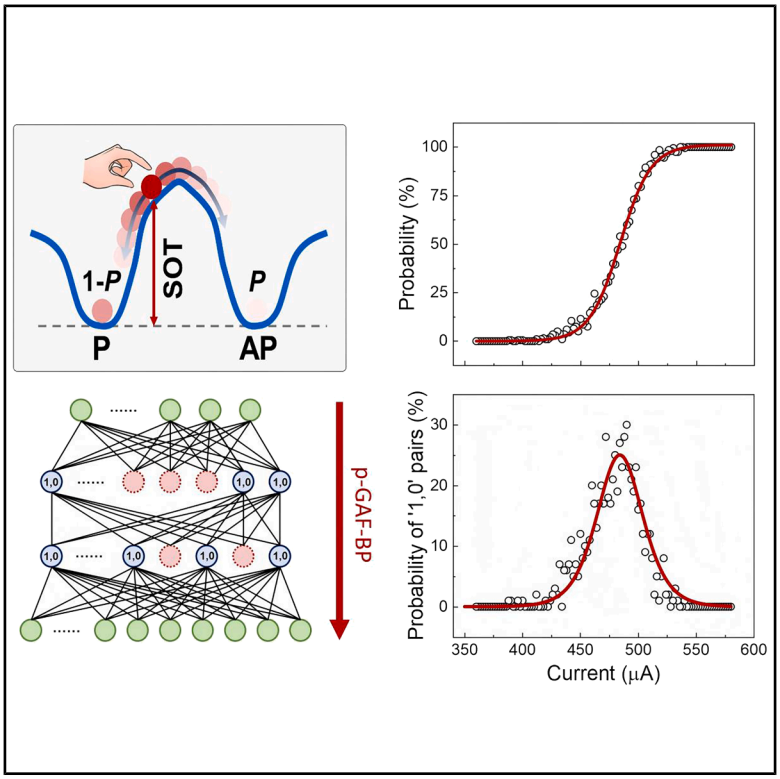
Newton

Article

Back propagation eased by spintronic-hardware-sampled probabilistic gradient of activation function

Graphical abstract



Highlights

- MTJ-based stochastic sampling emulates activation functions and their gradients
- Backpropagation is accelerated by MTJ-based stochastic sampling
- Probabilistic gradients of activation function reduce multiplications in BP

Authors

Yingqian Xu, Ran Zhang, Caihua Wan, ..., Dehao Kong, Guoqiang Yu, Xiufeng Han

Correspondence

wancaihua@iphy.ac.cn (C.W.), xfhan@iphy.ac.cn (X.H.)

In brief

Artificial intelligence (AI) holds transformative potential across science and society, yet training AI models remains computationally demanding. Xu et al. demonstrate an approach to accelerate backpropagation by replacing precise gradient computations with stochastic sampling using spin-orbit torque magnetic tunnel junctions. The resulting probabilistic gradient of the activation function identifies high-sensitivity neurons that require updates while bypassing saturated regions, reducing multiplication operations. This spintronic approach transforms costly deterministic operations into efficient sampling, offering a pathway toward energy-efficient AI.

Xu et al., 2026, Newton 2, 100520
August 3, 2026
https://doi.org/10.1016/j.newton.2026.100520

CellPress

Please cite this article in press as: Xu et al., Back propagation eased by spintronic-hardware-sampled probabilistic gradient of activation function, Newton (2026), https://doi.org/10.1016/j.newton.2026.100520

Newton

CellPress

Article

Back propagation eased by spintronic-hardware-sampled probabilistic gradient of activation function

Yingqian Xu,[1,2,4] Ran Zhang,[1,4] Caihua Wan,[1,3,*] Shengkai Xia,[1] Xiaohan Li,[1] Shiqiang Liu,[1] Shilong Xiong,[1] Dehao Kong,[1] Guoqiang Yu,[1,3] and Xiufeng Han[1,2,3,5,*]

[1]Beijing National Laboratory for Condensed Matter Physics, Institute of Physics, University of Chinese Academy of Sciences, Chinese Academy of Sciences, Beijing 100190, China

[2]Center of Materials Science and Optoelectronics Engineering, University of Chinese Academy of Sciences, Beijing 100049, China

[3]Songshan Lake Materials Laboratory, Dongguan, Guangdong 523808, China

[4]These authors contributed equally

[5]Lead contact

*Correspondence: wancaihua@iphy.ac.cn (C.W.), xfhan@iphy.ac.cn (X.H.)

https://doi.org/10.1016/j.newton.2026.100520

ACCESSIBLE OVERVIEW Artificial intelligence (AI) holds transformative potential across science and society. Among the algorithms that underpin AI, backpropagation (BP) is a cornerstone, providing the foundation for training artificial neural networks and driving the deep learning revolution. However, the BP algorithm remains computationally intensive, particularly due to its reliance on numerous multiplication operations. In this work, we focus on accelerating one of the computationally demanding steps in BP: the computation of activation function gradients. This is achieved using spin-orbit torque magnetic tunnel junctions, a class of spintronic devices that inherently exhibit stochastic switching behavior. By exploiting this intrinsic randomness, we transform deterministic gradient computations into consecutive sampling events. This yields a probabilistic gradient of the activation function, which replaces the exact gradient. This approach inherently identifies neurons operating in high-sensitivity regions that require weight updates while bypassing those in saturated regions, where gradients are negligible. This selective updating reduces the number of multiplication operations during the BP process. Overall, this work establishes spintronic devices as accelerators for activation function gradient computation, transforming a costly deterministic operation into efficient stochastic sampling and opening a pathway toward energy-efficient AI.

SUMMARY

The backpropagation (BP) algorithm, which updates parameters by following the reverse gradient of loss functions, is fundamental to artificial neural networks (ANNs). Its hardware realization can significantly impact the performance of artificial intelligence systems. Here, we present an approach that uses the probabilistic gradient of activation function (p-GAF) instead of the precise gradient to update networks. The p-GAF can be acquired in hardware via stochastic samplings of spintronic true random number generators (TRNGs) with an activation-function-like sampling performance. This approach significantly reduces the number of "multiplication" operations by automatically screening gradient components that contribute little to the update, thereby improving BP efficiency. The approach has been experimentally validated for an ANN performing the XOR logic gate and simulated as feasible for another ANN recognizing the MNIST dataset, where multiplication operations are reduced by ~80%, compared to the classic BP algorithm. Given the fundamental status of the BP algorithm, this probabilistic version provides a practical path toward BP hardware that can boost AI development.

Please cite this article in press as: Xu et al., Back propagation eased by spintronic-hardware-sampled probabilistic gradient of activation function, Newton (2026), https://doi.org/10.1016/j.newton.2026.100520

