Editors' Suggestion

Noise-based local learning using stochastic magnetic tunnel junctions

Kees Koenders[1,*] Leo Schnitzpan[2] Fabian Kammerbauer[2] Sinan Shu[2] Gerhard Jakob[2] Mathias Kläui[2,3] Johan H. Mentink[4] Nasir Ahmad[1] and Marcel van Gerven[1,†]

[1] *Department of Machine Learning and Neural Computing, Donders Institute for Brain, Cognition and Behaviour, Radboud University, Nijmegen, the Netherlands*

[2] *Institute of Physics, Johannes Gutenberg University Mainz, Mainz 55099, Germany*

[3] *Center for Quantum Spintronics, Norwegian University of Science and Technology, Trondheim 7491, Norway*

[4] *Department of Physics, Radboud University, Nijmegen, the Netherlands*

(Received 23 December 2024; revised 8 April 2025; accepted 10 April 2025; published 13 May 2025)

Brain-inspired learning in physical hardware has enormous potential for rapid learning with minimal energy expenditure. One of the characteristics of biological learning systems is their ability to learn in the presence of various noise sources. Inspired by this observation, we introduce a noise-based learning approach for physical systems implementing multilayer neural networks. Simulation results show that our approach allows for effective learning with a performance approaching that of the conventional backpropagation algorithm, which is effective but has a high energy cost. Using a spintronics hardware implementation, we demonstrate experimentally that learning can be achieved in a small network composed of physical stochastic magnetic tunnel junctions. These results provide a path toward efficient learning in general physical systems that embraces rather than mitigates the noise inherent in physical devices.

DOI: 10.1103/PhysRevApplied.23.054035

I. INTRODUCTION

Realizing learning in hardware by harnessing the physical properties of neuromorphic devices is currently a very active field of research. This is due to its potential to realize much more energy-efficient and conceptually unexplored routes toward intelligent machinery, which can adapt to its environment with minimal energy expenditure [1,2].

In recent years, various unconventional learning algorithms have been proposed as putative approaches to physics-based learning [3–8]. Contrastive methods such as equilibrium propagation (EP) have inspired particular interest because they circumvent the backpropagation (BP) of loss gradients. In EP, learning is achieved by comparing the steady state of a system run in a free phase and a weakly clamped phase [3,4,9].

A key challenge of these kinds of approaches are their sensitivity to the variability of the underlying devices. For example, exact twins are required for EP [10]. Although progress has been made toward removing this stringent constraint [11,12], it is well known that learning in biological systems can proceed by embracing noise as a principle for learning [13,14]. This is also very interesting for physical hardware, as noise is inherent to physical devices and emerges naturally and prominently when scaling to small spatial dimensions.

Purely stochastic hardware has recently attracted great interest with the development of probabilistic bits (p-bits) [15–18]; however, learning in stochastic networks is much more challenging than in conventional networks. For example, direct implementations of Boltzmann-like contrastive learning rules require the evaluation of the spatial correlations between node outputs, which is feasible for small networks but does not scale well to large and dense networks [19]. To circumvent this problem, a local learning algorithm that can directly leverage the noise of a single node alone would be desirable. Moreover, for many important generative models, measurement of correlations between node outputs is not sufficient for learning, and this strongly limits potential hardware applications.

Here, we focus on a different route to learning which, rather than avoiding noise or mitigating its impact, directly harnesses intrinsic device noise for learning. To this end, we focus on the recently developed decorrelated activity-based node perturbation (DANP) algorithm [20]. We show that with this algorithm it is feasible to learn by direct injection of physical noise at individual nodes. The DANP algorithm goes beyond the classical node-perturbation algorithm [21,22] and, in contrast to the classical formulation, has excellent convergence properties. The advantage of our approach is that learning requires just two forward

*Contact author: kees.koenders@donders.ru.nl
†Contact author: marcel.vangerven@donders.ru.nl

passes in a physical network under different noise perturbations, thereby bypassing the need for measuring the system in a clean (unperturbed) phase. Moreover, our algorithm does not require access to the noise fluctuations or their spatial correlations themselves, and the error signal can easily be extracted from a differential measurement of node activities, which is straightforward to realize in practice.

The demonstration of noise-based local learning is generally feasible with any neuromorphic material featuring stochastic behavior, such as memristive devices [23]. Here, we have judiciously chosen to work with magnetic systems, which are inherently well suited to unconventional computing due to their intrinsic nonlinearity and tunable nonvolatility [2,24]. In particular, magnetic systems feature specific advantages for stochastic hardware because their stochastic behavior stems from the spin degrees of freedom and hence does not feature the physical motion of atoms. This leads to superior endurance in comparison to other memristive materials. For example, the stochastic thermally induced dynamics can be exploited in magnetic systems for probabilistic computing [25] and Brownian reservoir computing [26]. Most conspicuous among these applications has been the use of stochastic magnetic tunnel junctions (sMTJs) [16]. In such systems, the random switching can be tuned over a wide range of frequencies, and biasing the switching can be easily achieved by coupling to magnetic layers or external fields, or by using spin torques induced by injected currents [27].

In this work, we extend the applicability of sMTJs to noise-based learning and explicitly harness the experimentally obtained hardware-generated noise of sMTJs for node perturbation. In particular, we show that direct injection of physical noise from an sMTJ to the nodes of the network is sufficient to realize learning. This is a significant advance over previous work demonstrating DANP-based learning using only idealized and physically unrealistic Gaussian noise. In contrast, in our work, actual sMTJ noise is directly harnessed as a resource for learning. This has the potential to lead to scalable and robust learning in physical systems.

