Applied Physics A (2023) 129:236
https://doi.org/10.1007/s00339-022-06365-4

Applied Physics A
Materials Science & Processing

S.I. : 50TH ANNIVERSARY OF APPLIED PHYSICS

Unconventional computing based on magnetic tunnel junction

Baofang Cai[1] · Yihan He[1] · Yue Xin[2] · Zhengping Yuan[2] · Xue Zhang[2] · Zhifeng Zhu[2,3] · Gengchiau Liang[1]

Received: 2 November 2022 / Accepted: 24 December 2022 / Published online: 3 March 2023

Abstract
The conventional computing method based on the von Neumann architecture is limited by a series of problems such as high energy consumption, finite data exchange bandwidth between processors and storage media, etc., and it is difficult to achieve higher computing efficiency. A more efficient unconventional computing architecture is urgently needed to overcome these problems. Neuromorphic computing and stochastic computing have been considered to be two competitive candidates for unconventional computing, due to their extraordinary potential for energy-efficient and high-performance computing. Although conventional electronic devices can mimic the topology of the human brain, these require high power consumption and large area. Spintronic devices represented by magnetic tunnel junctions (MTJs) exhibit remarkable high-energy efficiency, non-volatility, and similarity to biological nervous systems, making them one of the promising candidates for unconventional computing. In this work, we review the fundamentals of MTJs as well as the development of MTJ-based neurons, synapses, and probabilistic-bit. In the section on neuromorphic computing, we review a variety of neural networks composed of MTJ-based neurons and synapses, including multilayer perceptrons, convolutional neural networks, recurrent neural networks, and spiking neural networks, which are the closest to the biological neural system. In the section on stochastic computing, we review the applications of MTJ-based p-bits, including Boltzmann machines, Ising machines, and Bayesian networks. Furthermore, the challenges to developing these novel technologies are briefly discussed at the end of each section.

Keywords Magnetic tunnel junction · Neuromorphic computing · Unconventional computing · Spintronic neuron · Spintronic synapse · Stochastic switching

1 Introduction

Modern computers, based on von Neumann architecture that solves numerical problems in a serial, deterministic, and highly precise way, have been extensively developed for decades and are still the mainstream of the fashion for information processing at present. However, the emergence of big data with increasing volume and complexity challenged the von Neumann computing paradigm, in that shuttling such information between the processor and the storage inevitably causes substantial energy consumption. Therefore, in the context of seeking a solution for the "von Neumann bottleneck" [1], novel computing paradigms beyond von Neumann architecture, i.e., unconventional computing, are desired. Contrary to the von Neumann paradigm that gives out guaranteed and accurate results, approximate computing [2] employs redundant computation and returns approximate results that are sufficient for their objectives such as recognition, classification, prediction, optimization, and so on. Such emerging paradigms are expected to achieve high performance and energy-efficient computing when involving big data processing, in that 1) the approximate computing uses many low-precision or probabilistic calculations, and thus is inherently resilient to errors, 2) most paradigms for approximate computing are in parallel that would benefit for calculation speed, and 3) some of the approximate

Baofang Cai, Yihan He and Yue Xin have contributed equally and listed alphabetically by their last names.

✉ Zhifeng Zhu
zhuzhf@shanghaitech.edu.cn

✉ Gengchiau Liang
elelg@nus.edu.sg

[1] Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117576, Singapore

[2] School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China

[3] Shanghai Engineering Research Center of Energy Efficient and Custom AI IC, Shanghai 201210, China

computing associated paradigms are designed for storing information locally where it is processed so that extricating from the large energy dissipation caused by commuting data between processor and memory.

Among approximate computing, bio-inspired computing has recently attracted much interest due to its massive parallelism, high energy efficiency, adaptivity to varying and complex inputs, and inherent tolerance to fault and variation. Therefore, bio-inspired computing is especially useful for unstructured data processing such as recognition, one of the purposes of machine learning. The direct strategy for bio-inspired computing design is emulating the human brain, namely Neuromorphic computing [3]. Neuromorphic computing is based on a variety of artificial neural networks (ANNs) which are composed of the following two elementary units: artificial neurons and synapses. Synapses function as connectors with different variable weights (i.e., connection strength) to update and deliver information. While neurons that are interconnected by synapses receive signals from other neurons and emit spikes to the subsequent neurons if activated. Choosing a proper neural network for a particular computing task is one of the key issues associated with neuromorphic computing. Specifically, the rudimentary classification tasks that require differentiating binary states can be handled by the first generation of ANN [4], also called "perceptron". A perceptron is the simplest ANN that constitutes one layer of neurons for inputs and another layer of neurons for outputs and the two layers are connected by synapses. As shown in Fig. 1(a), however, the single perceptron can only solve classification problems in a linear way and thus function analogously to AND, NAND, and OR gate. Therefore, one of the second-generation ANNs, the multi-layer perceptron (MLP) [5] or deep neural network (DNN) [6] in which hidden layers play an important role has been proposed for solving the nonlinear classification problems which are analogous to XOR gate shown in Fig. 1(b). In DNN, neurons in one layer are fully connected by neurons in the neighboring layer and information would be delivered unidirectionally. Additionally, by adapting the mode that how neurons are connected and interacted, other multi-layered neural networks have been proposed to perform computation tasks with upgraded efficiency. For example, the convolutional neural network (CNN) [7] would be advantageous for image recognition due to the cooperation among the convolutional layer, max pooling layer, and fully connected layer that consist of the CNN. The data processed in the DNN and the CNN are time-independent or static, while recurrent neural network (RNN) [8] concerns processing sequence data. RNN not only allows for cycles that could achieve related data among adjacent time steps, but also has differing levels of connectivity; therefore, RNN is very desirable for sound recognition, natural language processing, computer vision, etc.

Compared to biological neural networks, the first- and second-generation ANNs are much more computationally driven and would be in the category of non-spiking neural networks (non-SNN), whereas in recent years, researchers started to design an ANN that could replicate biological behavior closely in that biological neural systems would inherently process information with high efficiency. Consequently, the third-generation ANN, the spiking neural networks (SNN) [9], would be more biomimetically driven and has attracted much attention. In additional to the potential for saving energy, SNN offers a platform for realizing spike-timing-dependent plasticity (STDP) [10], one of the most efficient unsupervised learning algorithms and is capable of training data online. It is worth noting that the capabilities of the spiking neuromorphic system have not been realized by training and learning mechanisms comprehensively and the superiority of computing performance for the SNN and non-SNN is still under debate.

On the other hand, stochastic computing exploits randomness, and its physics rules are also competitive for solving problems that neuromorphic computing concerns. The unit of stochastic computing is a random number generator with a tunable output probability, which is called a probabilistic-pit (p-bit) [11]. For machine learning, the Boltzmann machine (BM) [12] is widely used as the architecture of

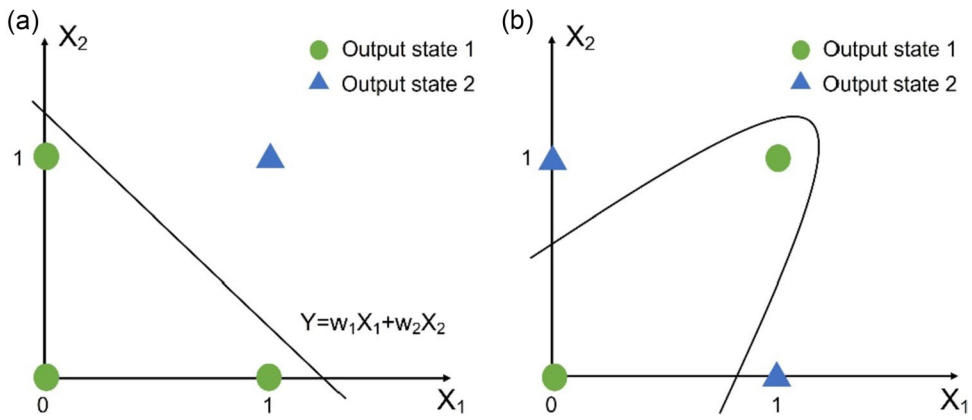

Fig. 1 **a** The classification problem can be solved by finding a straight line whose function is $Y = w_1X_1 + w_2X_2$, where w_i is weight, X_i is input ($i = 1, 2$) and Y is output. The logic is analogous to AND, NAND, and OR gate. **b** The classification problem needs to be solved by finding a nonlinear line in that the two output states cannot be separated by any straight line. The logic is analogous to the XOR gate

stochastic computing, which is also called "stochastic neural networks". A more commonly utilized Boltzmann model is the restricted Boltzmann machine [13]. Compared to a general BM, the restricted Boltzmann machine could speed up the training rate due to the restricted connection among the units. Another typical case of the stochastic neural networks is related to Bayesian calculations, i.e., the Bayesian network (BN) [14]. BN is a feed-forward neural network, and the nodes could be divided into parent and child nodes in terms of their causal sequences inherited from events, expecting parent-to-child directionality for the data delivery. Furthermore, stochastic computing is also capable of solving the computation tasks that adiabatic quantum computing concerns [15]. For example, the Ising model has been applied to solve combinatorial optimization problems (COP) such as the travel salesman problem (TSP) [16], while inverse problems such as integer factorization (IF) [17], which is very difficult for conventional computing, can be worked out by stochastic networks. Figure 2 summarizes the relationship among the aforementioned computing paradigms that will be discussed in more detail in this review.

The hardware unit implementation for such unconventional computing paradigms, furthermore, was initially supported by typically hundreds to thousands of transistors which would be undesirable for unconventional computing tasks because of the energy and area requirements. In recent years, it has been proposed that a single spintronic device would be capable of emulating the behavior of synapse, neuron, and p-bit in that such devices could be engineered to a variety of properties such as non-volatility, plasticity, stochasticity, and oscillation, which are key features of the computing units [18]. Magnetic tunnel junctions (MTJs), a typical structure of spintronic devices, have been investigated for not only information storage but also unconventional computing. Due to their versatile properties, together with the outstanding endurance, and CMOS-technology compatibility, MTJs are promising candidates for the hardware of unconventional computing with high performance. Moreover, the MTJs would be expected in different behaviors to cooperate with a particular computing paradigm, and this will also be discussed in more detail in this review.

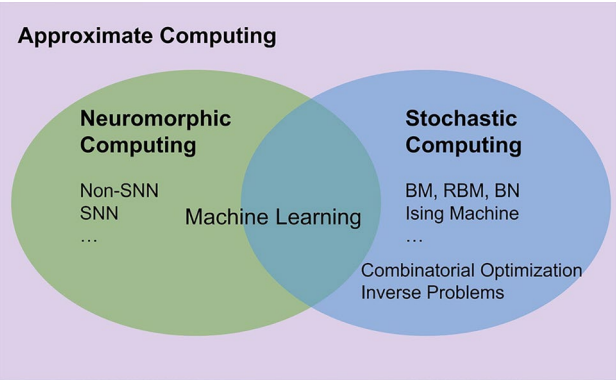

Fig. 2 The relationship among the unconventional computing paradigms

The structure of this review is conducted as follows: in Sect. 2, we discuss the hardware based on MTJ for unconventional computing on the device level, illustrating the mechanism for the operation of MTJ-based devices, the device features, design principles, and recent works. Then we divided the discussion of unconventional computing on the architecture level into neuromorphic computing (Sect. 3) and stochastic computing (Sect. 4), giving an overview of the aspects of computing tasks and applications to which these unconventional computing systems have been investigated. Finally, we enumerated some challenges that need to be tackled and concluded with promising perspectives for unconventional computing.

2 MTJ-based devices for unconventional computing: from mechanism to applications

2.1 Mechanism

For magnetic materials, magnetic orders stem from the neighboring localized, exchange-coupled electron spins. From the perspective of classical physics, magnetic orders are regarded as the magnetic moment which is controlled by the spin angular momentum. Due to the controllable magnetic orders, the device based on magnetic materials could achieve information storage, logic computation, and other novel functionalities. At an early stage, the magnetic order is controlled by a magnetic field, and then, in order to be compatible with circuits, people started to manipulate magnetization in electrical ways. Especially, the spin-transfer torque (STT) effect [19] plays an important role in electrically changing magnetic orders. The spin-polarized conduction electrons can change the magnetic moment by exchanging the spin angular momenta between the conduction electrons and spin electrons. More specifically, the STT effect can be experimentally achieved in a structure of a non-magnetic spacer sandwiched by two magnetic layers with large/small saturation magnetization called pinned/free layer, respectively. When the spin-polarized electrons which are filtered by pinned layer reached the free layer, the spin-polarization component that is parallel to the magnetization of the free layer can be transmitted, while the component which is perpendicular to the magnetization would be absorbed and thus led to the rotation of the magnetization due to angular momentum conservation.

Although STT is difficult to control the magnetization of magnetic materials with high resistance, the magnetization

could also be rotated by spin-orbital torque (SOT) [20] which does not require electrons to pass through the magnets. The SOT effect originated from spin-orbital coupling (SOC) [21], and the principle of SOC can be attributed to an effective magnetic field generated by an electrical field. A material with broken inversion symmetry could produce a net spin polarization due to the asymmetric spin scatterings in the bulk, and this is the spin-Hall effect (SHE) [22]. Then, the spin-polarized electrons would accumulate at the interface of the material and thus can be absorbed by an adjacent ferromagnet in the form of damping-like SOT. Another physical explanation is attributed to the Rashba effect [23]; electrons pass through an interface with the asymmetrical inversion and therefore obtain a spin polarization. The polarized electrons can generate a torque on the adjacent ferromagnet via the exchange coupling. Although the SHE and Rashba effects are dominant in conventional ferromagnet/heavy metal heterostructures, these two are not the only origin of SOT. The other effects, such as the quantum spin-Hall effect [24] in topological insulators, which could also generate SOT, are still under intensive investigation.

In addition to applying the spin torques to manipulate magnetizations, the magnetic anisotropy of the magnets can be changed and, therefore, control the magnetization alignment. This effect is the voltage-control magnetic anisotropy (VCMA) [25] that plays an important role in the stochastic MTJ. The VCMA-MTJ is important in the computing paradigms which require stochasticity, and this will be discussed in detail later.

Besides, an essential element in spintronic technology in the last two decades is the MTJ. The resistance of an MTJ depends on the relative orientation of the magnetizations in the pinned layer (i.e., reference layer) and the free layer. The discovery of the tunneling magnetoresistance (TMR) effect [26] is one of the milestones for integrating spin-based devices with CMOS technology. Specifically, TMR gives a very large magnetoresistance ratio so that it can provide enough signal strength to the CMOS sense amplifier [27].

2.2 Applications

2.2.1 Neuromorphic computing

Compared to conventional computing technology, one of the strategies for processing data more efficiently and energy-conserving is to emulate the brain. The brain consists of the following two elementary units: synapses, and neurons. Synapses operate as connectors of neurons, while neurons interconnected by synapses receive signals from other neurons and emit spikes to the subsequent neurons if activated. Inspired by the functionality of the brain, neuromorphic computing is being intensively developed and has exhibited outstanding performance for computational tasks such as classification, recognition, and prediction. Hence, at the device level, designing artificial synapses and neurons for high-performance neuromorphic computing is of foremost importance. In this part, we will explain the MTJ-based synapse and neuron in the aspects of device features, design principles, and recent works.

2.2.1.1 Artificial synapses based on MTJ Selecting the type of neural network is dependent on the specified applications. To be specific, the non-SNN are much more computationally driven, while SNN is proposed to explicitly reproduce biological behavior such as STDP. The selected neural network model defines the behavior of the synapse. Therefore, the artificial synapse could be classified into synapse for non-SNN and synapse for SNN.

For the non-SNNs, the synapses could be regarded as an unstable memory device that does not require a 10-year retention time. These synapses for non-SNN are required to 1) represent the strength of the connection (encoding to different weights) between the connected neurons, 2) update the weights according to the output of connected neurons to realize the learning process or plasticity-like properties, and 3) keep the connection strength within one iteration (short-term memory functionality). On the other hand, the synapses for SNNs are attempted to emulate biological behavior in a further precise way. One of the popular inclusions for more complex synapse properties is the STDP mechanism, which requires the connection strength to change over time.

Generally, memory devices can be used as the artificial synapse because they can memorize and be repetitively rewritten. So far, many works apply memory devices to synapses in both non-SNNs and SNNs. For example, floating-gate transistors are used as analog memory cells for synaptic weights storage [28], [29], while conductive-bridging RAM changes the connection strength via electrochemical properties [30]. Likewise, memristors based on ferroelectric materials [31], [32] and phase change memory [33] have been applied in the hardware of synaptic systems due to their plasticity-like and especially the STDP-like behavior.

Spintronic devices, additionally, have been considered as a competitive candidate for the hardware implementation of synapses. Spintronic devices can be non-volatile and allow for a variety of tunable spin dynamics such as intrinsic stochastic switching, the dynamics of domain wall (DW), and so on. These various spin dynamics could emulate synapses with different behaviors. For instance, because uniform and continuous variation of synaptic weights are required to guarantee the accuracy of the training [6, 34], linear synaptic behavior is desired in non-SNNs that exploit supervised learning such as the backpropagation (BP) learning rule. Memristor-like behavior that the synaptic weights that depend on both input amplitude and duration time is required for the SNNs with unsupervised learning,

especially for STDP learning rules [35], [36]. Spintronic devices could emulate the synaptic behavior for both non-SNNs and SNNs. The spintronic devices not only keep the merits of fast operation speed which outperform conductive-bridging RAM, phase change memory, and some of the non-spin-based memristors but also are energy-efficient compared to volatile floating-gate transistors.

Due to exhaustive back-and-forth memory-processor operations and inevitable leakage current, the early artificial synapse based on a group of transistors requires intensive energy [37]. The power consumption can be reduced by introducing non-volatile memory units into CMOS circuits. Compared to pure CMOS circuits, the proposed CMOS/MTJ-hybrid structures [38], [39] exhibit reduced energy consumption and computational latency when performing the classification and recognition tasks. Nevertheless, in such CMOS/MTJ-hybrid structures, the MTJs just function as associative memories to store the synaptic weights of hardware, which would not fully exploit the versatility of MTJs.

For the MTJ-based synapse, however, one of the key issues is how to use the binary MTJ to mimic the analog synapse, which would realize a gradual or semi-gradual change of synaptic weight and thus achieve high computational accuracy. There are mainly two strategies to represent the strength of the connection as follows: 1) gradually changed the probability of the binary switch (bistate), and 2) gradually changed states (multi-state).

At the very beginning, the binary MTJ is designed to be thermally stable to target 10-year information preservation by designing the high energy barrier between the different states. As a result, the energy consumption required to switch nonvolatile MTJ is relatively high, typically 100 fJ [40], as compared with 23 fJ per synaptic event [41]. Subsequently, if the circuit requires frequent changes in the stored information to realize rapid updates for the synaptic weights, the MTJs are not energy-efficient [42], [43]. Additionally, MTJs are required to have a minimum variation, which requires severe constraints on nanofabrication. When the energy barrier between the two states is comparable to thermal energy, changing the state of the MTJ requires less power but introduces much noise. For the paradigm of neuromorphic computing, on the other hand, the neural networks are tolerant to and even could harness noise, variability, and stochasticity for the computation [44].

Furthermore, in binary MTJs, the resistance cannot evolve gradually, but the probability of an MTJ switching during a voltage pulse can be tuned gradually by the amplitude and duration of the pulse. Using the bistate synapses makes learning slower but offers the network increased memory stability. Furthermore, given that spin-transfer-torque magneto-resistive random-access memory (STT-MRAM) has been intensively developed in both the academic and industrial world, the neuromorphic chips composed of spintronic devices would tend to start from STT-MTJ with binary switch behavior. STT-MTJs could comprehensively emulate the functionality of biological synapses because of their intrinsic stochastic switching behavior [45].

Based on this principle, Vincent et al. [46] designed an artificial synapse based on a single STT-MTJ. By encoding the binary states of the STT-MTJ to the two weights representing light and dark, as shown in Fig. 3(a), the artificial neural network composed of such STT-MTJ artificial synapse is capable of unsupervised learning. When an input neuron spikes, a brief read pulse is applied to the crossbar and currents will reach the different output neurons simultaneously. Then, by design choice, only the inputs coming from the P synapses are integrated by the output neurons. When an output neuron spikes, other output neurons will be inhibited and their internal variable is reset to zero. The synapses in the crossbar architecture successfully counted cars via recognizing the change of brightness in lanes, which is showed in Fig. 3(b). Due to the inherently stochastic switching, only two STT-MTJs switch states are enough for the presented example.

To further improve the performance of the STT-MTJ as a synapse, Locatelli et al. [47] reported strategies to effectively control the bit error rate by modulating the programming pulse amplitude or duration. Conversely, it is challenging for a synapse based on a single binary MTJ to handle complex computation tasks. Thus, the analog behavior of an artificial synapse is desired, resulting that such a synapse could exhibit multiple distinguished states corresponding to multiple discrete weights. To realize the analog behavior, embedding such binary-state STT-MTJ to crossbar frameworks has been proposed. Fig. 4(a), (b) shows that the optimized STT-MTJ crossbar synapse with multi-state is constructed by stacking several binary-state STT-MTJs [48] or connecting in the 2D architecture [49], named compound magnetoresistive synapse (CMS), respectively. The CMSs are offered an analog-like weight spectrum that results from different states of the individual MTJs leading to a gradual conductance modulation. CMSs are advantageous for number recognition tasks with high tolerance to fault and variation, nevertheless at the cost of device number and integration area. To solve this problem, Zhang et al. [49] proposed a 3D crossbar structure, that each MTJ is sandwiched by the vertical electrodes and the horizontal electrodes. The two vertical electrodes and the two horizontal electrodes are connected to the post-neurons and the pre-neurons, respectively.

Compared to STT-MTJs, in principle, SOT-MTJs are expected to perform better in terms of energy consumption, speed, and endurance [50]. Srinivasan et al. [51] demonstrated a SOT-MTJ which is a building block of the proposed

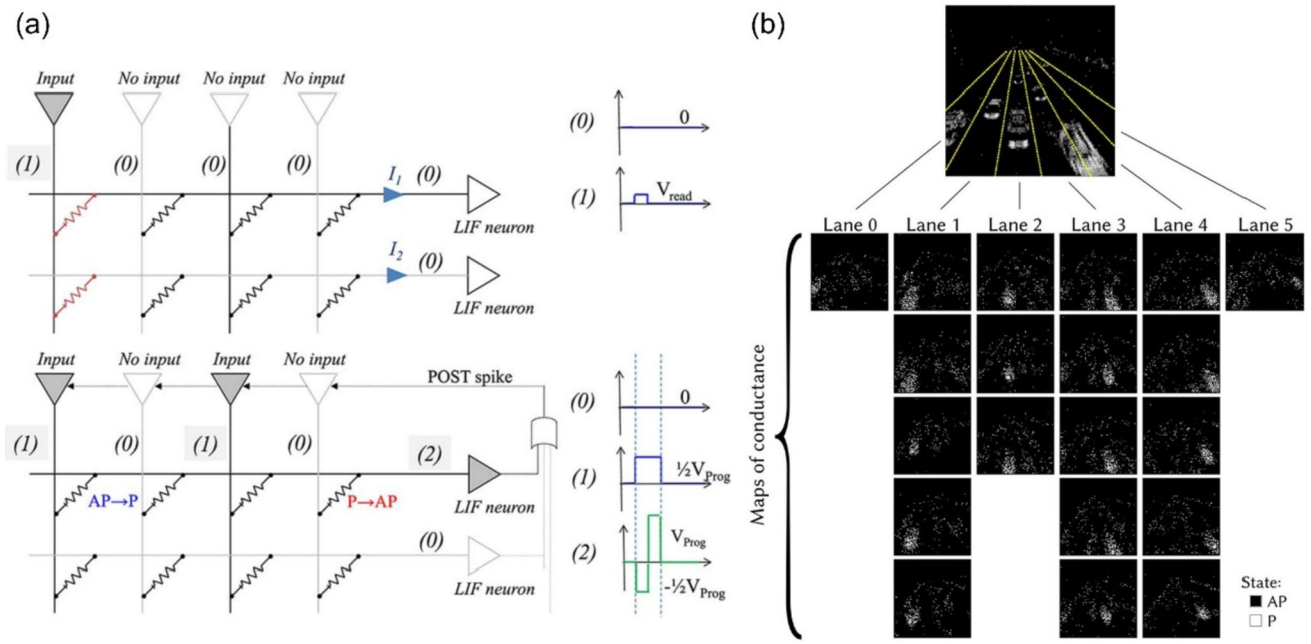

Fig. 3 The ANN is composed of binary STT-MTJs that function as synapses. **a** Schematic of the crossbar architecture. Read operation occurs when an input neuron spikes and STDP (write) operation occurs when an output neuron spikes. Waveforms (1) and (2) are applied concurrently. **b** The final state of the MTJs is organized as the input pixels in the image. White is *P*, black is AP state. Every sub-image represents one output neuron. The figures are adapted from Ref. [46] with the authors' permission

all-spin SNN which is highly energy-efficient. The synapse based on the SOT-MTJ consumes less than 36 fJ per spiking event. It is also helpful for realizing the stochastic-STDP learning algorithm. The CMS based on the SOT-MTJs has also been reported. Such CMS with analog-like behavior can handle more complex computation tasks with high accuracy while keeping the power consumption low [52]. Alternatively, as shown in Fig. 5(a), Ghanatian et al. [53] created multiple states by putting multiple SOT-MTJs on a shared heavy metal layer but with different cross-section areas. Although the SOT-MTJ-based synapses have been proposed with progressive significance, the challenge of scaling is unavoidable. Therefore, developing an artificial synapse based on a single MTJ with multi-state would be desirable for reducing the area.

The multi-state synapses are especially desired for SNN, which requires the connection strength to evolve continuously depending on the past activity of the connected neurons. The property is plasticity, which allows neural networks to learn and reconfigure. Magnetic devices are particularly well adapted for implementing plasticity [54] due to their memory effects and tunability. Embedding a magnetic DW in the MTJ structure can be used to implement synaptic plasticity. Such memristive behavior has been demonstrated in MTJ with more than 15 intermediate resistance states [55]. Furthermore, it has also been shown that similar continuous magnetization variations can be triggered by SOT in a magnetic stripe on top of an antiferromagnetic layer [56]. Memristive-like features can then be obtained by fabricating a tunnel junction on top of the bilayer stripe. These spintronic memristors could be used as multi-state synapses, similar to many strategies proposed for other memristive technologies [57], [58]. Moreover, Wang et al. [59] proposed a compact model of a synapse based on current-induced DW motion MTJ (CIDWM-MTJ), driven by the SOT, the CIDWM-MTJ exhibited reduced threshold current and a faster DW motion of 400 m/s compared to CIDWM-MTJ driven by the STT [60]. Cooperated with a peripheral circuit, the CIDWM-MTJ with low power consumption and high speed would be promising for high-performance SNN applications [61]. Besides, Siddiqui et al. [62] designed a linear synapse shown in Fig. 5(b) based on nine MTJs with a shared free layer to realize multilevel linear synaptic weight generation, which would be favorable for DNN applications. Lourembam et al. [63] reported a strategy for formatting and stabilizing metastable magnetic domains by the voltage pulse in the MTJ, combining binary switch and spin textures to achieve the four-state synapse by using only a single MTJ shown in Fig. 5(c). The MTJ was fabricated without any of the domain-wall pinning methods and, therefore, can alternatively realize metastable multi-domain states. Hong et al. [64] demonstrated a dual-domain-and-dual domain MTJ to realize the eight-state synapse.

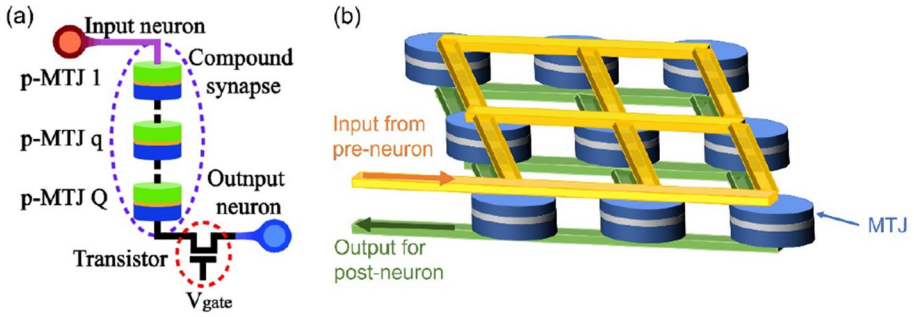

Fig. 4 The crossbar structure based on MTJ-array synapse for realizing multi-level weights. **a** multi-state synapse using several binary-state stacked STT-MTJs. The figures are adapted from Ref. [48] with the authors' permission. **b** STT-MTJ crossbar synapse connected in parallel

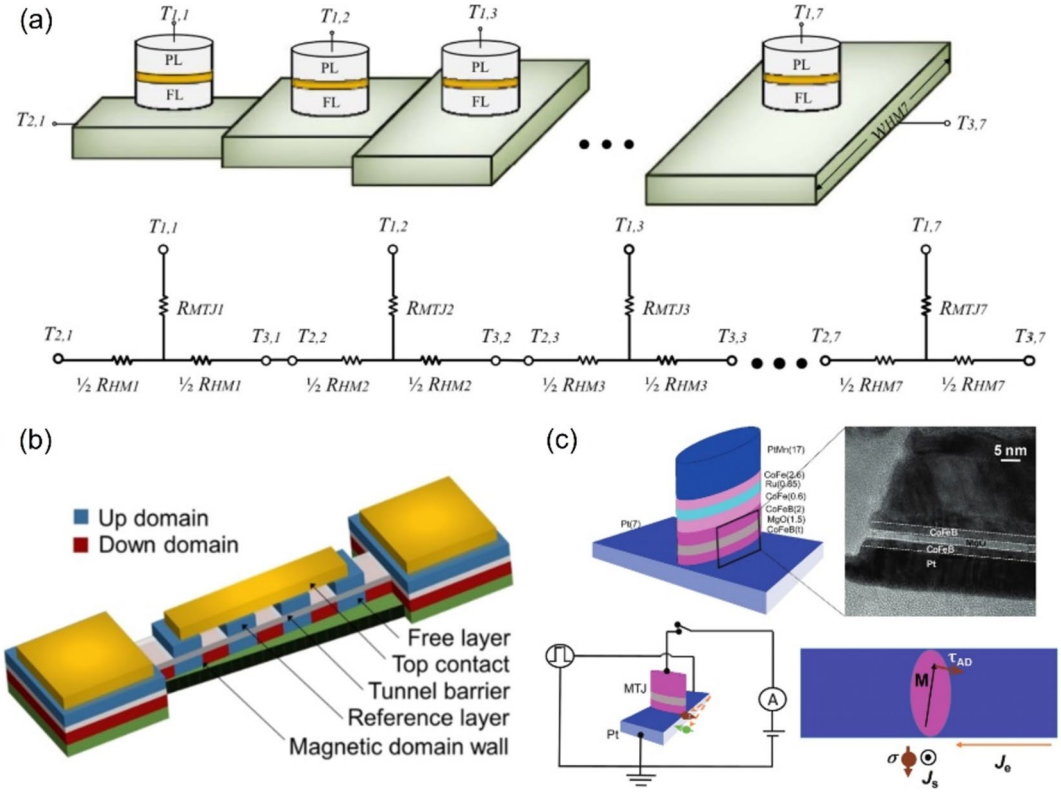

Fig. 5 The MTJ-based synapse for realizing multi-level weights. **a** SOT-MTJs on a shared heavy metal layer but with different cross-section areas. **b** MTJs with a shared free layer. **c** A single MTJ combining binary switch and spin textures to achieve the four-state synapse. The figures are adapted from Ref. [53, 62], and [63] with the authors' permission, respectively

2.2.1.2 Artificial neurons based on MTJ The typical behavior of a neuron is to accumulate charge when the neuron's membrane potential changes. The neuron would generate a spike when its membrane potential reaches a threshold.