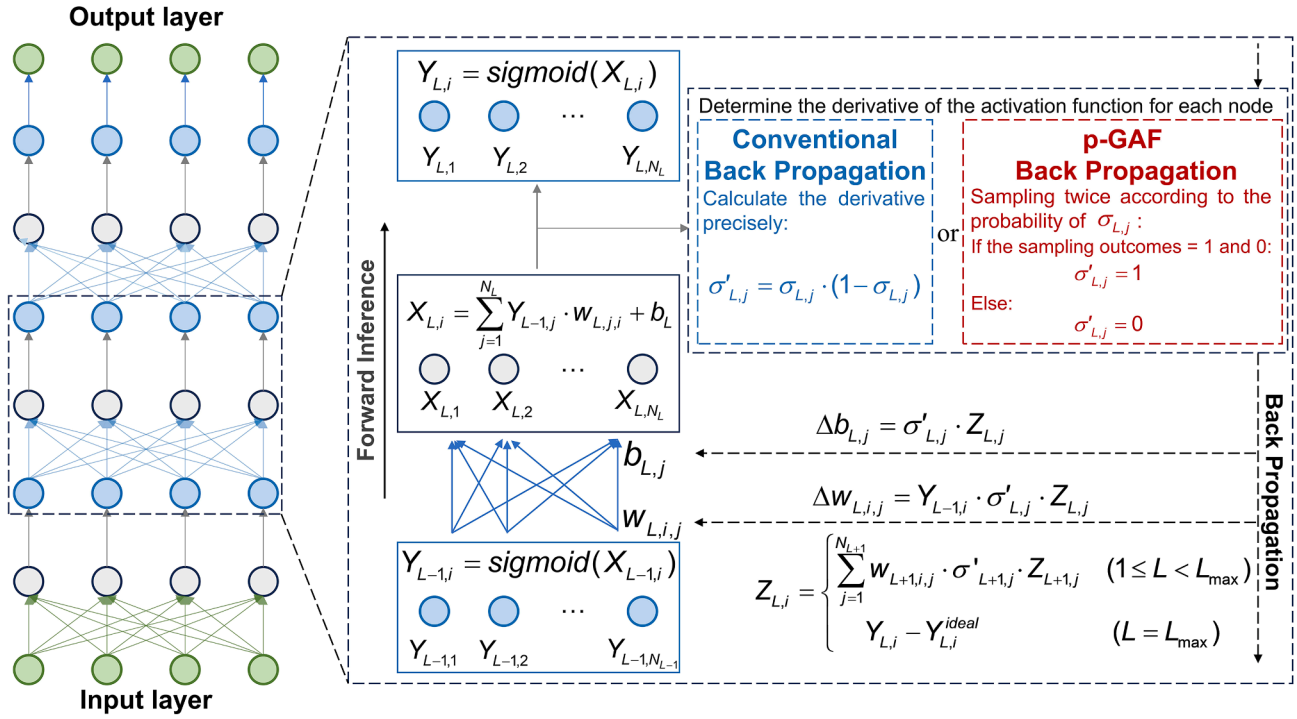


Figure 1. Streamline of the ANN training following the BP algorithm

The network contains up to L_{max} layers. The green nodes represent the input and output layers. The gray and blue nodes correspond to neurons that perform linear and nonlinear activation functions, respectively. N_L denotes the number of neurons in the L-th layer. σ and σ' represent the sigmoid-style activation function and its derivative, respectively. The BP algorithm consists of the forward inference and error BP steps. In the conventional BP algorithm, the derivative of the activation function is obtained through calculation (the blue dashed-line box), whereas the p-GAF-BP algorithm simplifies the differentiation operation by stochastic sampling (the red dashed-line box).

INTRODUCTION

Artificial intelligence (AI), as a transformative computation paradigm, holds incalculable value in revolutionizing industries and lives; however, its training demands significant energy and costs.[1] It is thus imperative in this context to develop hardware oriented to accelerate typical AI algorithms owing to its faster processing, lower latency, higher energy efficiency, and higher parallelism, especially for edge computing. Among numerous AI algorithms, the backpropagation (BP) algorithm exclusively occupies the foundational position, provides a rigorous method for artificial neural network (ANN) training, and promotes the pace of deep learning.[2–4] Though carefully designed photonic, mechanical, or electronic networks[5–7] can circumvent the BP algorithm and train themselves by leveraging their energy-relaxation process in physics, the flexibility, generality, versatility, and universality of the BP algorithm applicable to diverse network architectures, different tasks, distinct domains, and various datasets still solidify its cornerstone status in AI R&D. Hence, emerging microelectronic devices that can directly map basic operations in the BP algorithm and substantially speed it up are always desperately desired.

Every iteration of the BP algorithm includes a forward inference step and an error-BP step (Figure 1). In the former, one data point is fed to the input layer. Subsequently, by implementing the L-layers of weighted average $X_L = W_L Y_{L-1} + B_L$ and nonlinear activation $Y_L = F(X_L)$, the inference result $Y_{L\,max}$ is obtained. By comparing it with the ideal $Y_{L_{max}}^{ideal}$, we attain a loss function $loss = \left|Y_{L_{max}} - Y_{L_{max}}^{ideal}\right|^2$ of the ANN, which is essentially a high-dimensional energy landscape with regard to the parameters W_L and B_L. In the latter, by employing the chain rule and the conventional BP algorithm, we can compute the gradient of loss. Finally, those parameters are updated along the reverse direction of the gradient, leading to the minimization of *loss* iteration by iteration.

The BP algorithm includes 4 elementary operations: multiplication (×), addition (+), nonlinear activation (f(x)), and its differentiation (f′(x)). The former three have been hardwired. Nonvolatile memories (RRAM, FeRAM, and MRAM) with a crossbar layout have been utilized to perform multiply-accumulate operations in parallel, thereby accelerating both × and +.[8–12] Meanwhile, spintronic or resistive devices with nonlinear characteristics,[13–21] such as magnetic domain wall devices,[22,23] have been invented to emulate nonlinear functions. Only the differentiation operation remains to be implemented in hardware.

One efficient way to implement the last operation is to deploy unique activation functions whose derivatives are constant, as in ReLU, or depend only on their function values. For instance, for the sigmoid function, $\sigma'(x) = \sigma(1 - \sigma)$; for the tanh function, $tanh'(x) = 1 - tanh^2(x)$. Nevertheless, these derivatives still expend additional − and × operations. To further improve computational efficiency, a more hardware-friendly approach

Please cite this article in press as: Xu et al., Back propagation eased by spintronic-hardware-sampled probabilistic gradient of activation function, Newton (2026), https://doi.org/10.1016/j.newton.2026.100520

would be to directly measure both f′(*x*) and f(*x*) in physical device implementations.

Here, we demonstrate that magnetic tunnel junctions (MTJs), invaluable spintronic devices, can serve as a physical engine to simultaneously enhance the computational efficiency of both f′(*x*) and f(*x*) via the nonlinearity of the switching probability *P* on writing current *I* and, accordingly, the stochastic sampling. This property enables a probabilistic-gradient-of-activation-function-accelerated BP (p-GAF-BP) algorithm, in which (1) the derivatives of the activation function are determined through two stochastic samplings by MTJ rather than the precise calculation and (2) the p-GAF, instead of the precise gradient, is adopted to optimize the *loss* globally and, meanwhile, to save computational overheads. Then, we exploit MTJs as the derivative engine to streamline the training of an ANN for the XOR logic in experiments and for the MNIST image recognition in simulations. The latter reduces the number of $\times$ operations by ~80%. This work paves another avenue for spintronic devices to accelerate the fundamental differentiation operation and improve the training efficiency of ANNs.

RESULTS

The MTJ stack consists of W(3)/CoFeB(1.4)/MgO(1.5)/CoFeB(3)/W(0.4)/Co(2.7)/IrMn(10)/Ru(4 nm). After being fabricated into spin-orbit torque MTJs (SOT-MTJs),[24] the devices exhibit an aspect ratio of approximately 2:1 (inset in Figure 2D), favoring an in-plane magnetic easy axis along the long axis and the Y-type SOT switching mode. As illustrated in Figures 2C and 2D, the fabricated SOT-MTJs exhibit a high tunnel magnetoresistance (TMR) ratio of 110% and a critical current (density) of approximately 0.5 mA (2.8×10^7 A/cm^2) at 1 ms. These results demonstrate the high quality of the fabricated MTJs. We then generalize the SOT-MTJs as true random number generators (TRNGs). Our MTJs have an energy barrier $\geq 40\ k_BT$. To induce their probabilistic switching, we initialize them in the parallel state and then apply a current pulse with elevated amplitude to raise them to an energy level around the barrier top, where thermal fluctuation has a chance of stochastically determining the final state of MTJs (Figure 2B). In this scenario, MTJs can act as Bernoulli TRNGs with a tunable probability *P* or $(1 - P)$ to draw the high-resistance (R_H) or low-resistance (R_L) state, respectively. The population of the sampled R_H state grows with the increase in *I* (Figure 2E). This remarkable tunability in *P* is valuable and exquisitely exploited as shown below.

