A+B} y$, and a residual can be expressed as $r_{A+B}^2 = \|y - X_{A+B} w_{A+B}\|^2 = \|(I - P_{A+B})y\|^2$. Because w_{A+B} is a least-squares solution,

$$
\begin{aligned}
r_{A+B}^2 &= \|(I - P_{A+B})y\|^2 \\
&= \|y - X_{A+B} w_{A+B}\|^2 \\
&\leq \left\| y - X_{A+B} \begin{pmatrix} w_A \\ 0 \end{pmatrix} \right\|^2 \\
&= \left\| y - (X_A \ X_B) \begin{pmatrix} w_A \\ 0 \end{pmatrix} \right\|^2 \\
&= \|y - X_A w_A\|^2 \\
&= \|(I - P_A)y\|^2 = r_A^2.
\end{aligned}
$$

The equal sign can be used only when $P_{A+B} = P_A$. Likewise, $r_{A+B}^2 \leq r_B^2$ and $r_{A+B}^2 \leq \min\{r_A^2, r_B^2\}$ holds. This relation clearly provides the mechanism of the boost of

the computational power or improvement of the performance in spatial multiplexing, specifically implying that the improvement occurs against the performance of the original reservoir (reservoir A or B) in terms of the learning errors, the residuals. Furthermore, this relation shows the reason why the performance improves by increasing the computational nodes in the system in general. It also suggests that the couplings and interactions within the reservoir are not explicitly required for this improvement in theory. Conversely, this relation still holds even if there are some couplings between reservoirs A and B (although it may be referred to as one reservoir rather than two in this case).

Estimating the upper limit of the improvement in performance in terms of how the residual decreases by adding the reservoir B to the reservoir A is also possible. In general, projector P satisfies $P = P^{\mathrm{T}}$, $PP = P$, and if P is a projector, then $I - P$ is also a projector. Applying these properties to the above results, with a few transformations, we obtain

$$0 \le r_A^2 - r_{A+B}^2 = \langle (P_{A+B} - P_A)y, y \rangle,$$

where $\langle \cdot, \cdot \rangle$ is an inner product. This relation suggests that $Q_{A,A+B} := P_{A+B} - P_A$ is positive semidefinite, and the largest eigenvalue is

$$\lambda_{Q_{A,A+B}} := \sup_{y \ne 0} \frac{\langle (P_{A+B} - P_A)y, y \rangle}{\langle y, y \rangle}.$$

Thus,

$$0 \le r_A^2 - r_{A+B}^2 \le \lambda_{Q_{A,A+B}} \|y\|^2,$$

where $\lambda_{Q_{A,A+B}}$ expresses the supremum of the reductions of a normalized residual when adding reservoir B to reservoir A. Similarly, we obtain $0 \le r_B^2 - r_{A+B}^2 \le \lambda_{Q_{B,A+B}} \|y\|^2$, where $Q_{B,A+B} := P_{A+B} - P_B$ is also positive semidefinite and $\lambda_{Q_{B,A+B}}$ is the largest eigenvalue of $Q_{B,A+B}$. Using the above relations for $r_B^2 - r_{A+B}^2$ and $r_A^2 - r_{A+B}^2$, and $r_{A+B}^2 \le \min\{r_A^2, r_B^2\}$, we obtain

$$\max\{(r_A^2 - \lambda_{Q_{A,A+B}} \|y\|^2), (r_B^2 - \lambda_{Q_{B,A+B}} \|y\|^2)\}$$
$$\le r_{A+B}^2 \le \min\{r_A^2, r_B^2\},$$

where $0 \le \max\{(r_A^2 - \lambda_{Q_{A,A+B}} \|y\|^2), (r_B^2 - \lambda_{Q_{B,A+B}} \|y\|^2)\}$. Thus, we can evaluate and predict the extent of the improvement without actually performing the task using reservoir $A + B$.

We should be careful because the above facts do not always hold in practice. Two points should be noted. The first is overfitting. Spatial multiplexing can increase computational nodes drastically, so we should be careful when balancing between the size of the training data set and the system size. Since spatial multiplexing always results

in an improved performance for the training data set, if the performance worsens with spatial multiplexing in the evaluation phase, we can infer back that it is caused by overfitting.

Second, the prior facts are based on the assumption that X_A, X_B, and X_{A+B} are full rank. This condition does not always hold in practice. A typical example is a case in which synchronization occurs, which makes the reservoir dynamics identical or low dimensional. Notably, even if no coupling exists between the reservoirs in spatial multiplexing, the synchronization can still occur. This phenomenon is often called generalized synchronization [26] or common input (noise) synchronization [27,28]. Ironically, as investigated in Ref. [27], this property of common input synchronization is rather a required property for reservoirs in terms of the reproducibility of the signals (the opposite case is chaotic dynamics, where an arbitrarily small change in one state of a deterministic system results in large differences in a later state). For robust information processing, the same reservoir is preferred to respond the same according to the identical input stream, even if the initial states of the reservoir differ. For the scheme of spatial multiplexing, however, this property acts as a drawback that avoids the duplication of the same reservoir in use. Accordingly, for spatial multiplexing, preparing a different reservoir or the same setting of the reservoir with different input scaling or with a different choice of qubit for input injections is recommended.

III. PERFORMANCE ANALYSES

In this section, we use numerical experiments to investigate the effect of spatial multiplexing. By assessing the memory capacity and by using a benchmark task that evaluates the information processing capability to emulate nonlinear dynamical systems called "nonlinear autoregressive moving average" (NARMA) systems, we demonstrate how the order of spatial multiplexing affects the performance of our QR system systematically. These evaluation schemes adopted here are popular in the context of recurrent neural network learning.

For the dynamics of QR system, we employ the simplest quantum system, a fully connected transverse-field Ising model, as an example:

$$H = \sum_{ij} J_{ij} X_i X_j + h Z_i, \tag{16}$$

where the coupling strengths are randomly assigned such that J_{ij} is distributed randomly from $-J/2$ to $J/2$. Furthermore, a scale factor Δ is introduced to make $\tau\Delta$ and J/Δ dimensionless. In our numerical experiments, the quantum dynamics of the above Hamiltonian is exactly calculated without employing any approximation.

Here, the spatial multiplexing is implemented using QR systems having the same number of qubits N_c, the input

interval τ_c, and the virtual nodes V_c (which we simply denote N, τ, and V, from now on) but with different random coupling strengths of J_{ij}. In the following, we see the case when the number of qubits N of a single QR system, which implies the case without spatial multiplexing, is set to 5. As an example, we demonstrate in detail when the parameter $\tau\Delta$ is set to 1 and 2 for the memory capacity analyses and for the NARMA tasks, respectively. We varied the number of the virtual nodes V as 1, 5, and 25 and checked the dependence on the performance. (Note that the analyses for the different parameter settings, such as the cases for $N = 3, 4$ and $\tau\Delta = 0.5, 1, 2, 3, 4, 8, 16$, and 32, are given in the Appendix.) Throughout the following experiments, the input stream is randomly drawn from the range $[0, 1]$ and injected to the first qubit of each QR system. The order of spatial multiplexing, which is defined as the number of QR systems driven by a common input stream in parallel, is varied from 1 (without spatial multiplexing) to 5 for the analyses.

