NEUROSCIENCE

Self-backpropagation of synaptic modifications elevates the efficiency of spiking and artificial neural networks

Tielin Zhang[1,2], Xiang Cheng[1,2], Shuncheng Jia[1,2], Mu-ming Poo[2,3,4,5], Yi Zeng[1,2,4], Bo Xu[1,2,4]*

Copyright © 2021 The Authors, some rights reserved; exclusive licensee American Association for the Advancement of Science. No claim to original U.S. Government Works. Distributed under a Creative Commons Attribution NonCommercial License 4.0 (CC BY-NC).

Many synaptic plasticity rules found in natural circuits have not been incorporated into artificial neural networks (ANNs). We showed that incorporating a nonlocal feature of synaptic plasticity found in natural neural networks, whereby synaptic modification at output synapses of a neuron backpropagates to its input synapses made by upstream neurons, markedly reduced the computational cost without affecting the accuracy of spiking neural networks (SNNs) and ANNs in supervised learning for three benchmark tasks. For SNNs, synaptic modification at output neurons generated by spike timing–dependent plasticity was allowed to self-propagate to limited upstream synapses. For ANNs, modified synaptic weights via conventional backpropagation algorithm at output neurons self-backpropagated to limited upstream synapses. Such self-propagating plasticity may produce coordinated synaptic modifications across neuronal layers that reduce computational cost.

INTRODUCTION

Activity-dependent synaptic modification is essential for learning in natural and artificial neural networks (ANNs). In ANNs, the idea that synaptic weights could be tuned to achieve the correct output has led to the development of a computational algorithm for supervised learning known as backpropagation (BP) (*1*). In BP, errors in the output of ANNs with respect to the differences between output values and expected values are used to adjust synaptic weights of upstream synapses layer by layer until the output meets the expectation. BP has been widely used in various types of ANNs, including convolutional and recurrent neural networks (*2*). There is now increasing interest in brain-inspired machine learning algorithms (*3*–*6*) that incorporate distinct brain features and could achieve problem solving with high efficiency and low computational cost. Spiking neural networks (SNNs) are considered to be the third-generation ANNs (*7*), in which information is conveyed by discrete events, called spikes or action potentials. While SNNs could be more potent in processing temporal information (*8*), there remains difficulty in developing learning algorithms with an efficiency comparable to that of BP. Methods commonly used to train SNNs to perform machine learning tasks can be roughly classified into five categories: recursive least square (RLS)–based, gradient-based, reward-based, conversion-based, and plasticity-based. First-order reduced and controlled error (FORCE) is a special RLS-based method, which has been well applied on tasks of sequence storage and replay in a recurrent SNN (*9*). Gradient-based methods are borrowed from BP, including spatial and temporal types, to train SNNs in a supervised manner. Spatial types such as surrogate gradient (*10*) and approximate gradient [pseudo-BP (p-BP)] (*11*) are used to circumvent the nondifferential nature of spikes (*7*), and temporal types like BP through time (BPTT) (*12*) and SpikeProp (*13*) are designed for gradient learning recursively and temporally (with spike latency). The reward-based methods, including eligibility propagation (*14*) and reward propagation (*15*), are more efficient on sequential machine learning tasks. Conversion-based methods that directly convert learned ANNs to SNNs are simpler but still limited by BP on both biological interpretability and plausibility (*16*). Substantial efforts have also been made in applying biologically plausible plasticity rules found in natural neural circuits into SNNs, including Hebb's rule (*17*, *18*), short-term plasticity (STP) (*19*, *20*), long-term potentiation (LTP) (*21*, *22*), long-term depression (LTD) (*23*), and spike timing–dependent plasticity (STDP) (*24*, *25*). All these are local plasticity rules involving activity-dependent modification of synapses that induce the postsynaptic activity. Here, we focused more on plasticity-based methods and showed that introducing a novel form of nonlocal synaptic modification, termed "self-BP" (SBP) of synaptic potentiation and depression (*26*–*30*), helped to achieve a coordinated global propagation of synaptic modification, resulting in increased accuracy of SNNs and reduced computational cost of ANNs in performing supervised learning.

The phenomenon of SBP, first discovered in cultures of hippocampal neurons (*26*, *30*), involves cross-layer BP of LTP and LTD from output synapses to input synapses of a neuron (Fig. 1A). Although other forms of nonlocal spreads of LTP and LTD in the pre- and postsynaptic neurons have been observed in natural networks (*26*, *27*, *30*), we here confined our study on SBP, because its existence was confirmed in developing retinotectal circuits in vivo (*28*, *29*). The phenomenon of SBP represents a form of nonlocal activity–dependent synaptic plasticity that may endow developing neural circuits the capacity to modify the weights of input synapses on a neuron in accordance with the status of its output synapses (*27*). In training SNNs, potentiation and depression of synapses on output neurons could result from STDP due to relative spike timing between network-generated and targeted spike trains at the output neuron, thus representing positive and negative weight modifications based on supervised learning. Further SBP of potentiation and depression signals to upstream synapses provided an efficient approach for backpropagating the "correct" and "error" signals, respectively. In

[1]Research Center for Brain-inspired Intelligence, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China. [2]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China. [3]Institute of Neuroscience, State Key Laboratory of Neuroscience, Chinese Academy of Sciences, Shanghai 200031, China. [4]Center for Excellence in Brain Science and Intelligence Technology, Chinese Academy of Sciences, Shanghai 200031, China. [5]Shanghai Center for Brain Science and Brain-Inspired Intelligence Technology, Shanghai 201210, China. *Corresponding author. Email: xubo@ia.ac.cn.

Downloaded from https://www.science.org at Institute for Basic Science on January 21, 2022

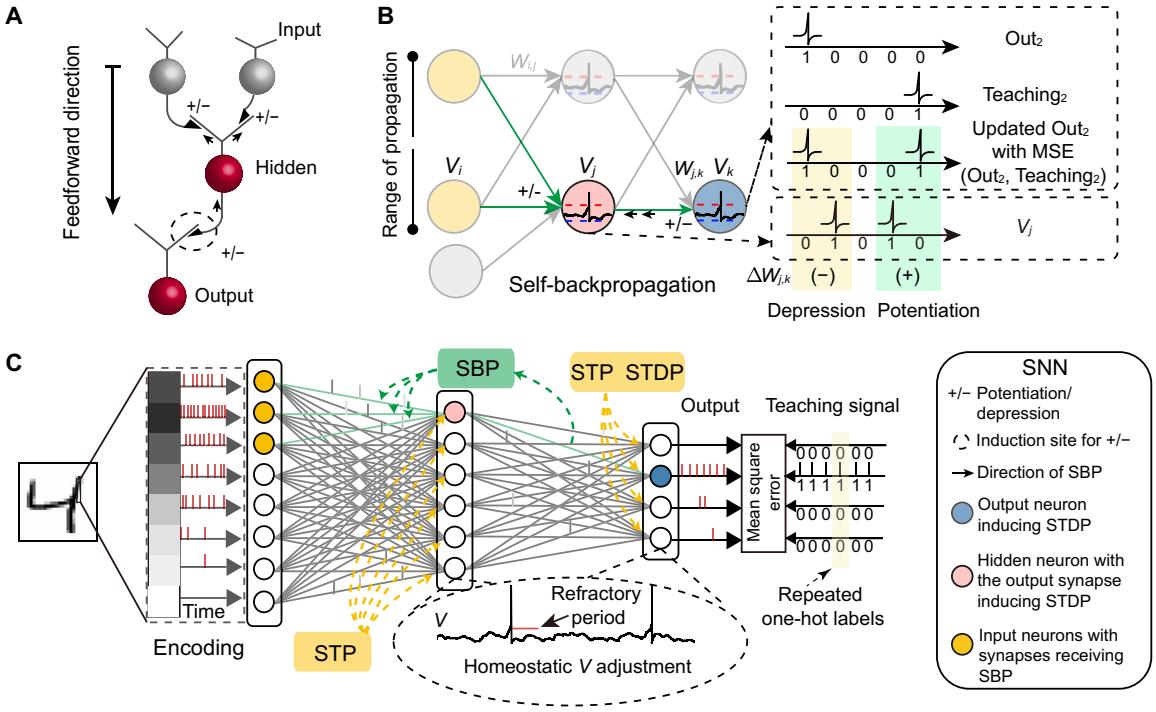


Fig. 1. Introducing biological SBP into SNNs. (**A**) Schematic diagram depicting the BP of potentiation ("+") or depression ("−") from the synapses at the output layer to those at the hidden layer in a three-layer network. The propagated synaptic modification had the same sign, consistent with the biological discovery of SBP (*27*). Similar configurations of fixed gradient mapping between neighborhood layers exist in artificial feedback alignment (*48*) and direct target propagation (*49*). (**B**) For a three-layer SNN, the induction of potentiation (+) or depression (−) occurred at the synapse $W_{j,k}$ on the output neuron by STDP, based on the timing of presynaptic spikes (in the hidden neuron) relative to the postsynaptic spikes in the output neuron, after updating by mean square error (MSE) of network-generated (Out_2) and teaching spike trains ($Teaching_2$). $W_{i,j}$ and $W_{j,k}$ represent synaptic weights of connections onto hidden and output neurons, respectively. The + and − signals created at synapses of hidden layer neuron (pink) onto an output neuron (blue) were allowed to spread to a percentage factor λ_p ($\lambda_p \in$ [10%,100%]) of input synapses with a fraction factor λ_f ($\lambda_f \in$ [0.1,1]) of the signals generated by the STDP (orange). V_j and V_k are membrane potentials at hidden and output layers, respectively. (**C**) The three-layer architecture of SNN, in which SBP and local plasticity (STP, STDP, and homeostatic *V* adjustment) were introduced at synapses at hidden and output layers, and the teaching spike train was given to the output LIF neurons. The diagram illustrates an output neuron inducing STDP (blue), a hidden neuron with the output synapse inducing STDP (pink), and input neurons with synapses receiving SBP (yellow).

training ANNs, traditional BP was used for synaptic weight adjustment at the output layer based on error signals. However, we found that replacing BP with SBP for weight adjustment of upstream synapses during a fraction of the learning period also produced beneficial effects.