The switching probability, or average value *P*, statistically derived from the R_H population as a function of the writing current *I*, is summarized in Figures 3B and 3D. Each data point originates from the statistics of 200 reset-sampling attempts. Depending on representation, binary or bipolar, the *P*-*I* relation can be well fitted by the sigmoid or hyperbolic tangent function, respectively. This correspondence allows MTJs to simulate nonlinear activation functions of artificial neurons as reported previously.[14,25]

Moreover, we observe that $\sigma'(k(I - a)) = k \cdot \sigma(1 - \sigma)$: the derivative of σ is exactly *k* times the occurrence probability of (10) pairs in two independent draws, where σ and $(1 - \sigma)$ represent the probabilities of sampling 1 and 0, respectively, in each individual draw. Therefore, if R_H (digital 1) or R_L (digital 0) is sampled using MTJ at a fixed *I* for a large number of times *N*, we not only obtain the probability of sampling 1, $P(1) = \sigma = N_1/N$, by counting the occurrence frequency of R_H but also its derivative, $\sigma'(k(I - a)) = k \cdot \sigma(1 - \sigma) = k \cdot P(10) = k \cdot 2N_{10}/N$, simultaneously (Figure 3A). N_1 and N_{10} denote the number of times that 1 and (10) pairs are sampled in the *N* sampling attempts, respectively. $P(1)$ and $P(10)$ denote the corresponding probabilities. Before sampling, N_1 and N_{10} are initialized to zero. During the sampling procedure, as 1 is sampled, N_1 is increased by one. However, only when the outcome of two adjacent samplings is the (10) pair is the value of N_{10} incremented by one; otherwise, the value of N_{10} remains unchanged. The obtained $P(10)$-*I* relation (Figure 3C) closely matches the $\sigma'(k(I - a))/k$ function. Here, the differential operation is effectively transformed into the stochastic sampling operation. The latter can be considerably faster and more efficient when implemented in MTJs.

If the *P*-*I* relation is fitted by the hyperbolic tangent function, the transformation can still be achieved following the relation $tanh'(k(I - a)) = k(1 - tanh^2(k(I - a))) = 4k\sigma(2k(I - a))\sigma(-2k(I - a)) = 4k\sigma(2k(I - a))[1 - \sigma(2k(I - a))] = 4k\sigma(k((2I - a) - a))[1 - \sigma(k((2I - a) - a))] = 4k \cdot P(1, -1)_{@\ 2I-a} = 4k \cdot 2N_{1,-1}/N$. The derivative of $tanh(k(I - a))$ at $x = I$ corresponds to the probability of the following two samplings, sequentially drawing R_H (digital 1) and R_L (digital −1) at $2I - a$. In practice, the MTJ is switched at $2I - a$ (not at *I*), and then $N_{1,-1}$ is incremented by one only when the (1,−1) pair is sampled. The obtained $P(1,-1)$-*I* relation closely matches $tanh'(k(I - a))/(4k)$ (Figure 3E). Figure 3 clearly demonstrates that MTJs can emulate the sigmoid and tanh functions along with their derivatives simultaneously. This approach of converting derivative operations into stochastic samplings can also be generalized to other nonlinear functions, such as ReLU or leaky ReLU (details are provided in Note S1), without introducing unacceptable complexities.

Recently, an interesting simulation work employed two MTJs with orthogonal reference layers to approximate both $tanh(x)$ and $[\tanh'(x)]^{1/2}$ concurrently.[26] This approach requires an additional square operation to obtain $tanh'(x)$. Our method is innovative in transforming the derivation calculation into the sampling operation, which can help to automatically identify the dominating components of the precise gradient of the *loss* function, as shown below. Furthermore, our method of sampling is strictly accurate and directly estimates both f(*x*) and f′(*x*) without additional computations. Besides, the used devices are SOT-MTJs,[24,27–30] which feature simpler structures. These advantages indicate their practical usage in AI training.

Hereafter, MTJ-TRNGs are employed as the accelerator to simplify the differentiation of loss functions, using $\sigma'(x)$ as an illustrative example. Two approaches are considered. (1) Many samplings are performed to obtain a precise $\sigma'(x)$ and the precise gradient Δw of *loss*, such as $\Delta w_{L,i,j} = Y_{L-1,i}\sigma'_{L,j}Z_{L,j}$ in Figure 1. (2) Only two samplings are drawn for each σ', and the results are used to obtain a p-GAF, denoted as $\Delta_P w$. The expectation value of the p-GAF for long-term samplings exactly equals the precise gradient $\langle \Delta_P w \rangle = \Delta w$. However, in practice, only when the result of 2 sequential samplings is the (10) pair does $\sigma' = 1$; only in this case, the probabilistic gradient $\Delta_P w_{L,i,j}$ is nonzero and thereby simplified as $\Delta_P w_{L,i,j} = Y_{L-1,i}Z_{L,j}$; otherwise, $\Delta_P w = 0$. Then, parameters are updated according to the p-GAF $\Delta_P w$,

Please cite this article in press as: Xu et al., Back propagation eased by spintronic-hardware-sampled probabilistic gradient of activation function, Newton (2026), https://doi.org/10.1016/j.newton.2026.100520

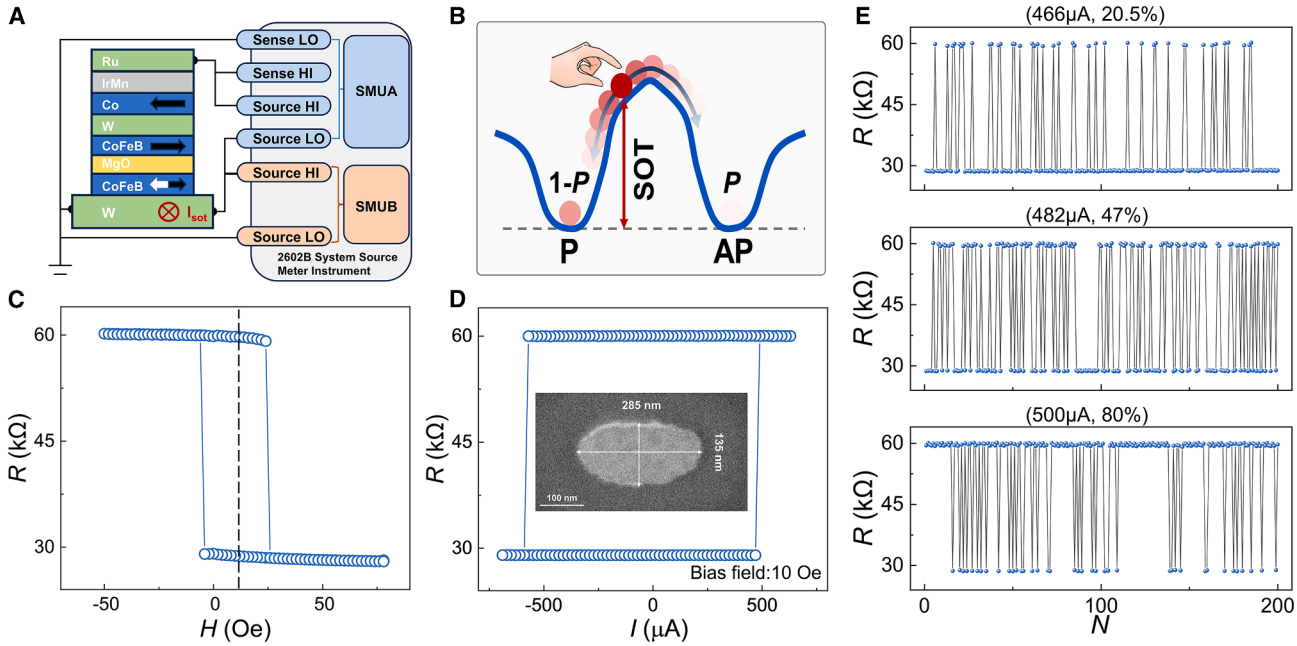


Figure 2. Performance of Y-type SOT-MTJ

(A) Schematic of the measurement setup. The stack structure of a Y-type SOT-MTJ is shown with the W layer as the spin current source, the neighboring CoFeB free layer, the MgO tunneling barrier, the CoFeB/W/Co/IrMn synthetic antiferromagnet structure, and the Ru capping layer. The blue block shows the 4-terminal source meter unit (SMU) to read out the resistance of the MgO MTJ, and the orange block shows the SMU to apply the spin-orbit torque current (I_{SOT}) to realize SOT switching of the bottom CoFeB free layer. Here, the source and sense represent the current source and voltmeter, respectively. LO and HI are short for low and high, respectively.