II. METHODS

A. Noise-based learning

1. The learning problem

The learning problem is set up as follows. We consider a fully connected neural network consisting of L layers (nonlinear transformations), each of which contains N_l neurons. Let x_0 be the network input. The output of the lth layer is given by

$$x_l = f\,(a_l),$$

where $a_l = W_l x_{l-1}$ is the preactivation with weight matrix W_l, f is the activation function, and x_l is the output of layer $l = 1, \ldots, L$. The loss is defined as

$$\mathcal{L} = ||y^* - y||^2,$$

where y^* is the target output and $y = x_L$ is the predicted output. To minimize the loss, we consider learning rules that update the weights of such a network using update steps

$$W_l \leftarrow W_l - \eta \Delta W_l,$$

where η is a small, constant learning rate, and ΔW_l is a parameter update. The regular gradient-descent update is given by

$$\delta W_l^{\mathrm{GD}} = \left\langle \frac{\partial \mathcal{L}}{\partial W_l} \right\rangle, \tag{1}$$

where the expectation is estimated from the empirical distribution consisting of M input-output pairs.

2. Node perturbation and forward gradients

Node perturbation (NP) is an alternative to gradient descent that does not require global transmission of exact gradients through the network. NP relies on comparing the network states of a clean pass and a noise-perturbed pass, in which noise is injected into the preactivation of each layer to yield a perturbed output

$$\tilde{x}_l = f\,(\tilde{a}_l + \epsilon_l)\,, \tag{2}$$

where $\epsilon_l \sim \mathcal{N}(0, \sigma^2 I_l)$ is a noise vector and I_l is an $N_l \times N_l$ identity matrix. We define the loss differential as $\delta\mathcal{L} = \mathcal{L}(\tilde{x}_L) - \mathcal{L}(x_L)$, in which $\mathcal{L}(\tilde{x}_L)$ is the loss for the noisy output and $\mathcal{L}(x_L)$ is the loss for the clean output. Conventional node perturbation employs the update rule

$$\delta W_l^{\mathrm{NP}} = \sigma^{-2} \left\langle \delta\mathcal{L} \epsilon_l x_{l-1}^{\top} \right\rangle. \tag{3}$$

The issue with this formulation is that it does not perform well relative to gradient descent, requires a clean forward pass, and assumes that the noise is measurable.

In Ref. [20], activity-based node perturbation (ANP) was introduced as an alternative to regular NP whose updates are more aligned with the gradient direction, similar in spirit to the forward gradient approach of Refs. [28,29]. ANP employs the more advanced update rule

$$\delta W_l^{\mathrm{ANP}} = N \left\langle \delta\mathcal{L}\, \frac{\delta\alpha_l}{||\delta\alpha||^2}\, x_{l-1}^{\top} \right\rangle, \tag{4}$$

where $N = \sum_{l=1}^{L} N_l$ is the total number of units in the network, $\delta\alpha_l = \tilde{a}_l^{(1)} - \tilde{a}_l^{(2)}$ is the activity difference at layer l for two forward passes perturbed by different noise realizations, and $\delta a = (\delta a_1, \ldots, \delta a_L)$ is the concatenation of all activity differences.

Note that ANP does not require a clean forward pass; instead, it can compare two noisy passes without the need for access to the noise signal. This is ideally suited to application in noisy physical hardware, which can provide controlled noise, as detailed below.

3. Input decorrelation

For single-layer networks, we can show that ANP is robust, is highly performant, and trains extremely well; however, although ANP is more aligned with the true gradient, its performance still lags significantly behind that of gradient descent in multilayer networks. It was shown in Ref. [20] that this performance gap can be overcome by ensuring that the inputs to each network layer are decorrelated. That is, using a decorrelating transform $\bar{x}_l = R_l x_l$, we replace x_l with its decorrelated representation $\bar{x}_l$ for multilayer networks. It was shown in Ref. [30] that the decorrelation matrix R_l can be learned in an iterative manner using the update rule

$$R_l \leftarrow R_l - \epsilon \left\langle \bar{x}_l \bar{x}_l^\top - \mathrm{diag}\left(\bar{x}_l^2\right) \right\rangle R_l,$$

where ϵ is a small constant learning rate and R_l is initialized as the identity matrix. Integrating this into the preceding leads to DANP, which has been shown to approach the performance of stochastic gradient descent in multilayer networks [20].

B. Stochastic magnetic tunnel junctions

Our goal is to demonstrate how noise-based learning can be implemented in networks composed of stochastic magnetic tunnel junctions. sMTJs are magnetoresistive elements in which the thermal fluctuation of the magnetization leads to random fluctuations of the resistance [16,31]. The origin of the resistance fluctuations is the fluctuation of a ferromagnetic free layer in a magnetic tunnel junction, while the fixed reference layer is designed to be stable over the timescale of the device lifetime. Specifically, by tuning the anisotropies, one can obtain two-level fluctuations, leading to a volatile resistive element with telegraph noise [32]. If the energy barrier between local energy minima is of the order of $k_B T$, this switching occurs on easily measurable timescales. This volatile behavior of the magnetic free layer is called superparamagnetism, and for a magnetic tunnel junction (MTJ) with a typically designed uniaxial anisotropy, this results in a two-level switching with dwell times down to nanoseconds [31,33].

The fluctuation time, or dwell time τ, can be described within the "macrospin" approximation using the Néel-Arrhenius law [34]

$$\tau = \tau_0 \exp(E_b/k_B T), \tag{5}$$

where E_b is the energy barrier between the states, T is the temperature, and τ_0 is the attempt time. Here, the magnetic tunnel junction is based on a CoFeB/MgO/CoFeB interface exhibiting a tunnel magnetoresistance ratio of over 100% and a resistance area product of approximately 15 $\Omega\,\mu\mathrm{m}^2$. The tunnel magnetoresistance (TMR) stack used for the MTJ devices was deposited by rf- and dc-magnetron sputtering (Singulus Rotaris) at room temperature and subsequently annealed at 300 °C for 1 h under a 300-mT in-plane magnetic field. The composition of the stack, derived from a previously developed configuration [35], is as follows, with film thickness specified in nanometers: Ta(10)/Ru(10)/Ta(10)/PtMn(20)/CoFe(2.2)/Ru(0.8)/CoFeB(2.4)/MgO(1.1)/CoFeB(3.0)/Ta(10)/Ru(30). MTJ nanopillars were patterned into circular shapes with a diameter of 60 nm using electron beam lithography. These MTJs exhibit low in-plane uniaxial anisotropy in the ferromagnetic free layer, resulting in superparamagnetic switching between the parallel (low-resistance) and antiparallel (high-resistance) states.

