

Physical neural networks with self-learning capabilities

Weichao Yu^{1,2}, Hangwen Guo^{1,2,4*}, Jiang Xiao^{1,2,3,4,5*}, and Jian Shen^{1,2,3,4,5*}

¹State Key Laboratory of Surface Physics and Institute for Nanoelectronic Devices and Quantum Computing,
Fudan University, Shanghai 200433, China;

²Zhangjiang Fudan International Innovation Center, Fudan University, Shanghai 201210, China;

³Department of Physics, Fudan University, Shanghai 200433, China;

⁴Shanghai Research Center for Quantum Sciences, Shanghai 201315, China;

⁵Collaborative Innovation Center of Advanced Microstructures, Nanjing 210093, China

Received February 8, 2024; accepted April 30, 2024; published online July 22, 2024

Physical neural networks are artificial neural networks that mimic synapses and neurons using physical systems or materials. These networks harness the distinctive characteristics of physical systems to carry out computations effectively, potentially surpassing the constraints of conventional digital neural networks. A recent advancement known as “physical self-learning” aims to achieve learning through intrinsic physical processes rather than relying on external computations. This article offers a comprehensive review of the progress made in implementing physical self-learning across various physical systems. Prevailing learning strategies that contribute to the realization of physical self-learning are discussed. Despite challenges in understanding the fundamental mechanism of learning, this work highlights the progress towards constructing intelligent hardware from the ground up, incorporating embedded self-organizing and self-adaptive dynamics in physical systems.

self-learning, physical neural networks, neuromorphic computing, physical learning

PACS number(s): 05.65.+b, 07.05.Mh, 85.35.-p, 05.10.-a, 84.35+i

Citation: W. Yu, H. Guo, J. Xiao, and J. Shen, Physical neural networks with self-learning capabilities, *Sci. China-Phys. Mech. Astron.* **67**, 287501 (2024), <https://doi.org/10.1007/s11433-024-2403-x>

1 Introduction

Biological brains outperform state-of-the-art machine intelligence in many aspects, especially in terms of energy efficiency. For example, it is estimated that the energy consumed by a modern supercomputer to train a natural language processing model is about 1000 kWh, which is sufficient for human brains to operate for a full six years [1].

To emulate the energy-efficient computational capabilities

of biological brains, the field of neuromorphic computing, or brain-inspired computing, has emerged. This computational paradigm traces its origins back to Carver Mead at Caltech in the 1980s [2], where the development of bio-inspired microelectronic devices took place. The proposition was made that large-scale adaptive analog systems have the potential to learn about their environment while consuming significantly less power¹⁾, thereby harnessing the full capabilities of silicon fabrication on a wafer scale [3]. Noteworthy ex-

*Corresponding authors (Hangwen Guo, email: hangwenguo@fudan.edu.cn; Jiang Xiao, email: xiaojiang@fudan.edu.cn; Jian Shen, email: shenj5494@fudan.edu.cn)

1) J. Mattheij, Another Way of Looking at Lee Sedol vs AlphaGo.

amples of state-of-the-art large-scale neuromorphic systems include TrueNorth, Neurogrid, BrainScaleS, SpinNNAker, Loihi, Qualcomm, and Tianjic, among others etc. [2,4-6].

While the mechanism behind how the brain's vast number of neurons (approximately one hundred billion) and synapses (around one hundred trillion) achieve flexible cortical configurations remains as a fundamental question in neuroscience [7], computational scientists have developed machine learning models that attempt to mimic the operational principles of the human brain. One such model is the artificial neural network (ANN). An ANN is a computational framework consisting of numerous interconnected elements, known as neurons, which are capable of receiving, processing, and transmitting information. The connections between neurons, referred to as synapses, are adjustable and possess the ability to adapt and learn. A typical ANN is structured into multiple layers, as depicted in Figure 1(a), where the value of each neuron, denoted as y_j , is determined by the collective input from all neurons, x_i , in the preceding layer [8]:

$$y_j = \mathcal{F}\left(\sum_i w_{ij}x_i + \theta_j\right), \quad (1)$$

with w_{ij} the adjustable synaptic weights describing how much the input neuron x_i contribute and θ_j the bias locally applied on output neuron. The activation function $\mathcal{F}$ is non-linear and can take various forms, such as Sigmoid, Tanh, and ReLU. There are two stages for a working ANN: (i)

The learning process, where the network learns from a large amount of training data by adjusting weights as well as bias. The state of the network can be characterized by the loss function $E(w_{ij}, x_i)$, and during the learning phase, the neural network essentially modulates the landscape of the loss function, as schemed in Figure 1(b) and (c) [9,10]. When the training is finished, the network moves to the next stage—(ii) the inference process, where the network makes predictions on new data according to the weights determined in the first stage. With an increase in depth (number of layers), a neural network can express highly complex functions and learn more complex features from the training data [11]. As a result, there is a widespread understanding that “Multi-layer neural networks can do everything” [8], including emulating physical phenomena and discovering new physical concepts [12,13].

Although the ANN model has achieved remarkable success, it faces significant challenges, including high energy costs in both learning and inference processes, as well as a heavy reliance on external algorithms during learning. Enhancing computational models, such as ANN, is a critical objective in the field of artificial intelligence and machine learning. Neuromorphic computing, on the other hand, aims to address these challenges by mapping computational models (e.g., the one shown in Figure 1(a)) onto hardware through the design of physical neurons and synapses. State-of-the-art neuromorphic chips, primarily based on complementary

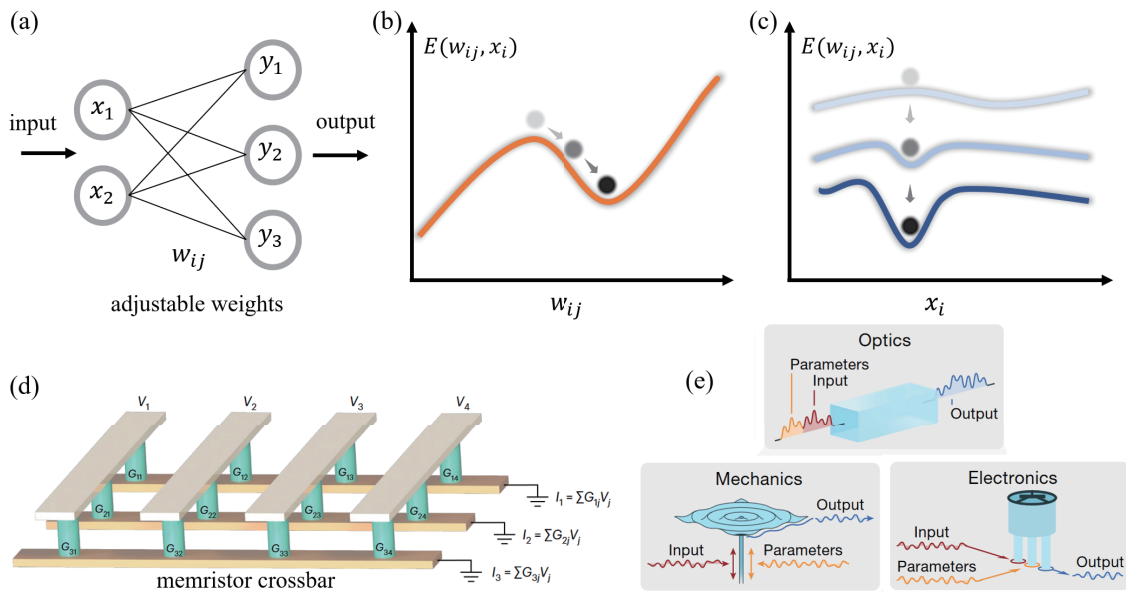


Figure 1 (Color online) (a) Structure of artificial neural networks (ANN). Only one layer is shown for simplicity. Data are received and transmitted in a feedforward way through the network. (b), (c) When the neural network is learning the vector x_i , it's actually looking for an appropriate set of weights w_{ij} that minimizes the loss function $E(w_{ij}, x_i)$. Bias θ_j is not shown for simplicity. (d) Memristor crossbar array is a type of PNN with isomorphic structure as ANN shown in (a). (e) Three types of PNNs which perform physical mapping from input to output, but not isomorphic to ANN shown in (a). The weights (and bias) of PNNs can be either internal physical degrees of freedom or external control parameters. The learning process of PNNs can be represented in the same way as demonstrated in (b) and (c). Reprinted with permission from refs. [9,10].

metal-oxide-semiconductor (CMOS) technology, have achieved tremendous success and surpassed conventional processors like the graphic processing unit (GPU) [14]. Other physical platforms, including photonics [15–17], spintronics [18–21], superconducting Josephson junctions [22], mechanical metamaterials [23, 24] and two-dimensional materials [25], among others [26, 27], are also making significant advancements.

This review focuses on a specific subgroup within neuromorphic computing known as physical neural networks (PNNs), where computation and training occur within the physical processes of the aforementioned physical systems [9] (Figure 1(e)). Both PNNs and CMOS neuromorphic chips can be considered as physical implementations of deep ANNs, which benefit from significant energy efficiency in the inference process due to physical acceleration. For instance, matrix multiplication can be efficiently performed in parallel either by leveraging Kirchhoff's law in electric circuits or by utilizing wave interference and scattering in photonic circuits [28, 29]. However, there is a fundamental distinction between PNNs and CMOS neuromorphic chips. While PNNs directly train the hardware's physical transformations to carry out desired computations, CMOS neuromorphic chips implement trained mathematical transformations by designing hardware with strict, operation-by-operation mathematical isomorphism [9]. Nowadays, most of the neuromorphic chips perform inference *in situ* but are trained *ex situ*, so that external computational resources are still required during the training process, which poses a bottleneck for further improving the speed and energy efficiency of neuromorphic computing. With their inherent capability of physical self-learning, PNNs offer a promising avenue for training without extensive reliance on external resources. Additionally, PNNs can harness the abundant internal degrees of freedom within physical systems, providing opportunities to enhance their capacity. However, several aspects of PNNs, including scalability, learning efficiency, noise tolerance, and the emergence of properties in complex systems beyond simple human-designed architectures, remain open questions that warrant further investigation. These areas present intriguing research directions to fully unlock the potential of PNNs and deepen our understanding of intelligent systems. This review primarily focuses on the learning (training) process in PNNs, particularly exploring the emerging self-learning capabilities embedded within these physical systems.

2 Physical neural networks

ANNs are virtual neural networks created in computers, with neurons and weights stored digitally. In contrast, physi-

cal neural networks (PNN) have neurons and weights represented in physical systems, which can be either digital or analog.

There are two main approaches to constructing PNN as depicted in Figure 2 [9, 30]. The top-down approach involves building and integrating individual physical neurons and synapses based on the blueprint from an ANN. The bottom-up approach entails creating a physical system with a stimulus-response relation that mimics the input-output relation of an artificial neural network, and has adjustable configurations for training.

The top-down approach for constructing PNN typically involves using discrete device elements as synapses, such as interconnecting non-volatile memories [31–35]. There are emerging memory technologies suitable for emulating a physical synapse [36], including metal-oxide-based memristors [37–40], also known as resistive random access memories (RRAM), spintronic-based memristors [41–49], ferromagnet/ferroelectric tunnel junctions [50, 51], ferroelectric memristors [52, 53], phase change materials (PCM) [54–58], and organic electrochemical devices [59]. Implementing PNNs in this way can significantly accelerate inference by taking advantage of the massively parallel operations allowed by physical laws, such as Kirchhoff's law. The memristor crossbar array (Figure 1(d)) is one of the most successful top-down type PNN, as demonstrated by various studies [60–63].

The bottom-up type PNN, in contrast, is usually based on a physical system that has some specific stimulus-response features. The individual neuron or synapse may or may not be identified. But overall, the system achieves the function of a neural network, which is to perform a nonlinear map-

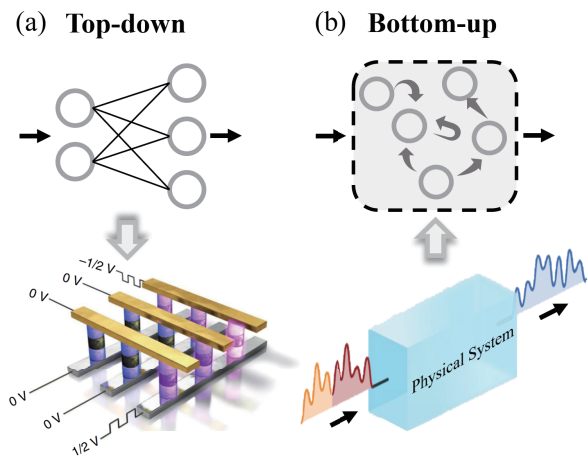


Figure 2 (Color online) Top-down vs. bottom-up. (a) Top-down approach maps the computational model of ANN isomorphically into neuromorphic hardware, such as memristor crossbar arrays. (b) Bottom-up approach treats a physical system as a PNN and performs training and inference based on its reconfigurable stimulus-response features. Reprinted with permission from refs. [9, 30].

ping from input to output, though the mapping of a PNN may not be explicitly generic. For instance, a magnetic thin film with complex magnetization patterns can be used to implement a Hopfield neural network [64]. Random nanocluster networks formed by disordered nanowires and nanoparticles can also exhibit collective intelligence [65]. Dynamical systems with high complexity, such as coupled oscillators, can operate on oscillatory signals, whose synchronization can be used to achieve non-trivial pattern classification tasks [66, 67]. Emerging devices like dynamic memristors [68–70], magnetic tunnel junctions (MTJs) [71] and material systems such as ferroelectrics [72, 73], synthetic active particles [74] and artificial spin ice [75] have been demonstrated to function as a physical reservoir [10] used in reservoir computing, which utilizes the intrinsic nonlinear dynamics of a given physical system as a computational resource for storage [76].

There are trade-offs associated with the top-down and bottom-up approaches for constructing physical neural networks. The top-down approach allows for a well-defined network structure and easy implementation of algorithms. However, it faces challenges in precisely replicating idealized network models with real physical devices that have variations and noise. The bottom-up approach leverages the natural dynamics of the physical platform. However, it is difficult to implement complex architectural designs, and the learning rules may not be fully determined. Overall, top-down PNNs have well-controlled implementations but struggle with non-idealities, while bottom-up approaches are more tolerant of hardware variations but lack programming flexibility. Nevertheless, both approaches can significantly enhance the architecture design of neuromorphic computing. A noteworthy example is the rotating neurons reservoir proposed by Liang et al. [77], which adopts a top-down approach by mapping the cyclic reservoir algorithm onto the physical rotation of a dynamic neuron array. This innovative design showcases the potential of integrating algorithmic concepts into physical systems to achieve efficient and effective computation. Conversely, researchers can consider a bottom-up approach by exploring physical systems, such as electronics and mechanical systems, that possess natural rotating components. By embedding and implementing algorithms like the cyclic reservoir within these systems, they can leverage the inherent properties of the physical components to achieve computational capabilities.

