


Dual sampling neural network: Learning without explicit optimizationJun-nosuke Teramae ^{*}*Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan*Yasuhiro Tsubo [†]*College of Information Science and Engineering, Ritsumeikan University, Shiga 525-8577, Japan* (Received 17 May 2021; revised 21 March 2022; accepted 19 August 2022; published 21 October 2022)

Artificial intelligence using neural networks has achieved remarkable success. However, optimization procedures of the learning algorithms require global and synchronous operations of variables, making it difficult to realize neuromorphic hardware, a promising candidate of low-cost and energy-efficient artificial intelligence. The optimization of learning algorithms also fails to explain the recently observed criticality of the brain. Cortical neurons show a critical power law implying the best balance between expressivity and robustness of the neural code. However, the optimization gives less robust codes without the criticality. To solve these two problems simultaneously, we propose a model neural network, dual sampling neural network, in which both neurons and synapses are commonly represented as a probabilistic bit like in the brain. The network can learn external signals without explicit optimization and stably retain memories while all entities are stochastic because seemingly optimized macroscopic behavior emerges from the microscopic stochasticity. The model reproduces various experimental results, including the critical power law. Providing a conceptual framework for computation by microscopic stochasticity without macroscopic optimization, the model will be a fundamental tool for developing scalable neuromorphic devices and revealing neural computation and learning.

DOI: [10.1103/PhysRevResearch.4.043051](https://doi.org/10.1103/PhysRevResearch.4.043051)**I. INTRODUCTION**

Seemingly optimized behaviors of macroscopic systems often emerge from stochastic interactions of microscopic entities in physics. Thermodynamic equilibrium specified by the minimization of the thermodynamic free energy is an outcome of statistics of large numbers of microscopic particles. The least action principle of classical mechanics is derived from the interference of the probability amplitude of microscopic quantum paths.

In the last two decades, machine learning algorithms based on neural networks have made substantial progress. The representative models include deep learning [1] and Boltzmann machine [2], consisting of deterministic and stochastic neuron models, respectively. Regardless of the neuron models, learning of the networks has been formalized as the optimization procedure of parameters, typically synaptic weights, of the network. Ingenious optimization techniques being developed have led to accelerations of the learning process using parallelized computation on graphical processing units (GPUs), resulting in the unprecedented performance of the networks

for various tasks. However, their massive energy consumption raises concerns of sustainability [3].

To reduce energy requirement of machine learning, there is a rapidly growing pursuit to realize brain-inspired neuromorphic computing hardware, the leading candidate for realizing low-cost and energy-efficient artificial intelligence [3–26]. Inspired by characteristic features of the biological brain, such as asynchronous spike generation of neurons, robustness against local failures of synaptic transmissions, and highly parallelized developments of neural and synaptic states, various architectures using highly functional materials, including nanoelectronics, spintronics, and photonic systems, have been proposed. Introducing these biological features, neuromorphic computing is expected to realize ubiquitous artificial intelligence applicable even to power critical situations, such as so-called edge devices [3,21].

A serious obstacle to realize neuromorphic computing is the global and synchronous nature of the optimization procedures of current learning algorithms. Error backpropagation learning, for instance, requires coordinated alternation of the forward and backward computations, synchronous updates of neural variables, and the propagation of the error signals over the entire network, in addition to feedforward network topology and explicitly defined objective functions [27,28]. They, however, have not been observed in the brain, which leads to recent extensive studies to solve the difficulty [29–39]. Learning algorithms of the Boltzmann machine also suffer biological implausibility [2,40]. These algorithms require symmetric synaptic weights, explicit switching between different computations during learning, and fine scheduling of

^{*}teramae@acs.i.kyoto-u.ac.jp[†]tsubo@fc.ritsumei.ac.jp

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

parameters, such as the temperature of the simulated annealing. Again, none of them have been observed in the brain.

Conventional optimization of neural networks also fails to explain a recently observed critical power law of the brain [41]. Experimental measurement of massive numbers of cortical neurons revealed that the covariance spectrum of their activity follows the power law with the exponent of -1 , indicating that the cortex realizes the best neural coding to balance expressivity and robustness. However, conventional supervised learning tends to push the network to as much robust as possible, leading the network to attain less expressive neural code without the critical power law, unlike the brain [42]. (A recent study reported that the covariance spectrum of the state variables of deep neural networks could show a power law. However, the exponents still differ from the critical value [43].)

This paper shows that the above seemingly independent problems of machine learning and neuroscience are solved simultaneously by considering the coexisting stochasticity of both the synapses and neurons in the brain. In the brain, neurons irregularly generate spikes with largely varying firing rates even across trials in which an animal performs a task in the same manner [41,44–48]. Besides neurons, synapses, which intermediate communication among neurons and are responsible for learning and memory in the brain, are also stochastic [49]. The formation, elimination, and volume change of synapses exhibit random fluctuations [50–56]. The release of neurotransmitters from synapses is also an inherently stochastic process [57–59]. Theoretically, it has been pointed out that these stochastic processes can carry out efficient computation through Bayesian inference [60–70]. So far, however, the stochastic behaviors of neurons and synapses have been studied separately, and it remains unclear whether there is a functional interaction between them.

We propose a neural network in which synapses, in addition to neurons, are modeled as binary stochastic variables like in the brain. The stochasticity of neurons and synapses are inseparably integrated into a simple framework of a sampling-based Bayesian inference model, which allows the network to learn external signals without explicit optimization and stably retain memories while all its entities are stochastic. The model is regarded as an extension of the Boltzmann machine in which the Bayesian posterior distribution, instead of thermal equilibrium of an energy function, is directly considered and in which synapses, besides neurons, are probability variables to be sampled. Learning is realized through local and asynchronous stochastic updates of variables in the network due to the extension. Because the algorithm is not derived as optimization of objective functions, it rarely exhibits serious overfitting. We call the model the “dual sampling neural network”.

Besides the critical power law, the model also reproduces various experimental results of the brain [71–77]. The derived algorithm well describes the plasticity of cortical synapses [71,72], the amplitude dependence of synaptic fluctuation [73,74], the response properties of cortical neurons, including Gabor-filter-like receptive fields [75,76], and the higher-order statistics of the topology of local cortical circuits [77]. These results strongly suggest that the stochastic behaviors of

neurons and synapses are inseparable and both essential attributes of neural computation and learning.

II. RESULTS

A. Architecture of the model

Most connections between cortical neurons are redundantly realized by multiple synapses [69,78]. Introducing this redundancy, we model a cortical network as a network of neurons whose connections are all realized by multiple synapses. Then, considering the stochastic behaviors of synapses and neurons, we represent each synapse and neuron in the model as a binary random variable [79–81] [Fig. 1(a), Appendix A]. We will see that the binary and stochastic representation of neurons and synapses enables us to obtain explicit stochastic evolution equations, which work as a learning algorithm of the network. The value of a neural variable determines whether that neuron generates a spike, whereas the value of a synaptic variable determines whether that synapse contacts a dendrite of the postsynaptic neuron [Fig. 1(a)]. The connection weight from the i th neuron to the j th neuron, which is generally asymmetric, is then given as a weighted sum of the synaptic states as $w_{ij} = \sum_{m=1}^M s_{ijm} a_{ijm}$, where $s_{ijm} \in \{0, 1\}$ is the state of the m th synapse of the connection from the i th neuron to the j th neuron. The constant a_{ijm} represents the contribution of the synapse to the weight, which corresponds to the amplitude of miniature postsynaptic potential of the synaptic contact. Note that a_{ijm} is a constant, and it is fixed even during learning. We set $w_{ii} = 0$ to avoid self-connections and use an evenly spaced sequence from $-a_0$ to a_0 for the values of the constants a_{ijm} throughout the main text for simplicity: $a_{ijm} = a_m = (2(m-1)/(M-1) - 1)a_0$ where $m = 1, 2, \dots, M$. We discuss other choices of the constants in Appendix B.

Similar to the Boltzmann machine, we assume that external inputs, including the target outputs of supervised learning, are presented to the network by fixing variables of some neurons, namely, visible neurons, to these input values. These input data, thus, should be represented by binary vectors. Other neurons in the network, which do not receive external data directly, are referred to as “hidden neurons.”

Note that the states of the neurons, including hidden and visible neurons, are generally different when a different datum of a dataset is presented to the network. In contrast, the states of synapses do not depend on each datum. They depend on the dataset as a whole because the neural network needs to find its connection weights, i.e., sets of synaptic states, that consistently account for all data of the dataset [Fig. 1(b), see Appendix A for full details]. For this reason, we write the state of the j th neuron at the time that the network is receiving the d th datum of the dataset as x_{dj} , using the data index, while the state of a synapse, s_{ijm} , does not have a data index.

Each neuron in the network receives inputs from its preceding, i.e., presynaptic, neurons. In this paper, we assume that, if the neural variable is not conditioned on states of its succeeding neurons, the neuron generates a spike with a prior probability

$$P(x_{dj} = 1 \mid \{x_{di}\}_{i \in p(j)}, \{s_{ijm}\}_{m \in M}) = \sigma(v_{dj}). \quad (1)$$

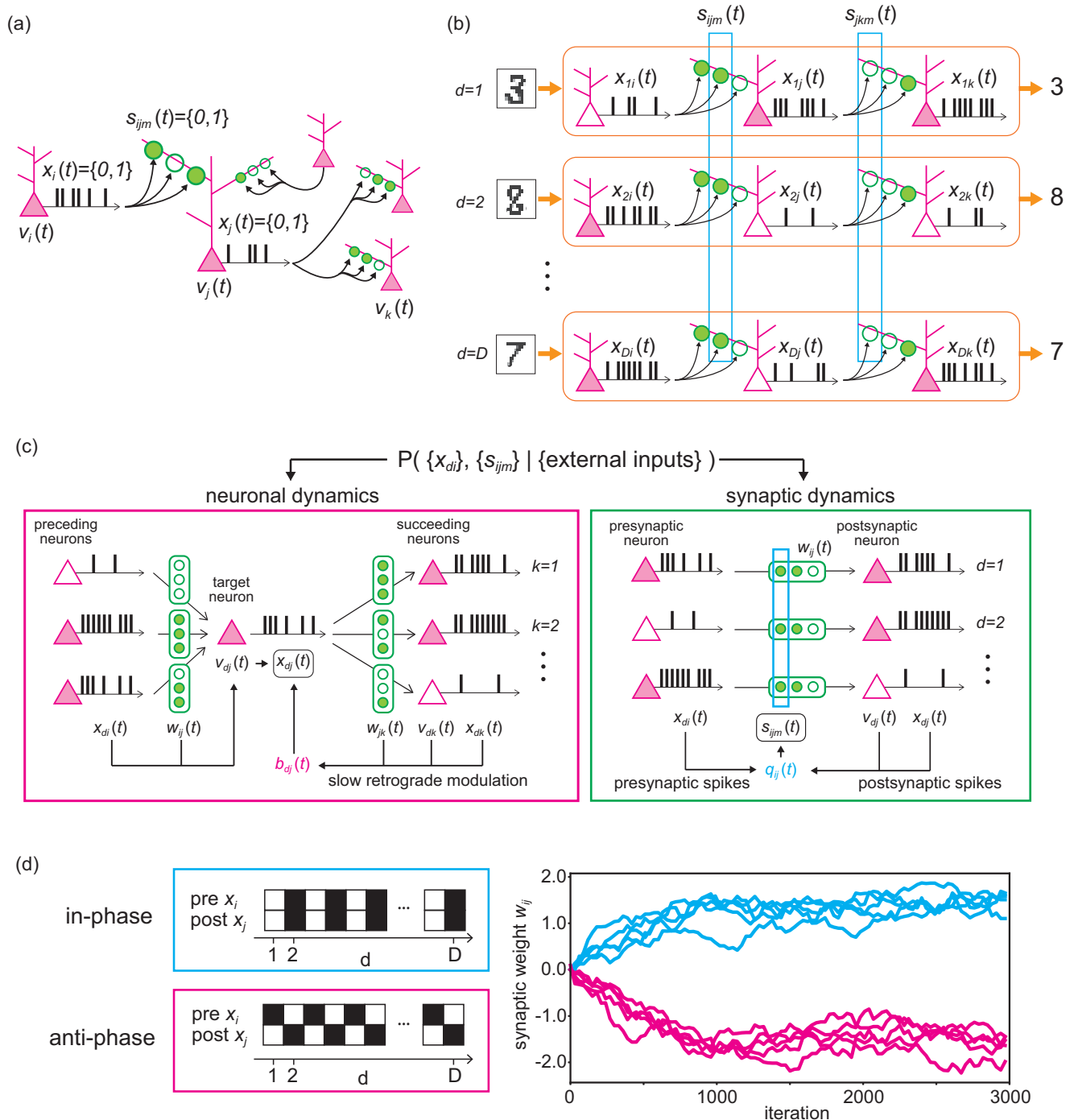


FIG. 1. Learning as a sampling of synaptic and neuronal states. (a) A neural network is modeled as a population of neurons connected via multiple synapses. (b) A neural network needs to respond consistently to all data of a given dataset while the state of the neurons is generally different for each given datum in the dataset. Thus, we denote the neural state with data index d like x_{di} , while the synaptic state s_{ijm} does not have a data index. (c) The Gibbs sampling from the Bayesian joint posterior distribution of synaptic and neuronal states conditioned on a given dataset provides stochastic and biologically plausible update rules of neurons and synapses. Owing to the difference in data dependency between the neural and synaptic variables illustrated in (b), the synaptic update requires summation over all data of a given dataset (right panel), while the neural update requires summation over the state of proximal neurons of the target neuron receiving a datum of the dataset (left panel). (d) Evolution of a synaptic weights (right panel) when presynaptic and postsynaptic neurons fire (left panel) in-phase (cyan) and antiphase synchronously (magenta).