Due to their inherent randomness, full complementary metal-oxide semiconductor (CMOS) compatibility, robustness, and energy efficiency, MTJ devices are excellent candidates for neuromorphic hardware harnessing noise-based learning algorithms such as ANP. This noise source not only allows for the generation of correlation-free true randomness [31] but also for the generation of noise based on specific probability distributions through appropriate interconnection of multiple sMTJs [36]. An additional advantage lies in the tunability of the MTJ state probability and fluctuation rate through applied magnetic fields or injected currents exerting torques on the free layer.

C. Learning with sMTJ noise

1. Analog learning

Since the main advantages of noise-based learning with sMTJs are realized in fully analog circuitry, we provide a minimal ANP-perceptron circuit design, as illustrated in Fig. 1. In this system, voltages function as inputs, while resistances function as weights. Noise due to the fluctuating resistance of the sMTJ is injected into the node as a volatile voltage source connected to an operational amplifier within a voltage adder circuit. This configuration enables the multiply-accumulate operation of the node. A leaky rectified linear unit (ReLU) activation function is implemented through a voltage-divider circuit with two diodes. The design can be expanded upon by connecting multiple sMTJs in series to deliver multilevel noise. This is the case for a resistance-based setup, as shown here. Alternatively, one might want to work with conductance, in which case a parallel setup should be used. A single sMTJ will provide two-level noise, and a series of n sMTJs will provide multilevel noise ranging from $n+1$ (if the TMR of each sMTJ is exactly the same) to 2^n (if

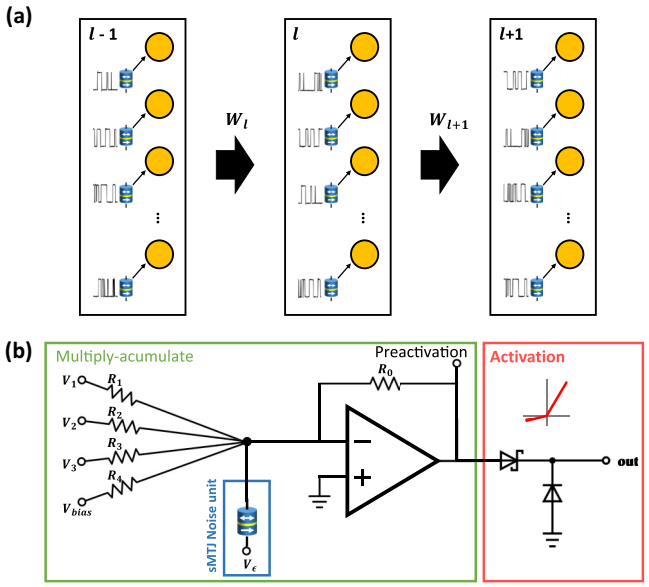

FIG. 1. (a) Illustration of the ANP network implementation with sMTJs as a perturbation source. A single sMTJ provides the required random noise for each node at each layer in the network. (b) ANP-perceptron circuit as an analog implementation of sMTJs and CMOS transistors. An sMTJ is used as a random stochastic perturbation source in the circuit. Weighting is implemented as (analog) resistive weights, and the ReLU-like activation function is implemented using two diodes. The multiply-accumulate operation is carried out by an operational amplifier in the configuration of a summing amplifier.

the TMR of each sMTJ is distinct). This offers more fine-grained perturbations, which can improve overall training performance. Additionally, the average fluctuation rate is increased by adding more sMTJs, as this is given by $\Gamma = 1/\tau^* = \sum_i \tau_i^{-1}$, where τ^* is the average dwell time of the sMTJ ensemble and τ_i the average dwell time of the ith sMTJ. A higher fluctuation rate increases the noise-generation bandwidth, allowing for shorter iteration times. Although a complete design including CMOS circuitry to implement the learning dynamics is feasible due to the local nature of the learning rule, this is reserved for future work.

2. Measured sMTJ noise

In our experiments, to demonstrate learning with real-world sMTJ noise, we use a time series measured from a single sMTJ as the perturbation source. This is implemented in a linear fashion, where neuron $i+1$ receives the signal sampled one timestep after neuron i. This approach inherently introduces strong autocorrelations, and it thus serves as a worst-case scenario in which a single sMTJ provides noise for an entire network.

In addition to using real sMTJ noise in larger networks, we provide a proof-of-concept demonstration in which an sMTJ is measured in real time using an Arduino to perform online learning.

3. Simulated sMTJ noise

To show the potential of the large-scale application of sMTJs, we employ an approach in which we construct a hidden Markov model with two states and a noisy observation. The parameters of this model are statistically derived from the time series of a single sMTJ. Details can be found in Appendix A.

