Article:
Ellis, M.O.A., Welbourne, A., Kyle, S.J. et al. (2023) Machine learning using magnetic stochastic synapses. Neuromorphic Computing and Engineering, 3 (2). 021001. ISSN: 2634-4386

https://doi.org/10.1088/2634-4386/acdb96

Machine learning using magnetic stochastic synapses

NEUROMORPHIC
Computing and Engineering

LETTER

Machine learning using magnetic stochastic synapses

Matthew O A Ellis[1,4,*], **Alexander Welbourne**[2,4], **Stephan J Kyle**[2], **Paul W Fry**[3], **Dan A Allwood**[2], **Thomas J Hayward**[2] **and Eleni Vasilaki**[1]

[1] Department of Computer Science, University of Sheffield, Sheffield S1 4DP, United Kingdom
[2] Department of Materials Science and Engineering, University of Sheffield, Sheffield S1 3JD, United Kingdom
[3] Department of Electronic and Electrical Engineering, University of Sheffield, Sheffield S1 3JD, United Kingdom
[4] These authors contributed equally to this work.
[*] Author to whom any correspondence should be addressed.

E-mail: m.o.ellis@sheffield.ac.uk

Keywords: neuromorphic, magnetic nanowire, binary stochastic synapses, gradient rule, spintronics

OPEN ACCESS

RECEIVED
9 March 2023

REVISED
30 May 2023

ACCEPTED FOR PUBLICATION
5 June 2023

PUBLISHED
23 June 2023

Abstract

The impressive performance of artificial neural networks has come at the cost of high energy usage and CO_2 emissions. Unconventional computing architectures, with magnetic systems as a candidate, have potential as alternative energy-efficient hardware, but, still face challenges, such as stochastic behaviour, in implementation. Here, we present a methodology for exploiting the traditionally detrimental stochastic effects in magnetic domain-wall motion in nanowires. We demonstrate functional binary stochastic synapses alongside a gradient learning rule that allows their training with applicability to a range of stochastic systems. The rule, utilising the mean and variance of the neuronal output distribution, finds a trade-off between synaptic stochasticity and energy efficiency depending on the number of measurements of each synapse. For single measurements, the rule results in binary synapses with minimal stochasticity, sacrificing potential performance for robustness. For multiple measurements, synaptic distributions are broad, approximating better-performing continuous synapses. This observation allows us to choose design principles depending on the desired performance and the device's operational speed and energy cost. We verify performance on physical hardware, showing it is comparable to a standard neural network.

1. Introduction

The meteoric rise of artificial intelligence (AI) as a part of modern life has brought many advantages. However, as AI programs become increasingly more complex, their energy footprint becomes larger [1, 2], with the training of one of today's state-of-the-art natural language processing models now requiring similar energy consumption to the childhood of an average American citizen [3]. Several non-traditional computing architectures aim to reduce this energy cost, including non-CMOS technologies [4–7]. However, competitive performance with non-CMOS technologies requires overcoming the latent advantage of years of development in CMOS.

In biological neural networks, synapses are considered all-or-none or graded and non-deterministic, unlike the fully analogue synapses modelled in artificial networks [8]. Inspired by biology, several approaches have considered networks with binary synapses and neurons, with the view that binary operations are simpler to compute and thus lower energy [9–12]. However, while these binarised neural networks are more robust to noise, they suffer from lower performance than analogue versions. In contrast, networks with stochastic synapses provide sampling mechanisms for probabilistic models [13] and can rival analogue networks at the expense of long sampling times [14–19]. Adapted training methods are required to provide higher performance for a lower number of samples, while implementations require hardware that can natively (with low energy cost) provide the stochasticity required. Magnetic architectures are one possible route for such unconventional computing. They have long promised a role in computing logic following the strong interest in the field stemming from the data storage market [6, 7, 20–26]. The non-volatility of

magnetic elements naturally allows for data storage, while ultra-low-power control mechanisms, such as spin-polarised currents or applied strain [27, 28] offer routes towards energy-efficient logic-in-memory computing. Recent demonstrations of spin-orbit torque based devices have shown how magnetic materials can be used as both binary and multi-level synapses for efficient neuromorphic systems [29–33]. Meanwhile, ongoing developments have shown how to manipulate magnetic domains to both move data and process it [22, 24, 34, 35]. However, magnetic domain wall (DW) logic is limited by stochastic effects, particularly when compared to the low error tolerance environment of CMOS computing [36, 37].

Here, we propose a methodology where, rather than seeking to eliminate stochastic effects, they become a crucial part of our computing architecture. As a proof of concept, we demonstrate how a magnetic nanowire (NW) is usable as a stochastic synapse able to perform handwritten digit recognition using multiplexing of one of the hardware synapses. We have developed a learning rule that can effectively train artificial neural networks made of such 'noisy' synapses by considering the synaptic distribution. When a single measurement is used to identify the state of the synapse the learning rule will adjust its parameter, i.e. the field at which the wall is propagated, to reduce the synaptic stochasticity. However, if multiple measurements are taken, the gradient rule will find parameters that allow for a broad synaptic distribution, mimicking a continuous synapse and improving performance. Without the stochasticity, the operation would be limited to binary operations, which lack the resolution power of analogue synapses. With stochasticity, we have a flexible system tunable between quick-run-time approximation and long-run-time performance. Our learning rule provides efficient network training despite the high or variable noise environment and differs from other stochastic neural network computing schemes that employ mean-field-based learning rules [14, 16, 19]. Here, the inclusion of the network variance allows the training to find better solutions in low sampling regimes, providing a trade-off between operational speed/energy cost and test accuracy.

We have verified the model performance experimentally by transferring the trained weights to a network utilising such a hardware synapse, with excellent agreement between the experimental performance and that of a simulated network. Our observations allow for a design framework where we can identify the number of required measurements (and hence energy requirements) for a given desired accuracy and vice versa.

This work opens up the prospect of utilising the low-energy-cost benefits of spintronic-based logic [5–7, 38]. In particular, it enables the use of DW-based NW devices [24, 35, 39, 40] whilst transforming the hitherto hindrance of noisy operation [36, 37] into the basis of a high-performance stochastic machine learning paradigm.

2. Results

2.1. Hardware stochastic synapse

Our proposed elementary computation unit is a binary stochastic synapse based on a ferromagnetic NW with two favourable magnetic orientations; parallel and antiparallel to the NW. The transitions between regions of differing magnetisation orientation are known as DWs. While different forms of DWs exist, here they form a 'vortex' pattern with a cyclical magnetisation texture. Our synapse was a 400 nm wide, 54 nm thick permalloy NW with notches patterned halfway along its length to create an artificial defect site. Figure 1(a) shows an SEM image of the system, with the inset enlarging the notch. DWs were nucleated at the left-hand side of the wire (false-coloured blue) by applying a voltage pulse across a gold current line (false-coloured orange).

The operation of this system as a stochastic synapse is described schematically in figure 1(b). A vortex DW [41] can be injected into the wire by applying a current pulse in the line. This corresponds to presenting the synapse with an input of 1, while no DW injection corresponds to an input of 0. An applied magnetic field is used to propagate the DW along the length of the wire. If the propagation field is sufficiently high, the DW does not pin at the defect site and can pass to the end of the wire, resulting in an output of 1. If the propagation field is low, the DW is pinned at the notch, resulting in an output of 0. For intermediate values of the field, the behaviour becomes stochastic but with a well defined pinning probability. Figure 1(c) shows the probability of not pinning (i.e. passing probability), as measured using the focused Magneto-Optical Kerr effect, as a function of the propagation field. The probability of passing behaves in a sigmoid-like manner, and the orange dashed line shows a fit using a logistic sigmoid function $f(h_{ij})$ (see methods). We can consider the propagation field as controlling the weight in a binary synapse with detecting a DW on the right hand side of the NW as the output of the synapse.

