COMPUTER SCIENCE

Designing spontaneous behavioral switching via chaotic itinerancy

Katsuma Inoue*, Kohei Nakajima*, Yasuo Kuniyoshi*†

Chaotic itinerancy is a frequently observed phenomenon in high-dimensional nonlinear dynamical systems and is characterized by itinerant transitions among multiple quasi-attractors. Several studies have pointed out that high-dimensional activity in animal brains can be observed to exhibit chaotic itinerancy, which is considered to play a critical role in the spontaneous behavior generation of animals. Thus, how to design desired chaotic itinerancy is a topic of great interest, particularly for neurorobotics researchers who wish to understand and implement autonomous behavioral controls. However, it is generally difficult to gain control over high-dimensional nonlinear dynamical systems. In this study, we propose a method for implementing chaotic itinerancy reproducibly in a high-dimensional chaotic neural network. We demonstrate that our method enables us to easily design both the trajectories of quasi-attractors and the transition rules among them simply by adjusting the limited number of system parameters and by using the intrinsic high-dimensional chaos.

INTRODUCTION

Designing a cognitive architecture that acts spontaneously in a real-world environment is one of the ultimate goals in the field of cognitive robotics (*1*). A cognitive agent is expected to have autonomy, i.e., the agent should behave independently of the designer's control while maintaining its identity. Adaptability is another requirement for cognitive functionality. Thus, the agent must select the appropriate behavior continuously and robustly in response to the changing environment in real time. To summarize, the agent's cognitive behavior should be implemented through the body-environment interaction while still enabling the agent to maintain its autonomy and adaptability.

In the conventional context of robotics and artificial intelligence, designers often take top-down approaches to provide an agent with a hierarchical structure corresponding to the behavioral category. This representation-based approach has a critical limitation in the design of a cognitive agent: The static, one-to-one relationship between the behavior and structure makes it difficult to adapt flexibly to the dynamically changing environment and developing body. For example, it has been considered that the motion control systems of living things, including humans, realize their high-order motion plans by combining reproducible motor patterns called motion primitives (*2*). Inspired by this viewpoint, we tend to realize an agent's behavior control with a predetermined static hierarchical structure. However, such a hierarchical structure does not exist in living organisms from the beginning; rather, these structures are cultivated through the body's development and dynamic interactions with the environment. Therefore, it is important to introduce a dynamical perspective to understand a hierarchical structure of behavior control generated in animals that have adaptability and flexible plasticity.

In robotics, approaches based on dynamical systems theory have been applied to analyze and control agents being modeled as sets of variables and parameters on a phase space (*3*, *4*). This dynamical systems approach can deal with both the functional hierarchy and the elementary motion in a unified form by expressing the physical constraints of the agent as the temporal development of state variables, namely, dynamics. For example, Jaeger (*4*) sketched a pioneering idea of an algorithm where the behavior of an agent is expressed as dynamics, and both the behavioral regularity (referred to as transient attractors) and the higher-order relationships among them are extracted in a bottom-up manner. Unlike the stochastic approach where the randomness of the system is realized by a probabilistic model (e.g., the Markov model), the dynamical system approach can consistently express the agent's seemingly random behaviors via its chaoticity (*5*). Furthermore, in the stochastic approach, hierarchical structures are inevitably introduced since the mechanism of random number generation is completely independent of the system's dynamics. Thus, the dynamical systems approach has the potential to model an agent's spontaneous hierarchical behavior in the form of dynamic interaction without top-down structure given by the external designer. The important issue here is that the concept of the dynamical system approach itself offers no general principle for implementing these behaviors on a large number of nonlinear couplings that constitute the body-environment interaction. Therefore, it is important to study and propose the methodologies for designing spontaneous hierarchical behavior with a consistent temporal evolution rule governing the system's dynamics.

Following this dynamical systems perspective, chaotic itinerancy (CI) (*6*–*8*) is a powerful option for modeling spontaneous behavior with the functional hierarchy realized through the dynamics. CI is a frequently observed, nonlinear phenomenon in high-dimensional dynamical systems, and it is characterized by chaotically itinerant transitions among locally contracting domains, namely, quasi-attractors (*8*). In general, a chaotic system has an initial sensitivity, and a slight difference in phase space is exponentially expanded in a certain direction with temporal development. Conversely, multiple transiently

Graduate School of Information Science and Technology, The University of Tokyo, Engineering Building 2, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan.
*Corresponding author. Email: k-inoue@isi.imi.i.u-tokyo.ac.jp (K.I.); k_nakajima@mech.t.u-tokyo.ac.jp (K.N.); kuniyosh@isi.imi.i.u-tokyo.ac.jp (Y.K.)
†Present address: Next Generation Artificial Intelligence Research Center, The University of Tokyo, Engineering Building 2, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan; RIKEN CBS-Toyota Collaboration Center, 2-1 Hirosawa, Wako, Saitama 351-0198, Japan; Center for Research and Development Strategy (CRDS), Japan Science and Technology Agency (JST), K's Gobancho Building, 7 Gobancho, Chiyoda-ku, Tokyo 102-0076, Japan; Department of Research Promotion, Research Division, JST, K's Gobancho Building, 7 Gobancho, Chiyoda-ku, Tokyo 102-0076, Japan; Aidemy Inc., Yamazin Building 3F, 1-1-16, Kanda Ogawamachi, Chiyoda-ku, Tokyo 101-0052, Japan.

predictable dynamics can be repeatedly observed in a chaotic system, yielding CI despite the global chaoticity and initial sensitivity. This type of hierarchical dynamics frequently emerges from high-dimensional chaos even without hierarchical mechanisms, implying that explicit structure is not necessarily needed for implementing hierarchical behaviors. The chaoticity plays an important role in forming the autonomy of an agent as it is virtually impossible for the designer to completely predict and control an agent's behavior because of the agent's initial sensitivities, which essentially ensures the agent's independence from the designer. Thus, CI would work as an effective tool for implementing the intellectual behavior of a cognitive agent by embedding the behavior in the form of a quasi-attractor and maintaining the autonomy of the agent with the chaoticity.

CI was first found in a model of optical turbulence (*6*). Since its discovery, similar phenomena have been numerically obtained in various setups (*7*, *9*, *10*). The properties of CI vary among previous studies, and some classes of CI present interesting features that are difficult to characterize using the stochastic processes. Tsuda *et al.* (*9*), for example, show that their asynchronous neural network model produces itinerant behavior whose transition frequency is characterized by a long-tailed distribution. The coupled map lattice proposed by Kaneko (*7*) potentially emits infinitely many states, whose transition rule should be represented by an infinite state machine. Several physiological studies have reported that CI-like dynamics have even occurred in brain activity, suggesting that CI might play an essential role in forming cognitive functions (*11*, *12*). For example, Freeman (*13*) revealed that an irregular transition among learned states was observed in the electroencephalogram pattern of a rabbit olfactory system when a novel input was given, indicating that the cognitive conditions corresponding to "I don't know" are internally realized as CI-like dynamics. Furthermore, a recent observation of rat auditory cortex cell activity revealed the existence of a random shift among different stereotypical activities corresponding to individual external stimuli during anesthesia (*14*). On the basis of these reports, Kurikawa and Kaneko (*15*) suggested the novel idea of "memory-as-bifurcation" to understand the mechanism of the transitory phenomenon. They reproduced it in an associative memory model in which several input-output functions were embedded by Hebbian learning. An intermittent switching among metastable patterns was also observed in a recurrent neural network (RNN) by installing multiple feedback loops trained to output a specific transient dynamics corresponding to external transient inputs (*16*). CI-like dynamics arise not only in nervous systems but also in interactions between agent's bodies and their surrounding environments (*17*–*19*).

For example, Kuniyoshi and Sangawa (*17*) developed a human fetal development model by coupling chaotic central pattern generators and a musculoskeletal system. They reported that several common behaviors, such as crawling and rolling over, emerged from the physical constraint. Therefore, CI is a nonlinear phenomenon of high-dimensional dynamical systems and is thought to play a substantial role in generating structural behavior.

Inspired by the contribution of CI to the cognitive functions and spontaneous motion generation of agents, CI has been used for motion control in the field of neurorobotics and cognitive robotics by designing the CI trajectory. For example, Namikawa *et al.* (*20*–*22*) designed stochastic motion switching among predetermined motion primitives in a humanoid robot by using a hierarchical, deterministic RNN controller. In this study, it was confirmed that lower-order RNNs with smaller time constants stably produced the trajectories of motion primitives, whereas higher-order RNNs with larger time constants realized a pseudo-stochastic transition by exploiting self-organized chaoticity. Steingrube *et al.* (*23*) designed a robot that skillfully broke a deadlock state in which the motion had completely stopped by using chaos in the RNN controller. Hence, it can be interpreted that CI-like dynamics were embedded in the coupling of the body and the surrounding environment.

While CI is an important phenomenon in high-dimensional dynamical systems, roboticists also find it a useful tool for designing an agent's behavior structure while maintaining the agent's autonomy. However, it has generally been difficult to embed desired quasi-attractors at will because of their nonlinearity and high dimensionality. For example, in the method of Namikawa *et al.* (*20*–*22*), the internal connections of an RNN ware trained with backpropagation through time (*24*); however, embedding a long-term input-output function in an RNN by the gradient descent method is generally unstable and requires a large number of learning epochs (*25*). Furthermore, their method required both a hierarchical structure and the same number of separated modules as the motion primitives, restricting the scalability and the range of its application. Yamashita and Tani (*26*) proposed a network model learning functional hierarchy without any modules, in which hierarchical behavior was implemented by the explicit hierarchical structure governed by predetermined multiple time scales. In addition, methods using the associated memory model (*10*, *27*–*29*) are also unsuitable for our purpose since it is difficult to embed the quasi-attractors with complicated spatiotemporal patterns.