McCulloch-Pitts Neuron model [65] is used in most ANNs. For this model, the output of neuron j is governed by the following equation:

$$y_i = f\left(\sum_{i=0}^{N} w_{ij} x_i\right) \tag{1}$$

where y_j is the output value, f is an activation function, N is the number of inputs into neuron j, $w_{i,j}$ is the weight of the synapse from neuron i to neuron j, and x_i is the output value of neuron i. In this neuron model, firstly, the neuron would integrate the weighted outputs of the pre-neurons through synapses. Next, this linearly combined integration is processed by the activation function of the neuron and then emits output to the next neuron. The activation function plays a key role in data processing. Choosing the activation function is heavily dependent on

the particular neural networks and the different activation functions can be realized by the behaviors of the devices.

For artificial neurons applied in non-SNN, there are a variety of implementations of the traditional McCulloch-Pitts neuron model. The perceptron composed of CMOS [66], which implements a simple thresholding function, is commonly used in hardware implementation. As shown in Fig. 6(a), the simplest activation function is the step function [41], and the neuron hardware for this activation function requires less area utilization and would not be computationally intensive. However, the mainstream learning algorithms such as the BP algorithm are gradient-based. As the step function is not differentiable and not suitable for this algorithm, the other hardware-based activation functions, including the ramp-saturation function [67], linear [68], and piecewise linear [69] functions shown in Fig. 6(b), (c), have been implemented to match the gradient-based learning algorithm. As the complexity of the activation function is increased, i.e., from linear to nonlinear function, the overall accuracy of the learning process is increased, in that nonlinear activation functions, shown in Fig. 6(d) such as the basic sigmoid function [70] and the hyperbolic tangent function [71], gives derivatives with continuous variation offering a high resolution for the gradient-based learning algorithm.

Nonlinear activation functions, unfortunately, would cause complexity in computation and hardware implementation, The MTJ-based neuron could alleviate this challenge due to its nonlinear dynamics. For example, the motion of DWs could realize neural-like integration and thresholding. Besides, thresholding can be achieved by using a standard MTJ, which switches only if the amount of current it receives is above the critical current. The neurons together with the activation functions mentioned above are mainly for the non-SNN which is computationally intensive, and when the neurons are activated, they would not necessarily return to their initial states.

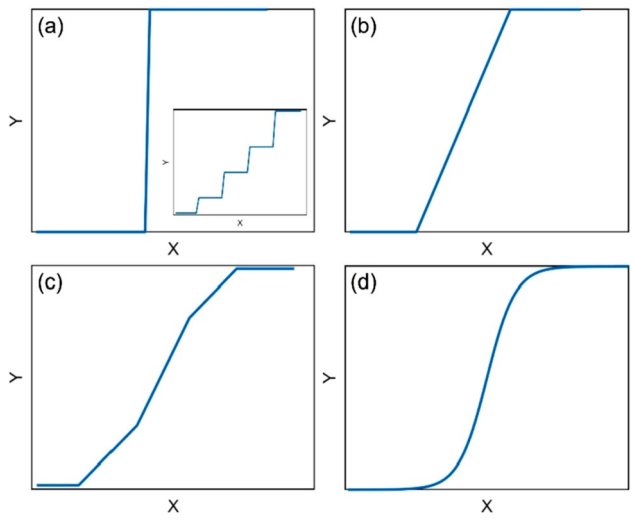

Fig. 6 Plots of activation functions for neurons. **a** Step function. **b** Ramp-saturation function. **c** Piecewise linear function. **d** Nonlinear activation function

On the other hand, for artificial neurons applied in SNN, the behavior of fire means that when the neurons are activated, they would emit spikes and then back to their initial state spontaneously. A simple set of spiking neuron models belongs to the integrate-and-fire family, which is a set of models that vary in complexity from relatively simple (the basic integrate-and-fire) to those approaching complexity levels near that of the Izhikevich model [72] and other more complex biologically-inspired models. In general, the neuron models where action potentials are described as events are called "integrate-and-fire" models. Integrate-and-fire models have two separate components that are necessary to define their dynamics: 1) an equation that describes the evolution of the membrane potential, and 2) a mechanism to generate spikes. Although the "integrate-and-fire" models are still less biologically realistic but produce enough complexity in behavior to be useful in spiking neural systems. The simplest integrate-and-fire model maintains the current charge level of the neuron. Furthermore, there is a leaky integrate-and-fire (LIF) [73] implementation that expands the simplest implementation by introducing a leak term to the model, which leads to the potential for a neuron to decay over time. The LIF models use the following two ingredients: 1) a linear differential equation to describe the evolution of the membrane potential, and 2) a threshold for spike firing. It is one of the most popular models used in neuromorphic systems. Spin-torque nano-oscillators are specific types of MTJ, and the oscillation amplitudes have memory due to finite magnetization relaxation, which can imitate the leaky integration of neurons [74], [75]. Moreover, the next level of complexity of the neuron model is the general nonlinear integrate-and-fire method, including the quadratic integrate-and-fire model that is used in some neuromorphic systems [76]. These have also been used in neuromorphic systems. Nonetheless, the models aforementioned make use of the fact that neuronal action potentials of a given neuron always have roughly the same form, and no attempt is made to describe the shape of an action potential. If the shape of an action potential is always the same, the shape cannot be used to transmit information, i.e., rather information is contained in the presence or absence of a spike. As a result, action potentials are reduced to events that happen at a precise moment in time. Alternatively, another level of complexity is added with the adaptive exponential integrate-and-fire model [77].

In addition to the previous analog-style spiking neuron models, there are also implementations of digital spiking neuron models. The dynamics in a digital spiking neuron

model are usually governed by a cellular automaton, rather than a set of nonlinear or linear differential equations. A hybrid analog/digital implementation has been created for neuromorphic implementations [78], as well as implementations of resonate-and-fire [79] and rotate-and-fire [80] digital spiking neurons. A generalized asynchronous digital spiking model has been created to enable the exhibition of nonlinear response characteristics [81]. Digital spiking neurons have also been utilized in pulse-coupled networks [82], and a neuron for a random neural network has been exploited in hardware [83].

2.2.2 Stochastic computing

Conventional computing is based on the binary representation of information in terms of "0" and "1", as known as "bits". These bits of information are processed and stored by stable deterministic devices like the MOSFETs or MTJs with stable magnets having energetic barriers of the order of 40–60 times the thermal energy at room temperature. Probabilistic spin logic (PSL) is a new paradigm of computing [84] that relies on probabilistic bits (p-bits for short) that fluctuate randomly between 0 and 1, with probabilities that can be tuned by an input. Besides, exploiting physics properties to do computation has become increasingly attractive in recent years, because such computation can naturally converge, which is governed by the physical laws instead of complex algorithms. In the field of physics-inspired computing, stochastic computing has gained significant interest due to its excellent performance in solving non-deterministic polynomial (NP)-hard problems. The unit of stochastic computing is *p*-bit which is realized by several devices with non-deterministic behavior. In this part, we will illustrate the device features, design principles, and recent works of the MTJ-based *p*-bits.

2.2.2.1 P-bit based on MTJ Compared to the traditional deterministic von Neumann approach, stochastic computing would endow improved efficiency for solving computationally hard problems such as the COP [85] and factorization [17]. For stochastic computing, a large number of independent sources of the stochastic signal are needed, in that they are often based on Markov chain Monte Carlo techniques such as Gibbs sampling [86]. Therefore, energy-efficient, high-density hardware for generating true-random noise sources is of significance.

P-bits are evolved from random number generators (RNGs) and the key feature of the p-bits is the tunability of the probability for their outputs, i.e., concerning the inputs, the outputs of hardware that function as a p-bit should obey a specific probability distribution, generally, the sigmoidal-like probability distribution. The ideal p-bit behavior is described by the following two equations:

$$\mathrm{m_i} = \mathrm{sgn}\left\{\tanh\left(\mathrm{I_i}\right) - \mathrm{r}\right\} \tag{2}$$

$$I_i = \sum_i J_{ij} m_j + h_i \tag{3}$$

where Eq. (2) represents the state of the *i*th p-bit (given by m_i) as a function of its input I_i. "*r*" is a random number with a uniform distribution between – 1 and 1, that captures the stochastic aspect of the output. Equation (3) provides the expression for the input I_i in terms of the connection strengths J_{ij} of other p-bits in the network to the *i*th p-bit and the local bias *hi*. This is analogous to the concept of a Binary stochastic neuron (BSN) used in the field of stochastic neural networks [12].

MTJs have been integrated into CMOS technologies for memory applications, and they are engineered to have stable magnetic states. However, MTJs could become naturally fluctuating if choosing proper materials or geometry, resulting in such MTJs being one of the natural candidates for p-bit hardware. Evaluating the speed of such fluctuation is essential because it relates to the speed of computation in a stochastic computing scheme [87]. Stochastic fluctuation has been reported in superparamagnetic MTJs [17, 88], [89] with low uniaxial anisotropy and energy barriers, operating in the millisecond time regime. For the superparamagnetic MTJs, the fluctuation rate follows an Arrhenius-like relation [90]:

$$\tau = \tau_0 exp(\frac{E_B}{k_B T}) \tag{4}$$

where the magnetization of a uniaxial anisotropy nanomagnet has two stable directions along its anisotropy axis: "Up" and "Down" for nomination. The two states are separated by an energy barrier, E_B, which stabilizes the magnetization in one of the states. τ_0 is called the attempt time, a material-dependent parameter of the nanomagnet. An exponential increase in fluctuation speed is expected upon reducing the energy barrier E_B, with a frequency scale set approximately by the attempt frequency of $1/\tau_o \propto \alpha\gamma H_k$, where α is the Gilbert damping coefficient, H_k is the anisotropy field, and γ is the gyro-magnetic ratio. However, because both energy barrier $E_B = mH_k/2$ where m is the total moment of the macro-spin, and attempt frequency $1/\tau_o$ reduce with the decreasing of H_k, the fluctuation speed of superparamagnetic MTJs is largely limited.

A potential approach to increase the fluctuation speed, consequently, is to exploit easy-plane anisotropy that can allow magnetic fluctuation confined in the plane and meanwhile keep high-speed fluctuation dynamics [91]. In the works [92], the energy barrier is determined by the shape of the MTJ. A low relative energy barrier could be achieved by constructing a circular in-plane junction. The attempt

frequency for this magnetization configuration is then related to the free layer's easy-plane anisotropy field, which can be remarkably higher than the easy-axis anisotropy field H_k, thus endowing a faster fluctuation speed. Furthermore, nanosecond fluctuation in the in-plane MTJs [93]-[94] has been achieved by investigating the mechanism for controlling relaxation time.

Introducing the VCMA effect to the MTJs is another strategy for achieving fast fluctuation speed. By applying voltage pulses, the magnetic anisotropy of the free layer would be switched between the in-plane and out-of-plane directions together with the thermal noise, achieving the random fluctuation of the magnetization without reducing the energy barrier. The VCMA-MTJs [95], [96] with stochasticity have been applied as not only true random number generators (TRNGs) but also p-bits the output probability could be tuned by the amplitude and the enduring time of the voltage pulses.

Besides the stochastic binary-switching MTJs, spin torque nano-oscillator (STNO) could also be operated as hardware of p-bit, exploiting intrinsic frequency fluctuation caused by thermal noise [97]. Cooperated by a peripheral circuit, the digital p-bit based on STNO would be able to act as a p-bit array by time division multiplexing, which overcomes the limitation of calibration and coupling connections encountered by synchronous p-bit arrays. More details of the applications of MTJs to stochastic computations will be discussed in Sect. 4.

3 Neuromorphic computing

In the previous section, MTJ-based artificial synapses and neurons were introduced. Facing the challenge of the Von Neumann bottleneck and the decline of Moore's Law [98], more efficient neuromorphic computing emerges as the times require, which is inspired by the human brain and can process and store the data simultaneously. Spintronic devices provide a feasible approach to building neuromorphic computing systems, due to their intrinsic dynamics being akin to biological synapses and neurons. Additionally, their low energy consumption, non-volatility, high speed, and potential for pure spin current transport make them one of the most promising candidates [99], [100].

Inspired by the human brain, ANN was created to mimic the functionality of the human brain to store and process information. Similar to the human brain, ANNs also consist of many synapses and neurons. As mentioned in Sect. 2.1, synapses and neurons are the fundamental building blocks of the brain. Among them, synapses are related to the formation of memory, while neurons are related to the information processing [101].

In the nervous system of the human brain, each synapse is a specialized junction with two neurons, which allows a neuron to transmit electrical or chemical signals to another neuron, as shown in Fig. 7(a). The information is transformed from the axon of the pre-neuron to the dendrite of the post-neuron through a synapse. In conventional ANN, there are generally two types of synapses: one requires multilevel memory, while another one relies on stochastic binary memory devices. As we have mentioned, one compact MTJ is sufficient to imitate the functionalities of the biological synapse.

Neurons play an essential role in producing and transmitting action potentials in neural networks. Their functionalities are intricate and plenteous [88]. However, in the conventional ANN, an artificial neuron is a mathematical function based on a model of biological neurons. Each neuron takes inputs, weighs them separately, sums them up, and passes this sum through a nonlinear function to produce output, where the nonlinear activation functions are extracted from the complex neural mechanisms [101]. The same as artificial synapses, MTJs exhibit great potential for mimicking artificial neurons. According to the characteristics of MTJ switching, the neuron models can be distinguished into two categories: deterministic and stochastic neurons [47].

The content for this section is organized as follows: First, the development and recent progress of neuromorphic computing are reviewed, including the MTJ-based single perceptron, multi-layered perceptron, conventional neural network, and recurrent neural network. The encoding approaches and learning methods are highlighted. The last part concludes the topic and envisions the challenge and prospects.

3.1 Artificial neural networks

In this part, several different neural networks are introduced, including MLP, CNN, RNN, and oscillator neural networks. From multi-layered perceptron to CNN, by increasing the number of layers or changing the network architecture, it is possible to build and perform tasks such as image recognition with a large number of neurons and synapses based on MTJs. In addition, RNNs and oscillator neural networks show potential for time-domain signal processing. ANN based on magnetic nano-oscillator and RNN will be discussed separately.

3.1.1 The perceptron based on MTJ

The development of neural networks has mainly gone through three periods: The first generation is a perceptron

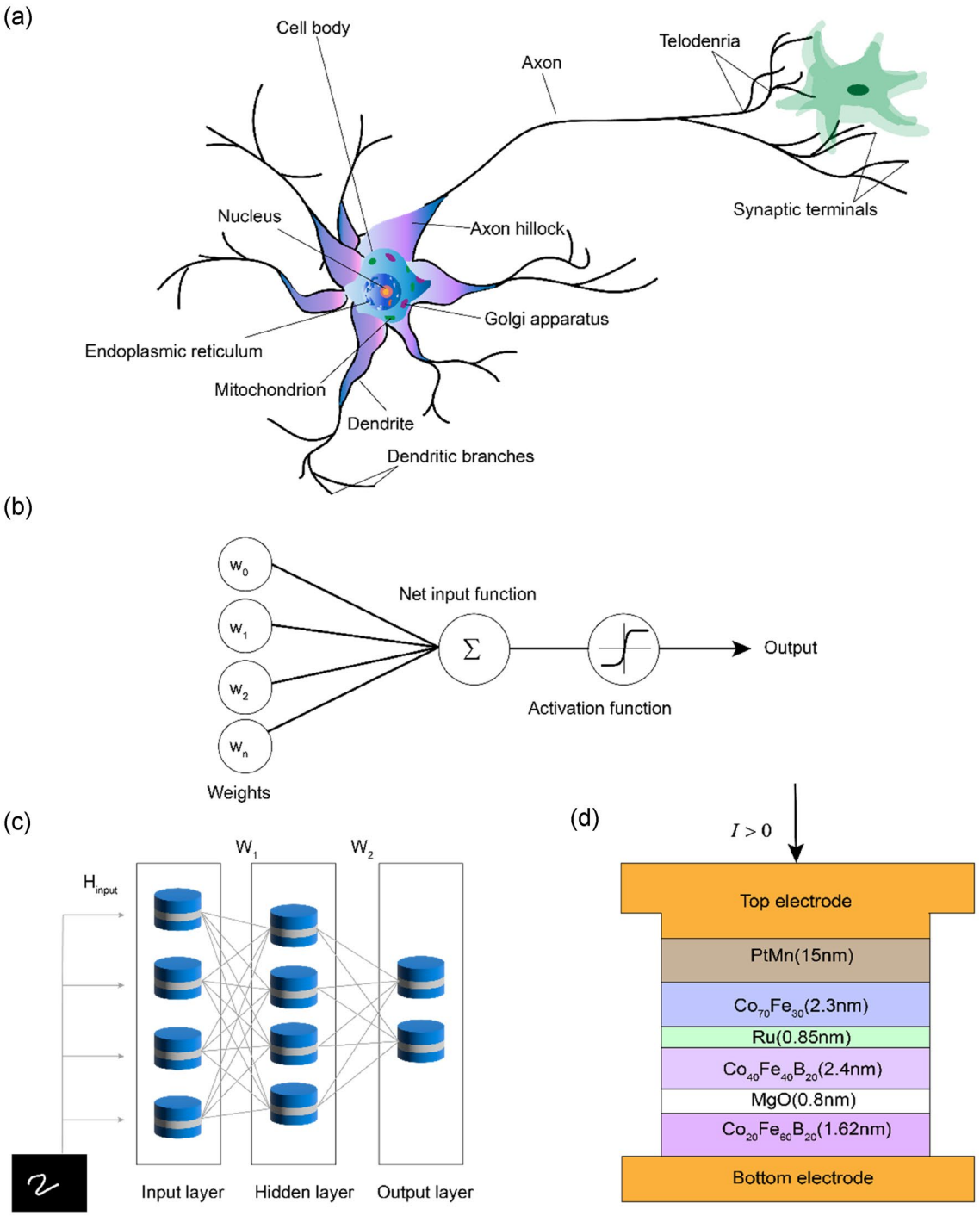

Fig. 7 **a** Schematic of biological neuron and synapse. **b** Diagram of perceptron with its weights, input function, activation function, and output. **c** Architecture of MTJ-based MLP for recognizing the handwritten digit. d The neuron model based on MTJ used in (**c**). The figures are adapted from Ref. [110] with the authors' permission

capable of binary operations. The second generation is an MLP and CNN with hidden layers, and the third generation is the event-driven SNN.

The concept of perceptron has a landmark effect on the development of the neural network. The single-layer perceptron model was proposed by Frank Rosenblatt in 1958 [102]. A perceptron is implemented as a binary classifier, which

decides whether an input belongs to a specific class. As shown in Fig. 7(b), a perceptron consists of four main parts including input values, weights, net sum, and an activation function. During the learning process, the input values are multiplied by their weights. Additionally, all of these multiplied values are added together to create the weighted sum. The weighted sum is, after that, applied to the activation function, producing the perceptron's output, and only when the weighted sum exceeds a certain threshold, the neuron is activated. To ensure the output is mapped between (0, 1) or (− 1,1), the step function is chosen to be the activation function. In addition to the step function, the activation function also includes the sigmoid function ($f(x) = 1/(1 + e^{-x})$) [103], the ReLU function ($\mathrm{ReLU}(x) = \max(0, x)$) [104], the tanh ($\tanh(x) = (1 - e^{-2x})/(1 + e^{-2x})$) [105], and so on. Since the step function is not differentiable at $x = 0$, which makes it unusable for BP. The sigmoid function is the most widely used class of activation functions, with an exponential shape, which is the closest to a neuron in the physical sense. The output range of the sigmoid is (0, 1), which has good properties and can be represented as probability or used for input normalization. However, sigmoid also has its own shortcomings. The first point, the most obvious, is saturation. Specifically, in the process of BP, the gradient of the sigmoid will contain a factor, once the input falls into the saturation region at both ends, the factor will become close to 0, resulting in the gradient becoming very small in BP. At the same time, the network parameters may not even be updated, making it difficult to train effectively. This phenomenon is called gradient disappearance. The sigmoid network will produce gradient disappearance within 5 layers. The second point is the offset phenomenon of the activation function. The output values of the sigmoid function are all greater than 0 so that the output is not the mean value of 0, which will cause the neurons in the latter layer to get the non-zero mean signal of the previous layer as input. To overcome this problem, the tanh function is proposed. Compared to the sigmoid function, its mean of output is 0, making it converge faster than the sigmoid and reducing the number of iterative updates. However, like sigmoid, the gradient will vanish. The ReLU function is proposed to solve the saturation of sigmoid and tanh. When $x > 0$, there is no saturation problem. Consequently, ReLU can keep the gradient from decaying when $x > 0$, thereby alleviating the problem of gradient disappearance.

3.1.2 Multi-layered perceptron and convolutional neural network

Nevertheless, the perceptron has only the output layer neurons for activation function processing, that is, only one layer of functional neurons, which limits its learning ability. In 1969, Minsky and Papert [106] proposed that the perceptron can only solve linearly separable problems, that is, if there is a plane that can separate the two types of modes, the learning process of the perceptron will definitely converge. Nevertheless, for nonlinear separable problems, the perceptron learning process will have fluctuations and cannot obtain a suitable solution, which makes the perceptron unable to solve even simple nonlinear separable problems such as XOR. After a downturn for the first generation of AI, multiple layers of functional neurons are considered. This led to the concept of MLP [107] [108], [109], also known as the neural networks (NN), in the 1980s. Unlike the single perceptron, MLP has multiple hidden layers, and it is capable of solving both linearly and nonlinearly separable problems.

Figure 7(c) shows an MLP built by voltage-controlled stochastic MTJs [110]. The structure of the stochastic neuron model is an MTJ, which consists of CoFeB/MgO/CoFeB layers, as shown in Fig. 7(d). Its stochastic switching behavior is attributable to the VCMA effect by altering bias voltages. The electric bias changes the switching probability between the stable parallel (P) and antiparallel (AP) states, which can be probed readily by measuring the time average of the resistance or voltage across the MTJ. More importantly, the switching probability curves under various external current densities resemble a commonly used activation function, the sigmoid function. The MLP composed of MTJs has trained to recognize the handwritten digits from the MNIST dataset with about 95% accuracy.

As the problems that need to be solved become more complex, more hidden layers will be needed, such as speech recognition often requiring 4 layers. However, it is also common for image recognition problems to require 20 layers, leading to the number of trainable parameters increasing dramatically. For example, assuming that the input picture is a 1 K × 1 K picture, the implicit layer has 1 M nodes, and there will be 10^{12} weights that need to be adjusted, which will easily lead to overfitting and local optimal solution problems. In this case, the learning efficiency of MLP is limited, therefore, the concept of DNN is proposed [111], and new architectures start to be used to improve computational efficiency. Typical representatives of new architectures include the CNN and the RNN, which are widely used neural network architectures nowadays.

As shown in Fig. 8(a), CNN [7, 112], [113] is widely used in image recognition, and its architecture includes different types of layers, including the convolutional layers, max pooling layers, and fully connected layers. The convolutional layer is used to find features. The features of the image can be extracted through the convolution operation so that some features of the original signal can be enhanced, and the noise can be reduced. The pooling layer is used to reduce the amount of data processing while retaining useful information. Sampling will neglect the specific position of a feature, because after a certain

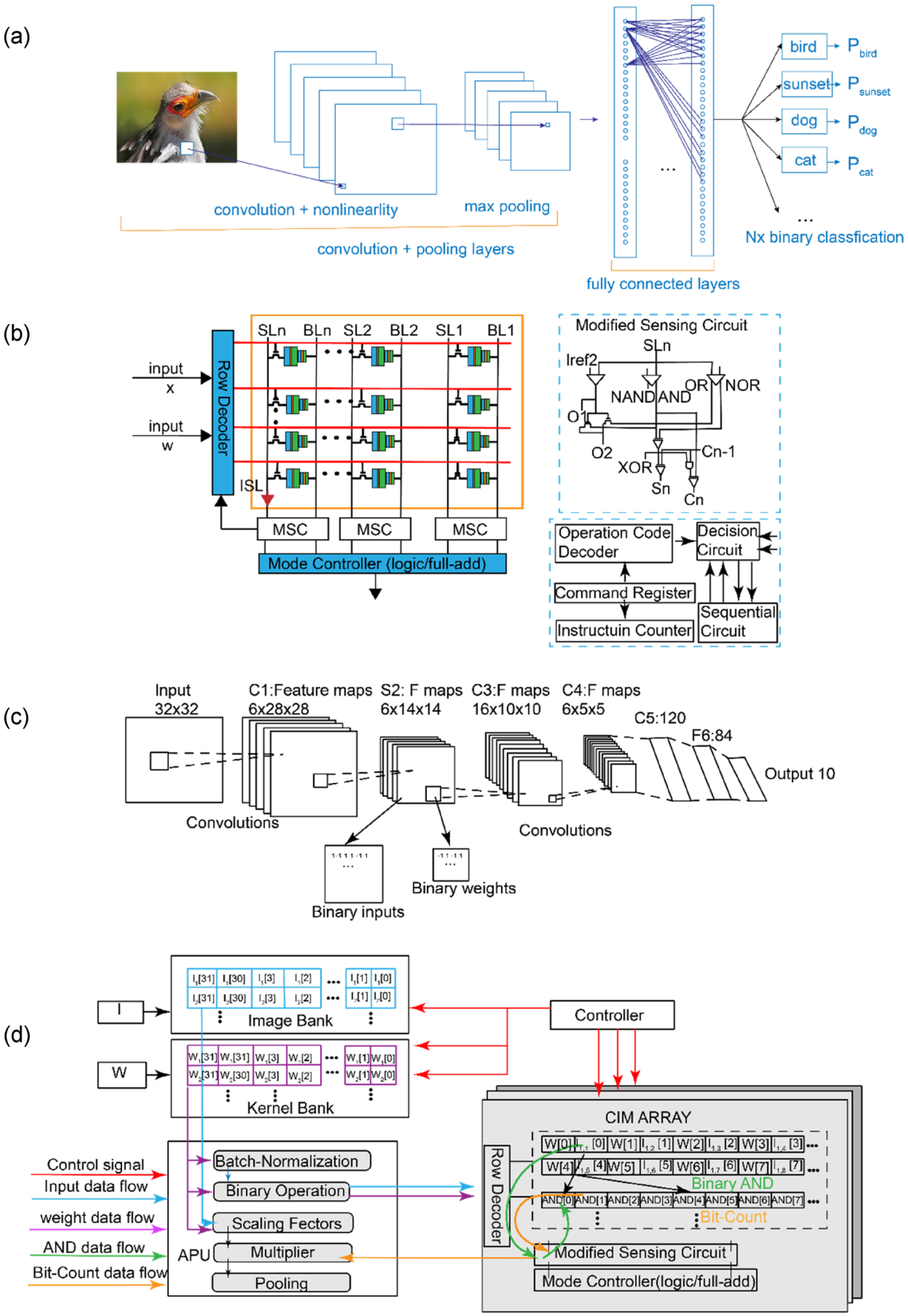

Fig. 8 **a** The architecture of CNN for image recognition. **b** The architecture of STT-computing in memory which implements the conventional operation. **c** XNOR-Net topology with STT-computing in memory as conventional layers. **d** The accelerator for BCNN and the main compute flow for convolutional layers of BCNN. The figures are adapted from Ref. [114], with the authors' permission

feature is found, its position is no longer important, and only the relative position of this feature and other features is necessary. At last, the fully connected layer is used to make classification judgments.

The input layer reads in a simple regularized image. The units in each layer take as input a set of small local neighbors in the previous layer. Through the local perception field, neurons can extract some basic visual features, such as directed edges, end-points, corners, and so on. These features are then used by higher-level neurons, and basic feature extractors that apply to a part also tend to apply to the entire image. By using this feature, CNN uses a group of units distributed in different positions of the image but with the same weight vector to obtain the features of the image and form a feature map. At each location, the units from different feature maps get their own types of features. Different units in a feature map are restricted to perform the same operation on local data at various locations in the input map. This operation is equivalent to convolving the input image with a small kernel. A convolutional layer usually contains multiple feature maps with different weight vectors, so that multiple different features can be obtained at the same location. Once a feature is detected, its absolute position in the image becomes less important as long as its relative position with respect to other features has not changed. Therefore, each convolutional layer is followed by a pooling layer. The pooling layer performs local averaging and down-sampling operations, reducing the resolution of the feature map and reducing the sensitivity of the network output to displacement and deformation. The role of the fully connected layer is mainly for classification. The features obtained through the convolution and pooling layers above are classified at the fully connected layer. The fully connected layer is a fully connected neural network. The proportion of feedback from each neuron is different. Finally, the classification results are obtained by adjusting the weights and the network.

In the overall system architecture, one CNN subarray output could relate to a long interconnect and amplifier to one or more inputs of another CNN subarray. Connections between CNN subarrays are programmed with multiplexers. Direct connections between layers speed up deep CNNs. CNN makes full use of the local information in the image. There are inherent local patterns in images (such as contours, boundaries, human eyes, noses, mouths, etc.) that can be exploited, and it is clear that the concepts in image processing should be combined with neural network techniques. For CNNs, not all neurons can be directly connected, but through the "convolutional kernel" as a mediation. The same convolutional kernel is shared within all images, and the image retains its original positional relationship after the convolution operation.

MTJs have been widely used to build CNNs [114–116]. Pan et al. [114] proposed a multilevel cell-based STT-MRAM computing in-memory accelerator for a binary convolutional neural network (BCNN). Fig. 8(b) shows the architecture of STT-computing in memory used in this paper. The modified sensing circuit is designed for logic and full-addition operation. In the meanwhile, the mode controller decides the exact working mode. In this architecture, one cell is composed of two MTJs and two bits are stored in one cell. The addition operation of the two bits is implemented within the unit, which reduces the number of required transistors and reduces the power consumption. Fig. 8(c) shows the XNOR-Net topology and XNF-Net is used as the fully connected layer. The convolution operation can be implemented by the above-mentioned STT-computing in memory. As shown in Fig. 8(d), first, the process of input preprocessing is performed, i.e., batch normalization and binarization of the input, corresponding to the path of the blue arrow (input data flow) in Fig. 8(d). The process of weight preprocessing is shown by the path of the purple arrow (weight data flow) on the left side of Fig. 8(d), and the weights are binarized. The preprocessed inputs and weights are fed into the proposed convolutional layers. The weights stored in the computing in-memory array are shared, as we mentioned as one of the advantages of CNNs. The green and orange arrows in the convolutional layers represent binary AND operations and bit counting operations, respectively. The trained scale factor and convolution result are passed to the multiplier in the APU, and the convolution calculation is completed. The final pooling operation further reduces the number of parameters.

Above all, CNN greatly reduces the trainable parameters while ensuring the depth of the network. For image recognition applications [117, 118], CNN can efficiently extract image features by convolution operations and perform tasks such as classification or recognition. Nonetheless, the deepening of its layers cannot reflect the effect in temporal sequence and is no longer suitable for processing time-domain problems such as speech recognition. Facing this, RNNs are proposed [119], which incorporate feedback operations.