3 Learning and inference

Neural network models are effective for two primary reasons. Firstly, they offer enough flexibility to accurately represent real-life situations, with a sufficient number of synapses or

weights. Secondly, and more importantly, these models are trainable through efficient algorithms or learning rules. There are two working stages for both ANNs and PNNs, i.e., learning and inference.

3.1 Learning for PNN

Learning or training is the process through which a neural network can adapt itself to the environment, allowing for the construction of a faithful internal representation of the environment [78, 79]. The main objective of the learning process is to identify a set of weights that enable the network to make accurate predictions on new data. The training/learning process can be paced in two steps: Firstly, determining how much the weights need to be modified at each learning epoch, and secondly, actually updating the weights stored in the network, either in digital memory or in an analog physical entity. The learning is achieved by following certain learning rules, which govern how the weights should be updated based on the information at hand. One of the most widely known learning rules is the Hebbian principle based on the synaptic plasticity [80]: Neurons that fire together wire together, which was first proposed by Donald O. Hebb [81] in his 1949 book *The Organization of Behavior*.

Two types of learning are distinguished based on how the learning rules are implemented, as shown in Figure 3: digital learning and physical learning. The primary difference between these two types of learning lies in the digitalization of the learning process. Digital learning implements learning rules in a digital form using algorithms (such as the back-propagation algorithm) running on digital computers, while physical learning embeds the learning rules in an analog form through physical processes within physical systems.

Understanding how to adjust the weight and by how much is a crucial part of the learning process. Digital learning uses external training algorithms to calculate this information, while physical learning incorporates this part directly into the physical system. Nowadays, ANNs are primarily trained using digital learning through training algorithms such as the back-propagation algorithm with gradient descent. However, PNNs can be trained using either digital learning or physical learning. Digital learning is more suitable for top-down PNNs, which have well-defined individual neurons and synapses. In contrast, bottom-up PNNs, which lack well-defined weights, are not typically compatible with digital learning. Instead, their learning process relies on the physical processes inherent in the physical system.

3.1.1 Digital learning

Top-down PNNs, which are designed to accelerate inference,

typically lack the ability to adjust independently. Therefore, training of these PNNs must be carried out using digital learning. In reality, digital learning of a PNN is quite similar to that of an ANN, with the key distinction lying in the actual update of the trained weight. In ANNs, the weight and its update are digital and stored in computer memories, while in PNNs, the weight and its update are usually carried out through physical control of the analog physical entities that store the weights, or be stored. Recent studies on deep PNNs [9, 82] demonstrate the possibility of storing weights (or parameters) digitally, where the PNNs are still controllable by these external parameters.

Depending on when and how the physical entities are involved in the training process, ideas of online and offline training used in machine learning are borrowed for training PNNs. In the context of PNN training, the online and offline training refers to whether the weights are updated physically in an incremental way based on new data or in a single batch process where the weights are calculated based on all training data collected beforehand. Both online and offline training methods have their advantages and disadvantages, and the choice of training method depends on the specific requirements of the neural network and the nature of the data being used for training.

Offline learning, also known as *ex situ* or *in silico* learning, refers to train a PNN completely in computers based on an ANN that is considered as a digital twin of the PNN. Once the ANN has been trained, its weights are injected into its PNN twin. Throughout the whole training process [4], the PNN plays no role at all, or regarded as offline. One advantage of offline learning is that synaptic weights can be predetermined using deep learning algorithms on conventional computers. Offline learning has been widely applied in memristor crossbar arrays [60–63] for pattern classification [83, 84], images storage [60], and realization of sparse coding trained by locally competitive algorithm [85]. However, there are still challenges that need to be addressed, including material advancements [86] and device improvements [87]. These challenges involve achieving target conductance values with high accuracy, maintaining low fluctuations and drifts in resistance states, obtaining high absolute resistance values for inference, and ensuring high endurance for repeated programming/training [63].

Online learning, also known as the *in situ* learning, involves updating the network's parameters after each individual data point or small batches of data. This approach is necessary when not all training patterns are available simultaneously [88]. The primary advantage of online learning is its ability to adapt to changing environments, where the data distribution gradually shifts over time [89]. Unlike offline learning, where the weights are determined using all training data

in a batch, online learning is a process where the old weight vector $\mathbf{w}$ is iteratively updated to a new one $\mathbf{w}'$. This ability allows the rate of convergence or the degree of accuracy to automatically increase based on whether the weight vector is significantly different from the optimal or nearly optimal solution [88, 90, 91]. Although online learning was initially proposed for software implementation, it has provided valuable guidance for enabling physical systems to interact with external circuits, leading to the concept of “on-chip” learning [92, 93]. The online learning strategy has found wide application in RRAM and memristor crossbar arrays for tasks such as digital pattern classification [92, 94–96], self-organizing map implementation [97], feature extraction and dimensionality reduction [98], unsupervised K-means clustering [99], and construction of deep belief neural networks [100]. Notably, online learning can also be applied to train spiking neural networks [101, 102], where artificial neurons exhibit leaky integrate-and-fire characteristics, as traditional learning rules for ANNs, such as the conventional backpropagation algorithm, cannot be directly used [103].

3.1.2 Physical learning

Physical learning for PNNs involves integrating knowledge of physical laws and principles into the training process of neural networks. This includes incorporating fundamental laws of physics, such as conservation of energy, momentum, and other physical constraints, into the design and training of the neural network. Depending on the involvement of physics in the learning process, we can distinguish between physics-aware learning, where physical laws play a more passive role, and physical self-learning, where physical laws play an active role. When physical laws actively contribute to learning, the PNN may acquire self-learning capability, similar to various life forms that adjust themselves to better adapt to their environment.

Physics-aware learning is a hybrid *in situ* - *in silico* algorithm to solve the mismatch between the analog physical weight and its digital representation in the machine learning algorithms, first proposed by Wright et al. [9]. Physics-aware learning combines the advantages of backpropagation algorithm, the *de facto* training method for large-scale neural networks, with the ability to train controllable physical systems. By breaking the traditional software-hardware division, the physics-aware learning allows for the training of deep PNNs made from layers of controllable physical systems, even when these physical layers lack mathematical isomorphism to conventional artificial neural network layers. This approach enables the scalability of backpropagation while automatically mitigating imperfections and noise achievable with *in situ* algorithms. The concept is univer-

sal and has been applied experimentally on optics, mechanics, and electronics systems [9]. A similar concept has been applied in a designless nanoparticle network, whose electrical properties depend on input-voltage signals from multiple electrodes. The nanoparticle network can be optimized using a genetic algorithm or a deep-learning approach, aiming to perform reconfigurable computational tasks after training [104, 105]. The physics-aware learning applies to both top-down and bottom-up PNN.

Physical self-learning, as described by Feldmann et al. [106], Yu et al. [64], and López-Pastor et al. [107], involves a learning process where the internal degrees of freedom of a PNN evolve in response to a given environment or stimulus based on physical laws enabled by the physics system, rather than relying on external computation. As a result, the weights encoded within these internal degrees of freedom can autonomously adjust themselves through physical evolution without the need for external computations, as depicted in Figure 3(d). It is important to note that the concept of physical self-learning discussed here is synonymous with the notion of autonomous training proposed in ref. [108], and it is closely related to autonomous physical learning as introduced in ref. [109], albeit with a more specific definition.

In order to facilitate the self-learning capability of a PNN, specific requirements must be met by the underlying physical system. The learning process involves feeding back the inference error to the neural network, and a component within the system will utilize this error information to adjust the neural network. In traditional machine learning, this component is the digital computer. This process is to learn from the mistakes, which is to tune the weights to reduce mistakes. Equally efficient, neural networks can also learn from correctness, which is to strengthen the weights when the correct answer is fed to the network. For self-learning, the weight-tuning must be carried out autonomously by the PNN itself without external intervention. Therefore, such error/correctness sensing should be built-in in the PNN, and the response to the sensed error/correctness must also be embedded in some physical process within the PNN.

An example of achieving autonomous sensing and updating involves a non-volatile resistor with positive feedback. For instance, the resistance decreases (increases) when a large (small) current flows through it, and this resistance change is maintained when the current stops flowing. This mechanism strengthens the correct flow pattern injected into the network, enabling self-learning. We anticipate that such self-learning capabilities could arise in large complex systems exhibiting emergent collective dynamics [7]. By harnessing the embedded learning rules within physical systems, the construction of a PNN becomes achievable using a bottom-up methodology [110], as opposed to the conven-

tional top-down design approach used for electric circuits.

3.2 Inference for PNN

The inference process differs significantly between traditional digital learning systems and emerging physical learning paradigms. In digital ANNs, inference involves feeding input data forward through the network layers via numeric matrix multiplications calculated step-by-step using computer processors. This digital computation of inferences consumes large amounts of energy and time, particularly for deep networks with many layers [1]. In contrast, PNNs as well as hardware implementation of ANNs implement inference directly within analog physical systems using massively parallel natural dynamics. For example, memristor cross-bars perform matrix-vector multiplications intrinsically via Kirchhoff's law [4], enabling ultrafast, low-power inferences by analog signal propagation through interconnected devices. Other approaches use nanophotonic circuits [106], spintronic oscillators [66, 111] or elastic/fluidic systems [112, 113] to infer outputs from optical, magnetic or mechanical inputs. This physical acceleration circumvents the intrinsic inefficiencies of conventional processors, which serves as a primary motivation driving the field of neuromorphic computing [29].

3.3 Four scenarios for neuromorphic computing

We categorize four scenarios for neuromorphic computing in Figure 3, regarding to whether the learning and inference

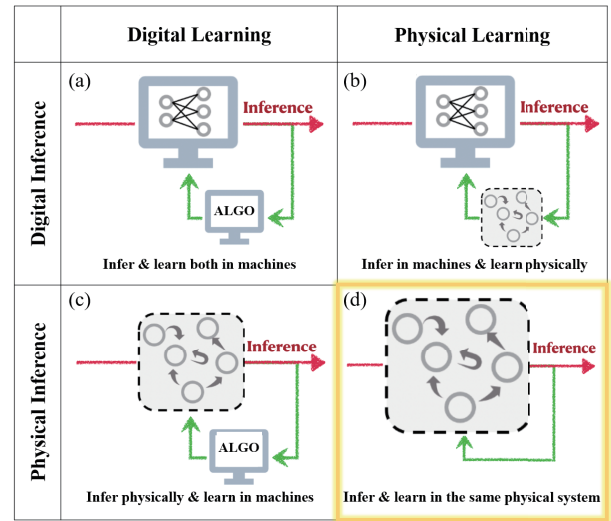


Figure 3 (Color online) Digital learning vs. physical learning. Neuromorphic computing can be divided into four categories regarding to whether the learning and inference are performed in ((a), (c)) digital machines by algorithms (ALGO) or ((b), (d)) physical systems by embedded physical process. Physical self-learning belongs to the paradigm where both inference and learning are carried out within the same physical system.

are performed in conventional computers by algorithms or in physical systems according to physical process.

Inference and learning are both performed in digital machines (Figure 3(a)). This aligns with the well-established field acknowledged as “machine learning” and falls outside the scope of this review.

Inference is performed in digital machines, while learning is performed in physical systems (Figure 3(b)). This is a relatively unexplored scenario, because the advantage of physical inference on physical acceleration was first realized by the community of neuromorphic computing while physical learning is still in its infant stage. However, a rapidly emerging field known as physics-informed neural networks (PINN) [114] has gained attention, which approximate solutions to partial differential equations (PDEs) for various domains, including fluidic [115], atomistic [116], and quantum dynamics [117].

Inference is performed physically and learning is performed in digital machines (Figure 3(c)). Most neuromorphic hardware, such as memristor crossbar arrays [61,63] and other types of PNNs [9,82], align with this scenario.

Both learning and inference are performed in the same physical system (Figure 3(d)). In this scenario, physical laws can either passively contribute through physics-aware learning or be actively involved according to physical self-learning. In the subsequent section, we will review recent advancements that have either implemented or hold the potential to achieve the concept of physical self-learning.

4 Physical systems capable of physical self-learning

4.1 Self-routing in random nanocluster networks

Disordered clusters of nanowires and nanoparticles can form networks that exhibit collective intelligence. Manning et al. [65] demonstrated the emergence of “winner-takes-all (WTA)” paths in macroscale nanowire networks. In WTA algorithms, neurons in a layer, typically the output layer, compete with each other for activation until only one neuron emerges as the winner and becomes active, representing the neuron associated with the strongest input signal. In this case, the WTA behavior is physically manifested through the clustering of mobile ions within the metal-insulator-metal junction, induced by an applied electric field. The largest cluster grows more rapidly than others, as depicted in Figure 4(a) and (b). The experiments indicate that the network dominated by junctions possesses the capability to autonomously select the lowest energy connectivity pathway within a complex random network. Similar to biological sys-

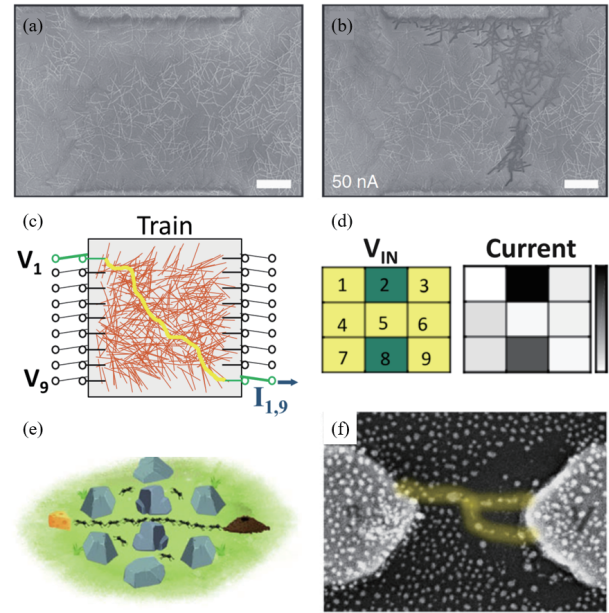


Figure 4 (Color online) A winner-takes-all conducting path is formed, corresponding to the lowest-energy connectivity path (a) before and (b) after the electric field is applied. Reprinted with permission from ref. [65]. (c) Interconnected nanowire pathway with low resistance is formed when voltage difference is applied on corresponding electrodes. (d) A 2-bit pattern is stored as an ensemble of higher connectivity pathways between the selected input/output channels. Reprinted with permission from ref. [119]. (e) Schematics of the ant colony algorithm. (f) Implementation of ant colony algorithm based on Ag nanoclusters, analogous to the ants. Reprinted with permission from ref. [121].

tems, memory is stored in the conductance state and encoded within specific connectivity pathways, which holds significant implications for the application of reservoir computing. A similar functionality can be achieved using molecular neuromorphic networks composed of single-walled carbon nanotubes complexed with polyoxometalate [118].