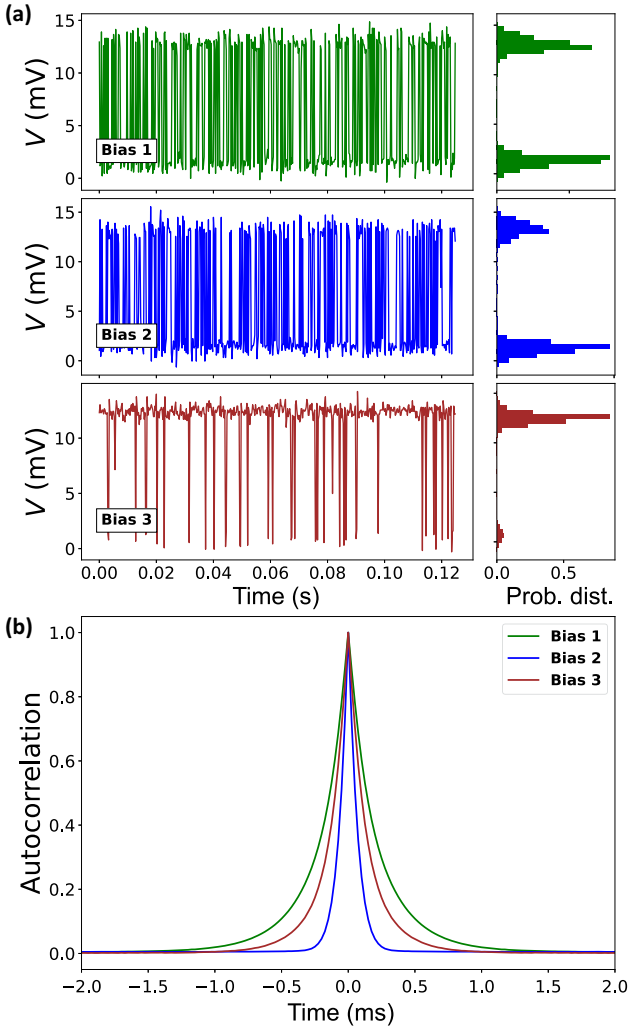

FIG. 2. (a) Time series of an sMTJ circuit for the generation of noise under three different biases originating from external in-plane magnetic fields ranging from approximately −3 to +3 mT. The top, center, and bottom panels display the switching behavior of the same sMTJ, along with histograms illustrating their respective probability distributions. (b) Autocorrelation of the sMTJ noise signal for sMTJs with the three different field biases shown in (a). The autocorrelation function was calculated from 2500-s-long time-series measurements.

4. Approximated sMTJ noise

A single sMTJ can be considered as a Bernoulli random variable with dwell times distributed according to a Poisson distribution. For this approximation, we assume that when noise is needed, the dwell time will already have passed. Thus, we draw from a symmetric Bernoulli distribution, scaling the result by a factor α. If multilevel noise is desired, h samples are drawn and summed.

III. RESULTS

A. Noise characterization

Figure 2 illustrates the stochastic random noise generated by superparamagnetic tunnel junctions. The data were acquired for 2500 s using an oscilloscope (with a sampling rate of 40 kHz, adapted to the dwell time of the sample used). A measurement from a single stochastic MTJ is shown in Fig. 2(a). The state probability is random, but it can be tuned electrically via spin-transfer torques [31] or magnetically via external in-plane magnetic fields. Biases may also arise intrinsically due to adjacent magnetic fields or as a result of the sMTJ fabrication process. We generate two-level noise with different probability distributions using external field biases.

When considering the autocorrelation function of these time series, the correlation time scale of the noise signal can be obtained, which specifically depends on the individual fluctuation rates of each sMTJ. The autocorrelation for the three different measured time series of Fig. 2(a) is plotted in Fig. 2(b). Depending on the sMTJ bias, the signal is decorrelated after 1 ms at the latest in the sample used; however, we note that with the demonstrated nanosecond dwell times, this can be orders of magnitude faster.

B. Training single-layer sMTJ networks

We tested the validity of our approach by examining learning behavior in simulated sMTJ networks. Figure 3 shows the final performance of single-layer networks (trained for 100 epochs) using the ANP formulation.

Importantly, the ANP approach is robust to a significant range of choices of parameters and noise sources. Figure 3 shows how training of a single layer is robust to learning rate and noise scale, regardless of whether the