3.1.3 Recurrent neural network

Although the fully connected neural network can predict something, the input of the previous data and the input of the latter data are completely independent, which makes it impossible to deal with the data with sequence information. In many scenarios, yet sequence information is indispensable. For instance, to guess what the next word of the text is, usually information from the front part of the text needs to be used, because all the content in the text does not exist alone. In order to solve the "current output of a sequence is also related to the previous output" problem, RNN was proposed [120], as an important branch

of artificial neural network. It contains a feedback mechanism in the hidden layer to achieve effective processing of sequence data. It is also known as a feedback neural network. RNNs have the powerful ability to store and process contextual information, and they have been widely used in recognition [121], natural language processing [122], computer vision [123] and other fields.

From the viewpoint of neuroscience, RNN aims at mimicking, in a reductionist scheme, how the human brain processes information. In this context, RNN assumes that the neurons are embedded in a randomly connected complex network whose intrinsic activity is modified by external stimuli. The persistent neural network activity makes the information processing of a given stimulus occur in the context of the response to previous excitations. The generated network activity is projected into other cortical areas that interpret or classify the outputs. It was this bio-inspired view that motivated one of the original RNN concepts. The main inspiration underlying RNN is the insight that the brain processes information generating patterns of transient neuronal activity excited by input sensory signals [124]. Information processing using a single dynamical node as a complex system.

Figure 9(a) shows the network structure of RNN. Through the loop connection on the hidden layer, the network state of the previous moment can be transmitted to the current moment; meanwhile, the state of the current moment can also be transmitted to the next moment. At time t, the hidden unit h receives data from two aspects, i.e., the value of the hidden unit at the previous moment of the network h_{t-1}, and the current input data x_t, and the output is calculated at the current moment through the value of the hidden unit. The input x_{t-1} at time $t-1$ can then influence the output at time t through a loop structure. The forward calculation of RNN is carried out in time series, and the parameters in the network are updated using the time-based BP algorithm. W_{sh} is the weight matrix from the input unit to the hidden unit. W_{hh} is the connection weight matrix between hidden units. W_{hy} is the connection between the hidden unit and the output unit weight matrix. b_y and b_h are the bias vectors. The parameters required in the calculation process are shared. As a result, RNN can process sequence data of any length. The calculation of h_t requires h_{t-1}, the calculation of h_{t-1} requires h_{t-2}, and so on. Therefore, the state at a certain moment in the RNN depends on all the states in the past. RNN can map sequence data to sequence data output. However, the length of the output sequence is not necessarily

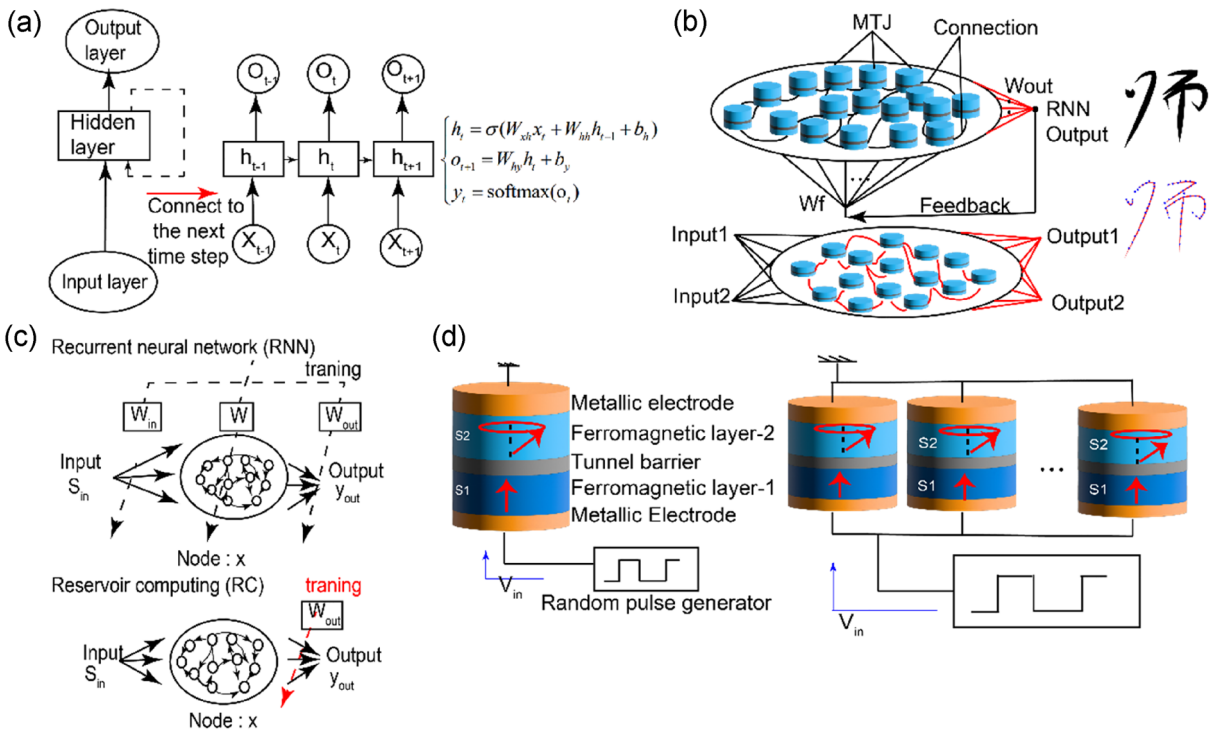

Fig. 9 **a** The typical diagram of RNN, and the connection to the next step, which is represented by the dashed line. **b** MTJ-based RNN for the Chinese character recognition. The red lines show the connections from every MTJ to output nodes with adjustable weights. The black lines show the feedback connections that transport the output signal to every MTJ in the RNN. **c** The difference between RNN and RC. **d** Schematic of a RC system using the spin dynamics in MTJs with S1 as the pinned layer and S2 as the free layer. The figures are adapted from Ref. [125, 126] with the authors' permission

the same as the length of the input sequence. According to different task requirements, there will be various correspondences. As shown in Fig. 9(b), an RNN consisting of 40 MTJs is trained for the Chinese character recognition [125]. The black lines show the feedback connections where the information is transported, and the red lines are connections between output nodes and every MTJ.

Reservoir computing is a computing framework derived from the theory of recurrent neural networks. Reservoir is a stationary, nonlinear system with internal dynamics that map input signals into a higher-dimensional computational space [120, 126, 127]. The architectural comparison of RNN and RC is shown in Fig. 9(c). RNN consists of input, intermediate, and output, and the information of the intermediate layer recursively propagates itself. The state of the middle layer is determined by the current input and the state of the past middle layer, that is, the middle layer in RNN has a memory effect. All weight matrices for the input ($\mathbf{W}_{\text{in}}$), middle ($\mathbf{W}$) and output ($\mathbf{W}_{\text{out}}$) are trained to obtain the desired output. However, when the middle layer has sufficient memory effects and non-linearities, computation can be achieved only by optimizing the output matrix ($\mathbf{W}_{\text{out}}$). This led to the concept of RC being proposed. The typical structure of RC consists of an input layer, an output layer, and a dynamic reservoir, as shown in the lower part of Fig. 9(c). The input layer feeds the input signals to the reservoir via fixed-weight connections which are randomly initialized. The reservoir maps the input signals into higher dimensions before processing them. This requires the reservoir to be sufficiently complex, nonlinear, sparsely populated, self-organized in a certain manner and capable of short-term memory. The reservoir usually consists of a large number of randomly interconnected nonlinear nodes, constituting a recurrent network, that is, a network that has internal feedback loops. Under the influence of input signals, the network exhibits transient responses. These transient responses are read out at the output layer via a linear weighted sum of the individual node states. The objective of RNN is to implement a specific nonlinear transformation of the input signal or to classify the inputs. Classification involves the discrimination between a set of input data, for example, identifying features of images, voices, time series, and so on. The only part of the system that is trained is the output layer weights with fixed connections. As shown in Fig. 9(d), RC based on MTJs has been proposed [126], where the MTJs are driven by STT.

Macrospin simulation is conducted for the spin-dynamics in MTJs, for RC. RNN can be seen as a neural network that passes on time, and its depth is the length of time. As we have mentioned, the "gradient disappearance" phenomenon is about to appear again, but on the timeline. As a result, RNNs have the problem of not being able to solve long-term dependencies. In order to solve the above problems, long short-term memory is proposed, which realizes the memory function in time through switching the cell door and prevents the gradient from disappearing.

In addition to ordinary MTJs, STNOs [18, 74] are used as the building blocks of neural networks, due to their several unique features. The structure of STNO is shown in Fig. 10(a). According to the principle of STT [128], the oscillation frequency of the STNO can be controlled by adjusting the input voltage [129]. In a biological neural network, synapses cannot be completely separated from neurons. the neuron-synapse relationship in STNO-based neural networks can better reflect this biological relationship. Further, the relationship between the oscillation frequency of STNO and the applied current or magnetic field is highly nonlinear, leading to a direct implementation of nonlinear activation functions. In addition, STNOs can be coupled by means such as direct exchange, magnetic fields, or currents, which gives them the potential to scale to large networks. As shown in Fig. 10(b), a single STNO is used to process the speech file using time multiplexing [130]. A single oscillator can simulate 400 neurons by periodically assigning time intervals to each neuron's state and using finite relaxation times to simulate coupling between neurons. This RC network can achieve a recognition rate of up to 99.6% for MNIST TI-46 speech digits. The upper part of Fig. 10(c) shows a coupled STNO-based neural network for vowel recognition [131]. The first neural layer consists of two individual neurons A and B. The input is represented by the frequency through two microwave signals f_{A}, and f_{B}. Changing the bias currents of the STNOs can change the intrinsic frequencies of the oscillators. The second layer is composed of 4 full-connected neurons. The lower part of Fig. 10(c) shows the specific implementation method of the above network. If the i-th neuron in the second layer is synchronized with neuron A in the first layer, the equality of their frequencies simulates a strong synaptic coupling. On the contrary, neuron A and neuron i with independent dynamics and frequencies simulate weak synaptic coupling between them. The strength of these synapses can be tuned by changing the bias current of each oscillator in the second layer. In many applications [130–133], STNO exhibits good stability as well as reliability and can achieve complex functions with fewer devices and higher energy efficiency.

Above all, the first-generation neural network, also known as the perceptron, was proposed around 1950. It has only two layers, the input layer and the output layer, which are mainly linear structures. It cannot solve linearly inseparable problems, and it cannot do anything with slightly more complicated functions, such as the XOR operation. In order to solve the defects of the first-generation neural network, Rumelhart, Williams et al. proposed the second-generation neural network i.e., MLP, around 1980. Compared to the

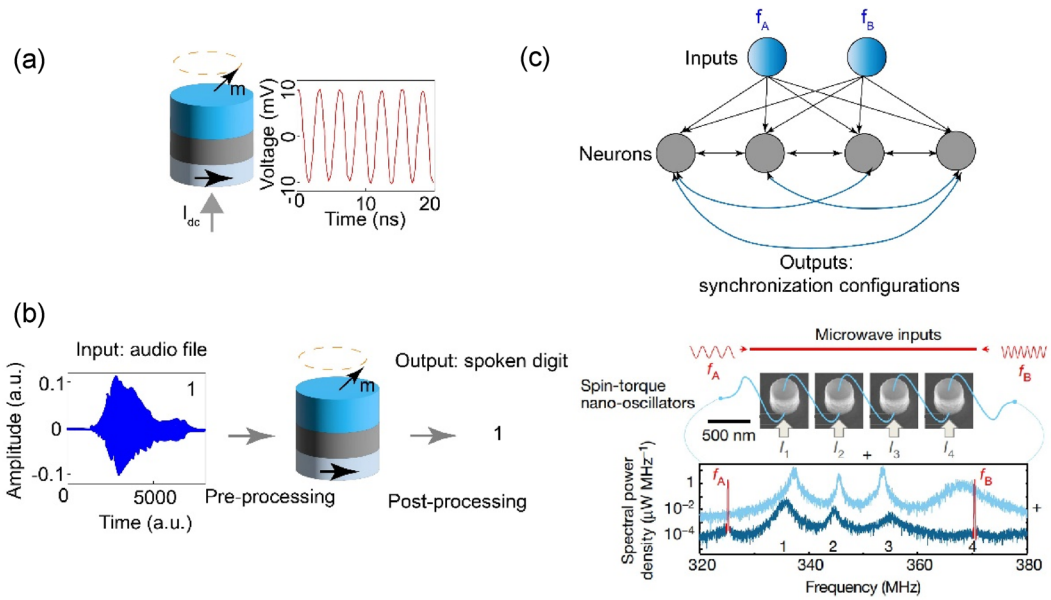

Fig. 10 NC with STNOs. **a** The structure of STNO. When a d.c. current I_{dc} is applied, the magnetization of FL gives an oscillating voltage due to the oscillating magnetoresistance. **b** The neuron model in RNN. Using time multiplexing in pre- and post-processing, a single STNO gives state of the art performance as a reservoir in a reservoir computing scheme, here recognizing the particular spoken digit as '1'. **c** Upside: schematic of RNN for vowel recognition. Downside: the input is represented by the frequencies of two microwaves applied through a strip line to the oscillators. I_{1-4} represent the bias currents and they can manipulate the natural frequencies of 4 STNOs. These STNOs can be tuned so that the synchronization pattern between the oscillators corresponds to the desired output. The figures are adapted from Ref. [130, 131] with the authors' permission

first-generation neural network, the second-generation has multiple hidden layers, which can introduce some nonlinear structures and solve the defect that the nonlinear problem could not be solved before. To conquer the problem of increasing the number of hidden layers and increasing the parameters sharply, CNN was proposed, which greatly improved the computational efficiency. To solve the sequence correlation problem, RNNs are proposed, and because the neurons are continuously interconnected, the second-generation neural network generally supports the BP [134] learning method, which is another enormous improvement in learning efficiency.

3.2 Spiking neural network

The neural networks mentioned above are usually fully connected, receiving continuous values and outputting continuous values. Although contemporary neural networks have achieved breakthroughs in many fields, they are biologically imprecise and do not essentially mimic the mechanisms of the human brain. Therefore, the third generation of a neural network, SNN, was proposed [9, 135], and uses models that best fit biological neuron mechanisms to perform computations and aims to bridge the gap between neuroscience and machine learning. Compared to the previous two generations of neural networks, SNNs are closer to biological neuron mechanisms. SNNs use spikes, which are discrete events that occur at points in time, rather than the usual continuous values in ANNs. Each peak is represented by a differential equation representing a biological process, the most important of which is the neuron's membrane potential. Essentially, once a neuron reaches a certain potential, a spike occurs, and neurons that subsequently reach that potential are reset. Furthermore, SNNs are usually sparsely connected and take advantage of special network topologies.

Neurons in an ANN communicate with each other using activations encoded with high precision and continuous values and only propagate information in the spatial domain (i.e., layer by layer). As can be seen from the above equations, the multiply-and-accumulate of inputs and weights is the main operation of the network. However, in the SNN, communication between spiking neurons is through binary events, rather than continuous activation values. The spikes from the previous neuron are transmitted to the dendrites through synapses and finally processed by soma. The equations of SNN are shown as follows,

$$\tau \frac{du(t)}{dt} = -[u(t) - u_{r1}] + \sum_j w_j \sum_{t_j^k \in S_j^{T_w}} K(t - t_j^k) \tag{5}$$

$$s(t) = 1, u(t) = u_{r2}, if u(t) \geq u_{th}$$

$$s(t) = 0, if u(t) \leq u_{th}$$

where t represents the time step, τ is a constant, and u and s represent the membrane potential and output peak. u_{r1} and u_{r2} are the resting potential and the reset potential, respectively. w_j is the weight of the jth input synapse. t_j^k is the moment when the kth pulse of the jth input synapse fires (i.e., the state is 1) within the integration time window T_w. $K(t - t_j^k)$ is the kernel function representing the delay effect. T_w is the integration time window. u_{th} is a threshold, which means whether to fire once or not.

When the membrane potential $u(t)$ (that is, the implicit potential of soma) is higher than the threshold u_{th}, the spiking neuron is regarded as fired, at which time the output potential $s(t)$ is set to 1, and then $u(t)$ returns to the reset potential u_{r2}. When $u(t)$ is lower than u_{th}, it does not fire, and the output remains at 0 at this time. At each time step, the update process of $u(t)$ satisfies a differential equation, as shown above. At each time step, the value of $u(t)$ should drop by a value as large as $u(t)$-u_{r1}, where u_{r1} is the resting potential. In the meanwhile, at each time step, the value of the membrane potential $u(t)$ should rise by a value, the value of which is related to the j input synapses of this neuron, and the weight of each input synapse is w_j, and the contribution of this synapse to the rise in membrane potential is $\sum_{t_j^k \in S_j^{T_w}} K(t - t_j^k)$, i.e., in $S_j^{T_w}$ pulses, if the input pulse at time t_j^k is the fire state (ie, 1 state), then $K(t - t_j^k)$ is calculated once and accumulated.

Unlike ANNs, SNNs use sequences of spikes to transmit information, and each spiking neuron experiences rich dynamic behaviors [135, 136]. Specifically, in addition to information propagation in the spatial domain, history in the temporal domain also has a close influence on the current state. As a result, neural networks typically have more temporal generality and lower accuracy than neural networks that primarily propagate through space and activate continuously. Since spikes are only fired when the membrane potential exceeds a threshold, the overall spike is usually sparse. Furthermore, since spikes are binary, i.e., 0 or 1, if the integration time window T_w is adjusted to 1, the multiplication between the input and the weights can be eliminated. For the above reasons, SNN networks can generally achieve lower power consumption compared to computationally intensive ANN networks.

Although SNN has many advantages such as biological proximity, low power consumption, etc. There has long been a debate about the utility of SNNs as computational tools in AI and neuromorphic computing [137, 138], especially compared to ANN. Over the past few years, these doubts have slowed down the development of neuromorphic computing, and with the rapid progress of deep learning, researchers have tried to alleviate this problem at the root, people want to strengthen the SNN by means such as improving the training algorithm [136, 139], [139–141] to alleviate this problem.

3.2.1 Biological synapses based on MTJs

In general, learning in the neural network is achieved by adjusting synaptic weights. Traditional ANNs mainly rely on gradient descent-based BP algorithms [139, 142], while in SNN, because the function of the spiking neuron is usually a non-derivable differential equation, it is extremely difficult to implement BP in SNN. There are three mainstream ways of SNN implementation: The first is to convert traditional ANN to SNN without considering any SNN characteristics [143]. However, the trained network is fully converted into a binary spike-based network. For input, the input signal needs to be encoded as a pulse train. All neurons need to be replaced with corresponding spiking neurons, and the weights obtained from training need to be quantified. The second method is BP [144]. Although it is true that the spike function of the spiking neuron cannot be directly derived to calculate the gradient, researchers have come up with many methods to estimate the gradient of the changing parameters in the network for BP, including Spikeprop [145], Slayer, etc. Although these algorithms are still controversial, they do reduce the training complexity of SNNs to some extent. The third is using STDP [146]. The principle is to use STDP to adjust the weights, thresholds, synaptic delays, and other parameters of the SNN during the training process and obtain parameters that meet the requirements of the indicators (such as classification, recognition accuracy, etc.) and the training process is completed. Lastly, the parameters are fixed and the trained SNN is obtained. Compared to the previous two approaches, STDP is closer to the actual situation in biology. It has been the most widely used method so far. Its key feature is that if presynaptic neuron activity (electrical impulse release) precedes postsynaptic neuron activity, it will cause an increase in the strength of synaptic connections. Nevertheless, if the presynaptic activity lags the postsynaptic activity, inhibition will result in weakening the synaptic connection. The effect of such temporal sequencing of presynaptic and postsynaptic activities on synaptic transmission has been thought to be directly related to brain learning and memory functions.

The learning method of biological neurons is unsupervised learning. Consequently, the initial training of SNN is considered to be unsupervised. The Hebb rules [146] provide a firm theoretical basis for the direct training of SNNs, which state that the strength of synaptic connections between two neurons changes as the neuron state changes. Extended from Hebb's rule, the STDP mechanism [147] not

only is the basis for the realization of biological learning and memory functions but also becomes the basic training principle of SNN. Long-term potentiation and long-term depression in synaptic transmission function are shown in Fig. 11(a). STDP studies the relationship between the time interval between pre-neuron and post-neuron firing and the strength of the synaptic connection between the two. When a post-neuron excites a spike sequence, if the excitation time is later than the arrival time of the spike from the previous neuron, the synaptic connection strength between the two is enhanced. The smaller the time difference, the greater the strength and the synaptic connection. The weight value is closer to the long-term potentiation in the upper half of the ordinate; on the contrary, if the excitation time is earlier than the arrival time of the pulse from the previous neuron, the synaptic connection strength between the two will be weakened. STDP can be expressed by the following equation:

$$W(s)^{\mathrm{STDP}} = \begin{cases} a_2^{\mathrm{post,pre}}(s) = +A_+ e^{\left(-\frac{s}{\tau_+}\right)}, s \geq 0 \\ a_2^{\mathrm{pre,post}}(-s) = -A_- e^{\left(\frac{s}{\tau_-}\right)}, s \leq 0 \end{cases} \quad (6)$$

where τ_+ and τ_- are time constants, and A_+ and A_- represent the maximum magnitudes of the synaptic value for the different time domains of s between the arrival and firing of neuron pulses before and after the synapse, respectively.

Spintronic devices have great potential to realize STDP [51, 148, 149]. The switching probability of the MTJ conforms to the relationship between the time and weight of the STDP [10]. The most used two-terminal MTJs [150] can be utilized to implement STDP, following its essential physical properties. The parameters associated with the heating and the switching pulses are summarized in the left part of Fig. 11(b), a high-current pulse is used to generate heat and a switching pulse is applied to the MTJ after interval Δt. The two terminals of the MTJ are connected to pre-neurons and post-neurons, respectively. The relationship between switching probability and t is shown on the right side of Fig. 11(b). When no external stimulus is applied to both terminals of the MTJ, the initial voltage is 0 V. When the pre-neuron fires, a high-current short-duration pulse is applied to the MTJ. According to Joule's law, this pulse generates heat, which results in the rapid rise of temperature as shown in Fig. 11(c). Before the spike of post-neuron arrives, the temperature of the MTJ will gradually decrease, after interval Δt, the post-neuron fires and a low-current long-duration pulse is applied to the MTJ, at which point the switching

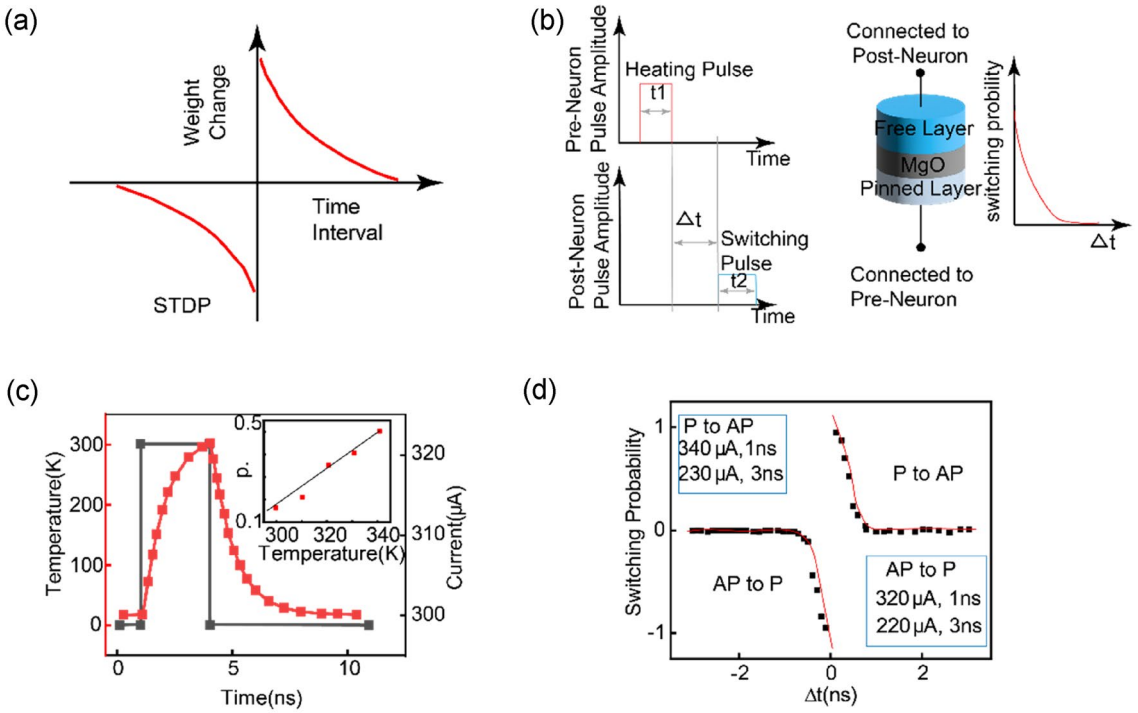

Fig. 11 **a** The typical STDP curve. If the presynaptic neuron spikes just before the postsynaptic neuron, the synaptic weight increases, and if the postsynaptic neuron spikes just before the presynaptic neuron, the synaptic weight decreases. **b** Left side: The heating pulse and the switching pulse applied on the MTJ and their time interval. Right side: The MTJ-based synapse following STDP, which is consist of two nanomagnets separated by a nonmagnetic spacer (MgO). The red curve is the switching probability of the artificial synapse as a function of pulse width. **c** The temperature (the red curve) response to the current pulse (the gray curve). The inset is the relationship of switching probability and temperature. **d** Experimental results of the proposed MTJ and its STDP behavior wherein the switching probability can be adjusted by changing the input pulses. The figures are adapted from Ref. [150], with the authors' permission

probability is measured. Since the switching probability has a strong linear relationship with the amplitude of thermal fluctuations, as shown in the inset of Fig. 11(c). If the device temperature is assumed to be constant within t_2, the final measured switching probability is related to Δt, as shown in Fig. 11(b). The switching probability decreases with the increase of Δt and the curve can be seen as the first quadrant of the STDP behavior. In contrast, when the post neuron is activated first, Δt is negative and the switching probability is negative, which also decreases as t increases, corresponding to the third quadrant of STDP behavior. It is worth noting that the negative value of the switching probability is due to the different switching directions from the previous condition. Fig. 11(d) shows the final realization of imitating STDP behavior with MTJ, which exposes the fact that MTJ is very suitable for implementing STDP.

3.2.2 Biological neurons based on MTJs

Biological neural networks possess complex action potential generation dynamics and network dynamics, while the network dynamics of SNNs are greatly simplified. The membrane potential of postsynaptic neurons is modulated by presynaptic neurons, which generate action potentials or spikes when the membrane potential exceeds a threshold. The earliest model to describe this phenomenon was proposed by Hodgkin and Huxley in 1952, namely the Hodgkin-Huxley model [151]. Since then, many models have been proposed, including Izhikevich model [72], LIF [152] neuron, etc. Although the Hodgkin-Huxley neuron model can accurately express various dynamic characteristics of biological neurons, it has too many parameters and too complex four-dimensional nonlinear differential equations, making it difficult to simulate large-scale networks. In the Izhikevich neuron model, when the membrane potential changes from the resting potential state to the fired state after being stimulated, the existence of the bifurcation mechanism makes the neuron fired. Although the model can realize various forms of pulse firing, its differential equation is still nonlinear. It is difficult to obtain the analytical expression of the state variable, and only approximate numerical simulation can be carried out. The Integrate-and-Fire (IF) [152] neuron model is defined as: when the magnitude of the accumulated membrane potential reaches a fixed threshold, a spike is sent to all neurons after the synapse. This gives the neuron model a higher level of abstraction. Meanwhile, the differential equation of the IF model is linear, and the LIF neuron model is a more simplified version that only considers leakage currents.

Figure 12(a) shows the equivalent circuit of the LIF neuron model [153]. The equivalent circuit shows that the membrane capacitance C_m and the membrane resistance R_m are connected in parallel inside the neuron model. If the presynaptic neuron sends a spike to the soma, a corresponding current I will be generated at the synapse connected to it. The current is used in two parts. One part will be used to charge the membrane capacitor C_m, which is equivalent to the process of accumulating voltage, and the other part will flow away from the membrane resistor R_m, which is equivalent to the leakage current. Once the accumulated voltage value on the neuron's membrane capacitance C_m exceeds the preset firing threshold, the neuron will fire a spike to the next neuron connected with the synapse. The first-order differential equation for the membrane potential V of the LIF model is as follows:

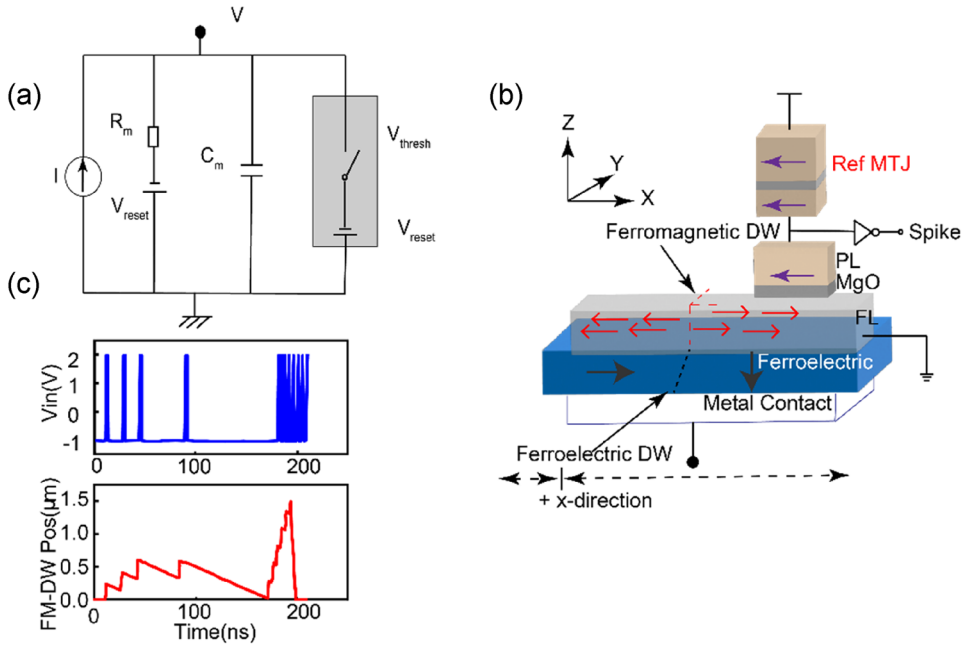

Fig. 12 **a** An equivalent circuit of an LIF neuron model. **b** The non-volatile LIF neuron based on elastic coupling between the FE-DW and FM-DW. The position of the FM-DW represents the membrane-potential, while the switching activity of the MTJ emulates the firing behavior of the neuron. **c** The LIF behavior of the neuron. The upside is the input voltage spike train received by the neuron. The downside shows the FM-DW position which acts as the membrane potential. The figures are adapted from Ref. [157], with the authors' permission

$$\tau_m \frac{dV}{dt} = -(V - V_{reset}) + R_m I \quad (7)$$

where $\tau_m = C_m R_m$ is the membrane time constant, and I is the sum of the synaptic currents received from the firing behavior of the previous group of neurons connected to each synapse. When the accumulated value of V exceeds the threshold V_{th}, a spike will be fired. The spike continues to conduct backwards with the connection of the neuron, and the membrane potential will be reset to V_{reset}. At this time, regardless of whether another pulse is received or not, the pulse will not be re-excited, and it is known as the refractory period. However, when V is less than the threshold V_{th}, the neuron will not emit a spike, and V will gradually decrease to V_{reset}. Due to its simple, linear, and event-driven characteristics, the LIF model has become the mainstream and the most widely used in SNN studies.