Therefore, our binary stochastic synapse is determined by

$$w_{ij} = \begin{cases} 1 & \text{with probability } f(h_{ij}) \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

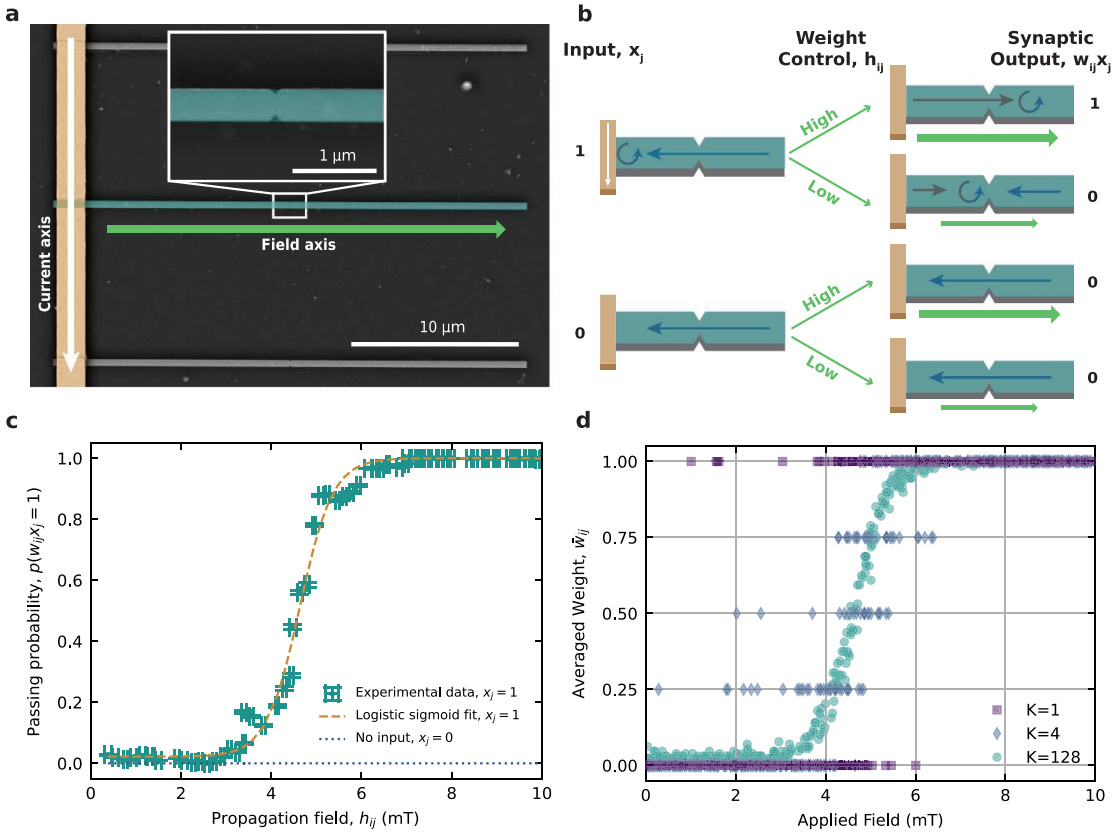

Figure 1. Characterisation of a notched permalloy nanowire (NW) as a stochastic synapse. (a), False-coloured SEM image of the permalloy NW (blue) and current injection line (orange). The inset shows detail of the artificial notch. The field (green) and current (white) axes are marked. (b), Schematic of the operating principle of the stochastic synapse. The current line allows input (x_j) of 1 (current pulse, DW injected) or 0 (no pulse, no DW). Field inline with the NW drives (if present) the DW through the system: high fields pass the DW through the notch and produce an output of 1, low fields result in the notch blocking the DW and an output of 0. Intermediary fields (not shown) provide intermediate probabilities of passing the notch. (c), Experimentally measured probability of an injected domain wall passing the notch. Tuning the propagation field can control this probability across the whole range in a logistic sigmoid-like fashion. Points are averages of 1000 samples, x error bars represent precision in choice of propagation field, and y error bars are given $p(1-p)/\sqrt{1000}$. The logistic sigmoid fit is given in methods. The nucleation field with no input (no injection, $x_j = 0$) is (10.74 ± 0.07) mT. Therefore, below 10 mT the passing probability for no input is zero. (d), Average synaptic weight for randomly selected propagation fields, as defined in equation (2). Depending on the number of samples, i.e. repetitions of the operation in (b), the effective synaptic weight varies from purely binary (one repetition/sample, $K = 1$) to almost continuous ($K = 128$ samples).

where $f(h_{ij})$ is the DW passing probability function, h_{ij} is the propagation field for the synapse connecting input neuron j with output neuron i. Through this definition our synapses are purely excitatory, which corresponds to the physical representation of a magnetic DW being pinned or not, rather than the complementary binary scheme, with values $\{-1, 1\}$, which is not naturally represented by the physical system. In general, the DW motion in the NW is complex [36] and the position of the notch relative to the perturbation of the DW structure may alter the pinning process. However, since the notches cover 60% of the NW width it is likely that the pinning probability remains relatively independent of the position of the notches in the NW.

Compared to binary synapses, neural networks with analogue or graded synapses tend to perform better due to the wider range of states [42, 43]. Here, we adopt a scheme similar to that of stochastic computing, where the average of a series of binary measurements or samples are used to represent a value. Thus, we allow for $K \geqslant 1$ measurements to identify the state of a synapse and denote the equivalent mean weight as

$$\bar{w}_{ij} = \frac{1}{K} \sum_{k} w_{ij}^{(k)}, \tag{2}$$

where K is the total number of samples taken and the superscript (k) indicates the individual sampling of the synaptic weights as per equation (1). The mean synapse has $K+1$ states, e.g. for $K = 1$ the two states will be 0 and 1, while for $K = 2$ the states will be 0, 0.5, and 1. It follows that for $K \to \infty$, $\bar{w}_{ij}$ will be equivalent to a sigmoidally-shaped continuous synapse, bounded between 0 and 1. An example demonstrating the average weight as a function of the number of samples can be seen in figure 1(d), where we plot equation (2) for

$K = 1$ (purple squares), 4 (blue diamonds) and 128 (green circles). Each example is calculated by sampling w_{ij} the desired number of times with a fixed h_{ij} that was selected randomly. In each case only discrete levels are available but when $K = 128$ the sampling is sufficient to provide an almost continuous representation.

In this way, our proposed binary stochastic synapse can be used to construct neural networks that will approach a bounded analogue network when multiple samples are taken. Physically, this is achieved by repeated operation of the hardware devices to accumulate the average values. Instead, to avoid the additional complexity of storing and summing values this could be achieved by physical replication of the synapses.

2.2. Stochastic network

We embed these synapses in an artificial neural network where the output of neuron i is given by

$$y_i = \sum_j \bar{w}_{ij} x_j, \tag{3}$$

where j is an index over the input dimension. The summation here implies that the DW pinning occurs in independent nanowires. If the DW was to interact with multiple pinning sites on the same wire it would instead perform a multiplication of the pinning probabilities. This multiplicative operation would have uses for encoding the input as a non-binary value but we restrict our approach here to binarised inputs.

We trained the network as a classifier for a problem of C classes with C independent neurons (perceptrons) each representing one class. This task was based on the well-known MNIST dataset but with each image downsampled to give images with a shape of 14 by 14 pixels instead of the standard 28 by 28. This was necessary to reduce the time of the operation when running on the prototype experimental hardware (see methods). In figure 2(a) we depict the perceptron that corresponds to class '0'. If we present to the neuron a representative of its corresponding class (in this case an image of the digit '0'), the neuron should produce a high activity for recognising the input as zero.

The experimental process is shown in figure 2(b). For ease of demonstration, only a single hardware synapse is used, with operations serialised in time. Potential devices would have multiple synapses running in parallel with a summation performed during the measurement. The perceptron parameters are stored on a computer, which sends the input and synaptic parameter to the external hardware synapse and requests the result. The process is repeated until K samples per synapse (see equation (2)) are collected. Summation of the results takes place on the computer with an additional bias term applied. To avoid redundant measurements, pixels corresponding to inputs of '0' (white pixels in our example image) were omitted, since the output is deterministically '0' by design. A synapse receiving a black pixel ($x_j = 1$) will produce '1' if the field is set at a high value or '0' if the field is set at a low value, see figure 2(c). Intermediate field values will produce outputs that vary stochastically, reflecting the passing probability $f(h_{ij})$. Using this experimental approach, approximately 50 synaptic operations are measured per second. Since the repeated sampling occurs sequentially the inference speed scales linearly with the total number of samples required.

2.3. Analysis of the stochastic learning rule

We now sketch the derivation of the learning rule that we apply to the synapses of the neural network. Each synapse $w_{ij}^{(k)}$ is an independent sample from a Bernoulli distribution, and therefore the sum of these samples will follow a Poisson-Binomial distribution. The mean, μ_i, and variance, σ_i^2, for each output neuron (calculated by equation (3)) are given by:

$$\mu_i = \sum_j f(h_{ij}) x_j, \tag{4}$$

$$\sigma_i^2 = \frac{1}{K} \sum_j f(h_{ij}) x_j \left[1 - f(h_{ij}) x_j\right]. \tag{5}$$

For a detailed calculation of these values see the appendix.

Given the number of inputs and the sampling process this sum will be over a large number of events, which means the Poisson-Binomial distribution can be approximated by the Gaussian distribution [44]. Using this approximation, the neuronal output can be re-parameterised so that the stochasticity is only in a term with no dependence on the trainable parameters. In this way, we write

$$\tilde{y}_i = \mu_i + \sigma_i \xi_i, \tag{6}$$

where $\tilde{y}_i$ denotes the approximation of neuronal output y_i and ξ_i is a sample from a Gaussian distribution with zero mean and unit variance.

