The Pennsylvania State University

The Graduate School

EXPLORING EMERGING DEVICE PHYSICS FOR EFFICIENT SPIN-BASED NEUROMORPHIC COMPUTING

A Dissertation in

Materials Science and Engineering

by

Kezhou Yang

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2023

The dissertation of Kezhou Yang was reviewed and approved by the following:

Abhronil Sengupta
Joseph R. and Janice M. Monkowski Career Development Assistant Professor of Electrical Engineering & Computer Science
Dissertation Advisor
Chair of Committee

Saptarshi Das
Associate Professor of Materials Science and Engineering

Swaroop Ghosh
Associate Professor of Electrical Engineering

Joseph Najem
Assistant Professor of Mechanical Engineering

John C. Mauro
Dorothy Pate Enright Professor of Materials Science and Engineering
Chair of Intercollege Graduate Degree Program in Materials Science and Engineering

ABSTRACT

In the past decade artificial intelligence has undergone vast development thanks to deep learning techniques. However, the large computation overhead limits the application of AI in scenarios where area and energy consumption are limited. This is due to the mismatch in architecture between von Neumann hardware computing systems and deep learning algorithms. As a promising solution to the problem, neuromorphic computing has attracted great research interest. While there are efforts to build neuromorphic computing systems based on CMOS technology, memristors which provide intrinsic dynamics similar to synapses and neurons are also under exploration. Among different types of memristors, this dissertation focus on spintronic devices, which offer more plentiful neural or synaptic functionalities with a low operating voltage. The work in this dissertation consists of both simulation and experimental part. On simulation side, a stochastic neuron design based on magnetic tunnel junction utilizing magnetic-electro effect is proposed. The stochastic neurons are used to build spiking neural networks, which show improved spike sparsity with good test accuracy. Apart from spiking neural network, an all-spin Bayesian neural network is proposed, where intrinsic stochasticity of scaled devices is utilized for random number generation. Voltage controlled magnetic anisotropy effect-based magnetic tunnel junction is explored and utilized to solve write sneak path problem in crossbar array structure. On experiment side, Hall bars are fabricated on ferromagnetic/heavy metal materials stacks and utilized as neurons. Relations between Hall bar characteristics and size are explored. Hardware-in-loop training has been studied with Hall bar neurons.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Here I would like to express my appreciation to my parents, who are doing their best to keep me from life pressure and provide me the chance to pursue and finally get my Ph.D. I also would like to thank my advisor, Dr. Abhronil Sengupta, who not only gives me the chance to do research here but also provides a enjoyable environment in the lab. I would like to thank all my committee members, who are Dr. Saptarshi Das, Dr. Swaroop Ghosh and Dr. Joseph Najem, for their valuable suggestions. I would like to thank our supportive staff at MSC, who helped me a lot in the cleanroom. I would like to thank all my friends here, inside and outside the lab for their encouragement and help. Finally, I would like to thank the NSF[1], the funding provider. It is their support that makes my research here possible.

[1] The work is supported by NSF (ECCS 2028213, CCF 1955815, DMR 1905783). The findings and conclusions do not necessarily reflect the view of the funding agency.

Chapter 1

Introduction

In 1965, Gordon Moore stated the doubling of number transistors on microchips every 18 months, which is known as Moore's law and predicted the rapid development of processors and related computing systems in the last few decades [1]. However, due to physical limitation, Moore's law is coming to its end. The conventional scaling of transistors is over due to prominent short-channel effects in small transistors, which prevents the improvement of device performance by simply scaling down transistors [2]. Furthermore, the fabrication cost becomes unaffordable for chips with extremely small transistors. To continue improving computing efficiency, efforts have been made to explore new transistor architectures and materials to continue improving computing performance [3]–[5]. On the other hand, new computing methodologies are also being explored to overcome the limitation of digital circuits. Quantum computing [6], [7], stochastic computing [8], [9] and analogue computing [10] are some examples of new computing methodologies. Among all the branches under exploration, neuromorphic computing is one of the most promising areas and is the topic of exploration for this thesis, which aims to emulate the architecture and function of biological brain to achieve high computing efficiency.

The idea of neuromorphic computing was proposed by Carver Mead in the 1980s [11]. With the rapid development of artificial intelligence (AI) and deep learning techniques in the past decade, neuromorphic computing is attracting more research interest since it provides a promising method to enhance the computing performance of AI systems.

Deep learning techniques enable AI application in a large plethora of areas such as speech processing [12], [13], video object recognition [14] and financial fraud detection [15], among others. However, as the dimensions of dataset grow rapidly, the computational overhead for those

AI systems increases extensively. For example, Alpha Go from Google, which beat human world champion in a Go board game in 2016 [16], consumed ~*MW* of power with 1920 CPUs and 280 GPUs. On the contrary, its opponent, human champion Lee, only needs tens of Watts for the same task. This stark contrast is attributed to the so-called von Neumann bottleneck, which arises from the data transportation between computing unit and memory unit in a von Neumann computer.

1. Von Neumann Bottleneck

A von Neumann computer mainly consists of CPU (central processing unit), memory and input/output devices. CPU, the computing unit, is separated from the memory, where instructions and data are stored. During the computation, the processor gets the instruction and required data from memory via data bus, which causes several limitations on the computing performance, as is shown in Figure **1-1**.

The first limitation is on the data processing speed, which is due to the throughput of data bus connecting CPU and memory as well as the processing speed mismatch between CPU and memory [[17]. The CPU data processing speed is faster than the maximum throughput of data bus. Also, the processor register (small memory cells within the CPU) works faster than memory. As a result, instead of the time required for calculation, the time to transfer data from memory to CPU accounts for most of the latency. Energy consumption is the second limitation. Most of the energy is consumed during data transporting between CPU and memory. For the same amount of data, the energy consumption during data transportation via data bus (~*nJ*) can be 1000 times larger than that required by CPU processing (~*pJ*) [17], [18]. These speed and energy limitations are known as von Neumann bottleneck since they result from the structure of von Neumann computers.

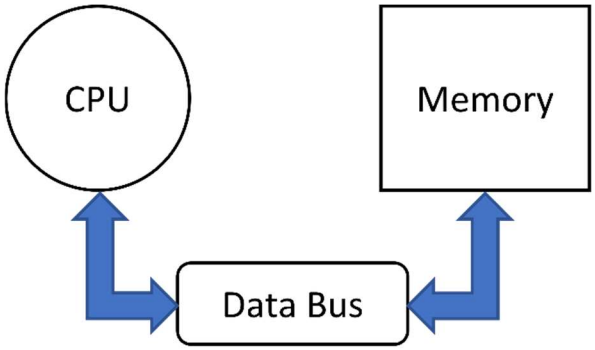

Figure **1-1**: Von Neumann bottleneck results from the separation between CPU and memory, which causes limitation in energy consumption and latency.

2. Advantages of Neuromorphic Computing

As controlling energy and speed expense becomes more and more important for device application such as Internet of Things (IOT) [19], wearable devices [20], among others, where scaled chips with real-time responding capability are needed, neuromorphic computing as a computing paradigm inspired by biological brain is attracting more research interest for high energy efficiency. Neuromorphic computing paradigm avoids the aforementioned von Neumann bottleneck problem. Research has revealed that features provided by neuromorphic computers can be beneficial to deep learning techniques, as they share the same neuron-synapse computing structure.

2.1. In-Memory Computation

Figure **1-2** shows the neuron-synapse model of a neuromorphic computing system. Neurons (circles) are small computing units with a built-in non-linear activation function. Neurons are connected by synapses (lines), which are the memory units of the network. In-memory computation is actualized since the memory units and computing units have in-situ connection. In-

memory computation eliminates the data transportation between the separate computing and memory units, which mitigates the latency and energy consumption caused by the von Neumann bottleneck [21].

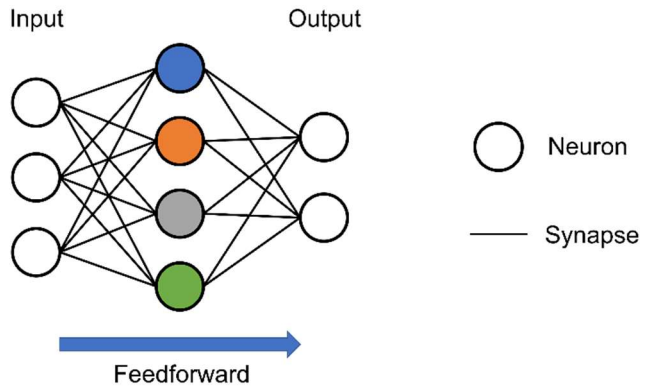

Figure **1-2**: Neuromorphic computing systems adopt neuron-synapse model. Computations are processed in a feedforward manner from input to output. In neuromorphic computing paradigm, multiple computations can be processed parallelly by different neurons in the same layer.

2.2. Massively Parallel Computation

The neuromorphic computing paradigm also enables massively parallel computation. Different from von Neumann systems, where the input instructions are processed sequentially, all neurons and synapses in a neuromorphic computer can process computations parallelly. Figure **1-2** shows an example of parallel computation conducted by a simple fully connected network. The four neurons in the hidden layer perform their own calculations (denoted with different colors) at the same time. Research has shown that with this parallel computing feature, the training process of a large deep neural network with a billion weights can achieve $30000\times$ acceleration on a neuromorphic computing system compared to the state-of-the-art CPU/GPU system [22].

2.3. Inherent Scalability

The structure of neuron-synapse computing model inherently possesses scalability. Since neural networks can be enlarged by adding more neurons and synapses, it is possible to build a large neuromorphic computing system by integrating multiple small neuromorphic chips. This scalability has proven validity for hardware neuromorphic computers such as SpiNNaker [23], [24] and Loihi [25].

2.4. Event-Driven Computation

Apart from the structure, neuromorphic computing paradigm also mimics the behavior of biological brain. Different from traditional analog neural networks (ANNs), where neurons encode and convey information in stable analog values, biological neurons have information encoded in binary spike trains with different temporal patterns [26]. The spike-based neural network is called the spiking neural network (SNN). In a spiking neural network, neurons and synapses are activated only when there are spikes to be generated or conveyed, which enables event-driven computation. Since devices are idle most of the time during computation where spikes are typically sparse, event-driven computation leads to significant energy efficiency [27].

2.5. Stochasticity

The spiking behavior of SNNs enables stochasticity, which is another feature different from standard ANNs. The generation and transmission of spikes in biological neurons and synapses contain randomness, which comes from the noise in the nervous system [28]. SNNs also mimics the probabilistic firing/transferring of spikes and enables stochastic computing, which converts noise in devices from a distractive factor to a computation source [29], [30].