A. Memory capacity

As discussed earlier, the information processing capability of reservoir dynamics can be characterized by its property of transforming the input stream. In particular, one of the important characteristics for the computational systems in solving a temporal machine-learning task is short-term memory, which is a property to store information of recent inputs to the system's current states. Focusing on this point, a measure to evaluate the system's short-term memory property, which is called memory capacity [29], is commonly used. In this section, we aim to analyze the memory capacity of the QR system and to quantify the effect of the spatial multiplexing in terms of it. To calculate the measure, the computational system should first learn to reproduce the injected random input of d timesteps before by using the current states of the system. This process is equivalent to setting the target output as $\hat{y}_k = u_{k-d}$, where u_{k-d} is set as a random sequence ranged in $[0, 1]$ in this study.

To evaluate the system's emulatability of the target sequence, the memory function MF_d is defined as

$$\text{MF}_d = \frac{\text{cov}^2(y_k, \hat{y}_k)}{\sigma^2(y_k)\sigma^2(\hat{y}_k)}, \quad (17)$$

where $\text{cov}(x, y)$ and $\sigma(x)$ express the covariance between x and y and the standard deviation of x, respectively. This measure can take the value from 0 to 1, and as the value gets larger, it suggests that the system's capability to reconstruct the previous input u_{k-d} gets higher. The memory capacity (MC) is defined as follows:

$$\text{MC} = \sum_{d=0}^{150} \text{MF}_d. \quad (18)$$

As explained in the earlier section, the training scheme of our QR system is based on supervised learning and, for each setting of d, the experimental trial consists of a washout phase (2000 timesteps), a training phase (2000 timesteps), and an evaluation phase (2000 timesteps). Using the time series data of 2000 timesteps in the training phase and the linear regression explained in Secs. C D, we optimize the readout weights, which we use to calculate the corresponding system output in the evaluation phase. For each order of spatial multiplexing, we iterate the above procedure by using new QR systems with different random coupling strengths for 100 trials and obtain the averaged MF_d and MC.

Figure 3(a) shows the averaged MF_d over the input delay d. By observing the behavior of MF_d against delay d, we can see that, according to the increase of the order of spatial multiplexing, the performance gradually improves, showing the relatively large value of MF_d in the region of the larger delay. This tendency can be captured more clearly in the behavior of MC [Fig. 3(b)]. Figure 3(b) plots how the order of spatial multiplexing affects the memory capacity of the QR system in each setting of virtual nodes.
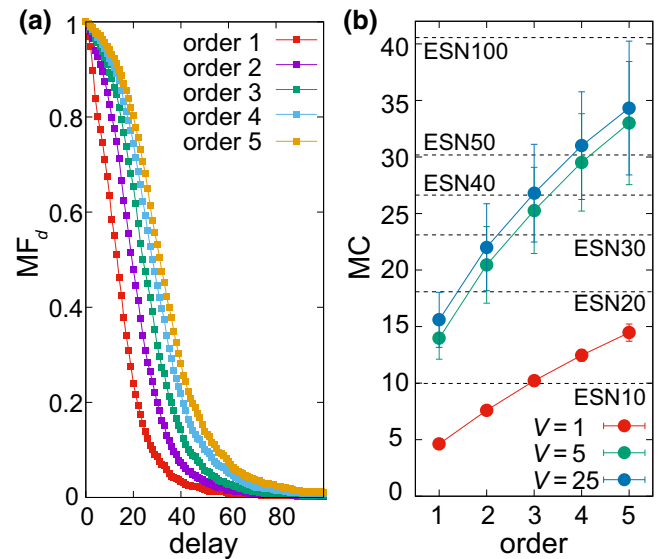

FIG. 3. Dependence of the order of spatial multiplexing on memory functions according to the delay and on the memory capacity. (a) The averaged memory functions MF_d are shown with the order of spatial multiplexing varied from 1 to 5. Each plot is calculated using 100 trials of different runs with different QR systems, where the virtual nodes are set to 25. (b) The averaged MCs are shown according to the order of spatial multiplexing. Each plot is calculated using 100 trials with different QR systems, and the cases with virtual nodes set to 1, 5, and 25 are overlaid. As a reference, each plot contains the performance of the conventional ESN. The notation "ESN20," for example, represents the averaged MC of the ESN with 20 nodes. For all the plots, the error bars show the standard deviations. A single QR system has five qubits, and the parameter $\tau\Delta$ is fixed to 1 throughout this analysis.

We can observe that, in all cases, the increase of the order of spatial multiplexing induces the improvement of memory capacity. Even in other parameter regions (e.g., for different settings of the parameter $\tau\Delta$ and the number of qubits) of the system, the improved memory capacity, according to the increased order of spatial multiplexing, is generally observed (see Fig. 8 in Appendix A for details). Interestingly, according to the setting of the parameter $\tau\Delta$, the amount of memory capacity that can be induced is different (Fig. 8 in Appendix A). That is, the memory capacity reaches relatively larger values when $\tau\Delta = 0.5$ and 1 than for the other settings of $\tau\Delta$.

B. NARMA tasks

The NARMA task is a commonly used benchmark task for evaluating the computational capability of the learning system to implement nonlinear processing with long time dependence. By calculating the deviations from the target trajectory in terms of errors, the NARMA task tests how well the target NARMA systems can be emulated by the learning system. According to the choice of the target NARMA system, it is possible to investigate which type of information processing can be performed in the learning system to be evaluated. The first NARMA system that we introduce is a second-order nonlinear dynamical system, which was used in Ref. [30], expressed as follows:

$$y_k = 0.4y_{k-1} + 0.4y_{k-1}y_{k-2} + 0.6u_k^3 + 0.1. \quad (19)$$

We call this system NARMA2 in this paper. The next NARMA system is the nth-order nonlinear dynamical system, which is written as follows:

$$y_k = \alpha y_{k-1} + \beta y_{k-1}\left(\sum_{j=0}^{n-1} y_{k-j-1}\right) + \gamma u_{k-n+1}u_k + \delta, \quad (20)$$

where α, β, γ, and δ are $0.3, 0.05, 1.5$, and 0.1, respectively. Here, n varies as $5, 10, 15$, and 20, and the corresponding systems are called NARMA5, NARMA10, NARMA15, and NARMA20, respectively. In particular, NARMA10 is frequently used in the context of evaluating the learning capability of recurrent neural networks (e.g., [4,30]). Here, we adopt the multitasking scheme, where the system should simultaneously emulate all the NARMA systems according to the input stream. For the input stream to the NARMA systems, the range is linearly scaled from $[0, 1]$ to $[0, 0.2]$ to set the range of y_k into the stable range.