(B) Probabilistic switching diagram of high-barrier SOT-MTJ. Via I_{SOT} close to the critical switching current, one can set the state of SOT-MTJ near the top of the energy barrier, which isolates the parallel (P) and antiparallel (AP) states, and after switching off the I_{SOT}, the SOT-MTJ thus has a probability of P to relax into the AP state or $(1 - P)$ to the other P state.

(C) *R-H* hysteresis of SOT-MTJ.

(D) Magnetization switching induced by current pulses under a bias field of 10 Oe to compensate for the dipolar field from the synthetic antiferromagnet structure. The inset shows an elliptical cross-section image of a typical SOT-MTJ with an aspect ratio of about 2 to induce a shape anisotropy along the long axis. The current is applied along the short axis and thus generates a spin current polarized along the long axis, which favors the Y-type SOT switching scheme.

(E) Sampling outcomes, R_H or R_L, at 466, 482, and 500 μA, corresponding to P = 20.5%, 47%, and 80%. N denotes the number of sampling attempts.

which consumes much less computing overhead than the precise gradient Δw, owing to a large number of probable zero $\Delta_P w$ components. Since the second approach efficiently saves both samplings and multiplications, it is adopted in both experiments and simulations to train the XOR ANN (details are provided in Notes S2 and S3).

The network structure is illustrated in Figure 4A. The convergence progress of the p-GAF-BP algorithm, both in experiment and simulation, is compared with the classical one in Figure 4B. In all cases, the loss function decreases as training proceeds. To analyze parameter updates in detail, the evolution of the weight w_{DC} is compared in Figure 4C (additional results are provided in Note S4). Unlike the conventional BP algorithm, in which Δw_{DC} is nonzero at every step but remains very small for most of the process, the p-GAF-BP algorithm updates the weight selectively only at occasional steps. This selective update, governed by the probabilistic nature of sampling, significantly reduces the computational complexity by decreasing the number of required multiplications and additions/subtractions. Moreover, it achieves effective training for the XOR gate, as shown in Figure 4D.

To further evaluate the effectiveness of the p-GAF-BP algorithm, we apply it in simulation to train an ANN for the image recognition of the MNIST dataset (details are provided in Note S7) and compare its performance in accuracy, convergence speed, and number of multiplications and additions with the classical one. As shown in Figures 5A and 5B, the ANN architecture consists of an input layer (784 nodes and one bias, corresponding to the pixels of input images), one or more hidden layers (64 nodes and one bias each), and an output layer with 10 nodes that uses a SoftMax activation function and cross-entropy loss to represent the recognition results (details are provided in Note S8). In contrast to the conventional BP (Figure 5A), a key feature of our p-GAF-BP algorithm is that it propagates errors exclusively through nodes with a sampling result of (1,0), as illustrated in Figure 5B. To evaluate the performance of our p-GAF-BP algorithm on networks of varying sizes, architectures with one, two, and three hidden layers were configured. As illustrated in Figures 5C–5E, the recognition accuracy was improved progressively with increasing training epochs, and the corresponding dependence of the number of nodes with p-GAF = 0 is provided in Note S9. In all three network configurations, both the conventional BP and our p-GAF-BP

Please cite this article in press as: Xu et al., Back propagation eased by spintronic-hardware-sampled probabilistic gradient of activation function, Newton (2026), https://doi.org/10.1016/j.newton.2026.100520

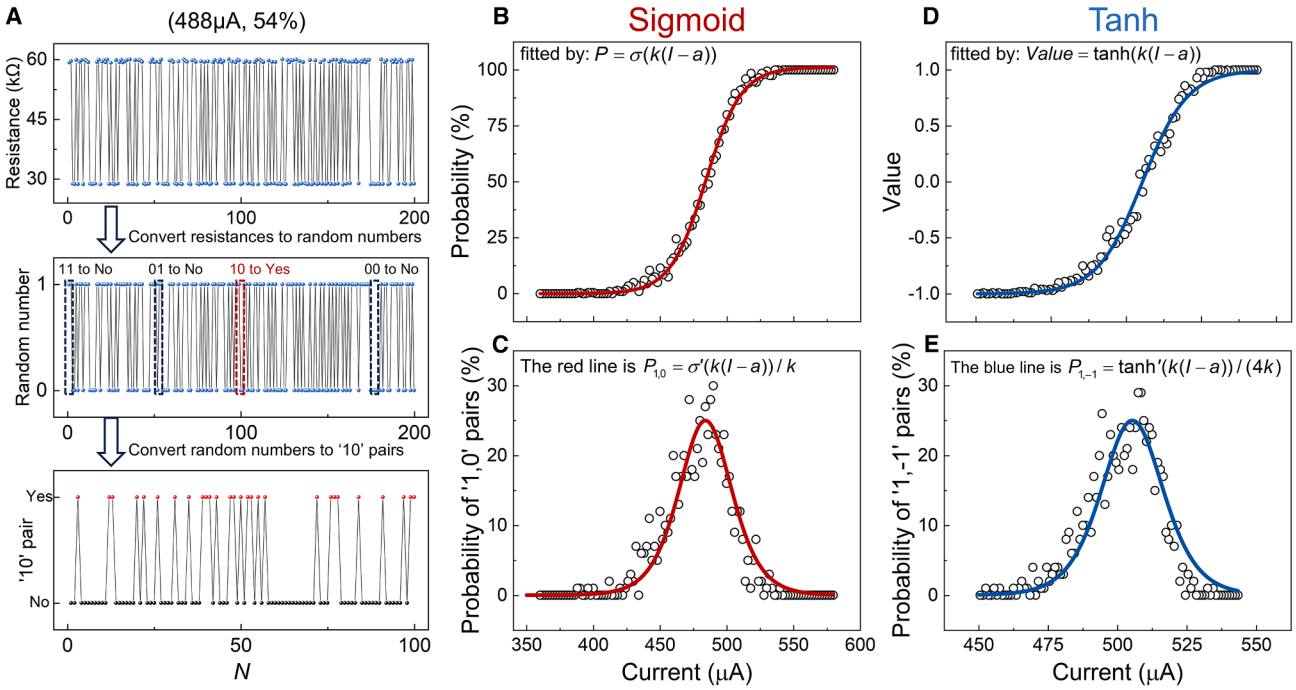


Figure 3. Automatic derivation of neurons by MTJ-TRNG

(A) Detailed procedure for obtaining the sampling counts of "10" pairs. For example, when we attempt to iteratively switch an SOT-MTJ at I = 488 μA for N = 200 trials, the resistance and digital output of the SOT-MTJ are shown in the top and middle images with a high-resistance probability of 54%. Then, we look into the details of two neighboring digital outputs. If the neighboring two outcomes are (11), (00), and (01), the indicator of the 10 pair is "no" or 0; only if the neighboring two outcomes are (10) is the indicator of the 10 pair "yes" or 1, as shown in the middle/bottom image. Then, N = 200 trials of single samplings are reduced to N = 100 trials of pair samplings, and we finally account for the probability of 1 in (B) and the (10) pair in (C) as non-biased estimations of the sigmoid function and its derivative, respectively.

(B and D) The relationship between the probability (or average value) of sampling R_H and writing current I follows the sigmoid function in the binary representation (B) and the hyperbolic tangent function in the bipolar representation (D).

(C and E) The probability (or average value) of sequentially sampling R_H and R_L as a function of writing current matches well with the derivative of the sigmoid function (C) and that of the hyperbolic tangent function (E).

algorithm achieved recognition accuracies exceeding 97%. To evaluate the energy efficiency of our p-GAF-BP in BP gradient computation, the number of arithmetic operations (Figure 5G) and the corresponding energy consumption (Figure 5H) required to reach 90% recognition accuracy were calculated. For instance, in the network with three hidden layers, the number of × and ± operations involved in gradient computation during BP decreased from 2.94×10^{10} and 3.80×10^{9} to 6.08×10^{9} and 9.74×10^{8}, respectively, only at the cost of 1.73×10^{8} sampling operations. After considering typical energy consumption per operation for ×, ±, and samplings (Figure 5F), the p-GAF-BP algorithm reduces the energy consumption associated with gradient computation by ~79% compared to conventional BP. Because the energy estimation is based on nontrivial circuitry design and MTJ optimization/downscaling and because the resolution choice of weights can be updated with the advancement of spintronics and CMOS, we therefore also used the operation numbers of multiplication, subtraction/addition, and sampling as another direct and straightforward set of benchmarking parameters free from the above issues.