Diaz-Alvarez et al. [119,120] designed and demonstrated an associative device based on nanowire networks with resistive switching and retained connectivity. As illustrated in Figure 4(c), associative routing between input node 1 and output node 9 can be established by reducing the resistance in the nanowire networks through voltage activation. This leads to the formation of multiple pathways with low resistance, thereby enhancing nanowire-to-nanowire connectivity. Subsequent cycles of training and testing strengthen the local connectivity between the selected channels, enabling the simultaneous training of multiple patterns within the same network. A 2-bit pattern, exemplified in Figure 4(d), can be stored, and the network accurately retrieves the trained pattern during the testing procedure.

Swarm intelligence-inspired algorithms, including genetic algorithms and ant colony algorithms, have played a crucial role in optimization and distributed artificial intelligence.

Xu et al. [121] presented a remarkable demonstration of a multi-agent hardware system that utilizes distributed Ag nanoclusters as physical agents. In response to external electrical stimuli, the Ag nanoclusters adaptively evolved to form different connectivity patterns based on combined input schemes, as illustrated in Figure 4(e) and (f). The resulting connection path represents the optimal decision, considering both the cost (distance) and reward (voltage). The self-organization of the Ag physical network, driven by positive feedback from information interaction, leads to a significant reduction in computational complexity. The proposed hardware system exhibits efficient solutions for graph problems and optimization tasks, such as demonstrating gradient descent route planning with self-adaptive obstacle avoidance.

4.2 Memory formation in adaptive elastic networks and flow networks

The effect of “Directed Aging” has been shown by Pashine et al. [23] that the elastic properties of materials can depend strongly on the history of the imposed deformations, which obeys a natural greedy algorithm and follows the path of most rapid and accessible relaxation, making elastic networks an intriguing platform to discuss about memory formation. Stern et al. [122] theoretically studied an elastic network whose links grow and strengthen over time according to the local geometry of the network. It is found that such a network can acquire desired stable states in sequence only if the elastic elements are sufficiently nonlinear. The elastic networks is constructed with N nodes, and each two nodes are connected by a spring of stiffness k_{ij} and rest length l_{ij} . The position of node i is described by $\mathbf{x}_i$, and the energy of the elastic network is determined by

$$E(\{\mathbf{x}\}) = \frac{1}{2} \sum_{i=1}^N \sum_{j=i+1}^N k_{ij} (r_{ij} - l_{ij})^\xi, \quad (2)$$

with $r_{ij} \equiv \|\mathbf{x}_i - \mathbf{x}_j\|$ the distances between nodes and ξ parametrizes the nonlinearity. When a spring is pulled away from its preferred length, it exerts a force given by $F \sim (r - l)^{\xi-1}$. The behavior of the spring depends on the exponent ξ . For $\xi = 2$, the spring is linear, meaning that the force increases as the spring is strained further. However, for $\xi < 2$, the spring exhibits softer restoring forces at larger distances. In particular, for $\xi < 1$, the spring’s response weakens as it is strained.

In the elastic network, two learning strategies can be applied: simultaneous stabilization and continual learning. Simultaneous stabilization aims to stabilize multiple desired states by solving the equations $0 = \partial_{\mathbf{x}_a} E$ on each node, finding appropriate stiffness values k_{ij} and rest lengths l_{ij} . The

capacity of such networks to store multiple stable states is expected to scale linearly with the system size. However, adding a new stable state requires a complete rewiring of the network with new springs. In contrast, continual learning, as proposed by Stern et al. [122], introduces the concept of an effective spring constant that grows with time for the rods connecting two nodes i and j , which is given by

$$\frac{dk_{ij}^{\text{eff}}}{dt} = k_0 f(r_{ij}). \quad (3)$$

Continual learning involves an incremental growth model, where unstretched elastic rods gradually form between every pair of nodes when the material is left in a configuration for a certain period. The growth rate $f(r_{ij})$ depends on the separation between the nodes, as illustrated in Figure 5(a) and (b). It has been found that continual learning without forgetting requires nonlinear springs. These nonlinearities are essential for a broad class of continual models where the spring network is updated based solely on the current configuration, without knowledge of past or future desired configurations. The frameworks of simultaneous stabilization and continual learning have complementary strengths. The implementation of physical continual learning requires a nonlinear spring while it possible to apply the concept of simultaneous stabilization applies for spring-node systems with any value of nonlinearity. Continual learning may outperform simultane-

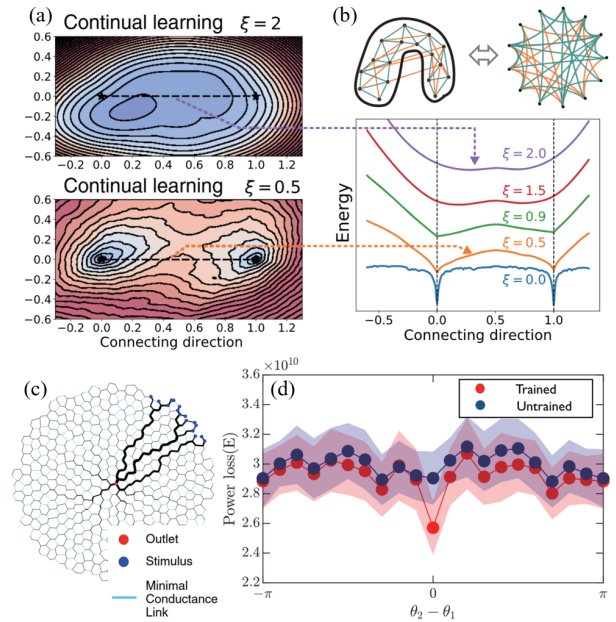


Figure 5 (Color online) (a) Continual learning in elastic networks can learn multiple states according to incremental growth of effective springs, as schematically shown in (b). Reprinted with permission from ref. [122]. (c) A flow network evolved with a stimulus equally distributed over the blue filled nodes. (d) Power loss before and after training, where training stimulus is applied at the angle $\theta_2 = \theta_1$. Reprinted with permission from ref. [112].

ous stabilization regarding to quality of encoding (such as barrier height and attractor size).

Similar to the elastic networks, the continuous adaptation of flow networks has been proven to play an important role in memory formation. Bhattacharyya et al. [112] theoretically investigated an adaptive flow network and show that adaptation dynamics allow a network to memorize the position of an applied load within its network morphology. As depicted in Figure 5(c), flow network is a physical model where links are strengthened and weakened over time adaptively. The network is defined as a graph of N nodes connected by links which have length of links l_{ij} and corresponding conductance $C_{ij}(t)$ varying over time t . The flow rate can be obtained as $Q_{ij}(t) = C_{ij}(t)(p_i(t) - p_j(t))$, where p_i represents potential at node i . The network can be characterized by an energy function (averaging over a period T):

$$E(t) = \sum_{\langle ij \rangle} \frac{\langle Q_{ij}(t)^2 \rangle_T}{C_{ij}(t)} - \lambda \sum_{\langle ij \rangle} (C_{ij}(t) l_{ij})^\gamma l_{ij}, \quad (4)$$

where the first term is exactly the power loss and the second term applies a global constraint in the presence of building cost, with λ the Lagrange multiplier and the parameter γ determining how link conductance contribute to the building cost. An adapting rule on $C_{ij}(t + \delta t)$ can be explicitly derived by solving the Lagrange multiplier. The adaption eventually minimizes the power loss of the whole network given the constraint of material conservation.

Notably, the parameter γ in the building cost has a significant impact on memory formation in flow networks. For instance, when an additional load is applied at a specific angle on the network's boundary over a duration, memory can be formed for $\gamma = 0.5$. This process can be considered as a training phase. After training, the power loss is minimized precisely for the angle at which the writing stimulus was applied, indicating that the network configuration has been optimized through memory of the stimulus (see Figure 5(d)). Memory formation, however, is only possible for $\gamma < 1$, while $\gamma = 1$ is typical for resistor networks or porous media.

In the field of neuroscience, it has been observed that memory formation is influenced by synaptic plasticity-dependent competition rules [123]. Biological networks can develop without centralized control, which allows them to yield reasonable solutions to combinatorial optimization problems [124, 125]. This section highlights that both the parameter ξ in elastic networks and the parameter γ in flow networks play vital roles in determining the nonlinearity of connections and whether memory formation is possible. Although the discussion focuses on physical networks, it is worth noting that there is a broader scientific community dedicated to understanding memory effects that appear in materials [126].

4.3 Magnetic textures with intrinsic Hebbian learning as associative memory

Energy-based models have garnered interests from both the physics and machine learning communities [127]. One specific type of artificial neural network is the Hopfield neural network, originally proposed by Hopfield [128]. This network architecture aims to mimic physical systems like the Ising spin-glass model [129], exhibiting emergent collective computational abilities. One of the prominent applications of the Hopfield neural network is associative memory. Associative memory is a dynamical system concerned with the storage and retrieval of memories [130]. In this framework, uncorrupted memories are associated with local minima in the network's energy landscape. Conversely, more corrupted memories correspond to higher energy values. The Hopfield network is capable of reconstructing an entire memory from partial information through a dynamic inference process, as long as the provided partial information is sufficient to identify a single memory.

Hopfield neural networks are a specific type of recurrent neural network architecture characterized by fully connected neurons. Due to their ease of implementation, these networks have been successfully mapped to various physical systems, enabling the realization of associative memory functions [53, 131-137]. In most of these works, the focus is primarily on the inference process, which involves retrieving information from the network after the training stage. During training, the weights of the network are computed using algorithms implemented on conventional computers, aligning with the paradigm depicted in Figure 2(c).

Yu et al. [64] proposed a hardware scheme that enables self-learning within the framework of Hopfield neural networks. The physical system they considered consists of conducting magnetic films with inhomogeneous textures, as depicted in Figure 6(a). When subjected to electric currents, the magnetic textures undergo spatial reconfigurations due to spin-transfer torque. This results in increment in conductance along the longitudinal direction (parallel to the current) and a decrease in conductance along the transverse direction (perpendicular to the current). The conductance evolution, illustrated in Figure 6(b), exhibits both excitatory and inhibitory features, indicating that the magnetic textures behave like adaptive synapses.

To demonstrate the feasibility of their approach, Yu et al. [64] mapped a four-node Hopfield neural network model onto a magnetic film with four electrodes. The synaptic weights were stored as a conductance matrix defined by distributed electric contacts. The relationship between the output current I_i and the input voltage V_j was described by Kirchhoff's law: $I_i = \sum_j G_{ij} V_j$. The energy function of the

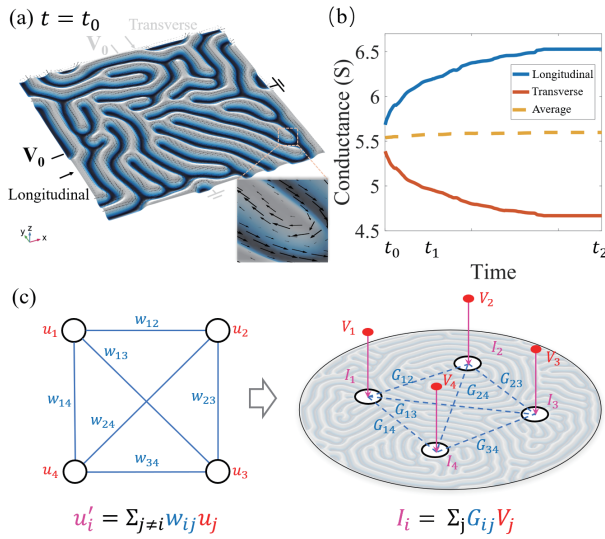


Figure 6 (Color online) (a) Snapshot of a maze domain structure. (b) With applied voltage, current flowing through the magnetic textures exerts spin-transfer torque, so that the spatial distribution of magnetic configuration as well as conductance state evolve with time. (c) The model of a four-node Hopfield neural network is mapped to a magnetic device with four electrodes, with input, output and weight encoded as voltage V_j , current I_i and conductance G_{ij} respectively. Reprinted with permission from ref. [64].

physical Hopfield neural network can be expressed as:

$$E = - \sum_{i,j} G_{ij} [\mathbf{m}(\mathbf{r})] V_i V_j, \quad (5)$$

where V_i is binary voltage applied on each electrode and $G_{ij}[\mathbf{m}(\mathbf{r})]$ is element of conductance matrix which depends on spatial distribution of magnetic textures, which is determined during the self-learning process. Because of the plasticity of the magnetic texture, electric currents can train the network by modulating the texture as well as corresponding conductance (weight) matrix, without interference of external computational algorithms.

In conventional training strategies for Hopfield neural networks, the synaptic weights are determined by computers using algorithms such as the outer product method $w_{ij} = \sum_{s=1}^k x_i^s x_j^s$ with $w_{ii} = 0$ [138, 139], where $\mathbf{x}^s$ represents the state vector of the pattern s being memorized. However, the approach proposed by Yu et al. [64] differs from conventional implementations. Instead of relying on external computation, the malleable magnetic textures autonomously train the network according to an incremental Hebbian learning principle:

$$G_{ij}^{n+1} = G_{ij}^n + \Delta_{ij}^n [V_i]. \quad (6)$$

A possible explicit form of this learning rule has been recently investigated²⁾, and it shares similarities with the well-known Oja's rule [140] which incorporates a “forgetting”

term to stop the unconstrained growing of weights guided by original Hebbian's rule. Moreover, this self-learning mechanism aligns conceptually with the synaptic plasticity rule proposed by Krotov and Hopfield [141]. The incorporation of synaptic “plasticity” and “competence” required by Hebb's rule is believed to influence memory formation in neuroscience [123, 142], and now these principles are intrinsically built into the physical process.

While the self-learning mechanism in malleable magnetic textures shows promise, there are several aspects that require further investigation. For example, it is important to explore whether the memory capacity [143-145] can be further improved using the physical self-learning rule and what is the scaling law of self-learning process on either energy consumption or learning duration when the number of nodes increases. Additionally, investigating the applicability of the self-learning strategy in multilayer networks, such as back-propagation [146], would be valuable. Recent experimental verification of this concept has been conducted using field-driven permalloy films, where the gradient descent nature of the learning rule has been discovered²⁾. Kumar et al. [147] have also demonstrated a similar intrinsic Hebbian rule experimentally using hybrid volatile/nonvolatile memory devices with ternary metal oxide as the active material. The concept is expected to extend to higher dimensions, such as three-dimensional nanoscale gyroid networks [148].