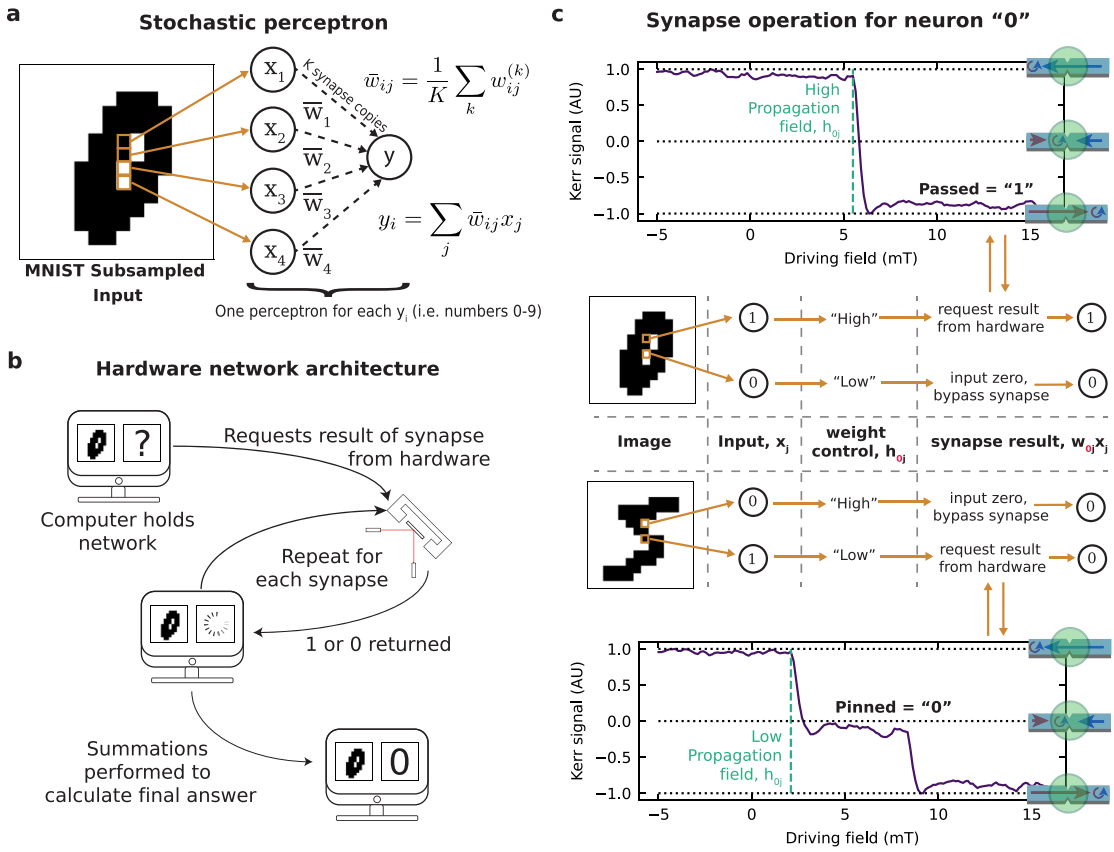

Figure 2. Stochastic network operation. (a), Sketch of the stochastic perceptron. Each input value from an image is fed via a mean weight to the neurons for each class. Here, the weighted inputs are summed to give the neuron's activity (as in equation (3)). In the case of the MNIST task, there is a neuron for each of the ten classes (numbers '0'–'9'; y_0 to y_9). When trained, the neuron for the class corresponding to the correct input, here y_0, should have the highest activity. Each mean weight in our network ($\bar{w}_{ij}$) is the average of multiple measurements ($K \geqslant 1$) of the output of a synapse with individual weight w_{ij} set by its trained propagation field (see equations (1) and (2)). A clear distinction should be made here from traditional neural networks that these weights are stochastic and will vary for each run of the network. The individual weights take the value '1' with the probability $f(h_{ij})$ (DW passing probability, as characterised in figure 1(c)) or '0' otherwise. The mean weights, therefore, take values from the distributions shown in figure 1(d). (b), The architecture of the hardware network. For the purpose of demonstrating successful performance, only the stochastic synapses are run directly on the hardware. The perceptrons are stored on a computer, which requests results ('1' or '0') from the magnetic stochastic synapses for a given synaptic parameter (trained propagation field, h_{ij}). After this is repeated for each synapse, summations are performed to predict the correct class of the input. (c), Idealised operation of single synapses *in materia* for the neuron y_0. The data path is shown for two inputs, or pixels, for the case of a correct image for the class ('0') and an incorrect image ('5'). The value of the weight control, the propagation field h_{ij}, is expected to be correlated with pixels in images from the correct class: where the pixels are 'on' for correct images, high values of the weight control are expected; when 'off', low values. If the input pixel value is '0', the synapse is bypassed as the result is '0' by construction. However, if it is '1', a result is requested from the hardware using the corresponding weight control. As shown in the top graph, high propagation fields result in the DW directly passing the notch (only a single step is seen) which is interpreted as an output of '1'. As in the lower graph, low propagation fields result in a two step procedure where the DW initially pins at the notch before depinning at a higher driving field. This is interpreted as a '0'. In practice, the results from the synapses will vary stochastically reflecting the passing probability $f(h_{ij})$.

If we assume that we are in a supervised learning framework and that E is the error function we would like to minimise (e.g. square mean error or cross-entropy), then E is a function of the pattern p we present to the network, which defines the desirable output target. E is also a function of the output neurons, represented by vector y, which also depends on p. The learning rule will update the values of the applied field to each synapse h_{ij} by Δh_{ij} according to the following 'online' gradient rule:

$$\Delta h_{ij} = -\eta \frac{\partial E(\tilde{y})}{\partial \tilde{y}_i} \frac{\partial \tilde{y}_i}{\partial h_{ij}} \tag{7}$$

$$= -\eta \frac{\partial E(\tilde{y})}{\partial \tilde{y}_i} \left(\frac{\partial \mu_i}{\partial h_{ij}} + \frac{\partial \sigma_i}{\partial h_{ij}} \xi_i \right), \tag{8}$$

where η is a small positive number representing the learning rate. We calculate the derivative of $\frac{\partial \tilde{y}_i}{\partial h_{ij}}$ from equations (6), (4) and (5). We also calculate the value ξ_i using equation (6), computing μ_i and σ_i from equations (4) and (5) and $\tilde{y}_i$ from equation (3) (setting $\tilde{y}_i = y_i$). It follows that for $K \to \infty$, $\sigma \to 0$ and we

obtain a 'mean-field' gradient rule that takes into account the mean but not the variance of the output neurons.

We have tested the performance of this rule on the downsampled MNIST dataset. During training, the number of repeats (samples) K is set as a parameter of the network, which we define as K_{train}, and as such modifies how the training progresses. The variance of the output has an important effect on the classification procedure; if the variance is high then mis-classification will be more likely, especially in classes that have similar mean values for each neuron. Therefore, during supervised training the network aims to minimise this variance. When K is low, this happens through changing the weights, controlled through the magnetic fields h_{ij}, so that the probabilities are close to either 1 or 0 (high or low applied field), as this minimises the single sample variance in equation (5). This leads to a solution that is almost a deterministic binary network. However, if K is large then the variance is reduced by the factor $1/K$ and therefore the system can tolerate higher synaptic variance than in the case of $K = 1$. Thus, a pseudo-analogue solution can be found.

Figure 3 describes the effect of the learning rule on the network synapses. We plot the probability distribution, calculated as a normalised histogram, of the propagation fields, h_{ij}, over all the neurons from 5 independent models before training (figure 3(a)), after training with $K_{\text{train}} = 1$ sample (figure 3(b)) and after training with $K_{\text{train}} = 128$ samples (figure 3(c)). The final distributions confirms the theoretical expectation that $K_{\text{train}} = 1$ leads to a binary network (low variance) while $K_{\text{train}} = 128$ approximates a standard perceptron with a continuous distribution of synaptic weight (high variance).

