Composable Probabilistic Inference Networks Using MRAM-based Stochastic Neurons

RAMTIN ZAND, University of Central Florida, USA
KEREM Y. CAMSARI, Purdue University, USA
SUPRIYO DATTA, Purdue University, USA
RONALD F. DEMARA, University of Central Florida, USA

Magnetoresistive random access memory (MRAM) technologies with thermally unstable nanomagnets are leveraged to develop an intrinsic stochastic neuron as a building block for restricted Boltzmann machines (RBMs) to form deep belief networks (DBNs). The embedded MRAM-based neuron is modeled using precise physics equations. The simulation results exhibit the desired sigmoidal relation between the input voltages and probability of the output state. A probabilistic inference network simulator (PIN-Sim) is developed to realize a circuit-level model of an RBM utilizing resistive crossbar arrays along with differential amplifiers to implement the positive and negative weight values. The PIN-Sim is composed of five main blocks to train a DBN, evaluate its accuracy, and measure its power consumption. The MNIST dataset is leveraged to investigate the energy and accuracy tradeoffs of seven distinct network topologies in SPICE using the 14nm HP-FinFET technology library with the nominal voltage of 0.8V, in which an MRAM-based neuron is used as the activation function. The software and hardware level simulations indicate that a $784 \times 200 \times 10$ topology can achieve less than 5% error rates with $\sim 400pJ$ energy consumption. The error rates can be reduced to 2.5% by using a $784 \times 500 \times 500 \times 500 \times 10$ DBN at the cost of $\sim 10\times$ higher energy consumption and significant area overhead. Finally, the effects of specific hardware-level parameters on power dissipation and accuracy tradeoffs are identified via the developed PIN-Sim framework.

CCS Concepts: • **Computing methodologies → Machine learning**; **Unsupervised learning**; • **Hardware → Very large scale integration design**; **Spintronics and magnetic technologies**;

Additional Key Words and Phrases: Deep belief network (DBN), restricted Boltzmann machine (RBM), magnetoresistive random access memory (MRAM), stochastic binary neuron, resistive crossbar array

ACM Reference Format:
Ramtin Zand, Kerem Y. Camsari, Supriyo Datta, and Ronald F. DeMara. 2010. Composable Probabilistic Inference Networks Using MRAM-based Stochastic Neurons. *ACM Comput. Entertain.* 9, 4, Article 39 (March 2010), 21 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

In recent years, innovation within the disciplines of machine intelligence and learning (ML) utilizing artificial neural networks (ANN) that aim to model biological brain behavior has grown significantly due to the existence of vast datasets available to train such networks. Some interesting projects within these fields include solving complicated classification

Authors' addresses: Ramtin Zand, University of Central Florida, Orlando, FL, 32816, USA, ramtinmz@knights.ucf.edu; Kerem Y. Camsari, Purdue University, West Lafayette, IN, 47906, USA, kcamsari@purdue.edu; Supriyo Datta, Purdue University, West Lafayette, IN, 47906, USA, datta@purdue.edu; Ronald F. DeMara, University of Central Florida, Orlando, FL, 32816, USA, ronald.demara@ucf.edu.

Manuscript submitted to ACM

problems by utilizing ANN's strength in information processing [Basheer and Hajmeer 2000], pattern recognition tasks [Bishop et al. 1995], and even out-maneuvering a world champion Go player to a historic defeat [Churchland and Sejnowski 2016].

The techniques most commonly used to train ANNs today typically require supervised learning, where the error rate is measured by comparing the output from the network with a known desired output. Then, using a subsequent training technique such as backpropagation, the corresponding weights within the network are adjusted [Hecht-Nielsen 1992]. However, unsupervised learning is attracting considerable attentions in recent years due to its compatibility with the nature of intelligent biological systems, which learn through observation, not by supervision [LeCun et al. 2015]. In unsupervised learning approaches, decision processes based on probabilistic inference are built upon constructing statistical correlation of the inputs into categories [Buesing et al. 2011]. Deep belief networks (DBNs) are an interesting class of ML techniques utilizing an unsupervised learning approach known as contrastive divergence (CD) [Carreira-Perpinan and Hinton 2005], which demonstrates outstanding learning abilities for various applications such as natural language understanding [Sarikaya et al. 2014]. DBNs are constructed by multiple Restricted Boltzmann machines (RBMs), which can be hierarchically connected to form a network [Hinton et al. 2006].

Research focused on software implementation of DBNs show that conventional von-Neumann architectures are poorly-matched to the processing flow in terms of the constituent operations at a fine granularity. Although software implementations on conventional architectures provide flexibility, they require significant execution time and energy caused by the memory-processor bandwidth bottleneck, which is intensified due to the large matrix multiplications required [Merolla et al. 2014]. Therefore, hardware-based RBM design research seeks to surmount these limitations. Previous work on RBM hardware implementations use conventional VLSI design techniques [Yuan and Parhi 2017], FPGA approaches [Kim et al. 2010; Ly and Chow 2010], and stochastic CMOS methods[Ardakani et al. 2017]. Moreover, emerging technologies such as resistive RAM (RRAM) [Bojnordi and Ipek 2016; Sheri et al. 2015] and phase change memory (PCM) [Eryilmaz et al. 2016] had been utilized as weighted connections within the DBN architecture to interconnect its various building blocks. The previous hybrid Memristor/CMOS designs attempt to realize an intrinsic implementation of the weighted connections. Recently, a current-driven low energy-barrier spintronic device has been proposed to be utilized in RBMs as the activation function [Zand et al. 2018], while similar devices have been previously proposed for spiking [Sengupta et al. 2016b,c] and hard axis clocked [Behin-Aein et al. 2016] neural systems. However, the current-mode operation of these devices imposes a significant power consumption to the activation functions, while requiring weighted connections with $M\Omega$ resistances. The design proposed herein takes a new approach from the device-level upward to overcome the challenges mentioned above by utilizing a voltage-driven spintronic device with embedded magnetoresistive random access memory (MRAM) constructed by low energy barrier nanomagnets, which leverages intrinsic thermal noise to provide a natural and power-efficient building block for RBMs. Moreover, we propose a simulation framework for probabilistic learning networks, called PIN-Sim, which is utilized herein to realize a feasible circuit-level implementation of DBN architectures using a SPICE model of our proposed embedded MRAM-based neuron. Specifically, the main contributions of this paper are as follows:

1. A transportable Probabilistic Inference Network Simulator (PIN-Sim) to realize a circuit-level implementation of DBN utilizing voltage-controlled embedded MRAM-based neurons as the probabilistic sigmoidal activation functions. The PIN-Sim framework can be utilized for design space exploration to achieve an optimized network implementation based on the application requirements.

2. Detailed results and analyses about the effects of various circuit-level and device-level tunable parameters on the accuracy and power consumption of the DBNs implemented by PIN-Sim framework.

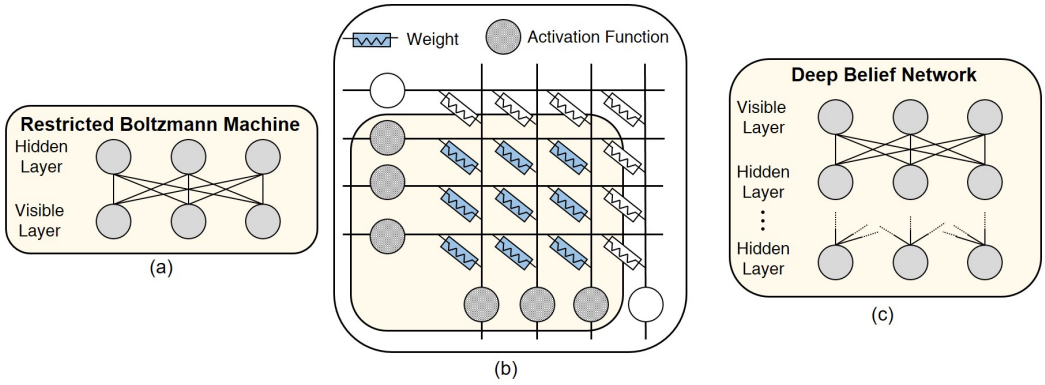

Fig. 1. (a) An RBM structure depicting neurons organized into hidden and visible layers, (b) a 3×3 RBM implemented within a 4×4 crossbar architecture using a weighted array to generate resistances needed to appropriately control the activation function, (c) a DBN structure constructed from multiple hidden layers which act to increase recognition accuracy.

3. Discussions regarding the effects of noise, and variations in the resistance of the weighted connections on the accuracy of our proposed probabilistic spin logic-based DBN circuits.

The remainder of the paper is organized as follows. Section 2 describes the fundamentals of the RBMs and the CD unsupervised learning algorithm. The structure and modeling methodology of the proposed neuron with embedded MRAM is elaborated in Section 3. Section 4 provides details about the circuit-level implementation of DBNs using our proposed PIN-Sim framework. The software and hardware level simulation results are provided in Section 5, as well as a comprehensive comparison between our proposed DBN realization and previous hardware implementations. Finally, Section 6 concludes the paper by relating its contributions, as well as the improvements achieved by the proposed MRAM-based neuron and PIN-Sim framework.

2 RESTRICTED BOLTZMANN MACHINES

Restricted Boltzmann machines (RBMs) are a class of recurrent stochastic neural networks, in which each state of the network, k, has an energy determined by the connection weights between nodes and the node bias as described by Equation 1, where s_i^k is the state of node i in k, b_i is the bias, or intrinsic excitability of node i, and w_{ij} is the connection weight between nodes i and j [Ackley et al. 1985].

$$E(k) = -\sum_i s_i^k b_i - \sum_{i<j} s_i^k s_j^k w_{ij} \tag{1}$$

Each node in an RBM has a probability to be in state one according to Equation 2, where σ is the sigmoid function. RBMs, when given sufficient time, reach a Boltzmann distribution where the probability of the system being in state $\boldsymbol{s}$ is found by Equation 3, where $\boldsymbol{u}$ could be any possible state of the system. Thus, the system is most likely to be found in states that have the lowest associated energy.