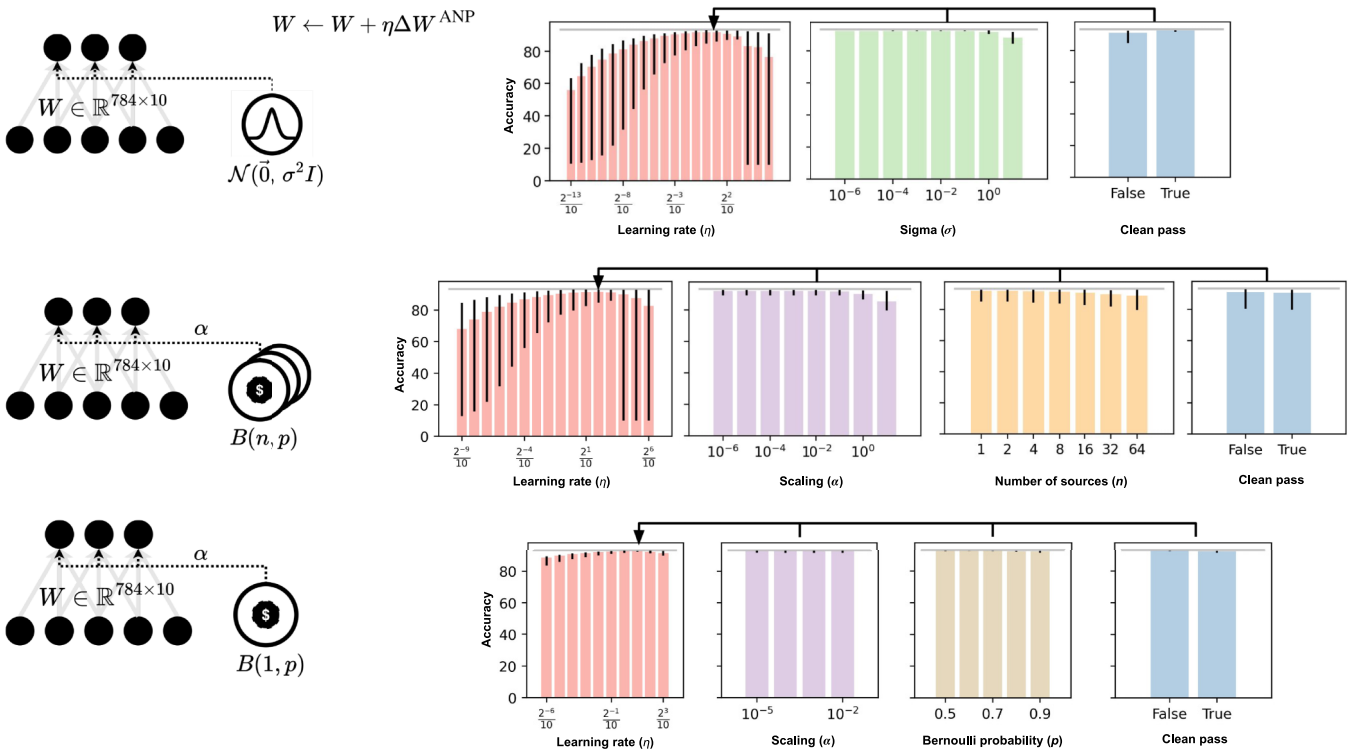

FIG. 3. Parameter robustness of ANP in a single linear-layer network trained on the MNIST dataset. The accuracies shown are the training accuracies reached after 100 epochs of training with the selected parameters using a categorical cross-entropy loss. Note that these results show slices of a parameter-sweep hypercube, such that the error bars show the maximum and minimum performance for a given parameter value as measured across all options for the *other* remaining hyperparameters. Thus, the error bars can be relatively large if they are susceptible to change from other hyperparameters. The hyperparameters include the learning rate (η), the noise variance (σ^2), a multiplicative scaling factor applied to the noise (α), the Bernoulli noise-source number (n), and the Bernoulli probability (p). The gray horizontal lines indicate the best BP performance based on the best simulation outcome when BP networks are trained for 100 epochs over learning rates $\eta \in \{0.1, 0.1 \times 2^{-1}, \ldots, 0.1 \times 2^{-9}\}$.

noise is sampled from a Gaussian or binomial distribution. Furthermore, in the regime of the proposed sMTJ setup (Bernoulli distribution), we can also observe successful learning. Note that the gray horizontal lines here correspond to the maximum accuracy of a BP-trained network after 100 epochs when trained across the same set of learning rates.

C. Training multilayer sMTJ networks

Next, we wished to determine whether multilayer neural networks can be trained with real-world sampled noise. We therefore applied our approach for training in multilayer perceptrons with noise sampled from a long recording session with a single sMTJ.

Figure 4 shows the results of these simulations, in which fully connected networks (comprising an input layer, three hidden layers of 500 nodes, and finally an output layer of 10 or 100 nodes) are trained using the DANP algorithm with noise sampled directly from a time series of sMTJ-generated noise. For this training, no clean passes were computed, and instead only noisy passes were computed for the DANP rule. Since backward gradient machinery is not feasible to implement in physical learning systems, we include the maximum values for the BP-trained variants of these networks as a theoretical optimum.

In these results, we can see that the DANP-trained simulations are capable of learning to a significant degree. Their performance does not reach as high as that of BP, but they still perform significantly above the 10% and 1% random-guess accuracy on the test set for CIFAR10 and CIFAR100, respectively. In particular, for CIFAR100, the test performance comes close to that of BP and is still increasing at the end of the simulation. For both CIFAR results, DANP has slightly higher accuracy in testing than in training. This can be attributed to the fact that the test set has no augmentations and the tendency for (D)ANP to generalize well, limiting overfitting on the training set [37].

Even in the extreme case, in which we use real-world data sampled from a single sMTJ and provide this as noise to each node sequentially, the learning closely follows that of the network with simulated sMTJ noise. When injecting this measured noise into the nodes of this system, subsequent samples of noise from that single sMTJ were taken as the injected noise, meaning that the networks experienced noise with high degrees of autocorrelation between neighboring units. This can be seen as a worst-case scenario for learning with real-world noise, and yet the networks are still shown to yield reasonable test accuracy. The fact that a state-of-the-art multilayer neural network can be trained without backward passes of the

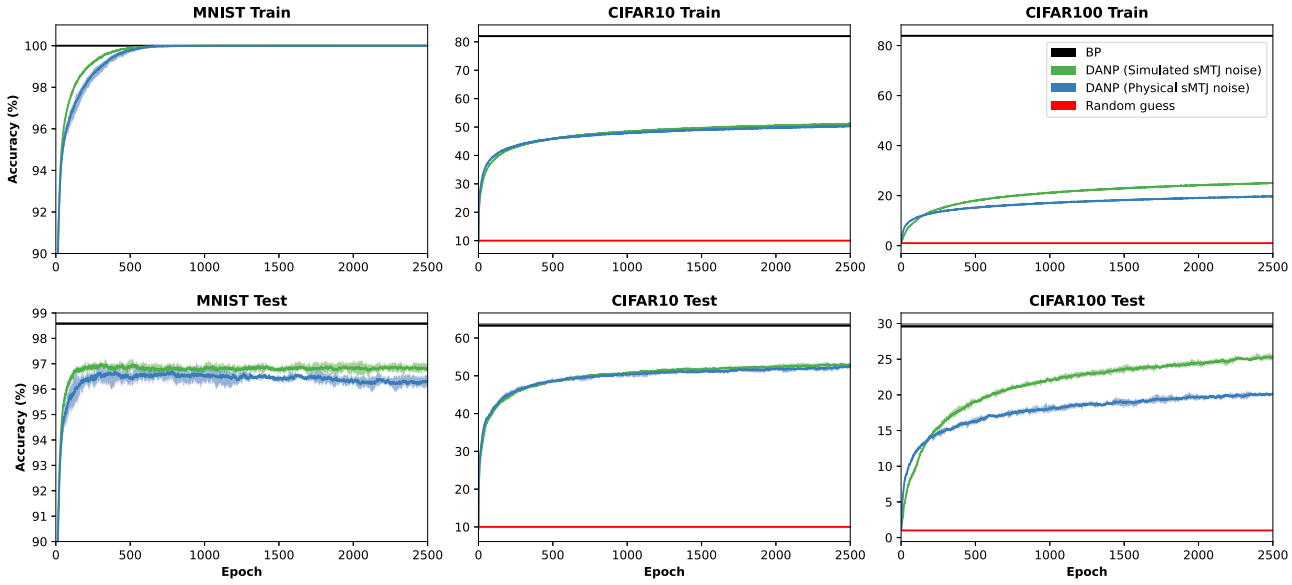

FIG. 4. Training convergence of multilayer neural networks trained by BP vs DANP with noise either simulated from multiple sMTJs or sampled from a real single sMTJ on three different tasks: MNIST, CIFAR10, and CIFAR100. For CIFAR, training samples have been augmented by randomly cropping and flipping them. These networks consist of the input layer, followed by three hidden layers of 500 units, and finally by an output layer of 10 or 100 units. The training parameters were chosen based upon a parameter sweep (see Appendix A). The maximum accuracy achieved by BP is shown using a black line as a theoretical upper bound, while the random-guess lower bound is shown in red. Envelopes represent the maximum and minimum accuracies across five randomly seeded repeats. Noise for the DANP simulations was either (1) simulated using an independent sMTJ noise source simulated for every node of a network, or (2) taken from data collected from a single sMTJ, such that the time-series data collected for the sMTJ were directly used as noise for nodes in a serial fashion. That is, each node is given noise in order from the sMTJ time series from the input layer of the network to the output layer. In all cases, the DANP algorithm operates with two noisy passes.

conventional, computationally expensive backpropagation algorithm, relying instead solely on noise injected from a real physical system, is encouraging—even under this limiting condition of a single noise source driving all units sequentially. Appendix A provides the hyperparameters and tuning process for these simulations.