In figures 3(e) and (f) we show the distributions of the neuronal output when presented with the same image repeatedly for the three training cases above and find that the neuronal distribution reflects the synaptic distribution. We now consider the case where during testing a different number of samples are drawn when calculating equation (2), which we define as K_{test}. The top row shows the distribution when $K_{\text{test}} = 1$, while the bottom row shows $K_{\text{test}} = 128$. The untrained neuron values exhibit a Gaussian distribution across all data samples, with fields initialised to give the largest possible variance; see figure 3(d). After training, with $K_{\text{train}} = 1$ or $K_{\text{train}} = 128$ and $K_{\text{test}} = K_{\text{train}}$ the distributions of the correct and incorrect class neurons minimally overlap. However, if we test the network with $K_{\text{test}} = 1$ after we train it with $K_{\text{train}} = 128$ there is a rather significant overlap (figure 3(f), upper panel) suggesting a high probability of miss-classification. In all cases, when testing with $K_{\text{test}} = 128$ (bottom row) the variance is reduced $1/128$ as given in equation (5) and allows for better resolution of the mean values. In the case $K_{\text{train}} = 128$, the learning rule has exploited this additional sampling and variance reduction by better utilising a continuous range of weights to boost performance. However, when $K_{\text{test}} < 128$ (as in figure 3(f), upper panel), the increase in variance decreases the probability of correct classification. In the other training case ($K_{\text{train}} = 1$, figure 3(e), upper panel), the learning rule adapts the weights to find a low variance, almost deterministic binary, solution. Further sampling during testing (figure 3(e), lower panel) reduces this variance further, as expected, but does not significantly change the overlap as it has already been optimised for the lower sampling regime. As we will show, this leads to higher performance when test sampling (K_{test}) is small, but capped high performance when test sampling is allowed to rise, in contrast to the large K_{train} case.

Figure 3(g) compares the average variance during testing with $K_{\text{test}} = 1$ samples (circles) and $K_{\text{test}} = K_{\text{train}}$ samples (squares) as a function of the number of samples used during training, K_{train}. As discussed before, when training with 1 sample the variance is kept low by having passing probabilities close to 0 or 1. However, when more samples are used during training, the variances for a single sample can increase as the variance of the averaged samples decreases.

This behaviour of minimising the variance to reduce miss-classification arises due to the variance term in the 'stochastic' learning rule. Other rules that only consider the mean term [14, 19] cannot find these deterministic solutions when using a stochastic network. Figure 3(h) shows the test accuracy with $K_{\text{test}} = K_{\text{train}}$ as a function of K_{train} samples for our stochastic learning rule (squares) vs the mean field rule (circles) averaged over five independently trained models. For both rules, increasing the number of samples leads to an improvement in the test accuracy as more levels are possible for the synapse averages (see figure 1(d)). However, in all cases, the stochastic learning rule out performs the mean-field rule, with convergence when a large number of samples ($K \geqslant 8$) is used for training and testing. The dashed line shows the performance for a fully mean-field network, where effectively an infinite number of samples are taken (i.e continuous but bounded synapses), and represents the best possible accuracy for such a network given the task.

2.4. Hardware and operational principles

We now proceed to demonstrate our neural networks working on physical hardware and not only within simulation. Figures 4(a) and (b) shows the test accuracy computed when the synaptic operation has been simulated (lines) and processed using the hardware (points) for models trained with either a, $K_{\text{train}} = 1$ or b,

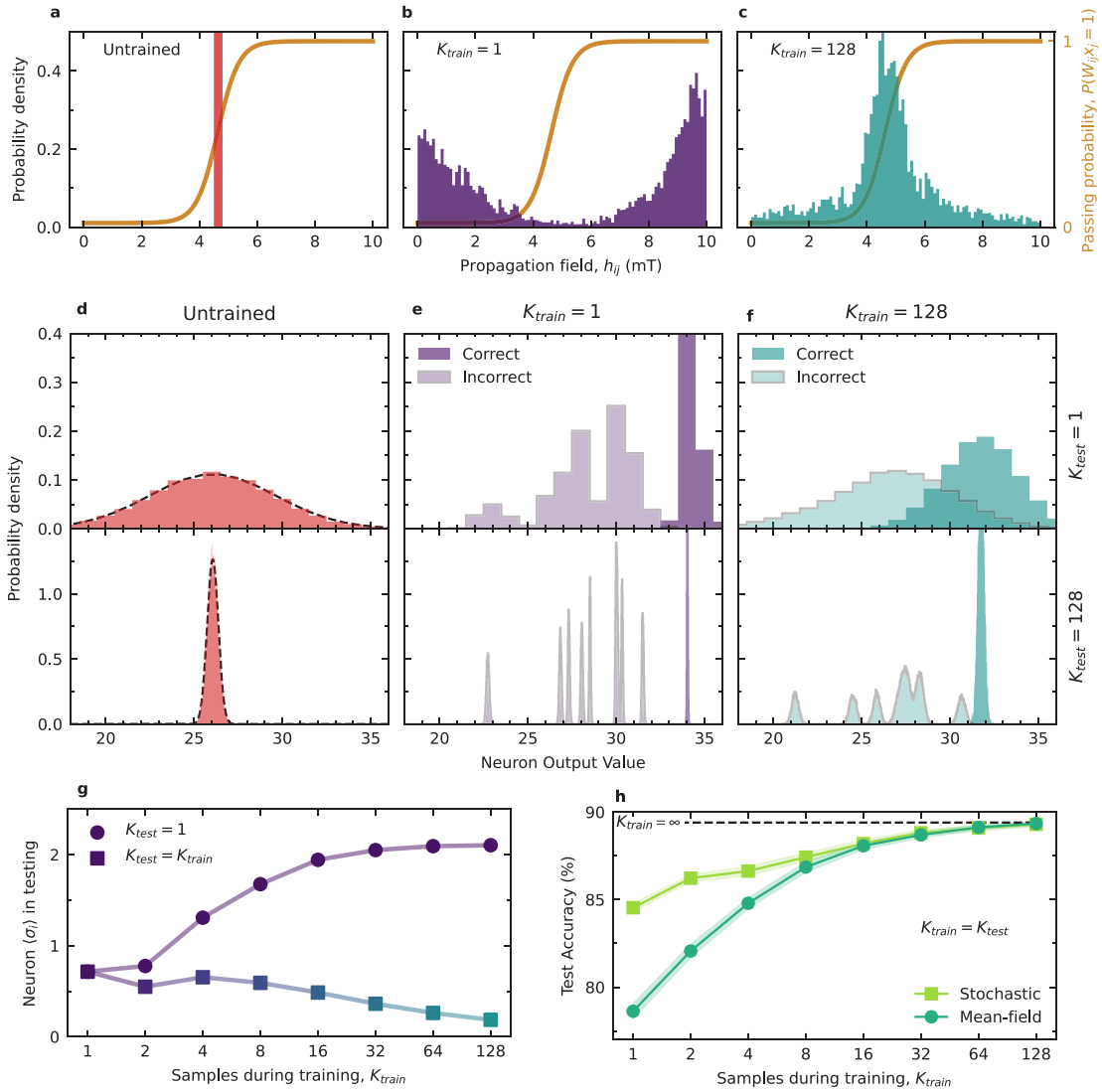

Figure 3. Analysis of the stochastic learning rule. (a)–(c), Probability density histograms of synaptic magnetic field parameters over 5 independent models. (a) shows the distribution before training, where all the fields are initialised so that the passing probability (shown in orange, right hand axis) is 0.5. (b) and (c) show the distributions when trained using 1 or 128 samples respectively. With 1 sample, the distribution is bimodal with peaks at fields with probabilities close to 0 or 1. While when training with 128 samples, the distribution is focused on the central region of the passing probability function. (d)–(f), Distribution of the neuron values **y** when an image of a zero is shown 10 000 times independently for neurons either identifying the correct (output 0 in this example) or incorrect (outputs 1–9) classification. In (d) the model is untrained so all outputs have the same distribution, while in (e) and (f) the distribution is split into the correct output neuron and the incorrect output neurons when training with 1 and 128 samples respectively. The top row shows $K_{\text{test}} = 1$ while the bottom row shows $K_{\text{test}} = 128$. Using more samples during testing reduces the variance and therefore the chance of mis-classification. (g), The standard deviation averaged over all the neurons when increasing number of samples are used in training, with $K_{\text{test}} = 1$ (circles) and $K_{\text{test}} = K_{\text{train}}$ (squares). This summarises the conclusions from the distribution plots in (d)–(f). More samples during training allows the standard deviation for a single sample to increase as the standard deviation over all samples is reduced. However, testing with $K_{\text{test}} < K_{\text{train}}$ results in an increased overall standard deviation. (h), Accuracy on the test set against number of samples during training when using the stochastic (dark green circles) or the mean-field (light green squares) learning rules. The points show the accuracy averaged over 5 independently trained models, while the shaded region indicates 1 standard deviation. The stochastic learning rule maintains a higher test accuracy when the number of samples is low.

$K_{\text{train}} = 128$ and tested with increasing numbers of samples (K_{test}). Due to the throughput of our prototype device, we only demonstrate experimental results up to $K_{\text{test}} = 8$.