RESULTS

Introducing SBP into three-layer SNNs and ANNs

For training SNNs, we used a three-layer SNN (Fig. 1C). In the first (input) layer, neurons received spike trains as inputs encoded by comparing raw signals from datasets with a train of generated random numbers (see Materials and Methods for more details). The second (hidden) layer consisted of both excitatory and inhibitory leaky integrate-and-fire (LIF) neurons that exhibited the refractory period, nonlinear integration, and nondifferentiable membrane potential. The third (output) layer consisted of excitatory LIF neurons that received spiking signals from hidden layer neurons, and the supervised teaching signals were presented only in training procedures. The learning process used both local form of synaptic modification, i.e., STP (*19*, *20*, *31*) and STDP (*32*, *33*), and nonlocal SBP via sequential steps. First, feedforward processing of spiking signals was performed without introducing synaptic plasticity. Second, we introduced STP and homeostatic adjustments of membrane potential (homeo-*V*) in hidden layer neurons to stabilize the spiking capability of the network (see Materials and Methods for details). Third, potentiation ("+") or depression ("−") of synaptic weights ($W_{j,k}$) was produced by the STDP rule at all synapses made by hidden neurons onto output neurons (Fig. 1B). Last, potentiation or depression of latter synapses was allowed to spread retrogradely by SBP to synapses made by input neurons on hidden neurons. For introducing some specificity in the amount of synaptic modification via SBP, we set a percentage factor ($\lambda_p \in$ [10%,100%]) and a fraction factor ($\lambda_f \in$ [0.1,1]), allowing SBP to cover only a percentage of upstream neurons (see Materials and Methods for details) and a fraction of synaptic modifications to undergo SBP, respectively.

The restricted Boltzmann machine (RBM) network (*34*) was used to examine the effect of introducing SBP into ANNs. The RBM contained three layers, artificial neurons with rectified linear unit (ReLU) activation functions, and fully connected feedforward connections (Fig. 2A). The learning procedure consisted of two phases: the unsupervised "sleep" phase and the supervised "wake" phase. During the sleep phase, only the energy function (see Materials and Methods for details) was used for calculating neuronal states toward the minimal energy (Fig. 2C). The wake phase included two types (I and II), interleaved by the sleep phase. In wake phase I, synaptic

Downloaded from https://www.science.org at Institute for Basic Science on January 21, 2022

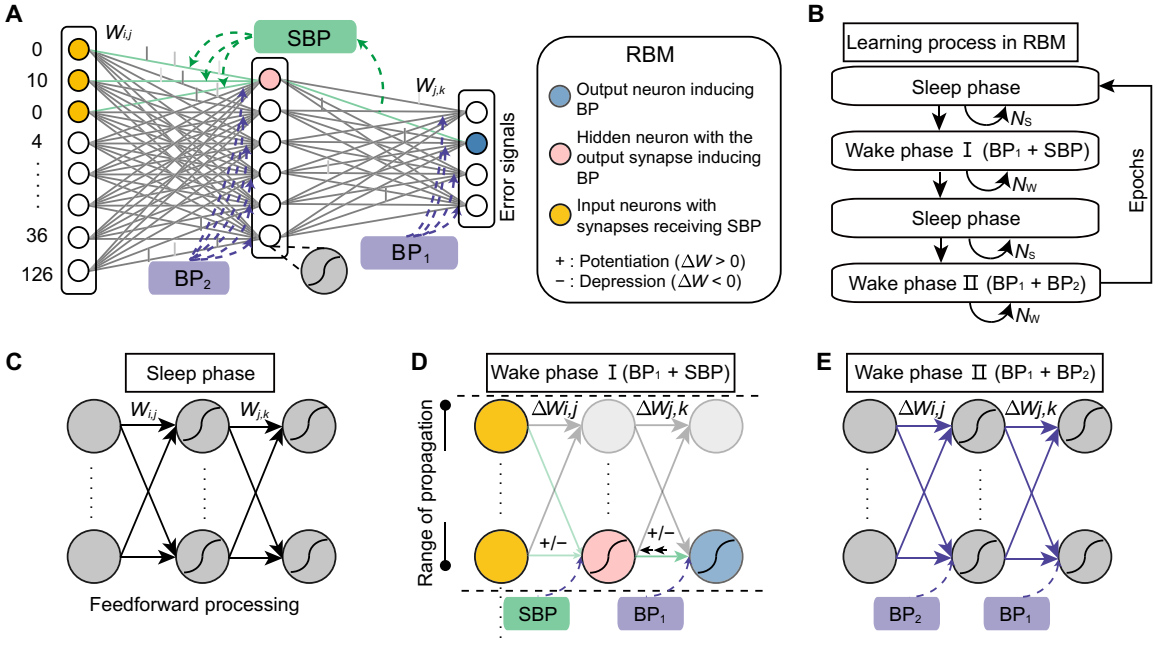


Fig. 2. Introducing biological SBP into ANNs. (**A**) Schematic diagram depicting the architecture of the shallow ANN, represented by a three-layer restricted Boltzmann machine (RBM), with full connections between neurons in neighborhood layers. Neurons in hidden and output layers were artificial rate neurons with ReLU activation functions. Two network state indicators were used: the unsupervised energy function (E^{RBM}; see Materials and Methods for details) describing the inner network state and the supervised cost function (C^{RBM}; see Materials and Methods for details) describing network output state. (**B**) Schematic diagram depicting the learning process of RBM using SBP, in which wake phase I using BP (BP$_1$) and SBP and wake phase II using only BP (BP$_1$ + BP$_2$) interleaved by the sleep phase. (**C**) Unsupervised sleep phase, in which both $W_{i,j}$ and $W_{j,k}$ were tuned toward minimal energy function E^{RBM}. (**D**) Wake phase I using both BP and SBP. The BP$_1$ produced potentiation ($\Delta W_{j,k} > 0$, +) or depression ($\Delta W_{j,k} < 0$, −) of $W_{j,k}$ between a hidden neuron (pink) and output neuron (blue), determined by differentiating the sum of C^{RBM} and E^{RBM}. The SBP induced + and − of $W_{i,j}$ based on $\Delta W_{i,j}$, with a percentage factor λ_p and a fraction factor λ_f as described in Fig. 1B. (**E**) Wake phase II using only BP containing both BP$_1$ and BP$_2$. $W_{i,j}$ and $W_{j,k}$ were updated on the basis of the minimization of both cost and energy functions with the chain rule of calculus.

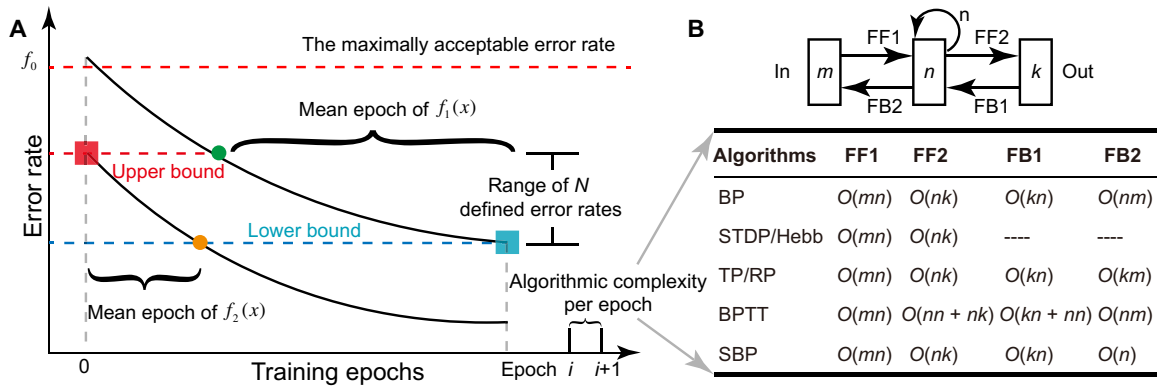
weights at the output layer ($W_{j,k}$) were updated according to standard BP algorithm with both energy and cost functions (BP$_1$; see Materials and Methods for details), resulting in potentiation (+) or depression (−), and this was followed by the SBP of + or − (Fig. 2D). In wake phase II, BP was performed in a conventional manner, with BP$_2$ (at the hidden layer) following BP$_1$ (Fig. 2E). The learning procedure is summarized in the schematic diagram in Fig. 2B. The amount of self-propagated modifications was set as a fraction of $\Delta W_{j,k}$, represented by the parameter θ_{sbp} in E^{RBM} and also the parameter λ_f (corresponding to that in SNNs), and the percentage of upstream neurons receiving SBP was described by λ_p as that in SNNs. Note that the replacement of BP$_2$ by SBP during wake phase I could notably reduce the computational cost normally required to perform the differential calculation of BP$_2$ during the learning process.

In this study, we examined the effects of introducing SBP on the accuracy and computational cost of SNNs and ANNs for three different learning tasks involving different extents of temporal information: (i) recognition of handwritten digits, using Modified National Institute of Standards and Technology (MNIST) dataset (fig. S1A) (*35*); (ii) phonetic transcription, using NETtalk dataset (fig. S1B) (*36*); and (iii) gesture recognition, using event-based dynamic vision sensor gesture (DvsGesture) dataset (fig. S1C) (*37*). The computational cost of networks during learning was defined by the product of the mean training epoch to achieve some defined accuracy levels (Fig. 3A) and algorithmic complexity per epoch (Fig. 3B). Our results demonstrated that introducing SBP into SNNs resulted in higher accuracies and lower computational costs in learning all three tasks. Furthermore, the combined use of SBP and BP in ANNs also resulted in similar benefits in all three tasks. These results underscored the usefulness of introducing a novel nonlocal plasticity rule found in natural neural networks into SNNs and ANNs.