D. Experimental validation

Finally, we demonstrate effective sMTJ learning applied within physical hardware. For this, we make use of the experimental setup shown in the inset of Fig. 5. We employ an Arduino Uno, which has 14-bit resolution on its analog pins to read the voltage level at the sMTJ and supply voltage via 3.3-V pins. The essential part is a voltage divider between a resistor of 100 kΩ and the MTJ, whose resistance is of the order of 3–6 kΩ. An MCP601 operational amplifier is used as a unity-gain buffer. The Arduino is set up with Telemetrix [38], which allows for direct communication via Python. The network runs within Python and reads the noise directly sampled by the Arduino. The noise is integrated via a noise-sampler function that saves a buffer array of 200 noise values, with new values continuously added and old ones removed. When a noise-sample call is received, the code waits for 1 ms before returning a shuffled list of the 200 stored values. The wait time is chosen to mimic the 1-kHz readout speed of the Arduino so that upon each call, at least one new value is present.

To quantify the learning performance, we tested the setup for its ability to approximate the input-output mapping of a ground-truth, randomly initialized, $2 \times 2 \times 2 \times 1$ network. A total of 100 samples was generated using this ground-truth network. Figure 5 demonstrates effective learning as weight changes reduce the loss, which is robust over 100 epochs, with small deviations for the loss after 50 epochs of training. With this, we demonstrate a simple and, in principle, scalable approach to reading sMTJs and using a physical system as a source of on-the-fly generated noise in our proof-of-concept hardware device.

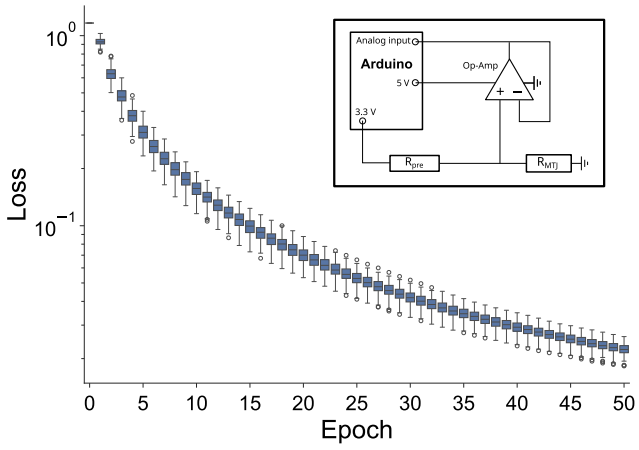

FIG. 5. Boxplot of the loss of an ANP network employing a single sMTJ as a noise source for 100 epochs. The whiskers show extremal values excluding outliers, while the boxes represent the 25th to 75th percentile values. A total of 100 samples was used for each epoch. The inset shows a schematic of the circuitry employed to read a single sMTJ using an Arduino Uno. The training parameters are reported in Appendix B.

IV. DISCUSSION

Our results demonstrate—both in simulations and an actual physical experiment—that learning based on physical noise is a viable strategy. We show that these results are robust under variations of normal and Bernoulli-distributed physical noise (see also Ref. [39] for a motivation for the use of the latter). Our findings suggest that our strategy may be generalizable to many physical noise sources, opening the door to noise-based learning in a wide range of physical systems.

A next step in the development of physical learning machines would be to incorporate not only the noise process but also the forward propagation of activity and input decorrelation, as well as the updating of parameters in the physical substrate. Importantly, activity-based node perturbation relies only on local operations, apart from a global feedback signal that encodes and scales the loss difference. Additionally, the iterative decorrelation is local at the layer level. These properties offer enormous potential for learning in physical hardware, reducing the circuitry required and leading to significant energy savings.

Beyond the current line of research, we also see potential for the further integration of noise-based learning approaches into noisy physical materials and noisy models. These include formulating noise-based learning within Hamiltonian system descriptions of physical systems, within sampling-based models for parallel sampling, and learning, among many other scenarios.

With the development of our advanced noise-based learning approach, we contribute to the development of a generation of intelligent systems that are directly realizable in a range of (noisy, analog) physics-based hardware implementations [40].

ACKNOWLEDGMENTS

We acknowledge support from the European Union, in particular from the Horizon 2020 Framework Program of the European Commission under FETOpen Grant Agreement No. 863155 (s-Nebula), ERC-2019-SyG Grant No. 856538 (3D MAGiC), and the Horizon Europe Project No. 101070290 (NIMFEIA). The group in Mainz acknowledges support from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Projects No. 403502522 (SPP 2137 Skyrmionics), 49741853, and 268565370 (SFB TRR173 projects A01, A12, and B02), as well as TopDyn and the Zeiss foundation through the Center for Emergent Algorithmic Intelligence. L.S. was

supported by the Max Planck Graduate Center with the Johannes Gutenberg-Universität Mainz (MPGC). J.H.M. acknowledges funding from VIDI Project No. 223.157 (CHASEMAG), which is financed by the Dutch Research Council (NWO). M.K. thanks the Research Council of Norway for support through its Centres of Excellence funding scheme, Project No. 262633 "QuSpin." M.G. and N.A. are supported by the Dutch Brain Interface Initiative (DBI2) under Project No. 024.005.022 of the research program Gravitation, which is financed by the Dutch Ministry of Education, Culture, and Science (OCW) via the Dutch Research Council (NWO).

DATA AVAILABILITY

The data that support the findings of this article are openly available [41].