The simulation and hardware results show excellent agreement and highlight different behaviours in models trained with different sampling levels. In the case trained on one repeat, the network is deterministic and as such the accuracy does not significantly improve when we average over more samples during testing. On the other hand, while the model trained with 128 repeats shows a lower performance with only one testing repeat, the accuracy improves as we increase the number of samples during testing. This arises from the increased stochasticity at low sampling levels and resultant increased precision at high sampling levels. This behaviour is corroborated by the corresponding neuronal distributions (figure 3(e)) and (figure 3(f)),

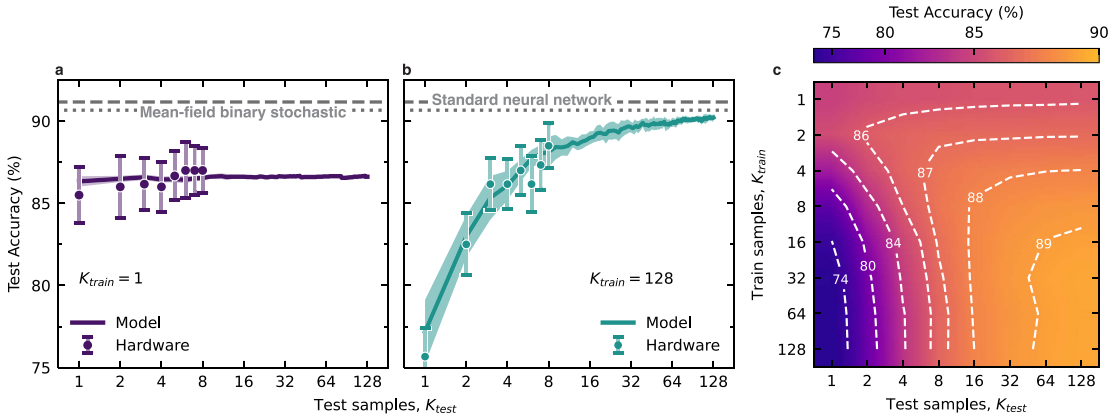

Figure 4. Hardware verification and choice of sampling. (a) and (b) Comparison of the testing accuracy computed using either the physical hardware (points) or from simulation (curves). The test data set is restricted to the first 600 images with an approximately equal balance of digits. In both, training was done using the model, and hardware testing was limited to eight samples due to throughput limitations of the prototype device. (a) shows the accuracy when the network was trained with only 1 sample while (b) was trained with 128 samples. As before, training with 1 sample reaches an almost deterministic solution, so repeated sampling during testing does not improve the accuracy. Training with 128 shows an increase in accuracy as more samples are used during testing, reaching higher peak performance (albeit with a lower initial base). The dashed line shows the performance on a standard neural network and the dotted-dashed is for a full mean-field binary stochastic network. In both cases the hardware performance shows excellent agreement with the model calculations. The model accuracy is averaged over 5 independent tests with the same trained weights, with the shaded area showing 1 standard deviation. This can be taken to represent the variability in performance for a given task due to the inherent stochasticity of the network. Naturally, it decreases as the number of test samples increases and is lower for the, more deterministic, $K_{\text{train}} = 1$ case. The hardware accuracy is from a single run over the 600 images, so the error bars show the standard error of the estimation of the accuracy over the mini-batches. (c) Test accuracy (as measured with the model) over different combinations of training and testing sampling for the sub-sampled MNIST task. The data is bi-linearly interpolated, which can be considered as averaging over fractions of the data set with different sampling rates. In general, testing with more samples increases accuracy, but, this is limited when $K_{\text{test}} > K_{\text{train}}$. In particular, in the $K_{\text{train}} = 1$ case, further sampling provides little improvement due to the deterministic weight distributions. Training with two samples is better in all test cases than when training with 1, but best overall accuracy is when 128 samples are used in both training and testing. Data such as this provide a guide to choosing training and testing samples depending on desired accuracy and operation times for a given task.

which show that the neuronal variance when training with one sample and testing with one sample is much lower than in the case of training with $K = 128$ samples and testing with one sample. It is akin to majority voting, where classifiers have to be diverse to improve performance (see [45] and references therein). Here, performance increase increases with increasing K_{test} (number of voters) when the neuronal distribution has a high variance.

Figure 4(c) allows further interrogation of the majority voting behaviour. It presents (using the now verified simulation model) a colour plot of the test accuracy as a function of the number of training and test samples. This variation in performance when testing using a different number of repeats raises an essential trade-off in speed vs accuracy. To a first approach, the results follow the behaviour of stochastic computing: fast approximation with increasing accuracy over time if required. This trend is matched on average with the extra repeats, implying an extra time and energy cost to accumulate the samples, but providing a boost in accuracy. However by utilising our learning rule's ability to enable low sampling deterministic solutions we can outperform the naive stochastic computing reasoning in the low sampling limit (as also seen in figure 3(h)). If fixing K_{train}, this leads to a competition between low repeat performance and ultimate high repeat accuracy, i.e if a model is trained on a high number of repeats, but uses a low number of repeats during testing (inference) time, then the accuracy will be sub-optimal. Similarly, if the model is trained on a low number of repeats, but tested on many, the ultimate accuracy suffers. One possibility is to always tie $K_{\text{test}} = K_{\text{train}}$, but this requires multiple trained weights. It is, therefore, constructive to utilise data such as figure 4(c) as a guide on training and testing the synapse depending on the desired accuracies and operational times. Whilst maintaining the simplicity of a single set of trained weights (fixing K_{train}), a horizontal range of testing values can be chosen to achieve the desired accuracies and energy cost envelope.

Analysing the performance over the space of training vs test repeats for the MNIST task, we find that in most cases, testing with a similar number of repeats to the training performs well. A significant outlier to this was that training with two samples consistently outperformed training with one across all levels of testing, including testing with one sample. We attribute this to the smaller step sizes in the parameter space with two samples compared to one, which allows for a better solution while the variance is still very low and remains small when testing with one sample.

3. Discussion

Neuromorphic devices are a promising route to developing low-energy-cost machine learning systems, seeking to overcome one of the chief drawbacks of traditional neural networks. Stochastic, binary neural networks have shown promise in this regard due to their reduced energy cost and simple implementation [9–13]. Multiple sampling of these networks allows their performance to rival analogue networks [14–19]. Outstanding problems, however, have been providing training rules to achieve high performance even at low sampling rates (where calculations can be performed faster and at less energy cost) and identifying hardware implementations that can natively provide the stochasticity required. We have developed a learning methodology for stochastic binary neural networks that we verify experimentally, using the behaviour of magnetic DWs in nanowires as stochastic synapses. Stochasticity has traditionally been considered a limiting factor in nanomagnetic logic devices [36, 37], but here is a functional aspect that drives learning. We have shown performance of the hardware network comparable to a standard neural network and demonstrated high performance at low sampling thanks to the novel learning rule.

Experimentally, we have observed that a DW injected into a NW with an artificial pinning site can be stochastically pinned and the pinning tuned by using an applied magnetic field. We have then demonstrated that this tunable stochastic pinning can create synapses for a neural network device. Due to the nature of the physical system, these synapses behave as binary stochastic synapses. Our fundamental ingredient for training such a network is a learning rule that considers the variance of the stochastic output of the network. This training method considers taking multiple samples ($K_{\text{train}}/K_{\text{test}}$) of the network output to compute a sample average and deviation. A low number of samples leads toward a predominantly deterministic binary solution and is fast to compute but has lower performance than a high number of samples that approximates a standard 'analogue' network and require more time (and energy). This trade-off allows flexibility in designing the network based on the required performance or operating speed.

Key is that the learning rule developed here has allowed us to find a range of operating regimes because the stochastic part of the output is considered. Other binary stochastic computing approaches, such as Hirtzlin *et al* [19], train using the expectation of the network (which we call mean-field and is equivalent to $K \to \infty$) and lead to a reduced accuracy when fewer samples are used during inference (testing). The Gaussian approximation was also used by Esser *et al* [46] to train a network with binary stochastic synapses on the IBM TrueNorth neurosynaptic system but the contribution from the variance term is considered to be negligible. The contribution from the variance term in our rule allows for weights to be trained that operate better in the low sampling regime compared to the mean-field versions.

Other learning methods where the variance is taken in account stem from the Likelihood-Ratio framework [47–49], which is related to policy gradient methods in reinforcement learning [50]. While these methods consider the stochasticity of the neurons and synapse, they depend heavily on the choice of baseline values for the loss which require complex approximation methods. Additionally, the reparameterisation method applied here allows for a direct feedback of the error signal to the synaptic field parameters and fits within existing backpropagation-based learning methodologies.

Overall, the stochastic learning rule presented in this paper has shown tunability in both high and low sampling regimes and can be implemented simply within backpropagation-style codes. The ability, due to consideration of the variance of the output, to tune between low-sampling deterministic binary and high-sampling stochastic 'analogue like' behaviour lends itself to the flexibility of our system between operational speed/energy cost and test accuracy.