In this study, we propose an algorithm, freely designing both the trajectories of quasi-attractors and transition rules among them in a setup of high-dimensional chaotic dynamical systems. We aim to design the properties of CI characterized by a finite state machine and finite switching time, such as those in the neurorobotics context (*19*, *21*). We prepare transition rules described by a Markov model and aim to emulate them through CI using a high-dimensional nonlinear dynamical system. Our method uses batch learning composed of the following three-step procedure (Fig. 1):

Step 1. Prepare a high-dimensional chaotic system where target quasi-attractors are embedded. We used a widely used echo state network (ESN) (*30*), one type of RNN, as a high-dimensional chaotic system. This ESN contains no hierarchical structure and modules (e.g., multiple time scales), and every network node shares the same time scale parameter. At the same time, modify the interactions (internal parameters) so that the system reproducibly generates intrinsic complex trajectories generated by an initial chaotic system (innate trajectories) corresponding to the type of the discrete inputs (named symbol). In parallel, train the linear regression model (named readout) to output the designated trajectories (output dynamics) by exploiting the embedded innate trajectory. This process can be potentially applied to the other chaotic dynamical systems not limited to RNN in silico (*31*) since neither modules nor hierarchical structures are required. In addition, this embedding process is accomplished by modifying fewer parameters using the method of reservoir computing (*32*, *33*). Therefore, our scheme is more stable and less computationally expensive than conventional methods using backpropagation to train the network parameters.

Step 2. Add a feedback classifier to the trained chaotic systems for autonomously generating specific symbolic dynamics. In the training of the feedback discriminator, the network's internal parameters are fixed, as with the readout in step 1. Thus, by using the

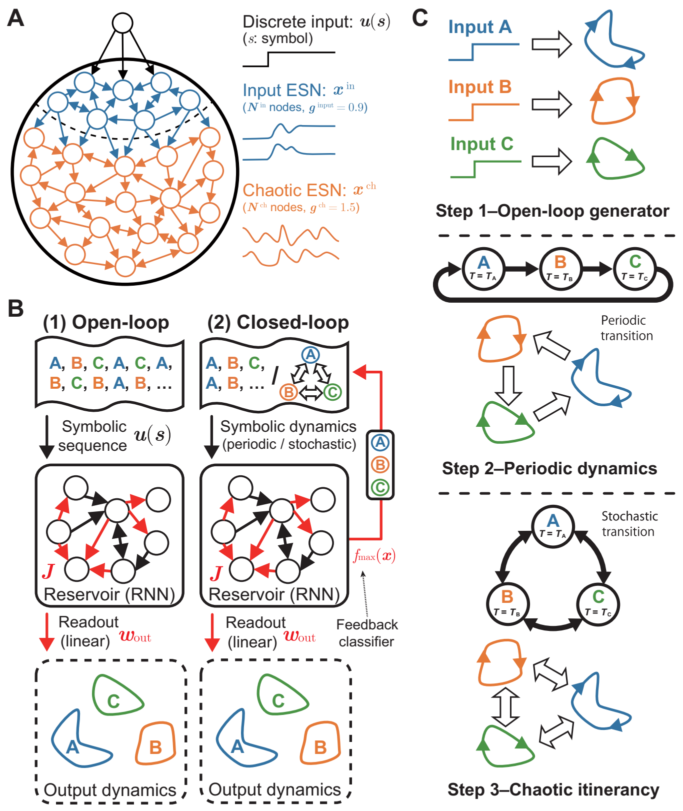

Fig. 1. Experimental setups. (**A**) Schematic diagram of a high-dimensional chaotic system prepared in our experiments. The system can be divided into two parts: input echo state network (ESN) and chaotic ESN. Input ESN acts as an interface between the discrete input and the chaotic ESN, generating transient dynamics projecting onto the chaotic ESN when the symbolic input switches. To prevent the chaotic ESN from becoming nonchaotic because of the bifurcation, the connection between the input ESN and the chaotic ESN is trained to output transient dynamics converging to 0 (see the Supplementary Materials for the detailed information about the transient dynamics). (**B**) Two experimental schemes. In the open-loop scheme, the symbolic input is externally given. On the other hand, in the closed-loop one, the symbolic input is autonomously generated by the additional feedback loop. In our method, we change the elements represented by red arrows to embed desired CI dynamics. (**C**) Outline diagrams of our batch learning methods composed of a three-step procedure. In step 1, the parameters of the network and readout are trained to output the quasi-attractors and the output dynamics corresponding to the symbols. In steps 2 and 3, the symbolic sequence is autonomously yielded. We prepare periodic symbol transition patterns as the target in step 2 and stochastic symbol transition rules in step 3.

embedded innate trajectory, the feedback discriminator achieves multiple symbol transition rules with minimum additional computational capacity (i.e., nonlinearity and memory).

Step 3. Regulate the feedback unit added in step 2 to design designated stochastic symbol transition rules. The deterministic system is expected to imitate the stochastic process by using intrinsic chaoticity. The system repeatedly generates the quasi-attractors embedded in step 1 in synchronization with the pseudo-stochastic symbol transition, meaning that the design of the desired CI dynamics is completed.

In this study, we demonstrate that the trajectories of quasi-attractors and their transition rules can be designed using the three steps described above. In step 1, we show that the desired output dynamics can be designed with high operability by using the embedded internal dynamics reproducibly generated after the innate training. Next, in step 2, we demonstrate that various types of periodic symbolic sequences switching at a certain interval can be implemented simply by adjusting the parameters of a feedback loop attached to the system. Last, in step 3, we prepare several stochastic symbol transition rules governed by a finite state machine and show that the system can simulate these stochastic dynamics by making use of the system's chaoticity. We also discuss the proposed method's validity and adaptability through several numerical experiments.

MATERIALS AND METHODS

System architecture

In our method, we aimed to embed M types of quasi-attractors and the transition rules among them in an RNN. We prepared M discrete symbols $s \in S$ ($S := \{s_1, s_2, \cdots s_M\}$). Each symbol corresponds to each quasi-attractor. We used an ESN as a high-dimensional chaotic system. As shown in Fig. 1A, we prepared an RNN composed of a nonchaotic input ESN (N^{in} nodes) working as an input transient generator as well as a chaotic ESN (N^{ch} nodes) yielding chaotic dynamics. The dynamics of input ESN $\boldsymbol{x}^{in}(t) \in \mathbb{R}^{Nin}$ and chaotic ESN $\boldsymbol{x}^{ch}(t) \in \mathbb{R}^{Nch}$ are given as the following differential equations

$$\tau \frac{d\boldsymbol{x}^{in}}{dt}(t) = -\boldsymbol{x}^{in}(t) + \tanh\left(g^{in} J^{in} \boldsymbol{x}^{in}(t) + \boldsymbol{u}^{in}(s(t))\right) \tag{1}$$

$$\tau \frac{d\boldsymbol{x}^{ch}}{dt}(t) = -\boldsymbol{x}^{ch}(t) + \tanh\left(g^{ch} J^{ch} \boldsymbol{x}^{ch}(t) + J^{ic} \boldsymbol{x}^{in}(t)\right) \tag{2}$$

where $\tau \in \mathbb{R}$ is a time constant, tanh is an element-wise hyperbolic tangent, g^{in} and $g^{ch} \in \mathbb{R}$ are scaling parameters, $u^{in}(s) \in \mathbb{R}^{Nin}$ is discrete input projected onto input ESN when symbol s is given, $J^{in} \in \mathbb{R}^{Nin \times Nin}$ and $J^{ch} \in \mathbb{R}^{Nch \times Nch}$ are connection matrices, and $J^{ic} \in \mathbb{R}^{Nch \times Nin}$ is a feed-forward connection matrix between input ESN and chaotic ESN. Each element of J^{in} is sampled from a normal distribution $\mathcal{N}\left(0, \frac{1}{N^{in}}\right)$. J^{ch} is a random sparse matrix with density $p = 0.1$ whose elements are also sampled from a normal distribution $\mathcal{N}\left(0, \frac{1}{pN^{ch}}\right)$. We used $\tau = 10.0$, $g^{in} = 0.9$, and $g^{ch} = 1.5$ to make input ESN nonchaotic and chaotic ESN chaotic (*34*). In addition, to prevent chaotic ESN from becoming nonchaotic because of the bifurcation caused by the strong bias term, we tuned J^{ic} before hand to project transient dynamics converging to 0 onto the chaotic ESN when the same symbolic input continues to be given (see the Supplementary Materials for detailed information about the transient dynamics). In any case, the whole RNN dynamics $\boldsymbol{x}(t) \in \mathbb{R}^{Nin + Nch}$ concatenating Eqs. 1 and 2 can be represented by the following single equation ($\odot$ represents an elementwise product)

$$\tau \frac{d\boldsymbol{x}}{dt}(t) = -\boldsymbol{x}(t) + \tanh\left(\boldsymbol{g} \odot (J\boldsymbol{x}(t)) + \boldsymbol{u}(s(t))\right) \tag{3}$$

where $\boldsymbol{x}$, $\boldsymbol{g}$, J, and $\boldsymbol{u}$ are defined by the following equations

$$\boldsymbol{x}(t) := [\boldsymbol{x}^{in}(t); \boldsymbol{x}^{ch}(t)] \tag{4}$$

$$\boldsymbol{g} := [\underbrace{g^{in}, \cdots g^{in}}_{N^{in}} \underbrace{g^{ch}, \cdots g^{ch}}_{N^{ch}}]^T \tag{5}$$

$$J := \begin{bmatrix} J^{in} & 0 \\ J^{ic} & J^{ch} \end{bmatrix} \tag{6}$$

$$\boldsymbol{u}(s) := [\boldsymbol{u}^{\text{in}}(s);\ 0] \tag{7}$$

The output dynamics are calculated by the linear transformation of the internal dynamics $\boldsymbol{x}(t)$, that is, the linear readout $w_{\text{out}} \in \mathbb{R}^{N\text{in} + N\text{ch}}$ is trained to approximate the following target dynamics $f_{\text{out}}(t)$

$$\boldsymbol{w}_{\text{out}}^T \boldsymbol{x}(t) \approx f_{\text{out}}(t) \tag{8}$$

The symbolic dynamics $s(t)$ itself, which is externally given in step 1, is lastly generated autonomously with a closed-loop system [Fig. 1B(2)]. In the feedback loop, the following classifier $f_{\max} : \mathbb{R}^{N\text{in} + N\text{ch}} \to S$ is attached

$$f_{\max}(\boldsymbol{x}(t)) := \underset{s \in S}{\arg\max}\, \boldsymbol{w}_s^T \boldsymbol{x}(t) \tag{9}$$

where $\boldsymbol{w}_s \in \mathbb{R}^{(N\text{in} + N\text{ch}) \times M}$ represents the connection matrix whose elements are trained to autonomously emulate the designated symbolic dynamics $s(t)$ [i.e., $s(t + \Delta t) \approx f_{\max}(\boldsymbol{x}(t))$, where $s(t + \Delta t)$ is the symbolic input for the next time step and Δt is a time width for discrete temporal evolution]. To summarize, we designed the desired quasi-attractors, output dynamics, and symbolic dynamics by tuning the parameters of the RNN connections J, the readout $\boldsymbol{w}_{\text{out}}$, and the classifier $\boldsymbol{w}_s$, respectively.