Much effort has been put into implementing LIF model [154–157]. CMOS-based spiking neurons often suffer from high leakage power consumption. The large-scale sparsity exhibited by SNNs makes non-volatile spintronic devices with zero standby power an excellent candidate. In addition, spintronic devices are also thought to exhibit neuronal behaviors [157]. As shown in Fig. 12(b), the LIF neuron is implemented with ferromagnetic domain wall (FM-DW) and a ferroelectric domain wall (FE-DW). The connection of the FM-DW to the underlying FE-DW allows for purely voltage control of the FM-DW. A 90°domain wall is in between the domains pointing in-plane (a-domains) and those pointing out-of-plane (c-domains). When a positive current is applied to the metal connect layer, the a-domain expands, and the c-domain decreases, causing the FM-DW to move toward the + x direction. Conversely, when a small negative voltage is applied, the c-domain expands, and the a-domain decreases, resulting in DW motion in the − x direction, which mimics the leakage behavior of neurons. As shown in Fig. 12(c), this structure simulates the leakage and firing of the LIF model well. The right terminal of the FM layer under the MTJ section can be regarded as the free layer of the MTJ. When the FL is P (AP) to the pinning layer, the MTJ is in a low-resistance state (high-resistance state). The reference MTJ is used to divide voltage, it needs to guarantee the output terminal shows a spike goes high when the lower MTJ is in a low-resistance state. The voltage-driven motion of the FM-DW enables the simulation of the behavior of biological neurons as the resistance of the MTJ changes. Due to ferroelectric materials being usually insulators, the negative voltage used to generate leakage behavior does not induce any short-circuit leakage current. Meanwhile, DW is non-volatile [158], which makes the DW-based LIF model exhibit low energy consumption.

3.2.3 Implementations of spiking neural network

The biological proximity and non-volatility exhibited by spintronic devices make them one of the candidates for implementing SNNs [99, 100]. There have been many studies successfully constructing the synapses and neurons needed to realize SNN with MTJs [150, 157]. As shown in Fig. 13(a), the SNN consists of three parts: pre-neurons, post-neurons, and the synapses as their connect junctions [159]. In the spintronic neuron shown in Fig. 13(b), during the writing process, the applied current integrates the resistance of the MTJ to the threshold value and then the neuron is fired. Noting that the path for the write current being gated off during the read mode. The neuron's response at this time can be obtained by applying a read current, additionally, the reset current is applied to initialize the neuron. These neurons are event-driven [51] and their working mode follows the cycles like the ones mentioned above. Figure 13(c) shows the use of a three-terminal SOT-MTJ as a synapse, which exhibits the STDP characteristics. Therefore, the previously proposed synapses and neurons can be connected in a crossbar array as shown in Fig. 13(d), composing the SNN architecture. This MTJ-based SNN has been used to learn the MNIST dataset with 200 neurons, achieving high energy efficiencies with an average energy consumption of 1.6 fJ.

3.3 Challenges for neuromorphic computing

The emergence of SNN facilitated the development of neuromorphic computing. In terms of learning methods, unsupervised learning mainly includes STDP learning methods based on Hebbian Rule, while supervised learning has developed representative learning methods such as the Remote Supervisor Method. In recent years, IT giants such as Apple, Google, Intel, and IBM have begun to enter the field of AI chips. IBM started the research and development of neuromorphic hardware as early as 2011 and announced in 2014 that the TrueNorth chip consists of 100 million neurons and 256 million synapses. It breaks through the architectural bottleneck of traditional computers when dealing with large-scale problems and further moves towards brain-like computing. Intel's Loihi chip is also a representative product for the development of neuromorphic hardware. Although SNN is favored for its advantages of low power consumption, high efficiency, and event-driven processing, its development and application are not smooth, and there are still many challenges.

The simulation of the real biological nervous system is too complicated: The working principle of the biological neural network has been generally grasped by researchers in many years of research. However, the neural network in the real biological body is too complex, and its structural

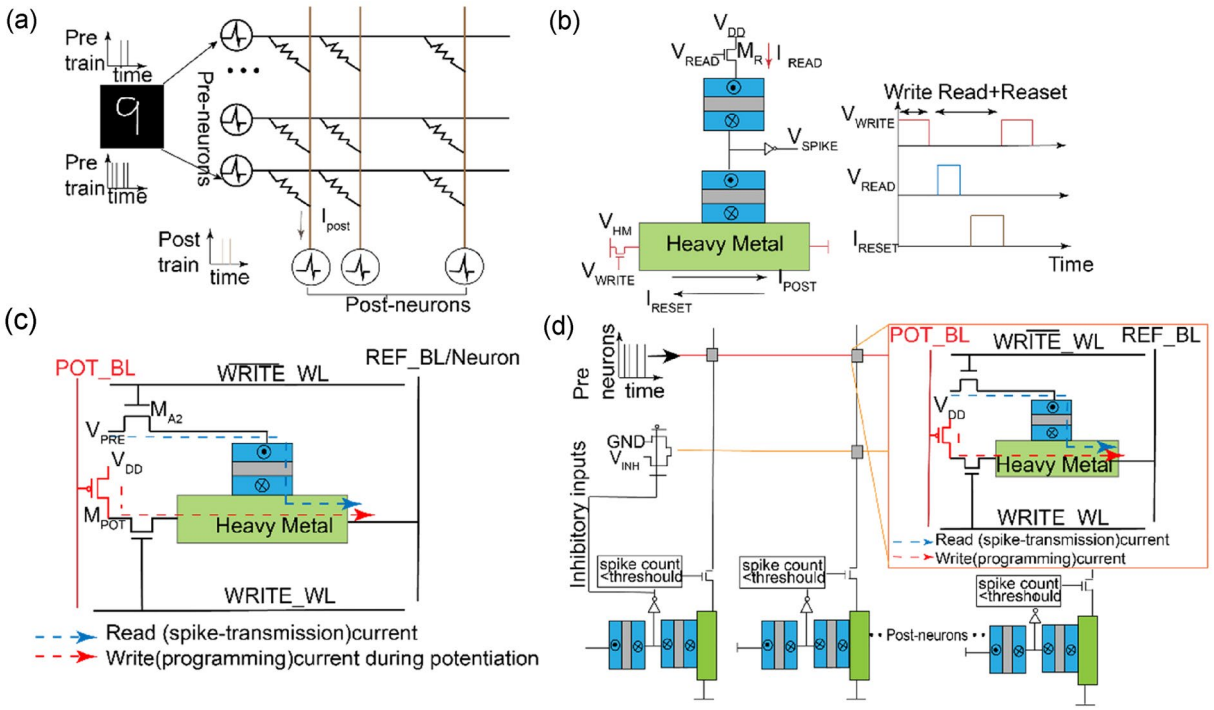

Fig. 13 **a** Schematic of SNN consisting of pre-neurons and post-neurons interconnected by synapses. The input image is encoded into spike trains by the neurons. **b** The left side shows the schematic of the spintronic neuron consisting of a reference MTJ (upside) and a neuron-MTJ (downside), both of which are initialized to the high-resistance state. The right side is the timing diagram illustrating the various operation modes of this MTJ-HM neuron. **c** Schematic of the MTJ-HM synaptic bit cell. **d** The whole architecture of SNN constructed using the spintronic neuron and synapses. The figures are adapted from Ref. [148], with the authors' permission

details are still mysteries. Designing neuromorphic computing systems based on real biological nervous systems is a huge challenge.

It is challenging to apply to practical scenarios: If an artificially designed SNN is used, according to its characteristics, it is generally more suitable for continuous recognition and inference of dynamic scenes. However, in the actual application process, how to make full use of the low power consumption, high-speed and event-driven characteristics of the SNN is complicated. In addition, the application and development of SNN also depend on the development of neural computing chips, because the new structure and computing mode of SNN cannot achieve the theoretical results on traditional chips.

Difficulty in training and learning: For the direct training of SNNs, most of the supervised learning methods are based on gradient settings, lacking biological rationality. Another way to obtain a trained SNN model is to use the trained traditional neural network. Although the transformation is performed directly, it is limited by the loss of precision caused by many aspects. Therefore, compared to the current relatively mature artificial neural network, the training and learning of the SNN on the real large deep network still have a long way to go.

Application accuracy is low on more complex tasks: SNNs have been controversial for a long time, one of the reasons is that their performance in application accuracy is often inferior to traditional AI networks. Both the encoding and training-learning issues mentioned above may lead to an impact on the accuracy of their application on more complex tasks. Therefore, how to improve the application accuracy of the SNN while retaining the original advantages and characteristics of the SNN is also a major challenge for the future development of the SNN.

To sum up, as the third-generation neural network technology, SNN has very prominent features and advantages nevertheless its development is also full of challenges.

4 Stochastic computing

In the previous section, we briefly introduced stochastic computing and its computational unit, p-bit. Based on various connection manners of p-bits, different network structures have been built to solve several kinds of hard computational problems. For example, the use of symmetric connected BMs and Ising machines (IMs) are favored for solving the IF and COP, respectively. On the other hand, BNs with asymmetric connected structures, i.e., directed

connections between p-bits, can be used to perform Bayesian inference. From the perspective of stochastic computing, this section provides an overview of the recent development of the above three networks which consist of MTJ-based p-bits, including the elaboration of the working principle of networks, the hardware implementation process of networks, current challenges as well as future directions.

4.1 Boltzmann machines for invertible logic

Standard binary Boolean logic circuitry and memories utilize stable and deterministic units to represent information. For example, the on and off states of a transistor or the relative magnetization orientations of a non-volatile nanomagnet can be used to represent binary states 0 and 1. This deterministic feature makes computational circuits directional in nature: once a logic gate is fabricated, its input and output ports will be determined, and the circuit is only capable of operating in the input-to-output forward mode. This intrinsic directionality poses a challenge to conventional Von Neumann architecture-based computers in solving some computational tasks, such as NP problems [160]. To address this issue, the invertible logic [17, 84, 161] has been widely studied in recent years. Compared to the deterministic bit used in traditional binary logic circuits, the building block of invertible logic makes use of an unstable and probabilistic unit called p-bit. Logic circuits made from such probabilistic devices possess various novel characteristics: 1) as the entire combinational logic circuits can operate both in forward and reverse modes, the functions of circuits become more diverse, which provides more design space, and 2) hardware costs could be greatly reduced for certain arithmetic computing tasks. For instance, an invertible multiplier can integrate the functions of multiplication, division, and product factorization into a single module, while similar functions require multiple sets of complex multiplier and divider circuits under conventional single-direction circuits. 3) Many hard computational problems like IF [162] and Boolean Satisfiability (SAT) [163] can be solved efficiently with invertible logic.

4.1.1 Boltzmann machine-based invertible logic

BM [164] is a term that often appears in the context of machine learning, which describes a network of stochastic spin glass. Figure 14(a) shows its general structure: 1) each node is fully connected to other nodes in the network, for example, a_1 is connected to all other nodes in the network, and 2) there are interactions between nodes. The strength of interactions is represented by the weight matrix J_{ij}, and each element in J represents the intensity of the coupling between two nodes. Such a bidirectional connection makes a diagonally symmetric weight matrix with all diagonal elements 0.

Invertible logic is exactly based on BMs with bidirectional configuration. Figure 14(b) shows a BM-based invertible AND gate composed of three nodes with the connection defined by matrices J_{AND} and h_{AND}. In this logic, each node or spin is represented by a probabilistic device with bipolar output, namely a p-bit [84]. A generic p-bit is shown in Fig. 14(c), and it exhibits an "S" shape input–output relationship described in Eq. (2).

Equation (2) originates from the physical mechanism of the p-bit itself and Eq. (3) describes the role of the synapse. Note that it is necessary to ensure that the transmission time $t_{synapse}$ is much smaller than the fluctuation time t_{pbit} of p-bits so that the signal produced by the former p-bit can transmit to other latter p-bits before the next change in its state. The extremely long transmission time will cause a failure

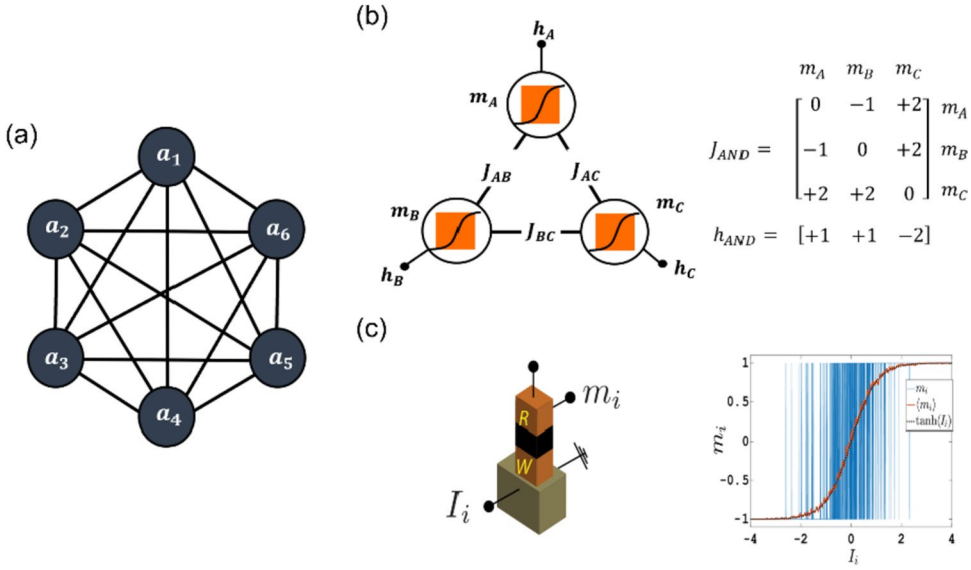

Fig. 14 **a** A graphical representation of an example BM. **b** A BM-based invertible AND gate defined by weight matrix J_{AND} and h_{AND} with three p-bits m_A, m_B and m_C. **c** A generic p-bit structure and its sigmoidal response. Figure(**c**) is adapted from Ref. [84] with the authors' permission

of the system. The energy of invertible logic is represented by Hamiltonian H:

$$H(\{m\}) = -I_0\left(\sum_{i<j}(J_{ij}m_i m_j) + \sum_i h_i m_i\right) \tag{8}$$

At a certain temperature (a given I_0), each p-bit of the network is updated sequentially in every round of iteration. After some operation time, the whole system will be stabilized at the thermal equilibrium state. The statistical result of different spin configurations can be verified using the Boltzmann law:

$$P(\{m\}) = \frac{\exp(-H)}{\sum_{i,j}\exp(-H)} \tag{9}$$

It can be seen intuitively from the above equations that the final probability distribution of the system has nothing to do with the initial values of p-bits but is governed by the energy of the network which is closely related to the spin configurations. These configurations are determined by the biases of p-bits and the coupling strength between p-bit pairs. For example, the invertible AND gate consists of three p-bits, resulting in that there are a total of 2^3 spin configurations, and for each configuration, there is a corresponding energy state. Assume $I_0 = 1$, the table in Fig. 15(a) lists all 8 possibilities. It can be noticed that by encoding appropriate J_{AND} and h_{AND}, spin configurations matched with the truth table of the AND gate filled with green can have the equal and lowest energy, -3. There are three operation modes of invertible AND gate, i.e., free mode, forward mode, and reversed mode. For the free mode shown in Fig. 15(a), none of the nodes of the AND gate is clamped. Therefore, states that accord with the truth table have the highest and almost equal possibilities ~ 25%. By clamping the inputs m_A and m_B, the invertible AND gate can operate in the forward mode. For example, clamp inputs m_A and m_B to 0 and 1, respectively. As a result, $m_C = 0$ has the highest possibility, which can be seen in Fig. 15(b). Fig. 15(c) shows the most striking feature of the invertible AND gate. By clamping the output m_C to 0, there are three possible solutions (A,B,C) = (0,0,0), (0,1,0) and (1,0,0) with almost equal possibilities ~ 33.3%. The statistical results of invertible AND operating in these three modes match with the analytical Boltzmann law calculated by Eq. (8) and Eq. (9). Therefore, to design a well-functioning invertible logic with a specific function, h and J coefficients need to be carefully designed. There is no learning process in BM-based invertible logic. Once h and J are determined, they will no longer change, which is different from BMs for machine learning purposes.

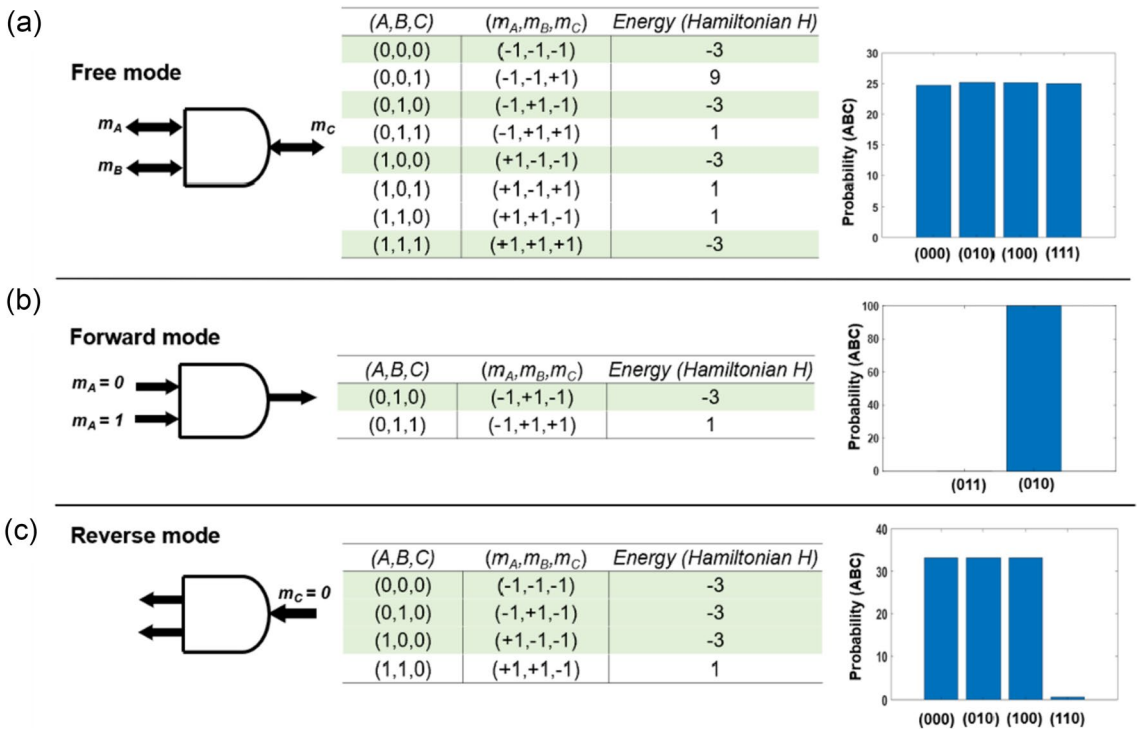

Fig. 15 An example of an invertible AND gate with three operation modes: free mode, forward mode, and reverse mode and corresponding simulation results. **a** Free mode: do not clamp any nodes of the AND gate. **b** Forward mode: clamp inputs m_A and m_B to 0 and 1, respectively. **c** Reverse mode: clamp the output m_C to 0

4.1.2 From small-scale invertible building blocks to large-scale invertible logic

Like the very large-scale integrated circuits, a striking feature of invertible logic is its composability [161, 165, 166]. Any arbitrarily large-scale invertible logic [167], like sin, cos, matrix product, etc., can be obtained by logical synthesis using small invertible networks. Commonly used small building blocks are invertible NOT, invertible AND, invertible OR gates, invertible half adders, and invertible full adders, etc. Complicated networks consisting of these building blocks have been demonstrated to find applications in solving IF [17, 84, 161, 165, 168], SAT [168, 169], training of neural networks [170] and machine learning [171, 172].

The equal footing [173] of every p-bit in invertible logic is the underlying reason for the bidirectional operations of BM-based invertible logic. Consequently, when designing small BM-based invertible logic, careful design of J and h is necessary. As defined in Eq. (8), these two matrices define the energy of the network, thereby determining the final Boltzmann distribution at the thermal equilibrium state. The ground-state spin logic [174, 175] has been proposed for the quantum system, which provides a compact design of h and J for the invertible AND gate. Camsari et al. [84] presented a mathematical transformation approach that can encode the truth table of any logic to the configuration of BMs. However, the shortcoming of this approach is that even for a simple gate, auxiliary p-bits are required, and the transformation process involves a series of matrix operations. Onizawa et al. [176] proposed a general method to design a compact J and h with the minimum number of nodes for small invertible building blocks using Linear Programming (LP). By solving the LP problem using the off-the-shelf LP toolkit for Python [177] or MATLAB, the configurations of small invertible logic can be easily obtained so that a configuration library for all the small invertible logic can be created efficiently. The key idea of such an approach is for any given logic, there is a specific truth table. The target of LP is to map states in the truth table to the lowest system energy so that appropriate J and h could be found. The detailed steps are as follows:

1) First, as shown in Fig. 16(a), convert logical values 1 and 0 to bipolar format, i.e., + 1 and − 1.
2) As shown in Fig. 16(b), set the energy of all states in the truth table equal to E_{min}, while the energy of other non-desirable states is larger than E_{min}.
3) Use LP to maximize d (the difference between the lowest energy level and the second lowest energy level) to obtain appropriate J and h for small invertible logic.

Figure 16(c) shows a configuration library of commonly used small invertible logic using such an LP approach. Note that for the invertible half adder and invertible full adder, both have two possible configurations, namely two J and h choices. The configuration for other invertible combinational logic circuits such as the 3-input/1-output invertible AND gate, invertible multiplier, and ripple carry adder with more complicated energy profiles cannot be directly solved by LP but needs to be constructed from the configuration library. Fig.17(a) shows an example of a composite 3-input/1-output invertible AND gate which is

(a)

A	B	Y	State
0	0	0	Valid
0	0	1	Invalid
0	1	0	Valid
0	1	1	Invalid
1	0	0	Valid
1	0	1	Invalid
1	1	0	Invalid
1	1	1	Valid

m_A	m_B	m_Y	H (Energy)
-1	-1	-1	**E0 = Emin**
-1	-1	1	E1 ≥ Emin + d
-1	1	-1	**E2 = Emin**
-1	1	1	E3 ≥ Emin + d
1	-1	-1	**E4 = Emin**
1	-1	1	E5 ≥ Emin + d
1	1	-1	E6 ≥ Emin + d
1	1	1	**E7 = Emin**

(b)

$$E_k = \begin{cases} -\sum_i h_i m_i - \sum_{i<j} J_{ij} m_i m_j = E_{\min} \\ \quad (f(x_1 \cdots x_p) = (y_1 \cdots y_q)) \\ -\sum_i h_i m_i - \sum_{i<j} J_{ij} m_i m_j \geq E_{\min} + d \\ \quad (\text{otherwise}) \end{cases}$$

(c)

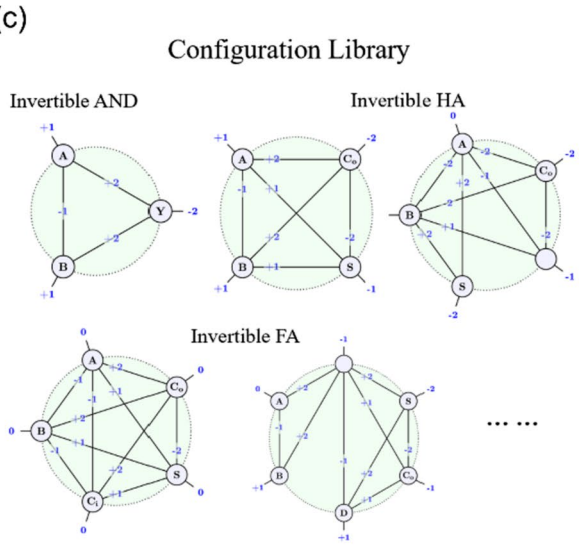

Fig. 16 Steps to design invertible logic. **a** Convert logical values 1 and 0 to bipolar format + 1 and − 1. **b** Map the states that accord with the truth table to the lowest network energy state, other invalid states have higher energy. Then use LP to solve this set of equations. **c** A configuration library of small invertible logic. The figures are adapted from Ref. [176, 178] with the authors' permission

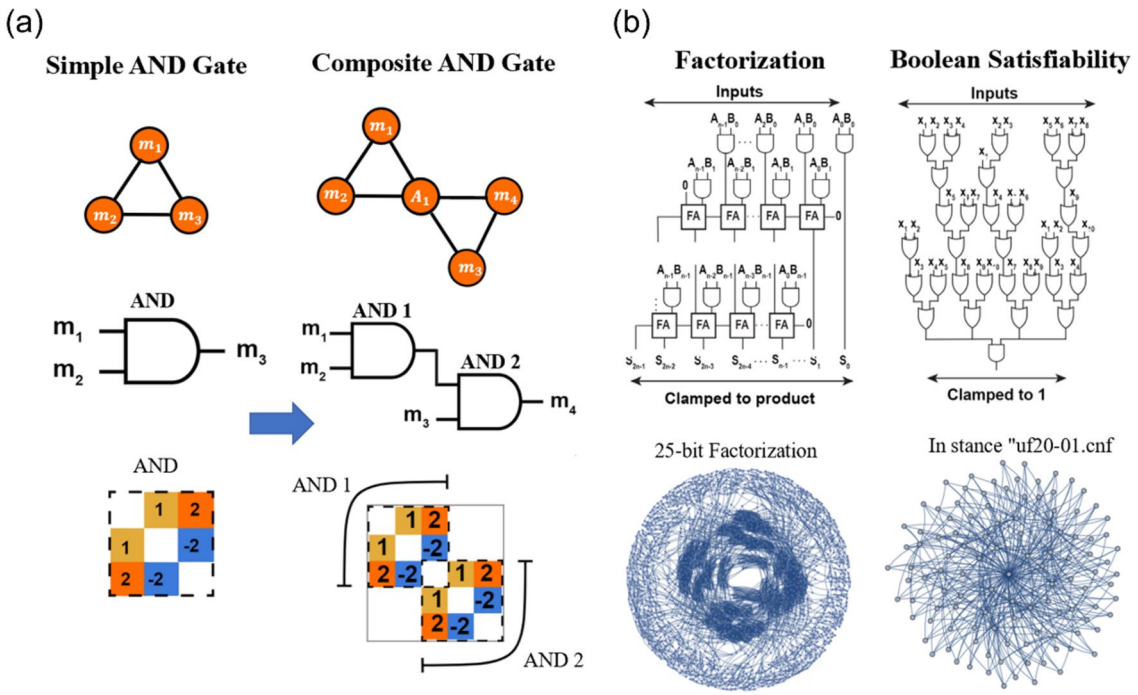

Fig. 17 **a** A composite 3-input/1-output invertible AND gate is composed of two basic 2-input/1-output invertible AND gates. **b** Logic schematics and graph representations of an integer factorizer and an SAT solver based on invertible logic. The figures are adapted from Ref. [165] with the authors' permission

composed of two basic 2-input/1-output invertible AND gates. The key point in the merging process is that the output of the AND1 is the input of the next-level AND2, which means the size of final J and h are not 6×6 and 1×6. In fact, due to the introduction of this common node or called an auxiliary node, the size of both matrices is reduced (5×5 for J, 1×5 for h). The auxiliary node is a bridge connecting the two building blocks. Large-scale invertible logic composed of more basic logic requires more auxiliary bits. As a result, more merging processes are required and such a merging process is time-consuming if done manually [178]. Much attention has to be paid to locate the position of auxiliary bits so that the J and h of smaller modules can be superimposed correctly. Kato et al. [167] proposed an automatic conversion tool from a gate-level netlist to an invertible logic circuit netlist using a standard hardware description language, which enhances the efficiency of merging greatly. Figure 17(b) shows the logic schematic and graph representations of an integer factorizer (or invertible multiplier) and SAT solver circuits designed for IF and SAT, respectively. As we can see**,** they are combinational logic circuits that are also composed of basic logic such as invertible AND gates, invertible half adders, invertible full adders, invertible NOT gates, and invertible OR gates. The graphs depict the connectivity among p-bits in these two large-scale networks.

4.1.3 Stochastic MTJs for implementation of BM-based Invertible logic

The hardware implementation of most BM-based invertible logic circuits is within the framework of stochastic computing using stochastic spintronic devices or probabilistic CMOS-based devices. This section mainly surveys recent progress on stochastic spintronic devices, more precisely, the stochastic MTJs. In most previous studies on nanomagnets, the deterministic property of nanomagnets is utilized. Deterministic nanomagnets normally have high E_B and are resistant to thermal noise. Therefore, they have been widely used in memory devices. The non-volatile MTJs made from nanomagnets with high E_B have binarized resistance states which can be used to store binary information 0 and 1 for more than 10 years. On the other hand, it can be predicted that as the energy barrier decreases, the nanomagnets will be more susceptible to ambient temperature. As a result, the stochasticity of MTJs based on nanomagnets with low E_B ($< 5\ k_BT$) will get stronger. The variables in BMs can be represented by such stochastic MTJ-based p-bits. Moreover, by connecting a certain number of p-bits in terms of J and h described in the previous section, the invertible logic can be constructed. These invertible networks can be used to solve IF and SAT.

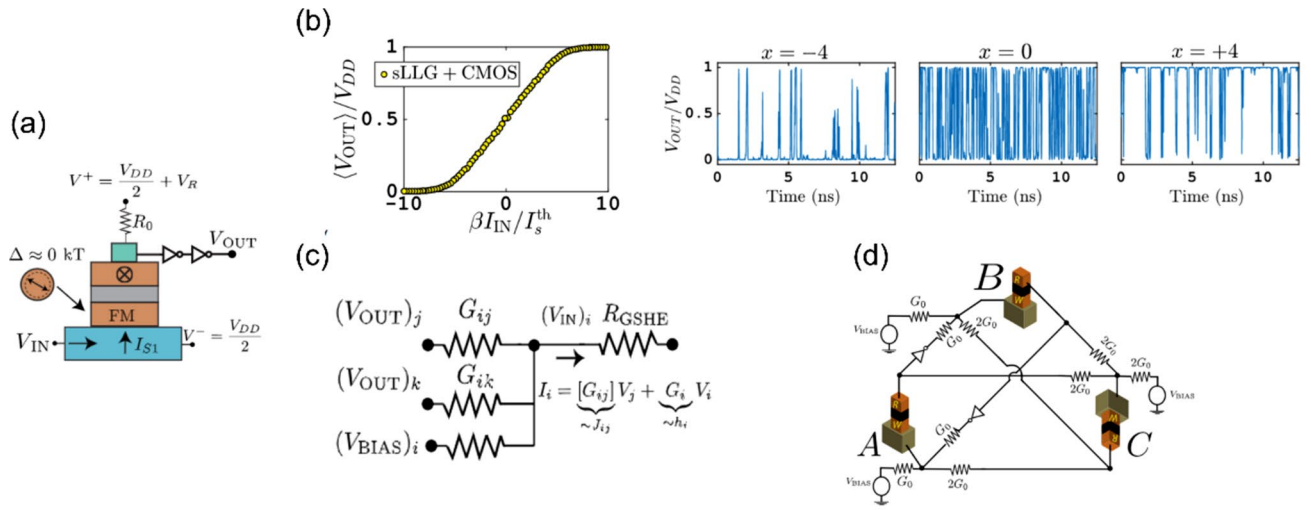

Fig. 18 **a** A possible three-terminal p-bit to serve as the building block of invertible logic. **b** The sigmoidal response of such GSHE-driven p-bit. Three bias points are chosen to illustrate that each point in the yellow fitting curve corresponds to a time-averaged output under certain bias voltages. **c** A passive resistor network is used to represent the interconnection strength among p-bits. For a certain p-bit, it receives feedback currents from all connected p-bits. **d** Circuit schematic of an invertible AND gate. The figures are adapted from Ref. [84] with the authors' permission

Camsari et al. [84] proposed a three-terminal p-bit for implementing invertible logic. As shown in Fig. 18(a), it is composed of a giant spin Hall effect (GSHE)-driven MTJ for tunable random number generation and two inverters for amplification. Note that the nanomagnets used in such MTJ have a circular shape rather than the usual elliptical ones. There is no preferred easy axis in circular nanomagnets. As a result, the shape anisotropy, and the energy barrier of nanomagnets approximately equals 0, which makes the magnetization of MTJ constantly fluctuate. The magnetization of the free layer can be pinned by a spin current generated from an injecting charge current and the probability of pinning can be tuned by the magnitude of the charge current flowing through the GSHE layer. Fig. 18(b) shows the fitted sigmoidal response of such p-bits and the real-time output waveforms under three charge currents with different magnitudes. Every point in the fitted curve corresponds to a time average of the real-time output voltage. This work utilized a passive resistor network to implement the interconnections among p-bits as shown in Fig. 18(c). Fig. 18(d) shows an example of an invertible AND gate built from the proposed p-bits. The conductance G_{ij} and G_i in the resistive network are matched with the discrete values of previously defined matrices J_{AND} and h_{AND}, respectively. The function of inverters is to reverse the direction of the charge current so that the negative values in matrices can be represented.