The magnetic DW synapse that we have demonstrated here is a proof of principle component and as such it important to look towards changes that would be necessary for a more 'production ready' neuromorphic device. Optimised devices would likely look towards spin-torque driven DW motion [35] alongside the use of local nanomagnetic elements to encode the weights. Also, readout of the synaptic state could employ magnetoresistive effects to read the state of a synapse, similar to approaches in racetrack memory devices [24]. For example, this could be done using a magnetic tunnel junction situated at the end of the NW to sense whether the DW has propagated that far and by connecting these in series a total resistance measurement could realise the sum over the synapses.

It is also possible to envisage our learning methodology applied to networks built of alternative magnetic elements with similar stochastic properties, such as magnetic tunnel junctions, amongst others [12, 14, 51–53]. Elsewhere, DW devices have been used as neurons [54] or activation functions [55] and magnetic elements in general have been demonstrated in a range of alternative low energy computation schemes [16, 51, 56–58] that exploit the stochasticity of magnetic devices. Our fundamental element, the magnetic stochastic synapse, could fit within such paradigms where efficient production of random bits is key. It is important, however, to state that the key result here is demonstrating performance as run on experimental

hardware, enabled by our stochastic learning rule. Further optimisation is a matter of future research and engineering development.

Whilst the single layer network demonstrated here can only solve linearly separable problems, it can be extended in a number of ways. Retaining the single layer simplicity and looking towards an all magnetic architecture, it has potential applications in the field of reservoir computing. In reservoir computing, a fixed reservoir performs a non-linear spatial-temporal transformation of an input sequence such that the output representation is linearly separable. The advantage of RC is that the reservoir transform can be offloaded to a physical system with appropriate properties and there has been considerable recent interest in developing magnetic (spintronics) based physical reservoir computing [7, 59–66]. There is potential to connect our magnetic DW based neural network to these reservoirs to create a complete hardware reservoir computing system. There is also the more traditional route of scaling our current approach towards multi-layer networks as the learning rule is compatible with back-propagation. An open research question in this avenue is whether the sampling procedure should apply at a local or global scale of the network. One approach is implementation of multi-layers using NW interconnects and logic gates, but if we look away from the limitation of all magnetic architectures, it is also possible to envisage hybrid magnetic-CMOS application specific integrated circuits (as in [67]) that might provide a route to larger scale network hardware. However, details of these implementations are beyond the scope of this current work.

In conclusion, we have developed a training methodology for binary stochastic synapses that considers the network's stochasticity during learning, and resampling of the stochastic output allows for a trade-off between device run time and desired accuracy. This approach has been demonstrated on a proof of concept magnetic DW-based stochastic synapse with excellent agreement between hardware and model during inference.

4. Methods

4.1. Device fabrication

The devices were fabricated using two-stage electron beam lithography with the CSAR-62 resist. Nanowires were deposited in the first stage using thermal evaporation of permalloy ($Ni_{81}Fe_{19}$) to a thickness of 54 nm (base pressure, 7×10^{-7} mbar; process pressure, $\sim 5 \times 10^{-5}$ mbar; rate, 0.5 Å s^{-1}). Current lines and connection pads were deposited in the second stage as Ti/Au (Nominally 10 nm/200 nm via thermal evaporation). Samples were electrically connected to PCB devices using silver DAG.

4.2. Device operation

The device operation procedes as in figure 2. An AVTECH pulse generator was used to apply 30 volt, 100-nanosecond pulses along the current line (resistance 290 Ω). An electromagnet was used to apply fields along the wire lengths. A National Instruments DAQ card was used to control timing between these two, with pulses being triggered at particular times during repeated sinusoidal field sequences. The field at which the pulse is triggered is the propagation field. On the fly calibration of timing enabled correction of any drift between the trigger and field sequence (due to heating) to $\lesssim$0.1 mT.

A focused-MOKE magnetometer (spot size ∼5 micrometer) was used to measure the NW response. Hysteresis loops were obtained with the laser spot positioned over the notch. Single steps in the hysteresis loop indicate the DW passing the notch (an output of 1). Double steps indicate a two-stage pinning/depinning process (an output of 0). An algorithmic method allowed automated evaluation of each hysteresis loop. The number of peaks were calculated in the differentiated Kerr signal; if two peaks were present then the DW had been pinned. To eliminate false positives, the steps in the raw signal corresponding to the peaks were required to be greater than 24% of the total signal change. This was optimised experimentally to allow for peak detection even with a slightly off centre laser spot (unequal step sizes), but to minimises erroneous detections arising from noise.

4.3. Domain passing probability

The probability of an DW not being pinned by the artificial defect site was observed to have a sigmoid-like behaviour. A functional form of this probability was used to simulate magnetic stochastic synapses for computational training of the networks. We fitted this probability using

$$f(h) = d + \frac{1-d}{1+\exp(-\Delta(h-h_0))}, \tag{9}$$

where $d = 0.0219$ is a finite passing probability at low field, $h_0 = 4.63$ mT is the field centre and $\Delta = 2.73$ mT^{-1} is the sigmoid width. We note that this exact form of the fitting function is not necessary for the stochastic learning rule used to train the network.

4.4. Stochastic learning rule

For a network comprised of binary stochastic synapses the value of each neuron can be approximated by a Gaussian given by equation (6), where the mean and variance of each neurons is defined by equations (4) and (5) respectively. Using this approximation, a gradient based learning rule can be derived as the random variable no longer has a dependence on the model parameters. The parameters of this network are the magnetic fields which determine the passing probability of the synapse so a gradient descent update is given by

$$\Delta h_{ij} = -\eta \frac{\partial E(\tilde{y})}{\partial \tilde{y}_i} \frac{\partial \tilde{y}_i}{\partial h_{ij}} \tag{10}$$

$$= -\eta \frac{\partial E(\tilde{y})}{\partial \tilde{y}_i} \left(\frac{\partial \mu_i}{\partial h_{ij}} + \frac{\partial \sigma_i}{\partial h_{ij}} \xi_i \right). \tag{11}$$

The gradient of the mean and variance with respect to the magnetic fields are

$$\frac{\partial \mu_i}{\partial h_{ij}} = f'(h_{ij}) x_j \tag{12}$$

$$\frac{\partial \sigma_i}{\partial h_{ij}} = \frac{(1 - 2f(h_{ij}) x_j)}{2\sigma_i} f'(h_{ij}) x_j, \tag{13}$$

where $f'(h) = \partial f(h)/\partial h$ is the derivative of the passing probability function. Combining this result into equation (10) gives the update rule

$$\Delta h_{ij} = -\eta \frac{\partial E(\tilde{y})}{\partial \tilde{y}_i} f'(h_{ij}) x_j \left(1 + \frac{1 - 2f(h_{ij}) x_j}{2\sigma_i} \xi_i \right). \tag{14}$$

In this form the rule contains the mean field component multiplied by a factor that depends on the variance. While for the derivation of the rule we have specified that ξ_i is a Guassian random variable with zero mean and unit variance, during training it is calculated exactly from the forward phase using $\xi_i = (y_i - \mu_i)/\sigma_i$, so if the neuron output is higher than the mean it will be positive while if it is lower it will be negative. This combines with the $1 - 2f(h_{ij})x_j$ to determine whether the factor increases the weight update or reduces it.

4.5. Model training details

As a benchmark we use the MNIST dataset [68] but to reduce the number of synaptic operations for the experimental hardware it was downsampled by using the MaxPool operation with a filter size of 2×2. This created a set of 14×14 pixel images which were mapped to a binary input by thresholding the pixel intensity at 0.5.

The training part of the dataset was randomly split into a 50 000 training and 10 000 validation subsets. A real valued bias was applied to the output of the simulated binary synapses and these values were converted into a probability using the Softmax function with the loss against the image labels measured using Cross-Entropy loss. Training was performed using mini-batches of 50 images, and iterated until the validation loss did not decrease over 20 epochs. The model with the lowest validation error before the end of training was returned as the trained model. The Adam optimiser was used with a learning rate $\eta = 0.001$ for $K \geqslant 2$ and $\eta = 0.01$ for $K = 1$, determined based on the lower validation error.

4.6. On device machine learning testing

For the demonstration of our stochastic network *in materia* we have used an automated control system to inject an DW into the magnetic NW at the desired magnetic field given by the synaptic weights. We first optimised the synaptic magnetic fields for our network models in simulation for the cases of $K_{\text{train}} = 1$ and 128, using the method detailed below. For each K_{train}, we trained five models before selecting the model that had the lowest error on the validation dataset. We then transferred these to the hardware with the control software loading the pixel binary values (x_j) from the test dataset and using the simulation trained magnetic fields (h_{ij}) to control the magnetic synapses. As detailed in figure 2, if the pixel value was 1 the control system would determine whether the DW has pinned or passed the defect site and return a 0 or 1 respectively. The result of this synaptic operation was then passed back to the program running the neural network inference, which computed the neuron values to predict the correct class of the test data.