First order–reduced and controlled-error learning and innate training

We used two reservoir computing techniques called first order–reduced and controlled-error (FORCE) learning (*35*) and innate training (*36*). Both FORCE learning and innate training are methods that harness the chaoticity of the system. Below, we briefly describe the algorithms of both FORCE learning and innate training.

FORCE learning is a method that embeds designated dynamics in a system by harnessing the chaoticity of dynamical systems. Suppose the following ESN dynamics with a single feedback loop

$$\tau \frac{d\boldsymbol{x}}{dt}(t) = -\boldsymbol{x}(t) + \tanh\left(\text{g}J\boldsymbol{x}(t) + \boldsymbol{u}z(t)\right) \tag{10}$$

$$z(t) = \boldsymbol{w}^T \boldsymbol{x}(t) \tag{11}$$

where $\boldsymbol{u}$ represents the linear feedback vector. Typically, the scaling parameter g is set to be greater than 1 to make the whole system chaotic (*34*). In FORCE learning, to embed the target dynamics $f(t)$ in the system, $\boldsymbol{w}$ is trained to optimize the following cost function C_{FORCE}

$$C_{\text{FORCE}} := \langle \| z(t) - f(t) \| \rangle^2 \tag{12}$$

Here, the bracket denotes the averaged value over several samples and trials. In particular, in the FORCE learning, $\boldsymbol{w}$ is optimized online with a least-square error algorithm. It was reported from numerical experiments using ESN that better training performance was obtained when the initial RNN was in a chaotic regime (*35*).

Innate training is also a scheme for harnessing chaotic dynamics and is accomplished by modifying the internal connection J using FORCE learning. The novel aspect of innate training is that the inner connection of ESN is trained in a semisupervised manner, that is, the connection matrix J of the ESN is modified to minimize the following cost function C_{innate} to reproduce the chaotic dynamics yielded by the initial chaotic RNN ($\boldsymbol{x}_{\text{target}}(t)$, innate trajectory)

$$C_{\text{innate}} := \langle \| \boldsymbol{x}(t) - \boldsymbol{x}_{\text{target}}(t) \| \rangle^2 \tag{13}$$

Intriguingly, the innate trajectory is reproducibly generated for a certain period with the input while maintaining the chaoticity after the training. In other words, innate training is a method that allows a chaotic system to reproducibly yield the innate trajectory with complicated spatiotemporal patterns. In addition, innate training applies the FORCE learning method to the modifications of the internal connection, that is, the presynaptic connection of a node in the network is considered as the linear weight from the other nodes and trained by FORCE learning. In this study, we propose a method of designing CI by using both FORCE learning and innate training techniques.

Recipe for designing CI

Our proposed method is a batch-learning scheme consisting of the following three-step process (Fig. 1C).

Step 1. Designing quasi-attractor

In step 1, the connection matrix J^{ch} of the chaotic ESN is adjusted by innate training to design the trajectories of quasi-attractors. First, the target trajectories $\boldsymbol{x}^s_{\text{target}}(t)$ are recorded for M symbols under an initial connection matrix J^{init} and some initial states $\boldsymbol{x}^s_{\text{target}}(0)$, where $\boldsymbol{x}^s_{\text{target}}(t)$ denotes chaotic dynamics when the symbol is switched to s at $t = 0$ ms (for simplification, the switching time is fixed to $t = 0$ ms in step 1; note that the symbol can be switched at any time). In step 1, J^{ch} is trained to optimize the following cost function $C_{1-\text{in}}$

$$C_{1-\text{in}} := \sum_{s \in S} \sum_{t=0}^{L_{\text{innate}}} \| \boldsymbol{x}^s(t) - \boldsymbol{x}^s_{\text{target}}(t) \|^2 \tag{14}$$

Here, $\boldsymbol{x}^s(t)$ represents the dynamics when the symbol is switched to s at $t = 0$ ms, and L_{innate} represents the time period of the target trajectory. We randomly choose half the network nodes ($N^{\text{ch}}/2$ nodes) and modify their presynaptic connections to reduce the redundancy of the training parameter. The selected elements in connection matrix J^{ch} are trained for 200 epochs for each s. We lastly use J^{ch} recording the minimum $C_{1-\text{in}}$ (see the Supplementary Materials for the detailed algorithm used in step 1). After the innate training in step 1, the system is expected to reproduce the recorded innate trajectories $\boldsymbol{x}^s_{\text{target}}$ for L_{innate}.

Although there are no specific criteria for determining the initial states of the multiple innate trajectories, the large distances among $\boldsymbol{x}^s_{\text{target}}(0)$ are preferred since the temporal pattern of quasi-attractors is likely to differ, enhancing the separability. Moreover, the important trick of the innate training lies in its semisupervised scheme, that is, the training stability increases by guiding the offset states during training to the neighborhood of $\boldsymbol{x}^s_{\text{target}}(0)$. Although a scheme for training the internal connections of the RNN has already been proposed in FORCE learning (*35*), the tuning of all connections to generate the same function is mostly unstable (*37*). Therefore, we randomly selected the offset state of the innate trajectory $\boldsymbol{x}^s_{\text{target}}(0)$ on the phase space.

Similarly, $\boldsymbol{w}_{\text{out}}$ is trained to produce designated output dynamics $f^s(t)$ corresponding to symbol s. The following cost function $C_{1-\text{out}}$ is optimized

$$C_{1-\text{out}} := \sum_{s \in S} \sum_{0}^{L_{\text{out}}} \| f^s(t) - \boldsymbol{w}_{\text{out}}^T \boldsymbol{x}^s(t) \|^2 \tag{15}$$

High-dimensional nonlinear dynamical systems generally have high separability for input information, that is, it becomes easier for a linear model to solve nonlinear input-output function tasks by projecting input information into the system (38). In particular, the innate trajectories of chaotic systems are known to have such high expressive capability that various orbits can be designed simply by adjustment of the attached linear model (36). In this study, the tuned readout is also expected to stably reproduce the prepared trajectory by exploiting the high dimensionality and nonlinearity of the innate trajectories. Here, note that L_{innate} does not always match L_{out}, that is, L_{out} can be greater than L_{innate}. The training is accomplished by an offline algorithm Ridge regression based on the recorded internal dynamics $\boldsymbol{x}^s(t)$.

Step 2. Embedding autonomous transitions of symbol

In step 2, we tune a feedback loop $f_{\max}$ to achieve the autonomous symbol transition. We especially prepare target periodic transition rules switching every T (ms). Suppose a target periodic symbolic time series $s_{\text{per}}(t)$. First, the network dynamics $\boldsymbol{x}(t)$ of the open-loop setup [Fig. 1B(1)] is recorded with a symbolic dynamics $s_{\text{per}}(t)$ for $T_{\text{rec}} := 500{,}000$ ms. On the basis of the recorded dataset, $f_{\max}$ is tuned to output the next symbolic input $s_{\text{per}}(t + \Delta t)$ from $\boldsymbol{x}(t)$. The parameters $\boldsymbol{w}_s$ of $f_{\max}$ is trained to optimize the following cost function C_2

$$C_2 := -\sum_{s\in S}\sum_{0}^{T_{\text{rec}}} 1\{s_{\text{per}}(t+\Delta t)=s\}\log\frac{e^{\boldsymbol{w}_s^T\boldsymbol{x}(t)}}{\sum_{k\in S}e^{\boldsymbol{w}_k^T\boldsymbol{x}(t)}} \tag{16}$$

As the optimization algorithm, we use the limited-memory Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm (39).

This optimization task is considered to be a type of timer task, a commonly used benchmark task to evaluate the temporal computational capacity, where the readout is trained to output a pulse-like wave with a certain delay after input is given. By projecting the input signal into a high-dimensional nonlinear dynamical system, the timer task can be achieved simply by adjusting the linear readout. In addition, the complex trajectory embedded by the innate training significantly increases the performance of the timer task compared with a nonchaotic random ESN (36). In our model, the tuned classifier $f_{\max}$ is expected to emulate the delayed symbolic switching by exploiting the embedded innate trajectory.

Step 3. Embedding stochastic transitions of symbol

In step 3, we implement a stochastic transition rule governed by a finite state machine by modifying a feedback loop $f_{\max}$. As discussed in Introduction, the chaoticity of the system is expected to be used to emulate the stochastic process in the deterministic setup. We prepared the target stochastic time series $s_{\text{sto}}(t)$ generated from a Markov model with certain switching periods. The process of the learning is same as that in step 2, that is, the pair of $(\boldsymbol{x}(t), s_{\text{sto}}(t))$ recorded in the open-loop setup for 500,000 ms is used to train the $f_{\max}$ to emulate $s_{\text{sto}}(t)$. Here, we use the following cost function C_3 in the training

$$C_3 := -\sum_{s\in S}\sum_{0}^{T_{\text{rec}}} 1\{s_{\text{sto}}(t+\Delta t)=s\}\log\frac{e^{\boldsymbol{w}_s^T\boldsymbol{x}(t)}}{\sum_{k\in S}e^{\boldsymbol{w}_k^T\boldsymbol{x}(t)}} \tag{17}$$

As with the optimization of the cost function C_2, C_3 is optimized with the limited-memory BFGS algorithm.

Note that the formulation of C_3 is the same as that of C_2, that is, the properties of target dynamics to be embedded are not expressed in the cost function formulation. Rather, both optimization processes in steps 2 and 3 are data-driven, meaning that the properties of prepared target symbolic sequences determine whether the embedded dynamics are required to be chaotic or nonchaotic.