Here, $p(j)$ denotes the set of preceding neurons of the j th neuron, M denotes the set of synapses for each connection, and $\sigma(x)$ is the activation function of a neuron, for which we use the sigmoidal function $\sigma(x) = (1 + e^{-x})^{-1}$ throughout the paper. The weighted sum of the inputs $v_{dj} = \sum_i x_{di} w_{ij} = \sum_i x_{di} \sum_m a_{ijm} s_{ijm}$ corresponds to the membrane potential of the j th neuron when the d th datum of the dataset is presented to the network. Similarly, we assumed that each synaptic variable takes its binary value independently each other if it is not conditioned on neural variables. We denote the logit of the prior probability $\log(P(s_{ijm} = 1)/P(s_{ijm} = 0))$ as $q_{0,ijm}$, which is equivalent to

$$P(s_{ijm} = 1) = \sigma(q_{0,ijm}). \quad (2)$$

Unless the prior distribution is biased, the logit simply vanishes. Thus, we simply assumed $q_{0,ijm} = 0$ throughout the paper.

B. Learning as a dual sampling

With the fundamentals as described above, we hypothesize that the stochastic dynamics of neurons and synapses constitute a continuing random process to generate a network that suitably interprets the external world. In other words, we define learning in the network as a continuing sampling process of all free stochastic variables, namely, variables of all synapses and hidden neurons, from their posterior distribution conditioned on a given dataset, $P(\{x_{dj}\}_{d \in D, j \in H}, \{s_{ijm}\}_{i, j \in N, m \in M} \mid \{x_{dj}\}_{j \in V})$ (see Appendix A). Here, D denotes the dataset, H and V denote the sets of hidden and visible neurons, N represents the set of all neurons. This formulation is naturally regarded as an extension of the Boltzmann machine in which synapses, as well as neurons, are random variables to be sampled and in which we directly use the Bayesian posterior distribution instead of a typically used energy function that is characterized by the first and the second-order statistics of random variables. While conventional approaches consider connection weights as model parameters to be optimized, we consider them as random variables to be sampled, dismissing explicit optimization procedures from the algorithm.

A sampling from the posterior distribution, however, is computationally and biologically intractable in general, due to the high dimensionality of the system and complex dependencies among its variables. To solve this problem, we hypothesize that the brain realizes this sampling as the Gibbs sampling [82]. The Gibbs sampling ensures that we can replace sampling from the high-dimensional posterior distribution with iterative samplings of each variable from a posterior distribution of only that variable conditioned on all other variables, thus from a one-dimensional distribution, $P(x_{dj} \mid \dots)$ and $P(s_{ijm} \mid \dots)$ where the dots represent all other variables than the target variable. Due to the flexibility of Gibbs sampling, each sampling can be performed in any order and with any frequency, which enables each neuron and synapse to update their variables asynchronously and irregularly with their own individually determined timings without any global schedule or coordination.

Assuming that the network is directed acyclic graph (DAG) and applying Bayes' theorem to the posterior distributions,

we can derive the procedures of the Gibbs sampling as the stochastic update rules for the neurons and the synapses [Fig. 1(c), see Appendix A], that is

$$P(x_{dj} = 1 \mid \dots) = \sigma(v_{dj} + b_{dj}) \quad (3)$$

$$b_{dj}(t + 1) = (1 - r_b)b_{dj}(t) + r_b \sum_k w_{jk}(x_{dk} - \sigma(v_{dk})) \quad (4)$$

for a neuron and

$$P(s_{ijm} = 1 \mid \dots) = \sigma(a_{ijm}q_{ij}) \quad (5)$$

$$q_{ij}(t + 1) = (1 - r_q)q_{ij}(t) + r_q \sum_d x_{di}(x_{dj} - \sigma(v_{dj})) \quad (6)$$

for a synapse. Equations (3) and (5) give the firing probability of the neuron x_{dj} and the connection probability of the synapse s_{ijm} , respectively, after considering all other variables. In addition to the preceding neurons' state included in the prior distribution of the firing probability via the membrane potential, the posterior distribution includes the succeeding neurons' state via the latent scalar variable b_{dj} . The state of each neuron's distal neurons and the data given to the network influence each variable as the variables of the network are updated. The derived equations show that the probabilities mutually interact via the latent scalar variables b_{dj} and q_{ij} assigned to each neuron and the synapse, respectively. This mutual interaction is summarized as follows [Fig. 1(c)]: Let us first focus on the firing probability of a neuron. We refer to the neuron as the target neuron for a while for clarity. If a preceding neuron of the target neuron generates a spike, it immediately changes the membrane potential v of the target neuron and changes the firing probability of the target neuron. If a succeeding neuron of the target neuron generates a spike, it slowly and retrogradely modulates the excitability of the target neuron via the latent variable b of the target neuron, whose evolution is governed by the sum of products of synaptic weights and generated spikes [Eq. (4)]. Then, let us focus on a synapse. The connection probability of the synapse s is a function of the latent variable q [Eq. (5)]. The evolution of the latent variable is governed by the sum of products of spikes of presynaptic neurons and those of postsynaptic neurons [Eq. (6)]. Constants r_b and r_q characterize evolution timescales of the latent variables b_{dj} and q_{ij} , respectively.

Note the difference between the summation indices in Eqs. (4) and (6). The synaptic update requires summation over all data of a given dataset, while the neural update does not [Fig. 1(c)]. This difference results from the difference between the data dependencies of the variables, as discussed above. Because of the difference, synapses need to accumulate the neural activities for many, ideally all, data in the dataset before updating their states, while neurons update their states independently in a manner that depends on each datum individually. This implies that synapses evolve much more slowly than neurons if data are provided sequentially to the network, as in the brain. This explains why the timescales of neurons and synapses are greatly different in the brain. In contrast, when we use the algorithm to train an artificial neural network, we can perform neural updates for all data of the dataset parallel. This greatly accelerates the learning speed. In the following numerical simulations, we adopt the parallel update

of neural variables to accelerate computational speed. (Results of the online implementation of the algorithm are discussed in Appendix E.)

The update rule for synapses given in Eqs. (5) and (6) yields plasticity similar to that exhibited by cortical synapses [71,72]. Each term in the summation in Eq. (6) vanishes unless the presynaptic neuron x_{di} fires. If the presynaptic neuron does fire, whether the summand of Eq. (6) will be positive or negative depends on whether the postsynaptic neuron x_{dj} fires simultaneously. For this reason, each synaptic weight increases if the network receives many data in which the presynaptic and postsynaptic neurons of the connection fire synchronously, while it decreases if the presynaptic and postsynaptic neurons fire asynchronously [Fig. 1(d)].

Interestingly, the synaptic update rule depends on membrane potential v_{dj} , in addition to the spikes x_{dj} , of the postsynaptic neuron. This is consistent with a recently proposed model of the spike-time-dependent plasticity that accounts for various properties of synaptic plasticity by introducing the average membrane potential of postsynaptic neurons into the model [72]. Due to the dependency on the membrane potential, unlike most existing models of synaptic plasticity, the plasticity derived here does not require any artificial bound on synaptic weights. The incremental variation of each connection weight automatically decays to zero when the magnitude of the weight becomes large [Fig. 1(d)] because $\sigma(v_{dj})$ is the expectation value of x_{dj} , and the difference between them, $x_{dj} - \sigma(v_{dj})$, decays to zero in such cases. Also, note that Eq. (5) implies that the state changes more frequently for synapses with smaller amplitude a_{ijm} , consistent with experimental observations of negative amplitude dependence of synaptic fluctuation [73,74].

Following the above equations, the state of each neuron and each synapse is irregularly, asynchronously, and concurrently updated with its own individually determined frequency. Because the equations are derived as an implementation of the Gibbs sampling, the concurrent updates of variables without any global scheduling enables the state of the hidden neurons and the synapses to generate samples following the posterior distribution conditioned on the given dataset, i.e., the network evolves to suitably explain the training dataset and generate the desired output with high probability. As we will see below, the network generates the desired output even when input data not included in the training dataset are fed to the input neurons, and the output neurons are kept free, meaning that the equations work as the learning algorithm of the network. While we obtained the stochastic update equations under the assumption that the network is DAG, which is not held by recurrent networks, we will numerically see that the updates work stably and effectively even for recurrent networks as a learning algorithm.

Unlike existing learning algorithms requiring the alternating execution of different computations, e.g., forward and backward computations or sampling and annealing procedures, our algorithm realizes learning through the iteration of a single computation for each variable. Furthermore, the equations are local in the sense that they depend only on neurons and synapses directly connected to the updated variable. Owing to the local nature of the learning dynamics, we can apply the algorithm to train recurrent networks, as well

as feedforward ones, with generally asymmetric connections. These features are particularly suited to biological and hardware implementations of the algorithm.

C. Feedforward networks

To see how the dual sampling enables learning of the network, we first apply the algorithm to a simple problem, a noisy and high-dimensional variant of the XOR problem, learned in a supervised manner by a three-layered network (see Appendix A for full details). We prepared training and test datasets consisting of pairs of input and target output and ran the algorithm on the network with feeding the training dataset. During the learning process, we measured the training and test accuracies by giving the input data to the input neurons while output neurons were kept free. Figure 2(a) displays the evolution of the training and test accuracies as functions of the number of the sampling iteration of the algorithm. It is seen that these accuracies nearly coincide, and they quickly increase to values close to unity and remain there.

Significantly, even while the accuracies remain nearly constant, the synaptic weights of the network continue to fluctuate greatly [Fig. 2(b)], and the firing patterns of the hidden neurons also continue to change, even when the same datum is given to the network, without converging to a fixed pattern [Fig. 2(c)]. These results are consistent with experimental observations of continuing fluctuations of synapses and the trial-to-trial variability of neural activity and answer why the brain can realize reliable computation and learning despite stochastic dynamics of neurons and synapses.

To study the robustness of the algorithm concerning its constant parameters, we measured test accuracies of the network after learning for various combinations of the number of synapses per connection, M , and the maximum amplitude of synapses, a_0 , [Fig. 2(d), see Appendix A]. Except in a narrow range in which one of these constants is so small that the possible maximum weight given by a_0M is small, the network almost perfectly learns to perform the task.

We then consider the influence of the parameters r_b and r_q [Figs. 2(e) and 2(f)] that characterizes timescales of the evolution of the latent scalar variables of neuron b and synapse q , respectively. Unless r_q is extremely small, and hence q evolves extremely slowly, the training and test accuracies will increase and reach values near unity, while their convergence speed decreases as r_b or r_q decreases. (Note that the synaptic evolution is already slower than the neuronal evolution even when $r_q = 1$, as discussed above. Therefore, a small value of r_q may result in unrealistically slow synaptic dynamics.)

To demonstrate the applicability of the method to practical problems, we next study the application of the algorithm to training multilayered feedforward networks using the MNIST dataset (Fig. 3). We found that the accuracies quickly increase to values near 95%, while the number of required iterations and the asymptotically realized accuracies slightly decrease as we increase the number of layers in the network [Figs. 3(a)–3(c)]. Figure 3(d) displays examples of numerals that the network fails to recognize. These are quite ambiguous and difficult even for a human to identify with confidence.

The spatial distribution of connection weights from the input neurons to a hidden neuron acts as the receptive field

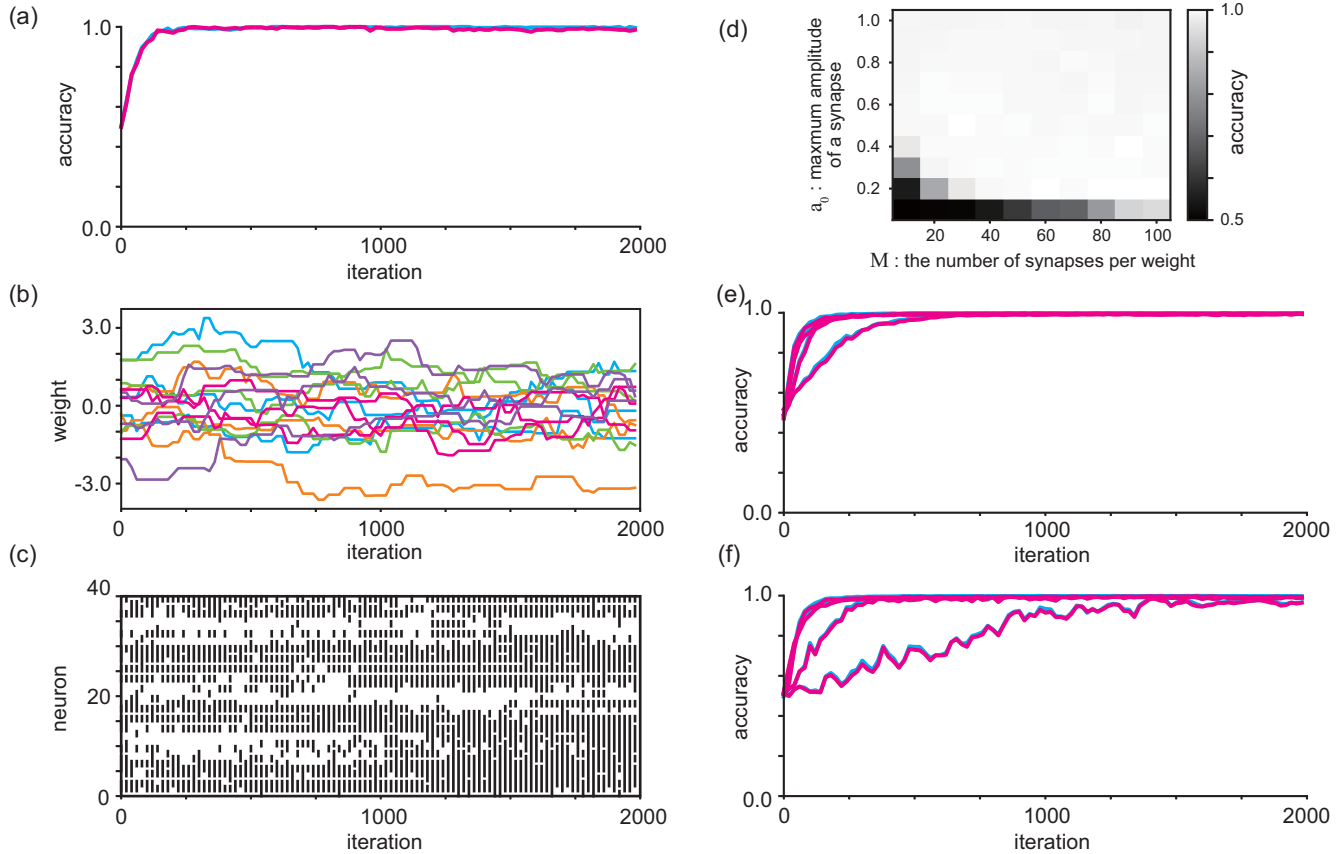


FIG. 2. Supervised learning of feedforward networks using a simple artificial dataset. (a) Training (cyan) and test (magenta) accuracies of a three-layered network as functions of the number of sampling iterations. (b) Evolution of randomly chosen synaptic weights of the network. Different colors are used for different presynaptic neurons. (c) Evolution of the states of randomly chosen hidden neurons when the same datum in the dataset is presented to the network. (d) Test accuracies realized by the learning algorithm with various values of the number of synapses per connection K , and maximum amplitude of a synapse a_0 . (e) Evolution of training (cyan) and test (magenta) accuracies for $r_b = 1.0, 0.1, 0.01, 0.001$, from top to bottom. (f) The same as (e) but for $r_q = 1.0, 0.1, 0.01, 0.001$, from top to bottom.

or the kernel of the hidden neuron, often having a localized structure in the brain. To see whether the network after learning acquired the receptive fields similar to ones in the brain, we illustrated their examples in Fig. 3(e). We can see that Gabor-filter-like localized structures that resemble receptive fields of neurons in the primary visual cortex [75] are organized through the learning. The introduction of a sparsity constraint similar to the ones in the brain on neural activity into the loss function of learning is known to provide Gabor-filter-like receptive fields for an artificial neural network [76]. Interestingly, the network can acquire similar localized structures of receptive fields, even without any explicit additional constraint on the learning algorithm derived from Gibbs sampling.