Faria et al. [179] and Debashis et al. [180] demonstrated that unstable magnets with a fraction of k_BT can be used to implement p-bits through numerical simulation results and experimental evidence, respectively. Based on the GSHE-driven stochastic MTJ, Faria et al. investigated the performance differences between in-plane anisotropy magnet-based p-bit and perpendicular anisotropy magnets-based p-bit. They discovered that the former p-bit design could provide a much faster fluctuation rate and the fluctuation is more telegraphic than nanomagnets with perpendicular anisotropy. Debashis et al. comprehensively studied the design of stochastic nanomagnets to find the most suitable way for implementing p-bits. By comparing three methods, i.e., reducing the anisotropy, reducing the net magnetic moment, or initializing the hard axis, the authors found the scaling of anisotropy provides a more effective way for implementing voltage-controlled p-bit.

The other p-bit design under the MTJ framework is based on the stochastic STT-MTJ. Fig. 19 shows two possible p-bit designs and their relevant applications. As shown in Fig. 19(a), Borders et al. [17] proposed a p-bit consisting of a stochastic MTJ, an NMOS transistor, a comparator, and a resistor. This 1 T-1MTJ structure is very similar to a conventional MRAM cell. The only difference is that in the p-bit design, the free layer is replaced with a nanomagnet with a relatively high E_B of $15k_BT$ rather than the traditional nanomagnet with high E_B used in memory devices. Fig. 19(b) depicts its sigmoidal input–output relationship. In a p-bit network, all p-bits are electrically connected by the synapse module which is composed of a digital-to-analog converter (DAC) and microcontroller. The target function, J and h are programmed inside the microcontroller in advance. As shown in Fig. 19(c), when the system is operating, for a specific p-bit, all digital output voltages $\{V_{OUT}\}$ from other p-bits are collected and converted into an analog voltage $\{V_{IN}\}$. Then $\{V_{IN}\}$ is fed into the gate terminal of the

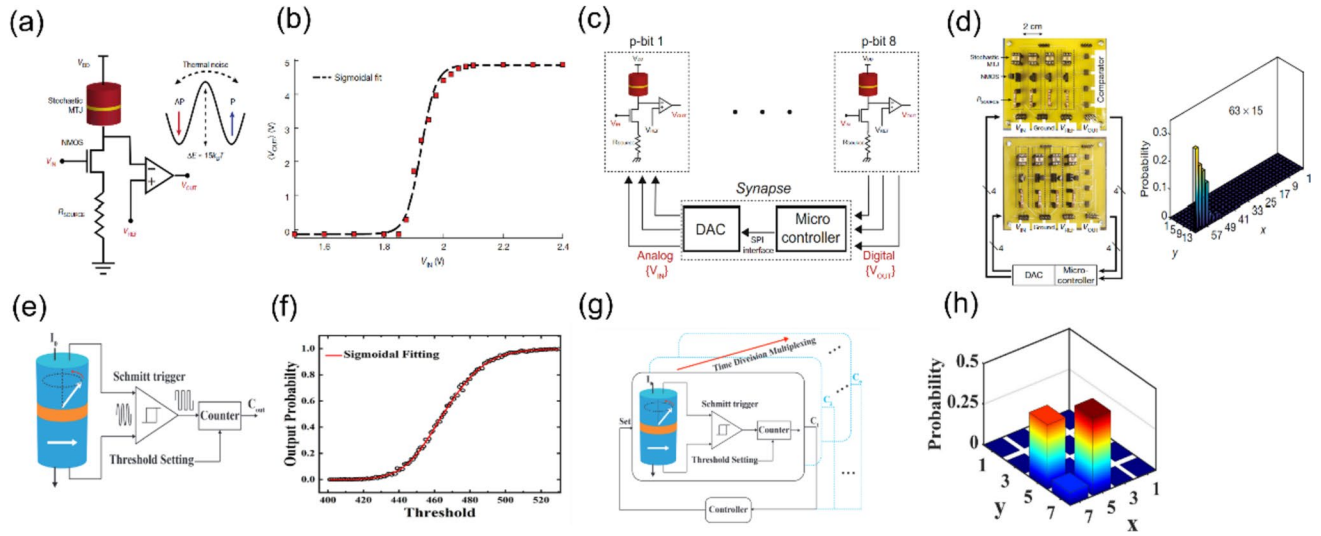

Fig. 19 Two p-bit designs based on stochastic STT-MTJ. **a**–**d**: design 1. **e**–**h**: design 2. **a** Schematic of p-bit 1 which is composed of a stochastic STT-MTJ, an NMOS transistor, a comparator, and a resistor. **b** The sigmoidal response of p-bit 1. **c** Diagram of a general probabilistic circuit based on p-bit 1. **d** A photograph of a printed integer factorizer circuit. **e** Schematic of p-bit 2 which is composed of an STNO, a Schmitt trigger, a counter, and a threshold circuit. **f** The sigmoidal response of p-bit 2. **g** Diagram of a time division multiplexing probabilistic circuit based on p-bit 2. **h** Simulation result of an integer 35. The figures are adapted from Ref. [17, 97] with the authors' permission

NMOS transistor so that the magnitude of charge current and induced STT current flowing through the stochastic MTJ can be controlled by the gate voltage. Therefore, the probabilistic switching characteristic of magnetization is tunable. In this work, a printed circuit board with eight p-bits shown in Fig. 19(d) has been fabricated and an IF task using such an asynchronous network has been demonstrated. The factorization results $945 = 63 \times 15$ can also be observed from Fig. 19(d).

Furthermore, Zhang et al. [97] reported another p-bit design based on STT, but the underlying mechanism of the proposed STNO-based p-bit is different from that of the previous work. In this work, the authors leveraged the anti-damping STT to balance with the Gilbert damping term so that a stable oscillation can be sustained, and its structure is depicted in Fig. 19(e). Due to the existence of thermal noise, the number of oscillations exhibits a Gaussian distribution under a certain sampling time. By setting a threshold to the counter, a digital p-bit exhibiting the sigmoidal response shown in Fig. 19(f) is obtained. Fig. 19(g) shows the schematic of a time division multiplexing circuit, in which different p-bits are coupled using a customized coupling rule. The simulation results for a 6-bit factorization $35 = 5 \times 7$ are shown in Fig. 19(h).

As the size of invertible logic becomes larger and larger, more p-bits are required, and the corresponding circuit implementations will become much more complicated. One possible solution to reduce the number of required p-bits is to utilize the many-body interactions [17, 181]. An example of a 4-bit invertible adder using three-body interactions has been reported in Ref. [181]. It not only reduces the number of required p-bits but also simplifies the energy landscapes of invertible logic by reducing the number of energy levels so that the solution-seeking process can be sped up. This brings benefits to the subsequent simulated annealing algorithm design because in this case, the system becomes intrinsically easier to get into the global minimum energy state due to a simpler energy profile. Besides, parallel annealing has also been demonstrated to solve IF and SAT with a faster convergence rate [165, 169].

Apart from stochastic MTJ-based p-bits, CMOS-based p-bit designs have also been extensively studied. Before the MTJ-based implementation of p-bits, Pervaiz et al. [173] used microcontrollers to emulate p-bits. The sigmoidal electrical response of p-bits is programmed into the microcontrollers. Together with a weighted logic composed of a microcontroller and a DAC, a BM-based 4-bit × 4-bit invertible multiplier and a 4-bit invertible ripple carry adder have been demonstrated. This work takes the first step toward implementing p-bits with nanodevices. Pervaiz et al. [182] presented a generalized tile of weighted p-bits using a field-programmable gate array (FPGA). A comparison between FPGA-based p-bit and MTJ-based p-bit in terms of energy consumption and required transistors number is presented in Ref. [17]. FPGA-based p-bits normally consist of linear feedback shift registers (LFSRs), look-up tables, and digital comparators. Only consider one LFSR, more than a thousand transistors are required and the consumed

energy for generating a random bit is 20fJ, while for MTJ-based 1 T/1MTJ structure p-bit, the number of transistors and the energy consumption per random bit are 4 and 2fJ, respectively.

4.2 Ising machines for combinatorial optimization

COP [183] is closely related to the daily life of human beings and can be found in various real-world applications such as logistics, vehicle routing, human resource allocation, circuit design [184–187], etc. Compared to the knot counting method of the primitive people in ancient times, highly integrated and powerful computers in modern society have provided great convenience for humans to solve various hard computational problems like COP, but despite this, Von Neumann architecture-based conventional computers still show limitations: the increase in problem size is accompanied by the growth in the number of signals need to be processed. As a result, the number of solutions also increases substantially, which leads to an exponential growth in computational complexity. Considering an extreme case, the amount of information that needs to be stored is even much larger than the storage space of the computer when the size of COP is sufficiently large, so the data processing becomes an impossible task.

To address the above issues, many heuristic algorithms have been proposed. Compared with brute force searching methods, such as the resource-consuming exhaustive method, a well-designed heuristic approach can always provide acceptable solutions for COP with higher accuracy and shorter computational time. However, this complicated algorithm requires higher demands on hardware implementation. On the other hand, researchers are committed to finding efficient computational models, among which, the physics-inspired Ising model [188] has entered the vision of researchers and has been researched extensively in recent years due to its simple structure, intuitive mapping, nature-friendly hardware implementation, and great potential in solving the COP efficiently.

4.2.1 Ising model

Ising model is a mathematical model describing the behavior of coupled magnetic spins in ferromagnetic systems [188] shown in Fig. 20(a). It is composed of discrete spins s_i that only can take values + 1(upward state) or − 1(downward state), interactions between pairs of spins J_{ij} and external magnetic field h_i. The total energy of the whole system H, which is determined by the spin configurations, is defined as:

$$H = -\sum_{i<j} J_{ij} s_i s_j - \sum_{i=1}^{N} h_i s_i. \quad (10)$$

The principle of solving COP using Ising model is to map the optimal solutions of a specific problem to the spin configurations with the lowest energy state. The schematic energy profile of the Ising model is shown in Fig. 20(b). To do this, the constraints and objective functions of specific COP need to be translated into the Ising model by programming the interactions among spins and biases. Once this problem is mapped, the system will tend to evolve towards lower energy with an appropriate annealing scheme and the configurations of spin states will update continuously. Finally, it will converge into a stable energy state. For example, if the interaction coefficients J_{ij} between pairs of spins (ignore couplings between spins and applied magnetic fields) are all programmed to + 1, it can be predicted that in order to reduce the energy, the final spin configuration must be parallel, and all spins are in + 1 states. Conversely, real-world COP normally has much more complicated energy profiles. As shown in Fig. 20(b), such an energy profile has a global energy minimum state and multiple local energy minimum states. Note that if there is no external disturbance, during the evolution process towards a lower energy state, the system has a very high probability to enter one of the local energy minimum states and cannot jump out, resulting in a wrong solution.

4.2.2 Ising machine

IMs are the hardware implementations of the Ising model and the carrier to solve COP. Various IMs based on different physical mechanisms have been extensively

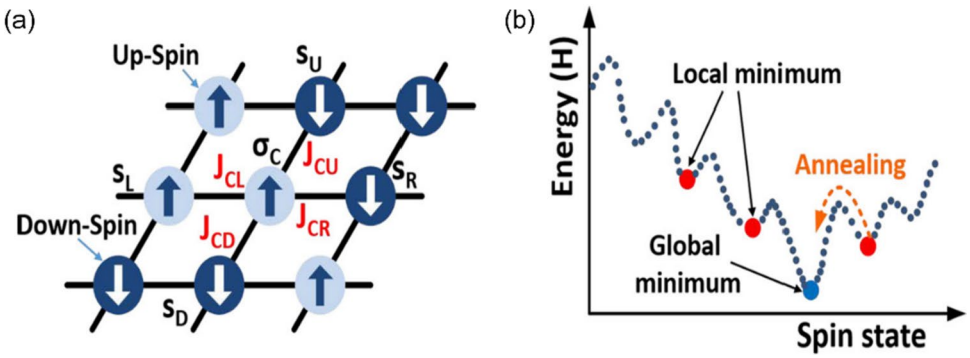

Fig. 20 **a** Ising spin model. **b** Energy profile of an Ising model system. The energy is determined by the spin configurations. The figures are adapted from Ref. [189] with the authors' permission

studied, including Quantum-based [190], Optical-based [191, 192], and Electrical LC-based [193, 194], but these approaches also have their problems. D-wave quantum annealing IMs use qubits to represent spins [190]. Its fatal shortcoming is that a cryogenic cooling system is required to operate normally, which brings high energy consumption and extremely low energy efficiency. Optical-based IMs use coherent light to represent spins and the coupling between spins is implemented by FPGA. Although this system achieves room-temperature operation, it requires kilometer-long optical fibers [191, 192], which poses a great challenge for miniaturization. Electrical LCs can be used as oscillators to implement IMs and the device size is greatly reduced [193, 194]. The binary states of spins are implemented by binarized phases of oscillators and couplings are encoded to the resistive network. The speed of this approach is in the millisecond scale, but still can be accelerated. MTJ-based IMs, more precisely, stochastic MTJ-based and STNO-based Ising solvers, are promising candidates for solving the above issues due to the following desired and unique characteristics: 1) the inherent randomness in stochastic MTJs or STNO enables the solver trapped in local minimum to jump out of these undesired solutions, 2) the energy landscape of the system can be explored on nanosecond timescales because such nanodevices can operate in the GHz frequency range, thereby speeding up the process of solution searching, and 3) various simple hardware implementation choices for couplings between Ising spins including electrical methods and magnetic methods. In this article, we focus on MTJ-based IMs. Typical COP solved by this approach like MAX-CUT, graph coloring problems, and TSP are surveyed. For other hardware implementations of IMs, interested readers can refer to Ref. [195] for comparisons in terms of their standard performance metrics, like the ground-state success probability and time-to-solution.

4.2.3 Stochastic MTJs for Ising model-based combinatorial optimization

4.2.3.1 Simulated annealing-based In recent years, nanomagnets with low E_B have attracted the attention of researchers. The decrease in the energy barrier of nanomagnets makes the inherent randomness brought by the thermal noise more and more severe. MTJs made from such nanomagnets with low E_B are called stochastic MTJs, which exhibit an "S" shape input–output relationship and can be leveraged to make natural annealers.

Using stochastic MTJs to solve COP can be traced back to Ref. [16]. In this proof-of-concept work, the proposed SHE-based stochastic MTJ works in the telegraphic noise region. It utilizes the superparamagnet to serve as the material of the free layer. The energy barrier of such nanomagnets is comparable to k_BT. Each superparamagnetic MTJ is implemented to represent a spin cell in the Ising model and the electrical response of such a cell is shown in Fig. 21(a) and the time-averaged magnetization can be tuned by the injecting current. Due to the controllable stochasticity, a naturally simulated annealing process is enabled. As shown in Fig. 21(b), by continuously increasing the magnitude of the controllable current between pairs of spin cells, a room-temperature implementation of a 16-city TSP has been demonstrated [16]. In this work, the system gradually converges to a local energy minimum state with appropriate simulated annealing. A not ideal but acceptable solution is found by reading the magnetization states of the 4 × 4 SHE-MTJs array. The flipping rate of the superparamagnetic MTJ is on a nanosecond timescale, which enables the system to explore the energy landscape with GHz frequency. As a preliminary work towards natural IMs, this work proves the great potential of superparamagnets in solving COP with high speed and ultralow power consumption, while the drawback of using superparamagnets is also obvious because they are extremely susceptible to process variation, which poses big

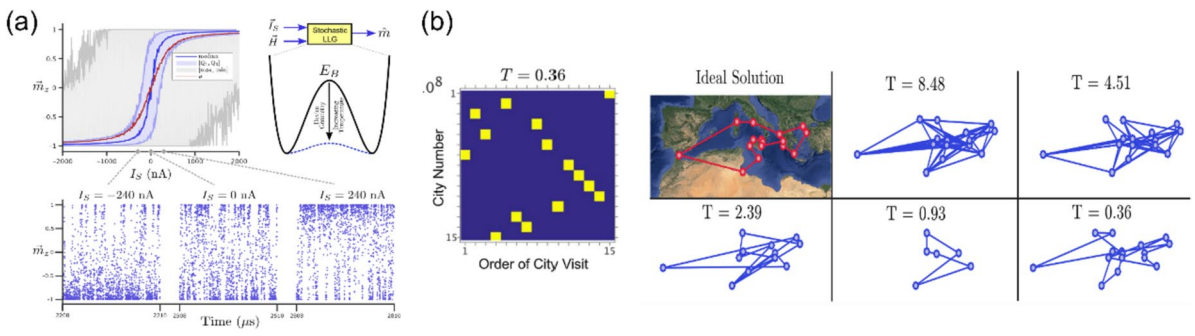

Fig. 21 **a** Sigmoidal response of a stochastic MTJ. Each point in the sigmoidal fitting curve is obtained by the time-averaged magnetization. **b** Through simulated annealing, a 16-city TSP can be solved. The figures are adapted from Ref. [16] with the authors' permission

challenges to device fabrication and reading circuit design. The reading circuits need to be designed appropriately to minimize the reading currents. Therefore, the possible pinning effect on the free layer arising from the reading current can also be minimized.

The above issues limit the application prospects of MTJ in solving COP. To mitigate this, Shim et al. [189, 196] still used a SHE-driven MTJ to work as the Ising cell but replaced its superparamagnetic free layer with a higher-energy-barrier one, and its structure is shown in Fig. 22(a). The thermal noise-induced stochasticity is still applied to serve as the entropy source. In this work, the authors used the charge current injected into the heavy-metal layer underlying the MTJ to manipulate the in-plane magnetic anisotropy. As shown in Fig. 22(b), for a specific MTJ, the majority vote function is implemented with multiple current sources and switches. Based on the nearest neighbor, the amount of charge current injecting, namely the results of votes, to a specific cell is determined by the states of surrounding spin cells. More votes correspond to a higher switching probability of magnetization. Therefore, the next spin configuration of MTJs is determined by the vote results. Fig. 22(c) shows the well-designed schematic of an Ising spin. Reference resistor R_{REF} serves as a voltage divider and cooperates with the output inverter to binarize the voltage levels. For the majority vote function part, CMOS logic gates together with series of transistors are used.

With the implementation of functions "Annealing" and "Majority Vote", several classical COP have been demonstrated. Fig. 23(a) demonstrates the solution process of a MAX-CUT problem which aims at finding two mutually exclusive subsets of spins by connecting edges to maximize the summation of weights along the edges. It can be noticed that from point (c) to (d), there is an abrupt energy drop. This phenomenon implies that the system evolves from a local energy minimum state to a global energy minimum state with an appropriate simulated annealing algorithm. This annealing process is realized by increasing the magnitude of coupling currents between spins. In other words, annealing means the behavior of stochastic MTJs transits from a stochastic manner to a more deterministic manner. Another example is a very famous NP-complete problem called graph coloring which is described as: is it possible to assign m-colors for n-vertices so that two adjacent vertices have the same color? For this problem, a total number of $m \times n$ spins are required to represent the spin configuration for the problem. Similarly, the target of an n-city TSP is to find the shortest possible route to visit each city exactly once and returns to the origin city. The list of cities and distances between each pair of cities are known. A total of $n \times n$ spins are required to solve such a problem. The simulation results for these two typical hard computational problems are shown in Fig. 23 (b) and (c), respectively. For the graph coloring problem, the author fixed the number of colors to 3, but

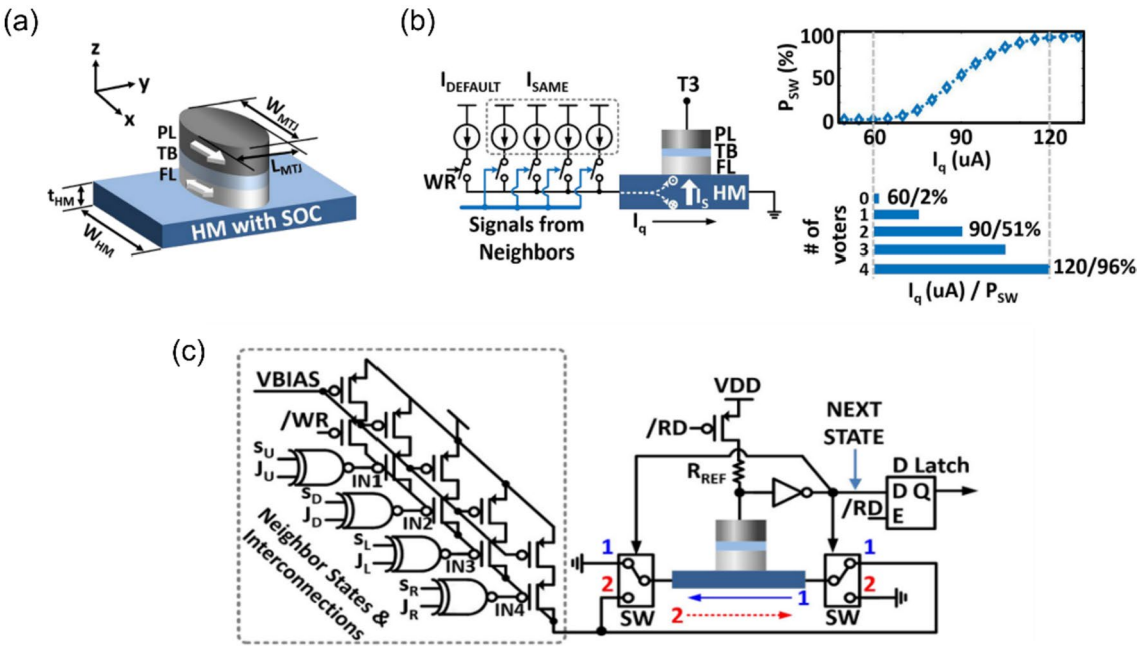

Fig. 22 **a** Three-terminal stochastic SHE-MTJ-based Ising cell. **b** Majority vote function implemented with multiple current copy branches. The switching probability of a selected MTJ is mapped to the amount of injecting charge current. **c** Detailed circuit design diagram of an Ising cell. The figures are adapted from Ref. [189, 196] with the authors' permission

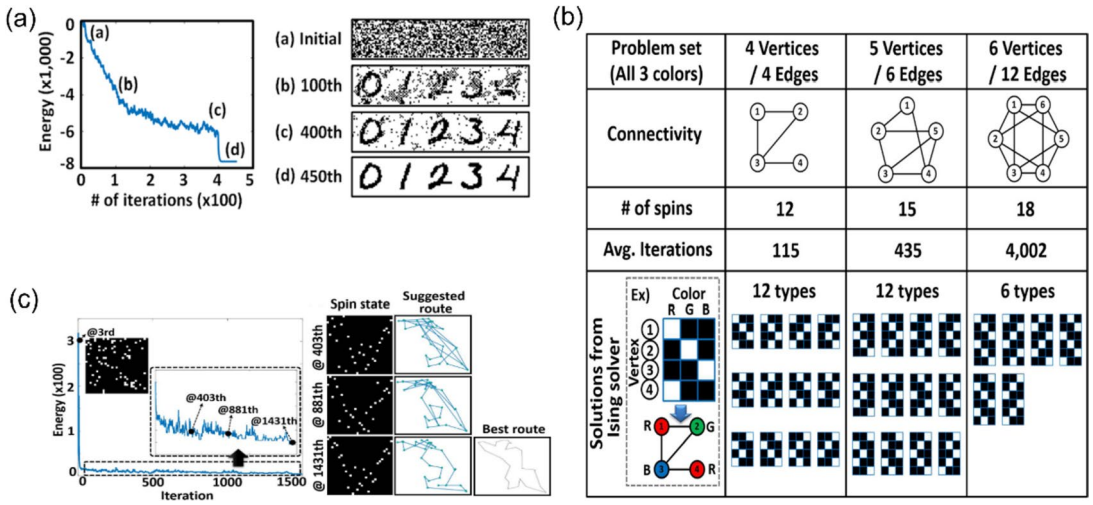

Fig. 23 Three NP problems can be solved by the SHE-MTJ-based Ising machine. **a** MAX-CUT. With appropriate interaction design, the system evolves toward the lower energy state and with enough iterations (450th here), a solution with the lowest energy is obtained. **b** Graph coloring problems. A total number of $m \times n$ spins are required to for a problem with m-colors for n-vertices. **c** TSP. The spin configurations are constantly updating during the iteration process and finally a route is suggested. The figures are adapted from Ref. [189, 196] with the authors' permission

the number of vertices is up to 6. For each subproblem, the author runs a thousand times to obtain an average iteration number when the system reaches an energy minimum state. For a 29-city TSP, the system with a natural annealing process converges quickly after 1431 rounds of iteration. Even though the final suggested route for the salesman to travel is not the optimal but it is reasonable and acceptable.

Apart from using charge currents to implement interactions, a voltage-based method has also been studied. Sharmin et al. [197] proposed a voltage-controlled Ising cell that utilizes the magnetoelectric effect of the multiferroics to minimize the current flowing through the network. This voltage-controlled IM can mitigate the scalability issue with the increasing size of the problem. Fig. 24(a) shows the structure of the proposed voltage-controlled nanodevice and its sigmoidal electrical response. This proposed Ising cell is based on a multiferroic Oxide/CoFeB heterostructure with manipulation of magnetization states + 1 and − 1 through the voltage drop. Fig. 24(b) shows how it is coupled with other cells. The output the former Ising cell is directly cascaded to the input terminal of the latter Ising cell. All other cells are coupled in this way. As a result, there is no need to amplify the output signal using additional CMOS amplifier circuits, but the complicated structure puts forward higher requirements for subsequent fabrication processes.

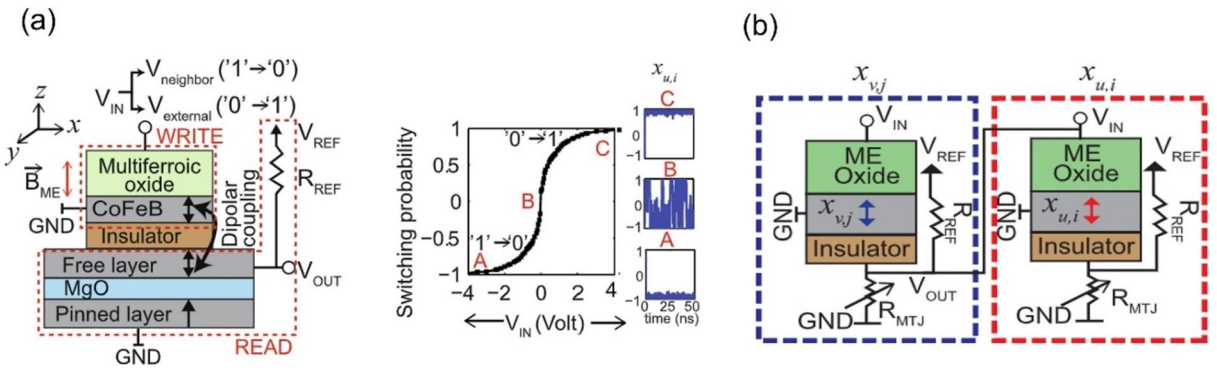

Fig. 24 **a** Structure of an Ising cell based on a voltage-controlled nanodevice and its electrical response. The input voltage, namely the write process, is applied across the multiferroic oxide/CoFeB heterostructure in which the magnetization state of the CoFeB layer can be manipulated to represent the + 1 and − 1 states. **b** Implementation of couplings among Ising cells. The figures are adapted from Ref. [197] with the authors' permission

In addition to advances in simulations, Safranski et al. [93] and Hayakawa et al. [94] provided experimental evidence of nanosecond scale fluctuations for stochastic-MTJ with low E_B, which marks a milestone towards hardware implementation of the Ising model. Furthermore, Borders et al. [17] experimentally demonstrated the solving of an IF problem with STT-based stochastic MTJs serving as the hardware spin cells. They fabricated a printed circuit board with eight stochastic MTJ-based p-bits which are interconnected through a microcontroller and a DAC; a factorization result of an integer up to 945 has been demonstrated. IF can be categorized as an NP-intermediate problem, and the solution principle of this kind of problem is similar to COP. Therefore, this work is a good reference for the subsequent use of stochastic MTJs with low E_B to solve large-scale COP.

4.2.3.2 Parallel annealing-based Parallel annealing [198] is the other annealing algorithm that has the potential in faster minimizing the system energy to obtain optimal or near-optimal solutions. Different from simulated annealing, parallel annealing makes use of a set of replicas of the p-bit network with well-designed discrete fixed temperature levels rather than only one network with decreasing temperature levels. As shown in Fig. 25(a), the higher temperate replica suffers more severe thermal fluctuations, thus it can explore a larger range of spin configurations. This is reflected in the fact that state 1 has a larger fluctuation energy range. Meanwhile, the lower temperate replica is also fluctuating but in a small range. Once the high-temperature replica finds a spin configuration whose energy is lower than the low-temperate one, the spin states between them will exchange.