SBP improved the efficiency of SNNs for three benchmark tasks

For learning hand digit recognition on MNIST dataset, we used an SNN comprising 784 input neurons, 500 hidden neurons (half excitatory and half inhibitory), and 10 output neurons, with other configuration parameters shown in table S1. We trained the SNN with a subset (60,000) of MNIST dataset and tested its accuracy using the remaining MNIST data (10,000). The values of λ_p (λ_p= 0.3) and λ_f (λ_f= 0.7) were chosen as the standard parameters after a range of values were tested for optimal performance of SNNs (Fig. 4B). We found that the training error rate of SNNs using SBP converged faster than that found without using SBP (fig. S2A). The test error rate reached 6.25 ± 0.52% (SD, n= 5 repeating experiments with different random seeds) after the 91st epoch when only local plasticity rules (STDP and STP) were used, and it was significantly reduced to 4.86 ± 0.12% (SD, n= 5) after the 100th epoch with the addition of SBP (P< 0.01, t test; Fig. 4A). Furthermore, we compared the accuracy of the hand digit recognition achieved by SBP with or without STP and found that STP helped to converge training (fig. S2D) and reduce test error rate of SNNs slightly (from 5.02 ± 0.14% to 4.86 ± 0.12%; SD, n= 5; P< 0.01, t test; fig. S2G). Moreover, the

Downloaded from https://www.science.org at Institute for Basic Science on January 21, 2022



Algorithms	FF1	FF2	FB1	FB2
BP	$O(mn)$	$O(nk)$	$O(kn)$	$O(nm)$
STDP/Hebb	$O(mn)$	$O(nk)$	----	----
TP/RP	$O(mn)$	$O(nk)$	$O(kn)$	$O(km)$
BPTT	$O(mn)$	$O(nn + nk)$	$O(kn + nn)$	$O(nm)$
SBP	$O(mn)$	$O(nk)$	$O(kn)$	$O(n)$

Fig. 3. The computational cost during learning. (**A**) Diagram depicting calculation of the mean epoch in N training epochs (N= 5) for curves of $f_1(x)$ and $f_2(x)$ to achieve some defined error rate levels between an upper bound and a lower bound. The upper bound and lower bound represent the lowest and highest values of the error rate curves at the beginning and the end of learning epochs, respectively, among the algorithms under comparison (see Materials and Methods for more details). The computational cost was calculated by averaging the cost at five error rate levels (including upper and lower bounds). (**B**) Algorithmic complexity $O(\cdot)$ in each epoch during learning. It includes feedforward propagation (FF) and feedback propagation (FB). m, n, and k are numbers of neurons in network's input, hidden, and output layers, respectively. The compared algorithms include BP, STDP (or Hebb), direct target propagation (TP) (*49*), reward propagation (RP) (*39*), BPTT (*12*), and SBP.

computational cost for SNNs with and without SBP was (1.92 ± 0.27) × 10^7 and (0.88 ± 0.09)×10^7 (SD, n= 5; P< 0.01, t test), respectively (Fig. 4C). Thus, the SBP elevated accuracy (by ~1.4%) and reduced computational cost (by ~54.2%) of SNNs. Last, we compared the performance and computational cost of our learning algorithms with those of other reported state-of-the-art SNN algorithms using plasticity-based (*20*, *32*, *38*) and gradient-based (*14*, *15*, *39*, *40*) rules. For an SNN model consisting of the same number of parameters as described above (the number of neurons in input, hidden, and output layers), our algorithms yielded a higher accuracy and lowered computational cost than previously reported plasticity-based algorithms (fig. S3, A and B, and table S2). We also examined a gradient-based SNN using SBP [spiking multilayer perception (spiking-MLP)] and found similar benefits (fig. S3, A and B).

For learning phonetic transcription on NETtalk dataset, the SNN had to deal with multiple target phonemes. We thus used an architecture comprising 189 input neurons, 500 hidden neurons (half excitatory and half inhibitory), and 26 output neurons (that yielded 116 classes). Other network parameters are shown in table S1. The introduction of SBP improved the efficiency of SNNs after training with λ_p= 0.3 and λ_f= 0.7 (Fig. 4E), showing a test error rate of 14.30 ± 0.12% (SD, n= 5), which was lower than that obtained by using only STP and STDP in the absence of SBP (15.74 ± 0.20%; SD, n= 5; P< 0.01, t test) on the same test dataset (Fig. 4D and fig. S2B). Local STP had little help in achieving higher accuracy, because the test error rate (14.30 ± 0.12%; SD, n= 5) when both STP and SBP were present was similar to that found when STP was absent (14.30 ± 0.13%; SD, n= 5; fig. S2, E and H). Furthermore, SBP also reduced computational cost from (4.47 ± 0.43) × 10^6 (without SBP) to (0.91 ± 0.37)× 10^6 (with SBP; SD, n= 5; P< 0.001, t test; Fig. 4F). Thus, the SBP elevated accuracy (by ~1.4%) and reduced computational cost (by ~79.6%) of SNNs. Our results also yielded higher accuracy and lowered computational cost than those obtained by us using our SNN structure and other reported plasticity-based (*20*) and gradient-based (*14*, *15*, *39*, *40*) algorithms for SNNs (fig. S3, C and D, and table S2).

For learning gesture recognition on DvsGesture dataset, we used an SNN architecture comprising 1024 input neurons, 500 hidden neurons (half excitatory and half inhibitory), and 11 output neurons (corresponding to 11 gesture types). Detailed network parameters are listed in table S1. The use of SBP in addition to STP and STDP reduced the test error rate from 16.56 ± 0.16 (SD, n= 5) to 15.24 ± 0.04% (SD, n= 5; P< 0.001, t test) (Fig. 4G and fig. S2C) after training with λ_p= 0.3 and λ_f= 0.7 (Fig. 4H). The test error rate achieved by SNN using SBP and STP was 15.24 ± 0.04% (SD, n= 5), which was slightly lower than that using SBP without STP (15.41 ± 0.11%; SD, n= 5; P< 0.01, t test) (fig. S2, F and I). Furthermore, the computational cost for SNNs was (0.92 ± 0.03) ×10^7 and (1.53 ± 0.40) × 10^7, with and without SBP, respectively (SD, n= 5; P< 0.001, t test; Fig. 4I). Hence, the SBP elevated SNN's accuracy (by ~1.3%) and reduced SNN's computational cost (by ~39.9%). Our results also yielded higher accuracy and lowered computational cost than that obtained by us using reported plasticity-based (*20*) and gradient-based (*14*, *15*, *39*–*41*) SNN algorithms on our network architecture (fig. S3, E and F, and table S2). In summary, the introduction of SBP during training for three benchmark tasks elevated the efficiency of SNNs by increasing the accuracy (up to 1.4%) and greatly reducing the computational cost (up to 79.6%).

SBP improved the efficiency of ANNs for three benchmark tasks

For learning hand digit recognition, we used an RBM network comprising 784 input neurons, 500 hidden neurons, and 10 output neurons. Other related key parameters are shown in table S1. We trained the RBM with a subset (60,000) of the MNIST dataset and tested the accuracy of the RBM during the course of learning with a separate subset (10,000) from the data. We compared the error rates for two types of training: First, only BP (BP_1 + BP_2) was used in all stages of wake phase, and second, SBP was introduced (BP_1 + SBP) in wake phase I. We found that the error rate gradually reduced in nearly identical manner as the training proceeded (fig. S2J). After 100 epochs of training with wake phase I including BP_1+ SBP, the test error rate reached 2.11 ± 0.08% (SD, n=5), a value lower (~0.31% improvement) than that obtained with 100 epochs of training with BP only (BP_1 + BP_2 in both wake phase I and phase II) (2.42 ± 0.13%; SD, n= 5; P< 0.001, t test; Fig. 5A). We further examined the dependence of RBM performance difference (Δ% accuracy during the test; top panel in Fig. 5B) on the number of iterations during sleep and wake

Downloaded from https://www.science.org at Institute for Basic Science on January 21, 2022