$$P(s_i = 1) = \sigma(b_i + \sum_j w_{ij} s_j) \tag{2}$$

$$P(s) = \frac{e^{-E(s)}}{\sum_u e^{-E(u)}} \tag{3}$$

Algorithm 1: Contrastive Divergence Unsupervised Learning Algorithm

Input: train dataset (D_{train}), # of training samples (S), # of RBMs (M)
Output: $weight(n).mat$, $bias(n).mat$, where n is the RBM number
Require: Maximum iteration ($MaxIter$), Learning Rate (η)
for *i= 1 : S* **do**
  $\mathbf{v} = D_{train}(i)$;
  for *j=1 : M* **do**
    for *k=1 : MaxIter* **do**
      Feed-Forward 1: $\mathbf{h} = \sigma(b + \sum w.\mathbf{v})$;
      Feed-Back: $\mathbf{v'} = \sigma(c + \sum w.\mathbf{h})$;
      Feed-Forward 2: $\mathbf{h'} = \sigma(b + \sum w.\mathbf{v'})$;
      Update:
      $\Delta\mathbf{W}(j) = \eta(vh^T - v'h'^T) \Rightarrow \mathbf{W}(j) = \mathbf{W}(j) + \Delta\mathbf{W}(j)$
      $\Delta\mathbf{B}(j) = \eta(h - h') \Rightarrow \mathbf{B}(j) = \mathbf{B}(j) + \Delta\mathbf{B}(j)$
      $\Delta\mathbf{C}(j) = \eta(v - v') \Rightarrow \mathbf{C}(j) = \mathbf{C}(j) + \Delta\mathbf{C}(j)$
    end
  end
end
for *j=1 : M* **do**
  $weight(j).mat \Leftarrow W(j)$;
  $bias(j).mat \Leftarrow B(j)$;
end

RBMs are constrained to two fully-connected non-recurrent layers called the *visible layer* and the *hidden layer*. As shown in Figure 1, RBMs can be readily implemented by a crossbar architecture. The most well-known approach for training RBMs is contrastive divergence (CD), which is an approximate gradient descent procedure using Gibbs sampling [Carreira-Perpinan and Hinton 2005]. CD operates in four steps as described below:

1. *Feed-Forward 1:* the training input vector, v, is applied to the visible layer, and the hidden layer, h, is sampled.
2. *Feed-back:* The sampled hidden layer output is fed-back and the generated input (v') is sampled.
3. *Feed-Forward 2:* v' is applied to the visible layer and the reconstructed hidden layer is sampled to obtain h'.
4. *Update:* The weights are updated according to Equation 4, where η is the learning rate and W is the weight matrix.

$$\Delta W = \eta(vh^T - v'h'^T) \tag{4}$$

RBMs can be readily stacked to realize a DBN, which can be trained similarly to RBMs. The training process is conducted by executing CD starting first with the visible layer and the first of the hidden layers within the network. The CD is repeated as many times as required, which will adjust the weights in a hierarchical flow as described in Algorithm 1.

3 EMBEDDED MRAM BASED NEURON AS A BUILDING BLOCK FOR RBMS

The basic building block of Boltzmann Machines is a stochastic binary neuron that produces a binary output with a given probability. This probability is modulated by the weighted input the neuron receives from the other neurons [Hinton et al. 1984], as shown Figure 2 (a). Here, we show that a recently proposed building block that leverages the highly scaled embedded magnetoresistive random access memory (MRAM) technology, which is conventionally used as

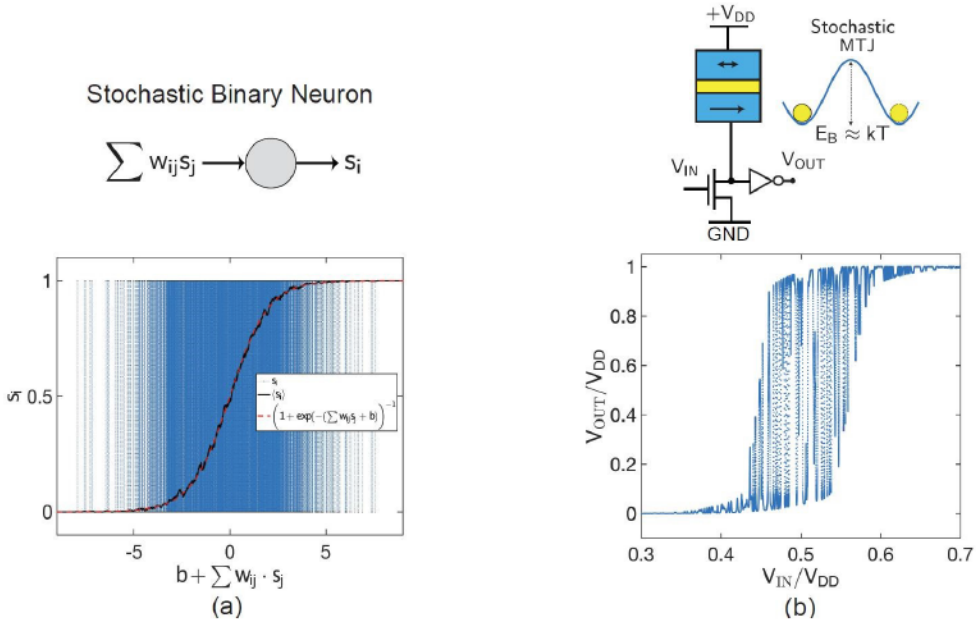

Fig. 2. a) The building block of the proposed spin-based RBMs, the stochastic binary neuron and its ideal input output characteristics are shown. The dashed red curve indicates the mean of the output that is given by the sigmoid function, $\sigma(z) = 1/(1 + \exp(-z))$, where z is the input. The dashed blue curve is the instantaneous output while the input is being swept. The running average of the output, as indicated by the black curve, shows a mean that is equal to the sigmoid function. b) A hardware representation of the stochastic binary neuron in terms of an Embedded Magnetic Tunnel Junction architecture is shown. The free layer of a conventional Embedded MTJ has an energy barrier E_B of 40-60 kT and thus is non-volatile. Reducing the energy barrier of the free layer results in a resistive behavior that is fluctuating between a low (R_P parallel orientation) and a high (R_{AP} anti-parallel) resistance. The gate voltage of the transistor (V_{IN}) controls the resistance of the transistor to regulate the output voltage to approximate the behavior of a stochastic binary neuron in hardware.

a memory device, can enable an approximate hardware representation of the binary stochastic neuron in RBM structure as shown in Figure 2 (b).

The functional component of an MRAM architecture is a magnetic tunnel junction (MTJ) that is a multilayer 2-terminal device that exhibits a resistance change depending on the orientation of its magnetic layers. One of these magnetic layers is designed to have a fixed magnetic orientation (fixed layer) while the magnetization of the other layer can be switched by a magnetic field or by a spin-polarized current (free layer). In the latter, a current that flows through the fixed layer can exert a "spin-transfer-torque" to switch the magnetization of the free layer allowing an electrical writing mechanism [Bhatti et al. 2017]. In conventional memory devices, the free layer is designed to have a large energy barrier with respect to the thermal energy (kT) so that the fixed layer can function as a non-volatile memory. In recent years the use of superparamagnetic MTJs that are not thermally stable have been experimentally and theoretically investigated in search of functional spintronic devices [Camsari et al. 2017a; Choi et al. 2014; Debashis et al. 2016; Fukushima et al. 2014; Liyanagedera et al. 2017; Locatelli et al. 2014; Mizrahi et al. 2018; Sutton et al. 2017; Zand et al. 2018].

In this paper, we use a recently proposed design that makes minimal modifications to the 1 Transistor / 1 MTJ architecture of the commercially available embedded MRAM technology [Camsari et al. 2017b]. The first modification

is to replace the stable free layer with a low-barrier nanomagnet ($E_B \ll 40kT$) that can be achieved by either reducing the total number of spins in the nanomagnet (by reducing M_sVol., where M_s is the saturation magnetization and Vol. is the volume [Bapna and Majetich 2017]) or by using circular disk magnets that have no preferential easy-axis [Debashis et al. 2016]. The resistance of an MTJ with such a low-barrier nanomagnet randomly fluctuates between high (R_{AP}) and low resistance states (R_P), creating a fluctuating output voltage at the drain of the NMOS transistor (Figure 2b). If the transistor resistance that is controlled by the input voltage (V_{IN}) is matched to that of the average MTJ resistance at $V_{IN} = V_{DD}/2$, large voltage fluctuations are obtained at the drain output. For typical R_{AP}/R_P ratios, a CMOS inverter can amplify these fluctuations to produce a rail-to-rail stochastic output at this input value. Changing the input voltage modulates the transistor resistance, and can suppress these fluctuating outputs either by making the transistor resistance too small and shorting the output to ground, or by making the transistor resistance too high and making the output node V_{DD}. The basic device operation can be understood by considering the MTJ conductance [Camsari et al. 2017b]:

$$G_{MTJ} = G_0 \left[1 + m_z \frac{TMR}{(2 + TMR)} \right] \tag{5}$$

where m_z is the instantaneous free layer magnetization that is fluctuating stochastically in the presence of thermal noise, G_0 is the average MTJ conductance, $(G_P + G_{AP})/2$, and TMR is the tunneling magnetoresistance ratio, that is defined as $TMR = (G_P - G_{AP})/G_{AP}$. The voltage division between the transistor and the MTJ (Figure 2b) produces a drain voltage that can be expressed as:

$$V_{DRAIN}/V_{DD} = \frac{(2 + TMR) + TMR\, m_z}{(2 + TMR)(1 + \alpha) + TMR\, m_z} \tag{6}$$

where we introduce a parameter, α, that is defined as the ratio of the transistor conductance (G_T) to the average MTJ conductance (G_0), i. e, $\alpha = G_T/G_0$. As the input voltage V_{IN} changes the transistor conductance G_T, the drain output behaves as a noisy inverter. It can be seen from Equation 6 that the noise amplitude at the drain is maximum when $\alpha \approx 1$, therefore the MTJ resistance is matched to the NMOS resistance ($\alpha = 1$) when $V_{IN}/V_{DD} = 0.5$ to obtain an output with large fluctuations at the symmetry point. Even though the drain voltage shows fluctuations of the order of hundreds of mV for typical TMR values, an additional inverter is used to amplify the noise to produce rail-to-rail voltages for a range of input voltages.