The learning scheme of our QR system is exactly the same as explained in the previous MC analysis. Each experimental trial consists of a washout phase (2,000 timesteps), a training phase (2,000 timesteps), and an evaluation phase (2,000 timesteps). We evaluate the performance by comparing the system output with the target output, which is the normalized mean-squared error (NMSE), expressed as follows:

$$\text{NMSE} = \frac{\sum_{k=4001}^{6000}(\hat{y}_k - y_k)^2}{\sum_{k=4001}^{6000}\hat{y}_k^2}, \quad (21)$$

where $\hat{y}_k$ and y_k are the target output and the system output at timestep k, respectively. For each τ setting, NMSEs for all the trials are calculated and averaged for the analysis. For each order of spatial multiplexing, we iterated the above procedure by using new QR systems with different
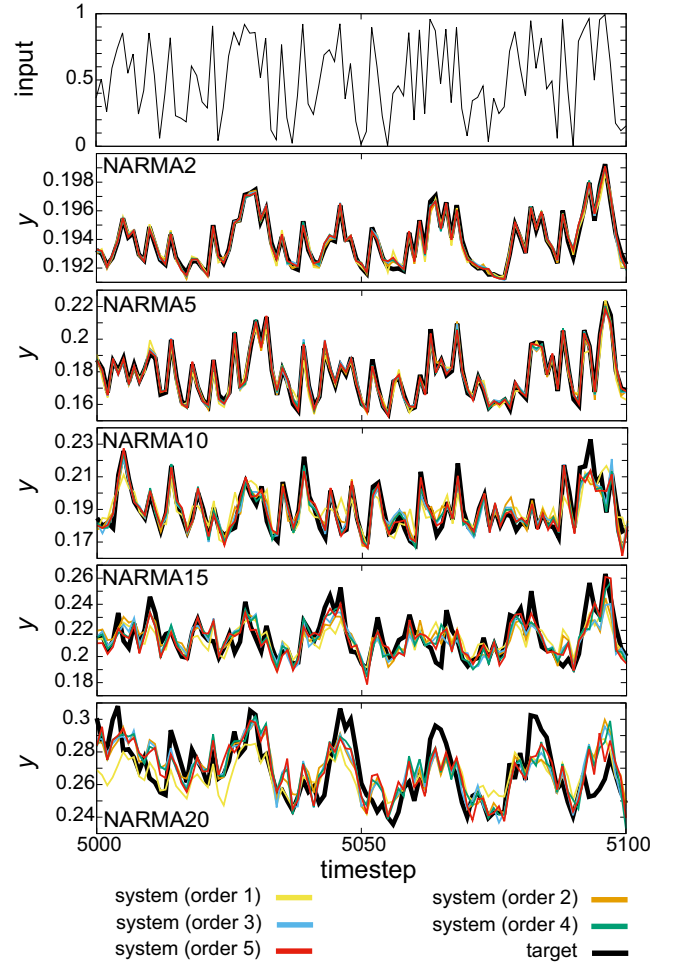

FIG. 4.   Typical system output time series for the NARMA tasks during the evaluation phase according to the order of spatial multiplexing. The uppermost plot shows the random input sequence, and the lower plots show the corresponding task performances for NARMA2, 5, 10, 15, and 20 in order from top to bottom. Each plot overlays the time series of the target output and system outputs, which exhibit multiplexing until five quantum systems with the number of qubits set to 5, the number of virtual nodes set to 25, and the parameter $\tau\Delta$ set to 2.

random coupling strengths for 100 trials and obtained the averaged NMSE.

Figure 4 shows the typical output time series for the NARMA tasks in the evaluation phase. First, it is clearly observed that according to the increase in the order of the NARMA system, the overall task performance gradually worsens, reflecting an increase of the difficulty of the tasks. For each NARMA task, according to the increase of the order of spatial multiplexing, we can see that the traceability of the QR system is improved (we can visually confirm this especially for the NARMA5, NARMA10, and NARMA15 tasks in Fig. 4). These observations can be quantitatively confirmed in the analyses of the averaged NMSE in Fig. 5. For each setting of the number of virtual nodes ($V = 1, 5$, and 25), the figure plots how the averaged NMSE behaves according to the increase of the order of spatial multiplexing in each NARMA task. Figure 5 shows that for all the NARMA tasks, the increase of the order of spatial multiplexing induces improvements in the task performance. In particular, when the order of the NARMA system is 2, 5, and 10, the effect of the increase of the order of spatial multiplexing is significantly high. We have checked that this tendency of the effect generally holds for other parameter settings of the QR system (see Fig. 9 in Appendix A for details). Furthermore, we have found that, for each NARMA task, a different setting of $\tau \Delta$ exists that shows the best performance through spatial multiplexing. For example, in the case for the NARMA2 task and NARMA5 task, the averaged NMSE shows the minimum value when $\tau \Delta = 32$, while in the case for the NARMA15 task and NARMA20 task, $\tau \Delta = 1$ shows the minimum,

both through spatial multiplexing of order 5 (Fig. 9 in Appendix A). These findings imply that the parameter $\tau \Delta$ can regulate which type of task the QR system is good at.

C. Temporal versus spatial multiplexing

Sections III A and III B demonstrate that, as the order of spatial multiplexing increases, the memory capacities increase and the performance of the NARMA tasks improves. In this section, we analyze the extent to which the order of spatial multiplexing plays a part in these improvements quantitatively.

Figure 6 plots how the improvement ratio behaves according to the increase of spatial multiplexing in each experimental case. The improvement ratio is defined by setting the performance (in terms of the averaged NMSE or MCs) when the order of spatial multiplexing is set to 1 as a basis. For both analyses, it is calculated by dividing each averaged MC and NMSE by those when the order of the spatial multiplexing is 1 in each parameter setting, respectively. As a comparison, the improvement ratios when the number of virtual nodes is increased from 1 to 5, from 1 to 25, and from 5 to 25, without spatial multiplexing (reflecting the effect of temporal multiplexing only), are shown in each plot of Fig. 6. We can clearly observe that, in almost all cases, the increase of the order of spatial multiplexing induces the improvements of the performance.

For the memory capacity, we can observe remarkable improvements, where the improvement ratio marks more than twice in all of the setting of virtual nodes, when increasing the order of the spatial multiplexing from 1 to 5
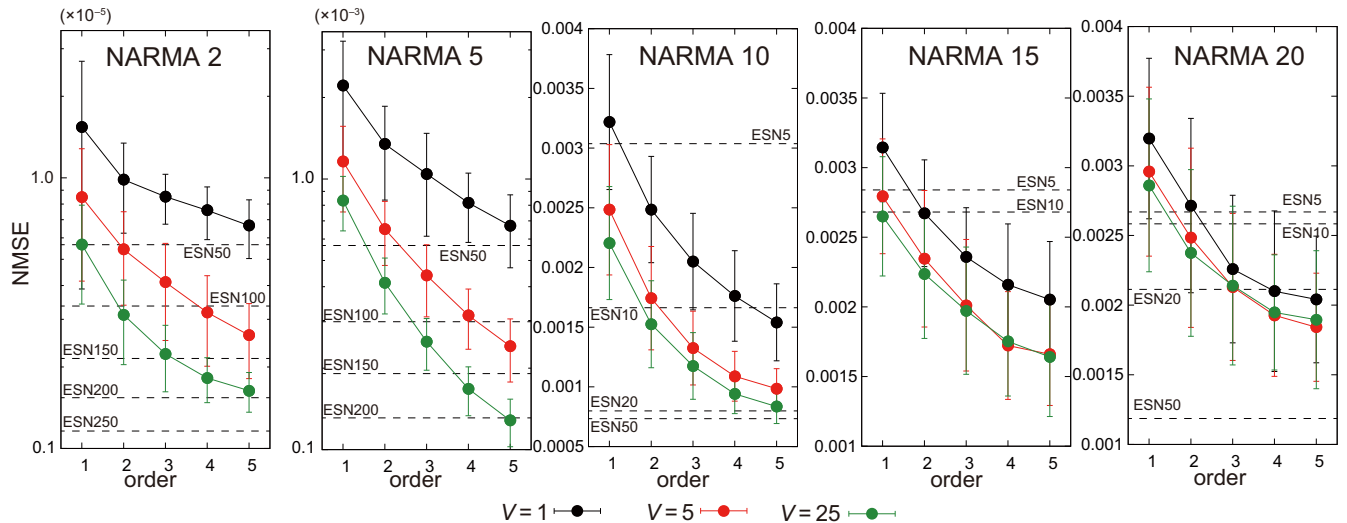