To evaluate the effectiveness of our proposed p-GAF-BP algorithm, we modified the network structures shown in Figure 5 for the MNIST dataset in three aspects: the types of activation function, the number of hidden layers, and the number of sampling operations used for derivative calculation. The simulation results corresponding to these architectural variations are summarized in Table S4 of Note S10. In all cases, an accuracy above 97% was achieved, demonstrating the effectiveness of the p-GAF-BP algorithm. Specifically, the effect of the number of sampling operations used to estimate each p-GAF is further illustrated in Figure S11 of Note S10, where the number of sampling operations per p-GAF is set to 2, 20, and 200, respectively. As the number of sampling operations per p-GAF increases, both the total number of operations and the energy consumption approach the value of classic BP. The influence of layer sizes is evaluated separately in Figure S12, where a network with three hidden layers is considered and the number of nodes per hidden layer is varied from 16 to 256. The recognition accuracy increases from 93% to approximately 97% and saturates when the number of nodes exceeds 128.

To further validate the efficacy of our p-GAF-BP algorithm for deeper networks and more complex datasets, its performance was assessed in a deeper network architecture, specifically ResNet-18 in the CIFAR-10 dataset. As shown in Figure 6A,

Please cite this article in press as: Xu et al., Back propagation eased by spintronic-hardware-sampled probabilistic gradient of activation function, Newton (2026), https://doi.org/10.1016/j.newton.2026.100520

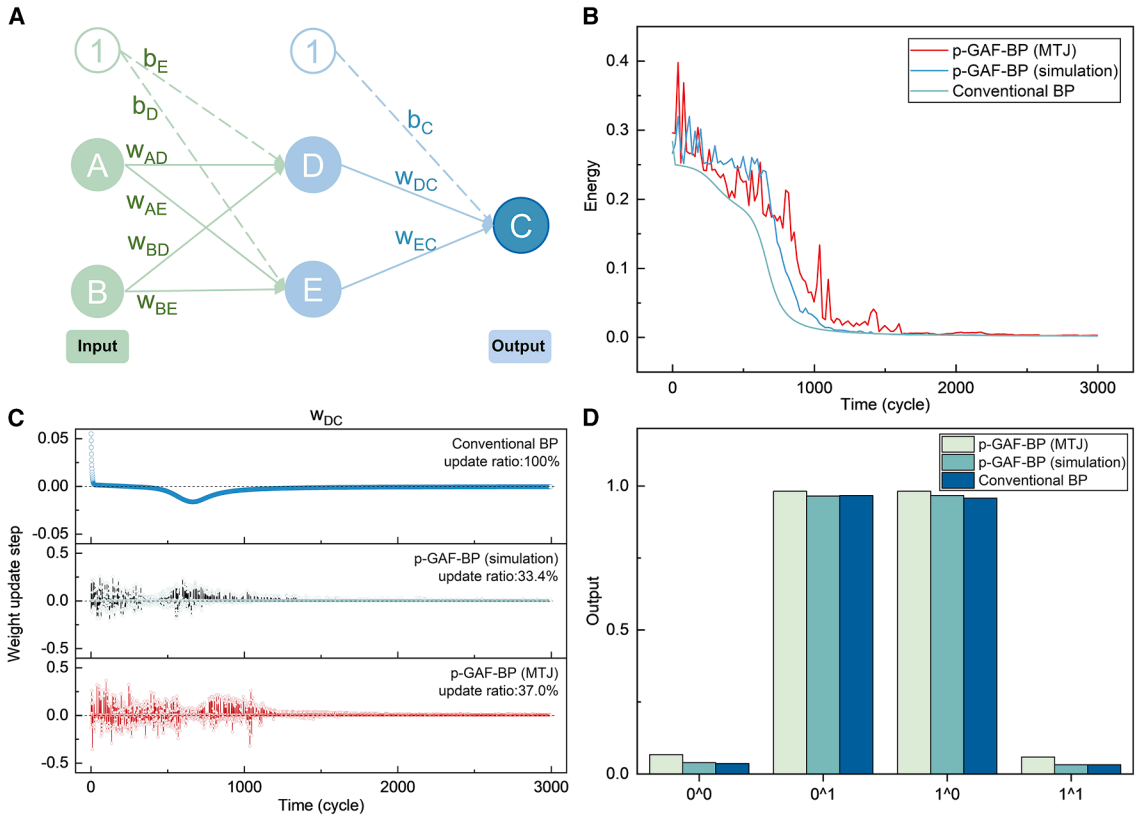


Figure 4. XOR gate trained using the p-GAF-BP algorithm in both experiment (sampling via MTJ) and simulation, compared with the conventional BP algorithm

(A) Network structure for the XOR gate. The green nodes A and B are used as binary logic inputs. They can be 1 or 0. The light blue nodes D and E are used as hidden nodes. The dark blue node C acts as the XOR output. The constant ① nodes are used for bias. The weights between nodes are also shown here.

(B) Energy evolution during training shows that the p-GAF-BP algorithm follows a training process similar to the conventional BP algorithm.

(C) Updated steps for the weight w_{DC} during training for the conventional BP, the simulated p-GAF-BP, and the experimental p-GAF-BP. While w_{DC} is nonzero, even though tiny, for the majority of the former case, it can be occasionally zero or spikily nonzero for the latter two.

(D) Training results of the XOR gate after 3,000 cycles demonstrate the effectiveness of the p-GAF-BP algorithm.

this architecture consists of an initial convolutional layer, four layers (each containing two basic blocks), and a fully connected layer. A notable modification from the conventional BP is the use of stochastic sampling to approximate the derivative of the leaky ReLU activation function, as implemented in the red-highlighted sections of Figures 6A and 6B. This implementation resulted in a final accuracy of 91.3% ± 0.3%, compared with 95.4% ± 0.2% achieved by the standard BP approach, as shown in Figure 6C. The test loss curves (Figure 6D) of the p-GAF-BP algorithm exhibit noticeable oscillations during training, which naturally arise from the probabilistic characteristics of the p-GAF-BP algorithm. These oscillations indicate that the p-GAF occasionally updates network parameters in suboptimal directions, thereby reducing the training accuracy at the cost of simpler computation. The significant broadening of the loss function across different trials, compared with the standard BP algorithm, also indicates that the convergence path of the loss function differs from trial to trial. This is expected as a characteristic feature of the p-GAF-BP algorithm. Luckily, the oscillation amplitude gradually decreases and the training process becomes more stable as the number of epochs increases, indicating that the p-GAF-BP algorithm can still finally achieve a stable training. Overall, these results collectively indicate that the proposed p-GAF-BP algorithm is applicable to deep network architectures and complex datasets as well, although its probabilistic nature may lead to some compromise in accuracy compared with the standard BP algorithm.

DISCUSSION

To reduce the computational cost of BP, several studies have explored alternative algorithms. For instance, direct feedback alignment (DFA)[32] uses fixed random feedback weights to propagate the error signal directly from the output layer to each hidden layer. This approach reduces computational and memory requirements by avoiding the backward propagation of errors through the entire network. However, our approach differs in that we use stochastic sampling of MTJs to accelerate the BP algorithm itself, reducing computational operations via p-GAF.