The full circuit behavior of the embedded MRAM based neuron is modeled by a solving the magnetization dynamics of the low barrier nanomagnet using the stochastic Landau-Lifshitz-Gilbert (LLG) equation self-consistently with the transport equations in a SPICE framework [Camsari et al. 2015]. The NMOS transistor is modeled by the predictive technology models (PTM) and for simplicity a bias-independent MTJ model is used that is modeled according to Equation 5. The magnetization input for the MTJ conductance is instantaneously provided from the stochastic LLG equation. The stochastic LLG reads:

$$(1 + \alpha^2)d\hat{m}/dt = -|\gamma|\hat{m} \times \vec{H} - \alpha|\gamma|(\hat{m} \times \hat{m} \times \vec{H}) + 1/qN(\hat{m} \times \vec{I}_S \times \hat{m}) + \left(\alpha/qN(\hat{m} \times \vec{I}_S)\right) \tag{7}$$

where α is the damping coefficient of the nanomagnet, γ is the electron gyromagnetic ratio, q is the electron charge, and $\vec{I}_S$ is the spin current incident to the free layer. The spin current is polarized along the direction of the fixed layer polarization ($\hat{z}$) and its amplitude is proportional to the charge current I_c flowing through the MTJ, such that $\vec{I}_S = PI_c\hat{z}$. N is the total number of spins in the free layer (CoFeB), $N = M_s\text{Vol.}/\mu_B$, where M_s is the saturation magnetization of CoFeB and μ_B is the Bohr magneton. For the free layer, we use a monodomain circular disk magnet whose effective field $\vec{H}$ is given as $-4\pi M_s m_x \hat{x} + \vec{H}_n$, $\hat{x}$ being the out-of-plane direction of the magnet. $\vec{H}_n$ is the isotropic thermal noise

Table 1. Parameters Used for Modeling and Simulation [Camsari et al. 2017b]

Parameters	Value
Saturation magnetization (CoFeB) (M_s)	$1100 emu/cc$ [Sankey et al. 2008]
Free Layer diameter, thickness	$22nm$, $2nm$
Polarization	0.59 [Lin et al. 2009]
TMR	110% [Lin et al. 2009]
MTJ RA-product	$9\Omega - \mu m^2$ [Lin et al. 2009]
Damping coefficient	0.01 [Sankey et al. 2008]
Temperature	$26.85°C$

field, uncorrelated in three directions: $\left(H_n^{x,y,z}\right)^2 = 2\alpha kT/(|\gamma|M_s \text{Vol.})$. The transistors are based on 14nm HP-FinFET PTM [pre [n. d.]].

In this paper, we use a circular disk magnet with $\ll kT$ energy barrier in the absence of any shape anisotropy. Such magnets have been fabricated and characterized in [Cowburn et al. 1999; Debashis et al. 2018]. Moreover, elliptical magnets showing GHz telegraphic oscillations have also been experimentally observed in [Pufall et al. 2004]. The demonstrated parameters listed in Table 2 [Camsari et al. 2017b] are used to generate all of the results that are provided within this paper. We also note for the chosen parameters with a circular free layer with an in-plane anisotropy that the results are not significantly influenced by the current that is flowing at the midpoint ($V_{IN} = V_{DD}/2$), and note that any pinning at higher input voltages benefits the switching operation of the device.

3.1 RBM Hardware Implementation

Figure 3 exhibits a feasible hardware implementation of an $n \times m$ RBM, in which neurons based on the concise embedded MRAM-based design described in the previous section are used to generate the required probabilistic sigmoidal activation function. The resistive crossbar arrays are utilized to realize the matrix multiplication elaborated in Equation 2. In this work, the weights are trained off-chip and the resistive weighted connections will be programmed accordingly. Any resistive devices such as memristors [Strukov et al. 2008] or spin-orbit torque (SOT)-driven domain wall motion (DWM) devices [Sengupta et al. 2016a] can be utilized for weighted connections without the loss of generality.

4 PROPOSED DBN STRUCTURE

To implement the positive and negative weights in the $\boldsymbol{w}$ matrix, two resistive weighted arrays with the same dimensions are required [Hu et al. 2012], as shown in Figure 3. The outputs of the positive and negative weighted connections are linked to differential amplifiers which are implemented by op-amps as shown in Figure 3. The output voltage of the op-amp, i.e. $V_{out} = \frac{R_1}{R_0}(V_{in}^+ - V_{in}^-)$, is applied to the MRAM-based neuron as an input signal. The neuron with embedded MRAM will generate an output voltage signal, which fluctuates between VDD and GND with a probability that is modulated based on the applied input voltage. Finally, a resistor-capacitor (RC) integrator circuit is utilized to convert the probabilistic output of the neuron to an analog voltage level, which can be later converted to a digital output through digital to analog conversion. In order to verify the functionality and assess the performance of our proposed RBM implementation, we have simulated a 2×2 RBM via SPICE circuit simulation using the 14nm HP-FinFET technology library with an MRAM-based neuron used as the activation function. The results obtained validate the functionality of our proposed design as elaborated in Figure 4.

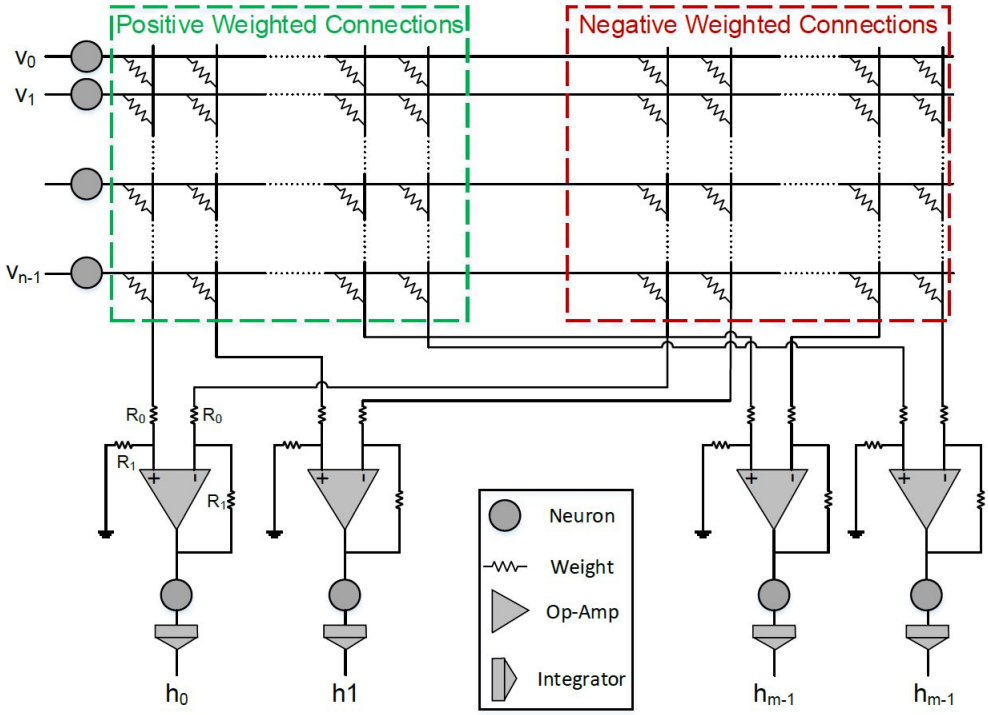

Fig. 3. An $n \times m$ RBM hardware implementation. Two resistive arrays are leveraged along with differential amplifiers to implement both positive and negative weights. The embedded MRAM-based neurons are used to evaluate the activation functions. The fluctuating output voltage of the neurons are integrated through an RC circuit to generate the output of the proposed RBM structure.

4.1 Probabilistic Inference Network Simulator (PIN-Sim)

In order to automate and scale up the design space exploration of DBNs at the circuit-level, we have developed a hierarchical simulation framework called PIN-Sim, which can be utilized to implement any probabilistic learning networks. The block diagram of the PIN-Sim framework used to implement DBNs in our work is shown in Figure 5, which is comprised of five primary blocks. The PIN-Sim methodology is described in Algorithm 2. First, we have modified a MATLAB implementation of DBN developed in [Tanaka and Okutomi 2014] to train the network and obtain the trained weight ($\boldsymbol{W}$) and bias ($\boldsymbol{B}$) matrices according to Algorithm 1. The extracted ($\boldsymbol{W}$) and ($\boldsymbol{B}$) matrices are then applied to a MATLAB module called ***mapWEIGHT***, the functionality of which is described in Algorithm 3. The *mapWEIGHT* module first converts each of the $\boldsymbol{W}$ and $\boldsymbol{B}$ matrices with positive and negative elements to two separate matrices with only positive elements as described below:

$$w^{+}_{(i,j)} = \begin{cases} w_{(i,j)}, & \text{if } w_{(i,j)} \geq 0 \\ 0, & \text{if } w_{(i,j)} < 0 \end{cases}, \qquad w^{-}_{(i,j)} = \begin{cases} 0, & \text{if } w_{(i,j)} \geq 0 \\ -w_{(i,j)}, & \text{if } w_{(i,j)} < 0 \end{cases} \tag{8}$$

$$b^{+}_{j} = \begin{cases} b_j, & \text{if } b_j \geq 0 \\ 0, & \text{if } b_j < 0 \end{cases}, \qquad b^{-}_{j} = \begin{cases} 0, & \text{if } b_j \geq 0 \\ -b_j, & \text{if } w_j < 0 \end{cases} \tag{9}$$