APPENDIX A: SIMULATION-BASED HYPERPARAMETERS

To produce the simulations shown in Fig. 4, multilayer neural networks were constructed using PYTORCH. These consist of three fully connected hidden layers of 500 units following the input, and finally a readout layer of either 10 or 100 units. The Adam optimizer was used for both BP and DANP, with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. A categorical cross-entropy loss was used for training.

To determine optimal parameters, a parameter sweep was carried out over a range of learning rates for η and ϵ. For every (pair of) values, a single seed of the neural network was trained for 1000 epochs to find the parameters that produced highest test accuracy at the end of training. We selected on test accuracy to mitigate overfitting. The learning rates were tested from the set $\{0.1, 0.1 \times 2^{-1}, 0.1 \times 2^{-2}, \ldots, 0.1 \times 2^{-20}\}$. The final selected learning rates are shown in Table I.

For the simulated sMTJ simulations, sMTJ responses were simulated as a hidden Markov model (HMM) with two states and noisy observation. The transition probability between the two HMM states was computed statistically from our single sMTJ noise data and was symmetric ($p = 0.0809$), the states had two distinct mean value outputs ($\mu_1 = 0.0480$, $\mu_2 = 0.0362$), and the observation noise of these states was Gaussian (mean zero and $\sigma = 0.001$).

TABLE I. Hyperparameters corresponding to simulations shown in Fig. 4.

Learning algorithm	Dataset	η	ϵ
BP	MNIST	0.1×2^{-9}	–
BP	CIFAR10	0.1×2^{-14}	–
BP	CIFAR100	0.1×2^{-13}	–
DANP, simulated noise	MNIST	0.1×2^{-11}	0.1×2^{-19}
DANP, simulated noise	CIFAR10	0.1×2^{-12}	0.1×2^{-19}
DANP, simulated noise	CIFAR100	0.1×2^{-12}	0.1×2^{-20}
DANP, measured noise	MNIST	0.1×2^{-11}	0.1×2^{-19}
DANP, measured noise	CIFAR10	0.1×2^{-12}	0.1×2^{-19}
DANP, measured noise	CIFAR100	0.1×2^{-10}	0.1×2^{-14}

APPENDIX B: HARDWARE-BASED HYPERPARAMETERS

To showcase the possibility of live learning using physical sMTJs, we use a $2 \times 2 \times 2 \times 1$ network, as described in the main text. In this way, we tested a parameter sweep over the learning rates $\eta \in \{1, 5 \times 10^{-1}, 1 \times 10^{-1}, 7 \times 10^{-2}, 4 \times 10^{-2}, 1 \times 10^{-2}, 7 \times 10^{-3}, 4 \times 10^{-3}, 1 \times 10^{-3}, 5 \times 10^{-4}\}$ and $\epsilon \in \{1 \times 10^{-2}, 5 \times 10^{-3}, 1 \times 10^{-3}, 7 \times 10^{-4}, 4 \times 10^{-4}, 1 \times 10^{-4}, 7 \times 10^{-5}, 4 \times 10^{-5}, 1 \times 10^{-5}, 5 \times 10^{-6}\}$. The parameters finally selected to produce Fig. 5 were $\eta = 1 \times 10^{-3}$ and $\epsilon = 1 \times 10^{-4}$.

APPENDIX C: CODE AND REPRODUCIBILITY

The code used to produce the results in Fig. 4 is available on GitHub at https://github.com/artcogsys/MTJ-Node-Perturbation.git. The sMTJ time-series data are available on Zenodo [41].

[1] C. Kaspar, B. J. Ravoo, W. G. van der Wiel, S. V. Wegner, and W. H. Pernice, The rise of intelligent matter, Nature **594**, 345 (2021).
[2] D. Markovic, A. Mizrahi, D. Querlioz, and J. Grollier, Physics for neuromorphic computing, Nat. Rev. Phys. **2**, 499 (2020).
[3] B. Scellier and Y. Bengio, Equilibrium propagation: Bridging the gap between energy-based models and backpropagation, Front. Comput. Neurosci. **11**, 00024 (2017).
[4] M. Stern, D. Hexner, J. W. Rocks, and A. J. Liu, Supervised learning in physical networks: From machine learning to learning machines, Phys. Rev. X **11**, 021045 (2021).
[5] M. Nakajima, K. Inoue, K. Tanaka, Y. Kuniyoshi, T. Hashimoto, and K. Nakajima, Physical deep learning with biologically inspired training method: Gradient-free approach for physical hardware, Nat. Commun. **13**, 1 (2022).
[6] V. López-Pastor and F. Marquardt, Self-learning machines based on Hamiltonian echo backpropagation, Phys. Rev. X **13**, 031020 (2023).
[7] A. Momeni, B. Rahmani, M. Malléjac, P. del Hougne, and R. Fleury, Backpropagation-free training of deep physical neural networks, Science **382**, 1297 (2023).
[8] E. R. W. Van Doremaele, T. Stevens, S. Ringeling, S. Spolaor, M. Fattori, and Y. van de Burgt, Hardware implementation of backpropagation using progressive gradient descent for in situ training of multilayer neural networks, Sci. Adv. **10**, 8999 (2024).
[9] J. Kendall, R. Pantone, K. Manickavasagam, Y. Bengio, and B. Scellier, Training end-to-end analog neural networks with equilibrium propagation, arXiv:2006.01981 [cs.NE].