These advantages drive researchers to explore neuromorphic computing paradigms from both algorithm and hardware sides.

3. Spiking Neural Network

3.1. Algorithms Research

Spiking neural network (SNN), as a third generation of artificial neural network (ANN), has attracted great research interest as it is able to capture the spike-based information processing, which is observed in biological brains [31]. Apart from the aforementioned low power consumption due to event-driven computing, SNNs also offer advantages in computing capacity and capability in dealing with information with temporal characteristics, compared to traditional ANNs which consist of perceptrons (first generation) or neurons with non-linear activation function (second generation). It has been proven that SNN is able to provide the same computing capability (if not better) as ANN with significantly fewer computing elements [32]. On the other hand, spikes-based computation is inherently beneficial for temporally distributed information, as spikes are propagated in a temporal manner. In order to build up a spike-based computing paradigm, information needs to be converted into spike trains.

Researchers have not yet come to a conclusion about the best encoding scheme. In fact, different data forms may have their own most suitable encoding scheme. Various different encoding schemes have been observed in biological systems. For example, it is observed that frog muscle responds to external stretching in a rate encoding manner, in which the frequency of fired spike train is related to the applied stretching intensity [33]. While in some retinal, tactile and other systems, information is converted via temporal encoding scheme [34], [35]. Rate encoding and temporal encoding are the most popular schemes. Rate encoding refers to the scheme where the

firing rate of a spike train is utilized to encode information. Rate encoding is frequently used as it is an easy approach for ANN-SNN conversion. However, since a long spike train including spikes is required to obtain the precise firing rate, rate encoding is energy consuming and not suitable for high speed processing. Temporal encoding scheme utilizes the precise timing of spikes to encode information. Typical approaches are time-to-first-spike (TTFS), where time difference between the stimulus and the first fired spike is used, and latency/inter-spike-interval (ISI), where time difference between two spikes is used. Since temporal encoding only needs limited number of spikes, it is capable of high speed information processing [36].

Apart from the encoding scheme, the learning mechanism is also an open problem for SNN. The training process of a neural network is to update the weight values of synapses so that the network is able to generate desirable output after the process. The simplest approach is ANN-SNN conversion [37]–[39], where the weight values of a trained ANN is mapped to an SNN of the same size. ANN-SNN conversion is used in conjunction with the rate encoding scheme to convert the analog values in ANNs to spike trains required by SNNs. This results in the loss of temporal characteristics of SNN, leading to the advantages of SNN not being fully utilized. Online gradient-based learning mechanisms are also developed. Standard ANN gradient descent approach meets difficulties in SNN training. Due to vanishing or exploding gradient, the training is limited to shallow SNNs and small datasets. Efforts have been made to address the problem. One possible solution include threshold-dependent batch normalization (tdBN) method, which is based on spatial-temporal backpropagation (STBP) [40]. In the work, 50-layer SNN was directly trained successfully, and achieved improved accuracy and reduced timestep on large datasets as well. Wei Fang and others induced learnable membrane time constant, which improved the accuracy on nearly all datasets (including large datasets) and reduced the training time [41]. Besides gradient-based learning mechanisms, bio-plausible learnings mechanisms (supervised and unsupervised), which allow for localized learning, are also being explored [42]–[44]. Although the performance

on large-scale tasks is not yet good enough, SNNs trained with spike timing dependent plasticity (STDP) is reported to be desirable in finding clusters in unlabeled data [x].

Despite the challenges remaining in the field, SNNs have already been implemented in many applications such as gesture recognition [45], [46], speech processing [47], [48], bio-medical signal analysis [49], [50], among others.

3.2. Hardware Research

Neuromorphic research and application require the simulation of SNNs, which is time and power consuming on von Neumann architectures. For this reason, researchers are also searching for suitable hardware platform for SNNs.

Standard ANN, which also encounter bottleneck problem when running on von Neumann computers, can be accelerated by graphic processing unit (GPU) or tensor processing unit (TPU). GPU provides parallel computation which alleviates the bottleneck by shifting between multiple instruction threads to avoid latency [51]. TPU reduces reads and writes via "systolic execution" and achieves acceleration and energy efficiency [52]. However, these options are not suitable for SNNs as they are not designed for multi-timestep processing.

The hardware implementation of neuromorphic systems, known as neuromorphic engineering, originates from Carver Mead's proposal, where he excogitated the idea of mimicking biological brain functionalities using analog circuits [11]. Soon after that, the first silicon neuron and silicon retina were implemented [53], [54]. Nowadays, a number of neuromorphic computing projects have been developed. In 2014, IBM proposed its TrueNorth digital chip, including 4096 neural cores with 5.4 billion transistors [55]. The chip is able to compute a SNN with 1 million programmable spiking neurons and 256 configurable synapses. TrueNorth is an inference-only chip, which is suitable for multi-object detection and classification. Intel launched Loihi chip in 2018 [25]. It contains 128 neural cores, each of which includes 1024 primitive spiking neural units,

leading to around 131 thousand neurons and roughly 130 million synapses in total. Loihi chip is capable of both learning and inferencing. There are also other neuromorphic projects such as Tianjic [56], SpiNNaker [23], [24], [57] , BrainScaleS [58], Neuronflow [59], etc. However, all these mentioned projects are still based on CMOS technology. However, CMOS transistors are not the best building block for neuromorphic computing platforms. Since transistors are quite different from neurons and synapses based on bottom-level functionalities, the structure of CMOS-based neuromorphic computing systems is complex, resulting in high power and area consumption. On the other hand, it is more and more difficult to scale down the feature size of transistors due to the physical limit, which slows down the reduction of computing power of CMOS-based systems. For this reason, memristors, which is one of the post-CMOS technologies, is attracting research interest.

A memristor is a memory and a resistor concurrently. For this reason, it should be able to maintain multiple (at least two) resistances, as well as switch between the different resistances. Memristors are suitable for neuromorphic computing for several reasons: 1) Their intrinsic dynamics are similar to those of neurons and synapses [60]–[63]. This enables one memristor device to serve as one neuron or synapse in the system, which vastly reduces the structure complexity as well as area consumption thanks to the small device size; 2) The memristors are non-volatile devices. This means no external energy is required to maintain the resistance in contrast to CMOS-based memories and is beneficial to power consumption; 3) In memory computing, which is the manner of working of biological brain, can be easily realized via crossbar array structure, which enables easy computing of dot-product via Kirchoff's law [64], [65]. Several types of memristors have been proposed, such as resistive random-access memory (RRAM) [66]–[68], phase-change memory (PCM) [62], [65], [69], spintronic devices or in other word, magnetoresistive random-access memory (MRAM) [70]–[72], etc. Spintronic devices, compared to other memristor technologies, offers more plentiful neural or synaptic functionalities with a low operating voltage [71]–[77]. The basic building block of spintronic-based systems is magnetic tunnel junctions (MTJs), which consists of two ferromagnetic layers sandwiching a spacer oxide

layer. By applying spin current or magnetic field accordingly, neural and synaptic functionalities can be achieved by the same device structure, which is beneficial for a compatible system. These characteristics establish spintronic technology as a good candidate for neuromorphic computing.

However, there are still novel physics in spintronic systems discovered but not yet introduced to neuromorphic computing applications. On the other hand, the algorithms developed also need to take those novel device characteristics into consideration. This dissertation aims to bridge the gap between device physics study (more specifically, spintronic device study) and algorithms study.

4. Bayesian Neural Network

Apart from the forementioned SNN, which offers power consumption and computing capacity improvement, there is another branch of improved ANN model called Bayesian neural network (BNN), which is beneficial for probabilistic inference, overfitting problem and decision interpretation [78].

Standard ANNs are not suitable for probabilistic inference tasks, which are essential in decision making process of biological brain, as they do not overtly contain uncertainty in the model. The predictions made by a standard ANN are based on point estimate, which means the network makes predictions with 100% confidence according to its built-in criterion (e.g., pick out the neuron with the minimum output value), even though sometimes several output neurons have close output values.

Overfitting is another problem that may occur in standard ANN, which is likely to happen when a small dataset is fed to a network with too many synapses. Overfitting leads to degradation of network performance on test data while maintaining a good performance on training data.

The lack of interpretation refers to the ignorance of the underlying decision making process in ANNs. Although people are able to develop training algorithms empirically, ANNs are still

black-box models and this prevents the application of ANN in high risk areas, as the results from ANNs are not considered reliable.

BNNs provide improvement on standard ANNs regarding the aforementioned drawbacks. BNNs enable probabilistic inference by replacing point synapses with synapses of which the weight is described by a distribution. For a certain input, the output falls in a distribution decided by the weight distribution. This enables the network to provide not only the output result, but also how "confident" the network is towards the result. The training process of BNNs is according to Bayes' theorem, which sheds light on how neural networks make decisions [79]. Overfitting problem also benefits from the Bayesian framework [80], which leads to natural regularization for model complexity.

Though BNN model offers improvement to ANN in a different way, it also faces the same challenges as SNN model, which is the computing overhead on hardware platforms. This can also benefit from the aforementioned spintronic devices development.

5. Dissertation Statement and Outline

5.1. Dissertation Statement

Neuromorphic computing is a large field involving researchers from many different areas including materials science, electrical engineering, and computer science. In such a large community, it is important to have information from different fields of study shared fluently so that new discovery in one field can benefit research in another field. This dissertation aims to bridge the gap between spintronic device study, where devices utilizing novel physics are explored, and algorithms study, where how spike-based network of devices is built and how training and inferencing are conducted are explored, from both bottom-up and top-down perspectives. Bottom-up perspective refers to how neuromorphic devices are designed to implement novel physics and

how algorithms are modified to comply with the proposed design. Top-down perspective refers to how algorithms can benefit from the characteristics of novel devices. As a summary, the following dissertation statement is given:

This dissertation explores how neuromorphic computing can benefit from novel device physics as well as what corresponding algorithm modifications are required to leverage device benefits.

5.2. Dissertation Outline

Chapter 1 is the introduction part. It clarifies why it is worthwhile to spend resources, time, and energy on neuromorphic computing, including SNN and BNN, as well as the current status of the area.

In chapter 2, the basic knowledge required to understand the work and the methodology adopted in this dissertation will be delivered.

Chapter 3-6 address the main work accomplished. Chapter 3 delivers a stochastic spintronic device proposal which enables independent control of state lifetime, which is used as spiking neurons in the network. The SNN is trained with a backpropagation-based algorithm and adopts temporal encoding. The results show that the proposed SNN achieves high accuracy on MNIST dataset with better spike sparsity compared to rate encoding SNN. Chapter 4 delivers an all-spin Bayesian neural network design, where device stochasticity is leveraged for Gaussian random number generation. Chapter 5 illustrates how synapses utilizing voltage-controlled magnetic anisotropy (VCMA) effect benefit sneak path issue in writing process in a crossbar array structure. Chapter 6 introduces an experimental work, where an SNN built with spin-based devices is used to illustrate the advantages of hardware-in-the-loop training.