Data availability statement

The data that support the findings of this study are available upon reasonable request from the authors.

Acknowledgments

The authors gratefully acknowledge funding from the Engineering and Physical Sciences Research Council (Grant: EP/S009647/1) and the Leverhulme Trust (Research Grant: RPG-2019-097). The authors would like to thank Luca Manneschi and Ian Vidamour for their help and feedback on this work.

Code availability

Code related to this paper is available at https://github.com/mattoaellis/binary_stochastic_synapses.

Appendix. Mean and variance of the Poisson-Binomial distribution

For a network of binary stochastic synapses, the output of each synapses is assumed to be an independent random binary event (Bernoulli trial). If these had the same probability then the sum of these events would result in the Binomial distribution but as each synapse has a different input and synaptic probability the value of the neuron will follow a Poisson-Binomial distribution. Since this distribution can be complex to calculate in full we approximate the distribution by a Gaussian [44]. The mean of the neuron output, y_i, for a given number of samples K is

$$\mu_i = \mathbb{E}[y_i] \tag{A1}$$

$$= \mathbb{E}\left[\frac{1}{K}\sum_k\sum_j w_{ij}^{(k)}x_j\right] \tag{A2}$$

$$= \frac{1}{K}\sum_k\sum_j \mathbb{E}\left[w_{ij}^{(k)}x_j\right] \tag{A3}$$

$$= \frac{1}{K}\sum_k\sum_j \left(0f(h_{ij})x_j + 1f(h_{ij})x_j\right) \tag{A4}$$

$$= \sum_j f(h_{ij})x_j. \tag{A5}$$

where in the final step here we note that the synaptic passing probability $f(h_{ij})$ is independent of k. The variance of the distribution is

$$\sigma_i^2 = \mathrm{var}[y_i] \tag{A6}$$

$$= \mathrm{var}\left[\frac{1}{K}\sum_k\sum_j w_{ij}^{(k)}x_j\right]. \tag{A7}$$

We now use the fact that the variance of a sum of independent random events is the sum of the variances, and that $\mathrm{var}[y/K] = \mathrm{var}[y]/K^2$ such that

$$\sigma_i^2 = \frac{1}{K^2}\sum_k\sum_j \mathrm{var}\left[w_{ij}^{(k)}x_j\right]. \tag{A8}$$

The variance of each synapse as a Bernoulli event is

$$\mathrm{var}\left[w_{ij}^{(k)}x_j\right] = f(h_{ij})x_j\left(1 - f(h_{ij})x_j\right), \tag{A9}$$

and again since this is independent of k then the variance of the neuron output is

$$\sigma_i^2 = \frac{K}{K^2}\sum_j f(h_{ij})x_j\left(1 - f(h_{ij})x_j\right) \tag{A10}$$

$$= \frac{1}{K}\sum_j f(h_{ij})x_j\left(1 - f(h_{ij})x_j\right). \tag{A11}$$

ORCID iDs

Matthew O A Ellis https://orcid.org/0000-0003-0338-8920
Alexander Welbourne https://orcid.org/0000-0002-9089-9712
Dan A Allwood https://orcid.org/0000-0002-3582-4355
Thomas J Hayward https://orcid.org/0000-0002-3732-3095
Eleni Vasilaki https://orcid.org/0000-0003-3705-7070

References

[1] Thompson N C, Greenewald K, Lee K and Manso G F 2021 Deep learning's diminishing returns: the cost of improvement is becoming unsustainable *IEEE Spectr.* **58** 50–55
[2] Sabry Aly M M *et al* 2019 The N3XT approach to energy-efficient abundant-data computing *Proc. IEEE* **107** 19–48
[3] Strubell E, Ganesh A and McCallum A 2019 Energy and policy considerations for deep learning in NLP (arXiv:1906.02243)
[4] Ziegler M 2020 Novel hardware and concepts for unconventional computing *Sci. Rep.* **10** 1–3
[5] Niemier M T *et al* 2011 Nanomagnet logic: progress toward system-level integration *J. Phys.: Condens. Matter* **23** 493202
[6] Finocchio G, Di Ventra M, Camsari K Y, Everschor-Sitte K, Amiri P K and Zeng Z 2021 The promise of spintronics for unconventional computing *J. Magn. Magn. Mater.* **521** 167506
[7] Grollier J, Querlioz D, Camsari K Y, Everschor-Sitte K, Fukami S and Stiles M D 2020 Neuromorphic spintronics *Nat. Electron.* **3** 360–70
[8] Petersen C C, Malenka R C, Nicoll R A and Hopfield J J 1998 All-or-none potentiation at CA3-CA1 synapses *Proc. Natl Acad. Sci.* **95** 4732–7
[9] Simons T and Lee D-J 2019 A review of binarized neural networks *Electronics* **8** 661
[10] Laborieux A, Bocquet M, Hirtzlin T, Klein J-O, Diez L H, Nowak E, Vianello E, Portal J-M and Querlioz D 2020 Low power in-memory implementation of ternary neural networks with resistive RAM-based synapse *2020 2nd IEEE Int. Conf. on Artificial Intelligence Circuits and Systems (AICAS)* pp 136–40
[11] 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
[12] Penkovsky B, Bocquet M, Hirtzlin T, Klein J-O, Nowak E, Vianello E, Portal J-M and Querlioz D 2020 In-memory resistive RAM implementation of binarized neural networks for medical applications *2020 Design, Automation & Test in Europe Conf. & Exhibition* pp 690–5
[13] Neftci E O, Pedroni B U, Joshi S, Al-Shedivat M and Cauwenberghs G 2016 Stochastic synapses enable efficient brain-inspired learning machines *Front. Neurosci.* **10** 241
[14] 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
[15] Li Z *et al* 2019 HEIF: highly efficient stochastic computing-based inference framework for deep neural networks *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **38** 1543–56
[16] Shao Y, Sinaga S L, Sunmola I O, Borland A S, Carey M J, Katine J A, Lopez-Dominguez V and Amiri P K 2021 Implementation of artificial neural networks using magnetoresistive random-access memory-based stochastic computing units *IEEE Magn. Lett.* **12** 1–5
[17] Muthappa P K, Neugebauer F, Polian I and Hayes J P 2020 Hardware-based fast real-time image classification with stochastic computing *2020 IEEE 38th Int. Conf. on Computer Design (ICCD)* pp 340–7
[18] Nisar A, Khanday F A and Kaushik B K 2020 Implementation of an efficient magnetic tunnel junction-based stochastic neural network with application to iris data classification *Nanotechnology* **31** 504001
[19] Hirtzlin T, Penkovsky B, Bocquet M, Klein J-O, Portal J-M and Querlioz D 2019 Stochastic computing for hardware implementation of binarized neural networks *IEEE Access* **7** 76394–403
[20] Baibich M N, Broto J M, Fert A, Van Dau F N, Petroff F, Etienne P, Creuzet G, Friederich A and Chazelas J 1988 Giant magnetoresistance of (001)Fe/(001)Cr magnetic superlattices *Phys. Rev. Lett.* **61** 2472–5
[21] Binasch G, Grünberg P, Saurenbach F and Zinn W 1989 Enhanced magnetoresistance in layered magnetic structures with antiferromagnetic interlayer exchange *Phys. Rev.* B **39** 4828–30
[22] Allwood D A, Xiong G, Faulkner C, Atkinson D, Petit D and Cowburn R 2005 Magnetic domain-wall logic *Science* **309** 1688–92
[23] Mccray W P 2009 How spintronics went from the lab to the iPod *Nat. Nanotechnol.* **4** 2–4
[24] Parkin S S P, Hayashi M and Thomas L 2008 Magnetic domain-wall racetrack memory *Science* **320** 190–4
[25] Lavrijsen R, Lee J-H, Fernández-Pacheco A, Petit D C M C, Mansell R and Cowburn R P 2013 Magnetic ratchet for three-dimensional spintronic memory and logic *Nature* **493** 647–50
[26] Fernández-Pacheco A *et al* 2016 Magnetic state of multilayered synthetic antiferromagnets during soliton nucleation and propagation for vertical data transfer *Adv. Mater. Interfaces* **3** 1600097
[27] Emori S, Bauer U, Ahn S-M, Martinez E and Beach G S 2013 Current-driven dynamics of chiral ferromagnetic domain walls *Nat. Mater.* **12** 611–6
[28] Hu J-M, Li Z, Chen L-Q and Nan C-W 2011 High-density magnetoresistive random access memory operating at ultralow voltage at room temperature *Nat. Commun.* **2** 1–8
[29] Kurenkov A, Dutta Gupta S, Zhang C, Fukami S, Horio Y and Ohno H 2019 Artificial neuron and synapse realized in an antiferromagnet/ferromagnet heterostructure using dynamics of spin–orbit torque switching *Adv. Mater.* **31** 1900636
[30] Lan X, Cao Y, Liu X, Xu K, Liu C, Zheng H and Wang K 2021 Gradient descent on multilevel spin–orbit synapses with tunable variations *Adv. Intell. Syst.* **3** 2000182
[31] Liu J, Xu T, Feng H, Zhao L, Tang J, Fang L and Jiang W 2022 Compensated ferrimagnet based artificial synapse and neuron for ultrafast neuromorphic computing *Adv. Funct. Mater.* **32** 2107870
[32] Li W, Lan X, Liu X, Zhang E, Deng Y and Wang K 2022 Switching plasticity in compensated ferrimagnetic multilayers for neuromorphic computing *Chin. Phys.* B **31** 117106
[33] Cao Y, Rushforth A, Sheng Y, Zheng H and Wang K 2019 Tuning a binary ferromagnet into a multistate synapse with spin–orbit-torque-induced plasticity *Adv. Funct. Mater.* **29** 1808104