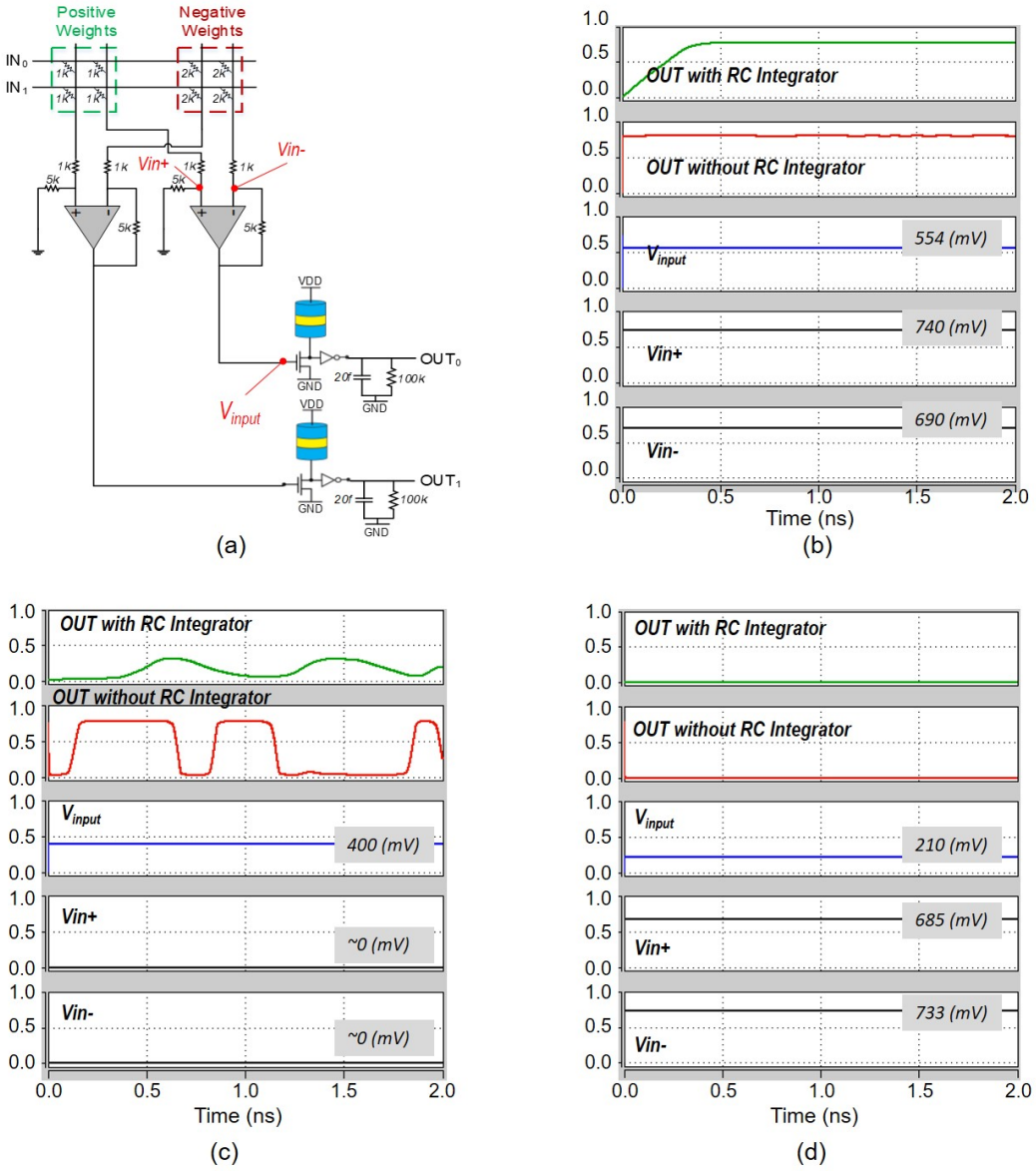

Fig. 4. (a) a 2×2 RBM implementation using the embedded MRAM based neuron. The DC bias voltage of $V_{DD}/2 = 400mV$ is added to the output of the differential amplifier to set our proposed neuron at its midpoint. (b) The behavior of the implemented RBM for $IN_0 = V_{DD}$ and $IN_1 = V_{DD}$ while the positive and negative weight resistances are $1k\Omega$ and $2k\Omega$, respectively. The input voltage connected to the positive terminal of the differential amplifier is larger than the negative terminal resulting in an output voltage larger than VDD/2. The output of the differential amplifier is connected to the input of the neuron, thus the $V_{IN}/V_{DD} =\sim 0.7$ for the neuron leading to output logic "1", as shown in Figure 2 (b). (c) The behavior of the RBM for $IN_0 = 0$ and $IN_1 = 0$. The inputs of the differential amplifiers are near zero, thus $V_{IN}/V_{DD} =\sim 0.5$ and the state of the neuron fluctuates between "0" and "1". (d) The RBM behavior for $IN_0 = V_{DD}$ and $IN_1 = V_{DD}$ while the positive and negative weight resistances are $2k\Omega$ and $1k\Omega$, respectively. The $V_{IN}/V_{DD} =\sim 0.3$ resulting in the neuron being in state "0" according to Figure 2 (b).

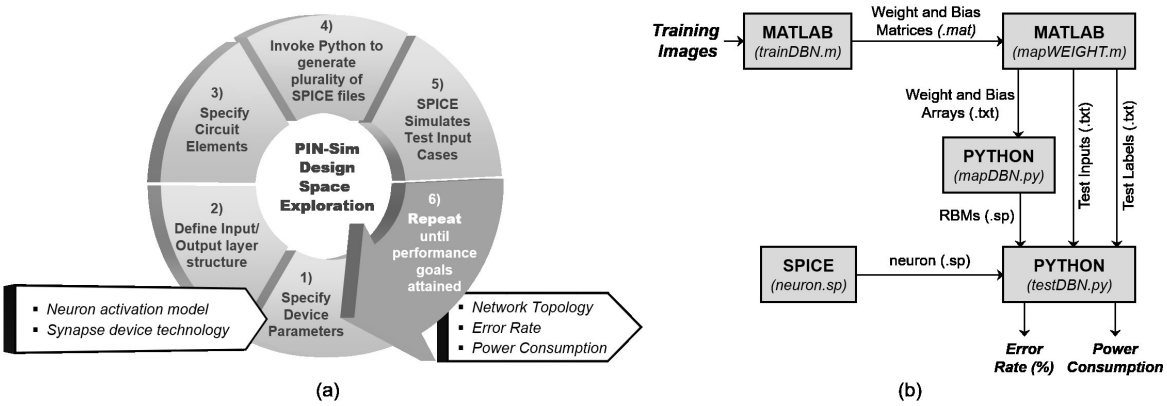

Fig. 5. (a) The PIN-Sim framework can be utilized to explore the design space to realize the optimized network implementation based on the application requirements. (b) The block diagram of the PIN-Sim framework, which consists of five main modules: *(1) trainDBN:* a MATLAB-based module used for training the DBN architecture. *(2)mapWeight:* a module developed in MATLAB that converts the trained weights and biases to their corresponding resistance values. *(3) mapDBN:* a Python-based module which provides a circuit-level implementation of the RBMs using the obtained weight and bias resistances. *(4) neuron:* A SPICE model of the MRAM-based stochastic neuron. *(5) testDBN:* the main module developed in Python that executes test evaluations to assess the error rate and power consumption using the outputs of the other modules in PIN-Sim.

Next, the *mapWEIGHT* module maps the elements in W^+, W^-, B^+, and B^- matrices to their corresponding conductance values using the below equations:

$$\forall w_{(i,j)} \in (W^+, W^-) : gw_{(i,j)} = \frac{(g_{max} - g_{min}) \times (w_{(i,j)} - w_{min})}{w_{max} - w_{min}} + g_{min} \tag{10}$$

$$\forall b_{(i,j)} \in (B^+, B^-) : gb_{(i,j)} = \frac{(g_{max} - g_{min}) \times (b_{(i,j)} - b_{min})}{b_{max} - b_{min}} + g_{min} \tag{11}$$

where $\forall g_{(i,j)} \in \mathbf{G} : g_{min} \leq g_{(i,j)} \leq g_{max}$, in which $g_{min} = 1/r_{max}$ and $g_{max} = 1/r_{min}$ are minimum and maximum conductances of all weighted connections in the crossbar weighted array. Moreover, b_{max}, b_{min}, w_{max}, and w_{min} are the maximum and minimum values in all of the bias and weight matrices, respectively. Finally, Equation 12 is utilized to convert and quantize all of the obtained conductance values to their corresponding resistance values, which can then be utilized to implement the required resistive crossbar array.

$$\forall g_{(i,j)} \in (GW^+, GW^-, GB^+, GB^-) : r_{(i,j)} = \frac{round(Q \times 1/g_{(i,j)})}{Q} \tag{12}$$

where Q is the quantization factor, and GW^+, GW^-, GB^+, and GB^- are positive weight, negative weight, positive bias, and negative bias conductance matrices, respectively.

Once the positive and negative weight and bias resistance matrices are obtained, they will be converted to *text* files and applied to a Python module called ***mapRBM.py***, shown in Figure 5, which produces plural crossbar weighted array circuits in SPICE automatically based on the defined network topology. Finally, a ***testDBN.py*** module is developed using Python scripts, which utilize the generated circuit of the DBN, and the model of the probabilistic neuron to perform a SPICE circuit simulation and calculate the error rate using the *test inputs* and *test labels*, which are provided for the *testDBN* module in form of *text* files.

Algorithm 2: PIN-Sim Methodology

Input: test dataset (D_{test}) with the target labels ($Label$), # of test samples(S), #of RBMs(M), #of nodes in hidden layer x (N_x)
Output: Error Rate
Initialize: $Err = 0$
$weight.mat, bias.mat \Leftarrow$ **Contrastive_Divergence** Algorithm
$posWeight.txt, negWeight.txt, posBias.txt, negBias.txt \Leftarrow$ **mapWeight**($Weight.mat, Bias.mat$)
for $i = 1 : S$ **do**
  $input\_data = D_{test}(i)$;
  for $j = 1 : M$ **do**
    $RBM(j).sp \Leftarrow$ **mapRBM**($input\_data, N_{j+1}, posWeight.txt, negWeight.txt, posBias.txt, negBias.txt$);
    Run $RBM(j).sp$ in HSPICE and store the obtained output voltages in array $outRBM$;
    for $k = 1 : N_j$ **do**
      Run $neuron.sp$ model with $outRBM(k)$ as the input of the k_{th} Neuron;
    end
    Store the output of the neurons in array $OUTPUT$;
    if $(j = M)$ **then**
      if $(OUTPUT \neq Label(i))$ **then**
        $Err+ = 1$;
      end
    else
      $input\_data = OUTPUT$;
    end
  end
end
$ErrorRate = Err/S$;

5 SIMULATION RESULTS AND DISCUSSION

Herein, we have leveraged a hierarchical simulation method to examine the performance of our DBN implementation. In software-level simulation, the behavioral results of the developed embedded MRAM-based neuron model are used to implement a DBN in MATLAB for MNIST pattern recognition application [Lecun et al. 1998]. In the hardware-level simulation, the proposed framework is used to develop a circuit-level DBN implementation using the p-bit SPICE model and 14nm CMOS technology in SPICE circuit simulator with 0.8V nominal voltage.

5.1 MATLAB simulation

Herein, we have modified the sigmoid activation function in a MATLAB implementation of DBN [Tanaka and Okutomi 2014] by using the device-level simulation results of the proposed embedded MRAM-based neuron. To assess the performance of the implemented DBN, we have used the MNIST data set [Lecun et al. 1998] including 60,000 training and 10,000 test sample images of hand-written digits, each of which having 28×28 pixels. We have used Error rate (ERR) and root-mean-square error (RMSE) metrics to evaluate the performance of the DBN, as expressed by the following equations [Tanaka and Okutomi 2014]:

$$ERR = \frac{N_F}{N} \tag{13}$$

Algorithm 3: mapWeight Methodology

Input: $weight.mat, bias.mat$, #of RBMs (M)
Output: $posWeight(n).txt, negWeight(n).txt, posBias(n).txt, negBias(n).txt$, where n is the RBM number
Require: r_{min}, r_{max}, Quantization Factor (Q)
$g_{max} = 1/r_{min}$;
$g_{min} = 1/r_{max}$;
$Q = Q/(r_{max} - r_{min})$
for $i = 1 : M$ **do**
  $W^+, W^- \Leftarrow weight(i)$ Matrix ;
  $B^+, B^- \Leftarrow bias(i)$ Matrix ;
  w_{min} = smallest weight value in W_{pos}, W_{neg} ;
  w_{max} = largest weight value in W_{pos}, W_{neg} ;
  b_{min} = smallest weight value in B_{pos}, B_{neg} ;
  b_{max} = largest weight value in B_{pos}, B_{neg} ;
  $GW^+ = \frac{(g_{max}-g_{min})\times(W^+-w_{min})}{w_{max}-w_{min}} + g_{min}$, $RW^+ = \frac{round(Q\times 1/GW^+)}{Q}$;
  $GW^- = \frac{(g_{max}-g_{min})\times(W^--w_{min})}{w_{max}-w_{min}} + g_{min}$, $RW^- = \frac{round(Q\times 1/GW^-)}{Q}$;
  $GB^+ = \frac{(g_{max}-g_{min})\times(B^+-b_{min})}{b_{max}-b_{min}} + g_{min}$, $RB^+ = \frac{round(Q\times 1/GB^+)}{Q}$;
  $GB^- = \frac{(g_{max}-g_{min})\times(B^--b_{min})}{b_{max}-b_{min}} + g_{min}$, $RB^- = \frac{round(Q\times 1/GB^-)}{Q}$;
  $posWeight(i).txt \Leftarrow RW^+$;
  $negWeight(i).txt \Leftarrow RW^-$;
  $posBias(i).txt \Leftarrow RB^+$;
  $negBias(i).txt \Leftarrow RB^-$;
end