Grimaldi et al. [169] applied parallel annealing to a stochastic MTJ-based Ising model. Firstly, based on the macrospin model, Fig. 25(b) shows the real-time magnetization fluctuations of a stochastic MTJ with thermal noise comparable to the energy barrier. The solving process of a MAX-3SAT with 70 variables and 700 clauses is demonstrated using 771 p-bits with parallel annealing. The simulation results are shown in Fig. 25(c). Four replicas with

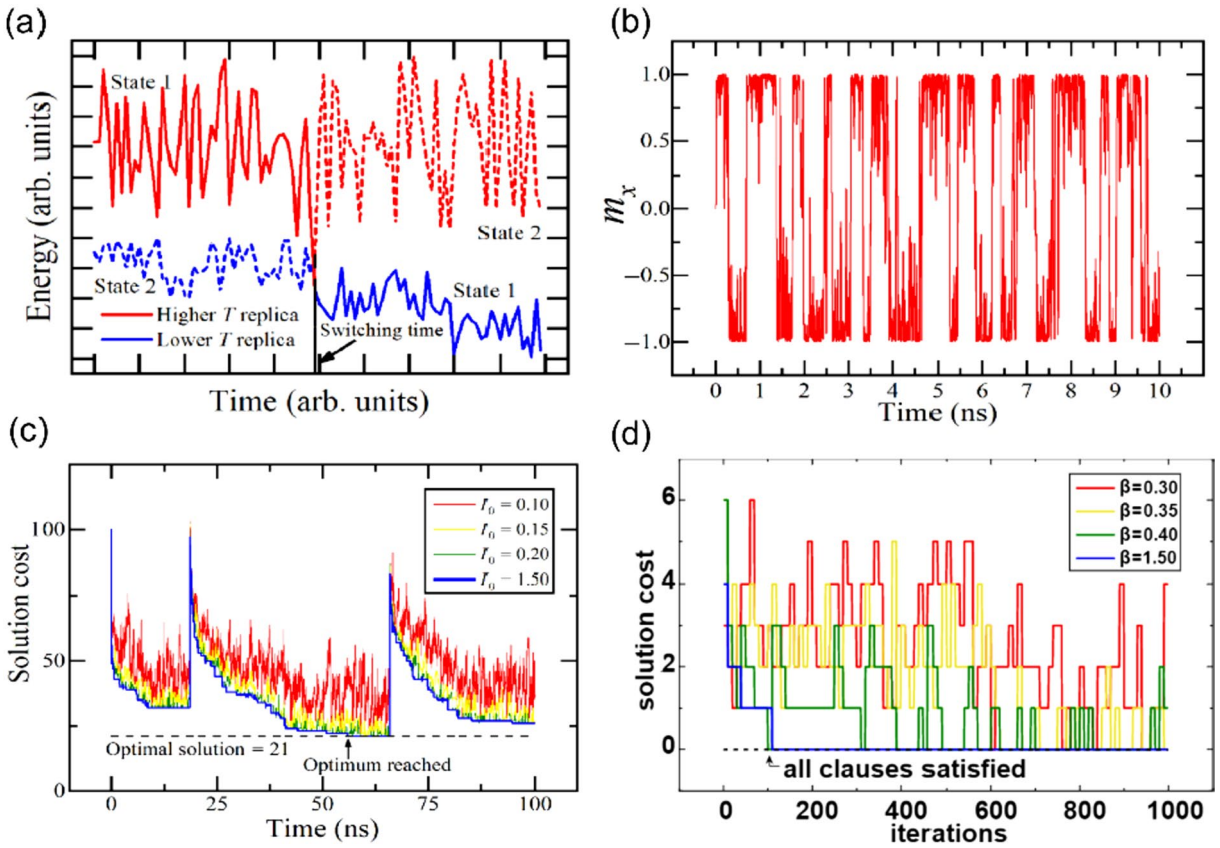

Fig. 25 **a** Illustration of parallel tempering with two replicas at low- and high-temperature levels. **b** Real-time magnetization state of a stochastic MTJ. **c** Simulation results of a MAX-3SAT with 70 variables and 700 clauses using parallel annealing. Four replicas with the pseudo-temperature I_0=0.10, 0.15, 0.20, and 1.50 are used. **d** Simulation results of an SAT "uf20-01.cnf" using parallel annealing. Four replicas with the pseudo-temperature β=0.30, 0.35, 0.40, and 1.50 are used The figures are adapted from Ref. [165, 169] with the authors' permission

four different temperate levels are adopted and the solver obtains the optimal solution within 60 ns. This ultrafast solution-finding process is enabled by the parallel updating of p-bits which is different from the sequentially updating of p-bits in the simulated annealing scheme. Aadit et al. [165] also confirmed the superiority of parallel annealing over simulated annealing in solving IF and SAT. The authors solved the same SAT "uf20-01.cnf" using these two annealing schemes. Figure 25(d) shows the optimal solution is achieved only after 100 iterations by a 4-replica parallel annealing. Although the parallel annealing algorithm can converge faster to find the solution for solving factorization and COP, as discussed above, multiple replicas cost more computational resources.

4.2.4 Spin torque nano-oscillators for Ising model-based combinatorial optimization

The other kind of IMs that can be used to solve COP utilize a completely different mechanism of the MTJ, namely the oscillation of the magnetization. The free layer of such MTJ is made of nanomagnet with a high energy barrier, which mitigates the aforementioned issues for MTJ state reading and difficulties in device fabrication. Nevertheless, compared to the stochastic MTJ, which directly uses the relative orientation of magnetization to the reference layer to intuitively represent the upward + 1 and downward -1 states of spins, the STNO-based Ising cell requires an additional phase binarization step.

4.2.4.1 Phase binarization by injection locking The phase binarization of the oscillator is usually achieved by subharmonic injection locking. As shown in Fig. 26, in the coupled oscillator network, a perturbation signal with a frequency of $2f_{inj}$ (f_{inj} is comparable to f_{SHNO}) is injected into each oscillator which has a natural frequency of f_{SHNO}. Then, the frequency of each oscillator will change from f_{SHNO} to f_{inj} with two stable phase-locked states. After this step, the oscillator no longer has an analog phase but stabilizes at some discrete phase points, that is 0 and π, and thus this bistate can be used to represent the two states of the spin. For an oscillator in a coupled oscillator network, in addition to external signal perturbations, it is also perturbed by the oscillators connected to it. The dynamics of its phase change with time can be accurately captured by the Kuramoto model [199]:

$$\frac{d}{dt}\phi_i(t) = -K \cdot \sum_{j=1, i\neq j}^{n} J_{ij} \cdot \sin\left(\phi_i(t) - \phi_j(t)\right) - K_s \cdot \sin\left(2\phi_i(t)\right) \tag{11}$$

where $\{\phi_i\}$ represents the phase of the i^{th} oscillator and $\{J_{ij}\}$ represents the coupling between oscillator i and oscillator j. The global parameter K adjusts the overall coupling strength between oscillators. K_s modulates the coupling strength from external perturbation. It can be observed that a perturbation signal with a frequency of $\omega_1 = 2\pi f_1$ introduces a coupling term with a period of π (i.e., $\sin(2\varphi)$) to the phase dynamics, while the period of the coupling term from other oscillators is 2π.

To see the evolution from the energy aspect, there is an energy-like Lyapunov Function associated with Eq. (11):

$$\begin{aligned} E\left(\vec{\phi}(t)\right) = &- K \cdot \sum_{i,j,i\neq j} J_{ij} \cdot \cos\left(\phi_i(t) - \phi_j(t)\right) \\ &- K_s \cdot \sum_{i,j,i\neq j} J_{ij} \cdot \cos\left(2\phi_i(t)\right) \end{aligned} \tag{12}$$

A global Lyapunov function is a quantity like the energy term in the Ising system. When $\{\phi_i\}$ settle at these discrete points, that is, ϕ_i(t) is either = 0 or = π, the last term of Eq. (12) is a constant offset term (cos 0 or cos 2π = 1). Ignore this constant term, the mapping relationship between the Lyapunov function and Hamiltonian is:

$$E\left(\vec{\phi}(t)\right) = -K \cdot \sum_{i,j,i\neq j} J_{ij} \cdot \cos\left(\phi_i(t) - \phi_j(t)\right) = -2K \cdot \sum_{i,j,i<j} J_{ij} \cdot s_i s_j. \tag{13}$$

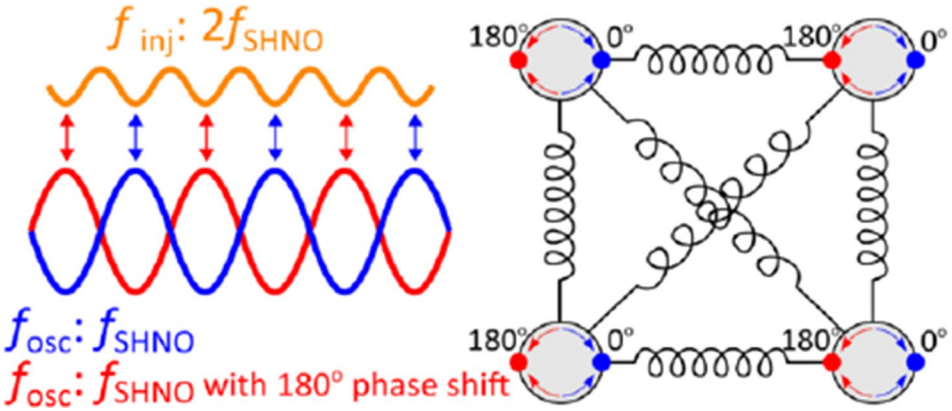

Fig. 26 Illustration of subharmonic injection locking. Coupled oscillators are stabilized at one of the binary phases under subharmonic injection locking. The figures are adapted from Ref [193] with the authors' permission

If $K = 0.5$ is chosen, the global Lyapunov function in Eq. (13) exactly matches the Ising Hamiltonian at these discrete phases. It implies that just like the configurations of spins keep flipping towards lower system energy, oscillators will interact with each other, and their phases are continuously changing to minimize the energy and finally settle into stable phases so that the solution to the mapped problem can be obtained. Detailed derivation for the dynamics of an oscillator under subharmonic injection locking from a single oscillator to coupled oscillator network is presented in [193].

4.2.4.2 Spin torque nano-oscillators-based Ising machines Albertsson et al. [200] demonstrated the feasibility and superiority of implementing IMs using MTJ-based STNOs shown in Fig. 27(a) using a numerical simulation model: the solution-searching speed for specific MAX-CUT problems can be accelerated to the order of ns and the nanoscale size of Ising cell provides a solution for the miniaturization of IMs. When using the developed numerical model for oscillator network simulation, there is no need to define the type of coupling. Although this makes the model more general, it also implies the impossibility to use the model to study the impact of various coupling designs on the system's performance. Moreover, this work implemented the annealing process by modulating the coupling strength between the external perturbation signal and spins, but it does not introduce the phase noise term. Therefore, the system is more likely to fall into these undesired local minima states when dealing with COP with complex energy profiles. This conclusion is confirmed by the simulation results shown in Fig. 27(b) when solving a MAX-CUT problem. The increase in the problem size significantly reduces the success probability of the IMs to obtain the optimal solution and the near-optimal solution.

McGoldrick et al. [201] developed a general analytical framework that not only can capture the dynamics of injection locking for STNOs shown in Fig. 27(c) with large oscillation angles but also models the phase noise using the impulse sensitivity function approach [202]. Previously, models in Ref. [203] can only explain injection locking with a small procession angle. Same to the treatment when dealing with the thermal fluctuations in Ref. [204], the thermal noise is treated as an effective field in STNO. Moreover, the authors emulated the fundamental features of the oscillators required for IMs and analyzed the performance of STNO networks at the circuit level. Results show that to solve the same 100-size MAX-CUT problem, the STNO scheme can achieve several orders of improvements in solution time and energy efficiency. Furthermore, due to proper modeling of phase noise existing at room temperature, as shown in Fig. 27(d), solver operating at 300 K has higher

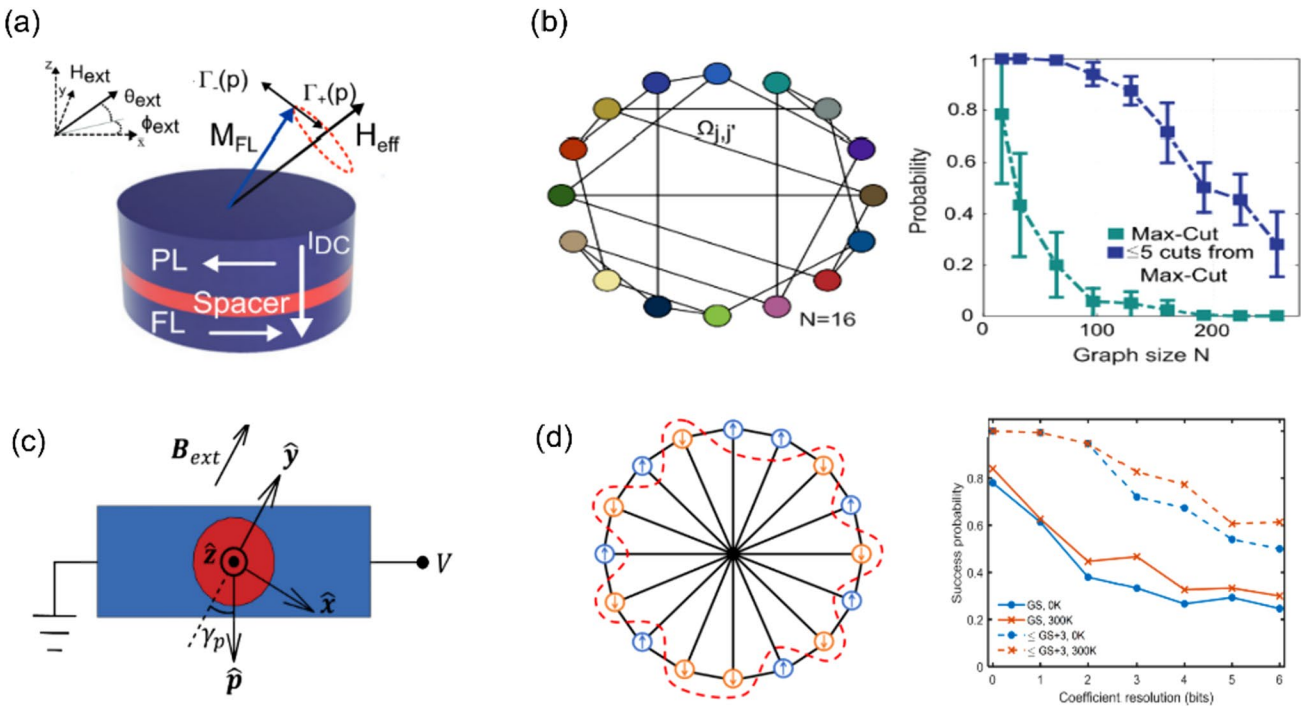

Fig. 27 **a** Illustration of an STNO. **b** As the size of the MAX-CUT problem increases, the success probability of obtaining the solution (green) and a sub-optimal solution (purple) decreases substantially when no phase noise exists. **c** Top view of 3-terminal SHNO. **d** With thermal noise, the success probability of obtaining the solution can be improved greatly. The figures are adapted from Ref. [200, 204] with the authors' permission

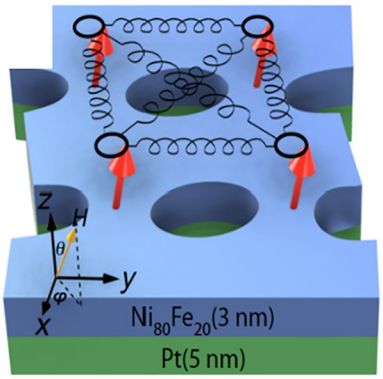

Fig. 28 A 2×2 phase-binarized STNO array. The figure is adapted from Ref. [200] with the authors' permission

success probability than the one working at 0 K Besides, it is found that the success probability of solutions is expected to scale with the array size of the coupled oscillator network as shown in Fig. 27(d).

From an experimental perspective, a 2×2 phase-binarized SHNO array has been experimentally demonstrated to solve a MAX-CUT problem [205], and an 8×8 SHNO 2D array has been fabricated for neuromorphic computing application [132]. The schematic presentation of SHNO arrays are shown in Fig. 28.

At present, the problems solved by SHNO-based IMs are relatively simple, only involving the MAX-CUT problem and the problem size is small. For other problems that require a more complicated coupling design, more theoretical and experimental verifications are needed. On the other hand, common coupling mechanisms between SHNOs and STNOs include electrical coupling, spin-wave, direct exchange, and dipolar. The coupling mechanism between spins shown in Fig. 28 is difficult to be programmed, especially for weighted coupling strength. Normally, a general-purpose IM requires an all-to-all coupling, which requires a careful design of the couplings. These experimental realizations of such an SHNO network represent a significant milestone toward SHNO-based IMs. How to encode coupling strength in an easier way, such as how to realize electrical coupling between oscillators in experiments, still needs follow-up research.

4.3 Bayesian networks for bayesian inference

BN [206] is a directed probabilistic graphical model that has been widely applied to understand the causal dependencies [207] among events. It aims to efficiently solve common but hard computational probabilistic tasks in real life, such as a series of problems with inherent causality represented by medical treatment decisions [208] and weather forecasting [209]. In a BN, random variables, the directionality, and strength of dependencies among random variables are represented by nodes, edges, and a set of conditional probability tables (CPTs), respectively. Nodes can be divided into parent and child nodes in terms of their causal sequences inherited from events. Edges map such parent-to-child directionality, while CPTs encode the strength of such dependencies. The implementation of Bayesian inference [210] in BNs, specifically, the process of deriving the posterior probability based on the prior probability and the likelihood function (derived from the probability model) requires substantial floating-point representations and operations. It makes conventional computing paradigm-based computers encounter a bottleneck in the pursuit of efficient inference with low resource consumption and fast computing speed. Moreover, as the size of BN grows, the dependencies between parent nodes and child nodes become more complicated. Therefore, the computational complexity increases greatly, and the calculation of conditional probabilities becomes intractable. To solve the above issues, researchers are committed to finding nature-friendly devices and circuits that can represent random variables and perform associated probability operations.

4.3.1 A classical four-variable Bayesian network

BN is a fusion of probability theory and graph theory [211]. It uses the language of graph theory to reveal the structure of the problem intuitively while it uses the principles of probability theory to solve inference and learning problems according to problem structure. Such a combination can be seen in a classical BN with 4 random variables [206]. The four variables represent four random events—whether the weather is cloudy "C", whether the weather is rainy "R", whether the sprinkler is on "S" and whether the grass is wet "W". By decomposing joint probabilities into a series of simple modules, the computing difficulty can be reduced in BN. The causality between nodes can be obtained from the direction of edges, and CPT describes the dependencies between parent and child nodes in the format of conditional probability.

Based on BNs, assuming that the grass has been observed to be wet (observed evidence), Bayesian inferences can be implemented. In this case, there are two hidden causes: the sprinkler is on, or it is raining. The posterior probability can be estimated using Bayes' rule as defined:

$$P(W) = \frac{P(S)P(S)}{P(W)} \tag{14}$$

where $P(S)$ is prior probability, that is, a judgment on the probability of the event "sprinkler is on" before the

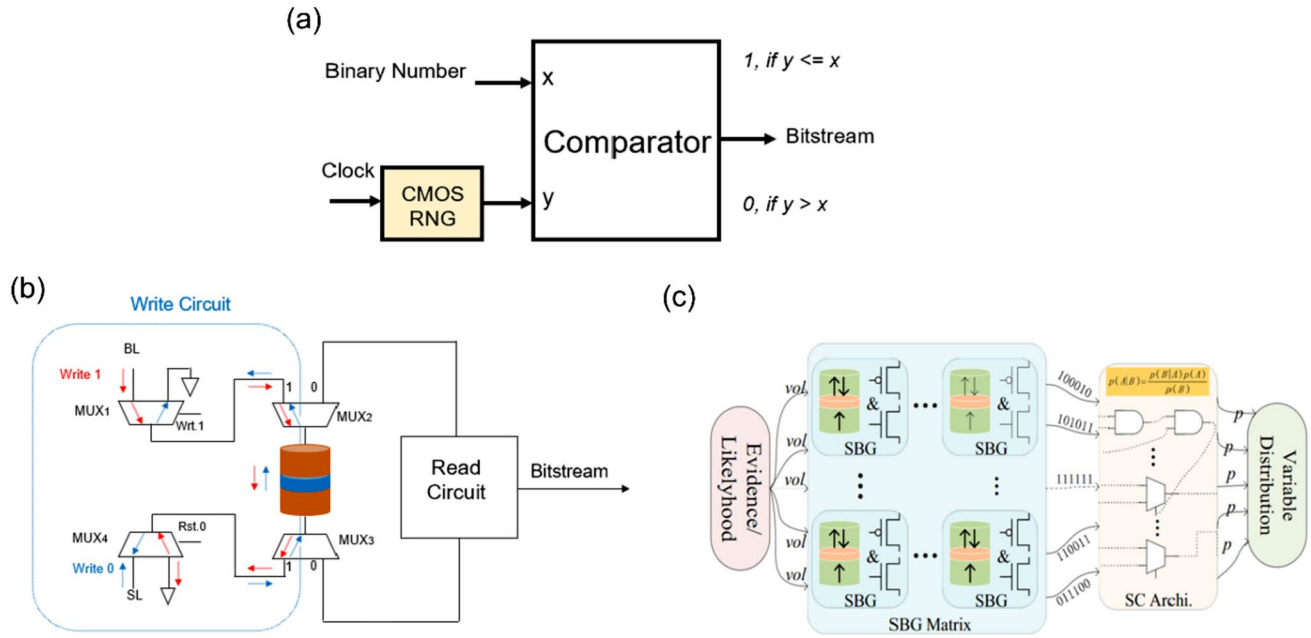

Fig. 29 **a** Diagram of the traditional SBG using an RNG and a comparator. **b** Stochastic MTJ-based SBG composed of MTJ and Multiplexers. By controlling the write and read processes, random bitstreams are generated. **c** Stochastic MTJ-based system for Bayesian inference. The three core modules of this system are Evidence/ Likelihood, SBG matrix and the SC Architecture. The figures are adapted from Ref. [223, 224] with the authors' permission

occurrence of the event "grass is wet", *P(S|W)* is the posterior probability, that is, the reassessment of the probability of the event "sprinkler is on" after the occurrence of the event "grass is wet", *P(W|S)/P(S)* is called likelihood function, which is an adjustment factor.

4.3.2 Stochastic MTJs for RNGs in Bayesian networks

Various schemes have been developed for direct hardware implementation of BNs. Under the CMOS-based framework, there are stochastic digital circuits with digital logic gates [212–214] and analog probabilistic computing circuits with stochastic devices [215–217]. A comprehensive overview of related CMOS-based BN hardware implementation can be found in Ref. [218], which pays more attention to the improvements in circuit implementation, architecture design, and algorithm optimizations. Moreover, emerging nanodevices, especially the stochastic MTJs, provide a compact and low hardware-cost solution to replace the core elements for randomness generation in BNs due to their unique stochastic feature. Fig. 29(a) shows a diagram of the traditional stochastic bitstream generator (SBG) in which the random bitstreams are generated utilizing two core modules: RNGs and comparators. In most of the previous work, LFSRs [219] are popular to function as RNGs. After the comparison process, random bitstreams with 50% ratios of 0 s and 1 s are generated, while this approach consumes a mass of transistors and the bitstreams generated are pseudo-random. The former problem brings a lot of area and energy consumption, while the latter implies that the computing accuracy using BNs will be degraded due to the correlation among bits in a bitstream. In Ref. [220–222], TRNG circuits using stochastic MTJs have been proposed, which utilize the inherent randomness of nanomagnets. Note that these stochastic MTJ-based RNG modules need to be cooperated with peripheral CMOS circuits to function as an SBG. Although the above designs have great potential in replacing LFSRs that require up to thousands of transistors, the tunable randomness nature of stochastic MTJs, i.e., the tunable output ratios of 1 s and 0 s, is not fully exploited. Also, the potential of SBGs in solving practical problems such as Bayesian inference is not reflected in these works.

The novel MTJ-based circuit proposed by Ref. [223, 224] can solve the above issues and demonstrates its application in data fusion and BN with higher speed and low hardware cost, in which the authors make full use of the "S" shape relationship between the input pulse voltage level and the switching probability of the stochastic MTJ. The schematic of the proposed SBG circuit, illustrated in Fig. 29(b), is composed of a write circuit and a read circuit. The writing part includes two stages: resetting to AP state and switching from AP to P state. 1 MTJ and 4 multiplexers (MUX) are used. The basic workflow and principle are as follows: 1) set *Write En* of MUX2 and MUX3 to 1 so that the circuit works in a write operation, 2) set *Rst.0* to 1 and *Wrt.1* to 0 to make sure the reset stage is on and the current direction is shown as

the blue arrow, and 3) contrary to previous step setting, set *Rst.0* to 0 and *Wrt.1* to 1 to make sure the circuit work in the switching phase. At this time, the current direction is shown as the red arrow, flowing from the free layer to the pinned layer. The probability of switching from AP to P state can be customized in terms of the occurrence probability of a specific event, which can be achieved by configuring the amplitude or duration of the applied voltage pulse. Such circuits can be directly used to work as an SBG to generate random bitstreams. Furthermore, such SBG matrix-based Bayesian inference systems can be employed by circuit designers to reduce power overhead and accelerate inference speed.

However, strictly speaking, the above-mentioned approaches are still under the domain of CMOS circuits as these random nanomagnet-based RNGs or SBGs cannot be manipulated individually in BN. As a result, they are not able to be directly used as stochastic nodes to represent stochastic variables as well as reflect the dependencies among nodes. Subsequent complicated CMOS arithmetic circuits shown in Fig. 29(c) are required with such a stochastic computing paradigm to implement the inference task. The magnitude of input voltage is proportional to the likelihood. The SBG matrix translates the input voltages to stochastic bitstreams. The stochastic computing architecture implements the inference process based on the input data processed by the SBG matrix. This module is constructed by multiple logic gates. In this review paper, we focus on the nanomagnets-based beyond-CMOS stochastic devices, expanding only using the random characteristics of nanomagnets as RNGs or SBGs to building blocks that can directly build BNs. These nanomagnet-based units can be independently engineered through underlying physical mechanisms to directly represent stochastic variables and couplings among them. We give descriptions of several different schemes for representing random variables, edges and CPTs, as well as their respective advantages and drawbacks.

4.3.3 Stochastic MTJs for direct implementation of Bayesian networks

Behin-Aein et al. [225] presented a proof-of-concept hardware implementation of a 3-variable BN using experimentally benchmarked models of nanomagnets. In this work, each stochastic variable is represented by a stochastic MTJ-based p-bit shown in Fig. 30(a). The CPT reflecting the dependencies between nodes corresponds to the ensemble average of the magnetization orientations after the initialization and relaxation process of the nanomagnets. In this preliminary work, the authors demonstrated the structure of a carrot-stick-performance BN shown in Fig. 30(b). Such a simple 3-node BN is often used for employee incentive policies in enterprises. Although details about how to translate real-world problems like this to such nanomagnet-based building blocks are not given, the authors make predictions about the underlying physical mechanisms that could potentially be exploited to directly implement BN, including how to write, how to read states of variables and how to implement interactions.

Faria et al. [14] reported a design framework for implementing BN in hardware, including 1) how to translate real-world BN to a behavioral model PSL with a set of mapping rules and associated formula expressions, and 2) how to map PSL to electronic circuit elements. Table 1 reports all the information on translating a graphical model of BN to real electronic elements in which PSL is a bridge between BN and real circuits, defined by the h and J coefficients. In PSL, the binarized magnetization orientations of nanomagnets represent the random variable, which corresponds to high and low levels binarized by the inverter in electronic elements. The conditional dependencies between random variables represented by CPT can be explained by Eq. 15(a) and 15 (b): there is a child node m_i, and its output is governed by input I_i by Eq. 15(a). Then, 15(b) describes I_i is obtained by summing the weighted states of each node connected to m_i and its own bias. Given the precondition of $m_j = 1$ (by applying sufficiently large current or voltage), the probability of $m_i = 1$ can be calculated from the above two equations. The direction from the parent node to the child node represented by edges can be implemented by the directivity in real circuits when connecting the input and output terminals of p-bits. Figure 30(c) shows the circuit of a building block, i.e., a p-bit in this work, which is composed of a stochastic MTJ, an operational amplifier, two inverters, and resistive elements. By sampling its time-averaged output, the occurrence probability of a particular event can be readily obtained. Please refer to Eqs. 16(a, b) and Eqs. 17(a–e) for the mapping from PSL to circuits. A set of translation process examples of zero-parent nodes, one-parent nodes, and two-parent nodes from BN to PSL to the circuit are also shown in Fig. 30(d).

Based on the inherent randomness of stochastic nanomagnets, Debashis et al. [226] proposed another novel p-bit design which is made of stochastic MTJs with perpendicular anisotropy and worked as the building block of the network. Figure 30(e) shows the ring-like structure of the proposed stochastic device which utilizes the GSHE originating from the SOC in heavy metal rather than spin transfer torque in previous work. Due to the design differences of underlying physical mechanisms, the translating process is slightly different from the previous case. More specifically, although both two pieces of work change the weights and local bias by changing the conductance or voltage, the subsequent current change in Ref. [14] directly leads to a strong change in the STT effect. However, in Ref. [226], the reversal of magnetization is indirectly affected by changing the magnetic

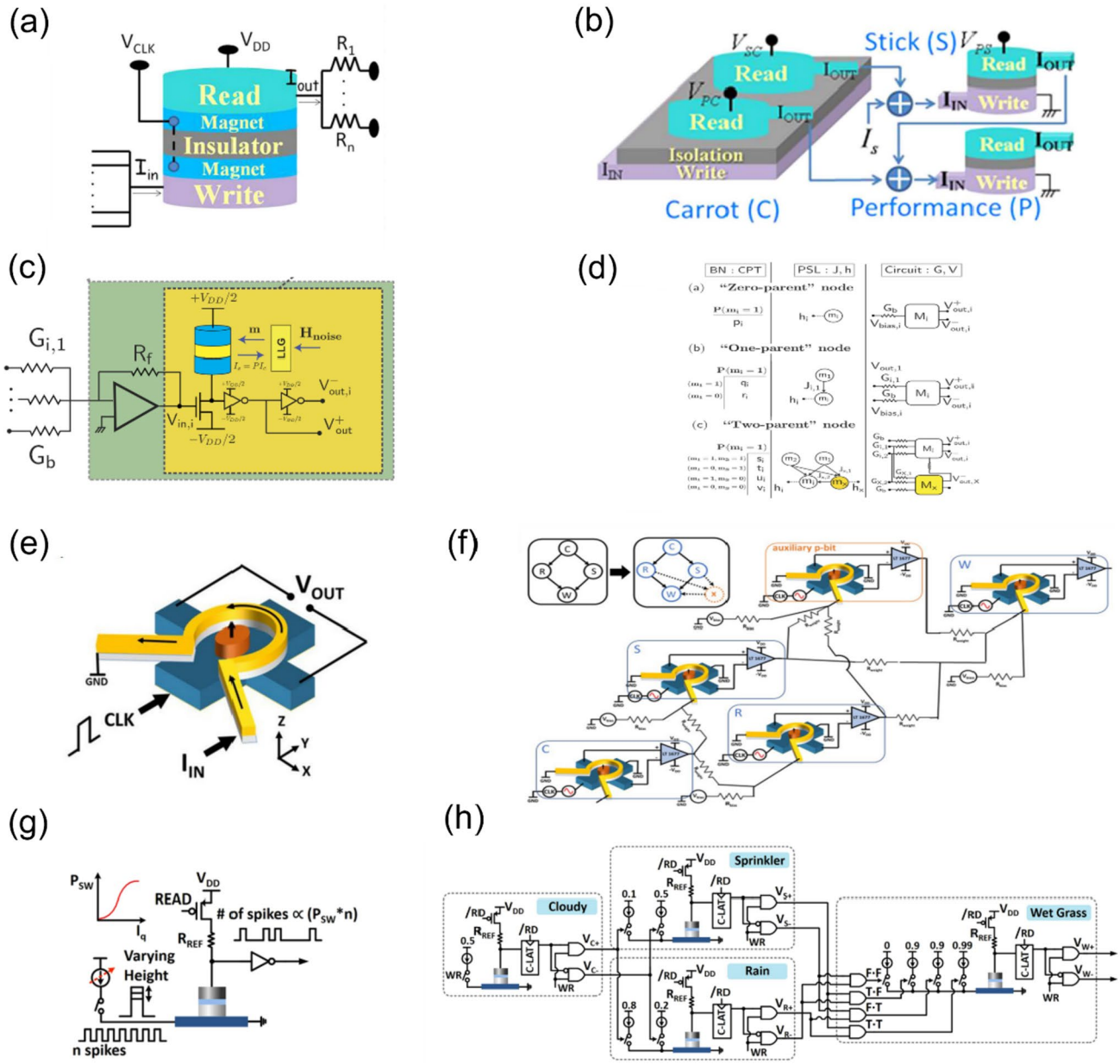

Fig. 30 Various p-bit designs based on stochastic-MTJs for direct implementation of BN. **a**–**b**: design 1. **c**–**d**: design 2. **e**–**f**: design 3. **g**–**h**: design 4. **a** A proof-of-concept p-bit based on a stochastic MTJ to represent a stochastic variable in BN. **b** A carrot-stick-performance BN based on design 1. **c** Schematic for p-bit design 2, which is composed of a stochastic STT-MTJ, an operational amplifier, two inverters, and resistive elements. **d** Examples of the translation process from BN to PSL to electronic elements based on design 2. **e** Schematic of p-bit design 3 based on an Oersted-ring assisted stochastic MTJ. **f** Circuit schematic of a 4-node BN using design 3. **g** Schematic for p-bit design 4 based on stochastic SHE-MTJ. **h** Circuit schematic of a 4-node BN using design 4. The figures are adapted from Ref. [14, 225–227] with the authors' permission

field generated from the injected current flowing through the Oersted ring. Figure 30(f) shows an experimental demonstration of a four-variable BN, in which stochastic devices are initialized and sampled by a pulse sequence, and all of them are electrically interconnected. Any given correlations can be captured by designing the weights and biases.