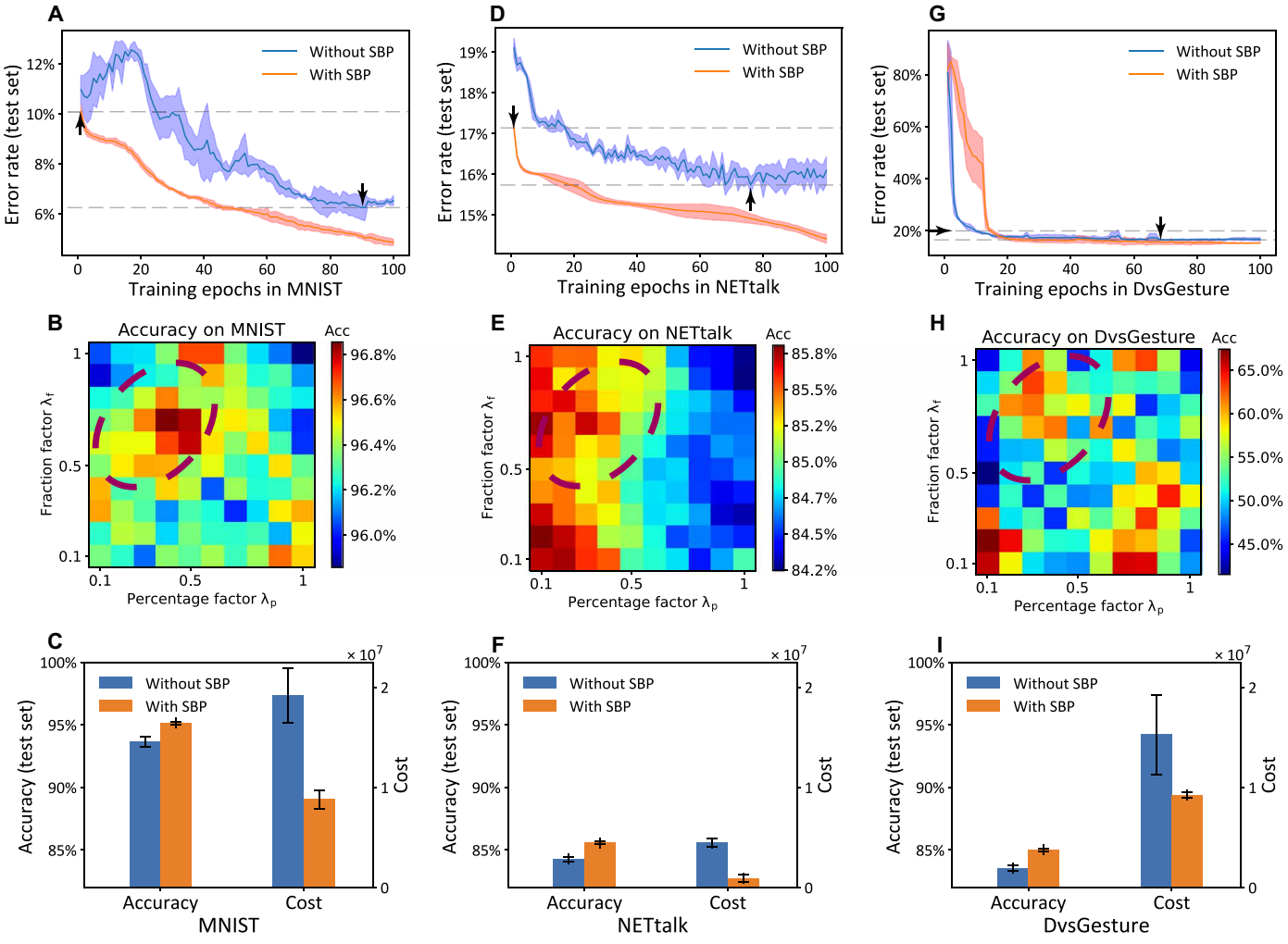


Fig. 4. SBP improved the performance of SNNs in three benchmark tasks. (**A** to **C**) Performance of SNNs for the hand digit recognition task using the MNIST dataset. (A) The SBP-improved SNN achieved higher test accuracy compared with that obtained without SBP (both with the same STDP, STP, and homeostatic *V* adjustment). Arrows point to the error rates of upper bound and lower bound, respectively, for comparing the computational cost. (B) Accuracies of SNNs using pairs of percentage factor (λ_p) and fraction factor (λ_f). The same pair of parameters (λ_p= 0.3 and λ_f= 0.7) was selected for a relatively better network performance (dashed circles) in all three tasks. The accuracy is coded in color by the scale shown on the right. (C) The SNN using SBP improved accuracy and reduced the computational cost [at error rates defined by arrowheads in (A)] compared to that found without SBP. (**D** to **F**) Performance of SNNs for the phonetic transcription task, presented in the same manner as that in (A) to (C). (**G** to **I**) Performance of SNNs for the gesture recognition task using the DvsGesture dataset presented in the same manner as that in (A) to (C). All figures are averaged over five repeating experiments with different random seeds.

phases (N_s and N_w) for each training epoch, which is directly related to the computational cost (bottom panel in Fig. 5B). By varying N_s and N_w from 1 to 26 with increments of 5, we obtained an accuracy difference map (ΔAcc) for the RBM performance with and without SBP for different combinations of N_s and N_w. We found that a broad distribution of N_s and N_w could yield higher accuracy by training with SBP in the wake phase, and the optimal pair of N_s and N_w to achieve the highest benefit was 11 and 6, respectively. We further estimated the computational cost difference of training RBM with and without SBP at these optimal values of N_s and N_w. The result showed that the computational cost for training with SBP [$(0.57 \pm 0.01) \times 10^7$; SD, n= 5] was substantially lower (a ~57.1% reduction) than that for training with BP only [$(1.33 \pm 0.03) \times 10^7$; SD, n=5; P< 0.001, t test; Fig. 5C].

For learning phonetic transcription, we used an RBM network comprising 189 input neurons, 500 hidden neurons, and 26 output neurons. Other related key parameters are shown in table S1. We trained the RBM with a subset (5033) of the NETtalk dataset and tested the accuracy of the RBM using a separate subset (500) from the NETtalk dataset during the course of learning. We found that the final converged error rates were similar with and without SBP for the training set (fig. S2K) and significantly lower for the test set for training with SBP (16.99 ± 0.28%), as compared to that without SBP (17.59 ± 0.64%; SD, n= 5; P= 0.002, t test) (Fig. 5D), representing ~0.6% improvement. Notably, the error rates converged much faster during the training with SBP for both the training and test datasets. This result showed that the introduction of SBP resulted in a lower error rate during both training and test, implicating higher

Downloaded from https://www.science.org at Institute for Basic Science on January 21, 2022

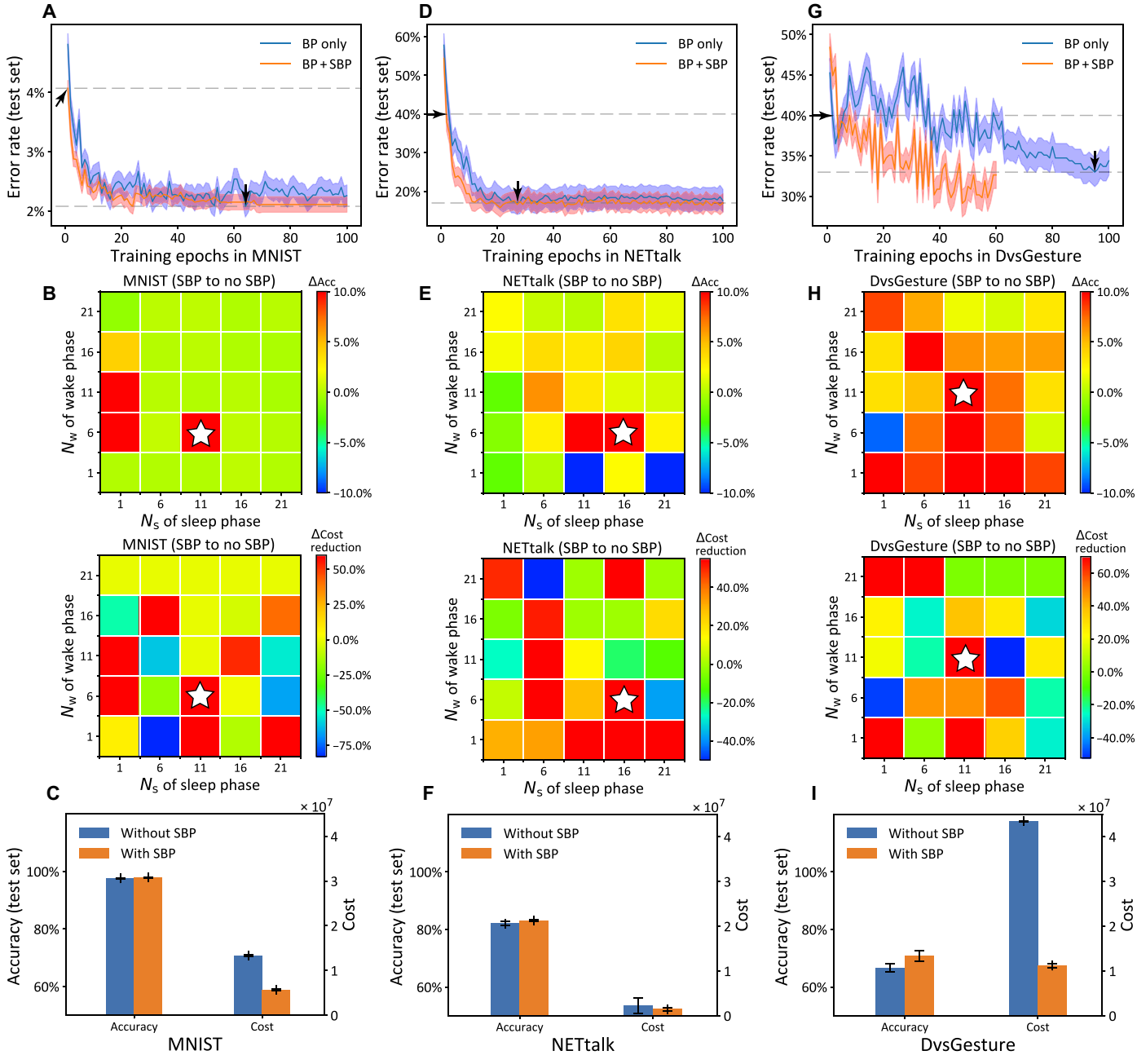


Fig. 5. SBP improved RBM performance in three benchmark tasks. (**A** to **C**) Performances of three-layer RBMs for hand digit recognition task. (A) Progressive reduction of the error rate (percentage of error trials) for the test dataset during RBM learning (see fig. S2, J to L, for error rates of three training datasets) with BP alone (blue) and with the addition of SBP (red) during the wake phase. Each curve consisted of 100 epochs of training that alternated between sleep and wake phases (see Fig. 2B). Arrows point to the error rates of upper bound and lower bound, respectively, for comparing the computational cost. (B) Dependence of RBM efficiency on training computational cost in sleep and wake phases. The gain in accuracy (upper) or computational cost (lower) of the RBM performance after training was indicated by the difference of accuracy (ΔAcc) or cost (ΔCost) obtained by training with and without SBP, and was plotted against the number of epochs during the sleep phase (N_s) and wake phase (N_w). The N_s and N_w values between 1 and 26 (with an increment of 5) were used. Optimal N_s and N_w, in terms of the gain in both accuracy and cost, were marked by white stars and chosen for presentation in (C). (C) Summary histograms on the average of maximal accuracy and computational costs marked by arrowheads in (A) at the optimal N_s and N_w chosen from the matrix in (B). (**D** to **I**) Performance of RBM for phonetic transcription task (D to F), and gesture recognition task (G to I), presented in the same manner as that in (A) to (C). (G) Early stopping was given at the 60th epoch for the test convergence.

network efficiency. Furthermore, the difference matrix of RBM with SBP learning also showed a broad distribution of N_s and N_w that yielded higher accuracy (top panel in Fig. 5E) and lower computational cost (bottom panel in Fig. 5E) compared to that without SBP, with optimal N_s and N_w values of 16 and 6, respectively. The use of SBP at these optimal N_s and N_w values reduced the computational cost by ~36.0% [from $(2.28 \pm 0.17) \times 10^6$ to $(1.46 \pm 0.04) \times 10^6$; SD, $n = 5$; $P < 0.001$, t test] (Fig. 5F).