The learning algorithm can avoid serious overfitting to the training data because it is not derived as a direct optimization of any objective functions. To see this point, we trained a three-layered feedforward network using small numbers of samples of the MNIST dataset and compared the resulting training and test accuracies with those obtained from back-propagation learning using the stochastic gradient descent (SGD) and Adam algorithms [83] [Fig. 3(f)]. We found that for both the SGD and Adam algorithms, as the size of the training dataset is increased, the test accuracy decreases (in

the SGD case) or remains nearly constant (in the Adam case), while the training accuracy increases monotonically, which implies overfitting to the training dataset. Contrastingly, with the proposed algorithm, as the size of the training dataset is increased, both the training accuracy and the test accuracy increase, maintaining a slight difference between them. This implies that serious overfitting does not occur in the proposed learning algorithm.

Continuing fluctuation of the dynamics of neurons and synapses implies that the network can forget the learned dataset after learning if we unclamp output neurons to make them free while keeping the stochastic evolution of the synapses. To see how long the network retains its learned function or how quickly it forgets the learned dataset, we measured the development of the accuracies along the learning followed by the unclamping of the output neurons from the training dataset (Fig. 4). We used the same three-layered network we had used in the situation of Fig. 3(a). As expected, the accuracies started to decrease after we removed target outputs from the output neurons. However, the decay is pretty slow as the accuracies maintain high values during several hundred learning iterations. It is in marked contrast with the rapid increase of the accuracies during the learning, where they suddenly increased in about 100 steps. The slow

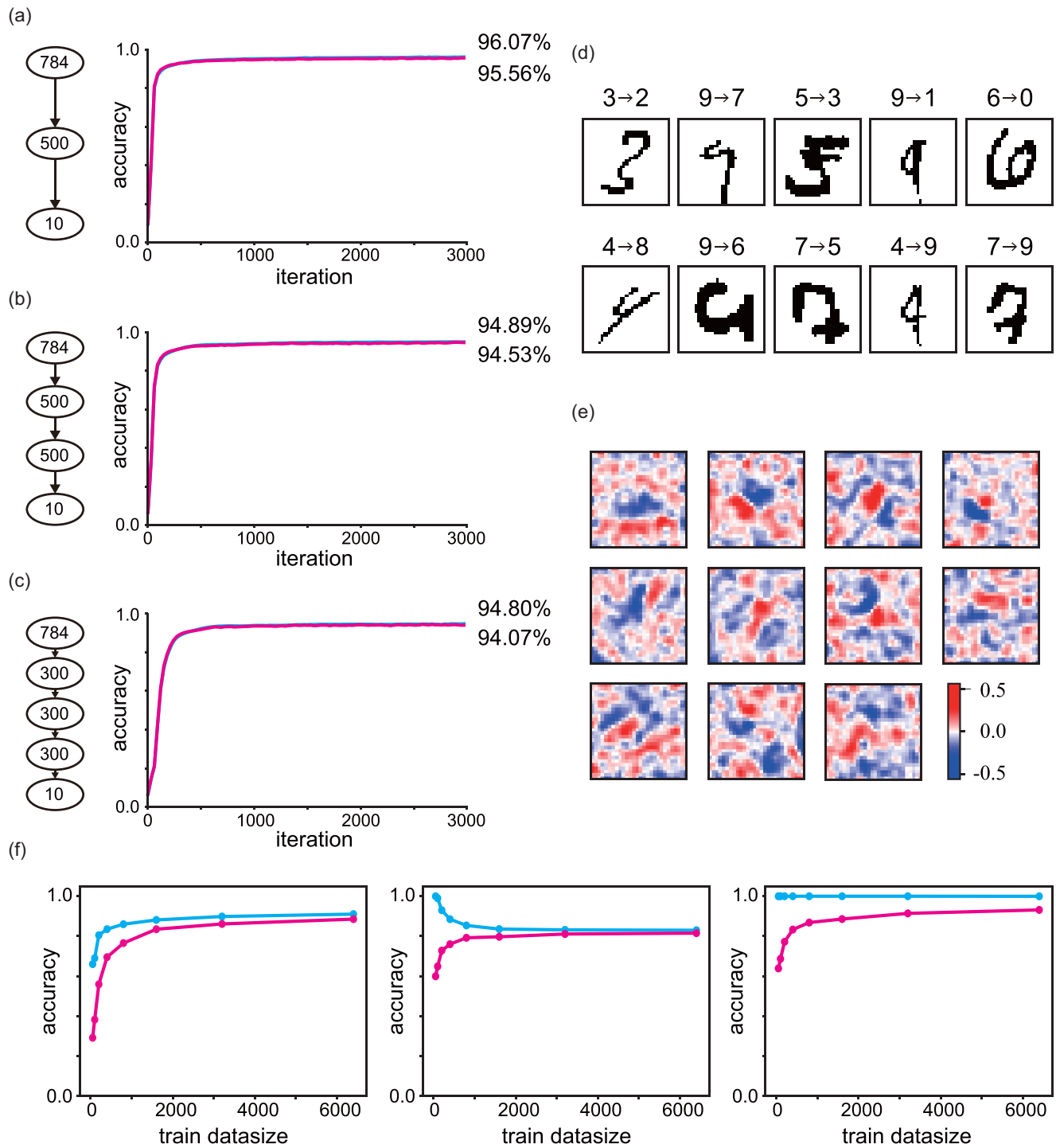


FIG. 3. Supervised learning of feedforward networks using the MNIST dataset. (a-c) Training (cyan) and test (magenta) accuracies as functions of the number of sampling iterations for (a) three-, (b) four-, and (c) five-layered networks. Each number in an oval indicates the number of neurons in the corresponding layer. (d) Exceptional examples of training images (upper row) and test images (lower row) that the three-layered network fails to recognize. (e) Examples of receptive fields of hidden neurons in the three-layered network. (f) Training (cyan) and test (magenta) accuracies of the three-layered network as functions of the size of the training dataset. The network was trained by the proposed algorithm (left panel), a backpropagation learning algorithm with a naïve stochastic gradient descent (middle panel), and with the Adam algorithm (right panel).

decay is explained by the smallness of the magnitude of the scalar variables b after learning. The learning has made the output neurons take their state value x with high probability, implying that the difference between the state value and its

expectation $\sigma(x)$, and thus the magnitude of the b , are small. Therefore, the preceding neurons of the output neurons and further preceding ones, in the same way, tend to keep their state until the fluctuation makes them degrades passively. It is

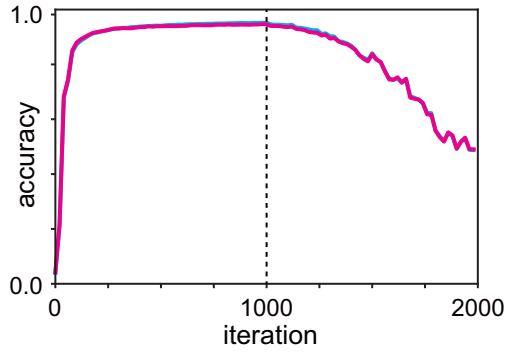


FIG. 4. Training (cyan) and test (magenta) accuracies as functions of the sampling iterations during and after learning. We unclamped the output neurons from the training dataset at 1000 sampling iteration (dashed line).

unlike the initial stage of the learning, where the retrograde signals b actively modulates the state of the neurons.

Unlike most artificial neural networks in which the sign of connection weights from a neuron can be either positive or negative, neurons in the brain follow Dale’s principle claiming that connection weights from an excitatory and an inhibitory neuron should exclusively be positive and negative, respectively. Several learning algorithms have been proposed because the standard learning algorithms are not viable for the sign-constrained network [84,85]. However, the algorithm derived here can apply to the sign-constrained networks without any changes because the sign of each synaptic strength does not change during learning. We prepared a feed-forward network with excitatory and inhibitory neurons and made it learn a task to confirm the validity of the algorithm under Dale’s principle (Fig. 5). The training and test accuracies rapidly increase as before while reached values are slightly lower than those of the network without the sign constraint [Fig. 5(a)]. Figures 5(b) and 5(c) show the evolution of the connection weights during learning and their density distribution after learning, respectively. We can see that the weights evolve with fluctuation as before but while keeping their signs.

D. Most efficient power-law coding

A recent biological experiment carried out by simultaneously recording the activity of a massive number of mouse primary visual cortex neurons revealed that the variance spectrum of the principal component of neural activities obeyed a power law with an exponent -1.04 that is slightly less than -1 [41]. The authors of the paper [41] mathematically proved that if the exponent is greater than -1 , the population code, i.e., representation of given stimuli by the population of neurons, lies on a nondifferential manifold and less robust, while if the exponent is less than -1 , the population code is restricted on a subspace with lower dimension than the number of neurons, implying that the code cannot fully utilize the neural resources. Thus, the experimentally observed power-law coding with an exponent slightly less than -1 is the most efficient in the sense that, in this case, the population response of the neurons lies on a manifold of the highest possible dimension while maintaining high generalizability.

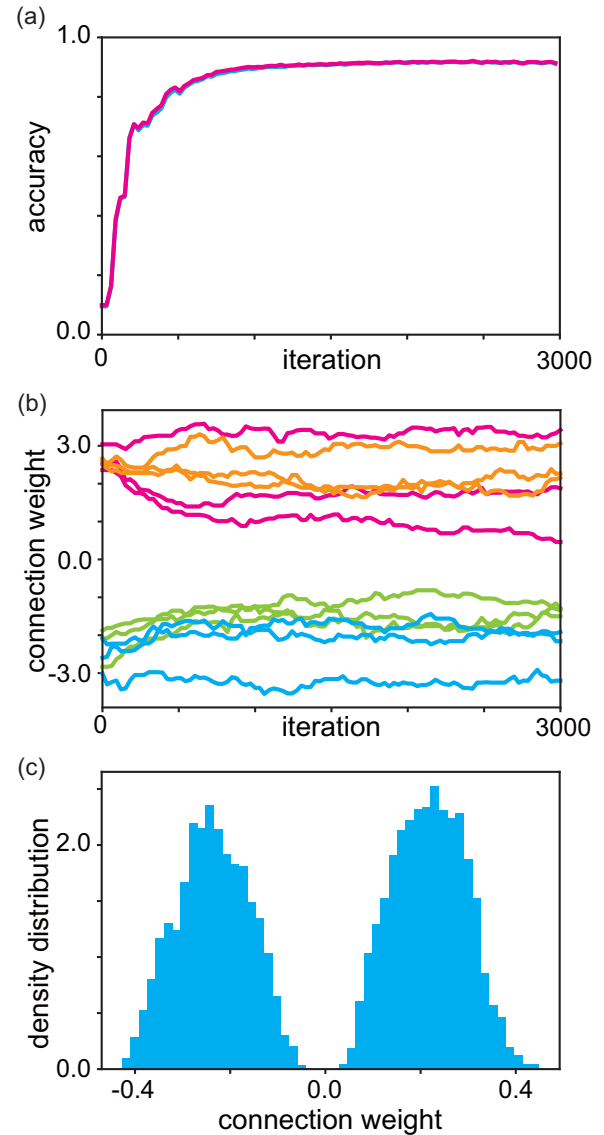


FIG. 5. Supervised learning of a feedforward network following Dale’s principle. (a) Training (cyan) and test (magenta) accuracies as functions of the number of sampling iterations. (b) Evolution of randomly chosen six excitatory (magenta and orange) and six inhibitory (cyan and green) connection weights. Different colors are used for connections having different postsynaptic neurons. (c) Density distribution of the connection weights after learning.