Shim et al. [227] developed a SOT-driven stochastic MTJ along with peripheral CMOS circuits to act as the building block, the variable of Bayesian inference engine. As shown in Fig. 30(g), a voltage pulse sequence with a fixed phase difference is injected into the input of the building block. Due to the "S" shape switching probability with respect to the input current, the probabilistic information is encoded

Table 1 Translation process of a graphical model of BN to real electronic elements

BN	From BN graphical model to PSL		From PSL to circuits	
Variable	Binarized magnetization orientations of nanomagnet		Binarized voltage levels	
CPT	$m_i(t+\Delta t) = \mathrm{sgn}(rand(-1,1) + \tanh I_i(t))$	(15a)	$V_{out,i} = \frac{V_{DD}}{2}\mathrm{sgn}\left(rand\ (-1,1) + \tanh\frac{V_{in,i}}{V_0}\right)$	(16a)
	$I_i(t) = I_0\left(h_i(t) + \sum_j J_{ij}m_j\right)$	(15b)	$V_{in,i} = V_{bias,i}G_bR_f + \sum_j V_{out,i}G_{ij}R_f$ (16b)	(16b)
	Mapping rules : $m_i = V_{out,i}/\frac{VDD}{2}$, $I_i = V_{in,i}/V_0$, $h_i = V_{bias,i}/\frac{VDD}{2}$, $J_{ij} = G_{ij}/G_b$, $I_0 = G_bR_fV_{DD}/2V_0$			17(a)–17(e)
Edges	Physical connection direction:			
Parent node → child node	Connect the output of parent node to the input of child node			

into the system and following pulse-based arithmetic will implement the inference. Fig. 30(h) shows the complete circuit schematic of a typical four-node BN case and demonstrates its efficiency in solving such inference tasks by using a direct mapping approach.

Furthermore, Zand et al. [228, 229] realized a model of a deep belief network using stochastic MTJ-based p-bits. The developed inference simulator is based the restricted Boltzmann machines and can be trained to recognize handwritten digits.

It should be noted that when using the above p-bits as building blocks to build BN, the update sequence of p-bits needs to be appropriately designed to make sure the network operates correctly. In the previous context of BM-based invertible logic, all p-bits have an equal footing. Therefore, the final statistical results, i.e., the Boltzmann distribution, presented at thermal equilibrium will not be affected even if the p-bits are randomly updated in each round of iteration. In BN, on the other hand, the story is different. The inherent causal relationship between events implies that the status of p-bits representing parent nodes and child nodes are different. As a result, it is necessary to ensure BN is updated sequentially and in order from the parent node to the child node. Otherwise, the circuit will not operate appropriately. Faria et al. [230] systematically studied the underlying causes and influencing parameters of such phenomena. A design criterion for designing autonomous and asynchronous BN circuits without any clocks or sequencers based on the p-bit of 1 T-1MTJ adopted in Ref. [14] is expounded. The signal transmission delay on synapse-like interconnection elements must be much smaller than the p-bit response time (the sum of the retention time and flipping time of the stochastic MTJ). In Ref. [226, 227], the sequential update order from the parent node to a child node is guaranteed by the timing of pulses injected into p-bits. The other main contribution of this work is that a behavioral model called parallel PSL for autonomous BN is developed, which provides a valuable reference to future clockless BN design based on emerging nanodevices.

4.4 Challenges and future directions for stochastic computing

In this section, we review recent research progress on stochastic nanodevices, especially the stochastic MTJ which can be used as a p-bit (a building block) to build complicated networks, in several typical applications of stochastic computing. Compared to CMOS-based implementations, stochastic MTJs are favored by researchers due to their inherent stochasticity and other desirable properties. These nature-friendly properties and good

compatibility with the CMOS process make stochastic MTJs one of the most promising candidates to directly implement probabilistic networks based on the stochastic computing paradigm. For example, in BM-based Invertible logic and IMs, the tunable stochasticity of stochastic MTJs makes it possible to implement the natural annealing process. Therefore, the hardware overhead of additionally implementing the annealing algorithm can be reduced. In BNs, stochastic MTJs can directly represent stochastic variables and conditional dependencies between events can also be easily mapped. Furthermore, stochastic MTJs provide a compact and low hardware cost solution which consumes much less area and power in stochastic bitstreams generation compared with CMOS-based SBG. However, there still exist several challenges ahead for direct hardware implementation of probabilistic networks using stochastic MTJ-based p-bit devices.

On the device-circuit level, although low barrier nanomagnets enable much faster flips of states and hence, the process of network exploring all the spin configurations can be accelerated, stochastic MTJs with low E_B not only pose difficulties to the fabrication process but also the circuit design, because a very limited energy barrier results in a very small critical current [231]. Therefore, the current flowing through the read circuit needs to be designed to be as small as possible to minimize its negative pinning effect on the switching of the free layer. On the other hand, nanomagnets are vulnerable to process variation, which implies even tiny differences in energy barrier may bring great change to device stochasticity. This undesirable change in stochasticity can be overcome for small-scale networks, but with the increase in network size, scalability emerges as the most important issue, because these negative effects can be accumulated and may eventually lead to a specific stochastic MTJ being unable to operate at its originally designed operating point, thus leading to malfunction of the whole network.

On the algorism level, with an increase in the problem size, a design scheme that can simplify the coefficients for BM-based invertible logic and IM is imperative to be developed, which is reflected in a more reasonable design of h and J when mapping the problem ready to be solved to the p-bits-based probabilistic network so that simpler energy profile, fewer p-bits required and reduced computational complexity are obtained. Recently, a design scheme based on many-body interactions has been demonstrated to be one of the promising solutions to solve the above issues [17, 181]. However, how to directly implement the multiple couplings among p-bits based on STT-MTJs, SOT-MTJs or oscillation-based MTJs requires further research. Furthermore, for large-size problems, the accuracy of the solution can be improved by annealing algorithms, but most of the current works adopt the simplest linear simulated annealing schedule. Other more advanced annealing schedules, like the design of a universal algorithm that can automatically adjust the annealing speed according to the energy profile of the networks, still need further discussion.

5 Conclusion

We review the MTJ-based neural networks of neuromorphic computing and several typical applications of MTJ-based stochastic computing. First, the fundamentals and research progress of MTJ-based neurons, synapses, and p-bits at the device level are introduced. The magnetization of the MTJ can be regulated by STT, SOT, or VCMA, and thus the resistance changes due to the presence of TMR. The STT-MTJ was first considered as an artificial synapse, and its switching probability can be adjusted from 0 to 100% by the amplitude of the pulse at a fixed pulse width, which enables it to implement the STDP rule. However, SOT-MTJ exhibits better stability and lower energy consumption. For an artificial neuron, what it needs to achieve is to accumulate charge and fire when the voltage reaches the threshold. MTJs exhibit an extraordinary ability to mimic this integrated and fire process. In addition, MTJ-based neurons have nonlinear dynamics, giving them excellent biological proximity. For p-bits, the naturally stochastic fluctuation of MTJs makes them one of the promising candidates. VCMA-MTJs with stochasticity has been implemented as TRNGs and p-bits.

In the section about neuromorphic computing, the neural networks based on MTJs are reviewed. There are three generation neural networks: The first generation is the single perceptron whereas it has only one layer of functional neurons, which makes it unable to solve non-linearly separable problems. The addition of one or more hidden layers between the input and output layers led to the birth of the second generation of neural networks and the concept of MLP. Facing the dramatic increase in parameters caused by the increase in the number of layers, CNN was proposed. MTJ-based CNNs are used to identify datasets such as MNIST, CIFAR-10, and ImageNet with high accuracy and low power consumption. In addition, to solve the sequence problem, RNNs are proposed. Building RNN and RC with MTJs is feasible and STNO exhibits high energy efficiency. The third generation is the event-driven SNN. Compared to the previous two generations, SNNs are closer to the nervous system in the human brain. Meanwhile, their high computing efficiency and low energy consumption highlight their excellent potential for efficient information processing. With the characteristics of non-volatility and high energy efficiency, MTJs can realize learning rules such as STDP and models such as LIF, which perfectly meet the requirements of SNN

for devices. Nonetheless, like all nascent technologies, the development of SNN is controversial and SNN still faces problems such as difficulties in practical use, difficulties in training and learning and low accuracy in complex tasks.

In addition, we review stochastic computing with the MTJs used as p-bits. The stochasticity and other desirable properties exhibited by the stochastic MTJ and its compatibility with CMOS technology make it a promising candidate for the p-bit. In application examples such as BMs, BN, etc., MTJ exhibits highly energy-efficient, compact, and low-cost solutions. Nevertheless, MTJ-based stochastic computing also faces some problems. The switching speed and energy barrier of MTJ are a pair of contradictory indicators. To achieve fast switching, a small energy barrier is required. Nevertheless, a small energy barrier will lead to a small critical current, which brings difficulties to circuit design. Affected by the manufacturing process, it is difficult for the MTJs in the network to have a completely uniform energy barrier, and a slight difference may have a huge impact on the final results. In addition, the mechanism of coupling between p-bits based on STT-MTJ, SOT-MTJ, and VCMA-MTJ is still unclear. Implementing more complex annealing schedules or even a general annealing algorithm also needs to be further investigated.

Acknowledgements This work at the ShanghaiTech University is supported by National Key R&D Program of China (Grant No. 2022YFB4401700), Shanghai Sailing Program (Grant No. 20YF1430400) and NSFC (Grant No. 12104301). This work at the National University of Singapore is supported by MOE-2017- T2-2-114, MOE-2019-T2-2-215, and FRC-A-8000194-01-00.

Data availability statement The datasets generated during and analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors declare no conflict of interest.

References

1. D. Monroe, Neuromorphic computing gets ready for the (really) big time. Commun. ACM **57**(6), 13–15 (2014). https://doi.org/10.1145/2601069
2. J. Han, M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design. IEEE Eur. Test Symp. (ETS) (2013). https://doi.org/10.1109/ETS.2013.6569370
3. C. Mead, Neuromorphic electronic systems. Proc. IEEE **78**(10), 1629–1636 (1990). https://doi.org/10.1109/5.58356
4. M. Mishra and M. Srivastava 2014 "A view of Artificial Neural Network," in 2014 International Conference on Advances in Engineering & Technology Research (ICAETR - 2014). Doi: https://doi.org/10.1109/ICAETR.2014.7012785.
5. F. Rossi, B. Conan-Guez, Functional multi-layer perceptron: a non-linear tool for functional data analysis. Neural Netw. **18**(1), 45–60 (2005). https://doi.org/10.1016/j.neunet.2004.07.001
6. V. Sze, Y.-H. Chen, T.-J. Yang, J.S. Emer, Efficient processing of deep neural networks: a tutorial and survey. Proc. IEEE **105**(12), 2295–2329 (2017). https://doi.org/10.1109/JPROC.2017.2761740
7. S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1–6. doi: https://doi.org/10.1109/ICEngTechnol.2017.8308186
8. T. Mikolov, S. Kombrink, L. Burget, J. Černocký, S. Khudanpur, Extensions of recurrent neural network language model. IEEE Int. Conf. Acoustics Speech Sig. Process. (2011). https://doi.org/10.1109/ICASSP.2011.5947611
9. S. Ghosh-Dastidar, H. Adeli, Spiking neural networks. Int. J. Neural Syst. **19**(04), 295–308 (2009). https://doi.org/10.1142/S0129065709002002
10. N. Caporale, Y. Dan, Spike timing-dependent plasticity: a Hebbian learning rule. Annu. Rev. Neurosci. **31**(1), 25–46 (2008)
11. K.Y. Camsari, S. Salahuddin, S. Datta, Implementing p-bits With Embedded MTJ. IEEE Electron Device Lett. **38**(12), 1767–1770 (2017). https://doi.org/10.1109/LED.2017.2768321
12. D.H. Ackley, G.E. Hinton, T.J. Sejnowski, A learning algorithm for boltzmann machines. Cogn. Sci. **9**(1), 147–169 (1985). https://doi.org/10.1016/S0364-0213(85)80012-4
13. N. Zhang, S. Ding, J. Zhang, Y. Xue, An overview on Restricted Boltzmann Machines. Neurocomputing **275**, 1186–1199 (2018). https://doi.org/10.1016/j.neucom.2017.09.065
14. R. Faria, K.Y. Camsari, S. Datta, Implementing Bayesian networks with embedded stochastic MRAM. AIP Adv. **8**(4), 045101 (2018). https://doi.org/10.1063/1.5021332
15. T. Albash, D.A. Lidar, Adiabatic quantum computation. Rev. Mod. Phys **90**(1), 015002 (2018). https://doi.org/10.1103/RevModPhys.90.015002
16. B. Sutton, K.Y. Camsari, B. Behin-Aein, S. Datta, Intrinsic optimization using stochastic nanomagnets. Sci. Rep. (2017). https://doi.org/10.1038/srep44370
17. W.A. Borders, A.Z. Pervaiz, S. Fukami, K.Y. Camsari, H. Ohno, S. Datta, Integer factorization using stochastic magnetic tunnel junctions. Nature **573**(7774), 393 (2019). https://doi.org/10.1038/s41586-019-1557-9
18. J. Grollier, D. Querlioz, K.Y. Camsari, K. Everschor-Sitte, S. Fukami, M.D. Stiles, Neuromorphic spintronics. Nat. Electron **3**(7), 360 (2020). https://doi.org/10.1038/s41928-019-0360-9
19. Z. Li, S. Zhang, Magnetization dynamics with a spin-transfer torque. Phys. Rev. B **68**(2), 024404 (2003). https://doi.org/10.1103/PhysRevB.68.024404
20. R. Ramaswamy, J.M. Lee, K. Cai, H. Yang, Recent advances in spin-orbit torques: Moving towards device applications. Appl.

Phys. Rev **5**(3), 031107 (2018). https://doi.org/10.1063/1.5041793
21. F. Mahfouzi, R. Mishra, P.-H. Chang, H. Yang, N. Kioussis, Microscopic origin of spin-orbit torque in ferromagnetic heterostructures: A first-principles approach. Phys. Rev. B **101**(6), 060405 (2020). https://doi.org/10.1103/PhysRevB.101.060405
22. G. Vignale, Ten years of spin hall effect. J. Supercond. Nov. Magn. **23**(1), 3 (2009). https://doi.org/10.1007/s10948-009-0547-9
23. H.C. Koo et al., Rashba effect in functional spintronic devices. Adv. Mater. **32**(51), 2002117 (2020). https://doi.org/10.1002/adma.202002117
24. B.A. Bernevig, S.-C. Zhang, Quantum spin hall effect. Phys. Rev. Lett **96**(10), 106802 (2006). https://doi.org/10.1103/PhysRevLett.96.106802
25. B. Rana, Y. Otani, Towards magnonic devices based on voltage-controlled magnetic anisotropy. Commun. Phys. (2019). https://doi.org/10.1038/s42005-019-0189-6
26. M. Julliere, Tunneling between ferromagnetic films. Phys. Lett. A **54**(3), 225–226 (1975). https://doi.org/10.1016/0375-9601(75)90174-7
27. S. Zuo, H. Fan, K. Nazarpour, H. Heidari, A CMOS analog front-end for tunnelling magnetoresistive spintronic sensing systems. IEEE Int. Symp. Circuits Syst. (ISCAS) (2019). https://doi.org/10.1109/ISCAS.2019.8702219
28. K. Rahimi, C. Diorio, C. Hernandez, M.D. Brockhausen, A simulation model for floating-gate MOS synapse transistors. IEEE Int. Symp.Circuits Syst. (ISCAS) (2002). https://doi.org/10.1109/ISCAS.2002.1011042
29. R.R. Harrison, J.A. Bragg, P. Hasler, B.A. Minch, S.P. Deweerth, A CMOS programmable analog memory-cell array using floating-gate circuits. IEEE Trans. Circuits Syst. II Analog Digit. Signal Process **48**(1), 4–11 (2001). https://doi.org/10.1109/82.913181
30. S. Yu, H.-S. Philip Wong, Modeling the switching dynamics of programmable-metallization-cell (PMC) memory and its application as synapse device for a neuromorphic computation system. Int. Electron Devices Meet. (2010). https://doi.org/10.1109/IEDM.2010.5703410
31. A. Aggarwal, B. Hamilton, "Training artificial neural networks with memristive synapses: HSPICE-matlab co-simulation. Symp. Neural Netw. Appl. Electr. Eng. (2012). https://doi.org/10.1109/NEUREL.2012.6419974
32. L. Zheng, S. Shin, S.-M.S. Kang, "Memristor-based synapses and neurons for neuromorphic computing", in. IEEE Int. Symp. Circuits Syst. (ISCAS) **2015**, 1150–1153 (2015). https://doi.org/10.1109/ISCAS.2015.7168842
33. J.M. Skelton, D. Loke, T. Lee, S.R. Elliott, Ab Initio Molecular-Dynamics Simulation of Neuromorphic Computing in Phase-Change Memory Materials. ACS Appl. Mater. Interfaces **7**(26), 14223–14230 (2015). https://doi.org/10.1021/acsami.5b01825
34. Y. LeCun, Y. Bengio, G. Hinton, Deep learning. Nature **521**(7553), 536 (2015). https://doi.org/10.1038/nature14539
35. V.-T. Nguyen, Q.-K. Trinh, R. Zhang, Y. Nakashima, STT-BSNN: an in-memory deep binary spiking neural network based on STT-MRAM. IEEE Access **9**, 151373–151385 (2021). https://doi.org/10.1109/ACCESS.2021.3125685
36. L.F. Abbott, B. DePasquale, R.-M. Memmesheimer, Building functional networks of spiking model neurons. Nat. Neurosci. **19**(3), 350 (2016). https://doi.org/10.1038/nn.4241
37. I. Hayashi et al., A 250-MHz 18-Mb Full Ternary CAM With Low-Voltage Matchline Sensing Scheme in 65-nm CMOS. IEEE J. Solid-State Circuits **48**(11), 2671–2680 (2013). https://doi.org/10.1109/JSSC.2013.2274888
38. A. Amirany, M.H. Moaiyeri, K. Jafari, Nonvolatile Associative Memory Design Based on Spintronic Synapses and CNTFET Neurons. IEEE Trans. Emerg. Top. Comput. **10**(1), 428–437 (2022). https://doi.org/10.1109/TETC.2020.3026179
39. Y. Ma et al., A 600-μW ultra-low-power associative processor for image pattern recognition employing magnetic tunnel junction-based nonvolatile memories with autonomic intelligent power-gating scheme. Jpn. J Appl. Phys. **55**(4), 15 (2016). https://doi.org/10.7567/JJAP.55.04EF15
40. E. Kitagawa et al., Impact of ultra low power and fast write operation of advanced perpendicular MTJ on power reduction for high-performance mobile CPU. Int. Electron Devices Meet. (2012). https://doi.org/10.1109/IEDM.2012.6479129
41. P. Lennie, The Cost of Cortical Computation. Curr. Biol. **13**(6), 493–497 (2003). https://doi.org/10.1016/S0960-9822(03)00135-0
42. K. Lee, J. J. Kan, and S. H. Kang, "Unified embedded non-volatile memory for emerging mobile markets", in Proceedings of the 2014 international symposium on Low power electronics and design, New York, (NY, USA, 2014) pp. 131–136. https://doi.org/10.1145/2627369.2631641
43. H. Noguchi et al., "A 250-MHz 256b-I/O 1-Mb STT-MRAM with advanced perpendicular MTJ based dual cell for nonvolatile magnetic caches to reduce active power of processors," in 2013 Symposium on VLSI Technology 2013, pp. C108–C109.
44. J. Grollier, D. Querlioz, M.D. Stiles, Spintronic Nanodevices for Bioinspired Computing. Proc. IEEE **104**(10), 2024–2039 (2016). https://doi.org/10.1109/JPROC.2016.2597152
45. Y. Zhang et al., Electrical modeling of stochastic spin transfer torque writing in magnetic tunnel junctions for memory and logic applications. IEEE Trans. Magn. **49**(7), 4375–4378 (2013). https://doi.org/10.1109/TMAG.2013.2242257
46. A.F. Vincent et al., Spin-transfer torque magnetic memory as a stochastic memristive synapse for neuromorphic systems. IEEE Trans. Biomed. Circuits Syst. **9**(2), 166–174 (2015). https://doi.org/10.1109/TBCAS.2015.2414423
47. N. Locatelli et al., Spintronic devices as key elements for energy-efficient neuroinspired architectures. Des. Autom. Test Eur. Conf. Exhib. (DATE) (2015). https://doi.org/10.7873/DATE.2015.1117
48. D. Zhang et al., "Energy-efficient neuromorphic computation based on compound spin synapse with stochastic learning", in 2015. IEEE Int. Symp. Circuits Syst. (ISCAS) (2015). https://doi.org/10.1109/ISCAS.2015.7168939
49. D. Zhang, L. Zeng, Y. Zhang, W. Zhao, J.O. Klein, "Stochastic spintronic device based synapses and spiking neurons for neuromorphic computation", in 2016. IEEE/ACM Int. Symp. Nanoscale Archit. (NANOARCH) (2016). https://doi.org/10.1145/2950067.2950105
50. K. Garello et al., "SOT-MRAM 300MM integration for low power and ultrafast embedded memories", in 2018. IEEE Symp. VLSI Circuits (2018). https://doi.org/10.1109/VLSIC.2018.8502269
51. G. Srinivasan, A. Sengupta, K. Roy, "Magnetic tunnel junction enabled all-spin stochastic spiking neural network", in. Des. Autom. Test Eur. Conf. Exhib. (DATE) (2017). https://doi.org/10.23919/DATE.2017.7927045
52. V. Ostwal, R. Zand, R. DeMara, J. Appenzeller, "A Novel Compound Synapse Using Probabilistic Spin–Orbit-Torque Switching for MTJ-Based Deep Neural Networks",. IEEE J. Explor. Solid-State Comput. Dev. Circuits **5**(2), 182–187 (2019). https://doi.org/10.1109/JXCDC.2019.2956468
53. H. Ghanatian, M. Ronchini, H. Farkhani, F. Moradi, STDP implementation using multi-state spin−orbit torque synapse. Semicond. Sci. Technol. **37**(2), 024004 (2021). https://doi.org/10.1088/1361-6641/ac419c

54. C. Timm, M. Di Ventra, Memristive properties of single-molecule magnets. Phys. Rev. B **86**(10), 104427 (2012). https://doi.org/10.1103/PhysRevB.86.104427
55. S. Lequeux et al., A magnetic synapse: multilevel spin-torque memristor with perpendicular anisotropy. Sci. Rep. (2016). https://doi.org/10.1038/srep31510
56. S. Fukami, C. Zhang, S. DuttaGupta, A. Kurenkov, H. Ohno, Magnetization switching by spin–orbit torque in an antiferromagnet–ferromagnet bilayer system. Nat. Mater. **15**(5), 535 (2016). https://doi.org/10.1038/nmat4566
57. D. Querlioz, O. Bichler, P. Dollfus, C. Gamrat, Immunity to device variations in a spiking neural network with memristive nanodevices. IEEE Trans. Nanotechnol. **12**(3), 288–295 (2013). https://doi.org/10.1109/TNANO.2013.2250995
58. M. Prezioso, F. Merrikh-Bayat, B.D. Hoskins, G.C. Adam, K.K. Likharev, D.B. Strukov, Training and operation of an integrated neuromorphic network based on metal-oxide memristors. Nature **521**(7550), 61 (2015). https://doi.org/10.1038/nature14441
59. M. Wang, Y. Jiang, Compact model of domain wall MTJ driven by spin orbit torque and Dzyaloshinskii-moriya interaction. IEEE Trans. Magn. (2021). https://doi.org/10.1109/TMAG.2021.3138191
60. S. Fukami et al., 2009 "Low-current perpendicular domain wall motion cell for scalable high-speed MRAM,". Symp. VLSI Technol. pp. 230–231.
61. A. Sengupta, A. Ankit, K. Roy, "Performance analysis and benchmarking of all-spin spiking neural networks (Special session paper)", in. Int. Joint Conf.Neural Netw. (IJCNN) **2017**, 4557–4563 (2017). https://doi.org/10.1109/IJCNN.2017.7966434
62. S.A. Siddiqui, S. Dutta, A. Tang, L. Liu, C.A. Ross, M.A. Baldo, Magnetic domain wall based synaptic and activation function generator for neuromorphic accelerators. Nano Lett. **20**(2), 1033–1040 (2020). https://doi.org/10.1021/acs.nanolett.9b04200
63. J. Lourembam et al., Multi-state magnetic tunnel junction programmable by nanosecond spin-orbit torque pulse sequence. Adv. Electron. Mater. **7**(4), 2001133 (2021). https://doi.org/10.1002/aelm.202001133
64. J. Hong et al., A dual magnetic tunnel junction-based neuromorphic device. Adv. Intell. Syst. **2**(12), 2000143 (2020). https://doi.org/10.1002/aisy.202000143
65. W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophys. **5**(4), 115–133 (1943). https://doi.org/10.1007/BF02478259
66. S. Aunet, B. Oelmann, S. Abdalla, Y. Berg, "Reconfigurable subthreshold CMOS perceptron", in 2004. IEEE Int. Joint Conf. Neural Netw. **3**, 1983–1988 (2004). https://doi.org/10.1109/IJCNN.2004.1380919
67. M.A. Bañuelos-Saucedo et al., Implementation of a neuron model using FPGAS. J. Appl. Res. Technol. (2003). https://doi.org/10.22201/icat.16656423.2003.1.03.611
68. S. Jeyanthi, M. Subadra, "Implementation of single neuron using various activation functions with FPGA", in 2014. IEEE Int. Conf. Adv. Commun. Control Comput. Technol. (2014). https://doi.org/10.1109/ICACCCT.2014.7019273
69. H. Hikawa, A digital hardware pulse-mode neuron with piecewise linear activation function. IEEE Trans. Neural Netw. **14**(5), 1028–1037 (2003). https://doi.org/10.1109/TNN.2003.816058
70. C.-H. Tsai, Y.-T. Chih, W.H. Wong, C.-Y. Lee, "A Hardware-Efficient Sigmoid Function With Adjustable Precision for a Neural Network System", *IEEE Trans*. Circuits Syst. II Express Briefs **62**(11), 1073–1077 (2015). https://doi.org/10.1109/TCSII.2015.2456531
71. D. Baptista, F. Morgado-Dias, Low-resource hardware implementation of the hyperbolic tangent for artificial neural networks. Neural Comput. Appl. **23**(3), 601–607 (2013). https://doi.org/10.1007/s00521-013-1407-x
72. E.M. Izhikevich, Simple model of spiking neurons. IEEE Trans. Neural Netw. **14**(6), 1569–1572 (2003). https://doi.org/10.1109/TNN.2003.820440
73. H. Lim et al., Reliability of neuronal information conveyed by unreliable neuristor-based leaky integrate-and-fire neurons: a model study. Sci. Rep (2015). https://doi.org/10.1038/srep09776
74. J. Torrejon et al., Neuromorphic computing with nanoscale spintronic oscillators. Nature **547**(7664), 428 (2017). https://doi.org/10.1038/nature23011
75. S. Tsunegi et al., Evaluation of memory capacity of spin torque oscillator for recurrent neural networks. Jpn. J. Appl. Phys **57**(12), 120307 (2018). https://doi.org/10.7567/JJAP.57.120307
76. E.J. Basham, D.W. Parent, "An analog circuit implementation of a quadratic integrate and fire neuron", in 2009. Ann. Int. Conf. IEEE Eng. Med. Biol. Soc. (2009). https://doi.org/10.1109/IEMBS.2009.5332655
77. S. Millner, A. Grübl, K. Meier, J. Schemmel, and M. Schwartz, 2010 "A VLSI Implementation of the Adaptive Exponential Integrate-and-Fire Neuron Model,".*Adv.Neural Inform. Process. Syst.*, vol. 23. Accessed: Sep. 01, 2022. [Online]. Available: https://proceedings.neurips.cc/paper/2010/hash/25b2822c2f5a3230abfadd476e8b04c9-Abstract.html
78. S. Hashimoto, H. Torikai, A Novel Hybrid Spiking Neuron: Bifurcations, Responses, and On-Chip Learning. IEEE Trans. Circuits Syst. Regul. Pap. **57**(8), 2168–2181 (2010). https://doi.org/10.1109/TCSI.2010.2041507
79. T. Hishiki and H. Torikai, 2009 Bifurcation Analysis of a Resonate and Fire Type Digital Spiking Neuron In: CS. Leung, M. Lee, Jonathan H. Chan (eds) Neural Information. Springer, USA, pp. 392–400
80. T. Matsubara, H. Torikai, T. Hishiki, "A generalized rotate-and-fire digital spiking neuron model and its on-FPGA Learning", *IEEE Trans*. Circuits Syst. II Express Briefs **58**(10), 677–681 (2011). https://doi.org/10.1109/TCSII.2011.2161705
81. T. Matsubara and H. Torikai, "Dynamic Response Behaviors of a Generalized Asynchronous Digital Spiking Neuron Model," in *Neural Information Processing*, Berlin, Heidelberg, 2011, pp. 395–404. doi: https://doi.org/10.1007/978-3-642-24965-5_45.
82. H. Torikai, A. Funew, T. Saito, "Approximation of Spike-trains by Digital Spiking Neuron", in. Int. Joint Conf.Neural Netw. **2007**, 2677–2682 (2007). https://doi.org/10.1109/IJCNN.2007.4371381
83. C. Cerkez, I. Aybay, U. Halici, A digital neuron realization for the random neural network model. Proceed. Int. Conf. Neural Netw. **2**, 1000–1004 (1997). https://doi.org/10.1109/ICNN.1997.616163
84. K.Y. Camsari, R. Faria, B.M. Sutton, S. Datta, Stochastic p -bits for invertible logic. Phys. Rev **7**(3), 031014 (2017). https://doi.org/10.1103/PhysRevX.7.031014
85. A. Lucas, Ising formulations of many NP problems. Front. Phys. (2014). https://doi.org/10.3389/fphy.2014.00005
86. S. Geman, D. Geman, Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. IEEE Trans. Pattern Anal. Mach. Intell. **6**(6), 721–741 (1984). https://doi.org/10.1109/tpami.1984.4767596
87. B. Sutton, R. Faria, L.A. Ghantasala, R. Jaiswal, K.Y. Camsari, S. Datta, Autonomous Probabilistic Coprocessing With Petaflips per Second. IEEE Access **8**, 157238–157252 (2020). https://doi.org/10.1109/ACCESS.2020.3018682
88. A. Fukushima et al., Spin dice: A scalable truly random number generator based on spintronics. Appl. Phys. Express **7**(8), 083001 (2014). https://doi.org/10.7567/APEX.7.083001
89. D. Vodenicarevic et al., Low-energy truly random number generation with superparamagnetic tunnel junctions for unconventional