4.4 An atomic Boltzmann machine capable of self-adaption

Boltzmann machines are recurrent neural networks that employ stochastic neurons and are capable of solving various challenging computational tasks [149-151]. Despite sharing a similar structure with Hopfield neural networks, finding an effective training algorithm for Boltzmann machines has been challenging due to their stochastic nature. Ackley et al. [152] proposed a typical learning rule for Boltzmann machines, where the weight change Δw_{ij} is given by $\Delta w_{ij} = \eta(p_{ij} - p'_{ij})$. Here, p_{ij} represents the average probability of both connected units being in an excited state when the visible units are clamped, p'_{ij} represents the same probability when the network is running freely, and η is the learning rate that scales the size of each weight change. Notably, this learning rule is local, relying only on the behavior of the two connected units, without the need for global statistical measures.

Boltzmann machines are powerful tools for solving problems in combinatorial optimization [153, 154], but their training algorithm poses challenges and tends to struggle with

2) C. Niu, et al. Self-learning magnetic textures with intrinsic gradient descent adaption. Unpublished.

large problem sizes [155]. Inspired by such a local learning rule, Kaiser et al. [156] designed a probabilistic circuit for autonomous learning, mapping a continuous version of the Boltzmann machine learning rule to a clockless autonomous circuit. While this learning rule can be implemented in a top-down manner, the effective implementation of such a rule in a stochastic physical system remains a challenging question.

In recent research, Kiraly et al. [157] made significant progress in pushing the concept of self-learning to the atomic level by creating a self-adapting Boltzmann machine using cobalt (Co) atoms on black phosphorus. They represented the stochastic neuron physically using the probability distribution of surface Co atoms, where each atom could occupy one of two possible locations, as probed by scanning tunneling microscopy (STM). In their setup, a configuration of three Co atoms was used, with two atoms representing stochastic neurons (s) and the third atom (k) influencing the synaptic weight w_{12} through substrate-mediated interactions (Figure 7(a)). Interestingly, the synaptic weight w_{12} could be tuned in a binary fashion depending on the position of atom k ($k = 0$ or 1), which in turn affected the probability distribution $P(s|k)$ of the two Co atoms (Figure 7(b)-(d)). Importantly, this self-adapting process occurred without the need for external computational power. The authors also demonstrated the ability to achieve multi-valued synaptic weights using additional Co atoms and the capability to tune the synaptic weights in a semi-continuous manner by adjusting the STM tip voltages. While challenges such as scalability and the underlying learning rule still need to be addressed, this work establishes a promising connection between atomic-level physical interactions and the self-learning capabilities of physical neural networks.

4.5 Spiking neural networks with STDP learning rule

Spiking neural networks (SNN) have emerged as one of the most promising platforms in both the machine learning and material science communities [6, 101, 106, 158-160]. Although still in its early stages compared to deep neural networks (DNN), SNN chips offers ultralow power consumption by encoding and delivering information in spikes, similar to the functioning of the human brain. Notably, SNN chips such as BrainScaleS, TrueNorth, and Loihi have demonstrated remarkable energy efficiency in implementing spiking-based training algorithms through offline or online training approaches [14, 161].

In the realm of learning in physical systems, spike timing dependent plasticity (STDP) has been extensively studied as an unsupervised learning rule across various material systems. STDP is a synapse-specific Hebbian form of plasticity that implements Hebb's original hypothesis by strengthening

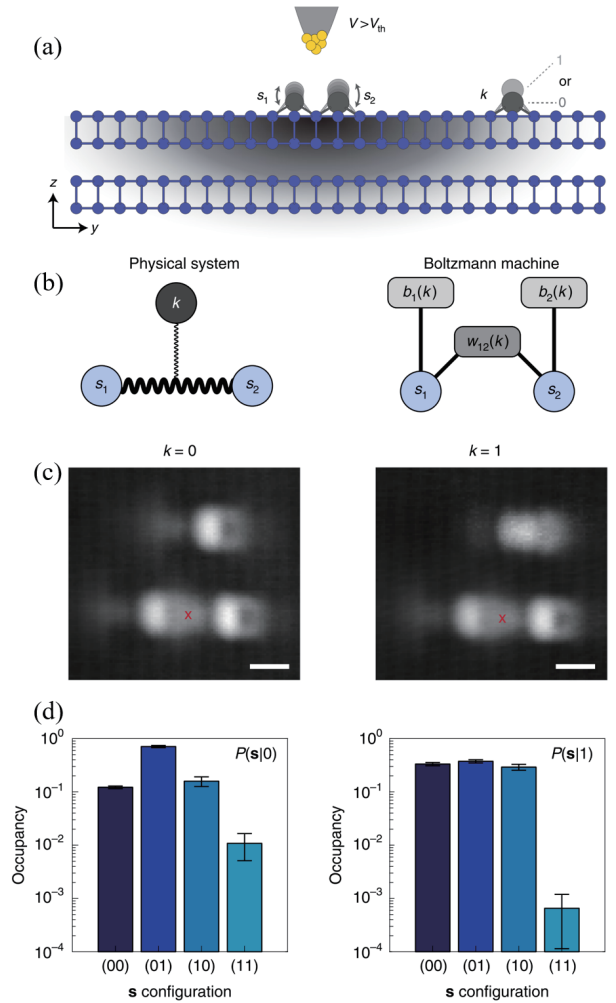


Figure 7 (Color online) Atomic Boltzmann machine with self-adaption. (a) Illustration of physical system consisting of Co atoms on black phosphorus surface, probed by STM technique. (b) Representation of physical system in (a) that mimics Boltzmann machine. (c) STM topography images of two Co atoms s with a third atom at state $k = 0, 1$. (d) Corresponding probability distributions $P(s|k)$ of two atoms (neurons) s with $k = 0, 1$, respectively. Reprinted with permission from ref. [157].

synapses when the presynaptic neuron fires before the postsynaptic neuron [4, 162]. This plasticity captures the fundamental workings of biological neurons and synapses, where chemical synapses can be excitatory or inhibitory, promoting or inhibiting the membrane action potential, respectively (Figure 8(a)) [160]. Jo et al. [37] pioneered the implementation of synaptic STDP behavior in memristors, where the synaptic weight is physically represented as electrical conductance in a two-terminal junction, modulated by the difference in pre and post spike timing (Figure 8(b)) [31, 32].

The utilization of STDP in physical systems offers several distinct advantages: (1) STDP behavior is applicable to a wide range of materials, including ferroelectric [51, 163], magnetic tunnel junctions [50, 164], spin-orbit-torque

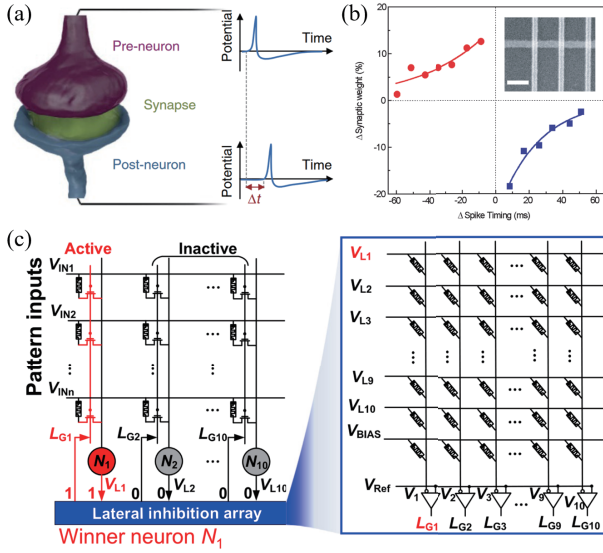


Figure 8 (Color online) Spiking neural networks with STDP for physical learning. (a) Illustration of biological synapses and neurons that encode and pass information with spike signals. Reprinted with permission from ref. [51]. (b) STDP behavior in memristive nanodevices. Reprinted with permission from ref. [37]. (c) Schematic of a lateral inhibition array circuit with crossbar array structures. Reprinted with permission from ref. [168].

devices [165], Mott materials [166], and optics [106]. This enables collaboration and advancements across multiple research communities, bridging materials science and computational science. (2) Many of these materials are CMOS compatible and can be fabricated as crossbar arrays (Figure 8(c)), facilitating analog calculations of matrix multiplication directly within the materials. This alleviates the energy consumption typically associated with software-based operations. (3) In SNN, neuronal behavior can be emulated using models such as leaky integrate-and-fire (LIF), which can be achieved with phase change, magnetic, ferroelectric, and Mott materials. Consequently, the development of fully material-implemented SNN hardware enables highly energy-efficient on-chip learning and self-learning in memristor crossbar arrays [101, 167, 168] and all-optical neural networks [106]. Ongoing research in this field focuses on improving benchmarks, including linearity, operation voltage, endurance, and other factors, to further enhance energy efficiency and network performance.

Despite the successful integration of STDP and physical devices, this research field is still in its early stages. In terms of SNN, STDP represents a relatively simple learning rule. SNNs, in general, are challenging to train due to the non-differentiable nature of spikes, and it remains uncertain whether SNN performance can match that of DNN-based algorithms. Active research areas include learning mechanisms such as prescribed error sensitivity, ANN-to-SNN conversion, and various forms of STDP [160]. The feasibility of implementing these learning mechanisms in materials remains

largely unexplored. Additionally, the monotonic increment in memristors may lead to synaptic conductance saturation, which necessitates additional circuit designs to address this issue. At the architectural level, constructing functional autonomous networks with spiking models for relevant tasks is an inspiring research direction [169]. The exploration of new physical systems with dynamics capable of responding to various network geometries, spike stimuli, and algorithms can introduce novel aspects of efficient learning in materials.

5 Learning strategies towards physical self-learning

While the concept of physical self-learning is still emerging, significant efforts have been dedicated to achieving efficient learning in PNNs. In this regard, various learning strategies have been explored on analog hardware as concluded in Table 1 [9, 10, 60, 61, 64, 65, 69, 70, 72, 74, 75, 77, 82, 93, 96, 97, 101, 104, 106, 111–113, 119, 121, 122, 157, 168, 170–181], moving away from digital approaches. These strategies leverage the intrinsic dynamics of physical systems and aim to minimize reliance on external computation, thereby paving the way for the implementation of physical learning. Although it remains uncertain whether these strategies fully achieve the goal of physical self-learning, they undoubtedly contribute to the advancement of this field.

Incremental Hebbian learning. Taking Hopfield network as an example, the incremental Hebbian learning was applied by Hopfield [128] to learn a new memory $\mathbf{x}^s$ and the increment of weight matrix elements takes the form as:

$$\Delta w_{ij} \propto x_i^s x_j^s, \quad (\text{incremental Hebbian learning}). \quad (7)$$

In order to enhance the performance of the network in accessing real memories and in minimizing spurious ones, Hopfield et al. [182] further proposed an unlearning strategy, where the final equilibrium state $\mathbf{x}^f$ is weakly unlearned by the incremental change

$$\Delta w_{ij} \propto -\epsilon x_i^f x_j^f, \quad (\text{unlearning}), \quad (8)$$

with $0 < \epsilon \ll 1$, so that the accessibility of stored memories can be made more nearly even. Other strategies have been proposed such as Grossberg's rule [183] and Oja's rule which considered "forgetting" mechanism by adding a constraint term into the learning rule [140]. Krotov and Hopfield [141] extended the Oja's rule to a biologically inspired learning algorithm, called synaptic plastic rule which considers competition between hidden units and implements the Hebbian-like plasticity subject to homostatic constraints on the length of synaptic weights. The strategies related to incremental Hebbian learning have been widely applied in top-down approach

Table 1 Some typical neuromorphic architectures and their features

Architectures	Design approach	Underlying physical process (Learning concept)	Physical substrate	Learning/inference mode	Demonstration
PNNs with crossbar array structures [60, 61, 93, 96, 97, 101, 170, 171]	top-down	digital learning	memristors, ferroelectrics, phase change materials, etc.	learn digitally, infer physically	image classification and speech recognition etc.
Physical reservoirs [10, 69, 70, 72, 74, 75, 77, 111, 172-177]	top-down	digital learning	artificial spin ices, dynamic memristors etc.	learn digitally, infer physically	time series prediction and spoken-digit recognition etc.
Photonic neural networks [178-181]	top-down	backpropagation	photonic chips	learn digitally, infer physically	data/image classification
Memristive SNN [168]	top-down	STDP rule	threshold switching and resistive switching devices	learn & infer physically	pattern recognition
All-optical SNN [106]	top-down	STDP rule	photonic waveguides and phase change materials	learn & infer physically	pattern recognition
Deep PNNs [9, 82]	bottom-up	backpropagation and physical local learning	mechanical, optical and electronic systems	learn & infer physically	image classification and vowel classification
Magnetic Hopfield network [64]	bottom-up	intrinsic Hebbian (Oja) learning	magnetic textures	learn & infer physically	associative memory
Atomic Boltzmann machine [157]	bottom-up	self-adaptive orbital memory	Co atoms on black phosphorus	learn & infer physically	Monte Carlo sampling
Nanoparticle networks [104]	bottom-up	Coulomb blockade and single-electron tunneling	Au nanoparticles	learn & infer physically	Boolean logic
Nanocluster networks [65, 119, 121]	bottom-up	aggregation of nanoclusters with positive feedback	Ag nanowires and nanoparticles	learn & infer physically	gradient descent route planning
Nonlinear elastic network [113, 122]	bottom-up	physical continual learning and physical coupled learning	actin networks, kirigami structure etc.	learn & infer physically	pattern recognition
Adaptive flow network [112]	bottom-up	global cost and function optimization	vasculatures, nonlinear resistors etc.	learn & infer physically	memory encoding

described in Figure 2. Regarding to physical self-learning, one straightforward way is to look for a non-volatile physical quantity as weights w_{ij} (e.g., conductance or resistance), whose increment is a function of input physical quantities $\mathbf{x}$ (e.g., voltages) as implemented by ref. [64].