Downloaded from https://www.science.org at Institute for Basic Science on January 21, 2022

For learning gesture recognition, the dataset was first reduced in size from 128 × 128 pixels to 32 × 32 pixels by a preprocessing procedure (see Materials and Methods for details) to fit the relatively low number of neurons in our three-layer RBM, which comprised 1024 input neurons, 500 hidden neurons, and 11 output neurons. We trained the RBM with a subset (1176) of DvsGesture data and tested the accuracy of RBM using a separate data subset (288) during the course of learning. We found that training with SBP yielded an error rate slightly lower than that of the training without SBP at the end of 100 training epochs (fig. S2L). For test set, the error rate became much lower for training with SBP than without SBP after 60 epochs of training. At the 50th epoch, the error rate for recognizing the test set reached the minimum of 29.12 ± 1.82% (SD, n= 5) for the network trained with SBP, a value significantly lower than that obtained without SBP (33.33 ± 1.48%; SD, n= 5; P< 0.01, t test) (Fig. 5G), representing ~4.2% improvement. The difference matrix at the 50th training epoch also showed a broad distribution of higher accuracy (top panel in Fig. 5H) and lower computational cost (bottom panel in Fig. 5H) for different combinations of N_s and N_w, with the optimal values of 16 and 11, respectively. The computational cost for training with SBP [$(1.12 \pm 0.04)\times10^7$] was also lower than that without SBP [$(4.35 \pm 0.01) \times 10^7$; SD, n= 5; P< 0.001, t test], representing ~74.3% reduction of the computational cost (Fig. 5I). In summary, we found that when SBP was introduced into the training of RBM, a type of ANNs, the performance on all three benchmark tests was improved to varying extents by reducing both the error rate (up to 4.2%) and computational cost (up to 74.3%).

DISCUSSION

In this work, we have introduced SBP of synaptic modification into SNNs and ANNs and examined its benefit for learning three benchmark tasks. For simplicity, we used three-layer feedforward networks comprising a variable number of neurons in the input, hidden, and output layers, depending on the task. The learning of the SNN consisted of two independent phases: first, the unsupervised learning phase of homeostatic adjustment of the membrane potential that maintained the firing capacity of the SNN and STP, which was found to be helpful in elevating the network efficiency (fig. S2, D to I), and second, the supervised learning phase that used STDP to initiate the correct and error signals in the form of potentiation and depression, respectively, and SBP for cross-layer synaptic weight adjustments. These SBP signals were generated by algebraic summation of synaptic changes based on the relative timing of all pairs of pre- and postsynaptic spikes using the standard pairwise STDP rule (*42*). Although not introduced in this study, additional constraints imposed by other STDP rules for natural spike trains in pre- and postsynaptic neurons (*43*, *44*) may further improve the network capability. Furthermore, other forms of nonlocal spread of synaptic modifications besides SBP, such as presynaptic lateral spread of LTP/LTD to synapses made by axon collaterals of the same presynaptic neurons (*26*, *30*) and to other converging inputs on the postsynaptic neuron (*28*, *29*), could be further explored for their potential benefits for SNNs.

For ANNs, we have examined the benefit of introducing SBP into the training of RBM, using its special feature of separating the training into supervised and nonsupervised phases. We have also examined multilayer spiking-MLP models and found similar benefits (fig. S4D). In supervised wake phase of RBM, the standard synaptic weight update was mostly based on the BP of error signals toward the minimization of the global loss function. Adding SBP would disturb the supervised tuning of the direction of the BP-induced gradient. A similar situation was found when Hebb's rule was added directly into BP (*45*). Perturbation of BP-induced synaptic weight updated by SBP could help drive the network modification toward an alternative direction, where the RBM may attain a higher accuracy with lower computational cost.

Only simple three-layer ANNs were used in the present study for all benchmark tests. Our studies on the SNN with four to six layers, using SBP in all hidden layers, showed that the benefit of introducing SBP was greatly degraded to a level below that achieved by the three-layer SNN. Training of RBM with four to six layers, with the SBP replacing BP in all hidden layers during wake phase I training, yielded no improvement in accuracy beyond that achieved by the three-layer RBM, despite higher computational costs (fig. S4, A and B). The degradation of accuracy in SNNs with more than three layers may be attributed to excessive spread of potentiation or depression signals when SBP was allowed to occur beyond the neuron that generates the original synaptic modification. In addition, the failed learning of SNNs using SBP for higher layers might also be caused by the nonconvergence problem of synaptic modifications. The previous work has shown that the recurrent SNN contains exploding gradients (*16*). The SNNs using SBP also show a nonconvergence learning problem, especially for deeper ones (fig. S4A), where the synaptic modifications between input and hidden layers are dominated by the STDP in hidden and output layers, and the influence of SBP from the induction layer to backpropagated layers is progressively weaker. This hypothesis was further verified in fig. S4C, where the distribution of synaptic modifications in three (or four) layers was properly norm-distributed, while that in five (or six) layers is left the same as that in initialization. Biologically experimental results of SBP in a network containing hippocampal neurons are consistent with this phenomenon, where the SBP also fails to propagate beyond one layer to more upstream neurons (*30*). The biological interpretation of this failure is that the potentiation/depression at input synapses due to SBP is based on cellular mechanisms distinctly different from those underlying LTP/LTD at the output synapses, thus incapable of generating further SBP in more upstream neurons. Notably, in some regions of the nervous system, such as retina, hippocampus, or neocortex, information processing could largely be characterized as a three-layer network operation within the region. Our finding that three-layer ANNs appear to be the optimal network to implement SBP suggests that ANNs may benefit from the use of three-layer networks as relatively independent basic modules, and more sophisticated ANNs could be built via parallel and serial connections among them.

In considering the efficiency of ANNs in performing standard benchmark tasks, previous studies using a variety of ANNs have largely focused on the accuracy in recognizing the test samples after network training. In this study, we have examined both the accuracy and the computational cost in learning tasks. Notably, the reduction of the computational cost represents the major benefit conferred by introducing SBP in both SNNs and ANNs. In estimating the computational cost, we used the product of the mean training epoch to achieve some defined accuracy levels (Fig. 3A) and algorithmic complexity per epoch (Fig. 3B) as an indicator. Other aspects of the cost, including the number of arithmetic operations and the number of bits required to specified synaptic weights and neuronal states

Downloaded from https://www.science.org at Institute for Basic Science on January 21, 2022

within each iteration, were not included here but could be further considered in the future work. In addition, we compared the computational cost during training in reaching the same given accuracy levels, rather than those for attaining the final converged accuracy by each operation. In most operations, the small increment in the final accuracy often requires disproportional large amount of computation. For efficient performance of the network, relatively high rather than the highest accuracy could be sufficient. The notion of balanced computational cost and accuracy is in line with the efficient information processing of the brain, where the rapidity in computation (with low energy cost) is as relevant as the accuracy.

Last, we note that the original experiment demonstrating SBP in cultured networks of hippocampal neurons (*26*) was inspired by the power of BP algorithm, although it seems to be biologically implausible (*46*). The SBP-associated information flow occurs in the neuronal cytoplasm, via retrograde fast axonal transport of molecular signals (*28*, *29*). The finding of SBP in natural networks has shown that an effective machine learning algorithm for ANNs can spur neuroscience discovery, and the present study further demonstrates that introducing algorithm-inspired biological discovery back to ANNs further elevates their efficiency. Such two-way interactions between neuroscience and artificial intelligence have much in store for the future.

MATERIALS AND METHODS

Definition of computational cost during training

The computational cost (Cost_i) of the algorithm i during training is defined by the product of the mean epoch number to achieve a defined error level (Fig. 3A) and a value $O(n)_i$ representing the algorithmic complexity per epoch (Fig. 3B). For the comparison of two algorithms (i= 1,2), the computational cost is calculated as follows

$$\mathrm{Cost}_i = \frac{1}{N}\sum_{l=1}^{N}\mathrm{Argmin}(f_i(x)=\mathrm{Err}_l)\times O(n)_i \tag{1}$$

where Argmin(·) is the argument of the minimum, $f_i(x)$ is the error rate curve with input epoch x, $O(n)_i$ is the algorithmic complexity with n depicting the number of parameters, and N is the number of predefined error levels (N= 5). Err_l is selected out from a range of error rates, with a lower bound of Max(Min (f_1), Min (f_2)), defined as the relatively higher minimal error rates of $f_1(x)$ and $f_2(x)$, and also with an upper bound of Min(Max (f_1), Max (f_2), f_0), defined as the relatively lower maximal error rates among $f_1(x)$, $f_2(x)$, and an additionally predefined error f_0 (the maximally acceptable error rate).

Preprocessing of datasets

For the MNIST dataset, the raw data were processed with normalization (i.e., subtract the minimum and divide by the range) and repeated T times to generate $I_{\mathrm{raw}}(t)$. For the NETtalk dataset, $I_{\mathrm{raw}}(t)$ was directly given by auditory signals. Then, input spike train $I_{\mathrm{spikes}}(t)$ was generated from $I_{\mathrm{raw}}(t)$ for these two datasets, shown as follows

$$I_{\mathrm{spikes}}(t)=\begin{cases}1 \ if\ \left(I_{\mathrm{raw}}(t)\geq\ I_{\mathrm{rd}}(t)\right)\\ 0 \ \ if\ \left(I_{\mathrm{raw}}(t)<\ I_{\mathrm{rd}}(t)\right)\end{cases} \tag{2}$$

where $I_{\mathrm{rd}}(t)$ is a uniformly sampled random number from 0 to 1. For the DvsGesture dataset, the raw signals were already event based; hence, an additional preprocessing for spike coding was not necessary.