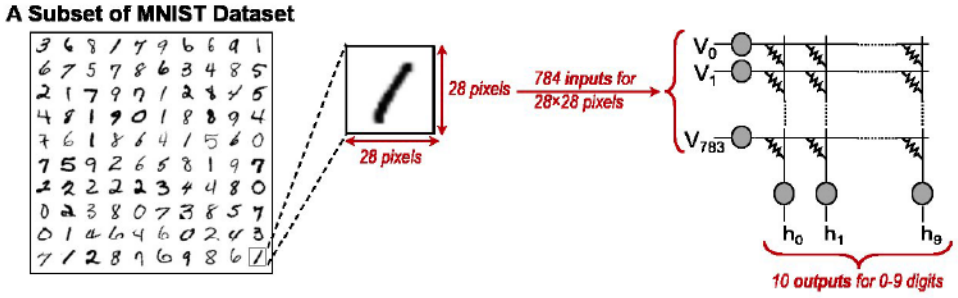

Fig. 6. The most elementary 784 × 10 DBN required for MNIST digit recognition application. The visible layer includes 784 nodes to handle 28 × 28 pixels of the input images, while the 10 nodes in hidden layer represent the output classes.

$$RMSE = \sqrt{\frac{1}{MN}\sum_{k=1}^{N}(y_k - F(x_k)^2)} \tag{14}$$

where M is the number of output classes, N is the number of input data, N_F is the number of false inference, F is the inference of the trained DBN, x_k is the k-th input data and y_k represents its corresponding target output.

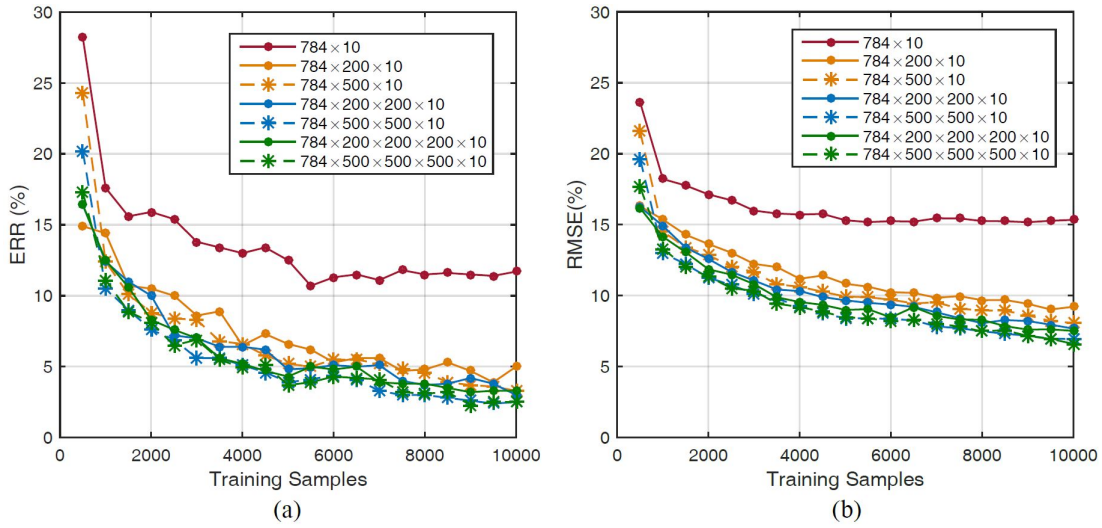

Fig. 7. (a) ERR vs. training samples for various DBN topologies, (b) RMSE vs. training samples for various DBN topologies.

As shown in Figure 6, the most elementary model of the DBN requires 784 nodes in visible layer for the 28×28 pixels of the input images, and 10 nodes in hidden layer for, which represents 0-9 output digits. Figure 7 shows the relation between the error rate and the number of training samples for seven distinct DBN topologies, which is obtained using 1,000 test samples. The results obtained by MATLAB simulation exhibit that an error rate of 28.2% for a 784×10 DBN trained by 500 training inputs can be decreased to a 2.5% error rate achieved using $784 \times 500 \times 500 \times 500 \times 10$ and $784 \times 500 \times 500 \times 10$ DBN topologies, which are trained by 10,000 input training samples. Thus, the recognition accuracy can be improved by increasing the number of hidden layers in the network, number of nodes in each layer, and number of training samples. However, these improvement can lead to higher power consumption and area overheads as investigated in the hardware-level simulations elaborated below.

5.2 PIN-Sim simulation

In this section, we utilize our proposed PIN-Sim framework to provide a circuit-level model of DBN architecture. Next, we will provide the energy and power consumption profiles of the seven different DBN topologies investigated in the previous section to analyze the energy and accuracy trade-offs of these networks. Finally, we will focus on the effect of various important hardware-level parameters. These are vital parameters during design space exploration that influence the accuracy of DBN architectures as tradeoffs necessary to obtain efficient hardware-level implementation for pattern recognition applications.

5.2.1 Power and Energy Consumption Analysis. Figure 8(a) depicts the power consumption of various DBN topologies while evaluating a single input image. As shown, a significant amount of power is consumed in the weighted connections, while less than 10% of the total power is consumed in the neurons of an embedded MRAM-based p-bit approach. For instance, the total power consumption of a $784 \times 200 \times 10$ DBN is approximately equal to 86 mW, only 5.6 mW of which is dissipated in the activation functions. This is achieved by using the proposed power-efficient embedded MRAM-based neurons to implement the activation functions, as opposed to more elaborate floating-point circuits and pseudo-random

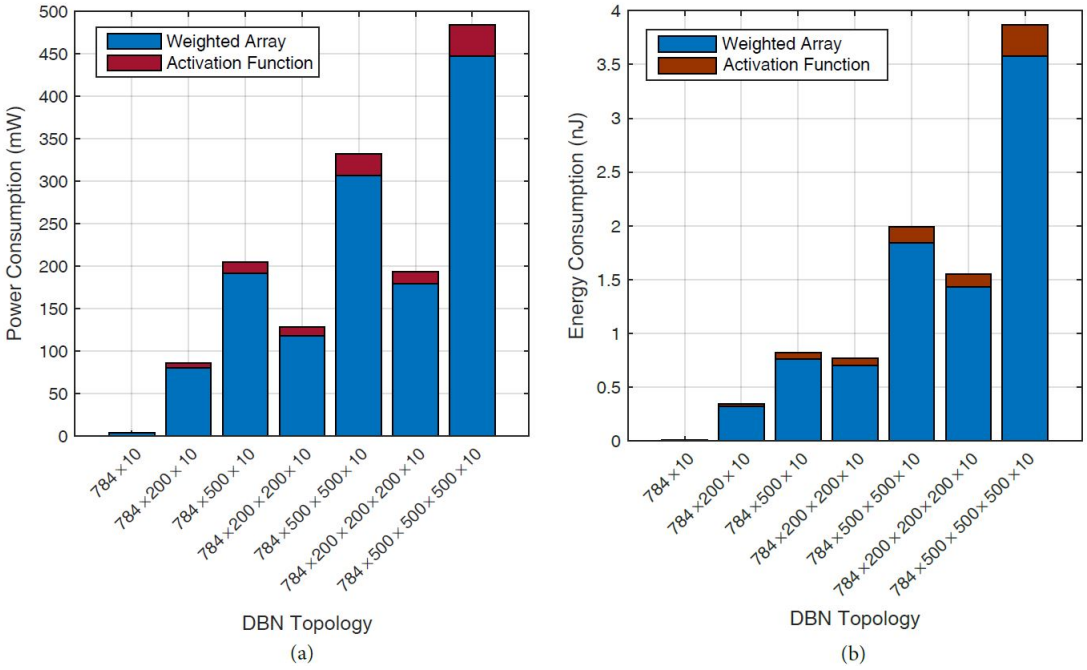

Fig. 8. Test operation: (a) Power Consumption for various DBN topologies, (b) Energy Consumption for various DBN topologies.

Table 2. PIN-Sim tunable parameters and their default values

Parameters	Description	Default Value
$Topology$	Defines the number of layers and nodes	$784 \times 200 \times 10$
$TrainNum$	# of training images	3,000
R_{min}	Minimum resistance of the weighted connections	1 $k\Omega$
ΔR_W	Difference between min and max resistances of weighted connections	400%
Q	Quantization factor	8
R_0, R_1	Resistances of the resistors in the differential amplifiers	1 $k\Omega$, 5 $k\Omega$
R_i, C_i	Resistance and capacitance of the RC integrator circuits	100 $k\Omega$, 20 fF

number generators. Moreover, it is shown that the total power consumption depends primarily upon the aggregate number of neurons that are used in a network and not the number of layers. For instance, the power consumption of a $784 \times 500 \times 10$ DBN is greater than that of a $784 \times 200 \times 200 \times 10$ network, although the latter has higher number of hidden layers. However, the test operation delay is linearly proportional to the number of hidden layers which is determined by the signal propagation and computation progression. In particular, the RC integrator circuit shown in Figure 3 is sampled every 2 ns, leading to an operating clock frequency of 500 MHz and a delay of 2 ns for each RBM. Thus, the $784 \times 200 \times 200 \times 10$ DBN mentioned above requires three clock cycles to complete the evaluation operation, while a $784 \times 500 \times 10$ DBN can produce its output in two clock cycles. Figure 8(b) shows the energy consumption for various DBN topologies, which simultaneously includes the impact of number of nodes and hidden layers on power consumption and delay, respectively.