FIG. 5. Analysis of the averaged NMSE for the NARMA tasks according to the order of spatial multiplexing. For each plot, the cases with virtual nodes set to 1, 5, and 25 are overlaid. In all the plots, a single QR system has 5 qubits, and the parameter $\tau \Delta$ is fixed to 2. The averaged NMSE is calculated using 100 trials with different QR systems. Note that the y axis for the plots of NARMA2 and NARMA5 tasks are in the logarithm scale. As a reference, each plot contains the performance of the conventional ESN. The notation "ESN20," for example, represents the averaged NMSE of the ESN with 20 nodes. For all the plots, the error bars show the standard deviations. See text for details on the experimental conditions.
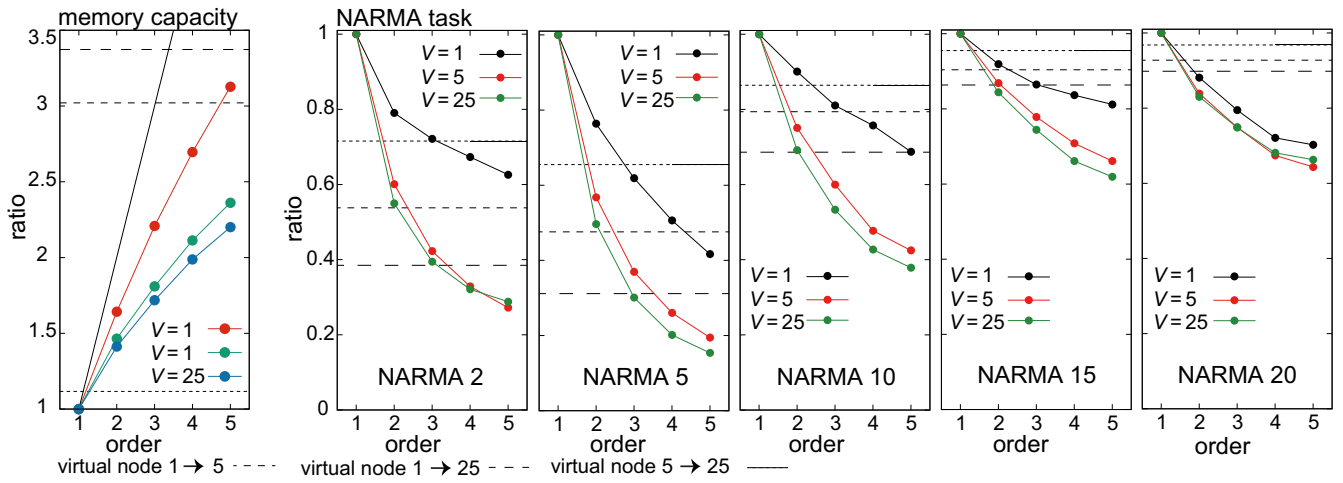
FIG. 6.   Analyses of the effect of the spatial multiplexing based on the improvement ratio and comparisons with that of the temporal multiplexing. Improvement ratios according to the order of spatial multiplexing in terms of the averaged memory capacity (left) and the averaged NMSE for the NARMA tasks (right) are investigated (the parameter settings are the same as those of the analyses in Secs. III A and III B). In the plot for the memory capacity, the solid line expresses $y = x$ as a reference. In each plot, the improvement ratios for the temporal multiplexing when the number of virtual nodes is increased from 1 to 5, from 1 to 25, and from 5 to 25 (without spatial multiplexing) are overlaid as a comparison. For all the analyses, the averaged MC and NMSE used to calculate the improvement ratio in each condition are obtained from the results of 100 trials with different QR systems.

(Fig. 6, left diagram). In particular, the effect of increasing the order of spatial multiplexing from 1 to 5 with the virtual node fixed to 1 [ratio(order = 1 → 5)] is similar or slightly superior to that of increasing the number of virtual nodes from 1 to 5 without spatial multiplexing [ratio($V = 1 → 5$)] in terms of capacity (Fig. 6, left diagram). These features are commonly observed in all the experimented parameter settings in this study. For example, when the order of spatial multiplexing is varied from 1 to 5, the averaged improvement ratio of the memory capacity calculated using all the parameter settings is 2.11, and the maximum improvement ratio among all is 3.23 when $\tau\Delta = 0.5$, $V = 1$, and $N = 5$ (Fig. 7, upper left diagram). Furthermore, we observe "ratio(order = 1 → 5) > ratio($V = 1 → 5$)" in almost all the parameter settings (Fig. 7, upper right diagram), which characterizes the range of effectiveness of the spatial multiplexing.

For the NARMA task, by increasing the order of spatial multiplexing, the value of the improvement ratio is decreased, suggesting the improvements of the performance (Fig. 6, right diagrams). In particular, in the NARMA5 task when $V = 25$, the value decreased by a factor of 10 when the order of spatial multiplexing is varied from 1 to 5. Interestingly, this improvement ratio is far superior to that of increasing the virtual node from 1 to 25 despite the larger increase in the computational nodes, which implies that cases exist in which the increase of the order of spatial multiplexing behaves superior to that of temporal multiplexing. This outcome is caused by the difference between the type of information processing that the spatial multiplexing and that temporal multiplexing

can provide, as well as the type of information processing capability that would be needed to perform the task. These tendencies follow in all the experimented parameter settings in this study. The performance of each NARMA task improves in each parameter setting by increasing the order of spatial multiplexing (Fig. 7, lower left diagram). For example, when the order of spatial multiplexing is varied from 1 to 5, the average improvement ratio calculated using all the parameter settings is 0.39 in the NARMA5 task and the minimum improvement ratio among all is 0.15 when $\tau\Delta = 2$, $V = 25$, and $N = 5$ in the NARMA5 task (Fig. 7, lower left diagram). Similarly to the case for the memory capacity, in each NARMA task, we observe "ratio(order = 1 → 5) < ratio($V = 1 → 5$)" in almost all the parameter settings (Fig. 7, lower right diagram). These results suggest that in some cases, spatial multiplexing adds a more effective number of computational nodes than does temporal multiplexing.