RESULTS

In this section, we show the demonstration and analytic results of the numerical experiments for each step.

Step 1. Designing quasi-attractor

As discussed in the previous section, the internal connection of the chaotic ESN J^{ch} is trained to reproducibly output the corresponding innate trajectories to the symbolic switching. Figure 2A demonstrates the change of the network dynamics of a 1500-node RNN ($N^{\text{in}} = 500$, $N^{\text{ch}} = 1000$) whose connection matrix is modified with innate training under the condition $(M, L_{\text{innate}}) = (1{,}1000)$. The trajectory quickly spreads before $t = L_{\text{innate}}$ in the pretrained system, whereas the target trajectory $\boldsymbol{x}_s^{\text{target}}$ (dotted line) is reproducibly yielded for 1000 ms (covered by the yellow rectangle) in the posttrained system. Moreover, intriguingly, the dispersion of the trajectories continues to be suppressed even after $t = L_{\text{innate}}$.

Next, Fig. 2B displays both the network dynamics and the output dynamics. The 1500-node RNN ($N^{\text{in}} = 500$, $N^{\text{ch}} = 1000$) trained under the condition $(M, L_{\text{innate}}) = (3{,}1000)$ was used. At first, the symbolic input was absent, and then symbols were switched with random intervals from the middle. In addition, the two-dimensional readout was trained to output the Lissajous curve for symbol A, the "at" sign for symbol B, and the *xz* coordinates of the Lorenz attractor for symbol C for $L_{\text{out}} = 1500$ ms (the target trajectory for the "at" sign was made from a centerline of font data). It was observed that the desired spatiotemporal patterns were stably and reproducibly generated for a certain period in every trajectory with different initial states after the symbol transition (see movie S1). Note that the same linear model $\boldsymbol{w}_{\text{out}}$ was used in the demonstration, implying that the trajectory of each quasi-attractor has rich enough information to independently output the designated time-series patterns even with the single linear regressor. Our scheme for designing transient dynamics would be highly useful in the field of robotics because the process in step 1 is easily achieved by adjusting the partial elements of a high-dimensional chaotic system. For example, the system working in a real-world environment should immediately and adaptively switch its motion according to the change of environmental input like a system developed by Ijspeert *et al.* (40), which can be easily accomplished by our computationally cheap method. In this way, our method would work effectively in the context of robotics, where fast responsiveness and adaptability are required.

We also examined both the scalability and the validity of innate training in detail through several numerical experiments (Fig. 3). First, we examined the relationship between the number of input symbols M and the accuracy of innate training. To evaluate the performance of innate training, we used the normalized mean square error (NMSE) between the output and the innate trajectory $\boldsymbol{x}^s_{\text{target}}$ represented by the following formula

$$\text{NMSE} := \frac{1}{M}\sum_{s\in S}\left\langle\frac{\sum_0^{L_{\text{innate}}}\|\boldsymbol{x}^s(t)-\boldsymbol{x}^s_{\text{target}}(t)\|^2}{\sum_0^{L_{\text{innate}}}\|\boldsymbol{x}^s_{\text{target}}(t)\|^2}\right\rangle \tag{18}$$

where the bracket represents the average over 10 trials for each symbol. We calculated the NMSE for 10 trials. Figure 3A shows the innate training performances with the different training conditions, suggesting that NMSEs are more likely to increase with a longer target trajectory and a larger number of symbols. This result implies that innate training has its limitation in the design of the quasi-attractors.

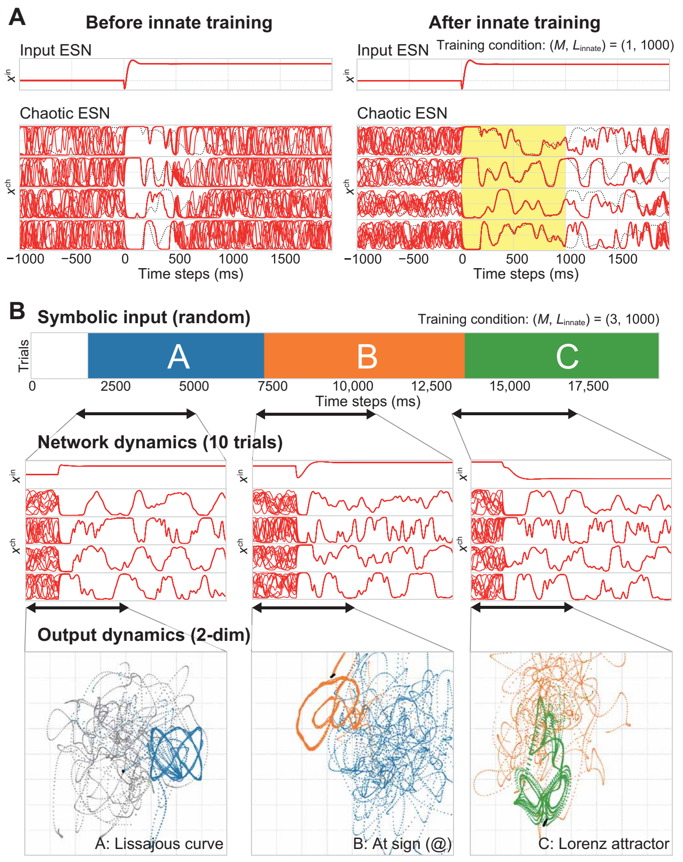

Fig. 2. Demonstration of step 1. (**A**) The dynamics of the reservoir before and after the innate training. In the figure, we show the RNN dynamics trained under the condition $(M, L_{\mathrm{innate}}) = (1{,}1000)$. The time-series data of a selected node in the input ESN are shown in the top column. Conversely, the four selected dynamics of the chaotic ESN are displayed in the bottom four columns. In each column, both the innate trajectory (black dotted) and 10 individual trajectories with different initial conditions (red) are exhibited. (**B**) Demonstration of open-loop dynamics. The network dynamics of the RNN trained under the condition $(M, L_{\mathrm{in}}, L_{\mathrm{out}}) = (3{,}1000{,}1500)$ is used in this demonstration. Both the network dynamics and output dynamics of the trained readout are depicted. The readout is trained to output the Lissajous curve for symbol A, the "at" sign for symbol B, and the xz coordinates of the Lorenz attractor for symbol C. Note that the intervals of the symbolic input were randomly decided.

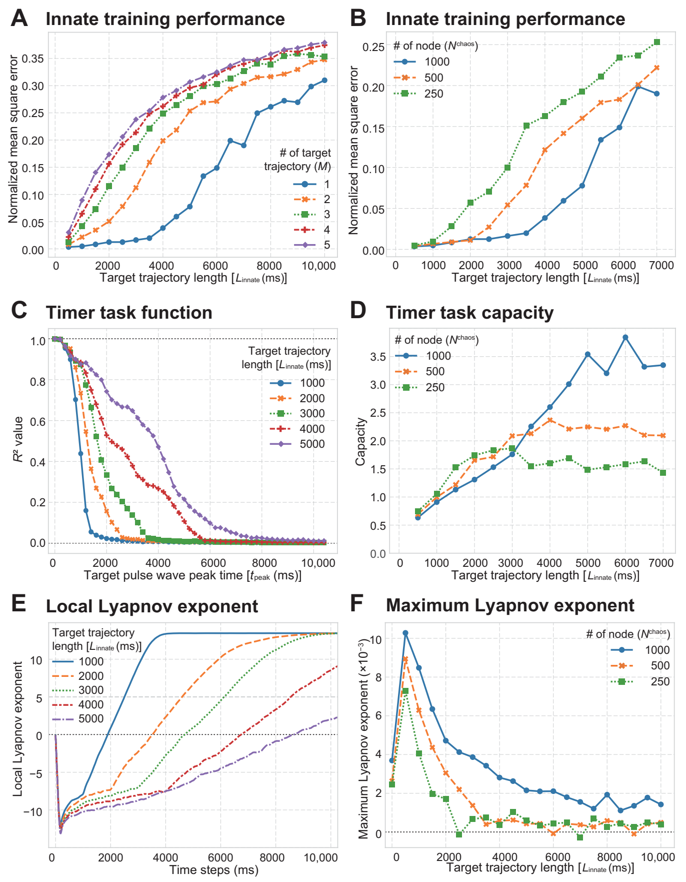

Fig. 3. Scalability and the validity of innate training used in step 1. (**A**) Performance of innate training over M symbols. The normalized mean square errors are calculated from the 10 trials. (**B**) Effect of network size N^{ch} on the performance of innate training. (**C**) Evaluation of the temporal information capacity with timer task. The averaged values for 10 trials are plotted. (**D**) Effect of the system size on timer task capacities. Timer task capacity is defined as the integral value of the timer task function. (**E**) Evaluation of the LLE. The LLE is measured with the time development of the perturbation of the chaotic ESN (see the Supplementary Materials for detailed information about the calculation method of the LLE). (**F**) Evaluation of the system's MLE.

We also examined the effect of network size on the capability to embed the quasi-attractors. We investigated the relationship between the number of nodes in the chaotic ESN N_c and the accuracy of innate training under the condition $M = 1$ (Fig. 2B), suggesting that the NMSEs were less likely to increase with a larger network. To summarize, our analysis indicates that longer trajectories can be embedded in a larger network by innate training.

Next, we evaluated the effect of innate training on the capacity of the system's information processing. We prepared a timer task and measured how long the inputted information was stored in the RNN. In the timer task, the pulse-like wave with a peak t_{peak} (ms) after the symbol transition was prepared as the target, and the performance was defined as the accuracy of the pulse-like wave reconstruction by a trained readout. Here, we defined the coefficient of determination value R^2 between the output and the pulse-like wave as the timer task function $R^2(t_{\mathrm{peak}})$. At the same time, we also calculated the integral value of the timer task function $\int_0^\infty R^2(t)\,\mathrm{d}t$ and define it as the timer task capacity (see the Supplementary Materials for detailed information about the setup of the timer task). Figure 4C shows the timer task function with different innate training conditions, indicating that RNNs trained with the longer-length target trajectory L_{innate} perform better. It was also observed that the timer task capacity saturated around L_{innate} = 5000 ms in the 1000-node RNN, and the border of the saturation decreased in a smaller system (Fig. 3D). These results imply that the temporal information capacity of the system is improved by innate training with the longer target length L_{innate} but saturates at a certain value, which is determined by the system size.