Chapter 7 is a summary of the dissertation and provides an outlook for future work.

Chapter 2

Background Knowledge and Methodology

This chapter covers the background knowledge from both algorithm and hardware sides required to understand the dissertation as well as the methodology applied.

1. Standard Artificial Neural Network

Before addressing the SNN, it is necessary to have a brief introduction of standard ANN. The structure of a one-hidden-layer ANN is shown in Figure **1-2.** The first layer is an input layer, which only receives and passes input signals to the next layer through synapses without calculation. Synapses are the connection between neurons. Each synapse possesses a weight value, which is applied to the signal passing through. Neurons conduct calculations based on their built-in activation function and pass the result to synapse connecting the next layer. The last layer is the output layer where results are generated. The process of signals passing from input layer to output layer is called feed forward process. Equation **2.1** describes the calculation conducted by neurons:

$$y_i = f\left(\sum_j w_{i,j} x_j\right) \tag{2.1}$$

In the equation, x_j is the output from presynaptic neuron j. $w_{i,j}$ is the weight value stored in the synapse between neuron j and the postsynaptic neuron i. The summation $\sum_j W_{i,j} x_j$ is the weighted sum from all connected presynaptic neurons. Function f is the non-linear activation function of neuron i and y_i is the output of neuron i. Several activation functions have been studied. In the first generation ANNs, perceptron neuron model is applied, where neuron output is binary, and the activation function is a step function shifted in x direction for different thresholds [81]. Later research has adopted other functions such as sigmoid function [82], [83], ramp saturation function [84], rectified linear unit (ReLU) [85], hyperbolic tangent function (tanh) [86],

etc. as the activation function with analog input and output values. These networks are viewed as second generation ANNs. The inference result via feed forward process is given by Equation **2.2**:

$$\boldsymbol{g}(\boldsymbol{x}) = f^N\left(\boldsymbol{W}^N f^{N-1}(\boldsymbol{W}^{N-1} \dots f^1(\boldsymbol{W}^1\boldsymbol{x}))\right) \tag{2.2}$$

In the equation, $\boldsymbol{x}$ is the input data, which is a vector. $\boldsymbol{W}^i$ is the weight matrix between layer $i-1$ and layer i. f^i is the activation function of neurons in layer i. N is the total number of layers excluding input layer since the input layer does not include any calculation.

Feedforward process is the inference process of the ANNs. To generate desired output, ANNs need to be trained. The training of the network refers to updating the weight values of all synapses so that desired results can be generated by the network. The basic supervised training algorithm is backpropagation.

In supervised training, the weight values are updated under the guidance of a loss function, $L(\boldsymbol{g}, \boldsymbol{y})$, which indicates the difference between current inference result, $\boldsymbol{g}(\boldsymbol{x})$, which is given by Equation **2.2**, and the correct answer, $\boldsymbol{y}$, for the fed training data denoted by $\boldsymbol{x}$. The training process refers to finding the weight values which minimize the loss function by gradient descent method, given by Equation **2.3**:

$$\Delta w_{i,j} = -\eta \frac{\partial L}{\partial w_{i,j}} \tag{2.3}$$

$\Delta w_{i,j}$ is the increment of weight $w_{i,j}$. η is the learning rate, which is a modifiable constant. $\Delta w_{i,j}$, given by Equation **2.3,** always decreases L and enables the minimum of L in a few steps. The derivative $\frac{\partial L}{\partial w_{i,j}^k}$ can be obtained by Equation **2.4**:

$$\frac{\partial L}{\partial w_{j,i}^k} = \frac{\partial L}{\partial o_p^N} \frac{\partial o_p^N}{\partial\left(\sum_q w_{q,p}^N o_q^{N-1}\right)} \frac{\partial\left(\sum_q w_{q,p}^N o_q^{N-1}\right)}{\partial o_q^{N-1}} \dots \frac{\partial o_i^k}{\partial\left(\sum_h w_{h,i}^k o_h^{k-1}\right)} \frac{\partial\left(\sum_h w_{h,i}^k o_h^{k-1}\right)}{\partial w_{j,i}^k} \tag{2.4}$$

o_i^k is the output of ith neuron in the kth layer, according to Equation **2.1**, with $o_i^0 = x_i$. $w_{i,j}^k$ is the weight value of synapse connecting the jth neuron in the kth layer and ith neuron in the $(k-1)$th layer. The loss function L needs to be chosen properly so that the derivative is well defined. A

commonly used loss function is $L = \sum_t^m (y_t - g_t)^2$, where m is the dimension of output. As is indicated by Equation **2.1**, the derivatives are calculated from the last layer to the first layer, which is shown in Figure **2-1**. For this reason, it is called a backpropagation process.

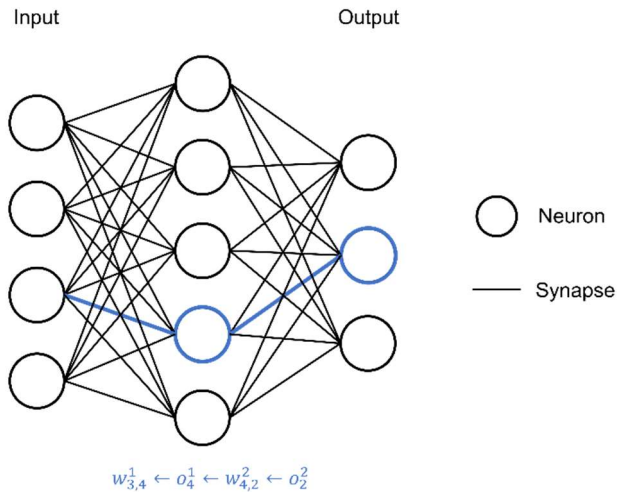

Figure **2-1**: Backpropagation process for weight updating is shown.

2. Spiking Neural Network

Compared to ANN model, SNN is a more bio-plausible model in which information is encoded and conveyed in binary spike trains. This leads to differences from the standard ANN model. For example, in ANN, computations are based on analog values synchronously propagating in the network, while neurons in SNNs receive and calculate spikes, which convey information and arrive at neurons asynchronously. For this reason, neuron models describing how spikes are generated by neurons and algorithms on how learning is conducted based on spikes are explored.

2.1. Neuron Models

Biological neurons are complex. Typically, a biological neuron is composed of a soma to process information, an axon to transmit information to other neurons (output) and dendrites to receive electrochemical signals and transmit information to the soma (input). Among various models to describe the dynamic in a neuron, the Hodgkin-Huxley model proposed in 1952 is the most popular one [87]. In the Hodgkin-Huxley model, a neuron is viewed as a group of electrical elements (capacitor, conductance, current source, and voltage source) describing the dynamic of ion channels. Since the model consists of four non-linear differential equations, it is mainly used for neural systems modeling rather than neural network computing due to the complexity [88], [89]. On the other hand, in most of the ANNs, derivatives of the McCulloch-Pitts neuron (described by Equation **2.1**) are adopted [90].

The Hodgkin-Huxley model and the McCulloch-Pitts neuron are at the two ends of the scale. The Hodgkin-Huxley model is highly bio-plausible but complex, while the McCulloch-Pitts neuron is simple but so abstract that it does not include the spiking behavior. Spiking neuron models for neuromorphic computing stand in the middle. For neuromorphic computing purposes, Integrate-and-Fire (IF) model is widely used [43], [91], [92].

2.1.1. Integrate-and-Fire (IF) and Leaky-Integrate-and-Fire (LIF) Model

IF/LIF model is a simpler spiking neuron model which can be derived from the Hodgkin-Huxley model [31]. The soma is modelled as the circuit structure shown in Figure **2-2**, which receives input spike train $I(t)$ and generates the membrane voltage $u(t)$. The input spike train is split into I_{R} and I_{C}. Once there is an input pulse at time t, the capacitor is charged by I_{C} and membrane voltage $u(t)$ builds up. When there is no input, the membrane voltage $u(t)$ is gradually lowered by the resistive current I_{R}.

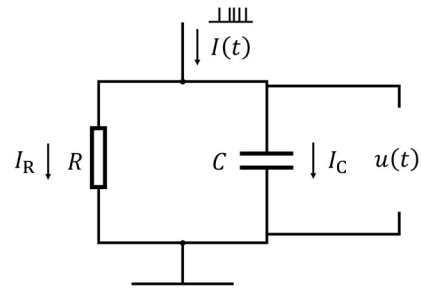

Figure **2-2**: In LIF model, a neuron is viewed as two parallelly connected capacitor (C) and resistor (R) receiving input spike train $I(t)$. The membrane voltage $u(t)$ is the voltage across the capacitor.

The corresponding behavior is described in Equation **2.5**:

$$\tau_m \frac{\mathrm{d}u(t)}{\mathrm{d}t} = -u(t) + RI(t) \tag{2.5}$$

In the equation, $\tau_m = RC$ is the membrane time constant. The first term describes the leakage of $u(t)$ caused by resistor and the second term describes the integration of $u(t)$ due to the capacitor.

The generated membrane voltage $u(t)$ is be compared to a threshold voltage ϑ. An output spike will be generated at time t_f if $u(t_\mathrm{f}) = \vartheta$. After an output spike is fired, the neuron undergoes a refractory period Δ^{abs} , during which the soma does not respond to the input spikes. The refractory period is introduced to prevent a particular neuron from firing excessively. The membrane voltage is reset after output spike firing. Figure **2-3** illustrates the process for a neuron to generate an output spike from an input spike train.

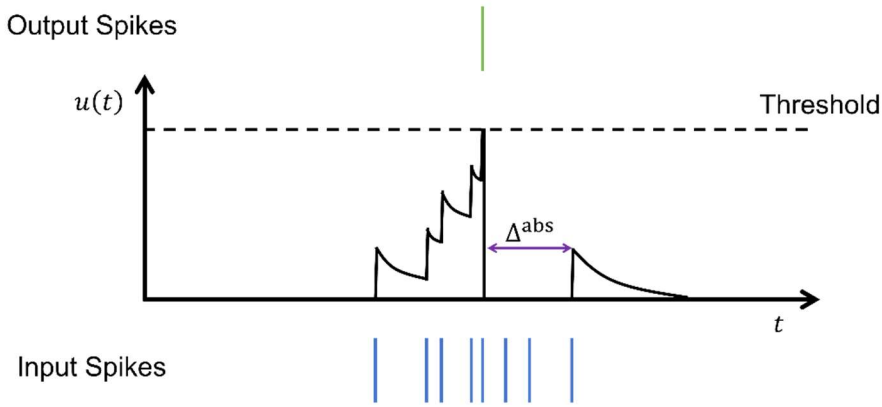

Figure **2-3**: The leaky-integrate-and-fire process is shown. The membrane voltage $u(t)$ builds up when an input spike arrives and decays when input is silent. Once $u(t)$ reaches the threshold value, an output spike is fired. After the spike is fired, the neuron undergoes a refractory period Δ^{abs}.