IV. TOWARD ENGINEERING QUANTUM RESERVOIR THROUGH SPATIAL MULTIPLEXING

As we see in Sec. II E and demonstrate in Sec. III, spatial multiplexing improves the performance of the system. In this section, we provide a few notes on the possibility to engineer QR through the spatial multiplexing scheme. As we discussed in Sec. II E, although we can improve performance by increasing the order of spatial multiplexing in theory, this does not always apply in actual experiments because of overfitting. In such cases, limiting the

memory capacity

average ⊞ max □

NARMA task

min □ NARMA10 ⊙ NARMA2 ●
NARMA2 ⊞ NARMA15 ● NARMA15 ●
NARMA5 ■ NARMA20 △ NARMA20 △
                        NARMA5 ■
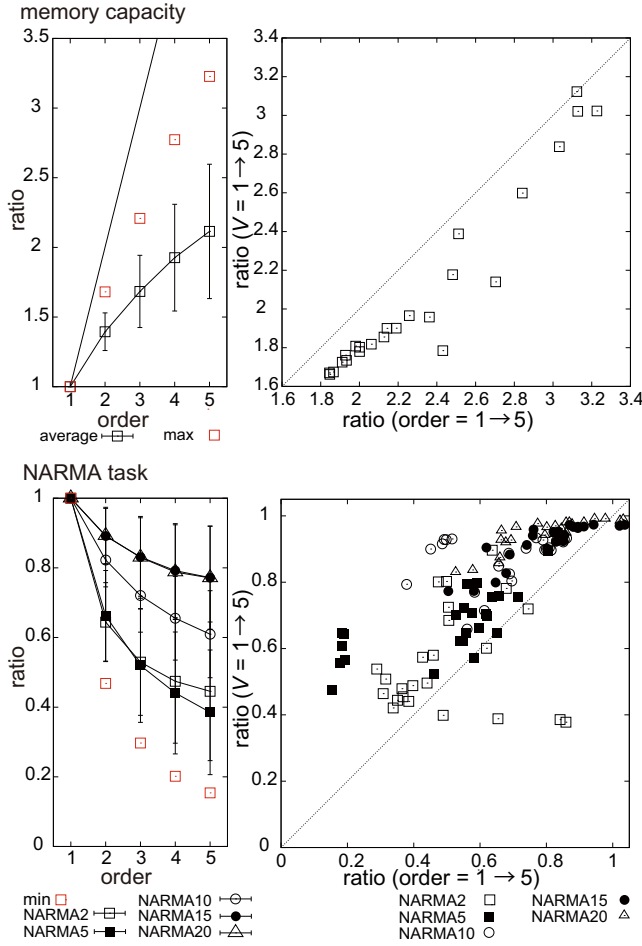   NARMA10 ○

FIG. 7.   Analyses of the averaged improvement ratios in terms of the averaged memory capacity (upper row) and the averaged NMSE for the NARMA tasks (lower row). According to the order of spatial multiplexing, the improvement ratios are calculated in all the combinations of the number of qubits (3, 4, and 5); virtual nodes (1, 5, and 25); and $\tau\Delta$ (0.5, 1, 2, 3, 4, 8, 16, and 32) and they are averaged using all these combinations (left diagram in each row). Note that the solid line in the upper left diagram expresses $y = x$ as a reference. The error bars show standard deviations, and the maximum or minimum improvement ratio among all the experimental conditions in each order of spatial multiplexing is also plotted. The right diagram in each row shows the averaged improvement ratios for spatial multiplexing against those for temporal multiplexing in each analysis. The ratio is calculated when the order is increased from 1 to 5 with the virtual node fixed to 1 for the spatial multiplexing (x axis) and when the virtual node is increased from 1 to 5 without spatial multiplexing for the temporal multiplexing (y axis) to make the increased number of computational nodes the same for comparison. The plots for all the experimental conditions are overlaid. For all the analyses, the averaged MC and NMSE used to calculate the improvement ratio in each condition are obtained from the results of 100 trials with different QR systems.

number of computational nodes is preferable. If we have two QRs with the same number of true nodes and if these true nodes originate from different quantum systems (or

different combinations of quantum systems) with different numbers of qubits, then—as we see in the previous sections—the computational capability of the QRs differs in general. This outcome is largely due to the difference of the number of hidden nodes, the exponential numbers of degrees of freedom based on the number of qubits behind the measurements, which influence the dynamics of the true nodes. Then, which combination is most efficient and can exert the performance the best? Given a fixed number of computational nodes, we investigate in this section a method to engineer the efficient combinations of reservoirs.

Similar to Sec. II E, let us assume that we have three reservoirs, A, B, and C, with reservoir A having N computational nodes and reservoirs B and C having the same number of nodes N'. We also assume that these reservoirs satisfy the basic properties of the regression equation setting and least-squares solutions presented in Sec. II E. At first, the inequality $r^2_{A+B+C} \leq \min\{r^2_A, r^2_B, r^2_C, r^2_{A+B}, r^2_{B+C}, r^2_{A+C}\}$ suggests that combining reservoirs A, B, and C performs best if we could avoid overfitting in practice. Now, by retaining the total number of nodes fixed to $N + N'$, we determine the better choice between reservoir B or C for combination with reservoir A to improve performance.

At first glance, choosing the reservoir that has better performance is preferable. However, this is not always the case. Given that reservoir B has better performance than reservoir C, that is, $r^2_B \leq r^2_C$, then $r^2_{A+B} \leq r^2_A$ and $r^2_{A+C} \leq r^2_A$ hold, but $r^2_{A+B} \leq r^2_{A+C}$ does not hold in general. [We can easily find a counterexample such as $y = (1\ 1\ 1)^{\mathrm{T}}$, $X_A = (0.25\ 1\ 0)^{\mathrm{T}}$, $X_B = (1\ 0\ 0)^{\mathrm{T}}$, $X_C = (0\ 1\ -1)^{\mathrm{T}}$.] We then apply the relations we obtain in Sec. II E to reservoirs $A + B$ and $A + C$, which are

$$\max\{(r^2_A - \lambda_{Q_{A,A+B}}\|y\|^2), (r^2_B - \lambda_{Q_{B,A+B}}\|y\|^2)\}$$
$$\leq r^2_{A+B} \leq \min\{r^2_A, r^2_B\}$$

and

$$\max\{(r^2_A - \lambda_{Q_{A,A+C}}\|y\|^2), (r^2_C - \lambda_{Q_{C,A+C}}\|y\|^2)\}$$
$$\leq r^2_{A+C} \leq \min\{r^2_A, r^2_C\}.$$

Given $r^2_B \leq r^2_C$, to evaluate r^2_{A+B} and r^2_{A+C}, we need to check how these ranges overlap. Only if no overlap exists can we safely predict the reservoir to add without actually performing the task. When $r^2_A \leq r^2_B \leq r^2_C$, these two ranges always overlap because $r^2_{A+B} \leq r^2_A$ and $r^2_{A+C} \leq r^2_A$. When $r^2_B \leq r^2_A \leq r^2_C$ or $r^2_B \leq r^2_C \leq r^2_A$, and if $r^2_B < \max\{(r^2_A - \lambda_{Q_{A,A+C}}\|y\|^2), (r^2_C - \lambda_{Q_{C,A+C}}\|y\|^2)\}$ holds, then we can safely decide to choose reservoir B as the appropriate partner for combination without actually performing the task, because it satisfies $r^2_B < r^2_{A+C}$ and accordingly $r^2_{A+B} < r^2_{A+C}$ holds.