Furthermore, we assessed the effect of innate training on the system's chaoticity by measuring the Lyapunov exponents of the system. Since the transition among quasi-attractors is driven by the system's chaoticity, it is necessary to keep the system chaotic. In this experiment, we measured the local Lyapunov exponent (LLE) to evaluate the degree of trajectory variation after the symbolic switching.

We also measured the maximum Lyapunov exponent (MLE) without any inputs ($\boldsymbol{u}(t) = 0$) to estimate the global chaoticity of the system (see the Supplementary Materials for the detailed calculation algorithm of both the LLE and MLE). Figure 3E displays the LLE values of the systems with the different target trajectory length L_{innate}, suggesting that the trajectories unevenly expand after the symbol transition. In particular, it was observed from the LLE analysis that contracting regions existed (regions with negative LLEs corresponding to the lengths of the quasi-attractors) caused by the transient dynamics projected by the input ESN, and the degree of the expansion became gradual in the trained period $t \in [0, L_{\text{innate}})$. These results imply that innate training yields a locally contractive phase space structure, that is, a quasi-attractor. Moreover, positive MLE values were constantly obtained from the MLE analysis depicted in Fig. 3F, supporting the conjecture that the system chaoticity was maintained especially well with the larger RNNs even after the innate training. (Note that a sharp increase in MLE was observed with shorter L_{innate}, which is caused by the increase in the spectral radius of the connection matrix J of the system. See the Supplementary Materials for detailed information of the analysis.)

Step 2. Periodic symbol transition

In step 2, the system autonomously generates a symbolic sequence externally given in step 1. The additional feedback loop realizes the autonomous periodic switching of the symbols. We demonstrate that various types of periodic symbolic sequences switching at a fixed interval can be easily designed simply by tuning the parameter of the feedback loop $f_{\max}$. Figure 4A demonstrates the embedding of the periodic symbolic sequence A-B-C (2000-ms interval and 6000-ms period) with a trained RNN ($(M, L_{\text{innate}}) = (3{,}1000)$). Figure 4B also exhibits the embedding of the periodic symbolic sequence A-B-C-D-E-F-G-H-I-J (500-ms interval and 5000-ms period), with the same RNN used as the demonstration in Fig. 4A (note that $f_{\max}$ was changed from one in Fig. 4A). In both demonstrations, the system succeeded not only in generating the desired symbol transition rules but also in stably outputting the designated output dynamics with high accuracy.

We also show that the system can solve tasks requiring higher-order memory in the same scheme. We prepared the two periodic symbolic sequences A-B-C-B and A-B-C-B-A and separately trained $f_{\max}$. These two symbolic sequences are more difficult to embed because the system must change the output according to the previous output. In the symbol transition A-B-C-B, for example, the system must output the next symbol depending on the previous symbol when switching from B, though the total number of symbols is the same as in the task A-B-C. We used the same RNN and setup used in the Fig. 4A and only changed the parameters in $f_{\max}$ to realize the symbol transitions. Figure 4C displays the network dynamics and symbol transition of the two tasks, showing that the system successfully achieves both the periodic sequence A-B-C-B with an 8000-ms period and A-B-A-B-C with a 10,000-ms period. These results suggest that the trained RNN had the higher-order memory capacity, that is, the generated trajectories have sufficient separability to distinguish the contextual situation depending on the previous symbol sequence (see movie S2). In robotics, periodic motion control has often been implemented by an additional oscillator (e.g., a central pattern generator) to yield limit cycles (*23*, *40*–*42*). Our method in step 2 would be useful in designing limit cycles with longer periods and more complicated patterns. The analysis in fig. S2 shows that our method outperforms FORCE learning in the embedding of a long-term periodic attractor including multiple transitions in order (see the Supplementary Materials and fig. S2).

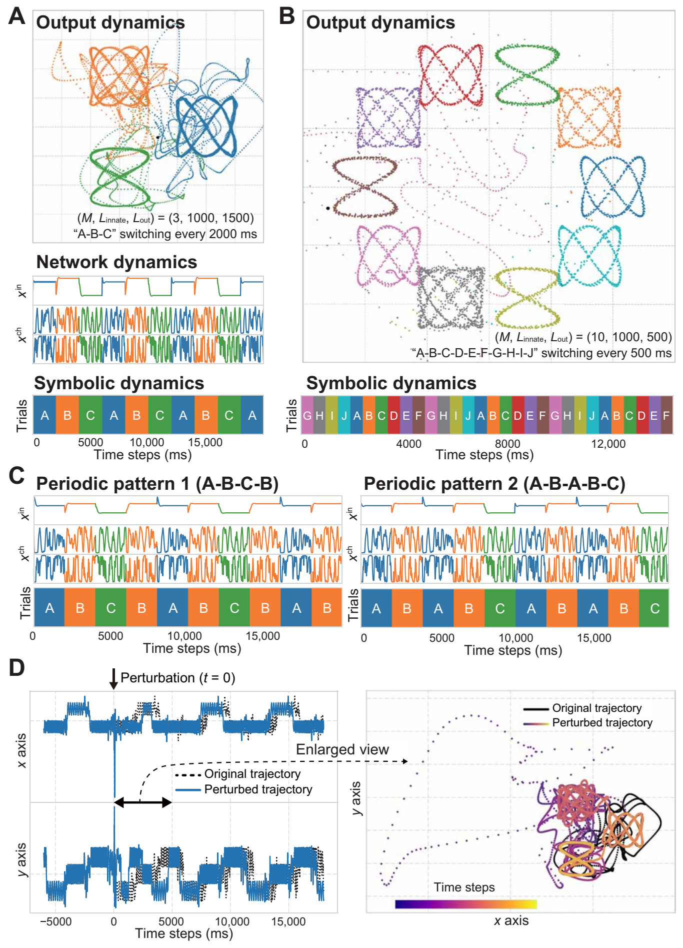

Fig. 4. Demonstrations of closed-loop dynamics in step 2. (**A**) Three-symbol periodic transition. We prepared an RNN trained under the condition $(M, L_{\text{innate}}) = (3{,}1000)$ and a readout trained under the condition $L_{\text{out}} = 1500$ ms to output three Lissajous curves corresponding to the symbolic input. The feedback loop $f_{\max}$ realizes the periodic symbol transition A-B-C switching at 2000-ms intervals. (**B**) Ten-symbol periodic transition. We prepared an RNN trained under the condition $(M, L_{\text{innate}}) = (10{,}500)$ and a readout trained under the condition $L_{\text{out}} = 500$ ms to output 10 different Lissajous curves corresponding to the symbolic input. The feedback loop $f_{\max}$ achieves the periodic symbol transition A-B-C-D-E-F-G-H-I-J switching at 500-ms intervals. (**C**) Demonstration of the tasks requiring higher-order memory to be solved. The same RNN was used in the demonstration of (A). The left panel displays the periodic symbol transition pattern A-B-C-B switching at 2000-ms intervals. The right one demonstrates the periodic symbol transition pattern A-B-A-B-C switching at 2000-ms intervals. These tasks were accomplished in the same way in the demonstrations of (A) and (B), that is, only the parameters in $f_{\max}$ were tuned. (**D**) Two output dynamics: original trajectory and perturbed trajectory. A small perturbation was given to the original trajectory at $t = 0$ ms.

We also analyzed the effect of perturbation to investigate the stability of the embedded symbol transition. Figure 4D shows the output dynamics of both the original and perturbed trajectories, clarifying that the trajectory returned to the original one after the addition of the perturbation. We also calculated the MLE values of the system and obtained the value -1.89×10^{-4}, which was very close to zero. These analyses indicate that the trained feedback loop $f_{\max}$ made the system nonchaotic, that is, the generated internal dynamics was a limit cycle.

Step 3. Stochastic symbol transition (CI)

In step 1, we constructed the trajectories of the quasi-attractors and the corresponding output dynamics. In step 2, we showed that periodic transitions among quasi-attractors can be freely designed by simply tuning the feedback loop $f_{\max}$. In step 3, we realize a stochastic transition, that is, CI. As discussed above, the system is expected to use its chaoticity to emulate a stochastic transition in deterministic dynamical systems.

First, we demonstrate that stochastic transition can be freely designed by adjusting $f_{\max}$ (see Fig. 5A and movie S3). In this demonstration, we used the same RNN as in Fig. 4A. We prepared a symbol transition rule uniformly switching among symbols A, B, and C at 3000-ms intervals. Figure 5B shows the symbolic dynamics, network dynamics, and output dynamics, suggesting that the symbol transitions started to spread at around t = 10,000 ms and lastly settle down to completely different transition patterns. Nevertheless, the system continued to stably generate Lissajous curves. These demonstrations imply that the system constantly reproduced quasi-attractors embedded by innate training, while the quasi-stochastic transition was achieved by the global chaoticity. [Note that, although we demonstrated our approach by embedding typical stochastic processes (i.e., Markov processes) to illustrate the usability of our scheme, our method can also design a history-dependent stochastic rule that cannot be represented by a Markov model. See our demonstration represented in the Supplementary Materials and fig. S3.]

To analyze the flexibility of our method, we measured the stochastic transition matrix and the average symbolic intervals (Fig. 5B). We prepared two stochastic symbol transition rules as the targets: the transition rule governed by the uniform finite state machine (pattern 1) and the transition rule governed by the finite state machine with a limited transition (pattern 2). Note that we used the same trained RNN as in the demonstration in step 2 and embedded the transition rules simply by adjusting $f_{\max}$. Figure 5B shows the results of the obtained trajectories, implying that the system successfully embedded patterns similar to the target rules, although there were some errors and variations in the transition probability and the switching time. The positive MLEs were obtained in both cases ($+2.01 \times 10^{-3}$ in pattern 1 and $+1.71 \times 10^{-3}$ in pattern 2), suggesting that the system was weakly chaotic as a whole. In addition, we analyzed the history dependence of the transition in detail, showing that transition probabilities did not differ so much according to the past symbol in both cases though the preferred routes were observed in output dynamics (see the Supplementary Materials and fig. S4). In this sense, it can be said that the system successfully expressed the random transitions in a macroscopic scale (i.e., a scale in symbol transitions) using the chaoticity.