Contrastive Hebbian learning. The strategy is a generalization of the incremental Hebbian learning including unlearning process mentioned above, which updates the weights proportionally to the difference in the cross products of activations in a clamped and a free running phase which can be applied in continuous Hopfield network [184] and Boltzmann machines [185]. The key idea of contrastive learning is to introduce a contrastive function $J = \bar{E}^C - \bar{E}^F$ which is difference of energy functions at equilibrium (denoted by “ $\bar{\cdot}$ ”) when the inputs and outputs are both clamped (C), and when the inputs are clamped and outputs are free (F), respectively. Therefore, the system performs the learning successfully when $J = 0$. By differentiating the contrastive function $\partial J / \partial w_{ij}$, one can obtain the contrastive learning rule

$$\Delta w_{ij} \propto \bar{x}_i^C \bar{x}_j^C - \bar{x}_i^F \bar{x}_j^F, \quad (\text{contrastive Hebbian learning}), \quad (9)$$

with $\bar{\mathbf{x}}^C$ and $\bar{\mathbf{x}}^F$ the state vector at equilibrium for clamped phase and freely running phase. After each learning step, the

difference in energy at equilibrium between the clamped and free states becomes smaller. It needs to be remarked that the contrastive Hebbian learning is equivalent to backpropagation learning when there are no hidden units [184], i.e.,

$$\Delta w_{ij} \propto I_i f'(x_j)(t_j - x_j), \quad (\text{backpropagation, no hidden units}), \quad (10)$$

with t_j the teacher vector at outputs, I_i the input vector, and f' the derivative of activation function. Backpropagation is one of the most successful algorithms to train neural networks [8]. However, it is considered neither biologically nor physically plausible [186] due to its requirement of a special computational circuit during training, requirement of accurate knowledge about the whole physical system [187] and the need for analytically calculating the derivative of the activation function [188].

Equilibrium propagation. Equilibrium propagation is a learning framework for energy-based models, sharing similarities with contrastive Hebbian learning and backpropagation, but requiring only one computational circuit and one type of computation for training [189]. Equilibrium propagation involves two phases: (1) In the first phase, the inputs are weakly clamped, and the network relaxes to a fixed point.

Predictions are then read out from the output units. (2) In the second phase, the outputs are nudged toward their targets, and the network relaxes to a new fixed point corresponding to a smaller prediction error. The system learns the second-phase fixed point by reducing its energy while unlearning the first-phase fixed point by increasing its energy.

Equilibrium propagation differs from contrastive Hebbian learning in certain aspects. The learning rule for equilibrium propagation can be expressed as:

$$\Delta w_{ij} \propto \frac{1}{\beta} (x_i^\beta x_j^\beta - x_i^0 x_j^0), \quad (\text{equilibrium propagation}), \quad (11)$$

where $\mathbf{x}^0$ is state vector at freely running phase (equivalent to $\mathbf{x}^F$ in the case of contrastive Hebbian learning) and $\mathbf{x}^\beta$ is the state vector at weakly clamped phase with β the clamping factor that controls whether the output is pushed toward the target or not and by how much [189]. When $\beta \rightarrow +\infty$, the learning rule comes to the strong clamped scenario as contrastive Hebbian learning ($\mathbf{x}^\beta \rightarrow \mathbf{x}^C$). The concept of equilibrium propagation has found applications in training electric resistor networks [190, 191] and an Ising machine [192]. Additionally, there are other strategies closely related to equilibrium propagation. One such strategy is contrastive divergence, which is used for training Boltzmann machines [193]. Another related strategy is recurrent backpropagation [146].

Physical coupled learning. Inspired by the concept of contrastive Hebbian learning and equilibrium propagation, Stern et al. [113] proposed a general and physically plausible framework called “coupled local supervised learning” which has two phases, namely free phase and clamped phase. Given any physical network with a known physical cost function $E(\mathbf{x}, \mathbf{w})$, one can drive the relevant coupled learning rule directly according to

$$\Delta w_{ij} \propto -\partial_{w_{ij}}(E^F - E^C), \quad (\text{physical coupled learning}) \quad (12)$$

where the weights w_{ij} are physical learning degrees of freedom such as pipe conductance in flow networks and spring constant in elastic networks. Because the cost function E can be written as a sum over individual costs of edges $E = \sum_i E_i$, the learning rule is local, i.e., each edge independently adjusts its learning degrees of freedom taking into account only its own response to the applied boundary conditions. Such locality of learning rules promises almost same amount of learning time when applied in networks with larger size. As a proof of principle, the strategy of coupled learning has been experimentally applied to electric resistor networks based on variable resistors [194–196]. In these cases, a circuitry is required to locally compare the response of two identical networks subjected to different sets of boundary conditions. The effect of directed aging demonstrated by Pashine et al. [23]

can be seen as a simplified form of coupled learning in the absence of the free term. Anisetti et al. [197] considered a flow network where chemicals can spread via a diffusion process, which can increase or decrease the conductance of pipelines. This approach allows the learning rule to compute gradients using local information for each weight without the need to store the free state. The work by Anisetti et al. [197] provides inspiration for conducting coupled learning using two independent physical quantities, which ensures non-interference between the input signal and the feedback signal.

Physical local learning. Momeni et al. [82] proposed a simple deep neural network architecture augmented by a physical local learning (PhyLL), which enables one to perform supervised and unsupervised learning on a deep PNN without detailed knowledge of the nonlinear physical layer’s properties. The strategy is closely related to the physics-aware learning [9], which aims to train a deep PNN by performing backpropagation on a twin digital model. The improvement of physical local learning is to replace the standard backward pass (performed by a digital computer) with an additional single forward pass through a physical system, eliminating the twin modelling phase. Instead of a forward and backward pass applied by standard backpropagation method, PhyLL shares similarity with the concept of “contrastive learning” by using two physical forward passes: a positive and a negative forward pass through the physical system, each running on different physical inputs. In each passes, the datasets as well as the correct (incorrect) labels are input together into the multiplayer PNNs and the output on each layer can be expressed as:

$$\mathbf{y}^{\text{pos/neg}} = \mathbf{w}_t \cdot \mathcal{F}(\mathbf{w}_p \cdot \mathbf{x}^{\text{pos/neg}}), \quad (13)$$

where $\mathbf{w}_p$ is untrainable weight matrix provided by physical interconnections and $\mathbf{w}_t$ is trainable weight matrix to be updated by minimizing a loss function

$$L = \log(1 + \exp(\theta(\cos_{\text{sim}}(\mathbf{y}^{\text{pos}}, \mathbf{y}^{\text{neg}})))), \quad (14)$$

with θ a scale factor and $\cos_{\text{sim}}$ denotes the cosine similarity defined as the cosine of the angle between the two arguments. One can obtain the learning rule according to

$$\Delta \mathbf{w}_t \propto -\frac{\partial L}{\partial \mathbf{w}_t}, \quad (\text{physical local learning}). \quad (15)$$

During the training of deep neural networks using backpropagation, three primary steps must be accomplished: (1) forward propagation, (2) error backpropagation, and (3) gradient computation (updating) steps. PhyLL demonstrates the capability to execute the initial two steps utilizing physical systems, leading to a notable 2/3 enhancement in both speed and energy efficiency when compared to fully digital learning [82]. In contrast, physics-aware learning achieves an approximate 1/3 improvement and the ultimate goal of physical

self-learning is to accomplish a perfect 3/3, whereby every step of the learning process occurs within physical systems.

Hamiltonian echo backpropagation. López-Pastor and Marquardt [107] introduced a general scheme for self-learning in any time-reversible Hamiltonian system. Although the idea of physical backpropagation has been proposed in optical neural networks decades ago [181] and more novel implementations have been realized thereafter [178-180, 198, 199], all presently existing proposals are suitable only for some setups with very specific nonlinearities. According to López-Pastor and Marquardt, the self-learning approach consists of three main steps: (1) Forward pass: The evaluation field $\Psi(-T)$ and the learning field $\Theta(-T)$ (which can represent any physical degrees of freedom in the Hamiltonian system) are prepared at time $t = -T$ and evolve according to Hamilton's equations for a given Hamiltonian. (2) At time $t = 0$, a perturbed echo is injected, and a time-reversal operation is performed. (3) Backward pass: The learning field continues to evolve until time $t = T$, and the final result of the entire process is equivalent to the following update of the learning field:

$$\Theta(T) = \Theta(-T) - i\epsilon \frac{\partial C}{\partial \Theta^*(-T)}, \text{ (echo backpropagation), } (16)$$

where C is the sample-specific cost function, Θ^* is the conjugate of learning field, and ϵ can be understood as the magnitude of variation as a perturbation travelling on top of the time-reversed field from output to input. This self-learning approach requires the physical system to be Hamiltonian and obey time-reversal symmetry. Examples of suitable physical platforms include, but are not limited to, nonlinear nanophotonic circuits [17, 54], superconducting microwave cavities [200-202], cold atoms trapped in optical lattices [203], and spin wave systems [204-206] without broken time-reversal symmetry.

6 Discussion and outlook

In the last section, we discuss distinctive features, challenges, and opportunities associated with the concept of physical self-learning.

Common attributes required for self-learning. As discussed in sect. 4, several common attributes are observed across various physical systems. To emulate Hebbian's principle, a positive feedback process is invariably necessary, which can be commonly observed in a nonlinear conducting system whose conductance strengthens with the flow of current, as introduced in sect. 4.1. However, a monotonous increase in synaptic weight is not universally feasible, either biologically or physically. This typically necessitates a global

constraint on weight evolution, and the specific form of this constraint is closely linked to the nonlinear characteristics of physical systems (sects. 4.2 and 4.3). Moreover, the physical quantities employed to encode weights need not be static and deterministic; they can be dynamic and stochastic, even in the absence of an explicit form (sects. 4.4 and 4.5). As discussed in sect. 5, the progression of learning strategies in PNNs begins with early versions such as incremental Hebbian learning (learning rule as a function of training data) and advances towards later versions like physical coupled learning (learning rule as a function of feedback from the physical systems). This shift signifies a departure from solely relying on training data and instead harnesses the rich dynamics and interactions within the physical domain.

Perfection vs imperfection. Modern architectures for neuromorphic computing, such as memristor crossbar arrays [85, 207, 208], strive for perfection. In these architectures, the variation of devices or computing units (such as memristors) is expected to be minimal, as training strategies assume ideal circumstances. However, a gap exists between the perfect neural network model and the imperfect realistic devices. This discrepancy poses challenges, as widely used backpropagation training algorithms for offline and online learning are generally incompatible with hardware due to mismatches between analytically calculated training information and the imprecision of actual analog devices. To address this issue, training strategies such as the activity-difference technique [170] and physics-aware training [9] have been developed. On the other hand, the autonomous learning process in PNNs appears to be more tolerant of imperfections. Strategies like coupled learning and Hamiltonian echo backpropagation are less sensitive to device variability and can effectively handle noise [107].

Nonlocal rules vs local rules. Conventional learning rules implemented on digital computers are generally nonlocal, obtained by performing gradient descent on a global cost function. Consequently, the weight updates have a complex nonlocal dependence on other neurons in the network, making it challenging to implement traditional strategies like backpropagation in PNNs [107, 113]. On the other hand, the physical learning strategies discussed in this review are all local. This means that the weight update between two neurons depends solely on the state of those two neurons and is independent of other neurons in the network. As a result, these local rules enhance the physical plausibility of PNNs, and the weight updates can be performed in parallel, leading to promising scalability of learning speed with increasing network size. While simple physical networks with local learning rules are sufficient for achieving physical self-learning, it remains an open question whether nonlocal learning rules exist in physical systems and whether they can improve learn-

ing performance.

Potential applications. In this review, we primarily focus on the implementation of physical (self-)learning in typical structures such as artificial neural networks (ANNs), spiking neural networks (SNNs), and recurrent Hopfield networks. However, extending the concept of self-learning to other structures presents intriguing opportunities. For instance, non-volatile memories acting as artificial synapses can be replaced by wireless connections [209], and artificial neurons with spiking dynamics can be substituted with nonlinear oscillators [67]. This allows for the construction of cellular neural networks (CNNs) where units or cells communicate with spatially dependent connections [210]. The concept of Kohonen neural networks can be applied by considering short-range lateral feedback between neighboring neurons. This enables self-organizing mapping from multi-dimensional data to lower-dimensional representations [211]. Such self-adaptive or self-organized behaviors are expected to be prevalent in physical systems. Constructing continuous attractor neural networks can predict the trajectories of moving objects without the need for training [212]. It is anticipated that more novel functions can be realized by applying physical self-learning to dynamical synapses. PNNs with self-learning capabilities can be combined with front-end devices to perform multimodal in-sensor computing [213]. For instance, artificial retinas can be designed by connecting self-adaptive PNNs with photoelectric receivers [214, 215]. Lastly, PNNs can enter the quantum regime by leveraging the infinite degrees of freedom in phase space [216, 217].

Challenges and opportunities. While significant advances have been made in realizing physical self-learning, both challenges and opportunities remain for this promising field. One of the key obstacles lies in enhancing device properties, such as nonlinear conductance thresholds, endurance, and variability tolerance. Additionally, there are unresolved architectural questions pertaining to the implementation of diverse network structures and complex algorithms within physical systems. At a systems level, comprehending emergent behaviors arising from collective physical dynamics poses analytical difficulties. From a materials standpoint, there are exciting prospects in identifying new platforms with embedded intelligence, such as 2D [25, 218] or spintronic [18] systems. Boosting capabilities can be achieved by developing learning rules inspired by life-like principles of homeostasis, competition, and memory formation. Moreover, integrating PNNs with sensing and actuation capabilities empowers potential applications like near-sensor computing [213], artificial retinas [219] and autonomous robotics [5]. By addressing these challenges through multidisciplinary collaboration, there is the potential to revolutionize computing using intelligent matter programmed by the laws of physics.

Conclusion. In this review, we have explored the concept of physical learning, particularly physical self-learning, which represents a promising frontier in the field of neuromorphic computing. Our investigation suggests that physical systems have the potential to exhibit intelligence by interacting with the environment, receiving and responding to external stimuli, and internally adapting their structures to facilitate the distribution and storage of information [220]. Generally speaking, the concept of physical self-learning can be understood as an optimization process performed by physical optimization machines, which evolve based on the principle such as least action and least power dissipation [221]. By delving deeper into the concept of self-learning, we aim to gain a better understanding of natural intelligence [222, 223]. Achieving this goal requires collaboration among various scientific and engineering disciplines, as they contribute to the exploration and advancement of self-learning.

This work was supported by the National Key Research and Development Program of China (Grant Nos. 2022YFA1403300, and 2020YFA0309100), the National Natural Science Foundation of China (Grant Nos. 12204107, and 12074073), Shanghai Municipal Science and Technology Major Project (Grant No. 2019SHZDZX01), Shanghai Pujiang Program (Grant No. 21PJ1401500), and Shanghai Science and Technology Committee (Grant Nos. 21JC1406200, and 20JC1415900). Weichao Yu acknowledges fruitful discussion with Huanyu Zhang, Chuanlong Xu, and Lin Xiong.

Conflict of interest The authors declare that they have no conflict of interest.