Furthermore, although we demonstrate spatial multiplexing by combining QR systems that have different coupling strengths with the other parameters fixed in our numerical experiments, note that the combination can consist of any reservoirs if they are not synchronized. The choice of the combination depends on the efficiency of the usage in each experimental setting. For example, parameter $\tau\Delta$ is dependent on the energy applied to the experimental platform and can be regulated if we consider energy efficiency.

V. DISCUSSION

In this paper, we introduce a scheme, spatial multiplexing, to boost the computational power in QRC. Considering the physical experiment, this scheme is operationally easy to implement but is remarkably effective, and we theoretically show that the scheme inevitably increases the computational power. The effect is demonstrated through numerical experiments using a number of benchmark tasks and the performance of learning is observed to be improved. We also examine the theoretical implications of the proposed scheme and discuss its range of validity and limitations. This scheme is useful and applicable not only for QRC but also for reservoir computing in general, including the case of conventional software implementations.

Although the scheme of spatial multiplexing allows us to efficiently increase the computational nodes, we should be sensitive to the case of overfitting in practical applications. In our experiments, we observe several performances, which are thought to be caused by overfitting [e.g., the results of NARMA tasks in higher values of $\tau\Delta$ (Fig. 9 in Appendix A)]. To avoid these situations, one can introduce a Ridge regression or Lasso for the training procedure, which assigns a penalty to readout weights for regressions. By combining with these sparse regressions, one can establish a scheme to selectively exploit effective degrees of freedom from massive computational nodes increased by spatial multiplexing.

The NMR ensemble system has been regarded as a strong candidate for a physical platform of QRC. In the NMR quantum reservoir system, the spatial multiplexing with some different molecules, introduced in Sec. II C, is an easier option to increase the computational power than increasing the number of addressable qubits. Another option is increasing the number of unaddressable qubits and virtual nodes, which will be introduced in detail in our future work. We can also introduce an easier implementation of spatial multiplexing even with the same molecule with NMR pulse techniques to change the interaction Hamiltonian effectively [31,32]. The pulse techniques are often utilized for quantum simulation experiments. For example, the Ising-type Hamiltonian X_iX_j can be changed to $X_iX_j + Y_iY_j + aZ_iZ_j$ for any parameters a, by applying

multiple pulse sequences [33]. It was shown in a quantum simulation experiment [34] that the dynamical behavior of a nuclear spin system with the interactions $X_iX_j + (2a - 1)Y_iY_j - 2aZ_iZ_j$ is substantially different depending on a. Just by changing the parameter of applying the pulse, we can easily implement the spatial multiplexing with some different quantum dynamical systems in the same molecule.

Spatial multiplexing offers an opportunity to increase the computational nodes and boost the computational power not only for QRs but also for any interacting systems that contain components that are operationally or experimentally difficult to manipulate and increase. By extending this line of thought, we can develop a concept of composing multiple reservoirs, each with different physical systems. For example, it might be worth composing photonic and quantum systems and treating them as one entire reservoir in some applications. According to how this scheme is applied in the real world, this concept would create options from which to flexibly choose the physical systems to use as a computational resource in a given situation.

Finally, one of the intriguing flavors in the framework of QRC is its exploitation of the quantum computational supremacy region, where the system possesses exponential degrees of freedom as hidden nodes. We reiterate that, as spatial multiplexing increases true nodes proportionally to its order, its increase of hidden nodes is also proportional, while increasing the number of qubits in the interacting system will directly lead to an exponential increase in hidden nodes. This fact implies that, even if we have the same number of true nodes, the number of hidden nodes can differ according to how the spin-ensemble molecular samples are prepared; hence, the computational power and preference also differ. We suggest that each experimenter regulate how to prepare their reservoirs based on their given experimental conditions and their operability of the system, and we believe that the spatial multiplexing technique will become one of the common and practical options for boosting the computational power of QRs in the near future.

APPENDIX A: EXTENDED NUMERICAL EXPERIMENTS AND ANALYSES

In the main text, we show the results of the numerical experiments for the QR system with its system parameters set to $N = 5$ and $\tau\Delta = 1, 2$. In this appendix, we show thorough and systematic analyses of different parameter settings, varying N as 3, 4, and 5 and varying $\tau\Delta$ as $0.5, 1, 2, 3, 4, 8, 16$, and 32, which are summarized in Figs. 8 and 9.

APPENDIX B: ECHO STATE NETWORK SETTINGS FOR COMPARISON

To characterize the computational capability of our system, in the main text, we compare its performance of the NARMA tasks and its memory capacity with those of a conventional ESN [2,35,36]. This appendix explains in detail the settings of the ESN used for the comparisons.

The ESNs are a type of random recurrent neural network that consists of internal computational nodes (the number of internal computational nodes is denoted as N_{ESN}), input nodes, and output nodes. The activation of the ith internal node at timestep k is expressed as x_k^i. The weights w^{ij} for the internal network connect the ith node to the jth node, and the input weights w_{in}^i connect the input node to the ith internal node. Internal computational nodes with one bias are connected to the output unit through readout weights w_{out}^i, where $x_k^0 = 1$ and w_{out}^0 is assigned for the bias term.

Learning of the readout weights w_{out}^i is performed using the same procedure explained in the main text for each task. The internal weights w^{ij} are randomly determined from the range $[-1.0, 1.0]$ and the spectral radius of the weights is regulated according to the setting for each task, as explained below. Similarly, the input weights w_{in}^i are randomly determined from the range $[-\sigma, \sigma]$, where σ is a scaling parameter explained later. The time evolution of the ESN is expressed as follows:

$$x_k^i = f\left(\sum_{j=1}^{N_{\text{ESN}}} w^{ij} x_{k-1}^j + w_{\text{in}}^i u_k\right), \quad \text{(B1)}$$

$$y_k = \sum_{i=0}^{N_{\text{ESN}}} w_{\text{out}}^j x_k^i, \quad \text{(B2)}$$

where $f(x)$ is set as $\tanh(x)$ in this paper. To make a fair comparison of the task performance, the I/O setting of the ESN is set to be the same as that of our system for each task. For example, the lengths of the washout, training, and evaluation phases and the evaluation procedures are kept the same. The detailed experimental conditions are given for each of these comparisons below.