To test whether a network trained by the proposed algorithm realizes the most efficient power-law coding with the exponent that is slightly less than -1 , we numerically calculated the variance spectrum of the principal components of the mean activity of the neurons in the hidden layer of a network trained with the MNIST dataset. As shown in Fig. 6(a), the variance spectrum exhibits clear power-law decay with an exponent of -1.06 . To check the generality of the result for another training dataset, we trained the network to the fashion-MNIST dataset too and obtained similar power-law decay of the variance spectrum with the exponent slightly less than -1 (see Appendix C). This value is very close to the experimental result and is indeed slightly less than -1 . We conclude

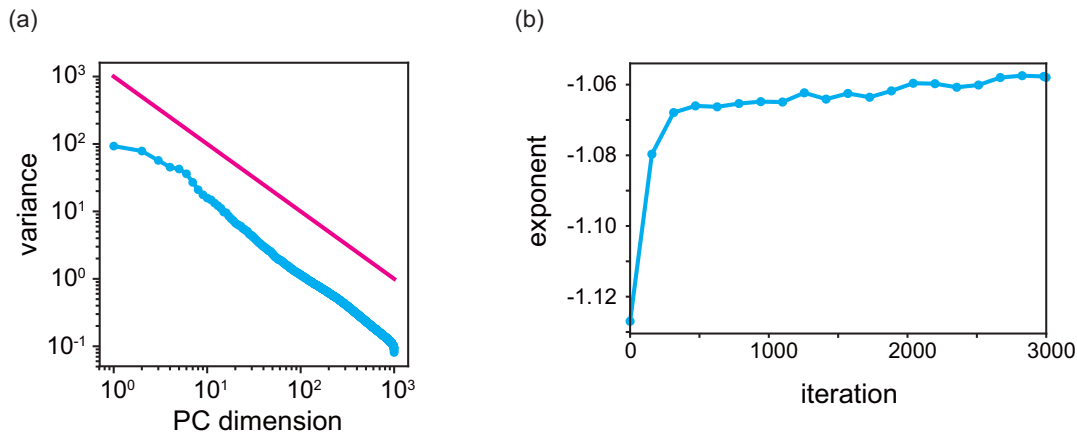


FIG. 6. Nearly optimal power-law decay of the variance spectrum for the principal component of the neural activity. (a) Variance spectrum of the principal components of the mean population activity of the hidden neurons across the training dataset for the three-layered network after training (cyan). The variances are arranged in descending order. The line (magenta) indicates the critical slope corresponding to an exponent of -1 . (b) Evolution of the power-law exponent of the variance spectrum as a function of the number of sampling iterations.

that the proposed learning algorithm leads the network to the most efficient coding. This may explain the high generalizability of the trained network that yields nearly coincident training and test accuracies, as seen in Figs. 2(a), 3(a), 3(b), and 3(c).

We next study how the exponent of the power law develops during learning. Figure 6(b) shows that the exponent approaches a value close to -1 from below as the learning proceeds. This result implies that the network does not start from a nonsmooth, almost perfect representation of the training dataset. Rather, it first learns a coarse representation of the dataset and then gradually acquires finer structures while maintaining differentiability of the representation of the data in coding space. This leads us to conclude that the robustness or generalizability of the population coding takes priority over the precision of the data representation in the learning. This priority must particularly be beneficial for animals that must survive in a ceaselessly changing environment.

Why the learning yields the critical power-law decay of the spectrum remains unclear. If we trained the network by the Adam or the SGD algorithm instead of the algorithm derived here, we obtained exponential decay or power-law decay with exponents significantly smaller than the critical value, i.e., faster decays (results not shown). The faster decays indicate that the above optimization algorithms let the network represent the given dataset using only a few degrees of freedom of internal neurons, presumably due to the overfitting of model parameters to the training dataset. We speculate that our algorithm can avoid the steeper decays because it avoids overfitting as it continues to generate random samples of connection weights instead of making them converge to single values. On the other hand, slower decays than the critical one imply that the representation of the dataset by the internal neurons is discontinuous and is highly vulnerable to random perturbation to the activity of the internal neurons. However, because our algorithm is intrinsically stochastic, the network needs to evolve under random fluctuations during learning. Thus, we infer that the algorithm can avoid the slower decays because it forces the network to obtain internal representations that are robust against the intrinsic fluctuation. To develop a

quantitative analysis about the evolution and resultant exponent by the learning is an important future subject.

E. Recurrent networks

We next applied the algorithm to train a network with generally asymmetric recurrent connections [Fig. 7(a), see Appendix A] using the MNIST dataset. Figure 7(b) displays the evolution of the training and test accuracies as functions of the number of sampling iterations. The accuracies were obtained from the states of the output neurons of the network measured after recursive evolutions of the states of the hidden neurons. We see that, as in the case of the results for the feedforward networks, the training and test accuracies nearly coincide and rapidly increase. This implies that the algorithm is able to train even a recurrent network as well as feedforward networks.

We speculate that the algorithm derived under the assumption of DAG practically works even for the recurrent network, that is not DAG, because most of the neurons in the network are divided into preceding and successive neurons for each neuron without overlap as the network rarely has direct feedback connections if connection probability is not very high. How the performance of the algorithm degrades as increasing the probability of the recurrent connections will be an interesting future subject.

1. Statistics of network motifs

It has been reported that local cortical circuits are highly nonrandom, and that connectivity patterns consisting of multiple neurons, known as network motifs, exhibit a characteristic distribution in which highly clustered patterns are overrepresented [77]. To study whether a recurrent network trained by the proposed algorithm acquires a similar distribution of the connectivity patterns, we determined the connectivity of triplets of neurons in a trained recurrent network. In the analysis, similarly with the experimental report, we restrict ourselves to patterns of triples of neurons consisting only of strong positive, i.e., excitatory, connections (see Appendix A and D). The statistics for the ratio of the actual counts of

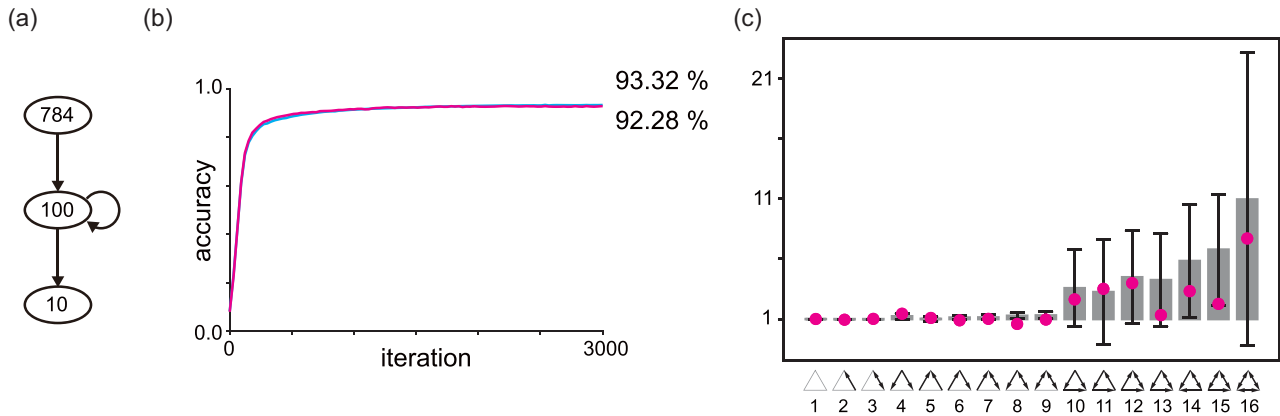


FIG. 7. Supervised learning of a recurrent network using the MNIST dataset. (a) A recurrent neural network. Each number in an oval indicates the number of neurons in the corresponding layer. (b) Evolution of the training (cyan) and test accuracies (magenta) of the recurrent network. (c) Average ratios of the actual numbers of three-neuron patterns in the trained recurrent network to those predicted by the null hypothesis (gray bars). The error bars indicate the range of $\pm 2\sigma$. The magenta circles are experimental results for real cortical circuits [77].

triplet patterns to the chance level are plotted in Fig. 7(c). The same ratios for the experimental results are also overlaid in the figure. While there are some exceptional cases in which the ratios obtained here are somewhat larger than those obtained experimentally, the two distributions of triplet patterns are surprisingly similar. As observed experimentally, highly connected motifs, i.e., those numbered 10 through 16 in Fig. 7(c), are overrepresented by a factor several times greater than the chance level. It is also interesting for the pattern that is only slightly higher than the chance level, as in the 4th pattern in Fig. 7(c), the model gives a result that is quite close to the experimental result. These results support the validity of the derived algorithm as a model describing the formation of local cortical circuits.

We speculate that the overrepresentation of the fully connected motifs can partly be accounted for by the receptive fields of the internal neurons, while we have not succeeded in the analytical derivation of the motif distribution. Previous experimental observation reported that the connection probability between neurons monotonically increases as the similarity of their receptive fields increases [86], which seems consistent with our result since synaptic weights between two neurons increase if the neurons generate spikes with a high rate for similar data sets, as shown in Fig. 1(d). Because the receptive field of the internal neurons is a realization of the internal representation of the dataset by the network, it must be an interesting future subject to develop a theory to connect the power law of the internal representation, the distribution of the receptive fields, and the probability of the recurrent connections to predict the motif distribution.

2. Temporal sequence learning

We next consider the application of the algorithm to train a recurrent network with a temporal sequence (Fig. 8). We prepared a temporal sequence of music notes in which the same temporal inputs, i.e., the same music notes, appear multiple times at different times [the top panel of Fig. 8(b)], and trained a recurrent network to predict the following input of the current sequence [the bottom panel of Fig. 8(b)]. In this case, the network needs to learn to store the history of inputs over some interval to generate the desired output, i.e., the following note.

The training procedure is the same as that used in the case considered in Figs. 2–7, except that we identify the iteration of the updates of the variables of the network as the time development. In contrast to the algorithm known as “backpropagation through time,” this procedure does not require virtually unfolding the recurrent connections of the network along the time axis. Therefore, the procedure ensures the biological plausibility of the algorithm, in which time proceeds naturally, and updating the neurons and synapses requires information regarding only their current and immediate past states.

The learning algorithm successfully trained the network to output the desired sequence [Fig. 8(b), the 4th panel from the top]. We find that the output produced by the hidden neurons after learning depends not only on the current input but also on past inputs [the 3rd panel of Fig. 8(b)]. This indicates that the algorithm enables the network to properly store input histories as the current activity of the hidden neurons.

III. DISCUSSION

This study proposed a model of neural networks that is an extension of the Boltzmann machine, in which neurons and synapses work together to realize efficient learning as a sampling-based Bayesian inference. The derived algorithm provided biologically plausible stochastic dynamics of neurons and synapses and gave results consistent with experimental findings of the brain, including the optimal power-law encoding of a given dataset. The obtained biological observations are not unique to our algorithm, as various previous studies have succeeded in explaining them. It is, however, fascinating that the learning algorithm derived here provides such various features even without any explicit constraints to reproduce them.

Recently, there has been a rapid growth of interest in developing biologically plausible learning algorithms and implementing them to energy-efficient devices, i.e., neuromorphic computing [3–26]. The algorithm proposed here seems to have various benefits as an algorithm for neuromorphic computing. Unlike most of the current learning algorithms, the algorithm intrinsically works asynchronously and locally due to the characteristics of the Gibbs sampling. It also does not require alternative switching between different

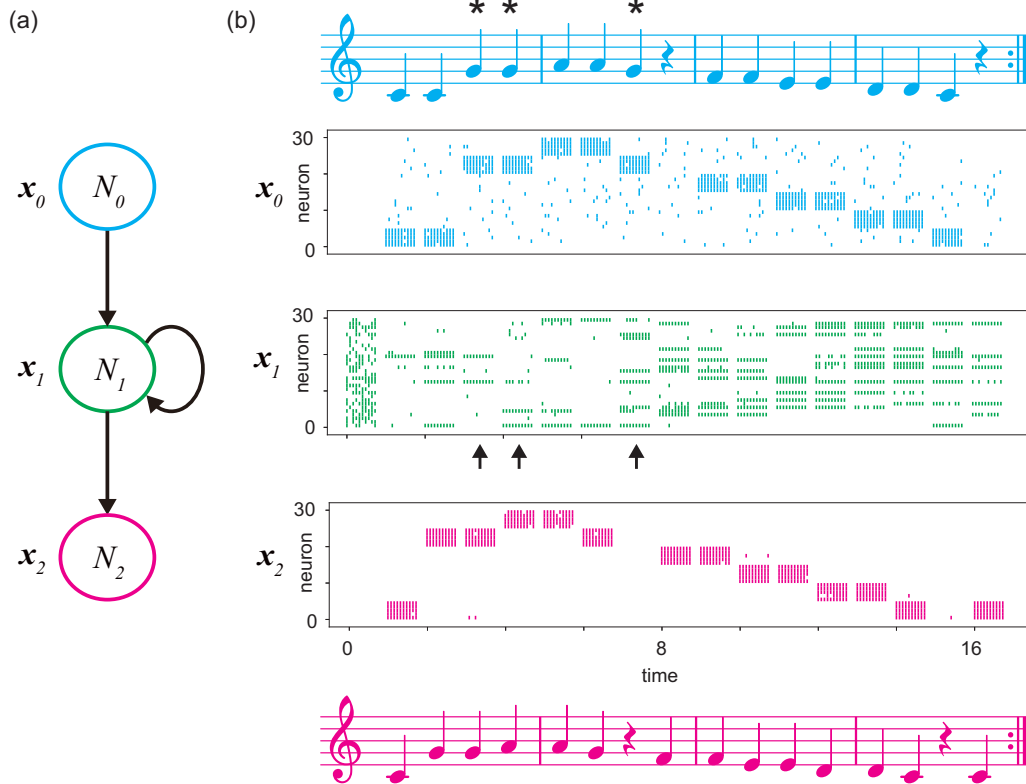


FIG. 8. Supervised learning of recurrent networks using temporal sequences. (a) Network structure. (b) The input sequence (the top panel) and the target output sequence (the bottom panel) of the dataset represented as sheet music. The asterisks indicate an example of times at which the network must output different notes while the current input to the network is the same. The second panel shows a raster plot of the activities of the input neurons. For visibility, we plotted the states of five cells per each of the six notes from C to A and overlaid the results of randomly chosen ten data of the dataset with a slight displacement of the time axis. The third and fourth panels show the activities of the hidden neurons and the output neurons, respectively, after learning of 500 sampling iterations. Arrows indicate times at which the current input to the network is the same. We plotted the states of randomly chosen 30 cells in the third panel and randomly chosen five cells for each of the six notes in the fourth panel with the slight displacement of the time axis for the ten data results.

computations such as feedforward and backward propagations or free and clamped phases during the learning. These features of the event-driven computation are suitable for neuromorphic computing as they will reduce the complexity of the circuit and allow it to avoid the high energy cost of communication to synchronize the entire network to a single clock.