[34] Sanz-Hernández D, Hamans R, Liao J-W, Welbourne A, Lavrijsen R and Fernández-Pacheco A 2017 Fabrication, detection and operation of a three-dimensional nanomagnetic conduit *ACS Nano* **11** 11066–73
[35] Luo Z, Hrabec A, Dao T P, Sala G, Finizio S, Feng J, Mayr S, Raabe J, Gambardella P and Heyderman L J 2020 Current-driven magnetic domain-wall logic *Nature* **579** 214–8
[36] Hayward T 2015 Intrinsic nature of stochastic domain wall pinning phenomena in magnetic nanowire devices *Sci. Rep.* **5** 1–12
[37] Kumar D, Jin T, Al Risi S, Sbiaa R, Lew W S and Piramanayagam S N 2019 Domain wall motion control for racetrack memory applications *IEEE Trans. Magn.* **55** 1–8
[38] Lambson B, Carlton D and Bokor J 2011 Exploring the thermodynamic limits of computation in integrated systems: magnetic memory, nanomagnetic logic and the Landauer limit *Phys. Rev. Lett.* **107** 010604
[39] Hayashi M, Thomas L, Moriya R, Rettner C and Parkin S S P 2008 Current-controlled magnetic domain-wall nanowire shift register *Science* **320** 209–11
[40] Yang S-H, Ryu K-S and Parkin S 2015 Domain-wall velocities of up to 750 m s^{-1} driven by exchange-coupling torque in synthetic antiferromagnets *Nat. Nanotechnol.* **10** 221–6
[41] Nakatani Y, Thiaville A and Miltat J 2005 Head-to-head domain walls in soft nano-strips: a refined phase diagram *J. Magn. Magn. Mater.* **290–291** 750–3
[42] Satel J, Trappenberg T and Fine A 2009 Are binary synapses superior to graded weight representations in stochastic attractor networks? *Cogn. Neurodyn.* **3** 243–50
[43] Dubreuil A M, Amit Y and Brunel N 2014 Memory capacity of networks with stochastic binary synapses *PLoS Comput. Biol.* **10** e1003727
[44] Hong Y 2013 On computing the distribution function for the Poisson binomial distribution *Comput. Stat. Data Anal.* **59** 41–51
[45] Vouros A, Gehring T V, Szydlowska K, Janusz A, Tu Z, Croucher M, Lukasiuk K, Konopka W, Sandi C and Vasilaki E 2018 A generalised framework for detailed classification of swimming paths inside the Morris Water Maze *Sci. Rep.* **8** 1–15
[46] Esser S K, Appuswamy R, Merolla P, Arthur J V and Modha D S 2015 Backpropagation for energy-efficient neuromorphic computing *Advances in Neural Information Processing Systems* vol 28
[47] Williams R J 1992 Simple statistical gradient-following algorithms for connectionist reinforcement learning *Mach. Learn.* **8** 229–56
[48] Gu S, Levine S, Sutskever I and Mnih A 2015 MuProp: unbiased backpropagation for stochastic neural networks (arXiv:1511.05176)
[49] Parmas P and Sugiyama M 2021 A unified view of likelihood ratio and reparameterization gradients *Int. Conf. on Artificial Intelligence and Statistics* (PMLR) pp 4078–86
[50] Vasilaki E, Frémaux N, Urbanczik R, Senn W and Gerstner W 2009 Spike-based reinforcement learning in continuous state and action space: when policy gradient methods fail *PLoS Comput. Biol.* **5** e1000586
[51] Azam M A, Bhattacharya D, Querlioz D, Ross C A and Atulasimha J 2020 Voltage control of domain walls in magnetic nanowires for energy-efficient neuromorphic devices *Nanotechnology* **31** 145201
[52] Sanz-Hernández D *et al* 2021 Tunable stochasticity in an artificial spin network *Adv. Mater.* **33** 2008135
[53] Misba W A, Lozano M, Querlioz D and Atulasimha J 2022 Energy efficient learning with low resolution stochastic domain wall synapse for deep neural networks *IEEE Access* **10** 84946–59
[54] Hassan N, Hu X, Jiang-Wei L, Brigner W H, Akinola O G, Garcia-Sanchez F, Pasquale M, Bennett C H, Incorvia J A C and Friedman J S 2018 Magnetic domain wall neuron with lateral inhibition *J. Appl. Phys.* **124** 152127
[55] Brigner W H, Hassan N, Hu X, Bennett C H, Garcia-Sanchez F, Cui C, Velasquez A, Marinella M J, Incorvia J A C and Friedman J S 2022 Domain wall leaky integrate-and-fire neurons with shape-based configurable activation functions *IEEE Trans. Electron Devices* **69** 2353–9
[56] Borders W A, Pervaiz A Z, Fukami S, Camsari K Y, Ohno H and Datta S 2019 Integer factorization using stochastic magnetic tunnel junctions *Nature* **573** 390–3
[57] Al Misba W, Lozano M, Querlioz D and Atulasimha J 2022 Energy efficient learning with low resolution stochastic domain wall synapse for deep neural networks *IEEE Access* **10** 84946–59
[58] 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** 2546–55
[59] Dawidek R W *et al* 2021 Dynamically-driven emergence in a nanomagnetic system *Adv. Funct. Mater.* **31** 2008389
[60] Ababei R V, Ellis M O A, Vidamour I T, Devadasan D S, Allwood D A, Vasilaki E and Hayward T J 2021 Neuromorphic computation with a single magnetic domain wall *Sci. Rep.* **11** 15587
[61] Welbourne A, Levy A L R, Ellis M O A, Chen H, Thompson M J, Vasilaki E, Allwood D A and Hayward T J 2021 Voltage-controlled superparamagnetic ensembles for low-power reservoir computing *Appl. Phys. Lett.* **118** 202402
[62] Gartside J C, Stenning K D, Vanstone A, Holder H H, Arroo D M, Dion T, Caravelli F, Kurebayashi H and Branford W R 2022 Reconfigurable training and reservoir computing in an artificial spin-vortex ice via spin-wave fingerprinting *Nat. Nanotechnol.* **17** 460–9
[63] Vidamour I T *et al* 2022 Quantifying the computational capability of a nanomagnetic reservoir computing platform with emergent magnetisation dynamics *Nanotechnology* **33** 485203
[64] Vidamour I, Swindells C, Venkat G, Fry P, Welbourne A, Rowan-Robinson R, Backes D, Maccherozzi F, Dhesi S, Vasilaki E, Allwood D and Hayward T 2022 Reservoir computing with emergent dynamics in a magnetic metamaterial (arXiv:2206.04446 [cond-mat])
[65] Allwood D A, Ellis M O A, Griffin D, Hayward T J, Manneschi L, Musameh M F K, O'Keefe S, Stepney S, Swindells C, Trefzer M A, Vasilaki E, Venkat G, Vidamour I and Wringe C 2022 A perspective on physical reservoir computing with nanomagnetic devices (arXiv:2212.04851 [physics])
[66] Stenning K D, Gartside J C, Manneschi L, Cheung C T S, Chen T, Vanstone A, Love J, Holder H H, Caravelli F, Everschor-Sitte K, Vasilaki E and Branford W R 2022 Adaptive programmable networks for in materia neuromorphic computing (arXiv:2211.06373 [cond-mat])
[67] Hirtzlin T, Bocquet M, Penkovsky B, Klein J-O, Nowak E, Vianello E, Portal J-M and Querlioz D 2020 Digital biologically plausible implementation of binarized neural networks with differential hafnium oxide resistive memory arrays *Front. Neurosci.* **13** 1383
[68] LeCun Y 1998 The MNIST database of handwritten digits (available at: http://yann.lecun.com/exdb/mnist/)