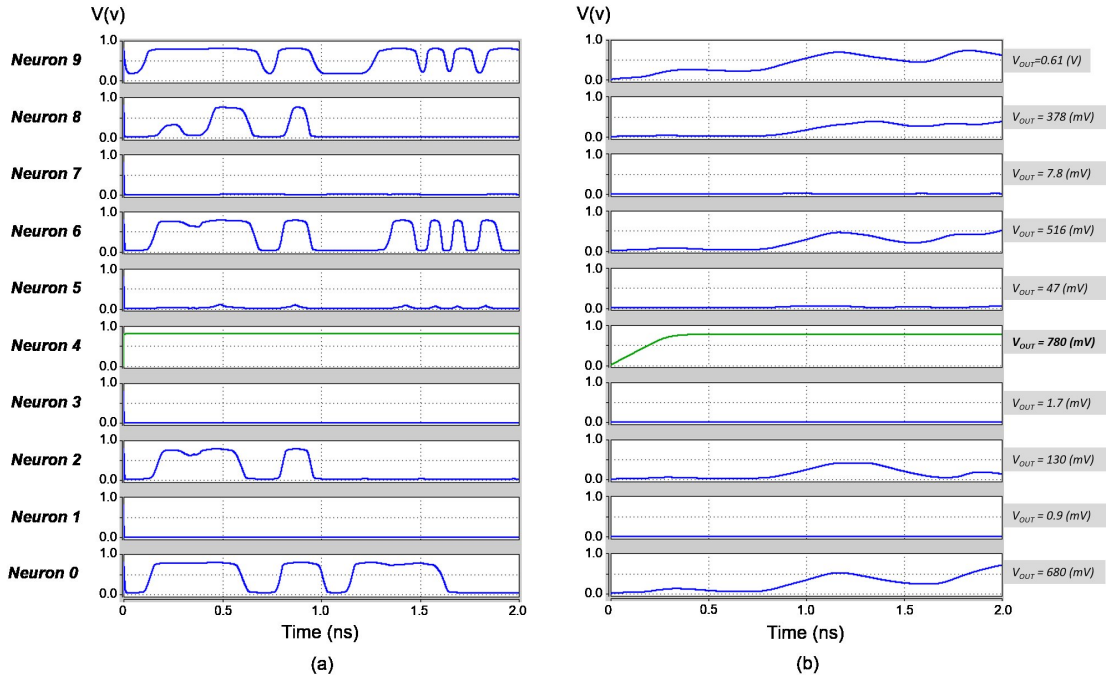

Fig. 9. Output of a $784 \times 200 \times 10$ DBN for a sample digit of "4" in the MNIST dataset: (a) Probabilistic output of the p-bit devices, (b) Output of the integrator circuit. The output voltage of the *neuron-4*, which represents the digit "4" in the output classes, is greater than the other output voltages verifying a correct evaluation operation.

5.2.2 PIN-Sim tunable parameters and their affect on DBN performance. Table 2 lists the tunable parameters in the PIN-Sim framework, which can be adjusted based on the application requirements. The last column of the table shows the default values that are utilized herein for the MNIST digit recognition application. Figure 9 shows the output voltages of the neurons in the last hidden layer of a $784 \times 200 \times 10$ DBN utilized for MNIST pattern recognition tasks, each of which represents an output class. The probabilistic outputs of the p-bit devices are shown in Figure 9(a), while Figure 9(b) exhibits the outputs of their corresponding integrator circuits. The outputs of the integrators are sampled after 2 ns, which is equal to the time constant of the integrator circuit. The output with the highest voltage amplitude represents the class to which the input image belongs. The results obtained exhibit a correct recognition operation for a sample input digit "4" within the MNIST dataset.

Next, we will focus on the effect of some of the tunable parameters on the accuracy and power consumption of DBN architectures implemented by the proposed PIN-Sim framework. First, the effect of ΔR_W is investigated, which defines the possible resistance range of weights and biases as follows, $r_{max} = (1 + \frac{\Delta R_W}{100}) \times r_{min}$. The r_{max} and r_{min} parameters are utilized in the *mapWEIGHT* module in the PIN-Sim tool to map the trained weights and biases to their corresponding resistance values according to Equations 10 and 11, respectively. Figure 10(a) shows the effect of ΔR_W on the recognition accuracy and power consumption of our default $784 \times 200 \times 10$ DBN implementation. As it can be seen in the figure, the error rate is reduced from 53% to 24% by increasing the ΔR_W from 100% to 400%, however a significant change in the error rate cannot be observed for ΔR_W values larger than 400%. These results are particularly

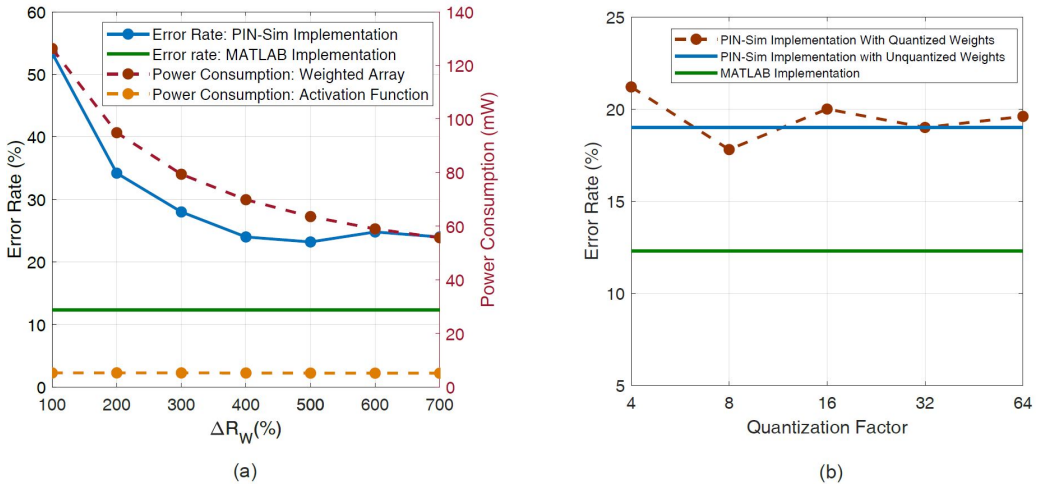

Fig. 10. (a) Error rate and power consumption versus ΔR_W, and (b) error rate versus quantization factor (Q) for a $784 \times 200 \times 10$ DBN trained by 3,000 training images. The software implementation is technology-independent, in which the ideal sigmoid activation function and weight values are utilized in MATLAB to calculate the error rate. Thus, the changes in the tunable parameters used in the circuit-level SPICE implementation do not affect the measured error rates.

beneficial for magnetic tunnel junction (MTJ)-based weighted connections [Roy et al. 2018; Sengupta et al. 2016a], in which the difference between maximum and minimum resistance is defined by the tunneling magneto-resistance (TMR) effect. The results obtained show that a TMR of 400% could be adequate to achieve the desired error rate. However, it is worth noting that this is quite application specific and can vary for different datasets. These results are worthy since the realization of higher TMR values would impose more complex fabrication processes [Parkin et al. 2004], of which 700% [Wang et al. 2009] have been demonstrated experimentally and others of 250% [Wang et al. 2018] via current scalable means. Moreover, as it is shown in Figure 10(a), increasing the ΔR_W results in reduced power dissipation in the weighted array, while the power dissipated in activation functions remains almost unchanged. The higher resistance range for the weighted connections increases the overall resistance of the weighted array. Therefore, since the input voltages remain unchanged the current flowing through the synapses will be decreased, which consequently reduces the power dissipated in the weighted array.

In practice, providing an accurate and continuous range of weight resistances at nanoscale is not attainable due to the fabrication complexities and process variation. Therefore, a realistic circuit-level model of the resistive crossbar architecture should leverage quantized weights. Thus, leveraging PIN-Sim framework for design space exploration, we have assigned a quantization factor (Q) parameter, which can be tuned by the user based on the application requirements. Figure 10(b) shows the effect of weight discretization on the recognition accuracy of a $784 \times 200 \times 10$ DBN with ΔR_W of 400% that is trained with 3,000 training samples. As shown, the error rate for the hardware implementation with $Q = 4$, which means the weights are discretized into four equal intervals between R_{min} and R_{max}, is increased to 21.2% from the 19% error rate that is achieved by the DBN with unquantized weights. As it is expected, this increase in the error rate is mainly caused by the information loss that occurs during the discretization. Moreover, implementations with larger Q values result in error rates closer to that of the DBN with unquantized weights, which can also be expected since the discretization intervals are so small that the weight values are getting close to their unquantized values. However, an

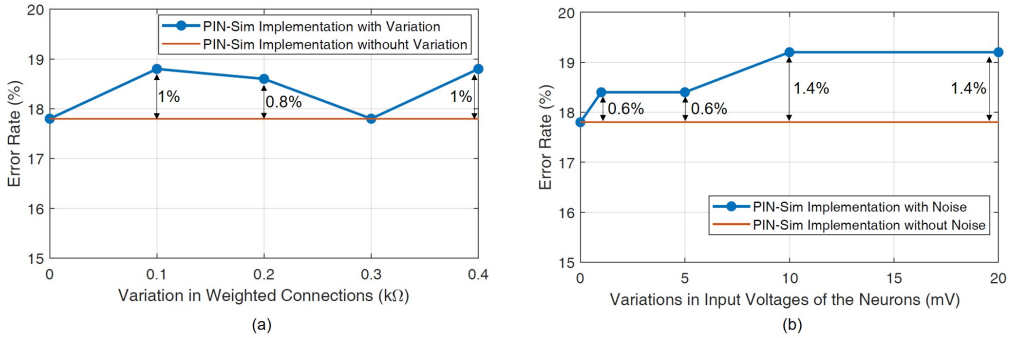

Fig. 11. (a) Error rate versus the variation in the resistance of weighted connections, and (b) error rate versus the variations in the input voltages of the neurons for a 784 × 200 × 10 DBN trained by 3,000 training images.

interesting phenomenon can be observed in the hardware implementation with $Q = 8$, where the error rate of 17.8% is realized which is lower than the error rate of the unquantized DBN. We have performed multiple tests to ensure that this is a repetitive behavior for the DBNs with $Q = 8$, and in all of the cases the error rate obtained was lower than that of the DBN with unquantized weights. These results can be particularly interesting in the hardware-implementation, since for instance in our examined case there is a 0.5 $k\Omega$ gap between various weight resistances, considering the $R_{min} = 1k\Omega$ and $\Delta R_W = 400\%$, which can provide some robustness against process variations without incurring a significant increase in the error rate. In particular, we have investigated the impacts of the variations in the input voltages of neurons, which can be induced by different noise sources, as well as variations in the resistance of the weighted connections on the recognition accuracy of the network. According to the results shown in Figure 11 (a), a 784 × 200 × 10 DBN trained by 3,000 images loses 1% accuracy in presence of variations in weighted connections ranging from 0.1 $k\Omega$ to 0.4 $k\Omega$. Moreover, Figure 11 (b) exhibits 1.4% increase in the error rate for variations in the input voltages of neurons with a standard deviation of 20 mV.