Last, we analyzed both the structures of the obtained chaotic attractors and the symbolic dynamics in detail. Figure 6A shows the effect of small perturbations on the symbolic dynamics, implying that the patterns of symbolic dynamics varied after a certain period. To analyze the structural change of the terminal symbolic state, we measured the symbolic dynamics accompanied by the temporal development of the set of initial states on a plane constructed by the two selected dimensions (Fig. 6B), clarifying that a complex terminal symbolic structure emerges after a certain period (Fig. 6B). In

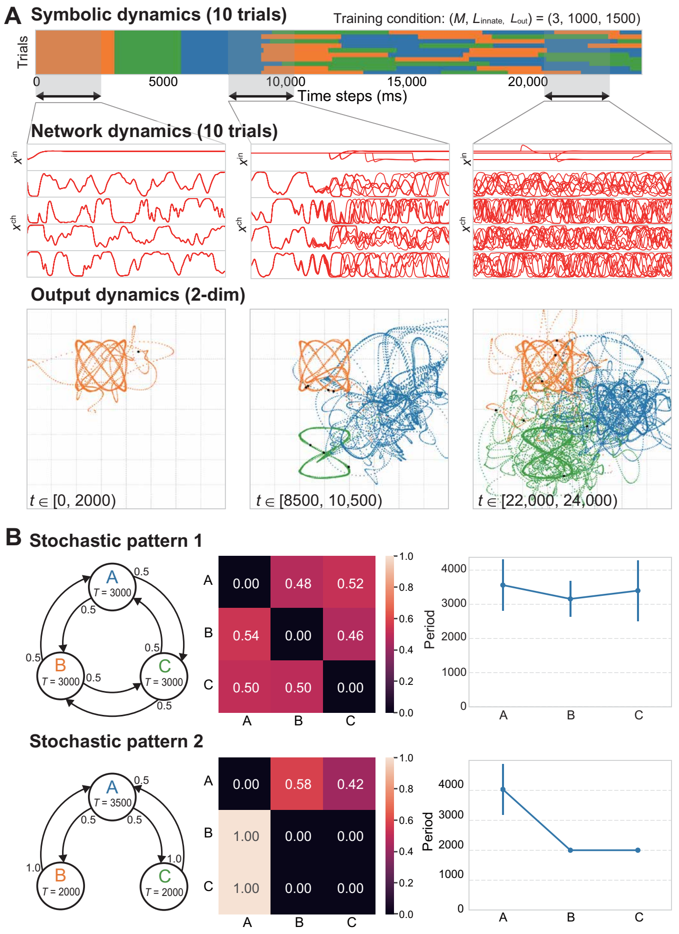

Fig. 5. Demonstration of step 3. (**A**) Network dynamics with $f_{\max}$ trained to imitate a stochastic transition rule. We used an RNN trained under the condition (M, L_{innate}) = (3,1000), and readout trained to output Lissajous curves under the condition L_{out} = 1500 ms. The feedback classifier $f_{\max}$ was trained to uniformly switch the symbol among the three symbols A, B, and C at 3000-ms intervals. Ten different trajectories with small perturbations are overwritten in the figure. (**B**) Evaluation of the embedding performance of a stochastic symbol transition. Two different stochastic symbol transition rules (patterns 1 and 2) were prepared as the target. The same RNN was used as in the demonstration of (A). The middle figures show the obtained probability density matrix, and the right ones show the average switching duration (the error bar represents SD).

particular, in the embedding of the pattern 1 rule, the entropy of the terminal symbolic pattern converges to a value close to the maximum entropy $\log_2 3^9 \approx 14.26$ (note that the entropy was measured on the basis of the probability distribution constructed by the frequency of 3×3 grid patterns). These results indicate that the symbol transition markedly changed even with a small perturbation and was unpredictable after a certain period, that is, the prediction of symbolic dynamics required the complete observation of the initial state value and calculation of the temporal development with infinite precision.

DISCUSSION

In this study, we proposed a method of designing CI based on reservoir computing techniques. We also showed that the various types of output

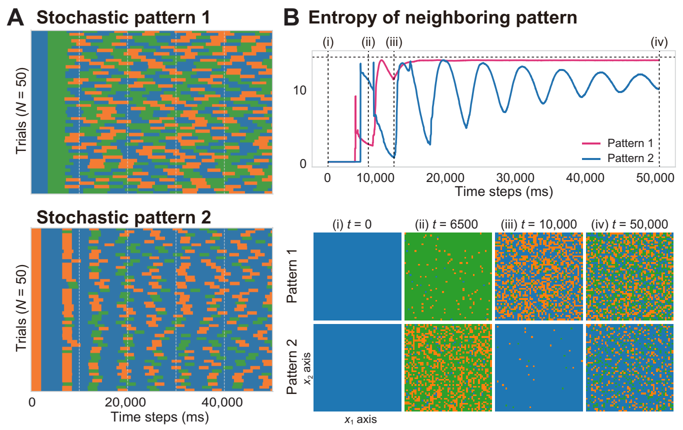

Fig. 6. Analysis of symbolic dynamics and the final state. (**A**) Effect of a small perturbation on the terminal symbolic dynamics. We evaluated the two closed-loop setups prepared in Fig. 5B. The figures display the symbolic dynamics generated by 50 trajectories with 50 different initial values. (**B**) Analysis of symbolic dynamics generated by the temporal development of the initial states on a small plane and its entropy of the symbolic pattern. Two dimensions (x_1 and x_2) on the phase space were selected from the chaotic ESN to construct the plane. We observed the symbolic dynamics generated by the temporal development of the states on the plane. To evaluate the randomness of the obtained pattern, we calculated the entropy of the obtained symbolic pattern based on the probability distribution constructed from the 3×3 grid patterns. Note that the horizontal dashed line shows the maximum entropy ($\log_2 3^9 \approx 14.26$).

dynamics and symbol transition rules could be designed with high operability simply by adjusting the partial parameters of a chaotic system with our three-step recipe. In this section, we first discuss the scalability of our method and the mechanism of how CIs are successfully embedded by reviewing several numerical analyses that verify the validity of our methods. Next, we discuss the effectiveness and significance of our method from multiple viewpoints.

Scalability and validity

First, the results of the innate training performances displayed in Fig. 3 (A and B) indicate that the number of RNN nodes constrains the total length of the quasi-attractors that can be embedded in the system by the innate training. However, the LLE analyses in Fig. 3E show that the system has the expanded region of the negative LLE even when the NMSE between the innate trajectory and the embedded trajectory becomes large (e.g., L_{innate} = 5000 ms). These results imply that, even when the innate trajectories are not successfully embedded in the system, the system stably yields high-dimensional trajectories with complicated spatiotemporal patterns for each symbol transition over L_{innate}, which is caused by the weakening of the system chaoticity. The same RNN trained under the condition (M, L_{innate}) = (3,1000) was repeatedly used in our series of demonstrations, the desired output dynamics (e.g., the Lissajous curves) being constantly generated for L_{out} = 1500-ms periods after the symbolic shift (Figs. 2, 4, and 5). We also demonstrated that the system can autonomously generate a symbol transition rule with an interval greater than L_{innate} (Figs. 4 and 5), suggesting that the system exploited high-dimensional reproducible trajectories longer than L_{innate}.

Moreover, it is assumed that the length of the quasi-attractors constrains the target stochastic transition rules that can be embedded. The system failed to imitate the stochastic transition, and the transition became periodic when the target transition had a shorter switching interval, whereas the training of f_{max} became unstable when it had a longer switching interval. These results suggest that the following two mechanisms should be required in the design of CI in our method: (i) The differences among the trajectories are sufficiently enlarged through the temporal development to realize the stochastic symbol transition and (ii) a similar spatiotemporal pattern should be reproducibly yielded until the switching moment to precisely discriminate the switching timing. These two mechanisms are contradictory, of course, and the desired CI can likely be embedded when both conditions are moderately satisfied.

Next, we discuss the validity of CI designed with our method through comparison with previous CI studies. We demonstrated that the embedded transition yields transition probability similar to the desired one by using the chaoticity of the network. Several CI works, however, pointed out that transition of CI is history dependent, and therefore, CI shows some preference in the transition determined by the history. Kaneko and Tsuda (*43*), for example, pointed out that the transition of CI has a specific order that is distinguished from a simple random hopping. Itoh and Kimoto (*44*) also reported that the transition presents a preferred route governed by the global phase space dynamics. Our experiment indicated that embedded CI also shows preferred trajectories in output dynamics according to the previous symbol before the switching (see the Supplementary Materials and fig. S4A). Therefore, the symbolic dynamics implemented with our methods should be history dependent.

Moreover, it should be noted that our proposed method can successfully emulate the randomness of the symbol transition despite the existence of the preferred trajectory. Our analysis revealed that the transition probabilities did not differ so much according to the difference in the previous symbol in both demonstrations (i.e., stochastic rules 1 and 2), implying that prediction of the next transition is still not easy if done only by referring to the macroscopic symbolic history (see the Supplementary Materials and fig. S4B). In this way, macroscopic symbol transitions still appear to be random, despite the trajectory preference. Thus, it can be said that our proposed method emulates the transition probability and its randomness on a high-dimensional nonlinear dynamical system by using the chaoticity of the system.

We show that our method can realize CI characterized by the random transition of a finite number of quasi-attractors as shown in (*19*, *21*). Some classes of CI, however, are difficult to design even with our method. There exists a CI whose transition frequency has a long-tailed distribution (*9*). In addition, CI can have infinitely many quasi-attractors whose dynamics should be described with an infinite state machine (*7*). These CI properties are hard to design with our method since an infinite number of auxiliary symbols and an infinite length of corresponding trajectories should be prepared, which should be solved in the future.