A generalization of LIF model is non-linear leaky-integrate-and-fire model, which can be achieved by replacing $-u(t)$ and R on the right-hand side of Equation **2.5** into non-linear function of u.

2.1.2.Stochastic Neuron Model

IF/LIF model is deterministic model, which means an input spike train leads to a certain output determined by the intrinsic parameters of the neuron. On the contrary, stochastic neuron model introduces randomness into spike generation process of a neuron [60], [93]. A stochastic neuron generates spikes based on a non-linear function of firing rate related to the neuron input, which is given in Equation **2.6**:

$$P(o_i = 1) = \frac{1}{1 + e^{-\sum_j w_{i,j} o_j}} \tag{2.6}$$

In the equation, o_i is the binary output of neuron i. $P(o_i = 1)$ is the probability to generate an output spike for neuron i with the received input $\sum_j w_{i,j} o_j$, which is the weighted sum of all pre-neuron signals o_j connected by synapse weight $w_{i,j}$. Since the output spike train is generated in a stochastic manner, computations are usually conducted based on statistics such as the average

number of spikes in a period depending on the encoding framework adopted. To easily implement the stochastic neuron model, device noise needs to be leveraged.

Apart from the IF/LIF and stochastic neuron model, there are also other models such as Izhikevich model, Spike response model (SRM), etc. More information can be found in reference [31], [94].

2.2. Synapse Models

Synapses are another important component of neural networks. In a nervous system, synapses outnumber neurons by several orders of magnitude. For this reason, synapse models for neuromorphic computing purposes tend to be simple. In most cases, synapses trained offline only serve as connections between neurons and provide fixed weight values ($w_{i,j}$ in Equation **2.1**) for weighted-sum calculations during inference process. However, things are different when synaptic plasticity is considered. In unsupervised learning framework, spike-timing-dependent plasticity (STDP) is adopted widely as a bio-plausible synaptic learning mechanism [95], [96], where the close pre- and postsynaptic spikes results in large change of synapse weight, as is indicated in Figure **2-4**.

Neuron and synapse models discuss how spikes are generated and conveyed in the neural network. To endow the generated spike trains with meanings, information encoding frameworks are also explored.

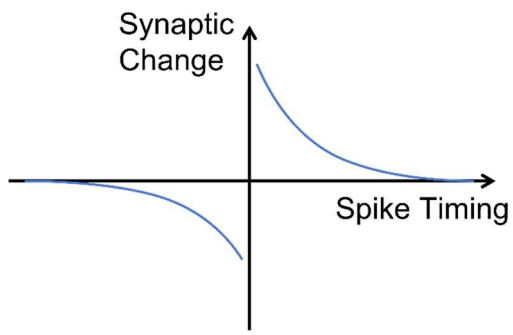

Figure **2-4**: The relation between synapse weight change and spike timing under STDP mechanism is shown. Spike timing (Δt) refers to the spike time difference between postsynaptic spike (t_{post}) and presynaptic spike (t_{pre}), i.e., $\Delta t = t_{\text{pre}} - t_{\text{post}}$.

2.3. Information Encoding Frameworks

Spike trains, in which information can be encoded, are '0's and '1's distributed in a period of time with different temporal patterns. In SNN computation, information encoding refers to how analog values are represented by different spike trains. Inevitably, errors will be induced during the encoding process, and there is always tradeoff between error and computation overhead such as time or energy consumption. Different information encoding frameworks have been studied, among which rate encoding and temporal encoding are the most popular ones [97].

2.3.1. Rate Encoding

Rate encoding refers to the method where information is encoded in the firing rate of a spike train, which was shown in 1926 by E.D. Adrian and Y. Zotterman [33]. There are 3 subcategories of rate encoding method: count rate encoding, density rate encoding and population rate encoding.

Count rate encoding is the simplest method, where information is encoded in the average number of spikes in a period of time. This can be implemented via Poisson encoding, where the spike train is generated on several time steps [92], [98]. At each time step, the normalized analog

value to encode, x, is compared to a uniform random number, $u \sim U(0,1)$. The neuron fires if the analog value wins the comparison ($x > u$). To achieve high encoding accuracy, a long spike train is required for smaller discretization step size, which leads to high inference latency [39], [99].

Density rate encoding refers to the scheme where the same input is fed to a neuron multiple times to obtain the spike density defined as average number of spikes during the period for several runs.

Population rate encoding requires the input to be fed to several neurons. Neurons do not need to have the same input-output relation. The firing rate refers to the average number of spikes during the period among all neurons. With a set of different neurons, analog values, vectors, and function fields can be encoded [100].

However, since the precise timing of spikes is not utilized in the encoding scheme, rate encoding is not beneficial to temporal information processing.

2.3.2. Temporal Encoding

Temporal encoding on the other hand, utilizes the precise timing of spikes to encode information. There are also several temporal encoding frameworks, of which the most common ones are global referenced encoding and latency encoding.

In global referenced encoding scheme, the spike timing to encode information is in reference to a global signal. For example, in rank-order coding (ROC), information is encoded to the firing order of several neurons after the stimulus [36]. However, since only the order of spikes is critical, the timing of spikes is not fully used. This results in limited encoding capacity and poor noise tolerance. As an improvement of ROC, time to first spike (TTFS) coding avoids the problems by encoding the information in the interval between the beginning of stimulus and the first spike firing time of neurons [101], [102], which makes it a simple process and viable in a single-neuron system. TTFS is a bio-plausible coding scheme and has been observed in biological systems such as retinal pathway [34] and human tactile system [35].

With no global signal required, latency encoding or inter-spike-interval (ISI) encoding scheme has the information encoded in the time interval between spikes fired by a group of neurons. This encoding scheme has been found in pyramidal cells [103].

Compared to the rate encoding framework, temporal encoding offers better spike sparsity and encoding capacity. Other temporal encoding method like phase encoding, correlation and synchrony coding and etc. can be found in reference [104]. However, due to the lack of appropriate training algorithms, the performance of SNNs based on temporal encoding scheme is not as good as that of rate encoding based SNNs [102].

2.4. Network Models and Learning Algorithms

With neurons and synapses, a network can be established, in which spikes coded with information are conveyed and computed. Network models describe the network topologies, that is, how neurons and synapses are connected and how information is conveyed in the network. Various kinds of network models have been developed, ranging from highly complicated models to mimic the behaviors of biological systems to simple non-spiking networks, for computing purposes. In neuromorphic computing study, the simplest adopted models are feedforward networks.

Feedforward network can be a fully connected network, in which neurons in adjacent layers have an all-to-all connection, or a convolutional network, where only nearby neurons are connected, forming a "kernel" for convolution calculation. In the inferring process of feedforward networks, data go through the network unidirectionally from input layer to output layer. Supervised and unsupervised learning can be achieved in spiking-based feedforward neural networks.

2.4.1. Supervised Learning Algorithms

Supervised learning refers to the case where neural networks learn from labelled data, as is described in the previous section. Due to the spiking behavior of SNNs, the learning algorithms are different from those of standard ANNs. Several algorithms have been developed for SNNs.

2.4.1.1. ANN-SNN Conversion

ANN-SNN conversion is an off-line learning algorithm for SNN, since the training is conducted on a standard ANN. In order to do ANN-SNN conversion, an ANN with ReLU neurons is trained first. After that, weight values of the trained ANNs are mapped to those of the target SNN with IF/LIF neurons. The reason that ReLU neurons and IF/LIF neurons are applied for ANN and SNN is that the input-output relation of ReLU neurons can be converted to spike rate of input and output spike trains of IF/LIF neurons if proper firing threshold is chosen [39]. ANN-SNN conversion enables SNN to achieve similar accuracy to that of ANN [37]–[39], [105]. However, since the firing threshold of IF/LIF neurons in SNN is related to both inference accuracy and inference latency, determining the optimal threshold value is a challenge.

2.4.1.2. Spike-Based Backpropagation

Backpropagation-based algorithms are also developed for SNNs. The main problem for spike-based backpropagation is that the spike train is nondifferentiable so that the derivative ($\frac{do}{du}$) required in backpropagation algorithm is not well defined, as is indicated in Figure **2-5**. The output signal encounters a step jump when firing condition is fulfilled, leading to a vanishing derivative with discontinuity at threshold membrane voltage. One solution to the problem is to introduce the surrogate gradients or pseudo-derivatives [46], [106], [107]. The derivative ($\frac{do}{dx} = \frac{do}{du} \times \frac{du}{dx}$) required for backpropagation can be calculated. Spike-based propagation has been applied in recent work [108], [109].

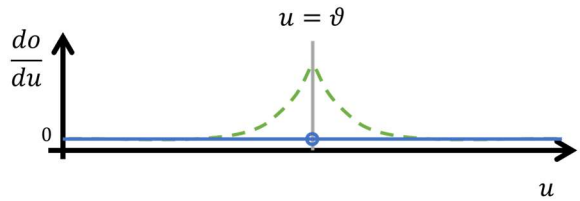

Figure **2-5**: The neuron output o encounters a step jump when membrane voltage u reaches the threshold value ϑ in an IF/LIF neuron, leading to vanishing derivative $\frac{do}{du}$ for $u \neq \vartheta$ and a discontinuity point for $u = \vartheta$ (solid line). To avoid the discontinuity, surrogate gradients/pseudo-derivatives are introduced (dashed line).

2.4.2. Unsupervised Learning Algorithm: Spike Timing Dependent Plasticity (STDP)

Unsupervised learning, in contrast to supervised learning, refers to the case where neural networks recognize patterns from unlabeled data. A widely used unsupervised learning algorithm is spike timing dependent plasticity (STDP) learning rule [95], [110].

STDP rule states that the strength of synapses is related to the relative timing of spikes of pre- and post-synaptic neurons, as is indicated in Figure **2-4.** The closer the timing of pre- and post-synaptic spikes, the larger the change in synapse strength will be. The relation is described by Equation **2.7** [111]:

$$\Delta w = \begin{cases} Ae^{\frac{-(|t_{pre}-t_{post}|)}{\tau}}, & t_{pre} - t_{post} \leq 0 \text{ and } A > 0 \\ Be^{\frac{-(|t_{pre}-t_{post}|)}{\tau}}, & t_{pre} - t_{post} \geq 0 \text{ and } A < 0 \end{cases} \quad \textbf{(2.7)}$$

In the equation, Δw is the change of weight value. A and B are constants. τ is the timing window. t_{pre} is the timing of pre-synaptic spike and t_{post} is the timing of post-synaptic spike.