[10] S. Dillavou, M. Stern, A. J. Liu, and D. J. Durian, Demonstration of decentralized physics-driven learning, Phys. Rev. Appl. **18**, 014040 (2022).
[11] A. Laborieux and F. Zenke, in *NeurIPS* (Curran Associates, Inc., New Orleans, Louisiana, 2022), Vol. 36, pp. 12950–12963.
[12] V. R. Anisetti, B. Scellier, and J. M. Schwarz, Learning by non-interfering feedback chemical signaling in physical networks, Phys. Rev. Res. **5**, 023024 (2023).
[13] A. A. Faisal, L. P. J. Selen, and D. M. Wolpert, Noise in the nervous system, Nat. Rev. Neurosci. **9**, 292 (2008).
[14] G. Shimizu, K. Yoshida, H. Kasai, and T. Toyoizumi, Computational roles of intrinsic synaptic dynamics, Curr. Opin. Neurobiol. **70**, 34 (2021).
[15] K. Y. Camsari, R. Faria, B. M. Sutton, and S. Datta, Stochastic p-bits for invertible logic, Phys. Rev. X **7**, 031014 (2017).
[16] W. A. Borders, A. Z. Pervaiz, S. Fukami, K. Y. Camsari, H. Ohno, and S. Datta, Integer factorization using stochastic magnetic tunnel junctions, Nature **573**, 390 (2019).
[17] A. Grimaldi, L. Mazza, E. Raimondo, P. Tullo, D. Rodrigues, K. Y. Camsari, V. Crupi, M. Carpentieri, V. Puliafito, and G. Finocchio, Evaluating spintronics-compatible implementations of Ising machines, Phys. Rev. Appl. **20**, 024005 (2023).
[18] J. Si, S. Yang, Y. Cen, J. Chen, Y. Huang, Z. Yao, D. J. Kim, K. Cai, J. Yoo, X. Fong, and H. Yang, Energy-efficient superparamagnetic Ising machine and its application to traveling salesman problems, Nat. Commun. **15**, 3457 (2024).
[19] J. Kaiser, W. A. Borders, K. Y. Camsari, S. Fukami, H. Ohno, and S. Datta, Hardware-aware in situ learning based on stochastic magnetic tunnel junctions, Phys. Rev. Appl. **17**, 014016 (2022).
[20] S. Dalm, M. van Gerven, and N. Ahmad, Effective learning with node perturbation in deep neural networks, arXiv:2310.00965 [cs.LG].
[21] A. Dembo and T. Kailath, Model-free distributed learning, IEEE Trans Neural Netw. **1**, 58 (1990).
[22] G. Cauwenberghs, in *NeurIPS* (Morgan-Kaufmann, Denver, Colorado, 1992), Vol. 5.
[23] M. Lanza, A. Sebastian, W. D. Lu, M. L. Gallo, M.-F. Chang, D. Akinwande, F. M. Puglisi, H. N. Alshareef, M. Liu, and J. B. Roldan, Memristive technologies for data storage, computation, encryption, and radio-frequency communication, Science **376**, eabj9979 (2022).
[24] J. Grollier, D. Querlioz, K. Camsari, K. Everschor-Sitte, S. Fukami, and M. Stiles, Neuromorphic spintronics, Nat. Electron. **3**, 360 (2020).
[25] J. Zázvorka, F. Jakobs, D. Heinze, N. Keil, S. Kromin, S. Jaiswal, K. Litzius, G. Jakob, P. Virnau, D. Pinna, K. Everschor-Sitte, L. Rózsa, A. Donges, U. Nowak, and M. Kläui, Thermal skyrmion diffusion used in a reshuffler device, Nat. Nanotechnol. **14**, 658 (2019).
[26] K. Raab, M. A. Brems, G. Beneke, T. Dohi, J. Rothörl, F. Kammerbauer, J. H. Mentink, and M. Kläui, Brownian reservoir computing realized using geometrically confined skyrmion dynamics, Nat. Commun. **13**, 6982 (2022).
[27] S. Kanai, K. Hayakawa, H. Ohno, and S. Fukami, Theory of relaxation time of stochastic nanomagnets, Phys. Rev. B **103**, 094423 (2021).
[28] A. G. Baydin, B. A. Pearlmutter, D. Syme, F. Wood, and P. Torr, Gradients without backpropagation, arXiv:2202.08587 [cs].
[29] M. Ren, S. Kornblith, R. Liao, and G. Hinton, in *ICLR* (OpenReview.net, Kigali, Rwanda, 2023), Vol. 11.
[30] N. Ahmad, E. Schrader, and M. van Gerven, Constrained parameter inference as a principle for learning, Trans. Mach. Learn. Res. (2023).
[31] L. Schnitzspan, M. Kläui, and G. Jakob, Nanosecond true-random-number generation with superparamagnetic tunnel junctions: Identification of Joule heating and spin-transfer-torque effects, Phys. Rev. Appl. **20**, 024002 (2023).
[32] K. Hayakawa, S. Kanai, T. Funatsu, J. Igarashi, B. Jinnai, W. Borders, H. Ohno, and S. Fukami, Nanosecond random telegraph noise in in-plane magnetic tunnel junctions, Phys. Rev. Lett. **126**, 117202 (2021).
[33] C. Safranski, J. Kaiser, P. Trouilloud, P. Hashemi, G. Hu, and J. Z. Sun, Demonstration of nanosecond operation in stochastic magnetic tunnel junctions, Nano Lett. **21**, 2040 (2021).
[34] L. Néel, in *Ann. Geophys.* (Centre National de la Recherche Scientifique, Paris, 1949), Vol. 5, p. 99.
[35] L. Schnitzspan, J. Cramer, J. Kubik, M. Tarequzzaman, G. Jakob, and M. Kläui, Impact of annealing temperature on tunneling magnetoresistance multilayer stacks, IEEE Magn. Lett. **11**, 1 (2020).
[36] L. Schnitzspan, M. Kläui, and G. Jakob, Electrical coupling of superparamagnetic tunnel junctions mediated by spin-transfer-torques, Appl. Phys. Lett. **123**, 232403 (2023).
[37] J. G. Fernández, S. Keemink, and M. van Gerven, Gradient-free training of recurrent neural networks using random perturbations, Front. Neurosci. **18**, 1439155 (2024).
[38] A. Yorinks, The Telemetrix Project, https://mryslab.github.io/telemetrix, accessed: 2010-09-30.
[39] G. Belouze, Optimization without backpropagation, arXiv:2209.06302v1 [cs.LG].
[40] H. Jaeger, B. Noheda, and W. G. van der Wiel, Toward a formal theory for computing machines made out of whatever physics offers, Nat. Commun. **14**, 1 (2023).
[41] F. Kammerbauer and L. Schnitzspan, Time series sMTJ switching data, https://doi.org/10.5281/zenodo.15222667 (2025).