Last, we discuss the speed of recovery from the transient state, that is, the stability of the embedded quasi-attractor. Ahmadi and Tani (*45*), for example, demonstrated variational RNN called predictive coding-inspired variational RNN (PV-RNN) to imitate learning of stochastic transition between presented primitives, in which the stability of the attractor is determined by the coefficient of the complexity of variational evidence lower bound in Bayesian inference expressed in the cost function. Therefore, in their approach, the formulation of the cost function can explain the mechanism of how the recovery speed is determined. On this point, the stability of the quasi-attractor in our approach is considered to be determined by the spectral radius, a parameter of ESN, and the connection strength between the input ESN and the chaotic ESN. As discussed above, a transition from nonchaotic to chaotic regime occurs when the spectral radius of ESN exceeds 1.0. In addition, the chaoticity of the system is strengthened by the increased spectral radius. Hence, it is speculated that the recovery speed becomes lower as the spectral radius becomes larger since the quasi-attractor becomes more unstable. Besides, the network state is more likely to synchronize when the signals sent to the chaotic ESN are larger. Thus, the recovery speed would be also controlled by the connection strength governing the amplitude between the input ESN and the chaotic ESN. To summarize, the ESN spectral radius and the connection strength are considered to determine the recovery speed in our architecture.

Effectiveness and significance

First, the high operability of our proposed model would be helpful to understand the underlying mechanism of brain's information processing from a certain perspective. Previous studies have pointed out that an enormous number of nonlinear units and their interaction essentially constitute an animal's nervous system and yields highly complicated activities characterized by nonlinear phenomena such as chaos and CI. It has been reported that chaotic behavior appears in a wide range of brain activities from the cell level (e.g., action potential) to the global measurement level (e.g., electroencephalogram) (*46*). In addition, it has been pointed out that the collective neural activities not only encode the external information but also transform it according to the history of activities (*47*, *48*), suggesting that the animal brain realizes its information processing through the high-dimensional activities. We found that high-dimensional chaos has enough rich expressive capability to design CI, implying that the high-dimensional chaotic brain activities potentially have the capability to realize various functional hierarchies. In this sense, our model would provide a clue to understand the mechanism of how high-dimensional chaos that contributes to the information processing in animal brains.

The designing method for the CI exhibited in this study offers fundamentally different benefits compared with the previous methods, which simply exploit chaotic dynamics. The dynamics of CI presents an interesting property and suggest that local coherence and global chaoticity coexist. This property of CI would effectively work especially in designing cognitive models where both autonomy and spontaneity are required. For example, a designer can implement the motion primitives through the quasi-attractor while maintaining the autonomy of the robot through the global unpredictability. Furthermore, our algorithm can design the probability distribution, that is, the global tendency of behavior, as shown in Fig. 5. This coexistence is, however, difficult to express when using only the conventional chaotic attractors. In addition, several studies pointed out that animals' cognitive functions, such as memory recall and association, would be realized through the transition phenomenon among stereotypical activities (*13*, *14*), suggesting that CI might play an important role in animal cognition. In this sense, our method would be used for implementing cognitive models in a high-dimensional dynamical system.

Although several studies have used chaotic dynamical systems to embed desired trajectories (*35*, *36*), these conventional methods are incapable of combining multiple predetermined transient dynamics like our method. It might also be possible to merge multiple transients into one big attractor and embed it simultaneously by a learning scheme such as FORCE learning. However, the additional experiment shows that FORCE learning is more unstable than our methods when it comes to embedding long-term periodic trajectories consisting of multiple transients (see the Supplementary Materials and fig. S2). In this sense, high operability in our method is unavailable in the conventional methods.

Unlike previous methods that construct desired trajectories by tuning the entire dynamics by backpropagation algorithm, our method is accomplished by adjusting the reduced number of parameters and using the intrinsic high-dimensional chaos, which alleviates the biological implausibility and computational complexity of backpropagation algorithms. For example, recent physiological studies on the motor cortex (*49*, *50*) suggest that a large variety of behaviors can be instantaneously generated by the partial plasticity of the nervous system, supporting the biological plausibility of our learning scheme. In addition, our learning scheme is computationally cheaper than backpropagation since adjustments of entire neural circuits are not necessary. These properties would be especially helpful in the context of bioinspired robotics, where fast responsiveness and real-time processing are required.

Another advantage of our method is that it does not require the explicit structure of dynamical systems. For example, in the method proposed by Namikawa and Tani (*20*–*22*), the controller needs a fixed hierarchical structure and modularity. Therefore, the trained controller was specialized in implementing a specific behavior, making it difficult to divert it for any other purpose. In addition, it may be possible to design CI-like dynamics in an architecture where the symbolic sequence and the corresponding trajectory are separately generated, which requires an external mechanism to hold the symbol and wait until the generation of lower-order trajectory finishes. Therefore, the separation of the symbolic sequence model and trajectory encoder implicitly uses the hierarchical structure and cannot be realized by a high-dimensional chaos alone. In contrast, we proposed a method of designing CI with a setup consisting of a single chaotic ESN, auxiliary symbols, and an interface between them (input ESN) with high scalability. Moreover, the modifications of internal connections in the chaotic ESN can be realized by adding multiple linear feedback loops and training them with the FORCE learning since the presynaptic connection in the chaotic ESN can be regarded as a linear connection. Thus, our method allows us to design the various trajectories and their transition rules in a consistent high-dimensional chaotic system, thereby greatly expanding the scope of application of high-dimensional chaotic dynamical systems. Neuromorphic devices based on physical reservoir computing frameworks would be an excellent candidate for implementing our scheme (*31*). Sprintronics devices, for example, have recently been shown to exhibit chaotic dynamics (*51*, *52*) and are actively exploited as physical reservoirs (*53*–*55*). We expect that this framework would provide one of the promising application scenarios for real-world implementations of our scheme.

Our method can use a priori knowledge through the introduction of auxiliary symbols. Several neurorobotics frameworks have been proposed so far in which symbolic dynamics are self-organized on the network by end-to-end learning (*26*, *56*). Although these methods are convenient since they do not require explicit a priori knowledge, they cannot actively use the prior symbolic structure, and thus, symbolic structure only appears after the training. In contrast, we showed that the learning performance is greatly improved by auxiliary symbols (see the Supplementary Materials and fig. S3). In that sense, our method has an advantage over the conventional end-to-end scheme.

Our method is also scalable to autonomous symbol generation required in more advanced functionality. For example, in our method, M kinds of auxiliary symbols are given as a priori knowledge. However, in a highly autonomous system, such as humans, symbols are dynamically generated and destroyed because of developmental processes. As demonstrated by Kuniyoshi and Sangawa (*17*), these self-organizing symbolic dynamics can be realized by providing an additional automatic labeling mechanism in the system. In addition, unsupervised algorithms for extracting discrete symbols from the dynamics like the one introduced in (*57*–*59*) can be incorporated into our system. In other words, it is possible to spontaneously generate symbols by embedding an unsupervised learning algorithm in the system; this is a subject for future work.

Last, the dynamic phenomena obtained by our method are significant from the viewpoint of high-dimensional dynamical systems. As shown in Fig. 6, we demonstrated that small differences in the initial network state were expanded by the chaoticity of the system, which eventually led to drastic change in both the global symbol transition pattern $s(t)$ and the local dynamics $\boldsymbol{x}(t)$. Such tight interaction between microlayer and macrolayer is a phenomenon unique to deterministic dynamical systems; that is, it cannot occur, in principle, in a system where the higher-order mechanism is completely separated from the lower-order one (e.g., independent random variables). In addition, the global characteristics of dynamical systems are often analyzed by the mean-field theory. However, the analysis by the mean-field approximation cannot capture the contribution of microscopic dynamics to the macroscopic change. Thus, our CI design method has a meaningful role in shedding light on the interaction between micro- and macrodynamics in deterministic chaotic dynamical systems.

SUPPLEMENTARY MATERIALS

Supplementary material for this article is available at http://advances.sciencemag.org/cgi/content/full/6/46/eabb3989/DC1