While STDP has been applied in a large number of research [112]–[114], the accuracy for multi-layer SNNs with STDP is still limited. One possible reason is that the change of weight for each synapse only depends locally on its pre- and post-synaptic neurons, lacking interaction with other parts of the network.

3. Bayesian Neural Network

BNN is another improvement from standard ANN targeting probabilistic inference. While standard ANNs possess deterministic weight values, Bayesian neural networks consider the weights of the network, $\boldsymbol{W}$, to be latent variables characterized by a probability distribution, instead of point estimates, as is shown in Figure **2-6**. More specifically, each weight in such a framework is a random number drawn from a posterior probability distribution (characterized by a mean and variance) that is conditioned on a prior probability distribution and the observed datapoints, D, which is the incoming patterns to the network. During inference, each incoming data pattern will propagate through synaptic weights, each of which is characterized by a probability distribution. Hence, as shown in Figure **2-6**, the final output of the neurons of a particular layer will also be described by a probability distribution characterized by a mean and variance (the uncertainty measure).

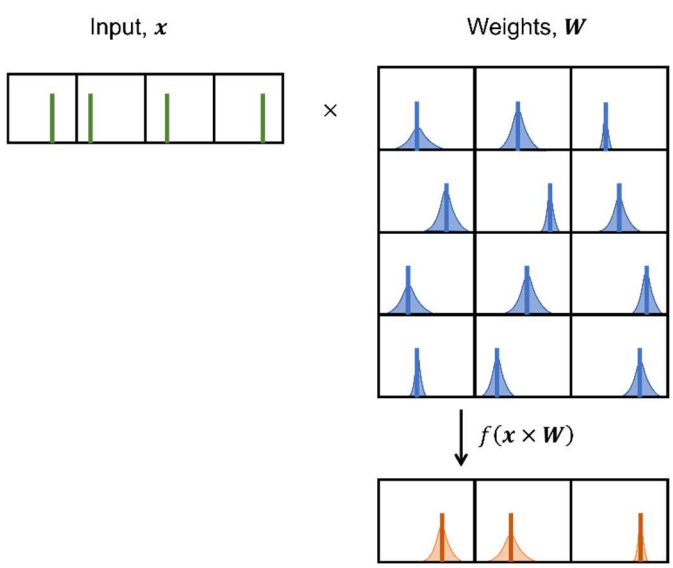

Figure **2-6**: In a Bayesian framework, each synaptic weight is represented by a Gaussian probability distribution (shaded area), which is different from standard ANN, where synaptic weight values

are deterministic values (solid line). The core computing kernel for a particular layer during inference is a dot-product between the inputs and a synaptic weight matrix sample drawn from the individual probability distributions. Learning involves the determination of the mean and variances of the probability distributions using Bayes' formulation.

Bayesian neural networks correspond to the family of deep learning networks where the weights are "learned" using Bayes' rule. The learning process here involves the estimation of the mean and variance of the weight posterior distribution. Following Bayes' rule, the posterior probability can be written as:

$$P(\boldsymbol{W}|D) = \frac{P(D|\boldsymbol{W})P(\boldsymbol{W})}{P(D)} \tag{2.8}$$

where $P(\boldsymbol{W})$ denotes the prior probability (probability of the latent variables before any data input to the network). $P(D|\boldsymbol{W})$ is the likelihood, corresponding to the feedforward pass of the network. In order to make the above-mentioned posterior probability density estimation tractable, two popular approaches are variational inference methods [115] or Markov chain Monte Carlo methods [116].

Posterior distribution is usually difficult to compute analytically. Variational inference methods avoid calculating posterior distribution by approximating it by a Gaussian distribution, $q(\boldsymbol{W}, \theta)$, characterized by parameters, $\theta = (\mu, \sigma)$, where μ and σ represent the mean and standard deviation vectors for the probability distributions representing $P(\boldsymbol{W}|D)$ [117]. In this way, the problem of computing $P(\boldsymbol{W}|D)$ is converted to finding a set of μ and σ so that $q(\boldsymbol{W}, \theta)$ well approximates $P(\boldsymbol{W}|D)$. This is conducted by minimizing Kullback-Leibler (KL) divergence between $q(\boldsymbol{W}, \theta)$ and $P(\boldsymbol{W}|D)$ given in Equation **2.9**, which is the evaluation of similarity between two distributions.

$$\mathrm{KL}(q(\boldsymbol{W}, \theta) \| P(\boldsymbol{W}|D)) = \int q(\boldsymbol{W}, \theta) \log \frac{q(\boldsymbol{W}, \theta)}{P(\boldsymbol{W}|D)} d\boldsymbol{W} \tag{2.9}$$

Equation **2.9** can be reconstructed to Equation **2.10** [78], where the intractable marginal distribution $P(D)$ is separated.

$$\mathrm{KL}(q(\boldsymbol{W},\theta)\|P(\boldsymbol{W}|D)) = -\mathcal{F} + \log p(D) \tag{2.10}$$

In Equation **2.10**, $\mathcal{F} = -\mathrm{KL}(q(\boldsymbol{W},\theta)\|P(\boldsymbol{W})) + \mathbb{E}_q(\log P(D|\boldsymbol{W}))$. Since $\log p(D)$ is independent of $\theta = (\mu, \sigma)$, Equation **2.10** can be minimized by finding appropriate μ and σ via backpropagation process [118].

4. Hardware Platform: Spintronic Device System

In this dissertation, the hardware study is focused on spintronic device system. The basic building block in spintronic hardware system is the magnetic tunnel junction (MTJ), which consists of two nanomagnets sandwiching a spacer layer (typically an oxide such as MgO), as is shown in Figure **2-7.** One of the ferromagnetic layers is called "pinned layer" (PL) because its magnetization direction is "pinned" and does not change during operation. The other ferromagnetic layer is called "free layer" (FL) since its magnetization can be switched freely by an external stimuli like spin current or magnetic field. Depending on the relative orientation of the two magnets, the device exhibits a high-resistance anti-parallel (AP) state (when the magnetizations of the two layers have opposite direction) and a low-resistance parallel (P) state (when the magnetizations of the two layers have the same direction). The resistance difference between P and AP state is evaluated by TMR ratio, which is defined as $TMR = \frac{R_{\mathrm{AP}} - R_{\mathrm{P}}}{R_{\mathrm{P}}}$, in which R_{AP} (R_{P}) is the electrical resistance in AP (P) state. These two states of the magnet are stabilized by an energy barrier determined by the anisotropy and magnet volume.

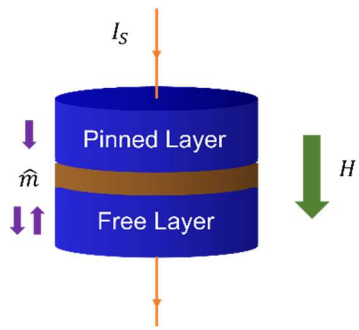

Figure **2-7**: An MTJ structure is shown. Orange arrow indicates the applied spin current I_S, and green arrow indicates the applied magnetic field H. $\hat{m}$ is the unit vector in the direction of magnetization direction of pinned layer/free layer, as indicated by purple arrows. Under I_S or/and H, free layer magnetization can rotate freely, while that of pinned layer is fixed.

4.1. Resistance Difference in AP and P State

One of the core characteristics of MTJ devices is the resistance difference in AP and P state. The state energy of different spins split due to the magnetic field in the system, as is shown in Figure **2-8**a. In this case, one spin direction becomes dominating and most electrons in the layer possess spin in this direction. When there is an applied voltage, electrons are able to tunnel through the oxide spacer between two ferromagnetic layers. The resistance difference accrues from the density of state (DOS) relation between the two ferromagnetic layers.

If the MTJ is in P state (Figure **2-8**b), the two ferromagnetic layers have the same dominating spin direction. In this case, electrons can tunnel easily from free layer on the right to pinned layer on the left, since majority/minority spin electrons (spin-up/spin-down) in free layer can be accepted by majority/minority spin states in pinned layer (spin-up/spin-down). This leads to low electrical resistance.

Things are different if the MTJ is in AP state (Figure **2-8**c). In this situation, mismatch occurs in spin states in pinned layer and free layer. Majority spin electrons in free layer (spin-down)

can only be accepted by minority spin states in pinned layer (spin-up) and vice versa. This makes the electron tunneling from free layer to pinned layer difficult as most of electrons are deflected, which results in high electrical resistance.

A more detailed explanation can be found in reference [119].

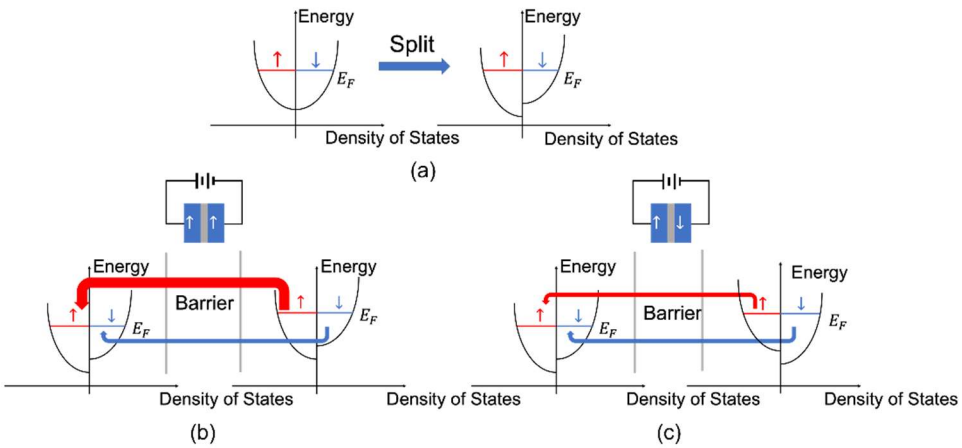

Figure **2-8**: Illustrative density of states (DOS) relation in an MTJ device is shown. E_F refers to Fermi energy. (a) Energy of electrons in different spin directions splits due to magnetic field. (b) Electrons can easily tunnel from free layer (right) to pinned layer (left) because of the matching electron states, leading to low electrical resistance. (c) The mismatch in spin states makes it difficult for electrons to tunnel from free layer to pinned layer, which results in high electrical resistance.

4.2. MTJs of Different Sizes

The MTJs of different sizes have different behavior.