The underlying mechanism of this p-GAF-BP approach is as follows. The proposed technique enables stochastic identification of neurons operating in high- versus low-sensitivity regions

Please cite this article in press as: Xu et al., Back propagation eased by spintronic-hardware-sampled probabilistic gradient of activation function, Newton (2026), https://doi.org/10.1016/j.newton.2026.100520

CellPress

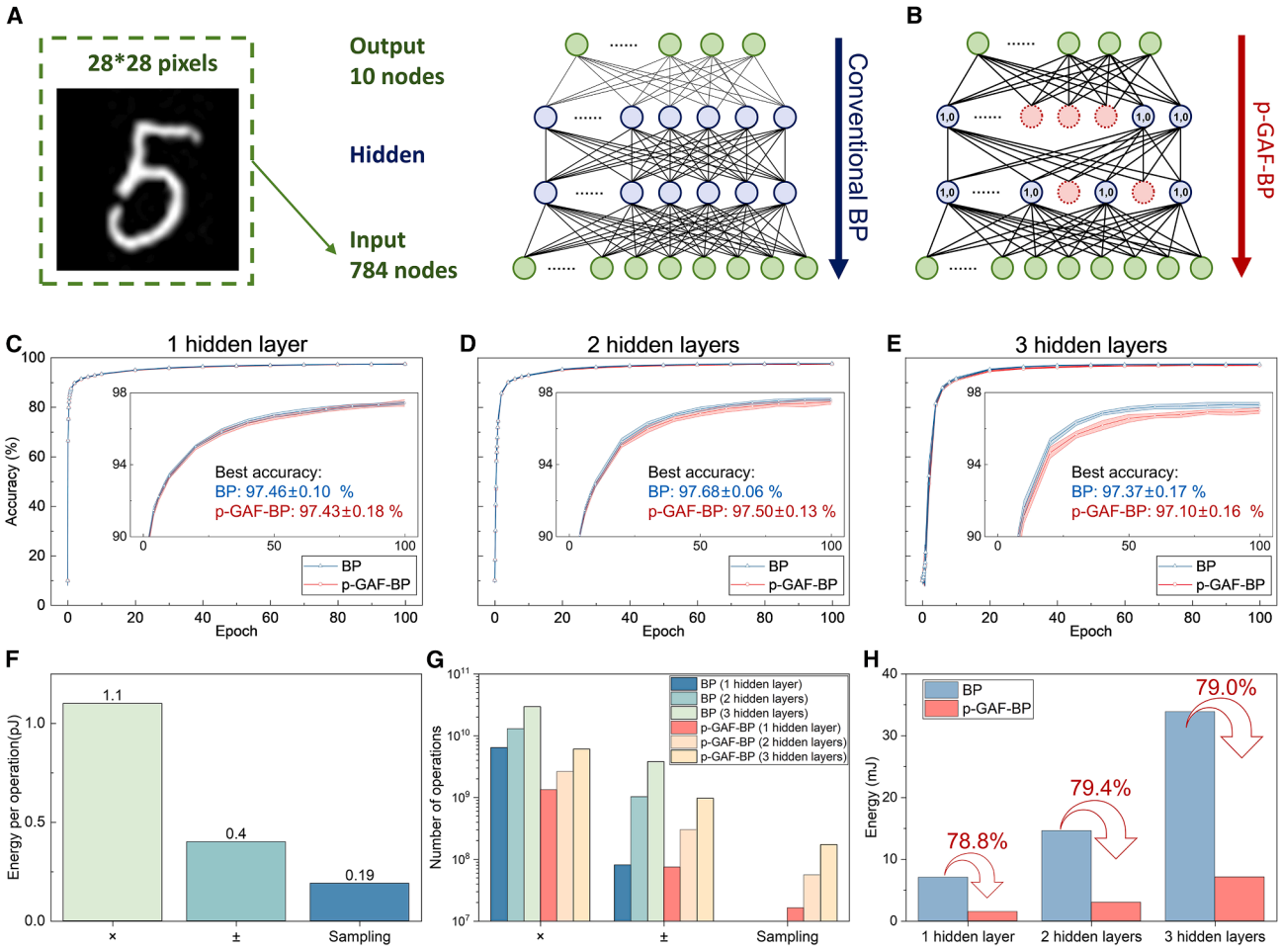


Figure 5. Image recognition using the MTJ-based p-GAF-BP algorithm in simulation, compared with the conventional BP algorithm

(A) ANN structure for image recognition. The input layer consists of 784 nodes (corresponding to a 28 × 28 pixel image in the MNIST dataset) and one bias node. Each hidden layer consists of 64 nodes and one bias node. The output layer consists of 10 nodes, corresponding to the recognition result for digits 0–9.

(B) In the p-GAF-BP algorithm, errors are propagated only through nodes whose sampling result is (1,0). The dashed red nodes represent nodes whose sampling results are not equal to (1,0) and function similarly to bias nodes.

(C–E) Recognition accuracy, averaged over ten trials, is shown as a function of training epochs for networks with one (C), two (D), and three (E) hidden layers, respectively. The insets highlight the phases where the accuracy exceeds 90%. For networks containing one to three hidden layers, both the conventional BP and p-GAF-BP algorithms achieve recognition accuracies exceeding 97%.

(F) Energy consumption per multiplication, addition/subtraction,[31] or MTJ sampling (the energy consumption of MTJ sampling is obtained from pre-layout simulations using LTspice, considering peripheral circuitry; details are provided in Note S5).

(G and H) Number of operations (G) and corresponding energy consumption (H) required for gradient computation in backpropagation to achieve 90% recognition accuracy. The energy consumption is obtained by combining the typical energy per operation in (F) with the number of operations in (G). The blue bars correspond to the energy consumption for gradient computation using conventional BP to achieve 90% recognition accuracy, whereas the red bars correspond to the p-GAF-BP algorithm. Across all three network structures, the p-GAF-BP algorithm reduces the energy consumption associated with gradient computation by approximately 79% compared to conventional BP. Details of the calculations in (G) and (H) are provided in Note S6.

of their activation characteristics by utilizing the statistical outcomes of repeated MTJ sampling operations. For activation functions with saturation behavior, such as the sigmoid function, neurons operating in the linear region (high sensitivity) are likely to yield complementary outcomes (e.g., (10) or (01)) under two stochastic sampling operations, whereas neurons in the saturation region (low sensitivity) predominantly produce identical outcomes (e.g., (00) or (11)). This sampling behavior provides an efficient indicator of the local sensitivity of the neuron output with respect to its input. From an optimization perspective, updating neurons operating in the high-sensitivity region is significantly more effective for reducing the error function, as these neurons exhibit non-negligible gradients. In contrast, neurons in the low-sensitivity region contribute little to gradient-based learning. The proposed method therefore serves to stochastically identify neurons operating in the high-sensitivity region, enabling efficient gradient approximation while avoiding unnecessary computations associated with neurons operating in the low-sensitivity region. The introduced stochasticity also helps escape local minima, complementing the natural variability

Please cite this article in press as: Xu et al., Back propagation eased by spintronic-hardware-sampled probabilistic gradient of activation function, Newton (2026), https://doi.org/10.1016/j.newton.2026.100520