For the NARMA task, we first prepare 10 different ESNs for each setting of N_{ESN}, which vary as 5, 10, 20, 30, 40, 50, 100, 150, 200, 250, and 300. The scaling parameter of the input weights σ is varied as 1.0, 0.5, 0.2, 0.1, 0.05, 0.01, 0.005, and 0.001, and the spectral radius of the internal weights is also varied from 0.1 to 2.0 in increments of 0.1. For each ESN, by fixing the spectral radius and the parameter σ, we run 10 different trials, driven by different random input sequences, and test the emulation tasks of all the NARMA systems (NARMA2, 5, 10, 15, and 20) using
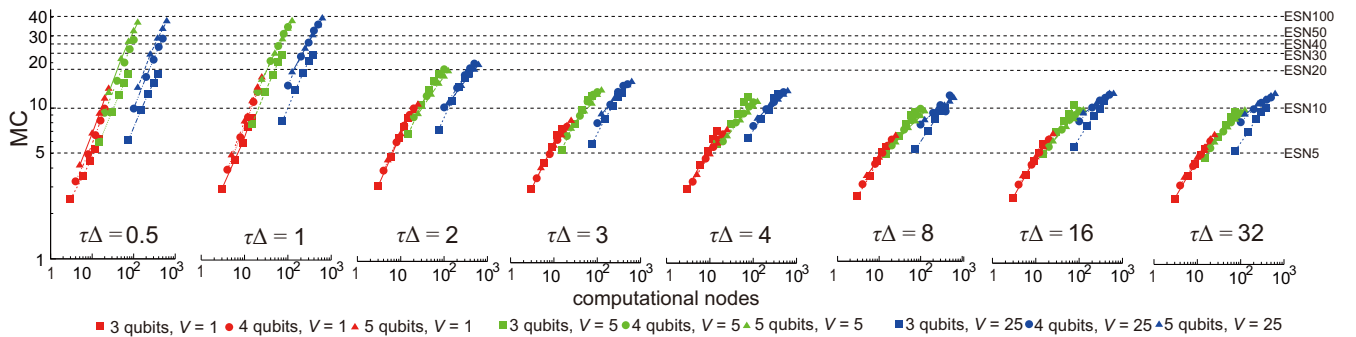

FIG. 8. Plots showing the effect of the spatial multiplexing in terms of the averaged MC for each number of qubits of a single QR system, each number of virtual nodes, and each parameter $\tau\Delta$. For each plot, the horizontal axis represents the total computational nodes in a QR system and the vertical axis represents the averaged MC. The parameter $\tau\Delta$ is varied as $0.5, 1, 2, 3, 4, 8, 16$, and 32. The number of qubits in a single QR system is represented by different point shapes (squares, circles, and triangles indicate the cases for 3, 4, and 5 qubits, respectively). The number of virtual nodes is represented as a difference in color (the numbers of virtual nodes $V = 1, 5$, and 25 are represented as black, red, and green, respectively). The plots connected with lines represent the results when the order of spatial multiplexing is increased from 1 to 5 (this can be seen from the increase in the total number of computational nodes) with other system parameters fixed. As a reference, each plot contains the performance of the conventional ESN. The notation "ESN20," for example, represents the averaged MC of the ESN with 20 nodes.
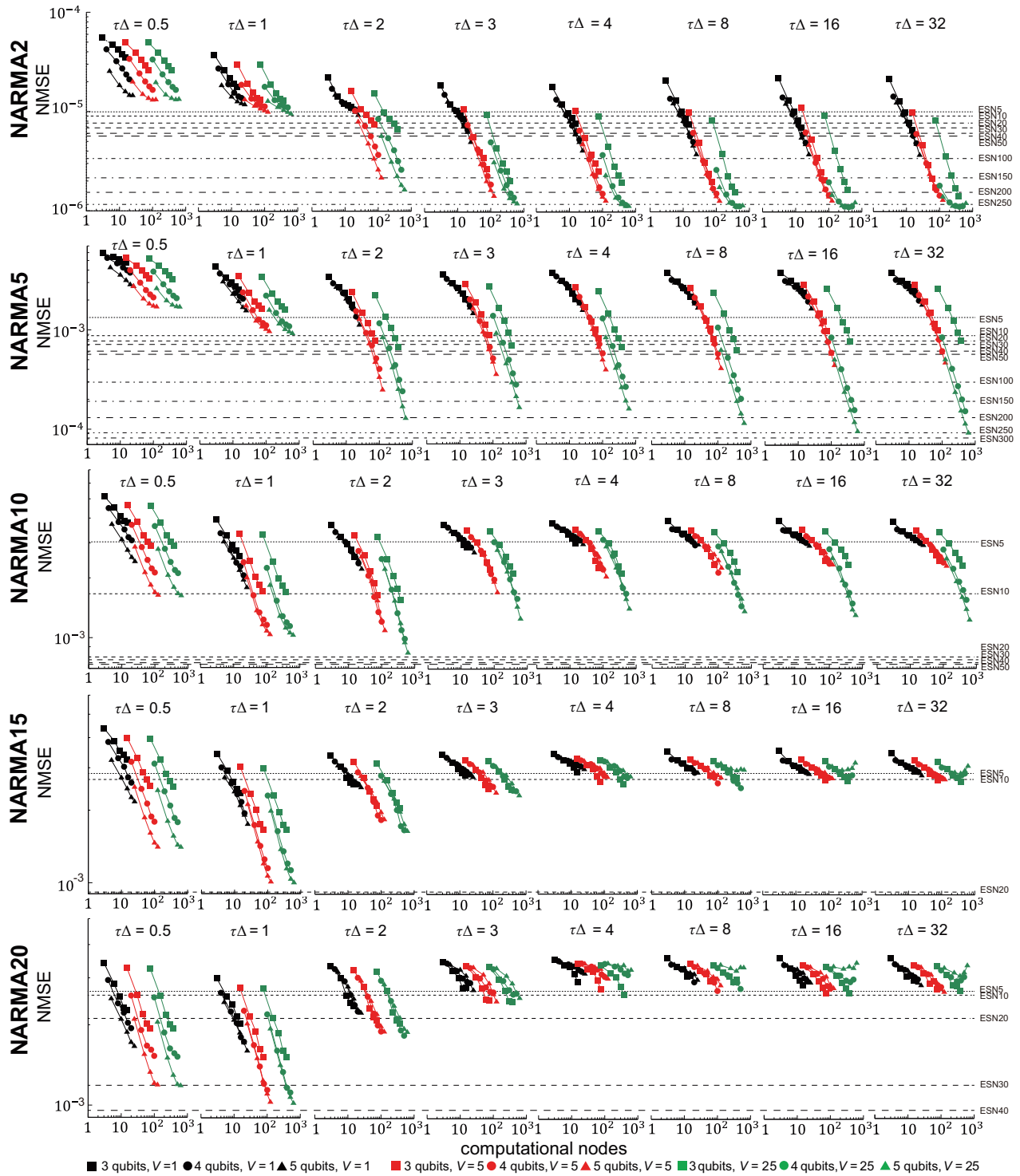
FIG. 9.   Plots showing the effect of the spatial multiplexing in terms of the averaged NMSE for the NARMA tasks for each number of qubits of a single QR system, each number of virtual nodes, and each parameter $\tau\Delta$. The cases for NARMA2, 5, 10, 15, and 20 are investigated. For each plot, the horizontal axis represents the total computational nodes in a QR system, and the vertical axis represents the averaged NMSE. The parameter $\tau\Delta$ is varied as 0.5, 1, 2, 3, 4, 8, 16, and 32. The number of qubits in a single QR system is represented by different point shapes (squares, circles, and triangles indicate the cases for 3, 4, and 5 qubits, respectively). The number of virtual nodes is represented as a difference in color (the numbers of virtual nodes $V = 1, 5$, and 25 are represented as black, red, and green, respectively). The plots connected with lines represent the results when the order of spatial multiplexing is increased from 1 to 5 (this can be seen from the increase in the total number of computational nodes) with other system parameters fixed. As a reference, each plot contains the performance of the conventional ESN. The notation "ESN20," for example, represents the averaged NMSE of the ESN with 20 nodes.

a multitasking scheme for each trial. After performing all the trials of the NARMA tasks with all the parameter settings varied for each ESN having the computational node N_{ESN}, we collect the lowest NMSE, which indicates the best performance in this experiment corresponding to the ESN, and calculate the averaged NMSE for each NARMA task over 10 different ESNs for each setting of N_{ESN}. These averaged NMSEs are used for comparison.