The learning procedure of SNNs

For simplicity, we use i, j, and k to represent the indices of neurons in input, hidden, and output layers, respectively. The approximate pathway of information propagation and plasticity propagation in SNN is shown as follows

$$\begin{array}{c}\overbrace{\underbrace{\Delta W_{i,j}^{\mathrm{SBP}}(t)}_{\mathrm{SBP}}\leftarrow\underbrace{\Delta W_{j,k}^{\mathrm{STDP}}(t)}_{\mathrm{STDP,Dale'slaw}}\leftarrow}^{\text{Plasticity propagation}}\\ \overbrace{\underbrace{V_j^E(t),V_k^E(t)}_{\mathrm{Homeo}-V}+\underbrace{V_j^F(t),V_k^F(t)}_{\mathrm{LIFpropagation}}\leftarrow\underbrace{I_{\mathrm{syn}}}_{\mathrm{STP}}\leftarrow\underbrace{I_{\mathrm{spikes}}(t)}_{\mathrm{Input}}}^{\text{Information propagation}}\end{array} \tag{3}$$

where the plasticity induction is calculated on $\Delta W_{j,k}$ between the hidden and output layers by STDP first and then propagated to $\Delta W_{i,j}$ between the input and hidden layers. Each procedure will be further described in the following subsections.

The STP in SNNs

For the local STP, we allow the amplitude of postsynaptic potential to increase (facilitation) or decrease (depression) when the spiking frequency is low or high, respectively. This is described by the equations below, following the formulation of previous studies (*19*, *31*)

$$\begin{cases}\dfrac{du}{dt} = -\dfrac{u}{\tau_{\mathrm{f}}}+U(1-u)\,I_{\mathrm{spikes}}(t)\\ \dfrac{dx}{dt} = \dfrac{1-x}{\tau_{\mathrm{d}}}-\mathrm{uxI}_{\mathrm{spikes}}(t)\\ \dfrac{I_{\mathrm{syn}}}{dt} = -\dfrac{I_{\mathrm{syn}}}{\tau_{\mathrm{s}}}+AW_{i,j}uxI_{\mathrm{spikes}}(t)\end{cases} \tag{4}$$

where u and x are normalized variables representing dynamical characteristics of synaptic facilitation and depression, respectively. τ_{f} and τ_{d} are recovery time constants for facilitation and depression, respectively. A is an adjustable constant for synaptic weight $W_{i,j}$. τ_{s} is recovery time for synaptic current I_{syn}, which is used together with Eq. 5 for introducing STP into SNNs.

The LIF propagation in SNNs

In the LIF neuron model, the spikes in presynaptic neurons trigger postsynaptic potentials, which are dynamically integrated and generate spikes in the postsynaptic neuron when the firing threshold is reached. A refractory period is used after each spike. The membrane potential $V(t)$ is calculated as follows

$$\tau_{\mathrm{m}}\frac{dV(t)}{dt} = -(V(t)-V_{\mathrm{L}})-\frac{g_{\mathrm{E|I}}}{g_{\mathrm{L}}}(V(t)-V_{\mathrm{E|I}})+\frac{I_{\mathrm{syn}}}{g_{\mathrm{L}}} \tag{5}$$

where $\tau_{\mathrm{m}} = C_{\mathrm{m}}/g_{\mathrm{L}}$, C_{m} is a constant representing membrane capacitance, V_{L} is the leaky potential, g_{L} is the leaky conductance, $g_{\mathrm{E\,|\,I}}$ represents excitatory conductance (g_{E}) or inhibitory conductance (g_{I}), I_{syn} is the postsynaptic current, and $V_{\mathrm{E\,|\,I}}$ represents reversal potentials for excitatory (V_{E}) or inhibitory (V_{I}) neurons. The membrane potential $V(t)$ will be reset on threshold crossing (V^{Tr}) and clamped to the resting potential V_{rest} during the refractory period τ_{ref}. The membrane potential $V(t)$ in the feedforward phase will be represented as $V^{\mathrm{F}}(t)$.

Downloaded from https://www.science.org at Institute for Basic Science on January 21, 2022

The homeostatic *V* adjustment in SNNs

To circumvent the problem of nondifferentiable membrane potential with spikes in SNNs, we use a previous balanced tuning approach (*20*, *33*) for the homeostatic adjustment of $V_j(t)$. The network state is presented by the energy function E_j^{SNN} as follows

$$E_j^{\mathrm{SNN}} = \alpha_a V_j(t)^2 + \alpha_b \sum_i^N \left(V_i(t)\, W_{i,j}\, V_j(t) \right) + \alpha_c V_j(t) \tag{6}$$

With $\alpha_a = 0.5$, $\alpha_b = -1$, $\alpha_c = -V^{\mathrm{Tr}}$, $\frac{\partial E_j^{\mathrm{SNN}}}{\partial V_j(t)}$ is obtained as follows

$$\frac{\partial E_j^{\mathrm{SNN}}}{\partial V_j(t)} = V_j(t) - \left(\sum_i^N (V_i(t)\, W_{i,j}) - V^{\mathrm{Tr}} \right) = V_j(t) - V_j(t+1) \tag{7}$$

which represents temporal differential of the network state with respect to differential of membrane potential state of postsynaptic neuron j between its current state $V_j(t)$ and its future state $V_j(t+1)$. The tuning direction of $\Delta V_j(t)$ will converge toward its stable state by $\Delta V_j^{\mathrm{E}}(t)$ with learning rate η^e, as follows

$$\Delta V_j^{\mathrm{E}}(t) = -\eta^e \left(V_j(t) - \left(\sum_i^N V_i(t)\, W_{i,j} - V^{\mathrm{Tr}} \right) \right) \tag{8}$$

For the total change of membrane potential, $\Delta V_j(t)$ is obtained as follows

$$\Delta V_j(t) = \frac{t_{\mathrm{e}}}{T_{\mathrm{e}}} \Delta V_j^{\mathrm{F}}(t) + \left(1 - \frac{t_{\mathrm{e}}}{T_{\mathrm{e}}} \right) \Delta V_j^{\mathrm{E}}(t) \tag{9}$$

where $\Delta V_j^{\mathrm{F}}(t)$ represents the neuron state update in the feedforward procedure described in Eq. 5, and $\Delta V_j^{\mathrm{F}}(t)$ and $\Delta V_j^{\mathrm{E}}(t)$ are summed with the weight factor $\frac{t_{\mathrm{e}}}{T_{\mathrm{e}}}$ and $\left(1 - \frac{t_{\mathrm{e}}}{T_{\mathrm{e}}}\right)$, respectively. t_{e} is the lapsed number of epochs, and T_{e} is a total number of that during training. As the training proceeds, the weight of $\Delta V_j^{\mathrm{F}}(t)$ increases gradually to 1, whereas that of $\Delta V_j^{\mathrm{E}}(t)$ decreases to 0.

The STDP in SNNs

The teaching signal is created by repeating T times of expected spiking states of output neurons. That means only target-class neurons contain spikes, while others are left silent. The activity difference between network-generated spike trains and teaching spike trains is described as D and obtained as follows

$$D = \sum_k^K \sum_t^T (V_k(t) - \delta(t - t_{\mathrm{s}}))^2 \tag{10}$$

It is a mean square error (MSE) distance during time T for all K neurons in the output layer, where t_{s} represents spiking time in the teaching signal. To minimize D, the update of neural states $V_k(t)$ in output neuron k is given by the following equation

$$\Delta V_k(t) = -\eta \left(\sum_t^T V_k(t) - \delta(t - t_{\mathrm{s}}) \right) \tag{11}$$

where η is the learning rate. The update of $V_j(t)$ in Eq. 9 and $V_k(t)$ in Eq. 11 will further be consolidated into synaptic modifications during next-step STDP. The weight adjustment at each output synapse is calculated by the standard biphasic STDP rule (*25*, *27*) as follows

$$\Delta W_{j,k}^{\mathrm{STDP}}(t_{j,s}, t_{k,s}) = \begin{cases} \Delta W_{j,k}^{\mathrm{STDP+}} = A_+ e^{\frac{t_{j,s} - t_{k,s}}{\tau_+}} & \text{if } (t_{j,s} - t_{k,s} \le 0) \\ \Delta W_{j,k}^{\mathrm{STDP-}} = -A_- e^{\frac{t_{j,s} - t_{k,s}}{\tau_-}} & \text{if } (t_{j,s} - t_{k,s} > 0) \end{cases} \tag{12}$$

where A_+ and A_- are the scaling factors, and $t_{j,s}$ and $t_{k,s}$ are the spiking time of each pair of pre- and postsynaptic neurons (j and k). τ_+ and τ_- are the delay time parameters of the potentiation and depression, respectively. The detailed parameters are shown in table S1.

The Dale's law in SNNs

Unlike conventional ANNs, in which the sign of synaptic output at different synapses from the same neuron can be both positive or negative. Here, we also follow the constraint of Dale's law (*47*), where the postsynaptic potentials of all synapses, either positive (excitatory) or negative (inhibitory), are identical in profile but opposite in sign, based on the initial assignment of the neuronal type in the hidden layer. The synaptic modifications during learning have no limitation given that $W_{i,j} \Delta W_{i,j} \ge 0$, but have a limitation of $\Delta W_{i,j} \in [-|W_{i,j}|, |W_{i,j}|]$ given that $W_{i,j} \Delta W_{i,j} < 0$, to make sure that the signs of excitatory and inhibitory synapses would not be changed (*20*).