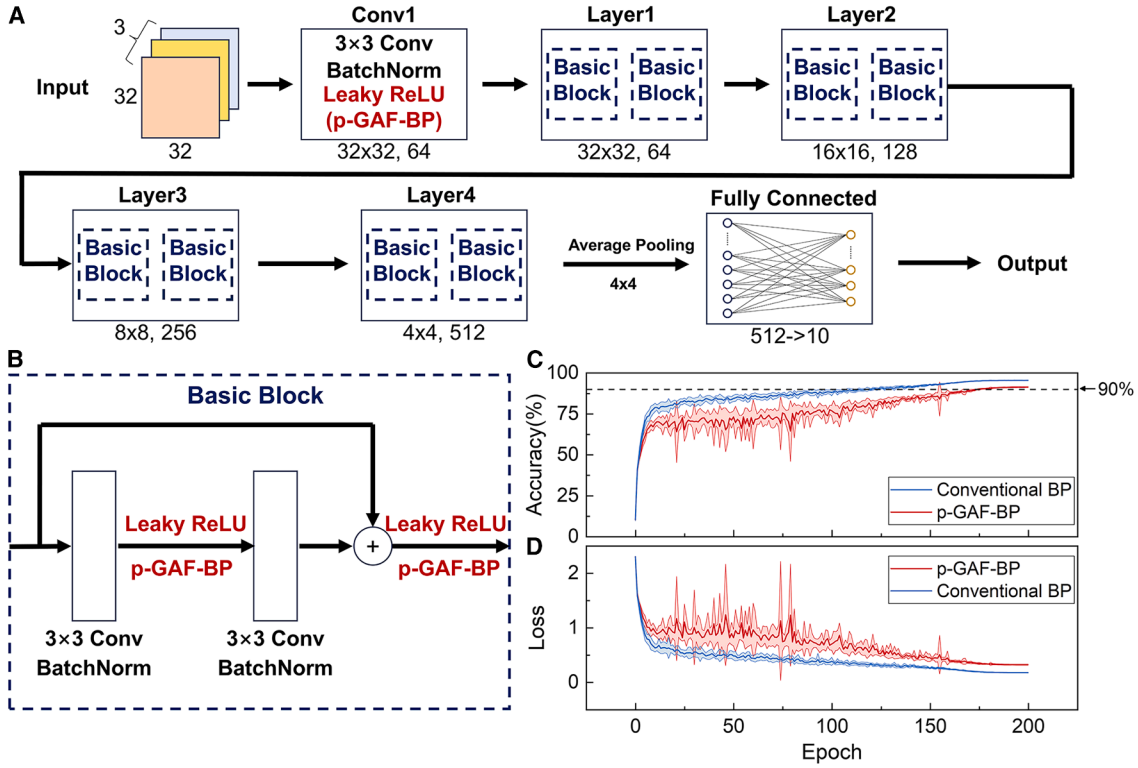


Figure 6. Evaluation of the p-GAF-BP algorithm implemented in a ResNet-18 architecture on the CIFAR-10 dataset

(A) Schematic of the ResNet-18 model integrated with the p-GAF-BP algorithm.

(B) Detailed structure of the basic residual block used in ResNet-18 under the p-GAF-BP algorithm.

(C and D) Comparison of test accuracy and test loss over training epochs between the p-GAF-BP algorithm and conventional BP.

Data in (C) and (D) were obtained from 10 independent runs with different random seeds. The shadowed area denotes the mean ± one standard deviation. Details of the parameters are provided in Note S8.

of high-dimensional data. These mechanisms form the core innovation of this work. This zero-gradient phenomenon may resemble that induced by the dropout algorithm,[33] but the two methods differ in both mechanisms and effectiveness (also see Note S11 for details).

For modern GPUs based on single instruction, multiple data (SIMD), the branching sensitivity inherent to SIMD presents challenges for integrating our p-GAF-BP scheme. Nonetheless, the potential savings of our p-GAF-BP scheme are not inherently unattainable. Through co-design of software optimizations (e.g., probability-based grouping) and hardware adaptations (e.g., specialized scheduling), GPU compatibility could be enhanced. However, the scheme's most immediate and full realization of savings lies in non-GPU applications, where its hardware-native probabilistic computing paradigm aligns naturally with efficiency demands—an advantage already supported by both our experimental and simulation results.

In summary, we present an approach that transforms the derivatives of nonlinear activation functions into two stochastic sampling events based on their probabilities. This method helps us to dynamically and temporally identify those weights associated with neurons operating in high-sensitivity regions and significantly reduces the number of multiplication and addition operations by automatically screening the large number of other weights associated with neurons operating in the saturated region during BP, thereby enhancing its computing efficiency through hardware-based spintronic TRNGs. The technique is applicable to typical noncollinear activation neurons, such as the sigmoid. It has been experimentally verified for an ANN performing XOR logic and simulated as feasible for another ANN for image recognition of the MNIST dataset, reducing multiplications and additions by ~80%. Furthermore, when applied to a deeper ResNet-18 architecture on the CIFAR-10 dataset, the p-GAF-BP algorithm still achieves stable training, with a trade-off in accuracy arising from its probabilistic nature relative to the standard BP algorithm. This p-GAF-BP algorithm can also simplify hardware implementation for the BP algorithm. Given the fundamental role of the BP algorithm, this probabilistic variant opens practical pathways for developing corresponding hardware to further boost AI training.

METHODS

The MTJ stack with W(3)/CoFeB(1.4)/MgO(1.5)/CoFeB(3)/W(0.4)/Co(2.7)/IrMn(10)/Ru(4 nm) was deposited by magnetron sputtering and then patterned into SOT-MTJ devices following the process detailed by Zhao et al.[24] The measurement (Figure 2A) was performed using a Keithley 2602B System

Source Meter Instrument with an external magnetic field applied by Helmholtz coils (East Changing Technologies). It enables fast current-induced switching and resistance detection of SOT-MTJs.

RESOURCE AVAILABILITY

Lead contact

Requests for further information and resources should be directed to and will be fulfilled by the lead contact, Xiufeng Han (xfhan@iphy.ac.cn).

Materials availability

This study did not generate new unique materials.

Data and code availability

- All data reported in this paper will be shared by the lead contact upon request.
- All original code is available in this paper's supplemental information.
- Any additional information required to reanalyze the data reported in this paper is available from the lead contact upon request.

ACKNOWLEDGMENTS

We appreciate Prof. Dr. Ningyuan Yin at the School of Electronics and Information Technology of Sun Yat-sen University and Prof. Dr. Yan Cui at the Institute of Microelectronics of the Chinese Academy of Sciences for fruitful discussions on designing circuitry for the SOT-MTJ-TRNGs and energy consumption evaluations. This work was supported by the National Key Research and Development Program of China (MOST) (grant no. 2022YFA1402800), the National Natural Science Foundation of China (NSFC) (grant nos. T2495210, 12134017, W2412079, and 12374131), and the Chinese Academy of Sciences President's International Fellowship Initiative (PIFI grant no. 2025PG0006).

AUTHOR CONTRIBUTIONS

C.W. and Y.X. planned the study. Y.X., C.W., R.Z., S. Xia, X.L., S.L., S. Xiong, and D.K. prepared and characterized the MTJ devices. C.W. and Y.X. developed the algorithm and experiments. X.H. was involved in all aspects of the study. All authors contributed to the writing of the manuscript. All authors discussed the results.

DECLARATION OF INTERESTS

The authors declare no competing interests.

DECLARATION OF GENERATIVE AI AND AI-ASSISTED TECHNOLOGIES IN THE WRITING PROCESS

During the preparation of this work, the authors used DeepSeek in order to improve language and readability. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.newton.2026.100520.