4.2.1. Large MTJs: Deterministic Devices

For a magnet with elongated shape, multiple domains can be stabilized in the FL, thereby leading to the realization of multiple stable resistive states. Such a domain-wall (DW) MTJ consists

of a DW separating the two oppositely magnetized regions and the DW position is programmed to modulate the MTJ resistance (due to the variation in the relative proportion of P and AP domains in the device) [120]. Figure **2-9** shows structure of a multi-domain device. In this device, the free layer contains two domains, which are filled with blue and yellow color with white arrow indicating the magnetization direction.

In magnetic heterostructures with high perpendicular magnetocrystalline anisotropy (for example, the interface between ferromagnetic (FM) layer and heavy metal (HM) layer in Figure **2-9**), spin–orbit coupling and broken inversion symmetry stabilizes the chiral DWs through Dzyaloshinskii–Moriya interaction (DMI) [121], [122]. Such an interfacial DMI at the magnet–HM interface results in the formation of a Néel DW. When an in-plane charge current is injected through the HM, the accumulated spins at the magnet–HM interface results in the Néel DW motion.

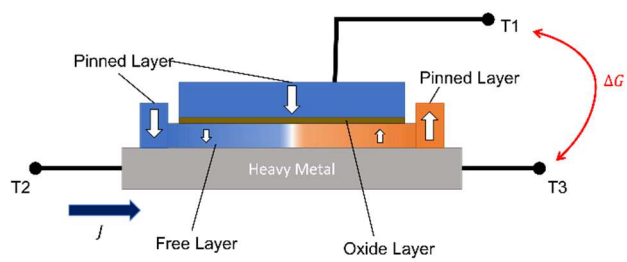

Figure **2-9**: A multi-domain MTJ device is shown. J is the magnitude of current flowing through HM. ΔG is the change in conductance between T1 and T3 results from J.

As shown in Figure **2-10**(a), for a given programming time duration, the current flowing through the HM underlayer causes the DW displacement proportional to its magnitude. The DW position determines the magnitude of the MTJ conductance. The MTJ conductance varies linearly with the DW position since it determines the relative proportion of the area of the parallel and antiparallel domains of the MTJ [see Figure **2-10**(b)]. Since such a device can be programmed to multilevel resistive states and is characterized by low switching current requirements and linear

device behavior (device conductance change varies in proportion to the magnitude of programming current), they are an ideal fit for implementing crossbar-based "in-memory" computing platforms. Experimentally, a multilevel DW motion-based resistive device was recently shown to exhibit 15–20 intermediate resistive states [123].

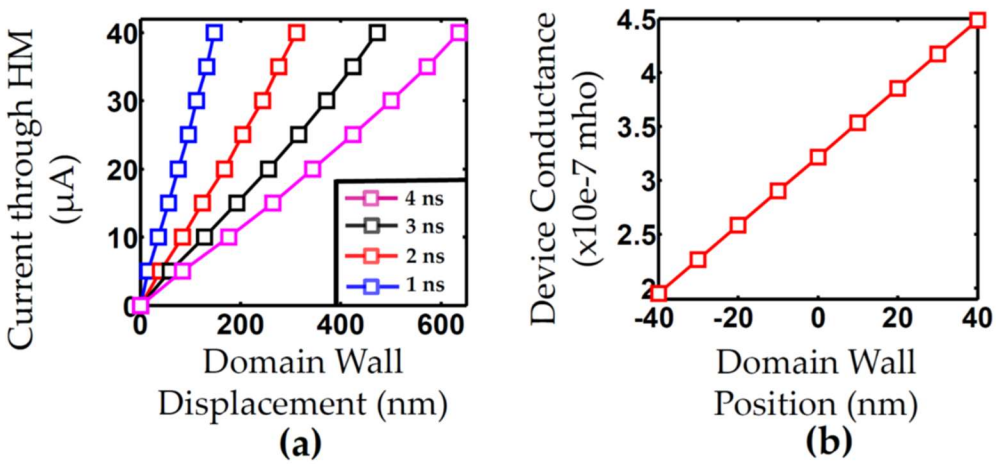

Figure **2-10**: Device characteristics are shown. (a) Programming current versus domain wall displacement profile and (b) device conductance versus domain wall position profile are shown for a 20-nm-wide and 0.6-nm-thickmagnet calibrated to experimental measurements [121]. The device characteristics illustrate that the programming current magnitude is directly proportional to the amount of conductance change [120].

4.2.2.Scaled MTJs: Stochastic Devices

When devices scale down, there can only be one domain in free layer. Also, barrier height which stabilizes AP and P states also decreases since it is volume-related. As a result, when barrier height is so small that it can be overcome by thermal noise alone, free layer magnetization no longer stays in a certain AP or P state. Instead, it exhibits stochastic switching behavior between the two states. Figure **2-11** depicts the temporal magnetization dynamics of a $\sim 2k_BT$ (k_B is the Boltzmann constant and T is the absolute temperature) barrier height magnet. The magnet resides in the P and

AP states with characteristic lifetimes τ_P and τ_{AP}. The lifetime of the device in each state can be controlled by the magnitude or direction of an external current flowing through the magnetic stack [124]. At zero bias, the lifetimes are equal and determined by the magnet barrier-height, as is indicated by Figure **2-11**(a). Note that this is a first-order modeling. In practical device implementation, 50% switching probability may not be achieved exactly at zero bias current due to the presence of device imperfections, stray fields, and other non-idealities. With the application of an external "write" current (induced by V_I in **Figure** 2-11b), the magnitude of the firing rate of the neuron, $\tau = \frac{\tau_{AP}}{\tau_{AP}+\tau_P}$, gets modulated. The rate of spiking of the neuron varies in a nonlinear sigmoid fashion with respect to the input current [125].

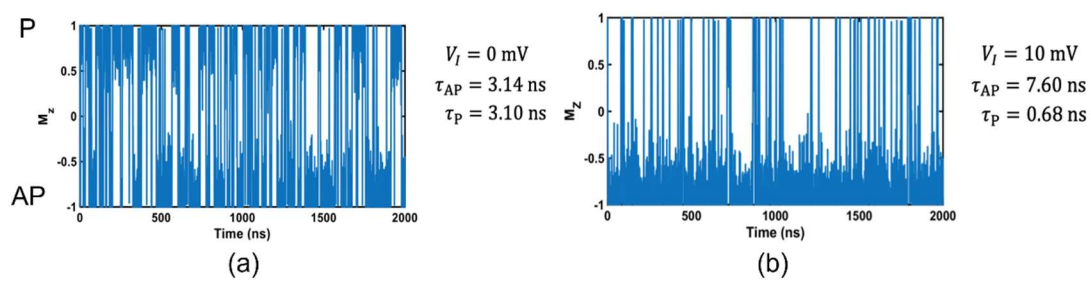

Figure **2-11**: Stochastic switching of free layer magnetization for a $\sim 2k_BT$ barrier height magnet is shown. M_z is the easy axis direction component of normalized free layer magnetization. (a) P and AP state lifetime τ_P and τ_{AP} are equal under zero bias. (b) The state lifetime can be modified by external bias.

4.3. Simulation: Landau–Lifshitz–Gilbert Equation

The behavior of magnetization under applied external stimuli (magnetic field and/or spin current) can be simulated by Landau-Liftshiz-Gilbert (LLG) equation, which is given in Equation **2.11**:

$$\frac{d\hat{\boldsymbol{m}}}{dt} = -\gamma(\hat{\boldsymbol{m}} \times \boldsymbol{H}_{\text{eff}}) + \alpha\left(\hat{\boldsymbol{m}} \times \frac{d\hat{\boldsymbol{m}}}{dt}\right) + \frac{1}{qN_s}(\hat{\boldsymbol{m}} \times \boldsymbol{I}_s \times \hat{\boldsymbol{m}}) \quad \textbf{(2.11)}$$

where $\hat{\boldsymbol{m}}$ is the unit vector of free layer magnetization, $\gamma = \frac{2\mu_{\text{B}}\mu_0}{\hbar}$ is the gyromagnetic ratio for electron (μ_{B} is Bohr magneton and μ_0 is vacuum magnetic permeability), α is Gilbert's damping ratio, $\boldsymbol{H}_{\text{eff}}$ is the effective magnetic field including thermal noise, shape anisotropy field for elliptic disks and applied magnetic field. Thermal field is modeled as $\boldsymbol{H}_{\text{thermal}} = \sqrt{\frac{\alpha}{1+\alpha^2}\frac{2K_{\text{B}}T}{\gamma\mu_0 M_{\text{S}} V \delta_t}}\, G_{0,1}$, where $G_{0,1}$ is a Gaussian distribution with zero mean and unit variance and δ_t is the simulation time step [126]. $N_s = \frac{M_{\text{S}}V}{\mu_{\text{B}}}$is the number of spins in free layer of volume V (M_{S} is saturation magnetization), and $\boldsymbol{I}_s$ is the input spin current. The equation describes the motion of $\hat{\boldsymbol{m}}$ via $\frac{d\hat{\boldsymbol{m}}}{dt}$, which is the velocity of $\hat{\boldsymbol{m}}$ on unit sphere. The terms on the right hand side are three contributions to $\frac{d\hat{\boldsymbol{m}}}{dt}$. The first term describes how magnetic field manipulates the motion of $\hat{\boldsymbol{m}}$. The second term is a damping term, which represents the tendency of $\hat{\boldsymbol{m}}$ to slow down and return to the stable state (i.e. the easy axis direction). The third term describes the contribution caused by spin current. By feeding appropriate device parameters, such as saturation magnetization, size, etc., LLG equation is able to provide free layer magnetization behavior verified by experiment.

4.4. Crossbar Array Structure

Multiple MTJs or other memristors can be mounted to a crossbar array structure, which enables easy calculations of dot-product [72], [127], [128]. Assuming each synapse to be represented by a MTJ, as shown in Figure **2-12**, they can be arranged in a crossbar structure. Each row of the array is driven by an analog voltage [output of digital-to-analog converters (DACs)] that corresponds to the magnitude of the input. The current flowing through each synapse is scaled by

the conductance of the device and due to Kirchhoff's law (Equation **2.12**), and all these currents get summed up along the column, thereby realizing the dot-product kernel.

$$I_j = \sum_i V_i \cdot G_{i,j} \tag{2.12}$$

In the equation, V_i is the input voltage signal, I_j is the output current signal, and $G_{i,j}$ is the conductance of MTJs in the array. Dot-products can be inherently calculated.

Note that negative synaptic weights can also be mapped by using two horizontal lines per input (driven by "positive" and "negative" supply voltages). In case a particular synaptic weight is "positive" ("negative"), then the corresponding conductance in the "positive" ("negative") line is set in accordance with the weight. The resultant currents get summed up along the column and pass as the input "write" current through the neurons in the next layer.