To evaluate the memory capacities, 100 different ESNs are driven by different random input sequences with a spectral radius fixed at 0.9 and the scaling parameter of the input weights fixed to $\sigma = 0.01$. The emulation tasks of five dynamical systems with different degrees of nonlinearity, which are explained in the main text, are performed for each trial using a multitasking scheme. Analyses of the performance are conducted using the same procedures used by our system and defined in the main text. The averaged capacities are calculated using these 100 trials and used for comparison.

[1] P. A. Merolla *et al.*, A million spiking-neuron integrated circuit with a scalable communication network and interface, Science **345**, 668 (2014).

[2] H. Jaeger and H. Haas, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, Science **304**, 78 (2004).

[3] W. Maass, T. Natschläger, and H. Markram, Real-time computing without stable states: A new framework for neural computation based on perturbations, Neural Comput. **14**, 2531 (2002).

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

[5] M. Lukoševičius and H. Jaeger, Reservoir computing approaches to recurrent neural network training, Comput. Sci. Rev. **3**, 127 (2009).

[6] A. Z. Stieg, A. V. Avizienis, H. O. Sillin, C. Martin-Olmos, M. Aono, and J. K. Gimzewski, Emergent criticality in complex turing b-type atomic switch networks, Adv. Mater. **24**, 286 (2012).

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

[8] C. Fernando and S. Sojakka, *Pattern recognition in a bucket*, Lecture Notes in Computer Science Vol. 2801 (Springer, Berlin, Heidelberg, 2003), p. 588.

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

[10] L. Larger, M. C. Soriano, D. Brunner, L. Appeltant, J. M. Gutierrez, L. Pesquera, C. R. Mirasso, and I. Fischer, Photonic information processing beyond Turing: An optoelectronic implementation of reservoir computing, Opt. Express **20**, 3241 (2012).

[11] J. Torrejon, M. Riou, F. A. Araujo, S. Tsunegi, G. Khalsa, D. Querlioz, P. Bortolotti, V. Cros, A. Fukushima, H. Kubota, S. Yuasa, M. D. Stiles, and J. Grollier, Neuromorphic computing with nanoscale spintronic oscillators, Nature **547**, 428 (2017).

[12] T. Furuta, K. Fujii, K. Nakajima, S. Tsunegi, H. Kubota, Y. Suzuki, and S. Miwa, Macromagnetic Simulation for Reservoir Computing Utilizing Spin Dynamics in Magnetic Tunnel Junctions, Phys. Rev. Appl. **10**, 034063 (2018).

[13] S. Tsunegi, T. Taniguchi, S. Miwa, K. Nakajima, K. Yakushiji, A. Fukushima, S. Yuasa, and H. Kubota, Evaluation of memory capacity of spin torque oscillator for recurrent neural networks, Jpn. J. Appl. Phys. **57**, 120307 (2018).

[14] K. Nakajima, H. Hauser, R. Kang, E. Guglielmino, D. G. Caldwell, and R. Pfeifer, A soft body as a reservoir: Case studies in a dynamic model of octopus-inspired soft robotic arm, Front. Comput. Neurosci. **7**, 1 (2013).

[15] K. Nakajima, T. Li, H. Hauser, and R. Pfeifer, Exploiting short-term memory in soft body dynamics as a computational resource, J. R. Soc. Interface **11**, 20140437 (2014).

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

[17] K. Nakajima, H. Hauser, T. Li, and R. Pfeifer, Exploiting the dynamics of soft materials for machine learning, Soft Robotics **5**, 339 (2018).

[18] K. Fujii and K. Nakajima, Harnessing Disordered-ensemble Quantum Dynamics for Machine Learning, Phys. Rev. Appl. **8**, 024030 (2017).

[19] J. Preskill, Quantum computing and the entanglement frontier, arXiv:1203.5813 (2012).

[20] D. G. Cory, R. Laflamme, E. Knill, L. Viola, T. F. Havel, N. Boulant, G. Boutis, E. Fortunato, S. Lloyd, R. Martinez, C. Negrevergne, M. Pravia, Y. Sharf, G. Teklemariam, Y. S. Weinstein, and W. H. Zurek, NMR based quantum information processing: Achievements and prospects, Fortschr. Phys. **48**, 875 (2000).

[21] J. A. Jones, Quantum computing with NMR, Prog. Nucl. Magn. Reson. Spectros. **59**, 91 (2011).

[22] M. Negoro, K. Mitarai, K. Fujii, K. Nakajima, and M. Kitagawa, Machine learning with controllable quantum dynamics of a nuclear spin ensemble in a solid, arXiv:1806.10910 (2018).

[23] In [18], the procedure of temporal multiplexing is called "time multiplexing," but it is the same thing. We use the term "temporal" to coordinate with the term "spatial" in this paper.

[24] C. Negrevergne, T. S. Mahesh, C. A. Ryan, M. Ditty, F. Cyr-Racine, W. Power, N. Boulant, T. Havel, D. G. Cory, and R. Laflamme, Benchmarking Quantum Control Methods on a 12-qubit System, Phys. Rev. Lett. **96**, 170501 (2006).

[25] D. Lu, K. Li, J. Li, H. Katiyar, A. J. Park, G. Feng, T. Xin, H. Li, G. Long, A. Brodutch, J. Baugh, B. Zeng, and R. Laflamme, Enhancing quantum control by bootstrapping a quantum processor of 12 qubits, npj Quantum Information **3**, 45 (2017).

[26] Z. Lu, B. R. Hunt, and E. Ott, Attractor reconstruction by machine learning, CHAOS **28**, 061104 (2018).

[27] N. Bertschinger and T. Natschläger, Real-time computation at the edge of chaos in recurrent neural networks, Neural Comput. **16**, 1413 (2004).

[28] R. Toral, C. R. Mirasso, E. Hernández-Garcia, and O. Piro, Analytical and numerical studies of noise-induced synchronization of chaotic systems, CHAOS **11**, 665 (2001).

[29] H. Jaeger, GMD Report No. 152, German National Research Center for Information Technology (2001).

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

[31] U. Haeberlen, *High Resolution NMR in Solids — Selective Averaging*, Advances in Magnetic Resonance Suppl. 1 (Academic Press, New York, 1976).

[32] L. M. Vandersypen and I. L. Chuang, NMR techniques for quantum control and computation, Rev. Mod. Phys. **76**, 1037 (2005).

[33] G. Roumpos, C. P. Master, and Y. Yamamoto, Quantum simulation of spin ordering with nuclear spins in a solid-state lattice, Phys. Rev. B **75**, 094415 (2007).

[34] G. A. Alvalez, D. Suter, and R. Kaiser, Localization-delocalization transition in the dynamics of dipolar-coupled nuclear spins, Science **349**, 846 (2015).

[35] H. Jaeger, GMD Report No. 159, German National Research Center for Information Technology (2002).

[36] H. Jaeger, GMD Report No. 148, German National Research Institute for Computer Science (2001).