- 1 D. Marković, A. Mizrahi, D. Querlioz, and J. Grollier, *Nat. Rev. Phys.* **2**, 499 (2020), arXiv: 2003.04711.
- 2 S. Furber, *J. Neural Eng.* **13**, 051001 (2016).
- 3 C. Mead, *Proc. IEEE* **78**, 1629 (1990).
- 4 D. S. Jeong, K. M. Kim, S. Kim, B. J. Choi, and C. S. Hwang, *Adv. Electron. Mater.* **2**, 1600090 (2016).
- 5 J. Pei, L. Deng, S. Song, M. Zhao, Y. Zhang, S. Wu, G. Wang, Z. Zou, Z. Wu, W. He, F. Chen, N. Deng, S. Wu, Y. Wang, Y. Wu, Z. Yang, C. Ma, G. Li, W. Han, H. Li, H. Wu, R. Zhao, Y. Xie, and L. Shi, *Nature* **572**, 106 (2019).
- 6 Y. Zhang, Z. Wang, J. Zhu, Y. Yang, M. Rao, W. Song, Y. Zhuo, X. Zhang, M. Cui, L. Shen, R. Huang, and J. Joshua Yang, *Appl. Phys. Rev.* **7**, 011308 (2020).
- 7 D. R. Chialvo, *Nat. Phys.* **6**, 744 (2010), arXiv: 1010.2530.
- 8 R. Rojas, *Neural Networks: A Systematic Introduction* (Springer, Berlin, Heidelberg, 2013).
- 9 L. G. Wright, T. Onodera, M. M. Stein, T. Wang, D. T. Schachter, Z. Hu, and P. L. McMahon, *Nature* **601**, 549 (2022).
- 10 X. Liang, J. Tang, Y. Zhong, B. Gao, H. Qian, and H. Wu, *Nat. Electron.* **7**, 193 (2024).
- 11 P. Mehta, M. Bukov, C. H. Wang, A. G. R. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, *Phys. Rep.* **810**, 1 (2019), arXiv: 1803.08823.
- 12 R. Iten, T. Metger, H. Wilming, L. del Rio, and R. Renner, *Phys. Rev. Lett.* **124**, 010508 (2020), arXiv: 1807.10300.
- 13 C. Gigli, A. Saba, A. B. Ayoub, and D. Psaltis, *APL Photon.* **8**, 026105 (2023), arXiv: 2208.05793.
- 14 W. Zhang, B. Gao, J. Tang, P. Yao, S. Yu, M. F. Chang, H. J. Yoo, H. Qian, and H. Wu, *Nat. Electron.* **3**, 371 (2020).

- 15 X. Sui, Q. Wu, J. Liu, Q. Chen, and G. Gu, *IEEE Access* **8**, 70773 (2020).
- 16 B. J. Shastri, A. N. Tait, T. Ferreira de Lima, W. H. P. Pernice, H. Bhaskaran, C. D. Wright, and P. R. Prucnal, *Nat. Photon.* **15**, 102 (2021), arXiv: [2011.00111](#).
- 17 F. Ashtiani, A. J. Geers, and F. Aflatouni, *Nature* **606**, 501 (2022), arXiv: [2106.11747](#).
- 18 J. Grollier, D. Querlioz, K. Y. Camsari, K. Everschor-Sitte, S. Fukami, and M. D. Stiles, *Nat. Electron.* **3**, 360 (2020).
- 19 J. Zhou, and J. Chen, *Adv. Electron. Mater.* **7**, 2100465 (2021).
- 20 Y. J. Zhang, Q. Zheng, X. R. Zhu, Z. Yuan, and K. Xia, *Sci. China-Phys. Mech. Astron.* **63**, 277531 (2020).
- 21 W. Cai, Y. Huang, X. Zhang, S. Wang, Y. Pan, J. Yin, K. Shi, and W. Zhao, *Sci. China-Phys. Mech. Astron.* **66**, 117503 (2023).
- 22 S. E. Russek, C. A. Donnelly, M. L. Schneider, B. Baek, M. R. Pufall, W. H. Rippard, P. F. Hopkins, P. D. Dresselhaus, and S. P. Benz, in *Stochastic single flux quantum neuromorphic computing using magnetically tunable Josephson junctions: Proceedings of the 2016 IEEE International Conference on Rebooting Computing (ICRC)* (San Diego, 2016), pp. 1-5.
- 23 N. Pashine, D. Hexner, A. J. Liu, and S. R. Nagel, *Sci. Adv.* **5**, eaax4215 (2019), arXiv: [1903.05776](#).
- 24 M. Stern, C. Arinze, L. Perez, S. E. Palmer, and A. Murugan, *Proc. Natl. Acad. Sci. USA* **117**, 14843 (2020).
- 25 K. Zhu, S. Pazos, F. Aguirre, Y. Shen, Y. Yuan, W. Zheng, O. Alharbi, M. A. Villena, B. Fang, X. Li, A. Milozzi, M. Farronato, M. Muñoz-Rojo, T. Wang, R. Li, H. Fariborzi, J. B. Roldan, G. Benstetter, X. Zhang, H. N. Alshareef, T. Grasser, H. Wu, D. Ielmini, and M. Lanza, *Nature* **618**, 57 (2023).
- 26 J. del Valle, J. G. Ramírez, M. J. Rozenberg, and I. K. Schuller, *J. Appl. Phys.* **124**, 211101 (2018), arXiv: [1812.01120](#).
- 27 M. Zhao, B. Gao, J. Tang, H. Qian, and H. Wu, *Appl. Phys. Rev.* **7**, 011301 (2020).
- 28 E. Khoram, A. Chen, D. Liu, L. Ying, Q. Wang, M. Yuan, and Z. Yu, *Photon. Res.* **7**, 823 (2019).
- 29 D. V. Christensen, R. Dittmann, B. Linares-Barranco, A. Sebastian, M. Le Gallo, A. Redaelli, S. Slesazeck, T. Mikolajick, S. Spiga, S. Menzel, I. Valov, G. Milano, C. Ricciardi, S. J. Liang, F. Miao, M. Lanza, T. J. Quill, S. T. Keene, A. Salleo, J. Grollier, D. Marković, A. Mizrahi, P. Yao, J. J. Yang, G. Indiveri, J. P. Strachan, S. Datta, E. Vianello, A. Valentian, J. Feldmann, X. Li, W. H. P. Pernice, H. Bhaskaran, S. Furber, E. Neftci, F. Scherr, W. Maass, S. Ramaswamy, J. Tapon, P. Panda, Y. Kim, G. Tanaka, S. Thorpe, C. Bartolozzi, T. A. Cleland, C. Posch, S. C. Liu, G. Panuccio, M. Mahmud, A. N. Mazumder, M. Hosseini, T. Mohsenin, E. Donati, S. Tolu, R. Galeazzi, M. E. Christensen, S. Holm, D. Ielmini, and N. Pryds, *Neuromorph. Comput. Eng.* **2**, 022501 (2022).
- 30 L. Sun, Y. Zhang, G. Han, G. Hwang, J. Jiang, B. Joo, K. Watanabe, T. Taniguchi, Y. M. Kim, W. J. Yu, B. S. Kong, R. Zhao, and H. Yang, *Nat. Commun.* **10**, 3161 (2019).
- 31 L. Chua, *IEEE Trans. Circuit Theor.* **18**, 507 (1971).
- 32 D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, *Nature* **453**, 80 (2008).
- 33 G. W. Burr, R. M. Shelby, A. Sebastian, S. Kim, S. Kim, S. Sidler, K. Virwani, M. Ishii, P. Narayanan, A. Fumarola, L. L. Sanches, I. Boybat, M. Le Gallo, K. Moon, J. Woo, H. Hwang, and Y. Leblebici, *Adv. Phys.-X* **2**, 89 (2017).
- 34 S. Yu, *Proc. IEEE* **106**, 260 (2018).
- 35 S. Ambrogio, P. Narayanan, H. Tsai, R. M. Shelby, I. Boybat, C. di Nolfo, S. Sidler, M. Giordano, M. Bodini, N. C. P. Farinha, B. Killeen, C. Cheng, Y. Jaoudi, and G. W. Burr, *Nature* **558**, 60 (2018).
- 36 B. Rajendran, and F. Alibart, *IEEE J. Emerg. Sel. Top. Circuits Syst.* **6**, 198 (2016).
- 37 S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, *Nano Lett.* **10**, 1297 (2010).
- 38 R. Mishra, D. Kumar, and H. Yang, *Phys. Rev. Appl.* **11**, 054065 (2019).
- 39 C. Niu, Y. Zhao, W. Hu, Q. Shi, T. Miao, Y. Yu, L. Yin, J. Xiao, H. Guo, and J. Shen, *Appl. Phys. Lett.* **119**, 204101 (2021).
- 40 M. Rao, H. Tang, J. Wu, W. Song, M. Zhang, W. Yin, Y. Zhuo, F. Kiani, B. Chen, X. Jiang, H. Liu, H. Y. Chen, R. Midya, F. Ye, H. Jiang, Z. Wang, M. Wu, M. Hu, H. Wang, Q. Xia, N. Ge, J. Li, and J. J. Yang, *Nature* **615**, 823 (2023).
- 41 V. Milo, D. Ielmini, and E. Chicca, in *Attractor networks and associative memories with STDP learning in RRAM synapses: Proceedings of the 2017 IEEE International Electron Devices Meeting (IEDM)*, (IEEE, San Francisco, 2017).
- 42 Y. Huang, W. Kang, X. Zhang, Y. Zhou, and W. Zhao, *Nanotechnology* **28**, 08LT02 (2017), arXiv: [1608.07955](#).
- 43 S. Li, W. Kang, Y. Huang, X. Zhang, Y. Zhou, and W. Zhao, *Nanotechnology* **28**, 31LT01 (2017).
- 44 S. Zhang, S. Luo, N. Xu, Q. Zou, M. Song, J. Yun, Q. Luo, Z. Guo, R. Li, W. Tian, X. Li, H. Zhou, H. Chen, Y. Zhang, X. Yang, W. Jiang, K. Shen, J. Hong, Z. Yuan, L. Xi, K. Xia, S. Salahuddin, B. Dieny, and L. You, *Adv. Electron. Mater.* **5**, 1800782 (2019).
- 45 K. M. Song, J. S. Jeong, B. Pan, X. Zhang, J. Xia, S. Cha, T. E. Park, K. Kim, S. Finizio, J. Raabe, J. Chang, Y. Zhou, W. Zhao, W. Kang, H. Ju, and S. Woo, *Nat. Electron.* **3**, 148 (2020).
- 46 S. Zhang, and Y. Tserkovnyak, *Phys. Rev. Lett.* **125**, 207202 (2020), arXiv: [2003.11058](#).
- 47 S. Fukami, and H. Ohno, *J. Appl. Phys.* **124**, 151904 (2018).
- 48 A. Kurenkov, S. Fukami, and H. Ohno, *J. Appl. Phys.* **128**, 010902 (2020).
- 49 S. Lequeux, J. Sampaio, V. Cros, K. Yakushiji, A. Fukushima, R. Matsumoto, H. Kubota, S. Yuasa, and J. Grollier, *Sci. Rep.* **6**, 31510 (2016).
- 50 A. Sengupta, and K. Roy, *Phys. Rev. Appl.* **5**, 024012 (2016), arXiv: [1511.00051](#).
- 51 S. Boyn, J. Grollier, G. Lecerf, B. Xu, N. Locatelli, S. Fusil, S. Girod, C. Carrétéro, K. Garcia, S. Xavier, J. Tomas, L. Bellaiche, M. Bibes, A. Barthélémy, S. Saïghi, and V. Garcia, *Nat. Commun.* **8**, 14736 (2017).
- 52 M. K. Kim, and J. S. Lee, *Nano Lett.* **19**, 2044 (2019).
- 53 Y. Ren, B. Tian, M. Yan, G. Feng, B. Gao, F. Yue, H. Peng, X. Tang, Q. Zhu, J. Chu, and C. Duan, *Sci. China Inf. Sci.* **66**, 122403 (2023).
- 54 Y. Shen, N. C. Harris, S. Skirlo, M. Prabhu, T. Baehr-Jones, M. Hochberg, X. Sun, S. Zhao, H. Larochelle, D. Englund, and M. Soljačić, *Nat. Photon* **11**, 441 (2017), arXiv: [1610.02365](#).
- 55 Z. Cheng, C. Ríos, W. H. P. Pernice, C. D. Wright, and H. Bhaskaran, *Sci. Adv.* **3**, e1700160 (2017).
- 56 E. Goi, Q. Zhang, X. Chen, H. Luan, and M. Gu, *PhotonIX* **1**, 3 (2020).
- 57 A. Sebastian, M. Le Gallo, G. W. Burr, S. Kim, M. BrightSky, and E. Eleftheriou, *J. Appl. Phys.* **124**, 111101 (2018).
- 58 S. G. Sarwat, B. Kersting, T. Moraitis, V. P. Jonnalagadda, and A. Sebastian, *Nat. Nanotechnol.* **17**, 507 (2022).
- 59 Y. van de Burgt, E. Lubberman, E. J. Fuller, S. T. Keene, G. C. Faria, S. Agarwal, M. J. Marinella, A. Alec Talin, and A. Salleo, *Nat. Mater.* **16**, 414 (2017).
- 60 K. H. Kim, S. Gaba, D. Wheeler, J. M. Cruz-Albrecht, T. Hussain, N. Srinivasa, and W. Lu, *Nano Lett.* **12**, 389 (2012).
- 61 X. Zhang, A. Huang, Q. Hu, Z. Xiao, and P. K. Chu, *Phys. Status Solidi (a)* **215**, 1700875 (2018).
- 62 M. A. Zidan, J. P. Strachan, and W. D. Lu, *Nat. Electron.* **1**, 22 (2018).
- 63 Q. Xia, and J. J. Yang, *Nat. Mater.* **18**, 309 (2019).