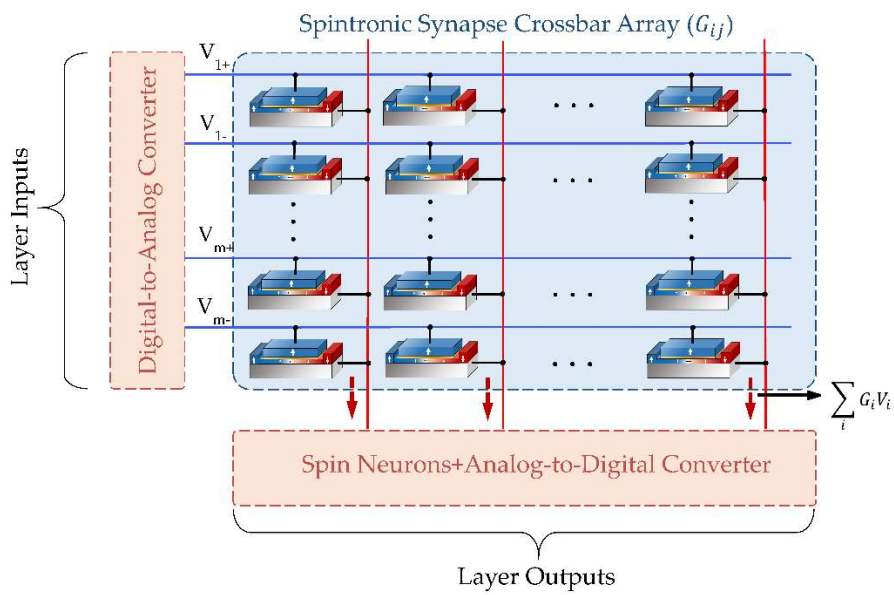

Figure **2-12**: Crossbar array structure is shown. Each MTJ possesses conductance $G_{i,j}$. The input is provided by voltage signals V_i from horizontal bars. The output is generated by current signals $I_j = \sum_i V_i \cdot G_{i,j}$ from vertical bars.

5. Device Characterization Techniques

This dissertation also includes experimental study, which is based on Hall bar structure.

5.1. Test Structure: Hall Bars

The Hall bar device structure, shown in Figure **2-13**, consists of an HM layer and a FM layer. When a charge current flows through the HM/FM structure, spin current is generated due to primarily two effects – 1) Spin Hall effect in the HM and 2) the Rashba effect at the FM/HM interface [129], [130]. The spin Hall effect originates from both intrinsic sources such as band structures and extrinsic sources like Mott scattering by impurities [131], [132]. Additionally, electrons moving in the interfacial electric field at HM/FM interface experience a magnetic field, known as the Rashba field, which introduces a spin-orbit term in the system Hamiltonian. The induced spin-orbit interaction splits the band of different spins, which enables non-zero net spin accumulation under applied electric field [133]–[135]. The generated spin current induces FM magnetization switching when charge current reaches switching current [136]. In this way, Hall bar devices possess the same underlying physics for switching behavior, which makes it a perfect test structure for spintronic device study. Due to thermal noise, the switching is probabilistic rather than deterministic when pulse current is applied. Thus, at any given time, the switching probability depends on the magnitude of the pulse current, which follows a sigmoidal function. This provides the non-linearity required for the operation of the neuron.

5.2. Hall Bar Fabrication

Hall bar devices are fabricated through a process including photolithography/Ebeam-lithography, etching and contact pad deposition. The fabrication process starts with samples with

deposited materials stack (Si/SiO2(300nm)/Ta (5nm)/CoFeB (1nm)/MgO (2.5nm)/Ta (2.5nm /5nm)). Bars with ≥ 1μm width are defined by photolithography, using SPR-3012 as photoresist. Hall bars with bar sizes smaller than 1μm are defined by E-beam lithography, using polymethyl methacrylate (PMMA) as resist layer. The defined patterns are etched by Ar until the Si/SiO2 substrate. The residual resist after etching is removed by soaking in PRS-3000 under 80℃ water-bath. Ti (5nm)/Au (100nm) stack are deposited as contact layer by E-beam evaporation after bars are fabricated.

5.3. Four-Probe Measurement

To characterize the devices, four-probe measurements are conducted. As is shown in Figure **2-13**, the write/read current pulses are applied through current channel and voltage difference is measured between voltage terminals. Current pulses are generated by Keithley 6221. The read current is set to be 50μA so as not to disturb the device state. The voltage difference is due to anomalous Hall effect (AHE) and measured by Keithley 2000 Multimeter. Hall resistance is defined by Equation **2.13**,

$$R_{\mathrm{AHE}} = \frac{V_{\mathrm{AHE}}}{I_{\mathrm{Channel}}} \tag{2.13}$$

Here, V_{AHE} is the voltage difference across the two voltage terminals, and I_{Channel} is the read current pulse amplitude flowing through the current channel.

The whole measuring system is placed on a probe station which is able to isolate environmental vibration.

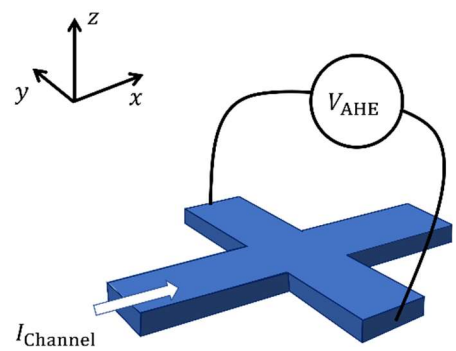

Figure **2-13**: Hall bar device structure is shown**.** To conduct a four-probe measurement, a read current I_{Channel} is passed through the current channel, and the voltage difference caused by AHE V_{AHE} is measured.

5.4. Magnetic Anisotropy Field Estimation

Magnetic anisotropy can be estimated by sweeping the applied in-plane magnetic field. The device magnetization gently tilts under the applied in-plane field, which causes a small deviation in Hall resistance. The deviating Hall resistance results in a bending hysteresis loop, which can be used to estimate magnetic anisotropy field. Figure **2-14(**a) shows the geometry of the sample surface, magnetization, and applied field. Figure **2-14**(b) shows a typical hysteresis loop for $\boldsymbol{H}_x$ sweeping measurement. The magnetization direction under in-plane field can be calculated by Equation **2.14**,

$$\frac{\partial E}{\partial \theta} = 0 \qquad \textbf{(2.14)}$$

Here, $E = -K_{\mathrm{eff}} \cos^2 \theta - M_{\mathrm{S}} \boldsymbol{H}_x cos\theta$ is the magnetic energy density. $\boldsymbol{H}_x$ is the applied in-plane field, θ is the angle between magnetization direction $\hat{m}$ and z direction, K_{eff} is the perpendicular magnetic anisotropy (PMA) energy density and M_{S} is the saturation magnetization. This leads to the following equation:

$$\frac{R_{\mathrm{AHE}}}{R_0} = 1 - \frac{1}{2}\left(\frac{1}{H_{\mathrm{an}}}\right)^2 \boldsymbol{H}_x{}^2 \tag{2.15}$$

Here, $H_{\mathrm{an}} = \frac{2K_{\mathrm{eff}}}{M_{\mathrm{S}}}$ is defined as the anisotropy field, R_0 is the Hall resistance when $\boldsymbol{H}_x$=0 and R is the Hall resistance during field sweeping. H_{an} can be estimated by fitting the equation to the obtained data, as is shown in Figure **2-14**(c).

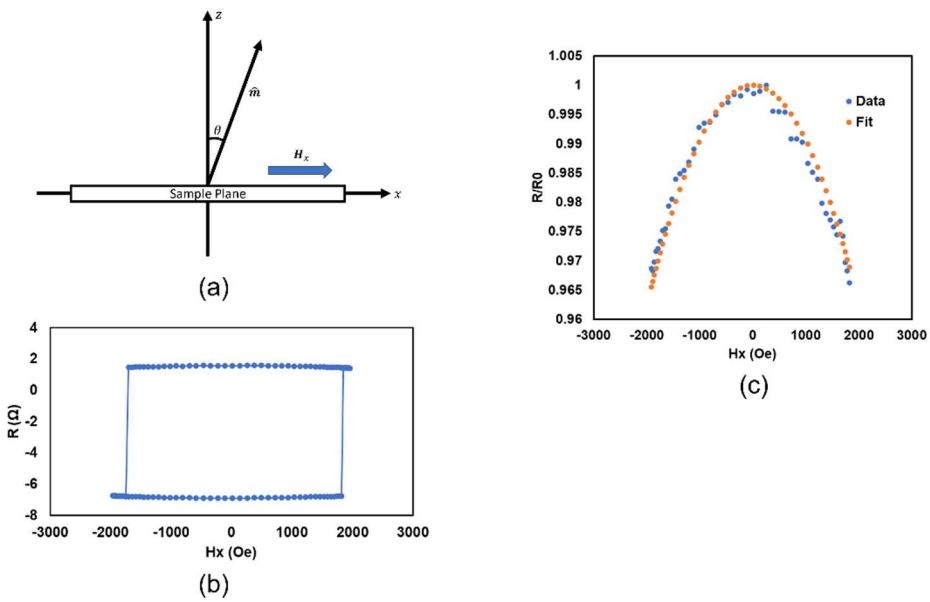

Figure **2-14**: (a) The directions of applied magnetic field, sample plane and sample magnetization are shown. $\hat{m}$ is the unit vector in the direction of sample magnetization. $\boldsymbol{H}_x$ is the applied in plane field. z is the normal direction of sample surface and θ is the angle between z direction and $\hat{m}$. x and z are in the same direction as indicated in Figure **2-13**. (b) The measured hysteresis loop is shown. The tilting of magnetization causes the bending of the curve. (c) The fitting process between data and the curve given by Equation **2.15**.

Chapter 3

Leveraging Probabilistic Switching in Superparamagnets for Temporal Information Encoding in Neuromorphic Systems

Brain-inspired computing - leveraging neuroscientific principles underpinning the unparalleled efficiency of the brain in solving cognitive tasks - is emerging to be a promising pathway to solve several algorithmic and computational challenges faced by deep learning today. Nonetheless, current research in neuromorphic computing is driven by our well-developed notions of running deep learning algorithms on computing platforms that perform deterministic operations. In this chapter, it is argued that taking a different route of performing temporal information encoding in probabilistic neuromorphic systems may help solve some of the current challenges in the field. The chapter considers superparamagnetic tunnel junctions as a potential pathway to enable a new generation of brain-inspired computing that combines the facets and associated advantages of two complementary insights from computational neuroscience – how information is encoded and how computing occurs in the brain. Hardware-algorithm co-design analysis demonstrates 97.41% accuracy of a state-compressed 3-layer spintronics enabled stochastic spiking network on the MNIST dataset with high spiking sparsity due to temporal information encoding.