Stochastic computation and binary communication of the algorithm also seem highly beneficial for neuromorphic computing. Communication using binary bit values rather than analog values will simplify the hardware and largely reduce communication costs for learning and inference. Furthermore, the stochasticity of the algorithm will improve the robustness of the hardware implementation of the algorithm and allow us to omit elaborate error detection and correction mechanisms. It will reduce the hardware’s energy consumption because much energy is used to protect and stabilize the binary states of devices from thermal or quantum noise. Inspired by the stochastic spike emission of neurons in the brain, studies of neuromorphic hardware using binary stochastic devices, including spintronics and memristive devices, are rapidly growing recently. Considering the above, we expect these devices with the algorithm proposed here will be promising candidates to realize energy-efficient neuromorphic hardware.

One of the possible drawbacks of the algorithm developed here is the requirement of extra hardware that would be needed to implement the multiple synapses and dendritic neurons. To evaluate the balance between the benefits and the possible drawbacks and to improve the algorithm to reduce the possible extra hardware requirement must be an important future subject.

The derived evolution equation for neurons has a term that results in the retrograde modulation of the excitability of a neuron by its succeeding neurons. Due to this term, the algorithm acts as a stochastic variant of algorithms with error backpropagation through which desired outputs provided to a part of the network can spread over the entire network. Recall that a desired output is given to the network by fixing the states of the corresponding visible neurons to the desired value. Assume that the k th neuron is one of these neurons. Then x_{dk} gives the desired value, and the term $x_{dk} - \sigma(v_{dk})$ in the bias b_{dj} of the j th neuron gives the difference between the desired value and the expected value of x_{dk} when the k th neuron is not fixed, which is identical to the error in backpropagation learning when the squared error is used as the loss function. (Recall that x_{dk} and v_{dk} are stochastic variables when the neuron is not fixed.) The term b_{dj} accumulates these errors

of the succeeding neurons of the j th neuron and regulates the excitability of this neuron in such a manner to reduce the errors. Even if the k th neuron is not in the output layer, the term $x_{dk} - \sigma(v_{dk})$ still represents an error of that neuron because this represents the difference between current and expected values of the neuron in the cases that the network does and does not receive the desired outputs, respectively.

In contrast to backpropagation learning, the retrograde modulation requires neither synchronous nor precisely coordinated operations, which are major reasons that backpropagation has not been regarded as the learning principle of the brain. Furthermore, the retrograde modulation need not be immediately affected by postsynaptic action potentials but, rather, can slowly integrate their effects. This assumption seems biologically plausible, as it can be implemented in real cortical circuits. To our knowledge, experiments to verify the existence of such slow retrograde modulation of the excitability of neurons have not yet been attempted. The mechanisms regarded as likely to be responsible for this behavior include the combination of action potential backpropagation from soma to dendritic spines [87] and retrograde transsynaptic transport of certain chemicals [88], glia-mediated modulation [89], and disynaptic connections from postsynaptic to presynaptic neurons. However, note that, while the slow time constants of the modulation r_b and the synaptic latent variable r_q work efficiently to train the network, we have not analytically derived them as an integral part of the theory. Instead, we have imposed them to the model to convert the discrete time steps of the sampling algorithm onto continuous time evolution and make the algorithm biologically plausible. Further study is needed for the analytical treatment of the time constants.

Here we summarize existing representative learning algorithms of neural networks from the stochasticity of their variables and discuss their relationships to ours. Regarding stochasticity, we can categorize existing neural network models into three: deterministic neural networks whose neurons and connection weights are deterministic, Boltzmann machines whose neurons are stochastic, but connection weights are deterministic, and Bayesian neural networks whose connections but not neurons are stochastic. One of the most efficient learning algorithms for deterministic neural networks is backpropagation learning, a gradient descent optimization of connection weights to minimize a given loss function. While the algorithm does not appear fully compatible with experimental observations of the biological brain, various solutions have been proposed to alleviate the problem [27–39].

Gradient-based optimization is also used to train Boltzmann machines because their connection weights are still deterministic, whereas neurons are stochastic [2]. However, to match the distribution of the stochastic neurons to the desired distribution, each iteration of the gradient learning requires the mean and the covariance of the neural activities under the current values of the connection weights, which makes the algorithm impractically time-consuming because the calculation of the statistics requires many samples of variables to converge. The contrastive divergence algorithm has been proposed to ease the difficulty [90]. However, its scope is limited to the restricted Boltzmann machine, where neurons form a bipartite graph.

Unlike the above two, the connection weights of Bayesian neural networks are stochastic. Thus, the desired weights are characterized by their posterior distribution conditioned on given datasets rather than fixed values. However, the computation of the posterior distribution is generally intractable due to its computationally demanding normalization factor. The best-known way to solve the problem is variational Bayesian inference, which approximates the posterior distribution with a parametrized proxy distribution, called variational distribution [91]. The approximation results in optimization of the parameters of the variational distribution to maximize the evidence lower bound, or the negative variational free energy, of the variational distribution. Hence, again, the learning results in gradient-based optimization of nonstochastic parameters that characterize the distribution. The typically used efficient algorithm of this includes Bayesian backpropagation [92,93].

The model proposed here shares the learning objective, the posterior weight distribution, with Bayesian neural networks as connection weights are stochastic, whereas computation of the distribution is generally intractable, as mentioned above. Our network provided a different solution to the difficulty by combining the binary representation of synapses, stochastic neurons, and Gibbs sampling. Binary synapses simplify the posterior distribution to the Bernoulli distribution, allowing us to analyze this via a similar procedure with the logistic regression [94]. However, the introduction of binary synapses alone does not fully solve the problem because it still requires backprop optimization of parameters of the Bernoulli distribution. Concurrent use of Gibbs sampling and stochastic neurons is the remedy. Gibbs sampling replaces computation of the posterior distribution by local and asynchronous updates of variables. Then, the introduction of stochastic neurons enables error signals to propagate stochastically to the entire network by using the retrograde modulation of the firing probability of the neurons. Thus, the model integrates the Bayesian neural network and Boltzmann machine and simultaneously provides a protocol for hierarchical Bayesian inference on stochastic neural networks.

ACKNOWLEDGMENTS

The author thanks to T. Fukai, S. Amari, N. Hiratani, and S. Shinomoto for valuable discussions and comments. This work was partially supported by JSPS KAKENHI Grants No. JP16H01719, No. JP17K00338, No. JP19K12165, and No. JP20K11987.

APPENDIX A: METHODS

1. Learning algorithm

The learning of the network is modeled as a Gibbs sampling of all free variables from their posterior joint distribution conditioned on the fixed variables,

$$P(\{x_{di}\}_{i \in H, d \in D}, \{s_{ijm}\}_{i, j \in N, m \in M} \mid \{x_{di}\}_{i \in V, d \in D}). \quad (\text{A1})$$

The Gibbs sampling allows us to replace the sampling with a repetition of samplings of each single variable from a

posterior distribution conditioned on all other variables,

$$P(x_{dj} | \{x_{di}\}_{i \neq j}, \{s_{ijm}\}_{i,j \in N, m \in M}) \tag{A2}$$

and

$$P(s_{ijm} | \{x_{di}\}_{i \in H, d \in D}, \{s_{klm}\}_{klm \neq ijm}) \tag{A3}$$

for a neuron and a synapse, respectively.

To derive an explicit description of the posterior distribution of a neuron, let us assume the network is a directed acyclic graph (DAG) and consider the log-likelihood ratio for x_{dj} . Using the Bayes rule, we obtain

$$\log \frac{P(x_{dj} = 1 | \dots)}{1 - P(x_{dj} = 1 | \dots)} \tag{A4}$$

$$= \log \frac{P(x_{dj} = 1 | \{x_{di}\}_{i \in p(j)}, \{x_{dk}\}_{k \in s(j)}, \dots)}{P(x_{dj} = 0 | \{x_{di}\}_{i \in p(j)}, \{x_{dk}\}_{k \in s(j)}, \dots)} \tag{A5}$$

$$= \log \frac{P(x_{dj} = 1 | \{x_{di}\}_{i \in p(j)}, \{s_{ijm}\}_{m \in M})}{P(x_{dj} = 0 | \{x_{di}\}_{i \in p(j)}, \{s_{ijm}\}_{m \in M})} \prod_{k \in s(j)} \frac{P(x_{dk} | x_{dj} = 1, \{s_{jkm}\}_{m \in M}, \dots)}{P(x_{dk} | x_{dj} = 0, \{s_{jkm}\}_{m \in M}, \dots)} \tag{A6}$$

$$= \log \frac{\sigma(v_{dj})}{1 - \sigma(v_{dj})} + \sum_{k \in s(j)} \begin{cases} \log \sigma(v_{dk, -j} + w_{jk}) - \log \sigma(v_{dk, -j}), & x_{dk} = 1 \\ \log(1 - \sigma(v_{dk, -j} + w_{jk})) - \log(1 - \sigma(v_{dk, -j})), & x_{dk} = 0 \end{cases} \tag{A7}$$

$$= v_{dj} + \sum_k \begin{cases} (1 - \sigma(v_{dk}))w_{jk}, & x_{dk} = 1 \\ -\sigma(v_{dk})w_{jk}, & x_{dk} = 0 \end{cases} \tag{A8}$$

$$= v_{dj} + \sum_k w_{jk}(x_{dk} - \sigma(v_{dk})), \tag{A9}$$

where the dots represent all variables other than explicitly given variables, $p(j)$ and $s(j)$ denote sets of the preceding and the succeeding neurons of the j th neuron, respectively. Owing to the assumption that the network is DAG, we can divide neurons into the disjoint sets of the preceding and succeeding neurons for the i th neuron and safely factorize posterior distributions of (A.5) to obtain (A.6). Here, we used Bayes' rule $P(A | B, C) \propto P(B | A, C)P(A | C)$ and conditional independence of variables x_{dk} as

$$\begin{aligned} &P(x_{dj} | \{x_{di}\}_{i \in p(j)}, \{x_{dk}\}_{k \in s(j)}, \dots) \\ &\propto P(\{x_{dk}\}_{k \in s(j)} | x_{dj}, \{x_{di}\}_{i \in p(j)}, \dots)P(x_{dj} | \{x_{di}\}_{i \in p(j)}, \dots) \\ &= P(x_{dj} | \{x_{di}\}_{i \in p(j)}, \{s_{ijm}\}_{i \in p(j), m \in M}) \\ &\quad \prod_{k \in s(j)} P(x_{dk} | x_{dj}, \{x_{dl}\}_{l \in p(k) \setminus j}, \{s_{lkm}\}_{l \in p(k), m \in M}). \end{aligned}$$

Equation (A7) follows from the fact that the firing probability of the k th neuron conditioned on $x_{dj} = 1$ is $\sigma(v_{dk, -j} + 1 \cdot w_{jk})$ while it conditioned on $x_{dj} = 0$ is $\sigma(v_{dk, -j} + 0 \cdot w_{jk})$, where $v_{dk, -j} = v_{dk} - x_{dj}w_{jk}$. To obtain (A8), we assumed that $\|v_{dk, -j}\| \gg \|w_{jk}\|$ and linearized each term of the summation of (A7) with respect to w_{jk} . Solving the above equation for $P(x_{dj} = 1 | \dots)$, we obtain Eq. (3) of the result section with

$$b_{dj} = \sum_k w_{jk}(x_{dk} - \sigma(v_{dk})) \tag{A10}$$

as the posterior distribution (i.e., the stochastic update rule) of the neuron.

Similarly, the log-likelihood ratio for the synapse s_{ijm} is

$$\log \frac{P(s_{ijm} = 1 | \dots)}{1 - P(s_{ijm} = 1 | \dots)} \tag{A11}$$

$$= \log \frac{P(s_{ijm} = 1 | \dots)}{P(s_{ijm} = 0 | \dots)} \tag{A12}$$

$$= \log \frac{P(s_{ijm} = 1)}{P(s_{ijm} = 0)} \prod_d \frac{P(x_{dj} | s_{ijm} = 1, \dots)}{P(x_{dj} | s_{ijm} = 0, \dots)} \tag{A13}$$

$$= q_{0,ijm} + \sum_d \begin{cases} \log \sigma(v_{dj, -im} + x_{di}a_{ijm}) - \log \sigma(v_{dj, -im}), & x_{dj} = 1 \\ \log(1 - \sigma(v_{dj, -im} + x_{di}a_{ijm})) - \log(1 - \sigma(v_{dj, -im})), & x_{dj} = 0 \end{cases} \tag{A14}$$

$$= q_{0,ijm} + \sum_d \begin{cases} (1 - \sigma(v_{dj}))x_{di}a_{ijm}, & x_{dj} = 1 \\ -\sigma(v_{dj})x_{di}a_{ijm}, & x_{dj} = 0 \end{cases} \tag{A15}$$

$$= q_{0,ijm} + a_{ijm} \sum_d x_{di}(x_{dj} - \sigma(v_{dj})), \tag{A16}$$

where the dots represent all variables other than explicitly given variables, i.e., $\{x_{di}\}$ and $\{s_{klm}\}_{klm \neq ijm}$, and we have $v_{dj,-im} = v_{dj} - x_{di}s_{ijm}a_{ijm}$. To obtain (A15), we have assumed $\|v_{dj,-im}\| \gg \|a_{ijm}\|$, and approximated each term in the summation as a quantity linear in a_{ijm} . The constant $q_{0,ijm}$ represents the log-likelihood ratio of the prior distribution $P(s_{ijm})$, which vanishes unless the prior distribution is biased. We assumed $q_{0,ijm} = 0$ throughout the work. Solving the above equation for $P(s_{ijm} = 1 | \dots)$ gives Eq. (5) and

$$q_{ij} = \sum_d x_{di}(x_{dj} - \sigma(v_{dj})) \quad (\text{A17})$$

which constitute the explicit description of the posterior distribution or the update rule of the synapse. While the derived update rules of neurons and synapses are exact only when the network is a DAG, we numerically confirmed that they work well even on recurrent networks, as far as the connection probability of the networks is not very high.