computing. Phys. Rev. Appl **8**(5), 054045 (2017). https://doi.org/10.1103/PhysRevApplied.8.054045
90. W.T. Coffey, Y.P. Kalmykov, Thermal fluctuations of magnetic nanoparticles: Fifty years after Brown. J. Appl. Phys **112**(12), 121301 (2012). https://doi.org/10.1063/1.4754272
91. W.F. Brown, Thermal fluctuations of a single-domain particle. Phys. Rev. **130**(5), 1677–1686 (1963). https://doi.org/10.1103/PhysRev.130.1677
92. J. Kaiser, A. Rustagi, K.Y. Camsari, J.Z. Sun, S. Datta, P. Upadhyaya, Subnanosecond fluctuations in low-barrier nanomagnets. Phys. Rev. Appl **12**(5), 054056 (2019). https://doi.org/10.1103/PhysRevApplied.12.054056
93. C. Safranski, J. Kaiser, P. Trouilloud, P. Hashemi, G. Hu, J.Z. Sun, Demonstration of nanosecond operation in stochastic magnetic tunnel junctions. Nano Lett. **21**(5), 2040–2045 (2021). https://doi.org/10.1021/acs.nanolett.0c04652
94. K. Hayakawa et al., Nanosecond Random Telegraph Noise in In-Plane Magnetic Tunnel Junctions. Phys. Rev. Lett. **126**(11), 117202 (2021). https://doi.org/10.1103/PhysRevLett.126.117202
95. J. Deng, V.P.K. Miriyala, Z. Zhu, X. Fong, G. Liang, Voltage-controlled spintronic stochastic neuron for restricted boltzmann machine with weight sparsity. IEEE Electron Device Lett. **41**(7), 1102–1105 (2020). https://doi.org/10.1109/LED.2020.2995874
96. Y.C.C. Wu et al., Voltage-gate-assisted spin-orbit-torque magnetic random-access memory for high-density and low-power embedded applications. Phys. Rev. Appl (2021). https://doi.org/10.1103/PhysRevApplied.15.064015
97. B. Zhang, Y. Liu, T. Gao, D. Zhang, W. Zhao, L. Zeng, "Time division multiplexing ising computer using single tunable true random number generator based on spin torque nano-oscillator", in. IEEE Int. Electron Dev. Meet. (IEDM) (2021). https://doi.org/10.1109/IEDM19574.2021.9720702
98. M. Suri, Ed., *Applications of Emerging Memory Technology: Beyond Storage*. Singapore: Springer Singapore, 2020. doi: https://doi.org/10.1007/978-981-13-8379-3.
99. J. Zhou, J. Chen, Prospect of spintronics in neuromorphic computing. Adv. Electron. Mater. **7**(9), 2100465 (2021). https://doi.org/10.1002/aelm.202100465
100. I. Chakraborty, A. Jaiswal, A.K. Saha, S.K. Gupta, K. Roy, Pathways to efficient neuromorphic computing with non-volatile memory technologies. Appl. Phys. Rev **7**(2), 021308 (2020). https://doi.org/10.1063/1.5113536
101. J. Zupan, Introduction to artificial neural network (ANN) methods: what they are and how to use them. Acta Chim. Slov. **41**(3), 327 (1994)
102. F. Rosenblatt, The perceptron: A probabilistic model for information storage and organization in the brain. Psychol. Rev. **65**(6), 386–408 (1958). https://doi.org/10.1037/h0042519
103. J. Han, C. Moraga, The influence of the sigmoid function parameters on the speed of backpropagation learning, in *From Natural to Artificial Neural Computation*. ed. by J. Mira, F. Sandoval (Heidelberg, Springer, Berlin Heidelberg, 1995), pp.195–201
104. D. Yarotsky, Error bounds for approximations with deep ReLU networks. Neural Netw **94**, 103–114 (2017). https://doi.org/10.1016/j.neunet.2017.07.002
105. E. Fan, Extended tanh-function method and its applications to nonlinear equations. Phys. Lett. A **277**(4–5), 212–218 (2000). https://doi.org/10.1016/S0375-9601(00)00725-8
106. M. Minsky, S.A. Papert, Perceptrons: an introduction to computational geometry. The MIT Press (2017). https://doi.org/10.7551/mitpress/11301.001.0001
107. H. Bourlard, Y. Kamp, Auto-association by multilayer perceptrons and singular value decomposition. Biol. Cybern. **59**(4–5), 291–294 (1988). https://doi.org/10.1007/BF00332918
108. E.B. Baum, On the capabilities of multilayer perceptrons. J. Complex. **4**(3), 193–215 (1988). https://doi.org/10.1016/0885-064X(88)90020-9
109. M.W. Gardner, S.R. Dorling, Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. Atmos. Environ. **32**(14–15), 2627–2636 (1998). https://doi.org/10.1016/S1352-2310(97)00447-0
110. J. Cai et al., Voltage-controlled spintronic stochastic neuron based on a magnetic tunnel junction. Phys. Rev. Appl **11**(3), 034015 (2019). https://doi.org/10.1103/PhysRevApplied.11.034015
111. J. Schmidhuber, Deep learning in neural networks: An overview. Neural Netw. **61**, 85–117 (2015). https://doi.org/10.1016/j.neunet.2014.09.003
112. K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks." arXiv, Dec. 02, 2015. Accessed: Oct. 28, 2022. [Online]. Available: http://arxiv.org/abs/1511.08458
113. J. Gu et al., Recent advances in convolutional neural networks. Pattern Recognit. **77**, 354–377 (2018). https://doi.org/10.1016/j.patcog.2017.10.013
114. Y. Pan et al., A multilevel Cell STT-MRAM-based computing in-memory accelerator for binary convolutional neural network. IEEE Trans. Magn. **54**(11), 1–5 (2018). https://doi.org/10.1109/TMAG.2018.2848625
115. C. Pan, A. Naeemi, A proposal for energy-efficient cellular neural network based on spintronic devices. IEEE Trans. Nanotechnol. **15**(5), 820–827 (2016). https://doi.org/10.1109/TNANO.2016.2598147
116. C. Pan, A. Naeemi, "Non-boolean computing benchmarking for beyond-CMOS devices based on cellular neural network", *IEEE*. J. Explor. Solid-State Comput. Devices Circuits **2**, 36–43 (2016). https://doi.org/10.1109/JXCDC.2016.2633251
117. S. Hijazi, R. Kumar, and C. Rowen, "Using Convolutional Neural Networks for Image Recognition," p. 12.
118. R. Chauhan, K.K. Ghanshala, R.C. Joshi, Convolutional neural network (CNN) for image detection and recognition. First Int. Conf. Secur. Cyber Comput. Commun. (2018). https://doi.org/10.1109/ICSCCC.2018.8703316
119. M. Schuster, K.K. Paliwal, Bidirectional recurrent neural networks. IEEE Trans. Signal Process. **45**(11), 2673–2681 (1997). https://doi.org/10.1109/78.650093
120. M. Lukoševičius, H. Jaeger, Reservoir computing approaches to recurrent neural network training. Comput. Sci. Rev. **3**(3), 127–149 (2009). https://doi.org/10.1016/j.cosrev.2009.03.005
121. K. Gregor, I. Danihelka, A. Graves, D.J. Rezende, D. Wierstra, DRAW: A recurrent neural network for image generation. Proc. Mach. Learn. Res. **37**, 1462–1471 (2015)
122. Y. Ming et al., "Understanding hidden memories of recurrent neural networks." IEEE Conf. Vis. Anal. Sci. Technol. (2017). https://doi.org/10.1109/VAST.2017.8585721
123. T. K. Dang, R. Wagner, J. Küng, N. Thoai, M. Takizawa, and E. Neuhold, Eds., Future Data and Security Engineering: Third International Conference, FDSE 2016, Can Tho City, Vietnam, November 23–25, 2016, Proceedings, vol. 10018. Cham: Springer International Publishing(2016). doi: https://doi.org/10.1007/978-3-319-48057-2.
124. L. Appeltant et al., Information processing using a single dynamical node as complex system. Nat. Commun. **2**(1), 468 (2011). https://doi.org/10.1038/ncomms1476
125. Q. Zheng, X. Zhu, Y. Mi, Z. Yuan, K. Xia, Recurrent neural networks made of magnetic tunnel junctions. AIP Adv. **10**(2), 025116 (2020). https://doi.org/10.1063/1.5143382
126. T. Furuta et al., Macromagnetic simulation for reservoir computing utilizing spin dynamics in magnetic tunnel junctions. Phys. Rev. Appl. **10**(3), 034063 (2018). https://doi.org/10.1103/PhysRevApplied.10.034063

127. G. Tanaka et al., Recent advances in physical reservoir computing: A review. Neural Netw. **115**, 100–123 (2019). https://doi.org/10.1016/j.neunet.2019.03.005
128. M.D. Stiles, A. Zangwill, Anatomy of spin-transfer torque. Phys. Rev. B **66**(1), 014407 (2002). https://doi.org/10.1103/PhysRevB.66.014407
129. J.-G. Zhu, Y. Wang, Microwave assisted magnetic recording utilizing perpendicular spin torque oscillator with switchable perpendicular electrodes. IEEE Trans. Magn. **46**(3), 751–757 (2010). https://doi.org/10.1109/TMAG.2009.2036588
130. M. Riou et al., Neuromorphic computing through time-multiplexing with a spin-torque nano-oscillator. IEEE Int. Electron Dev. Meet. (IEDM) (2017). https://doi.org/10.1109/IEDM.2017.8268505
131. M. Romera et al., Vowel recognition with four coupled spin-torque nano-oscillators. Nature **563**(7730), 230–234 (2018). https://doi.org/10.1038/s41586-018-0632-y
132. M. Zahedinejad et al., Two-dimensional mutually synchronized spin Hall nano-oscillator arrays for neuromorphic computing. Nat. Nanotechnol. **15**(1), 47–52 (2020). https://doi.org/10.1038/s41565-019-0593-9
133. A. J. Edwards *et al.*, "Passive frustrated nanomagnet reservoir computing." arXiv, Sep. 16, 2022. Accessed: Oct. 28, 2022. [Online]. Available: http://arxiv.org/abs/2103.09353
134. R. Hecht-Nielsen, "Theory of the Backpropagation Neural Network**Based on 'nonindent' by Robert Hecht-Nielsen, which appeared in Proceedings of the International Joint Conference on Neural Networks 1, 593–611, June 1989. © 1989 IEEE.," in Neural Networks for Perception, Elsevier, 1992, pp. 65–93. doi: https://doi.org/10.1016/B978-0-12-741252-8.50010-8.
135. A. Tavanaei, M. Ghodrati, S.R. Kheradpisheh, T. Masquelier, A. Maida, Deep learning in spiking neural networks. Neural Netw. **111**, 47–63 (2019). https://doi.org/10.1016/j.neunet.2018.12.002
136. Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, L. Shi, Direct training for spiking neural networks: faster, larger, better. Proc. AAAI Conf. Artif. Intell. **33**, 1311–1318 (2019). https://doi.org/10.1609/aaai.v33i01.33011311
137. M. Bouvier et al., Spiking neural networks hardware implementations and challenges: a survey. ACM J. Emerg. Technol. Comput. Syst. **15**(2), 1–35 (2019). https://doi.org/10.1145/3304103
138. A. Gruning and S. M. Bohte, "Spiking neural networks: principles and challenges," *Comput. Intell.* p. 10, 2014.
139. Y. Wu, L. Deng, G. Li, J. Zhu, L. Shi, Spatio-temporal backpropagation for training high-performance spiking neural networks. Front. Neurosci. **12**, 331 (2018). https://doi.org/10.3389/fnins.2018.00331
140. E.O. Neftci, H. Mostafa, F. Zenke, Surrogate gradient learning in spiking neural networks: bringing the power of gradient-based optimization to spiking neural networks. IEEE Signal Process. Mag. **36**(6), 51–63 (2019). https://doi.org/10.1109/MSP.2019.2931595
141. W. Fang, Z. Yu, Y. Chen, T. Huang, T. Masquelier, Y. Tian, Deep residual learning in spiking neural networks. Adv. Neural Inform. Process Syst. **34**, 1056 (2021)
142. P.J. Werbos, Backpropagation through time: what it does and how to do it. Proc. IEEE **78**(10), 1550–1560 (1990). https://doi.org/10.1109/5.58337
143. N.-D. Ho and I.-J. Chang, "TCL: an ANN-to-SNN Conversion with Trainable Clipping Layers," *58th ACM/IEEE Design Automation Conference (DAC)*, San Francisco, CA, USA, 2021, pp. 793–798. doi: https://doi.org/10.1109/DAC18074.2021.9586266.
144. J.H. Lee, T. Delbruck, M. Pfeiffer, Training deep spiking neural networks using backpropagation. Front. Neurosci. (2016). https://doi.org/10.3389/fnins.2016.00508
145. S. M. Bohte and J. N. Kok, "SpikeProp: Backpropagation for Networks of Spiking Neurons," p. 6.
146. J. Lisman, A mechanism for the Hebb and the anti-Hebb processes underlying learning and memory. Proc. Natl. Acad. Sci. **86**(23), 9574–9578 (1989). https://doi.org/10.1073/pnas.86.23.9574
147. E.M. Izhikevich, Solving the distal reward problem through linkage of STDP and dopamine signaling. Cereb. Cortex **17**(10), 2443–2452 (2007). https://doi.org/10.1093/cercor/bhl152
148. G. Srinivasan, A. Sengupta, K. Roy, Magnetic tunnel junction based long-term short-term stochastic synapse for a spiking neural network with on-chip STDP learning. Sci. Rep. **6**(1), 29545 (2016). https://doi.org/10.1038/srep29545
149. M.-C. Chen, A. Sengupta, K. Roy, Magnetic skyrmion as a spintronic deep learning spiking neuron processor. IEEE Trans. Magn. **54**(8), 1–7 (2018). https://doi.org/10.1109/TMAG.2018.2845890
150. H.I. Velarde, J. Nagaria, Z. Yin, A. Jacob, A. Jaiswal, Intrinsic spike-timing-dependent plasticity in stochastic magnetic tunnel junctions mediated by heat dynamics. IEEE Magn. Lett. **12**, 1–5 (2021). https://doi.org/10.1109/LMAG.2021.3136154
151. A.L. Hodgkin, A.F. Huxley, A quantitative description of membrane current and its application to conduction and excitation in nerve. J. Physiol. **117**(4), 500–544 (1952). https://doi.org/10.1113/jphysiol.1952.sp004764
152. A.N. Burkitt, A review of the integrate-and-fire neuron model: I. homogeneous synaptic input. Biol. Cybern. **95**(1), 1–19 (2006). https://doi.org/10.1007/s00422-006-0068-6
153. H. Huang et al., Quasi-hodgkin–huxley neurons with leaky integrate-and-fire functions physically realized with memristive devices. Adv. Mater. **31**(3), 1803849 (2019). https://doi.org/10.1002/adma.201803849
154. J.-W. Han, M. Meyyappan, Leaky integrate-and-fire biristor neuron. IEEE Electron Device Lett. **39**(9), 1457–1460 (2018). https://doi.org/10.1109/LED.2018.2856092
155. B. Datta Sahoo, Ring oscillator based sub-1V leaky integrate-and-fire neuron circuit. IEEE Int. Symp. Circuits Syst. (ISCAS) (2017). https://doi.org/10.1109/ISCAS.2017.8050980
156. D. Chatterjee, A. Kottantharayil, A CMOS compatible bulk FinFET-based ultra low energy leaky integrate and fire neuron for spiking neural networks. IEEE Electron Device Lett. **40**(8), 1301–1304 (2019). https://doi.org/10.1109/LED.2019.2924259
157. A. Jaiswal, A. Agrawal, P. Panda, K. Roy, Neural computing with magnetoelectric domain-wall-based neurosynaptic devices. IEEE Trans. Magn. **57**(2), 1–9 (2021). https://doi.org/10.1109/TMAG.2020.3010712
158. G. Tatara, H. Kohno, Theory of current-driven domain wall motion: spin transfer versus momentum transfer. Phys. Rev. Lett. **92**(8), 086601 (2004). https://doi.org/10.1103/PhysRevLett.92.086601
159. E. Ros, R. Carrillo, E.M. Ortigosa, B. Barbour, R. Agís, Event-Driven Simulation Scheme for Spiking Neural Networks Using Lookup Tables to Characterize Neuronal Dynamics. Neural Comput. **18**(12), 2959–2993 (2006). https://doi.org/10.1162/neco.2006.18.12.2959
160. A. Paz, S. Moran, Non deterministic polynomial optimization problems and their approximations. Theor. Comput. Sci. **15**(3), 251–277 (1981)
161. S. Patel, P. Canoza, S. Salahuddin, Logically synthesized and hardware-accelerated restricted Boltzmann machines for combinatorial optimization and integer factorization. Nat. Electron. **5**(2), 92–101 (2022)
162. R. Steinfeld and Y. Zheng, "A signcryption scheme based on integer factorization," in *International Workshop on Information Security*, 2000, pp. 308–322.

163. M.Y. Vardi, Boolean satisfiability: theory and engineering. Commun. ACM **57**(3), 5–5 (2014)
164. G.E. Hinton, T.J. Sejnowski, D.H. Ackley, *Boltzmann machines: Constraint satisfaction networks that learn* (Carnegie-Mellon University, Department of Computer Science Pittsburgh, PA, 1984)
165. N.A. Aadit, A. Grimaldi, M. Carpentieri, L. Theogarajan, G. Finocchio, K.Y. Camsari, "Computing with invertible logic: Combinatorial optimization with probabilistic bits", in. IEEE Int. Electron Devices Meet. (IEDM) **2021**, 40–43 (2021)
166. N. Onizawa, T. Hanyu, CMOS Invertible Logic: Bidirectional operation based on the probabilistic device model and stochastic computing. IEEE Nanotechnol. Mag. **16**(1), 33–46 (2021)
167. M. Kato, N. Onizawa, T. Hanyu, Design automation of invertible logic circuit from a standard HDL description. IfCoLoG J. Log. Their Appl. **8**(5), 1311–1333 (2021)
168. N.A. Aadit et al., Massively parallel probabilistic computing with sparse Ising machines. Nat. Electron **5**, 1–9 (2022)
169. A. Grimaldi et al., Spintronics-compatible approach to solving maximum-satisfiability problems with probabilistic computing, invertible logic, and parallel tempering. Phys. Rev. Appl. **17**(2), 024052 (2022)
170. D. Shin, N. Onizawa, W.J. Gross, T. Hanyu, Training hardware for binarized convolutional neural network based on CMOS invertible logic. IEEE Access **8**, 188004–188014 (2020)
171. J. Kaiser, W.A. Borders, K.Y. Camsari, S. Fukami, H. Ohno, S. Datta, Hardware-aware in situ learning based on stochastic magnetic tunnel junctions. Phys. Rev. Appl. **17**(1), 014016 (2022)
172. N. Onizawa, S.C. Smithson, B.H. Meyer, W.J. Gross, T. Hanyu, In-hardware training chip based on CMOS invertible logic for machine learning. IEEE Trans. Circuits Syst. Regul. Pap. **67**(5), 1541–1550 (2019)
173. A.Z. Pervaiz, L.A. Ghantasala, K.Y. Camsari, S. Datta, Hardware emulation of stochastic p-bits for invertible logic. Sci. Rep. **7**(1), 1–13 (2017)
174. J.D. Biamonte, Nonperturbative k-body to two-body commuting conversion Hamiltonians and embedding problem instances into Ising spins. Phys. Rev. A **77**(5), 052331 (2008)
175. J.D. Whitfield, M. Faccin, J.D. Biamonte, Ground-state spin logic. EPL Europhys. Lett. **99**(5), 57004 (2012)
176. N. Onizawa *et al.*, "A design framework for invertible logic," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, 2019, pp. 312–316.
177. S. Mitchell, M. OSullivan, and I. Dunning, "PuLP: a linear programming toolkit for python," Univ. Auckl. Auckl. N. Z 65 (2011).
178. S.C. Smithson, N. Onizawa, B.H. Meyer, W.J. Gross, T. Hanyu, Efficient CMOS invertible logic using stochastic computing. IEEE Trans. Circuits Syst. Regul. Pap. **66**(6), 2263–2274 (2019)
179. R. Faria, K.Y. Camsari, S. Datta, Low-barrier nanomagnets as p-bits for spin logic. IEEE Magn. Lett. **8**, 1–5 (2017)
180. P. Debashis, R. Faria, K.Y. Camsari, Z. Chen, Design of stochastic nanomagnets for probabilistic spin logic. IEEE Magn. Lett. **9**, 1–5 (2018)
181. N. Onizawa, T. Hanyu, "High convergence rates of CMOS invertible logic circuits based on many-body Hamiltonians", in. IEEE Int. Symp. Circuits Syst.(ISCAS) **2021**, 1–5 (2021)
182. A.Z. Pervaiz, B.M. Sutton, L.A. Ghantasala, K.Y. Camsari, Weighted $ p $-Bits for FPGA implementation of probabilistic circuits. IEEE Trans. Neural Netw. Learn. Syst. **30**(6), 1920–1926 (2018)
183. E. L. Lawler, *Combinatorial optimization: networks and matroids*. Courier Corporation, 2001.
184. A. Sbihi, R.W. Eglese, Combinatorial optimization and green logistics. Ann. Oper. Res. **175**(1), 159–175 (2010)
185. T.L. Magnanti, Combinatorial optimization and vehicle fleet planning: Perspectives and prospects. Networks **11**(2), 179–213 (1981)
186. C.-M. Lin, M. Gen, Multi-criteria human resource allocation for solving multistage combinatorial optimization problems using multiobjective hybrid genetic algorithm. Expert Syst. Appl. **34**(4), 2480–2490 (2008)
187. F. Barahona, M. Grötschel, M. Jünger, G. Reinelt, An application of combinatorial optimization to statistical physics and circuit layout design. Oper. Res. **36**(3), 493–513 (1988)
188. B.A. Cipra, An introduction to the Ising model. Am. Math. Mon. **94**(10), 937–959 (1987)
189. Y. Shim, A. Jaiswal, K. Roy, Ising computation based combinatorial optimization using spin-Hall effect (SHE) induced stochastic magnetization reversal. J. Appl. Phys. **121**(19), 193902 (2017)
190. M.W. Johnson et al., Quantum annealing with manufactured spins. Nature **473**(7346), 194–198 (2011)
191. T. Inagaki et al., A coherent Ising machine for 2000-node optimization problems. Science **354**(6312), 603–606 (2016)
192. P.L. McMahon et al., A fully programmable 100-spin coherent Ising machine with all-to-all connections. Science **354**(6312), 614–617 (2016)
193. T. Wang and J. Roychowdhury, "OIM: Oscillator-based Ising machines for solving combinatorial optimisation problems," In *International Conference on Unconventional Computation and Natural Computation*, 2019, pp. 232–256.
194. J. Chou, S. Bramhavar, S. Ghosh, W. Herzog, Analog coupled oscillator based weighted Ising machine. Sci. Rep. **9**(1), 1–10 (2019)
195. N. Mohseni, P.L. McMahon, T. Byrnes, Ising machines as hardware solvers of combinatorial optimization problems. Nat. Rev. Phys. **4**(6), 363–379 (2022)
196. Y. Shim, A. Jaiswal, K. Roy, "Stochastic Switching of SHE-MTJ as a Natural Annealer for Efficient Combinatorial Optimization", in. IEEE Int. Conf. Comput. Des. (ICCD) **2017**, 605–608 (2017)
197. S. Sharmin, Y. Shim, K. Roy, Magnetoelectric oxide based stochastic spin device towards solving combinatorial optimization problems. Sci. Rep. **7**(1), 1–9 (2017)
198. D.J. Earl, M.W. Deem, Parallel tempering: Theory, applications, and new perspectives. Phys. Chem. Chem. Phys. **7**(23), 3910–3916 (2005)
199. J.A. Acebrón, L.L. Bonilla, C.J.P. Vicente, F. Ritort, R. Spigler, The Kuramoto model: A simple paradigm for synchronization phenomena. Rev. Mod. Phys. **77**(1), 137 (2005)
200. D.I. Albertsson, M. Zahedinejad, A. Houshang, R. Khymyn, J. Åkerman, A. Rusu, Ultrafast Ising Machines using spin torque nano-oscillators. Appl. Phys. Lett. **118**(11), 112404 (2021)
201. B.C. McGoldrick, J.Z. Sun, L. Liu, Ising machine based on electrically coupled spin Hall nano-oscillators. Phys. Rev. Appl. **17**(1), 014006 (2022)
202. A. Hajimiri, T.H. Lee, A general theory of phase noise in electrical oscillators. IEEE J. Solid-State Circuits **33**(2), 179–194 (1998)
203. A. Slavin, V. Tiberkevich, Nonlinear auto-oscillator theory of microwave generation by spin-polarized current. IEEE Trans. Magn. **45**(4), 1875–1918 (2009)
204. J. Xiao, A. Zangwill, M.D. Stiles, Macrospin models of spin transfer dynamics. Phys. Rev. B **72**(1), 014446 (2005)
205. A. Houshang et al., Phase-binarized spin hall nano-oscillator arrays: towards spin hall ising machines. Phys. Rev. Appl. **17**(1), 014003 (2022)
206. J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan kaufmann, 1988.

207. D. Heckerman, C. Meek, and G. Cooper, "A Bayesian approach to causal discovery," in *Innovations in Machine Learning*, Springer, 2006, pp. 1–28.
208. M.B. Sesen, A.E. Nicholson, R. Banares-Alcantara, T. Kadir, M. Brady, Bayesian networks for clinical decision support in lung cancer care. PLoS ONE **8**(12), e82349 (2013)
209. A. S. Cofino, R. Cano Trueba, C. M. Sordo, and J. M. Gutiérrez Llorente, "Bayesian networks for probabilistic weather prediction," 2002.
210. E. T. Jaynes, *Probability theory: The logic of science*. Cambridge university press, 2003.
211. K. Murphy, "A Brief Introduction to Graphical Models and Bayesian Networks2," *Httpwww Cs Ubc Ca∼ MurphykBayesbnintro Html*, 1998.
212. V. K. Mansinghka, E. M. Jonas, and J. B. Tenenbaum, "Stochastic digital circuits for probabilistic inference," *Massachussets Inst. Technol. Tech. Rep. MITCSAIL-TR*, vol. 2069, 2008.
213. C.S. Thakur, S. Afshar, R.M. Wang, T.J. Hamilton, J. Tapson, A. Van Schaik, Bayesian estimation and inference using stochastic electronics. Front. Neurosci. **10**, 104 (2016)
214. J. Choi and R. A. Rutenbar, "Video-rate stereo matching using Markov random field TRW-S inference on a hybrid CPU+ FPGA computing platform," in *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays*, 2013, pp. 63–72.
215. Y. Akhmetov, A.P. James, "Probabilistic neural network with memristive crossbar circuits", in. IEEE Int. Symp. Circuits Syst. (ISCAS) **2019**, 1–5 (2019)
216. K. Wang et al., Threshold switching memristor-based stochastic neurons for probabilistic computing. Mater. Horiz. **8**(2), 619–629 (2021)
217. P. Mroszczyk, P. Dudek, "The accuracy and scalability of continuous-time Bayesian inference in analogue CMOS circuits", in. IEEE Int. Symp. Circuits Syst. (ISCAS) **2014**, 1576–1579 (2014)
218. L. Bagheriye, J.K. Kwisthout, Brain-inspired hardware solutions for inference in bayesian networks. Front. Neurosci (2021). https://doi.org/10.3389/fnins.2021.728086
219. P. Jeavons, D.A. Cohen, J. Shawe-Taylor, Generating binary sequences for stochastic computing. IEEE Trans. Inf. Theory **40**(3), 716–720 (1994)
220. L. A. de Barros Naviner, H. Cai, Y. Wang, W. Zhao, and A. B. Dhia, "Stochastic computation with spin torque transfer magnetic tunnel junction," in *2015 IEEE 13th International New Circuits and Systems Conference (NEWCAS)*, 2015, pp. 1–4.
221. Y. Wang, H. Cai, L.A. Naviner, J.-O. Klein, J. Yang, W. Zhao, "A novel circuit design of true random number generator using magnetic tunnel junction", in. IEEE/ACM Int. Symp. Nanoscale Archit. (NANOARCH) **2016**, 123–128 (2016)
222. S. Wang et al., "Hybrid VC-MTJ/CMOS non-volatile stochastic logic for efficient computing",. Des. Automation Test Eur. Conf. Exhib. (DATE) **2017**, 1438–1443 (2017)
223. X. Jia, J. Yang, Z. Wang, Y. Chen, H. H. Li, and W. Zhao, "Spintronics based stochastic computing for efficient Bayesian inference system," in *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2018, pp. 580–585.
224. X. Jia, J. Yang, P. Dai, R. Liu, Y. Chen, W. Zhao, "SPINBIS: Spintronics-based Bayesian inference system with stochastic computing", IEEE Trans. Comput. Aided Des. Integr. Circuits Syst **39**(4), 789–802 (2019)
225. B. Behin-Aein, V. Diep, S. Datta, A building block for hardware belief networks. Sci. Rep. **6**(1), 1–10 (2016)
226. P. Debashis, V. Ostwal, R. Faria, S. Datta, J. Appenzeller, Z. Chen, Hardware implementation of Bayesian network building blocks with stochastic spintronic devices. Sci. Rep. **10**(1), 1–11 (2020)
227. Y. Shim, S. Chen, A. Sengupta, K. Roy, Stochastic spin-orbit torque devices as elements for bayesian inference. Sci. Rep. **7**(1), 1–9 (2017)
228. R. Zand, K.Y. Camsari, S. Datta, R.F. Demara, Composable Probabilistic Inference Networks Using MRAM-based Stochastic Neurons. ACM J. Emerg. Technol. Comput. Syst. **15**(2), 1–22 (2019). https://doi.org/10.1145/3304105
229. R. Zand, K. Y. Camsari, S. D. Pyle, I. Ahmed, C. H. Kim, and R. F. DeMara, "Low-Energy Deep Belief Networks Using Intrinsic Sigmoidal Spintronic-based Probabilistic Neurons," in *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, Chicago IL USA, 2018. doi: https://doi.org/10.1145/3194554.3194558.
230. R. Faria, J. Kaiser, K.Y. Camsari, S. Datta, Hardware design for autonomous bayesian networks. Front. Comput. Neurosci. **15**, 584797 (2021)
231. C.M. Liyanagedera, A. Sengupta, A. Jaiswal, K. Roy, Stochastic spiking neural networks enabled by magnetic tunnel junctions: from nontelegraphic to telegraphic switching regimes. Phys. Rev. Appl. **8**(6), 064017 (2017). https://doi.org/10.1103/PhysRevApplied.8.064017