- 64 W. Yu, J. Xiao, and G. E. W. Bauer, *Phys. Rev. B* **104**, L180405 (2021).
- 65 H. G. Manning, F. Niosi, C. G. da Rocha, A. T. Bellew, C. O'Callaghan, S. Biswas, P. F. Flowers, B. J. Wiley, J. D. Holmes, M. S. Ferreira, and J. J. Boland, *Nat. Commun.* **9**, 3219 (2018), arXiv: [1804.10117](#).
- 66 M. Romera, P. Talatchian, S. Tsunegi, F. Abreu Araujo, V. Cros, P. Bortolotti, J. Trastoy, K. Yakushiji, A. Fukushima, H. Kubota, S. Yuasa, M. Ernault, D. Vodenicarevic, T. Hirtzlin, N. Locatelli, D. Querlioz, and J. Grollier, *Nature* **563**, 230 (2018), arXiv: [1711.02704](#).
- 67 G. Csaba, and W. Porod, *Appl. Phys. Rev.* **7**, 011302 (2020).
- 68 C. Du, F. Cai, M. A. Zidan, W. Ma, S. H. Lee, and W. D. Lu, *Nat. Commun.* **8**, 2204 (2017).
- 69 Y. Zhong, J. Tang, X. Li, B. Gao, H. Qian, and H. Wu, *Nat. Commun.* **12**, 408 (2021).
- 70 Y. Zhong, J. Tang, X. Li, X. Liang, Z. Liu, Y. Li, Y. Xi, P. Yao, Z. Hao, B. Gao, H. Qian, and H. Wu, *Nat. Electron.* **5**, 672 (2022).
- 71 J. Torrejon, M. Riou, F. A. Araujo, S. Tsunegi, G. Khalsa, D. Querlioz, P. Bortolotti, V. Cros, K. Yakushiji, A. Fukushima, H. Kubota, S. Yuasa, M. D. Stiles, and J. Grollier, *Nature* **547**, 428 (2017).
- 72 Z. Chen, W. Li, Z. Fan, S. Dong, Y. Chen, M. Qin, M. Zeng, X. Lu, G. Zhou, X. Gao, and J. M. Liu, *Nat. Commun.* **14**, 3585 (2023).
- 73 Y. Tan, J. Zheng, X. Niu, Y. Zhao, N. Zhong, B. Tian, and C. Duan, *Sci. China-Phys. Mech. Astron.* **66**, 117505 (2023).
- 74 X. Wang, and F. Cichos, *Nat. Commun.* **15**, 774 (2024), arXiv: [2307.15010](#).
- 75 W. Hu, Z. Zhang, Y. Liao, Q. Li, Y. Shi, H. Zhang, X. Zhang, C. Niu, Y. Wu, W. Yu, X. Zhou, H. Guo, W. Wang, J. Xiao, L. Yin, Q. Liu, and J. Shen, *Nat. Commun.* **14**, 2562 (2023).
- 76 G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, *Neural Netw.* **115**, 100 (2019).
- 77 X. Liang, Y. Zhong, J. Tang, Z. Liu, P. Yao, K. Sun, Q. Zhang, B. Gao, H. Heidari, H. Qian, and H. Wu, *Nat. Commun.* **13**, 1549 (2022).
- 78 T. M. Heskes, and B. Kappen, *Phys. Rev. A* **44**, 2718 (1991).
- 79 T. M. Heskes, E. T. P. Slijpen, and B. Kappen, *Phys. Rev. A* **46**, 5221 (1992).
- 80 L. F. Abbott, and S. B. Nelson, *Nat. Neurosci.* **3**, 1178 (2000).
- 81 D. O. Hebb, *The Organization of Behavior: A Neuropsychological Theory* (Psychology Press, New York, 2005).
- 82 A. Momeni, B. Rahmani, M. Malléjac, P. del Hougne, and R. Fleury, *Science* **382**, 1297 (2023).
- 83 F. Alibart, E. Zamanidoost, and D. B. Strukov, *Nat. Commun.* **4**, 2072 (2013).
- 84 F. M. Bayat, M. Prezioso, B. Chakrabarti, H. Nili, I. Kataeva, and D. Strukov, *Nat. Commun.* **9**, 2331 (2018).
- 85 P. M. Sheridan, F. Cai, C. Du, W. Ma, Z. Zhang, and W. D. Lu, *Nat. Nanotechnol.* **12**, 784 (2017).
- 86 A. Hoffmann, S. Ramanathan, J. Grollier, A. D. Kent, M. J. Rozenberg, I. K. Schuller, O. G. Shpyrko, R. C. Dynes, Y. Fainman, A. Frano, E. E. Fullerton, G. Galli, V. Lomakin, S. P. Ong, A. K. Petford-Long, J. A. Schuller, M. D. Stiles, Y. Takamura, and Y. Zhu, *APL Mater.* **10**, 070904 (2022), arXiv: [2204.01832](#).
- 87 J. Tang, F. Yuan, X. Shen, Z. Wang, M. Rao, Y. He, Y. Sun, X. Li, W. Zhang, Y. Li, B. Gao, H. Qian, G. Bi, S. Song, J. J. Yang, and H. Wu, *Adv. Mater.* **31**, 1902761 (2019).
- 88 T. M. Heskes, and B. Kappen, in *On-line learning processes in artificial neural networks: North-Holland Mathematical Library Volume 51*, edited by J. G. Taylor (Elsevier, 1993), pp. 199-233.
- 89 N. Murata, M. Kawanabe, A. Ziehe, K. R. Müller, and S. Amari, *Neural Netw.* **15**, 743 (2002).
- 90 S. Amari, *IEEE Trans. Electron. Comput.* **EC-16**, 299 (1967).
- 91 N. Barkai, H. S. Seung, and H. Sompolinsky, *Phys. Rev. Lett.* **75**, 1415 (1995).
- 92 Q. Zhang, H. Wu, P. Yao, W. Zhang, B. Gao, N. Deng, and H. Qian, *Neural Netw.* **108**, 217 (2018).
- 93 W. Zhang, P. Yao, B. Gao, Q. Liu, D. Wu, Q. Zhang, Y. Li, Q. Qin, J. Li, Z. Zhu, Y. Cai, D. Wu, J. Tang, H. Qian, Y. Wang, and H. Wu, *Science* **381**, 1205 (2023).
- 94 M. Prezioso, F. Merrih-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev, and D. B. Strukov, *Nature* **521**, 61 (2015).
- 95 C. Li, D. Belkin, Y. Li, P. Yan, M. Hu, N. Ge, H. Jiang, E. Montgomery, P. Lin, Z. Wang, W. Song, J. P. Strachan, M. Barnell, Q. Wu, R. S. Williams, J. J. Yang, and Q. Xia, *Nat. Commun.* **9**, 2385 (2018).
- 96 Z. Wang, C. Li, P. Lin, M. Rao, Y. Nie, W. Song, Q. Qiu, Y. Li, P. Yan, J. P. Strachan, N. Ge, N. McDonald, Q. Wu, M. Hu, H. Wu, R. S. Williams, Q. Xia, and J. J. Yang, *Nat. Mach. Intell.* **1**, 434 (2019).
- 97 R. Wang, T. Shi, X. Zhang, J. Wei, J. Lu, J. Zhu, Z. Wu, Q. Liu, and M. Liu, *Nat. Commun.* **13**, 2289 (2022).
- 98 S. Choi, J. H. Shin, J. Lee, P. Sheridan, and W. D. Lu, *Nano Lett.* **17**, 3113 (2017).
- 99 Y. J. Jeong, J. Lee, J. Moon, J. H. Shin, and W. D. Lu, *Nano Lett.* **18**, 4447 (2018).
- 100 W. Wang, L. Danial, Y. Li, E. Herbelin, E. Pikhay, Y. Roizin, B. Hoffer, Z. Wang, and S. Kvatsinsky, *Nat. Electron.* **5**, 870 (2022).
- 101 Z. Wang, S. Joshi, S. Savel'ev, W. Song, R. Midya, Y. Li, M. Rao, P. Yan, S. Asapu, Y. Zhuo, H. Jiang, P. Lin, C. Li, J. H. Yoon, N. K. Upadhyay, J. Zhang, M. Hu, J. P. Strachan, M. Barnell, Q. Wu, H. Wu, R. S. Williams, Q. Xia, and J. J. Yang, *Nat. Electron.* **1**, 137 (2018).
- 102 G. Bellec, F. Scherr, A. Subramoney, E. Hajek, D. Salaj, R. Legenstein, and W. Maass, *Nat. Commun.* **11**, 3625 (2020).
- 103 X. Wang, X. Lin, J. Zhao, and H. Ma, in *Supervised learning algorithm for spiking neurons based on nonlinear inner products of spike trains: Intelligent Computing Theories and Application*, edited by D.-S. Huang, and K.-H. Jo (Springer, Cham, 2016), pp. 95-104.
- 104 S. K. Bose, C. P. Lawrence, Z. Liu, K. S. Makarenko, R. M. J. van Damme, H. J. Broersma, and W. G. van der Wiel, *Nat. Nanotechnol.* **10**, 1048 (2015).
- 105 H. C. Ruiz Euler, M. N. Boon, J. T. Wildeboer, B. van de Ven, T. Chen, H. Broersma, P. A. Bobbert, and W. G. van der Wiel, *Nat. Nanotechnol.* **15**, 992 (2020).
- 106 J. Feldmann, N. Youngblood, C. D. Wright, H. Bhaskaran, and W. H. P. Pernice, *Nature* **569**, 208 (2019).
- 107 V. López-Pastor, and F. Marquardt, *Phys. Rev. X* **13**, 031020 (2023), arXiv: [2103.04992](#).
- 108 S. M. Buckley, A. N. Tait, A. N. McCaughan, and B. J. Shastri, *Nanophotonics* **12**, 833 (2023).
- 109 M. Stern, and A. Murugan, *Annu. Rev. Condens. Matter Phys.* **14**, 417 (2023), arXiv: [2206.05831](#).
- 110 J. Miller, and K. Downing, in *Evolution in materio: Looking beyond the silicon box: Proceedings of the 2002 NASA/DoD Conference on Evolvable Hardware* (IEEE, Washington, 2002), pp. 167-176.
- 111 T. Kanao, H. Suto, K. Mizushima, H. Goto, T. Tanamoto, and T. Nagasawa, *Phys. Rev. Appl.* **12**, 024052 (2019), arXiv: [1905.07937](#).
- 112 K. Bhattacharyya, D. Zwicker, and K. Alim, *Phys. Rev. Lett.* **129**, 028101 (2022).
- 113 M. Stern, D. Hexner, J. W. Rocks, and A. J. Liu, *Phys. Rev. X* **11**, 021045 (2021), arXiv: [2011.03861](#).
- 114 S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, *J. Sci. Comput.* **92**, 88 (2022).
- 115 S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, *Acta Mech. Sin.* **37**, 1727 (2021), arXiv: [2105.09506](#).
- 116 G. P. P. Pun, R. Batra, R. Ramprasad, and Y. Mishin, *Nat. Commun.* **10**, 2339 (2019).
- 117 A. Norambuena, M. Mattheakis, F. J. González, and R. Coto, *Phys.*