Equation (A10) implies that a spike of a neuron immediately changes b_{dj} and excitability of presynaptic neurons. However, such immediate retrograde modulation has not been experimentally reported and seems biologically implausible. Rather, it is biologically more natural that b_{dj} evolves slowly while accumulating the effects of postsynaptic spikes as Eq. (4), where r_b characterizes the timescale of the evolution. [Here, r_b satisfies $0 < r_b \leq 1$, while in the case $r_b = 1$, Eq. (4) reproduces to Eq. (A10).] Thus, b_{dj} is determined by the average spike history of the postsynaptic neurons over a finite, presumably quite long, duration. Similarly, we can also generalize the evolution equation for q_{ij} as Eq. (6). As demonstrated in Figs. 2(e) and 2(f), these generalizations rarely decrease the learning accuracy of the algorithm.

2. Numerical simulations

All numerical simulations are written in Python, with the open-source matrix library CuPy. In details of the procedures to train a feedforward network are as follows. (i) We first prepare ND binary variables for the N neurons and WM synapses where D is the number of data in the training dataset, W is the number of connections in the network, and M is the number of synapses per connection. (ii) Then we fix the variables of the visible neurons to the values of the data in the training dataset, and initialize the values of the hidden neurons and synapses randomly to 0 or 1 with probability 1/2. (iii) To avoid perfectly synchronized updates and to mimic asynchronous updates of variables, we randomly choose the ratio r_x for the hidden neurons and update their variables according to Eqs. (3) and (4). (iv) Similarly, we randomly choose the ratio r_s for the WM synapses to update according to Eqs. (5) and (6). (v) We repeat (iii) and (iv) as many times as desired. The procedure to obtain the prediction of the network is the same as that for the training procedure, except that we fix only the input neurons and update the hidden and output neurons according to Eqs. (3) and (4), keeping the synaptic values fixed. During the prediction procedure, to accelerate the computation, we can use the average activities of the neurons, $\sigma(v_{dj})$, instead of their binary variables, x_{dj} , and omit the biases, b_{dj} . The training and test accuracies are defined as the ratio of the number of inputs that enables the network to

generate the correct outputs to the total numbers of inputs of the training and test datasets, respectively.

3. Dataset

Except in the cases described by Figs. 2 and 8, we used the MNIST dataset, which consists of a training dataset of 60 000 examples and a test dataset of 10 000 examples in which each image has $28 \times 28 = 784$ pixels. Because pixels in the MNIST data range from 0 to 255, we replaced them with 0 or 1, depending on whether the value of the pixel is below or above $255/2$. We thus obtain 784-dimensional binary input vectors.

In the situation considered in Fig. 2, we trained a three-layered network consisting of 40 input, 40 hidden, and two output neurons to learn a simple task that is a noisy and high-dimensional variant of the XOR problem. The datasets were artificially generated as follows. We first prepared two-dimensional binary vectors (X_{d1}, X_{d2}) , where d is the data index of the dataset. Then, to obtain 40-dimensional binary input vectors (Y_{d1}, \dots, Y_{d40}) , we set $Y_{di} = X_{d1}$ for $i = 1, \dots, 20$ and $Y_{di} = X_{d2}$ for $i = 21, \dots, 40$, and then flipped their values randomly with a probability of $p_{\text{flip}} = 0.1$ to obtain randomized input datasets. The desired outputs of the two-dimensional vectors are given by $Z_{di} = (1, 0)$ if $\text{XOR}(X_{d1}, X_{d2}) = 0$ and $(0, 1)$ if $\text{XOR}(X_{d1}, X_{d2}) = 1$. The training dataset and test dataset each contains 400 examples.

In the situation considered in Fig. 8, we used datasets consisting of temporal sequences to train recurrent networks. Let us write the input data and desired outputs at time t as $X_{di}(t)$ and $Z_{di}(t)$. These are fed into the network by fixing the neurons in the input layer as $x_{0,di}(t) = X_{di}(t)$, $0 < i \leq N_0$, and those in the output layer as $x_{2,di}(t) = Z_{di}(t)$, $0 < i \leq N_2$, where N_0 and N_2 are numbers of neurons in the input and output layers, respectively. A dataset is prepared as follows. We first prepare an integer sequence $s(t)$, where $0 \leq s(t) \leq S$ and $1 \leq t \leq T$. We then set $X_i(t) = 1$ if $s(t) > 0$ and $(s(t) - 1)N_0/S < i \leq s(t)N_0/S$ and $X_i(t) = 0$ otherwise. The desired output is set as $Z_{di}(t) = X_i(t + 1)$ if $1 \leq t < T$ and $Z_{di}(T) = X_i(0)$ otherwise. To obtain randomized input vectors $X_{di}(t)$, we replicated $X_i(t)$ and randomly flipped them as $X_{di}(t) = X_i(t)$ with probability 0.95 and $X_{di}(t) = 1 - X_i(t)$ with probability 0.05. The integer sequence $s(t)$ was “1, 1, 5, 5, 6, 6, 5, 0, 4, 4, 3, 3, 2, 2, 1, 0” (that is taken from the melody of “Twinkle, Twinkle, Little Star”) with $N_0 = 300$, $S = 6$ and $T = 16$.

4. Parameters

We used $r_q = 1.0$, $r_s = 0.001$, $M = 200$, $a_0 = 0.1$, and $D = 100$ in the situation considered in Fig. 1(f), and $r_b = r_q = r_x = 1.0$, $r_s = 0.001$, $M = 50$, $a_0 = 0.5$, and $D = 400$ in the situation considered in Fig. 2. In the situation considered in the remaining figures, except Figs. 6–8, we used $r_b = 0.01$, $r_x = 0.9$, $r_q = 0.1$, $r_s = 0.001$, $M = 100$, and $a_0 = 0.1$. In the case of Fig. 6, we use $M = 200$, and in the case of Fig. 7, we used $r_x = 0.5$, $a_0 = 0.1$, and in the case of Fig. 8, we used $r_b = 0.1$, $r_x = 1$, $r_q = 0.005$, $r_s = 0.005$, and $D = 2000$. The numbers of hidden neurons in the three-layered network that are not specified in the figures are 1000 for Figs. 3(f) and 6,

100 for Fig. 7, and 500 for Fig. 8. Connection probability was 1.0 and 0.5 for feedforward connections and recurrent connections, respectively. In the case of Fig. 5, we randomly assign a value $g_i = +1$ or -1 to all hidden neurons with the probability of $1/2$. Then, using the assigned value, we set the amplitude of synapses as $a_{ijm} = g_i a'_m$ where a'_m is an evenly spaced sequence from 0 to a_0 .

5. Statistical analysis

a. Spectrum variance of principal components

After we trained the three-layered neural network considered in Fig. 3(a) using the MNIST dataset, we fixed the synapses and obtained the average activities of the hidden neurons across the training dataset $\{\sigma(v_{dj})\}_{d \in D, j \in H}$. Note that the quantities v_{dj} for the hidden neurons were deterministic in this case because both the input neurons and the synaptic connections from them were fixed. The principal component analysis was applied to the average activities after they were standardized. Then we obtained the explained variance of each principal component, which is the eigenvalue of the covariance matrix of the standardized average activities and ordered them in descending order. Namely, we obtained the eigenspectrum of the covariance matrix $C_{ij} = \frac{1}{D} \sum_{d=1}^D \sigma_{di} \sigma_{dj}$, where $\sigma_{di} = (\sigma(v_{di}) - \mu_i) / \chi_i$, $\mu_i = \frac{1}{D} \sum_d \sigma(v_{di})$, and $\chi_i^2 = \frac{1}{D} \sum_d (\sigma(v_{di}) - \mu_i)^2$. The exponent of the power law was estimated with a least-square linear fit of the variance spectrum in log-log space.

b. Statistics of network motifs

We trained a three-layered recurrent network with 100 hidden neurons. Then, we measured the number of connection patterns among the triplets of neurons over all possible combinations of 3 neurons chosen from 100, i.e., for $(100 \times 99 \times 98)/6 = 161\,700$ triples. Here, we only counted connections whose synaptic weights were greater than or equal to 0.27, to exclude small and negative connections. See Appendix D for the relationship between the threshold size and the connection weight distribution and results using different threshold values. Connection weights that are compared against the threshold 0.27 are determined from a sample of synaptic variables of each network after training. The null hypothesis of the counts is defined in the same way provided in the paper [77]. Namely, we determined the numbers of unidirectional and bidirectional connections in all pairs of neurons and calculated the predicted number of three-neuron patterns by assuming all constituent pairs of neurons in each triplet pattern are chosen independently, while maintaining the probabilities of the measured unidirectional and bidirectional connections. We performed 20 learning trials to obtain the mean and standard deviation, σ , of the ratio of the actual number of each triplet pattern to that obtained with the null hypothesis.

APPENDIX B: OTHER POSSIBILITIES OF SYNAPTIC CONSTANTS

We demonstrate other possibilities of constants $\{a_k\}$ of the synaptic strength instead of the equally spaced sequence used in the main text. The first is the geometric se-

quence with powers to 2, the most efficient expression of scalar numbers by the weighted sum of binary values. Considering sign of the representation, we have used 10 synapses for each connection, 5 positive and 5 negative values, where $\{a_k\} = \{2^0, 2^{-1}, 2^{-2}, 2^{-3}, 2^{-4}\} \cup \{-2^0, -2^{-1}, -2^{-2}, -2^{-3}, -2^{-4}\}$. Thus, each connection weight ranges from about -2 to 2 with the step of $1/16$. Figure 9(a) shows the development of training and test accuracies that corresponds with Fig. 3(a) in the main text, and Fig. 9(b) shows the evolution of several synaptic weights. We see that the accuracies stably increase along with the learning iteration while the reached values are slightly lower than those of Fig. 3(a). We speculate that the slight decrease in the performance is due to the lack of redundancy of the binary expression of the connection weights. Note that processing time and required memory capacity are largely reduced due to the smallness of the number of synapses here.

The second case is one synapse for each connection, the most compact but less expressive. We randomly set the amplitude of the synapses to -10 or 10 with the probability of $1/2$. (Note that this does not satisfy Dale's law because positive and negative synapses sharing a presynaptic neuron coexist.) Figures 9(c) and 9(d) shows the result of the learning. The algorithm works decently even for the simplest case, whereas achieved performance is not very high, indicating that using multiple synapses per connection is not the essential requirement for the algorithm while it improves the learning performance.

APPENDIX C: POWER-LAW DECAY OF THE VARIANCE SPECTRUM FOR ANOTHER DATASET

Figure 10 shows the variance spectrum of the principal components of the mean population activity of hidden neurons across the training dataset of the fashion-MNIST dataset. Again, we see that the variance spectrum follows the power law with the exponent slightly less than -1 . The power-law exponent obtained here is -1.04 , which is just the same value reported in the visual cortex experiment. The result, along with the result of Fig. 6(a) in the main text, implies that the critical power-law spectrum obtained here is not a specific outcome given by the MNIST dataset.

APPENDIX D: CONNECTIVITY PATTERNS OF TRIPLETS OF NEURONS FOR DIFFERENT THRESHOLD VALUES

The panel Fig. 11(a) shows the histogram of recurrent connection weights and the threshold 0.27 used in the main text to exclude weak connections when we analyzed connectivity patterns of triplets of neurons in the recurrent network. Panels in Figs. 11(b)–11(e) show connectivity patterns, the same with Fig. 7(c), but for different threshold values from the value 0.27 used in the main text. We see that the ratios of the actual counts of connected patterns, i.e., the patterns from 10 to 16, to the chance level increase as the threshold increases, while they decrease and approach to unity, indicating the chance level as the threshold decreases. This result means strong connections exist in the network more nonrandomly, whereas

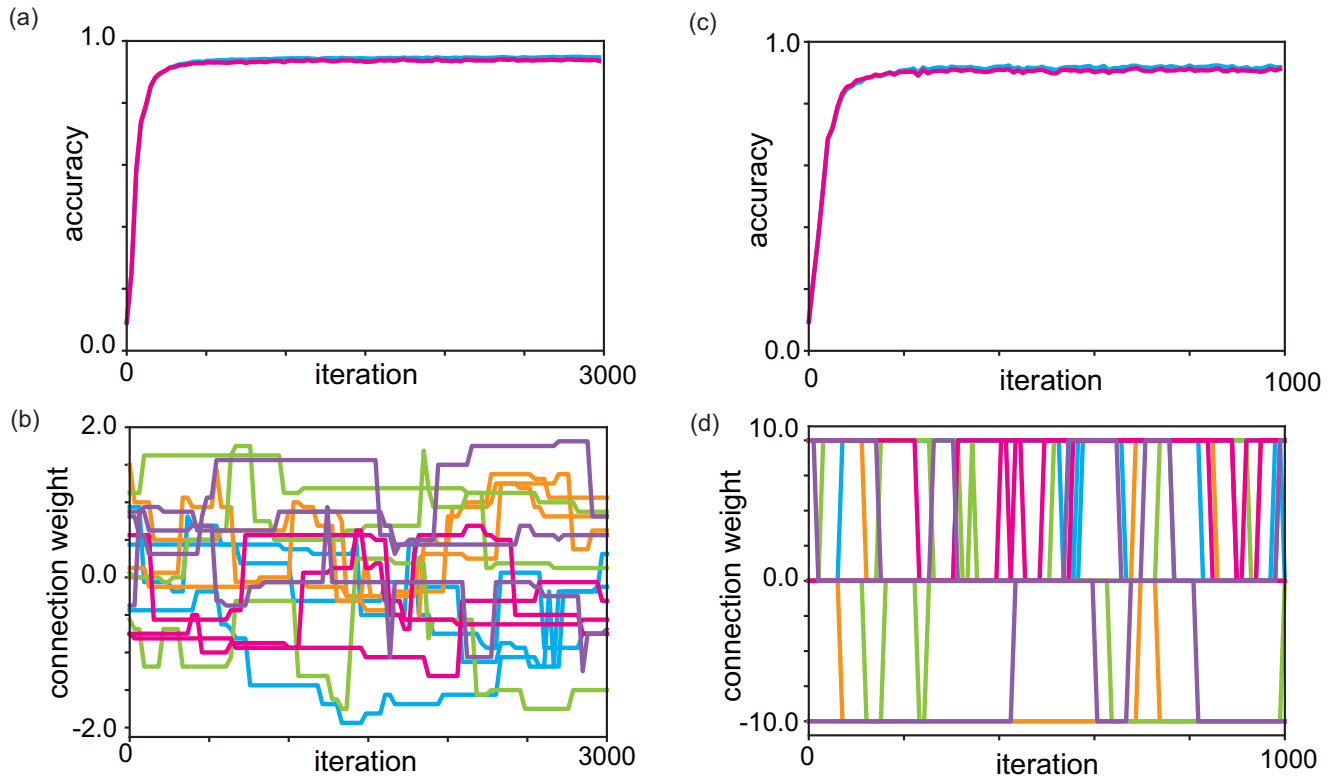


FIG. 9. Other possibilities of synaptic constants a_k instead of the one used in the main text. [(a),(b)] Geometric sequence with powers to 2. (a) Training (cyan) and test (magenta) accuracies as functions of sampling iteration. Their reached values are 94.72% and 93.56%, respectively. (b) evolution of connection weights. [(c),(d)] The same as [(a),(b)] but for the case of one synapse for each connection. Reached values of training and test accuracies are 91.82% and 91.22%, respectively.

weak connections are distributed randomly, consistent with the experimental observation. However, please note that we set the threshold value arbitrarily, and it does not fully agree with a small value of the threshold used in the experiment. We speculate that the discrepancy can be due to the limit of detection for weak synapses. While weak connections are difficult

to be detected in experiments, all connections, including weak synapses, are detectable in numerical simulations.