5.3 Discussion

Some of the previous hardware implementations of DBNs are listed in Table 3. The designs proposed in [Kim et al. 2010; Ly and Chow 2010] leverage FPGAs to achieve speedups of 25-145 compared to software implementations, however these approaches suffer from constrained clock frequencies and routing congestion, as well as major resource deficiencies due to the significant embedded memory utilization for both weighted connections and activation functions. In [Yuan and Parhi 2017], those authors have proposed optimization methods to reduce memory requirements for weights and biases, however implementing each activation function still requires dedicated piecewise linear approximator, random number generator (RNG), and comparator circuits which lead to increased area and energy consumption per neuron than the embedded MRAM-based approach herein. In [Ardakani et al. 2017], the low-complexity characteristics of stochastic CMOS-based arithmetic units are leveraged to implement RBM with reduced area and power consumption. However, the large number of linear feedback shift registers (LFSRs) that are required to generate the long input and weight bit-streams results in increased latencies that considerably limits the energy savings.

On the other hand, emerging technologies such as resistive RAM (RRAM) and phase change memory (PCM) have been recently utilized within the crossbar arrays to implement matrix multiplication within RBMs [Bojnordi and Ipek 2016; Eryilmaz et al. 2016; Sheri et al. 2015]. In particular, [Bojnordi and Ipek 2016] has achieved 100× and 10× improvement

Table 3. Various DBN hardware implementations with a focus on activation function structure.

Design	Weighted Connection	Activation Function	Energy per Neuron	Normalized area per neuron
[Kim et al. 2010]	Embedded multipliers	CMOS-based LUTs	N/A	N/A
[Ly and Chow 2010]	Embedded multipliers	- 2-kB BRAM - Piecewise Linear Interpolator - Random number Generator	~10-100 nJ	~ 3000×
[Yuan and Parhi 2017]	- Multiplier - Adder tree	- Piecewise Linear approximator - Random number Generator - Comparator	~10-100 nJ	~ 2000×
[Ardakani et al. 2017]	- LFSR - bit-stream - AND/OR gates	-LFSR - Bit-wise AND - tree adder - FSM-based *tanh* unit	~10-100 nJ	~ 90×
[Sheri et al. 2015]	RRAM Memristor	Off-chip	N/A	N/A
[Bojnordi and Ipek 2016]	RRAM	- 64 × 16 LUTs - Pseudo Random Number Generator - Comparator	~1-10 nJ	~ 1250×
[Eryilmaz et al. 2016]	PCM	Off-chip	N/A	N/A
Proposed Herein	Memristive Devices	Embedded MRAM-based Stocahstic Neuron	Neuron: ~1-10 fJ Integrator: ~10-20 fJ	Neuron: 1× Integrator: ~ 3×

in terms of operation speed and energy consumption, respectively, compared to single-threaded cores by using RRAM devices as weighted connections. In all of the above-mentioned designs, CMOS-based circuits such as multipliers and RNGs are utilized to realize the probabilistic behavior of activation functions. In [Zand et al. 2018], authors have utilized low energy barrier spin-orbit torque (SOT) MTJs to implement the probabilistic sigmoidal activation function, which realizes significant area and energy reductions. However, the current-mode behavior of the SOT-MTJ devices imposes significant power consumption to the activation functions, while requiring weighted connections in $M\Omega$ resistances which can incur significant area overhead and fabrication complexity [Sengupta et al. 2016a; Yuasa et al. 2004]. The work presented herein utilizes a voltage-driven embedded MRAM-based neuron with low energy barrier unstable nanomagnets, which leverages the intrinsic thermal noise to generate sigmoidal probabilistic activation functions required for RBMs within a power-efficient package. As listed in Table 3, the proposed RBM implementation using embedded MRAM-based neuron can achieve approximately three orders of magnitude energy reduction compared to the previous energy-efficient CMOS-based implementations, while realizing at least 90× device count reduction. However, as it was described in previous sections, the embedded MRAM based neuron requires an RC circuit to integrate its output voltage. The SPICE circuit simulation results exhibits an approximate average energy consumption of 10-20 fJ for the RC circuit as listed in Table 3. Moreover, the area required to implement the RC circuit with 100 $K\Omega$ resistor and $20fF$ capacitor is approximately three times larger than that of the MRAM-based neuron [Scott 1998; Stengel and Spaldin 2006]. Thus, the proposed MRAM-based activation function can achieve approximately 20× and 300× area reduction compared to the CMOS-based stochastic neurons proposed in [Ardakani et al. 2017] and [Bojnordi and Ipek 2016], respectively. The area of the MRAM-based neuron, which is utilized as the baseline for the area comparisons, is approximately equal to $32\lambda \times 32\lambda$, that is obtained by the layout design, in which λ is a technology-dependent parameter. Herein, we have used the 14nm FinFET technology, which leads to the approximate area consumption of $0.05\mu m^2$ per neuron. MRAM devices can be fabricated on top of the transistors, thus incurring near-zero area overhead.

6 CONCLUSIONS

Herein, it was shown that embedded MRAM-based neurons with thermally unstable superparamagnetic MTJs can realize a probabilistic output that can be modulated by an input voltage. The magnetization dynamics of the MRAM-based stochastic neuron is modeled by solving the LLG equations for a low energy barrier nanomagnet. The device-level simulations exhibited a desired sigmoidal relation between the input voltages and output probability of the neuron. Once the functionality of the proposed stochastic neuron was verified, we have developed an embedded MRAM-based RBM leveraging two resistive crossbar arrays with differential amplifiers to implement the matrix multiplication operation for both positive and negative weights. SPICE circuit simulations for a 2×2 weighted array validated the functionality of the proposed embedded MRAM-based RBM.

To provide a circuit-level implementation of DBN, we have developed a PIN-Sim framework which is a transportable framework for rapid, automated, and accurate design space exploration of hybrid CMOS and post-CMOS neuromorphic circuits. PIN-Sim is composed of five main modules to train the network, map the trained weights to their corresponding resistances, create the SPICE model of the RBMs, and measure the accuracy and energy consumption. MNIST dataset is utilized to investigate the accuracy and energy tradeoffs for seven distinct DBN topologies implemented by the developed PIN-Sim framework. The simulation results showed that at least two hidden layers are required to achieve suitable error rates. In particular, a $784 \times 200 \times 10$ DBN can realize 5% error rate while consuming less than 500 pJ energy. The error rates could be decreased to 2.5% by using a $784 \times 500 \times 500 \times 500 \times 10$ DBN topologies trained by 10,000 input training samples at the cost of $\sim 10\times$ higher energy consumption and significantly larger area overheads. Moreover, PIN-Sim can be used to optimize network topologies based on different application requirements for energy versus accuracy tradeoffs.

Next, we have focused on the effect of various hardware-level parameters that can be adjusted in the PIN-Sim tool on the performance of the network. One particular parameter which is specifically important for MTJ and RRAM based crossbar architectures is the difference between the largest and smallest possible resistance values in a weighted connection (ΔR_W). It was shown that at least a ΔR_W of 400% is required to realize suitable error rates, however it is worth noting that increasing the ΔR_W to values larger than 400% does not lead to a significant reduction in error rate. Therefore, some fabrication complexities for increasing the ΔR_W in MTJ-based weighted connections can be avoided. Moreover, to realize a realistic hardware implementation we have studied the effect of weight quantization on the accuracy of our network. It was shown that a quantization factor of eight, which provides eight different resistive levels in each weighted connection, can lead to even lower error rates compared to a network with unquantized weights. This also shows the robustness of our proposed circuit-level DBN implementation to minor variations in the resistance of the weighted connections, which is inevitable during the fabrication process. Finally, the comparison results exhibited that the embedded MRAM-based neuron can contribute to several orders of magnitude energy reduction, and reduce the area requirement by 20-fold, with respect to recent energy-optimized designs. Although this is a simulation-based result, hardware realization may endure significant process variation and impacts of sneak currents in large crossbar arrays. While on-chip training can help to mitigate these somewhat, alternate approaches using binarized weights are options explored in other works with varying results [Courbariaux et al. 2015]. To address these further, the development of the PIN-Sim framework provides several possibilities for future work, including: (1) leveraging optimization techniques to reduce the performance gap between the ideal implementation of the DBN using simulation tools such as MATLAB, and the realistic circuit-level implementation of DBN using PIN-Sim framework, (2) training DBNs with binary weights

which can be implemented by MTJs or RRAMs, (3) implementing convolutional DBNs using PIN-Sim for more complex pattern recognition applications.

ACKNOWLEDGMENTS

This work was supported in part by the Center for Probabilistic Spin Logic for Low-Energy Boolean and Non-Boolean Computing (CAPSL), one of the Nanoelectronic Computing Research (nCORE) Centers as task 2759.006, a Semiconductor Research Corporation (SRC) program sponsored by the NSF through CCF 1739635.

REFERENCES

[n. d.]. Predictive Technology Model (PTM) (http://ptm.asu.edu/).

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. 1985. A learning algorithm for Boltzmann machines. *Cognitive science* 9, 1 (1985), 147–169.

A. Ardakani, F. Leduc-Primeau, N. Onizawa, T. Hanyu, and W. J. Gross. 2017. VLSI Implementation of Deep Neural Network Using Integral Stochastic Computing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25, 10 (2017).

Mukund Bapna and Sara A Majetich. 2017. Current control of time-averaged magnetization in superparamagnetic tunnel junctions. *Applied Physics Letters* 111, 24 (2017), 243107.

I.A Basheer and M Hajmeer. 2000. Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods* 43, 1 (2000), 3 – 31. https://doi.org/10.1016/S0167-7012(00)00201-3 Neural Computting in Micrbiology.

Behtash Behin-Aein, Vinh Diep, and Supriyo Datta. 2016. A building block for hardware belief networks. *Scientific reports* 6 (2016).

Sabpreet Bhatti, Rachid Sbiaa, Atsufumi Hirohata, Hideo Ohno, Shunsuke Fukami, and SN Piramanayagam. 2017. Spintronics based random access memory: a review. *Materials Today* (2017).

Chris Bishop, Christopher M Bishop, et al. 1995. *Neural networks for pattern recognition*. Oxford university press.

M. N. Bojnordi and E. Ipek. 2016. Memristive Boltzmann machine: A hardware accelerator for combinatorial optimization and deep learning. In *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*.

Lars Buesing, Johannes Bill, Bernhard Nessler, and Wolfgang Maass. 2011. Neural dynamics as sampling: a model for stochastic computation in recurrent networks of spiking neurons. *PLoS computational biology* 7, 11 (2011), e1002211.

Kerem Yunus Camsari, Rafatul Faria, Brian M Sutton, and Supriyo Datta. 2017a. Stochastic p-bits for invertible logic. *Physical Review X* 7, 3 (2017), 031014.