REFERENCES AND NOTES

1. R. Pfeifer, C. Scheier, *Understanding Intelligence* (MIT Press, 2001).
2. T. Flash, B. Hochner, Motor primitives in vertebrates and invertebrates. *Curr. Opin. Neurobiol.* **15**, 660–666 (2005).
3. R. D. Beer, A dynamical systems perspective on agent-environment interaction. *Artif. Intell.* **72**, 173–215 (1995).
4. H. Jaeger, *Identification of behaviors in an agent's phase space* (Citeseer, 1995). This work is a technical report. The document is available at http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.40.3355&rep=rep1&type=pdf.
5. J. Tani, N. Fukumura, Embedding a grammatical description in deterministic chaos: An experiment in recurrent neural learning. *Biol. Cybern.* **72**, 365–370 (1995).
6. K. Ikeda, K. Otsuka, K. Matsumoto, Maxwell-bloch turbulence. *Prog. Theor. Phys. Suppl.* **99**, 295–324 (1989).
7. K. Kaneko, Clustering, coding, switching, hierarchical ordering, and control in a network of chaotic elements. *Physica D.* **41**, 137–172 (1990).
8. I. Tsuda, Chaotic itinerancy as a dynamical basis of hermeneutics in brain and mind. *World Futures* **32**, 167–184 (1991).
9. I. Tsuda, E. Koerner, H. Shimizu, Memory dynamics in asynchronous neural networks. *Prog. Theor. Phys.* **78**, 51–71 (1987).
10. M. Adachi, K. Aihara, Associative dynamics in a chaotic neural network. *Neural Netw.* **10**, 83–98 (1997).
11. I. Tsuda, Toward an interpretation of dynamic neural activity in terms of chaotic dynamical systems. *Behav. Brain Sci.* **24**, 793–810 (2001).
12. I. Tsuda, Chaotic itinerancy and its roles in cognitive neurodynamics. *Curr. Opin. Neurobiol.* **31**, 67–71 (2015).
13. W. J. Freeman, Simulation of chaotic EEG patterns with a dynamic model of the olfactory system. *Biol. Cybern.* **56**, 139–150 (1987).
14. A. Luczak, P. Barthó, K. D. Harris, Spontaneous events outline the realm of possible sensory responses in neocortical populations. *Neuron* **62**, 413–425 (2009).
15. T. Kurikawa, K. Kaneko, Embedding responses in spontaneous neural activity shaped through sequential learning. *PLOS Comput. Biol.* **9**, e1002943 (2013).
16. H. Suetani, *International Conference on Artificial Neural Networks* (Springer, 2019), pp. 76–81.
17. Y. Kuniyoshi, S. Sangawa, Early motor development from partially ordered neural-body dynamics: Experiments with a cortico-spinal-musculo-skeletal model. *Biol. Cybern.* **95**, 589–605 (2006).
18. T. Ikegami, Simulating active perception and mental imagery with embodied chaotic itinerancy. *J. Conscious. Stud.* **14**, 111–125 (2007).
19. J. Park, H. Mori, Y. Okuyama, M. Asada, Chaotic itinerancy within the coupled dynamics between a physical body and neural oscillator networks. *PLOS ONE* **12**, e0182518 (2017).
20. J. Namikawa, J. Tani, A model for learning to segment temporal sequences, utilizing a mixture of rnn experts together with adaptive variance. *Neural Netw.* **21**, 1466–1475 (2008).
21. J. Namikawa, J. Tani, Learning to imitate stochastic time series in a compositional way by chaos. *Neural Netw.* **23**, 625–638 (2010).
22. J. Namikawa, R. Nishimoto, J. Tani, A neurodynamic account of spontaneous behaviour. *PLOS Comput. Biol.* **7**, e1002221 (2011).
23. S. Steingrube, M. Timme, F. Wörgötter, P. Manoonpong, Self-organized adaptation of a simple neural circuit enables complex robot behaviour. *Nat. Phys.* **6**, 224–230 (2010).
24. P. J. Werbos, Backpropagation through time: What it does and how to do it. *Proc. IEEE* **78**, 1550–1560 (1990).
25. Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.* **5**, 157–166 (1994).
26. Y. Yamashita, J. Tani, Emergence of functional hierarchy in a multiple timescale neural network model: A humanoid robot experiment. *PLOS Comput. Biol.* **4**, e1000220 (2008).
27. Y. Mori, P. Davis, S. Nara, Pattern retrieval in an asymmetric neural network with embedded limit cycles. *J. Phys. A Math. Gen.* **22**, L525–L532 (1989).
28. K. Nützel, J. Kien, K. Bauer, J. S. Altman, U. Krey, Dynamics of diluted attractor neural networks with delays. *Biol. Cybern.* **70**, 553–561 (1994).
29. V. Folli, G. Gosti, M. Leonetti, G. Ruocco, Effect of dilution in asymmetric recurrent neural networks. *Neural Netw.* **104**, 50–59 (2018).
30. H. Jaeger, The "echo state"approach to analysing and training recurrent neural networks with an erratum note. Bonn, Germany: German National Research Center for Information Technology GMD Technical Report (2001), vol. 148, 13 p.
31. K. Nakajima, Physical reservoir computing–an introductory perspective. *Jpn. J. Appl. Phys.* **59**, 060501 (2020).
32. W. Maass, T. Natschläger, H. Markram, Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Comput.* **14**, 2531–2560 (2002).
33. H. Jaeger, H. Haas, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science* **304**, 78–80 (2004).
34. H. Sompolinsky, A. Crisanti, H.-J. Sommers, Chaos in random neural networks. *Phys. Rev. Lett.* **61**, 259–262 (1988).
35. D. Sussillo, L. F. Abbott, Generating coherent patterns of activity from chaotic neural Networks. *Neuron* **63**, 544–557 (2009).
36. R. Laje, D. V. Buonomano, Robust timing and motor patterns by taming chaos in recurrent neural networks. *Nat. Neurosci.* **16**, 925–933 (2013).
37. B. DePasquale, C. J. Cueva, K. Rajan, G. S. Escola, L. Abbott, full-force: A target-based method for training recurrent networks. *PLOS ONE* **13**, e0191527 (2018).
38. S. Fusi, E. K. Miller, M. Rigotti, Why neurons mix: High dimensionality for higher cognition. *Curr. Opin. Neurobiol.* **37**, 66–74 (2016).
39. R. H. Byrd, P. Lu, J. Nocedal, C. Zhu, A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.* **16**, 1190–1208 (1995).
40. A. J. Ijspeert, A. Crespi, D. Ryczko, J.-M. Cabelguen, From swimming to walking with a salamander robot driven by a spinal cord model. *Science* **315**, 1416–1420 (2007).
41. C. Liu, D. Wang, Q. Chen, Central pattern generator inspired control for adaptive walking of biped robots. *IEEE Trans. Syst. Man Cybern. Syst.* **43**, 1206–1215 (2013).
42. D. Owaki, T. Kano, K. Nagasawa, A. Tero, A. Ishiguro, Simple robot suggests physical interlimb communication is essential for quadruped walking. *J. R. Soc. Interface* **10**, 20120669 (2013).
43. K. Kaneko, I. Tsuda, Chaotic itinerancy. *Chaos.* **13**, 926–936 (2003).
44. H. Itoh, M. Kimoto, Multiple attractors and chaotic itinerancy in a quasigeostrophic model with realistic topography: Implications for weather regimes and low-frequency variability. *J. Atmos. Sci.* **53**, 2217–2231 (1996).
45. A. Ahmadi, J. Tani, A novel predictive-coding-inspired variational RNN model for online prediction and recognition. *Neural Comput.* **31**, 2025–2074 (2019).

46. H. Korn, P. Faure, Is there chaos in the brain? II. Experimental evidence and related models. *C. R. Biol.* **326**, 787–840 (2003).
47. D. V. Buonomano, W. Maass, State-dependent computations: Spatiotemporal processing in cortical networks. *Nat. Rev. Neurosci.* **10**, 113–125 (2009).
48. R. Yuste, From the neuron doctrine to neural networks. *Nat. Rev. Neurosci.* **16**, 487–497 (2015).
49. J. P. Stroud, M. A. Porter, G. Hennequin, T. P. Vogels, Motor primitives in space and time via targeted gain modulation in cortical networks. *Nat. Neurosci.* **21**, 1774–1783 (2018).
50. M. G. Perich, J. A. Gallego, L. E. Miller, A neural population mechanism for rapid learning. *Neuron* **100**, 964–976.e7 (2018).
51. T. Taniguchi, N. Akashi, H. Notsu, M. Kimura, H. Tsukahara, K. Nakajima, Chaos in nanomagnet via feedback current. *Phys. Rev. B* **100**, 174425 (2019).
52. T. Yamaguchi, N. Akashi, K. Nakajima, S. Tsunegi, H. Kubota, T. Taniguchi, Synchronization and chaos in a spin-torque oscillator with a perpendicularly magnetized free layer. *Phys. Rev. B* **100**, 224422 (2019).
53. J. Torrejon, M. Riou, F. A. Araujo, S. Tsunegi, G. Khalsa, D. Querlioz, P. Bortolotti, V. Cros, K. Yakushiji, A. Fukushima, H. Kubota, S. Yuasa, M. D. Stiles, J. Grollier, Neuromorphic computing with nanoscale spintronic oscillators. *Nature* **547**, 428–431 (2017).
54. T. Furuta, K. Fujii, K. Nakajima, S. Tsunegi, H. Kubota, Y. Suzuki, S. Miwa, Macromagnetic simulation for reservoir computing utilizing spin dynamics in magnetic tunnel junctions. *Phys. Rev. Appl.* **10**, 034063 (2018).
55. S. Tsunegi, T. Taniguchi, K. Nakajima, S. Miwa, K. Yakushiji, A. Fukushima, S. Yuasa, H. Kubota, Physical reservoir computing based on spin torque oscillator with forced synchronization. *Appl. Phys. Lett.* **114**, 164101 (2019).
56. R. Nishimoto, J. Tani, Development of hierarchical structures for actions and motor imagery: A constructivist view from synthetic neuro-robotics study. *Psychol. Res.* **73**, 545–558 (2009).
57. S. L. Frank, H. Jacobsson, Sentence-processing in echo state networks: A qualitative analysis by finite state machine extraction. *Connect. Sci.* **22**, 135–155 (2010).
58. N. Gianniotis, S. D. Kügler, P. Tiňo, K. L. Polsterer, Model-coupled autoencoder for time series visualisation. *Neurocomputing* **192**, 139–146 (2016).
59. H. Strobelt, S. Gehrmann, H. Pfister, A. M. Rush, Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE Trans. Vis. Comput. Graph.* **24**, 667–676 (2018).
60. I. Shimada, T. Nagashima, A numerical approach to ergodic problem of dissipative dynamical systems. *Prog. Theor. Phys.* **61**, 1605–1616 (1979).

Acknowledgments: This work was based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO). **Funding:** K.I. was supported by JSPS KAKENHI (grant number JP20J12815). K.N. was supported by JSPS KAKENHI (grant number JP18H05472) and by MEXT Quantum Leap Flagship Program (MEXT Q-LEAP) (grant number JPMXS0120319794). This work was supported by NEDO [serial numbers 15101156-0 (dated 24 June 2016) and 18101806-0 (dated 5 September 2018)] and Chair for Frontier AI Education, School of Information Science and Technology and Next Generation AI Research Center [serial number not applicable (dated 1 June 2016)]. **Author contributions:** K.I. and K.N. conceived the idea and designed the experiments. K.I. carried out the experiments and created the demonstration. K.I. and K.N. wrote the manuscript. All authors discussed and commented on the manuscript. K.N. and Y.K. directed the project. **Competing interests:** The authors declare that they have no competing interests. **Data and materials availability:** All data needed to evaluate the conclusions in the paper are present in the paper and/or the Supplementary Materials. Additional data related to this paper may be requested from the authors.

Submitted 22 February 2020
Accepted 24 September 2020
Published 11 November 2020
10.1126/sciadv.abb3989

Citation: K. Inoue, K. Nakajima, Y. Kuniyoshi, Designing spontaneous behavioral switching via chaotic itinerancy. *Sci. Adv.* **6**, eabb3989 (2020).

Designing spontaneous behavioral switching via chaotic itinerancy

Katsuma Inoue, Kohei Nakajima and Yasuo Kuniyoshi

Sci Adv **6** (46), eabb3989.
DOI: 10.1126/sciadv.abb3989