APPENDIX E: AN ONLINE IMPLEMENTATION OF THE LEARNING ALGORITHM

So far, to accelerate computational speed, we have performed all numerical simulations of the algorithm using an implementation in which neurons that receive different data of a dataset update their variables parallelly, and synapses sum up these values to update their variables, as is given in the summation across data index d of equation (6) in the main text. However, in the brain, each datum of the dataset is given to the network serially in time rather parallelly, and the synapses must accumulate neural states over a relatively long time to update their state. Such online implementation of the algorithm can also be beneficial for neuromorphic computing, besides neuroscience, because it may allow the development of edge devices that continuously learn from information available over time, as animals and humans realize lifelong learning in an ever-changing environment.

Here we provide an online implementation of the algorithm in which a sequence of data is fed to a network, and neurons and synapses randomly update their states over time with designated frequencies. The algorithm is almost the same as equations from (3) to (6) in the main text, except that neural variables do not have the data subscript d , and the latent

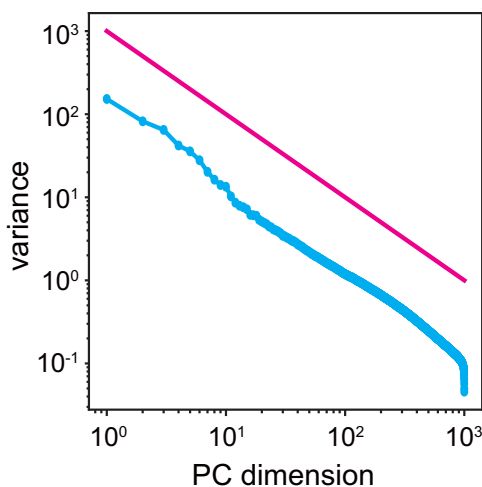


FIG. 10. The same as Fig. 6(a) of the main text but for the fashion-MNIST dataset.

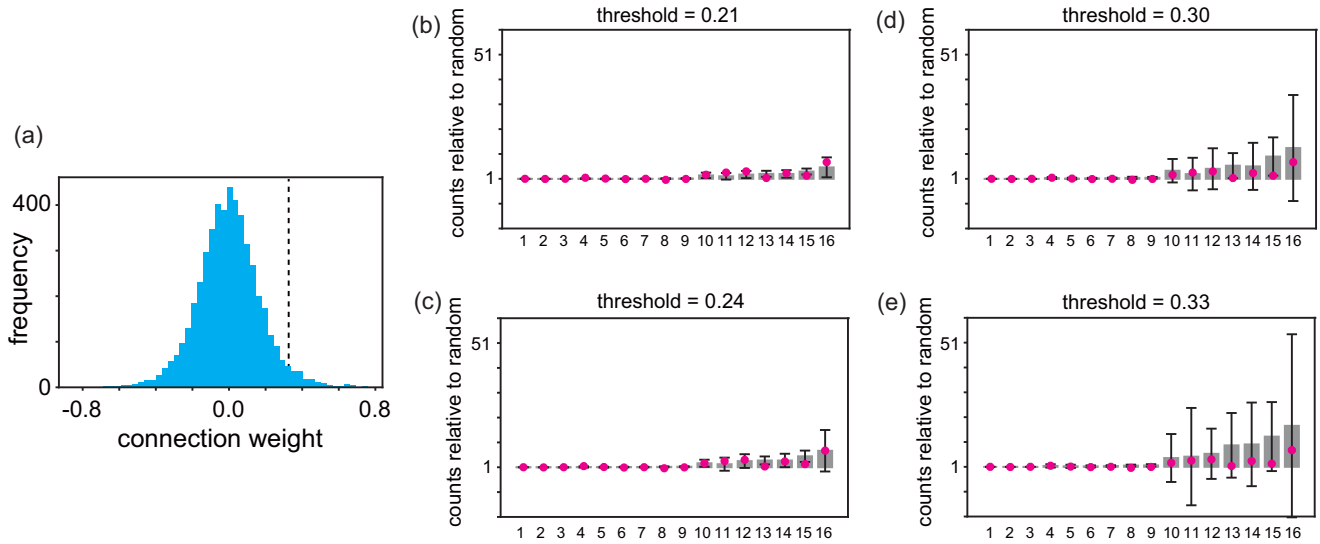


FIG. 11. Connectivity patterns of triplets of neurons in the recurrent network. (a) The histogram of recurrent connection weights with the threshold 0.27 (dashed line) used in the main text. [(b)–(e)] The same as Fig. 7(c) but for different values of the threshold from 0.27: (b) 0.21, (c) 0.24, (d) 0.30, and (e) 0.33. Note that magenta points indicating experimental evidence are identical across thresholds, including Fig. 7(c) in the main text.

synaptic variables q do not explicitly sum up neural variables across data. Rather, they accumulate neural states over time. Thus, we obtain

$$P(x_j = 1 | \dots) = \sigma(v_j + b_j) \tag{E1}$$

$$b_j(t + 1) = (1 - r_b)b_j(t) + r_b \sum_k w_{jk}(x_k - \sigma(v_k)) \tag{E2}$$

for a neuron and

$$P(s_{ijm} = 1 | \dots) = \sigma(a_{ijm}q_{ij}) \tag{E3}$$

$$q_{ij}(t + 1) = (1 - r_q)q_{ij}(t) + x_i(x_j - \sigma(v_j)) \tag{E4}$$

for a synapse.

Figure 12 shows the results of a numerical simulation that trains a feedforward network in a supervised manner using the online learning algorithm. The network learns the same type of

artificial dataset we have used for Fig. 2, i.e., a noisy and high-dimensional variant of the XOR problem in the main text, while here, unlike Fig. 2, the data is serially given to the network. During the learning, we randomly chose a datum from the dataset every 20 timesteps and continued to present it to the network until the next datum was randomly chosen. Note that hidden neurons and synapses just repeat random updates of their variables without resetting variables or switching between different computations during learning. Parameters used for the simulation are the same as Fig. 2(a) in the main text, except that we use a smaller dataset with $D = 200$ and $p_{\text{flip}} = 0.05$, and the lower, i.e., slower, rate constants of synaptic evolutions $r_q = 10^{-3}$ and $r_s = 10^{-5}$. Similar to the results of Fig. 2, we can see that training and test accuracies almost monotonically increase until they reach values close to one while connection weights continue to fluctuate during learning, whereas the learning requires a longer time than the parallel implementation, as expected.

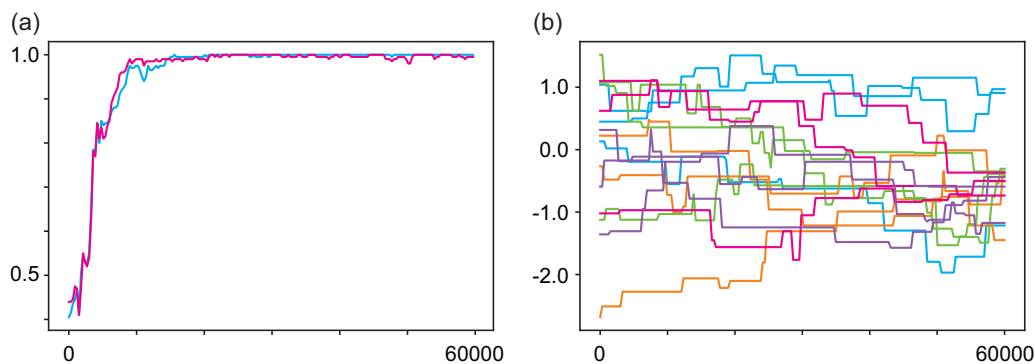


FIG. 12. Supervised learning of a feedforward network using an online implementation of the learning algorithm. (a) Training (cyan) and test (magenta) of a three-layered network as functions of time. (b) Evolution of randomly chosen synaptic weights of the network. Similar to Fig. 2 in the main text, different colors are used for different presynaptic neurons.

- [1] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, *Nature (London)* **521**, 436 (2015).
- [2] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, A learning algorithm for Boltzmann machines, *Cognit. Sci.* **9**, 147 (1985).
- [3] D. Marković, A. Mizrahi, D. Querlioz, and J. Grollier, Physics for neuromorphic computing, *Nat. Rev. Phys.* **2**, 499 (2020).
- [4] T. Pfeil, J. Jordan, T. Tetzlaff, A. Grübl, J. Schemmel, M. Diesmann, and K. Meier, Effect of Heterogeneity on Decorrelation Mechanisms in Spiking Neural Networks: A Neuromorphic-Hardware Study, *Phys. Rev. X* **6**, 021023 (2016).
- [5] J. Torrejon, M. Riou, F. A. Araujo, S. Tsunegi, G. Khalsa, D. Querlioz, P. Bortolotti, V. Cros, K. Yakushiji, A. Fukushima *et al.*, Neuromorphic computing with nanoscale spintronic oscillators, *Nature (London)* **547**, 428 (2017).
- [6] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, *Nature (London)* **549**, 195 (2017).
- [7] A. Z. Pervaiz, L. A. Ghantasala, K. Y. Camsari, and S. Datta, Hardware emulation of stochastic p-bits for invertible logic, *Sci. Rep.* **7**, 10994 (2017).
- [8] Z. Wang, S. Joshi, S. E. Savel'ev, H. Jiang, R. Midya, P. Lin, M. Hu, N. Ge, J. P. Strachan, Z. Li *et al.*, Memristors with diffusive dynamics as synaptic emulators for neuromorphic computing, *Nat. Mater.* **16**, 101 (2017).
- [9] N. Gong, T. Idé, S. Kim, I. Boybat, A. Sebastian, V. Narayanan, and T. Ando, Signal and noise extraction from analog memory elements for neuromorphic computing, *Nat. Commun.* **9**, 2102 (2018).
- [10] I. Boybat, M. Le Gallo, S. Nandakumar, T. Moraitis, T. Parnell, T. Tuma, B. Rajendran, Y. Leblebici, A. Sebastian, and E. Eleftheriou, Neuromorphic computing with multi-memristive synapses, *Nat. Commun.* **9**, 2514 (2018).
- [11] A. Mizrahi, T. Hirtzlin, A. Fukushima, H. Kubota, S. Yuasa, J. Grollier, and D. Querlioz, Neural-like computing with populations of superparamagnetic basis functions, *Nat. Commun.* **9**, 1533 (2018).
- [12] Y. van de Burgt, A. Melianas, S. T. Keene, G. Malliaras, and A. Salleo, Organic electronics for neuromorphic computing, *Nat. Electron.* **1**, 386 (2018).
- [13] J. Feldmann, N. Youngblood, C. D. Wright, H. Bhaskaran, and W. H. Pernice, All-optical spiking neurosynaptic networks with self-learning capabilities, *Nature (London)* **569**, 208 (2019).
- [14] W. A. Borders, A. Z. Pervaiz, S. Fukami, K. Y. Camsari, H. Ohno, and S. Datta, Integer factorization using stochastic magnetic tunnel junctions, *Nature (London)* **573**, 390 (2019).
- [15] K. Roy, A. Jaiswal, and P. Panda, Towards spike-based machine intelligence with neuromorphic computing, *Nature (London)* **575**, 607 (2019).
- [16] M. Mahmoodi, M. Prezioso, and D. Strukov, Versatile stochastic dot product circuits based on nonvolatile memories for high performance neurocomputing and neurooptimization, *Nat. Commun.* **10**, 5113 (2019).
- [17] E. J. Fuller, S. T. Keene, A. Melianas, Z. Wang, S. Agarwal, Y. Li, Y. Tuchman, C. D. James, M. J. Marinella, J. J. Yang *et al.*, Parallel programming of an ionic floating-gate memory array for scalable neuromorphic computing, *Science* **364**, 570 (2019).
- [18] R. Hamerly, L. Bernstein, A. Sludds, M. Soljačić, and D. Englund, Large-Scale Optical Neural Networks Based on Photoelectric Multiplication, *Phys. Rev. X* **9**, 021032 (2019).
- [19] S. Zhang and Y. Tserkovnyak, Antiferromagnet-Based Neuromorphics Using Dynamics of Topological Charges, *Phys. Rev. Lett.* **125**, 207202 (2020).
- [20] J. Grollier, D. Querlioz, K. Camsari, K. Everschor-Sitte, S. Fukami, and M. D. Stiles, Neuromorphic spintronics, *Nat. Electron.* **3**, 360 (2020).
- [21] W. Zhang, B. Gao, J. Tang, P. Yao, S. Yu, M.-F. Chang, H.-J. Yoo, H. Qian, and H. Wu, Neuro-inspired computing chips, *Nat. Electron.* **3**, 371 (2020).
- [22] V. K. Sangwan and M. C. Hersam, Neuromorphic nanoelectronic materials, *Nat. Nanotechnol.* **15**, 517 (2020).
- [23] P. Debashis, V. Ostwal, R. Faria, S. Datta, J. Appenzeller, and Z. Chen, Hardware implementation of Bayesian network building blocks with stochastic spintronic devices, *Sci. Rep.* **10**, 16002 (2020).
- [24] K. Y. Camsari, M. M. Torunbalci, W. A. Borders, H. Ohno, and S. Fukami, Double-Free-Layer Magnetic Tunnel Junctions for Probabilistic Bits, *Phys. Rev. Appl.* **15**, 044049 (2021).
- [25] B. J. Shastri, A. N. Tait, T. F. de Lima, W. H. Pernice, H. Bhaskaran, C. D. Wright, and P. R. Prucnal, Photonics for artificial intelligence and neuromorphic computing, *Nat. Photonics* **15**, 102 (2021).
- [26] B. Kiraly, E. J. Knol, W. M. van Weerdenburg, H. J. Kappen, and A. A. Khajetoorians, An atomic Boltzmann machine capable of self-adaption, *Nat. Nanotechnol.* **16**, 414 (2021).
- [27] Y. Bengio, D.-H. Lee, J. Bornschein, T. Mesnard, and Z. Lin, Towards biologically plausible deep learning, [arXiv:1502.04156](https://arxiv.org/abs/1502.04156).
- [28] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, Random synaptic feedback weights support error backpropagation for deep learning, *Nat. Commun.* **7**, 13276 (2016).
- [29] J. H. Lee, T. Delbruck, and M. Pfeiffer, Training deep spiking neural networks using backpropagation, *Front. Neurosci.* **10**, 508 (2016).
- [30] M. Pfeiffer and T. Pfeil, Deep learning with spiking neurons: Opportunities and challenges, *Front. Neurosci.* **12**, 774 (2018).
- [31] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, Spatio-temporal backpropagation for training high-performance spiking neural networks, *Front. Neurosci.* **12**, 331 (2018).
- [32] A. Tavaneai, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, Deep learning in spiking neural networks, *Neural Networks* **111**, 47 (2019).
- [33] E. O. Nefci, H. Mostafa, and F. Zenke, Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks, *IEEE Signal Process. Mag.* **36**, 51 (2019).
- [34] R. Kim, Y. Li, and T. J. Sejnowski, Simple framework for constructing functional spiking recurrent neural networks, *Proc. Natl. Acad. Sci. U.S.A.* **116**, 22811 (2019).
- [35] D. Rotermund and K. R. Pawelzik, Back-propagation learning in deep spike-by-spike networks, *Front. Comput. Neurosci.* **13**, 55 (2019).
- [36] G. Bellec, F. Scherr, A. Subramoney, E. Hajek, D. Salaj, R. Legenstein, and W. Maass, A solution to the learning dilemma for recurrent networks of spiking neurons, *Nat. Commun.* **11**, 3625 (2020).