- Rev. Lett. **132**, 010801 (2024), arXiv: [2206.06287](#).
- 118 H. Tanaka, M. Akai-Kasaya, A. Termehyousefi, L. Hong, L. Fu, H. Tamukoh, D. Tanaka, T. Asai, and T. Ogawa, *Nat. Commun.* **9**, 2693 (2018).
 - 119 A. Diaz-Alvarez, R. Higuchi, Q. Li, Y. Shingaya, and T. Nakayama, *AIP Adv.* **10**, 025134 (2020).
 - 120 A. Diaz-Alvarez, R. Higuchi, P. Sanz-Leon, I. Marcus, Y. Shingaya, A. Z. Stieg, J. K. Gimzewski, Z. Kuncic, and T. Nakayama, *Sci. Rep.* **9**, 14920 (2019).
 - 121 L. Xu, J. Zhu, B. Chen, Z. Yang, K. Liu, B. Dang, T. Zhang, Y. Yang, and R. Huang, *Nat. Commun.* **13**, 4698 (2022).
 - 122 M. Stern, M. B. Pinson, and A. Murugan, *Phys. Rev. X* **10**, 031044 (2020).
 - 123 Y. Jeong, H. Y. Cho, M. Kim, J. P. Oh, M. S. Kang, M. Yoo, H. S. Lee, and J. H. Han, *Nat. Commun.* **12**, 3915 (2021).
 - 124 A. Tero, S. Takagi, T. Saigusa, K. Ito, D. P. Bebber, M. D. Fricker, K. Yumiki, R. Kobayashi, and T. Nakagaki, *Science* **327**, 439 (2010).
 - 125 M. Kramar, and K. Alim, *Proc. Natl. Acad. Sci. USA* **118**, e2007815118 (2021).
 - 126 N. C. Keim, J. D. Paulsen, Z. Zeravcic, S. Sastry, and S. R. Nagel, *Rev. Mod. Phys.* **91**, 035002 (2019), arXiv: [1810.08587](#).
 - 127 P. Huembeli, J. M. Arrazola, N. Killoran, M. Mohseni, and P. Wittek, *Quantum Mach. Intell.* **4**, 1 (2022).
 - 128 J. J. Hopfield, *Proc. Natl. Acad. Sci. USA* **79**, 2554 (1982).
 - 129 D. J. Amit, H. Gutfreund, and H. Sompolinsky, *Phys. Rev. A* **32**, 1007 (1985).
 - 130 B. Hoover, H. Strobelt, D. Krotov, J. Hoffman, Z. Kira, and D. H. Chau, arXiv: [2309.16750](#).
 - 131 S. G. Hu, Y. Liu, Z. Liu, T. P. Chen, J. J. Wang, Q. Yu, L. J. Deng, Y. Yin, and S. Hosaka, *Nat. Commun.* **6**, 7522 (2015).
 - 132 S. Fukami, C. Zhang, S. Duttagupta, A. Kurenkov, and H. Ohno, *Nat. Mater.* **15**, 535 (2016), arXiv: [1507.00888](#).
 - 133 W. A. Borders, H. Akima, S. Fukami, S. Moriya, S. Kurihara, Y. Horio, S. Sato, and H. Ohno, *Appl. Phys. Express* **10**, 013007 (2017).
 - 134 E. Fiorelli, M. Marcuzzi, P. Rotondo, F. Carollo, and I. Lesanovsky, *Phys. Rev. Lett.* **125**, 070604 (2020), arXiv: [2003.01004](#).
 - 135 F. Cai, S. Kumar, T. Van Vaerenbergh, X. Sheng, R. Liu, C. Li, Z. Liu, M. Foltin, S. Yu, Q. Xia, J. J. Yang, R. Beausoleil, W. D. Lu, and J. P. Strachan, *Nat. Electron.* **3**, 409 (2020).
 - 136 M. Saccone, F. Caravelli, K. Hofhuis, S. Parchenko, Y. A. Birkhölzer, S. Dhuey, A. Kleibert, S. van Dijken, C. Nisoli, and A. Farhan, *Nat. Phys.* **18**, 517 (2022), arXiv: [2202.02372](#).
 - 137 O. H. Schnack, L. Peliti, and A. Nourmohammad, *Phys. Rev. X* **12**, 021063 (2022), arXiv: [2106.02186](#).
 - 138 P. Baldi, *Phys. Rev. Lett.* **59**, 1976 (1987).
 - 139 P. Tolmachev, and J. H. Manton, in *New insights on learning rules for hopfield networks: Memory and objective function minimisation: Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN)* (Glasgow, IEEE, 2020), pp. 1-8.
 - 140 E. Oja, *J. Math. Biol.* **15**, 267 (1982).
 - 141 D. Krotov, and J. J. Hopfield, *Proc. Natl. Acad. Sci. USA* **116**, 7723 (2019), arXiv: [1806.10181](#).
 - 142 K. D. Miller, and D. J. C. MacKay, *Neural Comput.* **6**, 100 (1994).
 - 143 D. J. Amit, H. Gutfreund, and H. Sompolinsky, *Phys. Rev. Lett.* **55**, 1530 (1985).
 - 144 R. McEliece, E. Posner, E. Rodemich, and S. Venkatesh, *IEEE Trans. Inform. Theor.* **33**, 461 (1987).
 - 145 S. Fusi, arXiv: [2108.07839](#).
 - 146 F. J. Pineda, *Phys. Rev. Lett.* **59**, 2229 (1987).
 - 147 M. Kumar, S. I. Han, S. W. Lim, and H. Seo, *ACS Appl. Electron. Mater.* **5**, 896 (2023).
 - 148 A. S. Koshikawa, J. Llandro, M. Ohzeki, S. Fukami, H. Ohno, and N. Leo, *Phys. Rev. B* **108**, 024414 (2023), arXiv: [2303.05816](#).
 - 149 E. H. L. Aarts, and J. H. M. Korst, in *Boltzmann machines and their applications: PARLE Parallel Architectures and Languages Europe*, edited by J. W. de Bakker, A. J. Nijman, and P. C. Treleaven (Springer, Berlin, Heidelberg 1987), pp. 34-50.
 - 150 Z. Fahimi, M. R. Mahmoodi, H. Nili, V. Polishchuk, and D. B. Strukov, *Sci. Rep.* **11**, 16383 (2021).
 - 151 J. Fernandez-de-Cossio-Diaz, S. Cocco, and R. Monasson, *Phys. Rev. X* **13**, 021003 (2023), arXiv: [2206.11600](#).
 - 152 D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, *Cogn. Sci.* **9**, 147 (1985).
 - 153 J. J. Hopfield, and D. W. Tank, *Biol. Cybern.* **52**, 141 (1985).
 - 154 H. Goto, K. Tatsumura, and A. R. Dixon, *Sci. Adv.* **5**, eaav2372 (2019).
 - 155 S. Patel, P. Canoza, and S. Salahuddin, *Nat. Electron.* **5**, 92 (2022).
 - 156 J. Kaiser, R. Faria, K. Y. Camsari, and S. Datta, *Front. Comput. Neurosci.* **14**, 14 (2020).
 - 157 B. Kiraly, E. J. Knol, W. M. J. van Weerdenburg, H. J. Kappen, and A. A. Khajetoorians, *Nat. Nanotechnol.* **16**, 414 (2021), arXiv: [2005.01547](#).
 - 158 D. A. Nguyen, X. T. Tran, and F. Iacopi, *J. Low Power Electron. Appl.* **11**, 23 (2021).
 - 159 J. Han, S. Yun, S. Lee, J. Yu, and Y. Choi, *Adv. Funct. Mater.* **32**, 2204102 (2022).
 - 160 K. Yamazaki, V. K. Vo-Ho, D. Bulsara, and N. Le, *Brain Sci.* **12**, 863 (2022).
 - 161 P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, *Science* **345**, 668 (2014).
 - 162 S. Song, K. D. Miller, and L. F. Abbott, *Nat. Neurosci.* **3**, 919 (2000).
 - 163 M. Yan, Q. Zhu, S. Wang, Y. Ren, G. Feng, L. Liu, H. Peng, Y. He, J. Wang, P. Zhou, X. Meng, X. Tang, J. Chu, B. Dkhil, B. Tian, and C. Duan, *Adv. Electron. Mater.* **7**, 2001276 (2021).
 - 164 W. A. Borders, A. Z. Pervaiz, S. Fukami, K. Y. Camsari, H. Ohno, and S. Datta, *Nature* **573**, 390 (2019).
 - 165 H. Li, L. Li, K. Zhou, C. Yan, Z. Gao, Z. Li, and R. Liu, *Sci. China-Phys. Mech. Astron.* **66**, 257512 (2023), arXiv: [2304.08856](#).
 - 166 X. Zhang, Y. Zhuo, Q. Luo, Z. Wu, R. Midya, Z. Wang, W. Song, R. Wang, N. K. Upadhyay, Y. Fang, F. Kiani, M. Rao, Y. Yang, Q. Xia, Q. Liu, M. Liu, and J. J. Yang, *Nat. Commun.* **11**, 51 (2020).
 - 167 Z. Zhao, L. Qu, L. Wang, Q. Deng, N. Li, Z. Kang, S. Guo, and W. Xu, *IEEE Trans. Circuits Syst. II* **67**, 931 (2020).
 - 168 X. Zhang, J. Lu, Z. Wang, R. Wang, J. Wei, T. Shi, C. Dou, Z. Wu, J. Zhu, D. Shang, G. Xing, M. Chan, Q. Liu, and M. Liu, *Sci. Bull.* **66**, 1624 (2021).
 - 169 L. F. Abbott, B. DePasquale, and R. M. Memmesheimer, *Nat. Neurosci.* **19**, 350 (2016).
 - 170 S. Yi, J. D. Kendall, R. S. Williams, and S. Kumar, *Nat. Electron.* **6**, 45 (2022).
 - 171 B. Gao, Y. Zhou, Q. Zhang, S. Zhang, P. Yao, Y. Xi, Q. Liu, M. Zhao, W. Zhang, Z. Liu, X. Li, J. Tang, H. Qian, and H. Wu, *Nat. Commun.* **13**, 2026 (2022).
 - 172 H. Nomura, H. Kubota, and Y. Suzuki, in *Reservoir computing with dipole-coupled nanomagnets: Reservoir Computing: Theory, Physical Implementations, and Applications, Natural Computing Series*, edited by K. Nakajima, and I. Fischer (Springer, Singapore, 2021), pp. 361-374.
 - 173 A. Papp, G. Csaba, and W. Porod, *Appl. Phys. Lett.* **119**, 112403 (2021).
 - 174 D. Pinna, G. Bourianoff, and K. Everschor-Sitte, *Phys. Rev. Appl.* **14**, 054020 (2020).
 - 175 R. Nakane, G. Tanaka, and A. Hirose, *IEEE Access* **6**, 4462 (2018).

- 176 I. T. Vidamour, C. Swindells, G. Venkat, L. Manneschi, P. W. Fry, A. Welbourne, R. M. Rowan-Robinson, D. Backes, F. Maccherozzi, S. S. Dhesi, E. Vasilaki, D. A. Allwood, and T. J. Hayward, *Commun. Phys.* **6**, 230 (2023), arXiv: [2206.04446](#).
- 177 M. K. Lee, and M. Mochizuki, *Phys. Rev. Appl.* **18**, 014074 (2022), arXiv: [2203.02160](#).
- 178 S. Pai, Z. Sun, T. W. Hughes, T. Park, B. Bartlett, I. A. D. Williamson, M. Minkov, M. Milanizadeh, N. Abebe, F. Morichetti, A. Melloni, S. Fan, O. Solgaard, and D. A. B. Miller, *Science* **380**, 398 (2023), arXiv: [2205.08501](#).
- 179 J. Wang, S. P. Rodrigues, E. M. Dede, and S. Fan, *Opt. Express* **31**, 18871 (2023).
- 180 X. Guo, T. D. Barrett, Z. M. Wang, and A. I. Lvovsky, *Photon. Res.* **9**, B71 (2021).
- 181 K. Wagner, and D. Psaltis, *Appl. Opt.* **26**, 5061 (1987).
- 182 J. J. Hopfield, D. I. Feinstein, and R. G. Palmer, *Nature* **304**, 158 (1983).
- 183 S. Grossberg, *J. Stat. Phys.* **1**, 319 (1969).
- 184 J. R. Movellan, in *Contrastive Hebbian learning in the continuous Hopfield model: Connectionist Models*, edited by D. S. Touretzky, J. L. Elman, T. J. Sejnowski, and G. E. Hinton (Morgan, Kaufmann, 1991), pp. 10-17.
- 185 C. C. Galland, and G. E. Hinton, in *Deterministic Boltzmann Learning in networks with asymmetric connectivity: Connectionist Models*, edited by D. S. Touretzky, J. L. Elman, T. J. Sejnowski, and G. E. Hinton (Morgan, Kaufmann, 1991), pp. 3-9.
- 186 T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman, and G. Hinton, *Nat. Rev. Neurosci.* **21**, 335 (2020).
- 187 M. Nakajima, K. Inoue, K. Tanaka, Y. Kuniyoshi, T. Hashimoto, and K. Nakajima, *Nat. Commun.* **13**, 7847 (2022).
- 188 J. Hertz, A. Krogh, B. Lautrup, and T. Lehmann, *IEEE Trans. Neural Networks* **8**, 1321 (1997).
- 189 B. Scellier, and Y. Bengio, *Front. Comput. Neurosci.* **11**, 24 (2017).
- 190 J. Kendall, R. Pantone, K. Manickavasagam, Y. Bengio, and B. Scellier, arXiv: [2006.01981](#).
- 191 A. Laborieux, M. Ernout, B. Scellier, Y. Bengio, J. Grollier, and D. Querlioz, arXiv: [2006.03824](#).
- 192 J. Laydevant, D. Markovic, and J. Grollier, arXiv: [2305.18321](#).
- 193 G. E. Hinton, *Neural Comput.* **14**, 1771 (2002).
- 194 J. F. Wycoff, S. Dillavou, M. Stern, A. J. Liu, and D. J. Durian, *J. Chem. Phys.* **156**, 144903 (2022), arXiv: [2201.04626](#).
- 195 M. Stern, S. Dillavou, M. Z. Miskin, D. J. Durian, and A. J. Liu, *Phys. Rev. Res.* **4**, L022037 (2022).
- 196 S. Dillavou, M. Stern, A. J. Liu, and D. J. Durian, *Phys. Rev. Appl.* **18**, 014040 (2022), arXiv: [2108.00275](#).
- 197 V. R. Anisetti, B. Scellier, and J. M. Schwarz, *Phys. Rev. Res.* **5**, 023024 (2023).
- 198 S. R. Skinner, A. A. Cruz-Cabrera, E. C. Behrman, and J. E. Steck, *Appl. Opt.* **34**, 4129 (1995).
- 199 T. W. Hughes, M. Minkov, Y. Shi, and S. Fan, *Optica* **5**, 864 (2018), arXiv: [1805.09943](#).
- 200 A. Blais, R. S. Huang, A. Wallraff, S. M. Girvin, and R. J. Schoelkopf, *Phys. Rev. A* **69**, 062320 (2004), arXiv: [cond-mat/0402216](#).
- 201 A. Wallraff, D. I. Schuster, A. Blais, L. Frunzio, R. S. Huang, J. Majer, S. Kumar, S. M. Girvin, and R. J. Schoelkopf, *Nature* **431**, 162 (2004), arXiv: [cond-mat/0407325](#).
- 202 J. Majer, J. M. Chow, J. M. Gambetta, J. Koch, B. R. Johnson, J. A. Schreier, L. Frunzio, D. I. Schuster, A. A. Houck, A. Wallraff, A. Blais, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf, *Nature* **449**, 443 (2007), arXiv: [0709.2135](#).
- 203 V. W. Scarola, L. Pollet, J. Oitmaa, and M. Troyer, *Phys. Rev. Lett.* **102**, 135302 (2009), arXiv: [0809.3239](#).
- 204 J. Lan, W. Yu, R. Wu, and J. Xiao, *Phys. Rev. X* **5**, 041049 (2015), arXiv: [1509.05095](#).
- 205 H. Yu, J. Xiao, and H. Schultheiss, *Phys. Rep.* **905**, 1 (2021), arXiv: [2010.09180](#).
- 206 A. V. Chumak, V. I. Vasyuchka, A. A. Serga, and B. Hillebrands, *Nat. Phys.* **11**, 453 (2015).
- 207 M. A. Zidan, Y. J. Jeong, J. Lee, B. Chen, S. Huang, M. J. Kushner, and W. D. Lu, *Nat. Electron.* **1**, 411 (2018).
- 208 Z. Sun, G. Pedretti, E. Ambrosi, A. Bricalli, W. Wang, and D. Ielmini, *Proc. Natl. Acad. Sci. USA* **116**, 4123 (2019).
- 209 A. Ross, N. Leroux, A. De Riz, D. Marković, D. Sanz-Hernández, J. Trastoy, P. Bortolotti, D. Querlioz, L. Martins, L. Benetti, M. S. Claro, P. Anacleto, A. Schulman, T. Taris, J. B. Begueret, S. Saïghi, A. S. Jenkins, R. Ferreira, A. F. Vincent, F. A. Mizrahi, and J. Grollier, *Nat. Nanotechnol.* **18**, 1273 (2023), arXiv: [2211.03659](#).
- 210 L. O. Chua, and L. Yang, *IEEE Trans. Circuits Syst.* **35**, 1257 (1988).
- 211 T. Kohonen, *Biol. Cybern.* **43**, 59 (1982).
- 212 Q. Zheng, Y. Mi, X. Zhu, Z. Yuan, and K. Xia, *Phys. Rev. Appl.* **14**, 044060 (2020), arXiv: [2005.02574](#).
- 213 F. Zhou, and Y. Chai, *Nat. Electron.* **3**, 664 (2020).
- 214 G. Feng, X. Zhang, B. Tian, and C. Duan, *InfoMat* **5**, e12473 (2023).
- 215 F. Liao, Z. Zhou, B. J. Kim, J. Chen, J. Wang, T. Wan, Y. Zhou, A. T. Hoang, C. Wang, J. Kang, J. H. Ahn, and Y. Chai, *Nat. Electron.* **5**, 84 (2022).
- 216 D. Marković, and J. Grollier, *Appl. Phys. Lett.* **117**, 150501 (2020), arXiv: [2006.15111](#).
- 217 A. Labay-Mora, R. Zambrini, and G. L. Giorgi, *Phys. Rev. Lett.* **130**, 190602 (2023), arXiv: [2205.09491](#).
- 218 X. Pan, Y. Li, B. Cheng, S. J. Liang, and F. Miao, *Sci. China-Phys. Mech. Astron.* **66**, 117504 (2023).
- 219 M. A. Mahowald, and C. Mead, *Sci. Am.* **264**, 76 (1991).
- 220 C. Kaspar, B. J. Ravoo, W. G. van der Wiel, S. V. Wegner, and W. H. P. Pernice, *Nature* **594**, 345 (2021).
- 221 S. K. Vadlamani, T. P. Xiao, and E. Yablonovitch, *Proc. Natl. Acad. Sci. USA* **117**, 26639 (2020).
- 222 A. M. de Callataÿ, *Natural and Artificial Intelligence: Misconceptions about Brains and Neural Networks* (Elsevier, Amsterdam, 1992).
- 223 M. van Gerven, *Front. Comput. Neurosci.* **11**, 112 (2017).