The SBP in SNNs

When STDP is induced at some specific output synapses, the synaptic weight adjustment $\Delta W_{j,k}^{\mathrm{STDP+}}$ and $\Delta W_{j,k}^{\mathrm{STDP-}}$ will backpropagate with different proportions of LTP and LTD to produce weight adjustment of $\Delta W_{i,j}^{\mathrm{SBP+}}$ and $\Delta W_{i,j}^{\mathrm{SBP-}}$ at hidden layer synapses, as shown by an example below for three hidden neurons (j= 1,2,3) and two output neurons (k= 1, 2)

$$\Delta W_{j,k}^{\mathrm{STDP}} = \begin{bmatrix} 0.1 & -0.2 & 0 \\ -0.4 & 0.5 & -0.6 \end{bmatrix} = \underbrace{\begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.5 & 0 \end{bmatrix}}_{\Delta W_{j,k}^{\mathrm{STDP+}}(\mathrm{LTP})} + \underbrace{\begin{bmatrix} 0 & -0.2 & 0 \\ -0.4 & 0 & -0.6 \end{bmatrix}}_{\Delta W_{j,k}^{\mathrm{STDP-}}(\mathrm{LTD})} \tag{13}$$

where values in the matrix are obtained by Eq. 12. Positive, negative, and zero values in the matrix indicate LTP, LTD, and no STDP, respectively. Then, $\sum_k \Delta W_{j,k}^{\mathrm{STDP+}}$ is calculated by summating all $\Delta W_{j,k}^{\mathrm{STDP+}}$ connected to hidden neuron j. The update of $\Delta W_{i,j}^{\mathrm{SBP+}}$ and $\Delta W_{i,j}^{\mathrm{SBP-}}$ can be described as follows

$$\Delta W_{i,j}^{\mathrm{SBP+}} = \left(\lambda_{\mathrm{f}} E_{\mathrm{diag},i} \left(I_j + \lambda_{\mathrm{p}} \sigma_{\mathrm{n}} \left(\sum_k \Delta W_{j,k}^{\mathrm{STDP+}} \right) \right) \right) \Delta W_{i,j}^{\mathrm{STDP+}} \tag{14}$$

$$\Delta W_{i,j}^{\mathrm{SBP-}} = \left(\lambda_{\mathrm{f}} E_{\mathrm{diag},i} \left(I_j - \lambda_{\mathrm{p}} \sigma_{\mathrm{n}} \left(\sum_k \Delta W_{j,k}^{\mathrm{STDP-}} \right) \right) \right) \Delta W_{i,j}^{\mathrm{STDP-}} \tag{15}$$

where $\sigma_{\mathrm{n}}(\cdot)$ denotes the normalization function with $\sigma_{\mathrm{n}}(x) = \frac{x}{\sum_i x_i}$, I is an all-ones vector (with I_j= 1), λ_{p} is a proportional factor ($\lambda_{\mathrm{p}} \in [10\%, 100\%]$), λ_{f} is a fraction factor ($\lambda_{\mathrm{f}} \in [0.1, 1]$), and $y = E_{\mathrm{diag},i}(x)$ denotes the function of expanding a vector x to a diagonal matrix y with $y_{j,j} = x_j$.

The learning procedure of the RBM

For a three-layer RBM, the network states at input, hidden, and output layers are represented as u_i, u_j, and u_k, respectively. The information propagation and plasticity propagation in the RBM is shown as follows

Downloaded from https://www.science.org at Institute for Basic Science on January 21, 2022

$$\overbrace{\underbrace{\Delta W_{i,j}^{\mathrm{SBP}}}_{\mathrm{SBP}} \leftarrow \underbrace{\Delta W_{j,k}^{\mathrm{BP}}}_{\mathrm{BPinduction}} \leftarrow \underbrace{C^{\mathrm{RBM}}}_{\mathrm{Loss}} + \underbrace{E^{\mathrm{RBM}}}_{\mathrm{Energy}}}^{\text{Plasticity propagation}} \leftarrow \overbrace{\underbrace{u_j, u_k}_{\mathrm{Propagation}} \leftarrow \underbrace{u_i}_{\mathrm{Input}}}^{\text{Information propagation}} \tag{16}$$

where the synaptic modifications between input and hidden layers ($\Delta W_{i,j}$) are constrained by that between hidden and output layers ($\Delta W_{j,k}$).

The loss function in RBM

The loss function of RBM is defined as the standard MSE, shown as follows

$$C^{\mathrm{RBM}} = \frac{1}{2}\sum_{k=1}^{K}(u_k - o_k)^2 \tag{17}$$

where cost is the difference of output u_k and expected teaching output o_k. For the RBM using pure BP, the synaptic weight adjustment $\Delta W_{j,k}^{\mathrm{BP}}$ and $\Delta W_{i,j}^{\mathrm{BP}}$ can be calculated by the differential chain rule as follows

$$\begin{cases} \Delta W_{j,k}^{\mathrm{BP}} = -\eta_{\mathrm{bp}}\dfrac{\partial C^{\mathrm{RBM}}}{\partial W_{j,k}} \\ \Delta W_{i,j}^{\mathrm{BP}} = -\eta_{\mathrm{bp}}\dfrac{\partial C^{\mathrm{RBM}}}{\partial W_{i,j}} \end{cases} \tag{18}$$

where η_{bp} is the learning rate.

The energy function in RBM

The SBP is implemented by a linear relationship between $\Delta W_{j,k}$ and $\Delta W_{i,j}$. We apply a special energy function E^{RBM} to constrain this linear relationship during training via the following equation

$$\begin{aligned} E^{\mathrm{RBM}} = &\left(\left(\sum_{i,j} u_i(t)\, W_{i,j}\, u_j(t)\right) + \left(\sum_{i}(u_i(t))^2\right)\right) \\ &+\theta_{\mathrm{sbp}}\left(\left(\sum_{j,k} u_j(t)\, W_{j,k}\, u_k(t)\right) + \left(\sum_{j}(u_j(t))^2\right)\right) \end{aligned} \tag{19}$$

where θ_{sbp} is scalar variable for setting the influence of SBP. The total cost function for RBM using SBP is shown as follows

$$C_{\mathrm{loss}} = \beta C^{\mathrm{RBM}} + E^{\mathrm{RBM}} \tag{20}$$

where β is a decay factor.

The SBP in RBM

When SBP is introduced, we replace BP at $\Delta W_{i,j}$ at some iterations (during wake phase I) as follows

$$\Delta W_{i,j}^{\mathrm{SBP}} = \eta_{\mathrm{SBP}}\lambda_{\mathrm{f}} E_{\mathrm{diag},i}\left(I_j + \sigma_{\mathrm{s}}\left(\lambda_{\mathrm{p}}\sum_{k}\Delta W_{j,k}^{\mathrm{BP}}\right)\right) u_i u_j \tag{21}$$

where η_{SBP} is a learning rate, $\Delta W_{i,j}^{\mathrm{SBP}} \in [0, 2\eta_{\mathrm{SBP}} u_i u_j]$, I is an all-ones vector ($I_j = 1$), $E_{\mathrm{diag},i}(\cdot)$ is the same function with that used in SNNs, σ_{s} denotes sigmoid nonlinear activation function, λ_{p} is a proportional factor ($\lambda_{\mathrm{p}} \in [10\%, 100\%]$), and λ_{f} is a fraction factor ($\lambda_f \in [0.1, 1]$). The inclusion of the term $u_i u_j$ constrains the $\Delta W_{i,j}^{\mathrm{SBP}}$ to the synapses made by coactive input and hidden neurons in the spirit of Hebbian learning (*25*).

Learning in the spiking-MLP

Spiking-MLP is an MLP of ANN after replacing activation functions with LIF neurons, which is also a special type of SNNs. i, j, and k are neuron indices of input, hidden, and output layers, respectively. Spiking-MLP applies feedforward information propagation through a simpler version of LIF neurons, shown as follows

$$V_j(t) = g_j V_j(t-1)\left(1 - S_j(t-1)\right) + I_j(t) \tag{22}$$

where $S_j(t-1)$ is the firing state after membrane potential $V_j(t-1)$ reaching a firing threshold V_{th}, becoming 1 when $V_j(t) \geq V_{\mathrm{th}}$ or else 0 for $V_j(t) < V_{\mathrm{th}}$. $I_j(t)$ is input current with the simple format of $I_j(t) = \sum_i W_{i,j} S_i(t)$. g_j is a leaky item [$g_j \in (0,1)$]. During learning, the input signal from datasets is first encoded into spike trains (the same as that in previous SNNs) in a time window T. After feedforward propagation of spikes, the average firing rates of neurons during T in output layers are used for classification and regression. MSE C^{MLP} and energy function E^{MLP} are integrated together as the loss function via the following function

$$C^{\mathrm{MLP}} = \frac{1}{2K}\sum_{k=1}^{K}\left(\frac{1}{T}\sum_{t=1}^{T} S_k(t) - o_k\right)^2 \tag{23}$$

where o_k is the expected firing rate and K is the total number of output layers. E^{MLP} is the same as E^{RBM} in Eq. 19 but using spikes $S_i(t)$, $S_j(t)$, and $S_k(t)$ instead of firing rates $u_i(t)$, $u_j(t)$, and $u_k(t)$. Other calculations of C_{loss} and $\Delta W_{i,j}^{\mathrm{SBP}}$ are the same as those in Eqs. 20 and 21. The p-BP (*11*, *40*) is used for getting around the nondifferential feature of spiking-MLP during training by directly giving an approximate finite number (here is 1 for simplicity) to replace the infinite gradient (at a neighborhood of V_{th}) during the gradient BP.

Accuracy definition

In our experiments, the accuracy of MNIST or DvsGesture is defined as the number of correctly identifying samples dividing by the number of all samples. Different from it, the accuracy of NETtalk is defined as the cosine similarity distance of identified phonemes and real phonemes for the consideration of the multiphonemes in the same sample.

SUPPLEMENTARY MATERIALS

Supplementary material for this article is available at https://science.org/doi/10.1126/sciadv.abh0146

REFERENCES AND NOTES

1. D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors. *Nature* **323**, 533–536 (1986).
2. Y. LeCun, Y. Bengio, G. Hinton, Deep learning. *Nature* **521**, 436–444 (2015).
3. G. E. Hinton, P. Dayan, B. J. Frey, R. M. Neal, The "wake-sleep" algorithm for unsupervised neural networks. *Science* **268**, 1158–1161 (1995).
4. S. Z. Muller, A. N. Zadina, L. F. Abbott, N. B. Sawtell, Continual learning in a multi-layer network of an electric fish. *Cell* **179**, 1382–1392.e10 (2019).
5. H. Jaeger, Artificial intelligence: Deep neural reasoning. *Nature* **538**, 467–468 (2016).
6. G. Zeng, Y. Chen, B. Cui, S. Yu, Continual learning of context-dependent processing in neural networks. *Nat. Mach. Intell.* **1**, 364–372 (2019).
7. W. Maass, Networks of spiking neurons: The third generation of neural network models. *Neural Netw.* **10**, 1659–1671 (1997).