Kerem Yunus Camsari, Samiran Ganguly, and Supriyo Datta. 2015. Modular approach to spintronics. *Scientific reports* 5 (2015), 10571.

Kerem Yunus Camsari, Sayeef Salahuddin, and Supriyo Datta. 2017b. Implementing p-bits with embedded mtj. *IEEE Electron Device Letters* 38, 12 (2017), 1767–1770.

Miguel A Carreira-Perpinan and Geoffrey E Hinton. 2005. On contrastive divergence learning.. In *Aistats*, Vol. 10. 33–40.

Won Ho Choi, Yang Lv, Jongyeon Kim, Abhishek Deshpande, Gyuseong Kang, Jian-Ping Wang, and Chris H Kim. 2014. A magnetic tunnel junction based true random number generator with conditional perturb and real-time output probability tracking. In *Electron Devices Meeting (IEDM), 2014 IEEE International*. IEEE, 12–5.

Patricia S Churchland and Terrence J Sejnowski. 2016. *The computational brain*. MIT press.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. 2015. BinaryConnect: Training Deep Neural Networks with binary weights during propagations. In *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.). Curran Associates, Inc., 3123–3131. http://papers.nips.cc/paper/5647-binaryconnect-training-deep-neural-networks-with-binary-weights-during-propagations.pdf

R. P. Cowburn, D. K. Koltsov, A. O. Adeyeye, M. E. Welland, and D. M. Tricker. 1999. Single-Domain Circular Nanomagnets. *Phys. Rev. Lett.* 83 (Aug 1999), 1042–1045. Issue 5. https://doi.org/10.1103/PhysRevLett.83.1042

Punyashloka Debashis, Rafatul Faria, Kerem Y Camsari, Joerg Appenzeller, Supriyo Datta, and Zhihong Chen. 2016. Experimental demonstration of nanomagnet networks as hardware for ising computing. In *Electron Devices Meeting (IEDM), 2016 IEEE International*. IEEE, 34–3.

P. Debashis, R. Faria, K. Y. Camsari, and Z. Chen. 2018. Design of Stochastic Nanomagnets for Probabilistic Spin Logic. *IEEE Magnetics Letters* 9 (2018), 1–5. https://doi.org/10.1109/LMAG.2018.2860547

S. B. Eryilmaz, E. Neftci, S. Joshi, S. Kim, M. BrightSky, H. L. Lung, C. Lam, G. Cauwenberghs, and H. S. P. Wong. 2016. Training a Probabilistic Graphical Model With Resistive Switching Electronic Synapses. *IEEE Transactions on Electron Devices* 63, 12 (Dec 2016), 5004–5011.

Akio Fukushima, Takayuki Seki, Kay Yakushiji, Hitoshi Kubota, Hiroshi Imamura, Shinji Yuasa, and Koji Ando. 2014. Spin dice: A scalable truly random number generator based on spintronics. *Applied Physics Express* 7, 8 (2014), 083001.

Robert Hecht-Nielsen. 1992. Theory of the backpropagation neural network. In *Neural networks for perception*. Elsevier, 65–93.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation* 18, 7 (2006), 1527–1554.

Geoffrey E Hinton, Terrence J Sejnowski, and David H Ackley. 1984. *Boltzmann machines: Constraint satisfaction networks that learn*. Carnegie-Mellon University, Department of Computer Science Pittsburgh, PA.

Miao Hu, Hai Li, Qing Wu, and Garrett S. Rose. 2012. Hardware Realization of BSB Recall Function Using Memristor Crossbar Arrays. In *Proceedings of the 49th Annual Design Automation Conference (DAC '12)*. ACM, New York, NY, USA, 498–503. https://doi.org/10.1145/2228360.2228448

S. K. Kim, P. L. McMahon, and K. Olukotun. 2010. A Large-Scale Architecture for Restricted Boltzmann Machines. In *2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines*. 201–208.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (Nov 1998), 2278–2324. https://doi.org/10.1109/5.726791

CJ Lin, SH Kang, YJ Wang, K Lee, X Zhu, WC Chen, X Li, WN Hsu, YC Kao, MT Liu, et al. 2009. 45nm low power CMOS logic compatible embedded STT MRAM utilizing a reverse-connection 1T/1MTJ cell. In *Electron Devices Meeting (IEDM), 2009 IEEE International*. IEEE, 1–4.

Chamika M Liyanagedera, Abhronil Sengupta, Akhilesh Jaiswal, and Kaushik Roy. 2017. Magnetic tunnel junction enabled stochastic spiking neural networks: From non-telegraphic to telegraphic switching regime. *arXiv preprint arXiv:1709.09247* (2017).

Nicolas Locatelli, Alice Mizrahi, A Accioly, Rie Matsumoto, Akio Fukushima, Hitoshi Kubota, Shinji Yuasa, Vincent Cros, Luis Gustavo Pereira, Damien Querlioz, et al. 2014. Noise-enhanced synchronization of stochastic magnetic oscillators. *Physical Review Applied* 2, 3 (2014), 034009.

D. L. Ly and P. Chow. 2010. High-Performance Reconfigurable Hardware Architecture for Restricted Boltzmann Machines. *IEEE Transactions on Neural Networks* 21, 11 (Nov 2010), 1780–1792.

Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. 2014. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* 345, 6197 (2014), 668–673.

Alice Mizrahi, Tifenn Hirtzlin, Akio Fukushima, Hitoshi Kubota, Shinji Yuasa, Julie Grollier, and Damien Querlioz. 2018. Neural-like computing with populations of superparamagnetic basis functions. *Nature communications* 9, 1 (2018), 1533.

Stuart SP Parkin, Christian Kaiser, Alex Panchula, Philip M Rice, Brian Hughes, Mahesh Samant, and See-Hun Yang. 2004. Giant tunnelling magnetoresistance at room temperature with MgO (100) tunnel barriers. *Nature materials* 3, 12 (2004), 862.

M. R. Pufall, W. H. Rippard, Shehzaad Kaka, S. E. Russek, T. J. Silva, Jordan Katine, and Matt Carey. 2004. Large-angle, gigahertz-rate random telegraph switching induced by spin-momentum transfer. *Phys. Rev. B* 69 (Jun 2004), 214409. Issue 21. https://doi.org/10.1103/PhysRevB.69.214409

Kaushik Roy, Abhronil Sengupta, and Yong Shim. 2018. Perspective: Stochastic magnetic devices for cognitive computing. *Journal of Applied Physics* 123, 21 (2018), 210901.

Jack C Sankey, Yong-Tao Cui, Jonathan Z Sun, John C Slonczewski, Robert A Buhrman, and Daniel C Ralph. 2008. Measurement of the spin-transfer-torque vector in magnetic tunnel junctions. *Nature Physics* 4, 1 (2008), 67.

R. Sarikaya, G. E. Hinton, and A. Deoras. 2014. Application of Deep Belief Networks for Natural Language Understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22, 4 (April 2014), 778–784. https://doi.org/10.1109/TASLP.2014.2303296

JF Scott. 1998. High-dielectric constant thin films for dynamic random access memories (DRAM). *Annual review of materials science* 28, 1 (1998), 79–100.

Abhronil Sengupta, Aparajita Banerjee, and Kaushik Roy. 2016a. Hybrid Spintronic-CMOS Spiking Neural Network with On-Chip Learning: Devices, Circuits, and Systems. *Phys. Rev. Applied* 6 (Dec 2016), 064003. Issue 6.

Abhronil Sengupta, Priyadarshini Panda, Parami Wijesinghe, Yusung Kim, and Kaushik Roy. 2016b. Magnetic tunnel junction mimics stochastic cortical spiking neurons. *Scientific reports* 6 (2016), 30039.

A. Sengupta, M. Parsa, B. Han, and K. Roy. 2016c. Probabilistic Deep Spiking Neural Systems Enabled by Magnetic Tunnel Junction. *IEEE Transactions on Electron Devices* 63, 7 (July 2016), 2963–2970. https://doi.org/10.1109/TED.2016.2568762

Ahmad Muqeem Sheri, Aasim Rafique, Witold Pedrycz, and Moongu Jeon. 2015. Contrastive divergence for memristor-based restricted Boltzmann machine. *Engineering Applications of Artificial Intelligence* 37 (2015), 336 – 342.

Massimiliano Stengel and Nicola A Spaldin. 2006. Origin of the dielectric dead layer in nanoscale capacitors. *Nature* 443, 7112 (2006), 679.

Dmitri B Strukov, Gregory S Snider, Duncan R Stewart, and R Stanley Williams. 2008. The missing memristor found. *nature* 453, 7191 (2008), 80.

Brian Sutton, Kerem Yunus Camsari, Behtash Behin-Aein, and Supriyo Datta. 2017. Intrinsic optimization using stochastic nanomagnets. *Scientific reports* 7 (2017), 44370.

M. Tanaka and M. Okutomi. 2014. A Novel Inference of a Restricted Boltzmann Machine. In *2014 22nd International Conference on Pattern Recognition*. 1526–1531. https://doi.org/10.1109/ICPR.2014.271

Mengxing Wang, Wenlong Cai, Kaihua Cao, Jiaqi Zhou, Jerzy Wrona, Shouzhong Peng, Huaiwen Yang, Jiaqi Wei, Wang Kang, Youguang Zhang, et al. 2018. Current-induced magnetization switching in atom-thick tungsten engineered perpendicular magnetic tunnel junctions with large tunnel magnetoresistance. *Nature communications* 9, 1 (2018), 671.

Wenhong Wang, Hiroaki Sukegawa, Rong Shan, Seiji Mitani, and Koichiro Inomata. 2009. Giant tunneling magnetoresistance up to 330% at room temperature in sputter deposited Co 2 FeAl/MgO/CoFe magnetic tunnel junctions. *Applied Physics Letters* 95, 18 (2009), 182502.

Bo Yuan and Keshab K. Parhi. 2017. VLSI Architectures for the Restricted Boltzmann Machine. *J. Emerg. Technol. Comput. Syst.* 13, 3, Article 35 (May 2017), 19 pages. https://doi.org/10.1145/3007193

Shinji Yuasa, Taro Nagahama, Akio Fukushima, Yoshishige Suzuki, and Koji Ando. 2004. Giant room-temperature magnetoresistance in single-crystal Fe/MgO/Fe magnetic tunnel junctions. *Nature materials* 3, 12 (2004).

Ramtin Zand, Kerem Yunus Camsari, Steven D. Pyle, Ibrahim Ahmed, Chris H. Kim, and Ronald F. DeMara. 2018. Low-Energy Deep Belief Networks Using Intrinsic Sigmoidal Spintronic-based Probabilistic Neurons. In *Proceedings of the 2018 on Great Lakes Symposium on VLSI (GLSVLSI '18)*. ACM, Chicago, IL, USA, 15–20. https://doi.org/10.1145/3194554.3194558