- [37] C. Lee, S. S. Sarwar, P. Panda, G. Srinivasan, and K. Roy, Enabling spike-based backpropagation for training deep neural network architectures, *Front. Neurosci.* **14**, 119 (2020).
- [38] S. Woźniak, A. Pantazi, T. Bohnstingl, and E. Eleftheriou, Deep learning incorporating biologically inspired neural dynamics and in-memory computing, *Nat. Mach. Intell.* **2**, 325 (2020).
- [39] T. C. Wunderlich and C. Pehle, Event-based backpropagation can compute exact gradients for spiking neural networks, *Sci. Rep.* **11**, 12829 (2021).
- [40] G. E. Hinton and R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, *Science* **313**, 504 (2006).
- [41] C. Stringer, M. Pachitariu, N. Steinmetz, M. Carandini, and K. D. Harris, High-dimensional geometry of population responses in visual cortex, *Nature (London)* **571**, 361 (2019).
- [42] J. Nassar, P. A. Sokol, S. Chung, K. D. Harris, and I. M. Park, On 1/n neural representation and robustness, *Adv. Neural Inf. Process. Syst.*, **33**, 6211 (2020).
- [43] N. C. Kong, E. Margalit, J. L. Gardner, and A. M. Norcia, Increasing neural network robustness improves match to macaque v1 eigenspectrum, spatial frequency preference and predictivity, *PLoS Comput. Biol.* **18**, e1009739 (2022).
- [44] M. N. Shadlen and W. T. Newsome, Noise, neural codes and cortical organization, *Curr. Opin. Neurobiol.* **4**, 569 (1994).
- [45] A. Arieli, A. Sterkin, A. Grinvald, and A. Aertsen, Dynamics of ongoing activity: explanation of the large variability in evoked cortical responses, *Science* **273**, 1868 (1996).
- [46] T. Kenet, D. Bibitchkov, M. Tsodyks, A. Grinvald, and A. Arieli, Spontaneously emerging cortical representations of visual attributes, *Nature (London)* **425**, 954 (2003).
- [47] P. Berkes, G. Orbán, M. Lengyel, and J. Fiser, Spontaneous cortical activity reveals hallmarks of an optimal internal model of the environment, *Science* **331**, 83 (2011).
- [48] A. Pouget, J. M. Beck, W. J. Ma, and P. E. Latham, Probabilistic brains: Knowns and unknowns, *Nat. Neurosci.* **16**, 1170 (2013).
- [49] M. Llera-Montero, J. Sacramento, and R. P. Costa, Computational roles of plastic probabilistic synapses, *Curr. Opin. Neurobiol.* **54**, 90 (2019).
- [50] D. D. Stettler, H. Yamahachi, W. Li, W. Denk, and C. D. Gilbert, Axons and synaptic boutons are highly dynamic in adult visual cortex, *Neuron* **49**, 877 (2006).
- [51] N. Yasumatsu, M. Matsuzaki, T. Miyazaki, J. Noguchi, and H. Kasai, Principles of long-term dynamics of dendritic spines, *J. Neurosci.* **28**, 13592 (2008).
- [52] A. Holtmaat and K. Svoboda, Experience-dependent structural synaptic plasticity in the mammalian brain, *Nat. Rev. Neurosci.* **10**, 647 (2009).
- [53] X. Chen, U. Leischner, N. L. Rochefort, I. Nelken, and A. Konnerth, Functional mapping of single spines in cortical neurons in vivo, *Nature (London)* **475**, 501 (2011).
- [54] A. Attardo, J. E. Fitzgerald, and M. J. Schnitzer, Impermanence of dendritic spines in live adult cal hippocampus, *Nature (London)* **523**, 592 (2015).
- [55] G. Mongillo, S. Rumpel, and Y. Loewenstein, Intrinsic volatility of synaptic connections—a challenge to the synaptic trace theory of memory, *Curr. Opin. Neurobiol.* **46**, 7 (2017).
- [56] V. Pascoli, A. Hiver, R. Van Zessen, M. Loureiro, R. Achargui, M. Harada, J. Flakowski, and C. Lüscher, Stochastic synaptic plasticity underlying compulsion in a model of addiction, *Nature (London)* **564**, 366 (2018).
- [57] C. Allen and C. F. Stevens, An evaluation of causes for unreliability of synaptic transmission, *Proc. Natl. Acad. Sci. U.S.A.* **91**, 10380 (1994).
- [58] L. C. Katz and C. J. Shatz, Synaptic activity and the construction of cortical circuits, *Science* **274**, 1133 (1996).
- [59] T. Branco and K. Staras, The probability of neurotransmitter release: variability and feedback control at single synapses, *Nat. Rev. Neurosci.* **10**, 373 (2009).
- [60] S. Deneve, P. E. Latham, and A. Pouget, Efficient computation and cue integration with noisy population codes, *Nat. Neurosci.* **4**, 826 (2001).
- [61] W. J. Ma, J. M. Beck, P. E. Latham, and A. Pouget, Bayesian inference with probabilistic population codes, *Nat. Neurosci.* **9**, 1432 (2006).
- [62] K. Doya, S. Ishii, A. Pouget, and R. P. Rao, *Bayesian Brain: Probabilistic Approaches to Neural Coding* (MIT Press, Cambridge, 2007).
- [63] A. Soltani and X.-J. Wang, Synaptic computation underlying probabilistic inference, *Nat. Neurosci.* **13**, 112 (2010).
- [64] L. Buesing, J. Bill, B. Nessler, and W. Maass, Neural dynamics as sampling: A model for stochastic computation in recurrent networks of spiking neurons, *PLoS Comput. Biol.* **7**, e1002211 (2011).
- [65] D. Kappel, S. Habenschuss, R. Legenstein, and W. Maass, Network plasticity as Bayesian inference, *PLoS Comput. Biol.* **11**, e1004485 (2015).
- [66] L. Aitchison and P. E. Latham, Synaptic sampling: A connection between psp variability and uncertainty explains neurophysiological observations, [arXiv:1505.04544](https://arxiv.org/abs/1505.04544).
- [67] G. Orbán, P. Berkes, J. Fiser, and M. Lengyel, Neural variability and sampling-based probabilistic representations in the visual cortex, *Neuron* **92**, 530 (2016).
- [68] E. O. Neftci, B. U. Pedroni, S. Joshi, M. Al-Shedivat, and G. Cauwenberghs, Stochastic synapses enable efficient brain-inspired learning machines, *Front. Neurosci.* **10**, 241 (2016).
- [69] N. Hiratani and T. Fukai, Redundancy in synaptic connections enables neurons to learn optimally, *Proc. Natl. Acad. Sci. U.S.A.* **115**, E6871 (2018).
- [70] C. Baldassi, F. Gerace, H. J. Kappen, C. Lucibello, L. Saglietti, E. Tartaglione, and R. Zecchina, Role of Synaptic Stochasticity in Training Low-Precision Neural Networks, *Phys. Rev. Lett.* **120**, 268103 (2018).
- [71] G.-q. Bi and M.-m. Poo, Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type, *J. Neurosci.* **18**, 10464 (1998).
- [72] C. Clopath, L. Büsing, E. Vasilaki, and W. Gerstner, Connectivity reflects coding: A model of voltage-based STDP with homeostasis, *Nat. Neurosci.* **13**, 344 (2010).
- [73] D. Feldmeyer and B. Sakmann, Synaptic efficacy and reliability of excitatory connections between the principal neurones of the input (layer 4) and output layer (layer 5) of the neocortex, *J. Physiol.* **525**, 31 (2000).
- [74] S. Lefort, C. Tómm, J.-C. F. Sarria, and C. C. Petersen, The excitatory neuronal network of the C2 barrel column in mouse primary somatosensory cortex, *Neuron* **61**, 301 (2009).

- [75] J. P. Jones and L. A. Palmer, An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex, *J. Neurophysiol.* **58**, 1233 (1987).
- [76] B. A. Olshausen and D. J. Field, Emergence of simple-cell receptive field properties by learning a sparse code for natural images, *Nature (London)* **381**, 607 (1996).
- [77] S. Song, P. J. Sjöström, M. Reigl, S. Nelson, and D. B. Chklovskii, Highly nonrandom features of synaptic connectivity in local cortical circuits, *PLoS Biol.* **3**, e68 (2005).
- [78] D. Feldmeyer, V. Egger, J. Lübke, and B. Sakmann, Reliable synaptic connections between pairs of excitatory layer 4 neurons within a single ‘barrel’ of developing rat somatosensory cortex, *J. Physiol.* **521**, 169 (1999).
- [79] S. Dorkenwald, N. L. Turner, T. Macrina, K. Lee, R. Lu, J. Wu, A. L. Bodor, A. A. Bleckert, D. Brittain, N. Kemnitz *et al.*, Binary and analog variation of synapses between cortical pyramidal neurons, bioRxiv, <https://doi.org/10.1101/2019.12.29.890319v2>.
- [80] A. Braunstein and R. Zecchina, Learning by Message Passing in Networks of Discrete Synapses, *Phys. Rev. Lett.* **96**, 030201 (2006).
- [81] C. Baldassi, A. Ingrosso, C. Lucibello, L. Saglietti, and R. Zecchina, Subdominant Dense Clusters Allow for Simple Learning and High Computational Performance in Neural Networks with Discrete Synapses, *Phys. Rev. Lett.* **115**, 128101 (2015).
- [82] S. Geman and D. Geman, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, *IEEE Trans. Pattern Anal. Mach. Intell., PAMI-6*, 721 (1984).
- [83] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, [arXiv:1412.6980](https://arxiv.org/abs/1412.6980).
- [84] H. F. Song, G. R. Yang, and X.-J. Wang, Training excitatory-inhibitory recurrent neural networks for cognitive tasks: A simple and flexible framework, *PLoS Comput. Biol.* **12**, e1004792 (2016).
- [85] A. Ingrosso and L. Abbott, Training dynamically balanced excitatory-inhibitory networks, *PLoS One* **14**, e0220547 (2019).
- [86] L. Cossell, M. F. Iacaruso, D. R. Muir, R. Houlton, E. N. Sader, H. Ko, S. B. Hofer, and T. D. Mrsic-Flogel, Functional organization of excitatory synaptic strength in primary visual cortex, *Nature (London)* **518**, 399 (2015).
- [87] G. Stuart, N. Spruston, B. Sakmann, and M. Häusser, Action potential initiation and backpropagation in neurons of the mammalian CNS, *Trends Neurosci.* **20**, 125 (1997).
- [88] W. G. Regehr, M. R. Carey, and A. R. Best, Activity-dependent regulation of synapses by retrograde messengers, *Neuron* **63**, 154 (2009).
- [89] R. D. Fields and B. Stevens-Graham, New insights into neuron-glia communication, *Science* **298**, 556 (2002).
- [90] G. E. Hinton, Training products of experts by minimizing contrastive divergence, *Neural Comput.* **14**, 1771 (2002).
- [91] A. Graves, Practical variational inference for neural networks, *Adv. Neural Inf. Process. Syst.* **24**, 2348 (2011).
- [92] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, Weight uncertainty in neural networks, *Proc. Mach. Learn. Res.* **37**, 1613 (2015).
- [93] D. P. Kingma, T. Salimans, and M. Welling, Variational dropout and the local reparameterization trick, *Adv. Neural Inf. Process. Syst.* **28**, 2575 (2015).
- [94] D. R. Cox, The regression analysis of binary sequences, *J. R. Stat. Soc.: Series B Stat. Method.* **20**, 215 (1958).