Received: June 10, 2025
Revised: March 10, 2026
Accepted: April 17, 2026

REFERENCES

1. Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L.M., Rothchild, D., So, D., Texier, M., and Dean, J. (2021). Carbon emissions and large neural network training. Preprint at arXiv. https://doi.org/10.48550/arXiv.2104.10350.
2. Hecht-Nielsen, R. (1992). Theory of the backpropagation neural network. In Neural networks for perception (Academic Press), pp. 65–93. https://doi.org/10.1016/B978-0-12-741252-8.50010-8.
3. LeCun, Y., Touresky, D., Hinton, G., and Sejnowski, T. (1988). A theoretical framework for back-propagation. In Proceedings of the 1988 connectionist models summer school, pp. 21–28.
4. Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1986). Learning representations by back-propagating errors. Nature *323*, 533–536. https://doi.org/10.1038/323533a0.
5. Ma, W.D.K., Lewis, J.P., and Kleijn, W.B. (2020). The HSIC bottleneck: Deep learning without back-propagation. In Proceedings of the AAAI conference on artificial intelligence, pp. 5085–5092. https://doi.org/10.1609/aaai.v34i04.5950.
6. Momeni, A., Rahmani, B., Malléjac, M., Del Hougne, P., and Fleury, R. (2023). Backpropagation-free training of deep physical neural networks. Science *382*, 1297–1303. https://doi.org/10.1126/science.adi8474.
7. Xue, Z., Zhou, T., Xu, Z., Yu, S., Dai, Q., and Fang, L. (2024). Fully forward mode training for optical neural networks. Nature *632*, 280–286. https://doi.org/10.1038/s41586-024-07687-4.
8. Kim, I.J., Kim, M.K., and Lee, J.S. (2023). Highly-scaled and fully-integrated 3-dimensional ferroelectric transistor array for hardware implementation of neural networks. Nat. Commun. *14*, 504. https://doi.org/10.1038/s41467-023-36270-0.
9. Bao, L., Wang, Z., Wang, Q., Yang, Y., Gao, Y., Shan, L., Sun, J., Yang, Y., Ling, Y., Zhang, H., et al. (2023). Hybrid-domain in-memory polynomial acceleration based on 40nm RRAM multi-core chip for machine vision calibration. In 2023 International Electron Devices Meeting (IEDM) (IEEE), pp. 1–4. https://doi.org/10.1109/IEDM45741.2023.10413781.
10. Jung, S., Lee, H., Myung, S., Kim, H., Yoon, S.K., Kwon, S.W., Ju, Y., Kim, M., Yi, W., Han, S., et al. (2022). A crossbar array of magnetoresistive memory devices for in-memory computing. Nature *601*, 211–216. https://doi.org/10.1038/s41586-021-04196-6.
11. Yao, P., Wu, H., Gao, B., Tang, J., Zhang, Q., Zhang, W., Yang, J.J., and Qian, H. (2020). Fully hardware-implemented memristor convolutional neural network. Nature *577*, 641–646. https://doi.org/10.1038/s41586-020-1942-4.
12. Wan, W., Kubendran, R., Schaefer, C., Eryilmaz, S.B., Zhang, W., Wu, D., Deiss, S., Raina, P., Qian, H., Gao, B., et al. (2022). A compute-in-memory chip based on resistive random-access memory. Nature *608*, 504–512. https://doi.org/10.1038/s41586-022-04992-8.
13. Zhang, R., Wan, C., Xu, Y., Li, X., Hoffmann, R., Hindenberg, M., Liu, S., Kong, D., Xiong, S., He, S., et al. (2025). Engineering-Oriented Design of Drift-Resilient MTJ Random Number Generator via Hybrid Control Strategies. Preprint at arXiv. https://doi.org/10.48550/arXiv.2501.15206.
14. Li, X., Wan, C., Zhang, R., Zhao, M., Xiong, S., Kong, D., Luo, X., He, B., Liu, S., Xia, J., et al. (2024). Restricted Boltzmann Machines Implemented by Spin-Orbit Torque Magnetic Tunnel Junctions. Nano Lett. *24*, 5420–5428. https://doi.org/10.1021/acs.nanolett.3c04820.
15. Xu, Y.Q., Li, X.H., Zhang, R., Wan, C.H., Wang, Y.Z., Liu, S.Q., Luo, X.M., Lan, G.B., Xia, J.H., Yu, G.Q., and Han, X.F. (2024). Self-stabilized true random number generator based on spin–orbit torque magnetic tunnel junctions without calibration. Appl. Phys. Lett. *125*, 132403. https://doi.org/10.1063/5.0226007.
16. Zhang, R., Li, X., Zhao, M., Wan, C., Luo, X., Liu, S., Zhang, Y., Wang, Y., Yu, G., and Han, X. (2024). Probability-Distribution-Configurable True Random Number Generators Based on Spin-Orbit Torque Magnetic

Tunnel Junctions. Adv. Sci. *11*, 2402182. https://doi.org/10.1002/advs.202402182.

17. Li, X.H., Zhao, M.K., Zhang, R., Wan, C.H., Wang, Y.Z., Luo, X.M., Liu, S.Q., Xia, J.H., Yu, G.Q., and Han, X.F. (2023). True random number generator based on spin–orbit torque magnetic tunnel junctions. Appl. Phys. Lett. *123*, 142403. https://doi.org/10.1063/5.0171768.
18. Ellis, M.O.A., Welbourne, A., Kyle, S.J., Fry, P.W., Allwood, D.A., Hayward, T.J., and Vasilaki, E. (2023). Machine learning using magnetic stochastic synapses. Neuromorphic Comput. Eng. *3*, 021001. https://doi.org/10.1088/2634-4386/acdb96.
19. Daniels, M.W., Madhavan, A., Talatchian, P., Mizrahi, A., and Stiles, M.D. (2020). Energy-efficient stochastic computing with superparamagnetic tunnel junctions. Phys. Rev. Appl. *13*, 034016. https://doi.org/10.1103/PhysRevApplied.13.034016.
20. Koo, M., Srinivasan, G., Shim, Y., and Roy, K. (2020). SBSNN: Stochastic-bits enabled binary spiking neural network with on-chip learning for energy efficient neuromorphic computing at the edge. IEEE Trans. Circuits Syst. I. *67*, 2546–2555. https://doi.org/10.1109/TCSI.2020.2979826.
21. Yu, S., Gao, B., Fang, Z., Yu, H., Kang, J., and Wong, H.S.P. (2013). Stochastic learning in oxide binary synaptic device for neuromorphic computing. Front. Neurosci. *7*, 186. https://doi.org/10.3389/fnins.2013.00186.
22. Yang, Q., Mishra, R., Cen, Y., Shi, G., Sharma, R., Fong, X., and Yang, H. (2022). Spintronic integrate-fire-reset neuron with stochasticity for neuromorphic computing. Nano Lett. *22*, 8437–8444. https://doi.org/10.1021/acs.nanolett.2c02409.
23. Zhou, J., Zhao, T., Shu, X., Liu, L., Lin, W., Chen, S., Shi, S., Yan, X., Liu, X., and Chen, J. (2021). Spin-orbit torque-induced domain nucleation for neuromorphic computing. Adv. Mater. *33*, 2103672. https://doi.org/10.1002/adma.202103672.
24. Zhao, M.K., Zhang, R., Wan, C.H., Luo, X.M., Zhang, Y., He, W.Q., Wang, Y.Z., Yang, W.L., Yu, G.Q., and Han, X.F. (2022). Type-Y magnetic tunnel junctions with CoFeB doped tungsten as spin current source. Appl. Phys. Lett. *120*, 182405. https://doi.org/10.1063/5.0086860.
25. Niazi, S., Chowdhury, S., Aadit, N.A., Mohseni, M., Qin, Y., and Camsari, K.Y. (2024). Training deep Boltzmann networks with sparse Ising machines. Nat. Electron. *7*, 610–619. https://doi.org/10.1038/s41928-024-01182-4.
26. Rodrigues, D., Raimondo, E., Tomasello, R., Carpentieri, M., and Finocchio, G. (2025). A design of magnetic tunnel junctions for the deployment of neuromorphic hardware for edge computing. Appl. Phys. Lett. *126*, 092402. https://doi.org/10.1063/5.0237090.
27. Liu, L., Pai, C.F., Li, Y., Tseng, H.W., Ralph, D.C., and Buhrman, R.A. (2012). Spin-torque switching with the giant spin Hall effect of tantalum. Science *336*, 555–558. https://doi.org/10.1126/science.1218197.
28. Liu, L., Lee, O.J., Gudmundsen, T.J., Ralph, D.C., and Buhrman, R.A. (2012). Current-induced switching of perpendicularly magnetized magnetic layers using spin torque from the Spin Hall effect. Phys. Rev. Lett. *109*, 096602. https://doi.org/10.1103/PhysRevLett.109.096602.
29. Miron, I.M., Garello, K., Gaudin, G., Zermatten, P.J., Costache, M.V., Auffret, S., Bandiera, S., Rodmacq, B., Schuhl, A., and Gambardella, P. (2011). Perpendicular switching of a single ferromagnetic layer induced by in-plane current injection. Nature *476*, 189–193. https://doi.org/10.1038/nature10309.
30. Han, X., Wang, X., Wan, C., Yu, G., and Lv, X. (2021). Spin-orbit torques: Materials, physics, and devices. Appl. Phys. Lett. *118*, 120502. https://doi.org/10.1063/5.0039147.
31. Horowitz, M. (2014). Computing's energy problem (and what we can do about it). In 2014 IEEE International Solid-State Circuits Conference (ISSCC) (IEEE), pp. 10–14. https://doi.org/10.1109/ISSCC.2014.6757323.
32. Nøkland, A. (2016). Direct feedback alignment provides learning in deep neural networks. Preprint at arXiv. https://doi.org/10.48550/arXiv.1609.01596.
33. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R.R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. Preprint at arXiv. https://doi.org/10.48550/arXiv.1207.0580.