Downloaded from https://www.science.org at Institute for Basic Science on January 21, 2022

8. L. F. Abbott, B. DePasquale, R. M. Memmesheimer, Building functional networks of spiking model neurons. *Nat. Neurosci.* **19**, 350–355 (2016).
9. W. Nicola, C. Clopath, Supervised learning in spiking neural networks with FORCE training. *Nat. Commun.* **8**, 2208 (2017).
10. E. O. Neftci, H. Mostafa, F. Zenke, Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Sig. Process. Mag.* **36**, 51–63 (2019).
11. C. Lee, S. S. Sarwar, P. Panda, G. Srinivasan, K. Roy, Enabling spike-based backpropagation for training deep neural network architectures. *Front. Neurosci.* **14**, 119 (2020).
12. D. Huh, T. J. Sejnowski, *Advances in Neural Information Processing Systems* (Curran Associates Inc., 2018), vol. 31, pp. 1433–1443.
13. S. M. Bohte, J. N. Kok, H. La Poutre, Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing* **48**, 17–37 (2002).
14. G. Bellec, F. Scherr, A. Subramoney, E. Hajek, D. Salaj, R. Legenstein, W. Maass, A solution to the learning dilemma for recurrent networks of spiking neurons. *Nat. Commun.* **11**, 3625 (2020).
15. S. Jia, T. Zhang, X. Cheng, H. Liu, B. Xu, Neuronal-plasticity and reward-propagation improved recurrent spiking neural networks. *Front. Neurosci.* **15**, 654786 (2021).
16. R. Kim, Y. Li, T. J. Sejnowski, Simple framework for constructing functional spiking recurrent neural networks. *Proc. Natl. Acad. Sci. U.S.A.* **116**, 22811–22820 (2019).
17. S. Song, K. D. Miller, L. F. Abbott, Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nat. Neurosci.* **3**, 919–926 (2000).
18. F. Zenke, E. J. Agnes, W. Gerstner, Diverse synaptic plasticity mechanisms orchestrated to form and retrieve memories in spiking neural networks. *Nat. Commun.* **6**, 6922 (2015).
19. R. S. Zucker, Short-term synaptic plasticity. *Annu. Rev. Neurosci.* **12**, 13–31 (1989).
20. T. Zhang, Y. Zeng, D. Zhao, B. Xu, Brain-inspired balanced tuning for spiking neural networks, in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence* (IJCAI, 2018), pp. 1653–1659.
21. T. J. Teyler, P. DiScenna, Long-term potentiation. *Annu. Rev. Neurosci.* **10**, 131–161 (1987).
22. T. V. Bliss, G. L. Collingridge, A synaptic model of memory: Long-term potentiation in the hippocampus. *Nature* **361**, 31–39 (1993).
23. M. Ito, Long-term depression. *Annu. Rev. Neurosci.* **12**, 85–102 (1989).
24. Y. Dan, M. M. Poo, Spike timing-dependent plasticity of neural circuits. *Neuron* **44**, 23–30 (2004).
25. Y. Bengio, T. Mesnard, A. Fischer, S. Zhang, Y. Wu, STDP-compatible approximation of backpropagation in an energy-based model. *Neural Comput.* **29**, 555–577 (2017).
26. R. M. Fitzsimonds, H. J. Song, M. M. Poo, Propagation of activity-dependent synaptic depression in simple neural networks. *Nature* **388**, 439–448 (1997).
27. G. Bi, M. Poo, Synaptic modification by correlated activity: Hebb's postulate revisited. *Annu. Rev. Neurosci.* **24**, 139–166 (2001).
28. J. L. Du, H. P. Wei, Z. R. Wang, S. T. Wong, M. M. Poo, Long-range retrograde spread of LTP and LTD from optic tectum to retina. *Proc. Natl. Acad. Sci. U.S.A.* **106**, 18890–18896 (2009).
29. J. L. Du, M. M. Poo, Rapid BDNF-induced retrograde synaptic modification in a developing retinotectal system. *Nature* **429**, 878–883 (2004).
30. H.-Z. W. Tao, L. I. Zhang, G.-Q. Bi, M.-M. Poo, Selective presynaptic propagation of long-term potentiation in defined neural networks. *J. Neurosci.* **20**, 3233–3243 (2000).
31. M. Tsodyks, K. Pawelzik, H. Markram, Neural networks with dynamic synapses. *Neural Comput.* **10**, 821–835 (1998).
32. P. U. Diehl, M. Cook, Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Front. Comput. Neurosci.* **9**, 99 (2015).
33. T. Zhang, Y. Zeng, D. Zhao, M. Shi, A plasticity-centric approach to train the non-differential spiking neural networks, in *Proceedings of the AAAI Conference on Artificial Intelligence* (AAAI, 2018).
34. B. Scellier, Y. Bengio, Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Front. Comput. Neurosci.* **11**, 24 (2017).
35. Y. LeCun, The MNIST database of handwritten digits (1998); http://yann.lecun.com/exdb/mnist/.
36. T. J. Sejnowski, C. R. Rosenberg, Parallel networks that learn to pronounce English text. *Complex Syst.* **1**, 145–168 (1987).
37. A. Amir, B. Taba, D. Berg, T. Melano, J. Mckinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza, J. Kusnitz, M. Debole, S. Esser, T. Delbruck, M. Flickner, D. Modha, A low power, fully event-based gesture recognition system, in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition* (CVPR, 2017), pp. 7243–7252.
38. H. Hazan, D. Saunders, D. T. Sanghavi, H. Siegelmann, R. Kozma, Unsupervised learning with self-organizing spiking neural networks, in *Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2018), pp. 1–6.
39. T. Zhang, S. Jia, X. Cheng, B. Xu, Tuning convolutional spiking neural network with biologically plausible reward propagation. *IEEE Trans. Neural Netw. Learn. Syst.* (2021).
40. S. Wozniak, A. Pantazi, T. Bohnstingl, E. Eleftheriou, Deep learning incorporating biologically inspired neural dynamics and in-memory computing. *Nat. Mach. Intell.* **2**, 325–336 (2020).
41. Y. Wu, L. Deng, G. Li, J. Zhu, L. Shi, Spatio-temporal backpropagation for training high-performance spiking neural networks. *Front. Neurosci.* **12**, 331 (2018).
42. G. Q. Bi, M. M. Poo, Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *J. Neurosci.* **18**, 10464–10472 (1998).
43. R. C. Froemke, Y. Dan, Spike-timing-dependent synaptic modification induced by natural spike trains. *Nature* **416**, 433–438 (2002).
44. K. Kobayashi, M. M. Poo, Spike train timing-dependent associative modification of hippocampal CA3 recurrent synapses by mossy fibers. *Neuron* **41**, 445–454 (2004).
45. T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman, G. Hinton, Backpropagation and the brain. *Nat. Rev. Neurosci.* **21**, 335–346 (2020).
46. F. Crick, The recent excitement about neural networks. *Nature* **337**, 129–132 (1989).
47. T. Hokfelt, O. Johansson, M. Goldstein, Chemical anatomy of the brain. *Science* **225**, 1326–1334 (1984).
48. T. P. Lillicrap, D. Cownden, D. B. Tweed, C. J. Akerman, Random synaptic feedback weights support error backpropagation for deep learning. *Nat. Commun.* **7**, 13276 (2016).
49. A. Meulemans, F. S. Carzaniga, J. Suykens, J. Sacramento, B. F. Grewe, A theoretical framework for target propagation, in *Advances in Neural Information Processing Systems,* H. Larochelle, M. A. Ranzato, R. Hadsell, M.-F. Balcan, H.-T. Lin, Eds. (Curran Associates Inc., 2020), vol. 33, pp. 20024–20036.

Acknowledgments: We thank M. Shi for refining Fig. 1 and fig. S1. **Funding:** This work was supported by the National Key R&D Program of China (2020AAA0104305); the National Natural Science Foundation of China (61806195); the Strategic Priority Research Program of the Chinese Academy of Sciences (XDA27010404 and XDB32070000); the Key Research Program of Frontier Sciences, Chinese Academy of Sciences (QYZDY-SSW-SMCO01); the International Partnership Program of Chinese Academy of Sciences (153D31KYSB20170059); the Shanghai Municipal Science and Technology Major Project (2018SHZDZX05); and the Shanghai Key Basic Research Project (18JC1410100). **Author contributions:** B.X., T.Z., Y.Z., and M.-m.P. designed the study. T.Z., B.X., X.C., and S.J. performed the experiments and analyses. M.-m.P., B.X., T.Z., and Y.Z. wrote the paper. **Competing interests:** The authors declare that they have no competing interests. **Data and materials availability:** All data needed to evaluate the conclusions in the paper are present in the paper and/or the Supplementary Materials. All source codes can be downloaded from https://doi.org/10.5281/zenodo.5278798.

Submitted 9 February 2021
Accepted 27 August 2021
Published 20 October 2021
10.1126/sciadv.abh0146

Citation: T. Zhang, X. Cheng, S. Jia, M.-m. Poo, Y. Zeng, B. Xu, Self-backpropagation of synaptic modifications elevates the efficiency of spiking and artificial neural networks. *Sci. Adv.* **7**, eabh0146 (2021).

Downloaded from https://www.science.org at Institute for Basic Science on January 21, 2022

ScienceAdvances

Self-backpropagation of synaptic modifications elevates the efficiency of spiking and artificial neural networks

Tielin ZhangXiang ChengShuncheng JiaMu-ming PooYi ZengBo Xu

Sci. Adv., 7 (43), eabh0146. • DOI: 10.1126/sciadv.abh0146

View the article online
https://www.science.org/doi/10.1126/sciadv.abh0146
Permissions
https://www.science.org/help/reprints-and-permissions

Downloaded from https://www.science.org at Institute for Basic Science on January 21, 2022

Use of think article is subject to the Terms of service

Science Advances (ISSN) is published by the American Association for the Advancement of Science. 1200 New York Avenue NW, Washington, DC 20005. The title *Science Advances* is a registered trademark of AAAS.
Copyright © 2021 The Authors, some rights reserved; exclusive licensee American Association for the Advancement of Science. No claim to original U.S. Government Works. Distributed under a Creative Commons Attribution NonCommercial License 4.0 (CC BY-NC).