1. Motivation

As is discussed in the previous chapters, SNN, as a neuromorphic computing paradigm, is promising for enabling low-power, asynchronous "compute only when needed" neuromorphic hardware. While SNNs have shown initial promise as such alternative computing paradigm, significant challenges remain from both the algorithms and hardware perspective to ensure scalability in terms of key performance metrics like recognition accuracy, hardware power, energy

and area efficiency. Most prior studies have used smaller sub-problems or have converted non-spiking Deep Neural Networks (DNNs) to SNNs [92] - a non-optimal approach in demonstrating the abilities of SNNs. Currently, SNNs remain very similar to non-spiking networks with the analog neural computation in DNNs distributed as binary information over time in the case of a spiking neuron – with the temporal aspect remaining largely unexploited. This has significantly limited SNN efficiency in large-scale problems [137]. In order to address these limitations, the solution is formulated against two complementary backdrops.

1.1. Information Encoding (Goal - Enhanced Sparsity and Reduced Latency)

The vast majority of SNN algorithm formulations have been based on rate coding [46], [138]. However, in temporal-encoding, the precise time duration required to spike is believed to encode the neuron output information. The principal advantages of using temporal encoding [139] for modelling spiking behavior are multiple. Since information is now transmitted in precise spike timings instead of the signal rate, such neural codes can be sparse and much faster to avoid temporal-averaging effect.

1.2. Computing Paradigm (Goal - State-Compressed Hardware)

The computing perspective is motivated by a bottom-up hardware viewpoint that emerging technologies like spintronics exhibit stochastic switching behavior (due to thermal noise) at room temperature, especially at aggressively scaled dimensions [71], [140]. The potential benefits of such a computing framework from the hardware implementation perspective is that they allow multi-level neural/synaptic state compression to single bit (in turn, leading to scaled device implementations) due to the additional probabilistic encoding of information. However, such stochastic SNNs have been mostly utilized in the rate encoding framework.

In order to leverage the benefits of increased information capacity in SNNs for enhanced power, latency and energy metrics and simultaneously to utilize the advantages of state-compressed hardware enabled by these nanomagnetic devices, this chapter explores a device-algorithm co-design approach – where we explore the implementation of spintronics enabled stochastic SNNs bearing temporal domain encoding of information.

2. Leveraging the Dynamic Temporal Behavior of MTJs

As is discussed in previous chapter, due to thermal noise and low barrier height, aggressively scaled MTJs, which are superparamagnets, lose non-volatility and exhibit spontaneous stochastic switching behavior that can be characterized by P and AP state lifetime τ_{P} and τ_{AP} and spike rate $R = \frac{\tau_{\mathrm{AP}}}{\tau_{\mathrm{AP}}+\tau_{\mathrm{P}}}$. The main advantage of transitioning to a superparamagnetic system would lie in the faster operating speeds and asynchronous operation [125]. However, careful peripheral circuitry design, sensitivity to noise and variations remain open challenges. In addition to neuromorphic applications [71], [74], [125], [141]–[143], stochasticity inherent in magnetic devices (superparamagnets or higher barrier height magnets) have been leveraged to implement true random number generators [144], and even for other unconventional computing platforms like Ising computing, quantum-inspired algorithms, combinatorial optimization problems, on-chip temperature sensors, among others [140], [145]–[147]. While the intrinsic temporal dynamics of superparamagnets have been utilized in certain applications like Ising computing, the vast majority of neuromorphic SNN applications have primarily leveraged the superparamagnetic device characteristics in the rate encoding regime, i.e. the continuous-time dynamic behavior of superparamagnets have been ignored and the time-averaged behavior has been used from the computing perspective. This leads us to the question - *Can the unique probabilistic switching*

behavior of superparamagnetic devices be utilized for temporal information encoding in stochastic SNNs?

The answer to this question is yes. In order to design a magnetic device where the intrinsic physics is able to support temporal information encoding, one needs to precisely control the device lifetimes τ_{P} and τ_{AP}. This is difficult in a superparamagnet under sole external current stimulation. As shown in Figure **2-10**, the external current magnitude (modified by V_I) controls the time averaged firing rate of the device and both the device lifetimes get modulated together with change in the external current magnitude. Under the application of $V_I = 10\mathrm{mV}$, τ_{P} changes from $3.10\mathrm{ns}$ to $0.68\mathrm{ns}$ and τ_{AP} changes from $3.14\mathrm{ns}$ to $7.60\mathrm{ns}$.

However, as explained in LLG equation (Equation **2.11)**, the magnetization dynamics is a function of both external current and external magnetic field which opens up the possibility of tuning the two device lifetimes by two separate independent control knobs. When an external "write" voltage is applied to the MTJ (resulting in spin-torque) along with an external magnetic field, the lower MTJ resistance in the P state results in much larger modulation of τ_{P} than τ_{AP} due to an external voltage, as is shown in Figure **3-1**. Consequently, the external spin current can be used to control τ_{P}. On the other hand, the magnetic field can be used to tune τ_{AP} by manipulating the energy profile. In this manner, under certain conditions [148], independent control of τ_{P} and τ_{AP} can be realized by adjusting the externally applied magnetic field and current. Recent experiments [149] and theoretical modelling [148] have shown that such a controlling scheme can be realized in a CoFeB MTJ stack within a range of applied field and current.

However, using an external tunable magnetic field to bias MTJ spiking neurons is not feasible from the scalability perspective for neuromorphic computing applications. Significant energy consumption would be required to generate the field. Additionally, since a tuned magnetic field is required for each specific neural magnet, this would limit the magnet spacing to avoid stray

field effects. A potential alternative path can be to design novel device structures exploiting emerging devices physics like the magnetoelectric effect [150].

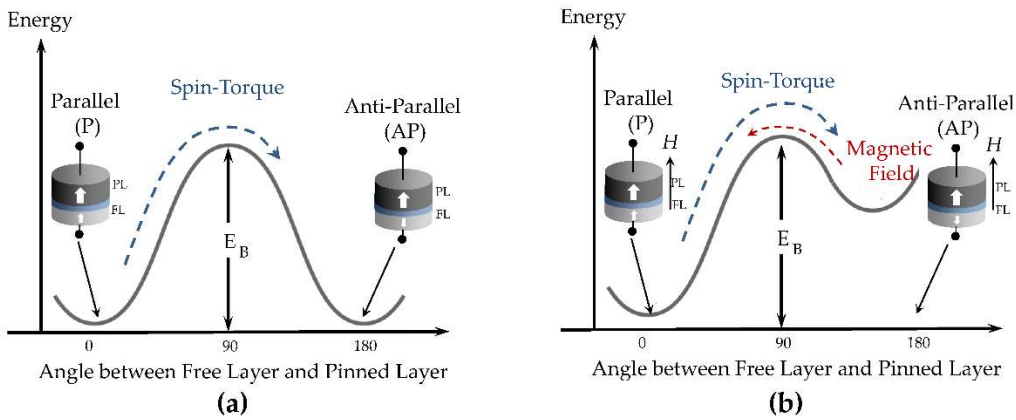

Figure **3-1**: (a) In the absence of a magnetic field, the device spiking rate is modulated by the spin-torque generated by an external "write" current. (b) Application of a magnetic field and spin-torque allows for independent control knobs for the individual device lifetimes.

2.1. Magnetoelectric Effect

Recent experiments on multilayered stacks consisting of a multiferroic material lying underneath a magnetic layer have revealed that a transverse magnetic field is induced in the nanomagnet lying on top due to the application of a voltage across the multi-ferroic material. This is attributed to Magneto-electric Effect (ME) [151]. ME generates from coupling between the spin polarization and the electric polarization of the material [152]. The coupling is induced by Dzyaloshinskii–Moriya (DM) interaction, bringing an additional Hamiltonian given in Equation **3.1** via a DM vector:

$$H_{i,j} = \boldsymbol{D}_{i,j} \cdot \left(\boldsymbol{S}_i \times \boldsymbol{S}_j\right) \tag{3.1}$$

In the equation, $H_{i,j}$ is the Hamiltonian of spin $\boldsymbol{S}_i$ and $\boldsymbol{S}_j$. $\boldsymbol{D}_{i,j}$ is the DM vector for $\boldsymbol{S}_i$ and $\boldsymbol{S}_j$ system. To minimize the total energy of the system, the spins are canted for a small angle, which gives rise to a weak ferromagnetism, as is indicated in Figure **3-2**.

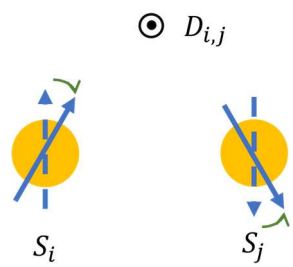

Figure **3-2**: The DM interaction on a two-spin ($\boldsymbol{S}_i$ and $\boldsymbol{S}_j$) system is shown. The DM vector causes a small canting angle from the original direction of spins.

DM interaction occurs in crystal structures with certain symmetries [153], [154]. ME has been observed in multiferroic materials such as $BiFeO_3$ [155]. The applied electric field causes the displacement of bismuth ions inside $BiFeO_3$, followed by the rotation of oxygen octahedra. The shift of the ions results in the direction switching of ferroelectric polarization and magnetization. The switching of magnetization of $BiFeO_3$ acts as a bias to the contacting nanomagnet via exchange bias at the interface. Note that this is just one possible route for realizing ME based devices. Other types of ME induced switching mechanisms [150] can be potentially leveraged for our device design.

The probabilistic switching characteristics of an MTJ under the application of spin current and ME can be analyzed by the LLG equation given in Equation **2.11**. The spin current favors the AP state and is induced by an electric field applied across the MTJ stack, $I_{\mathrm{S}} = \frac{V_I}{R_{\mathrm{MTJ}}}$, where V_I is the applied voltage to generate spin current, I_{S}, and R_{MTJ} is the resistance of the MTJ stack. ME effect is usually modeled by considering the effect of an external magnetic field acting on the magnet. The magnitude of the field is directly proportional to the applied voltage [152], [156], [157], with

the proportionality factor being a material property. Note that this is agnostic to the underlying origin of ME and such a first-order relationship between applied voltage and induced magnetic field dependency has been extensively used for modeling and benchmarking magneto-electric devices [152], [156], [157]. The applied magnetic field which favors P state due to the ME effect is given by Equation **3.2**:

$$\boldsymbol{H}_{\mathrm{ME}} = \left(0, \quad 0, \quad \frac{1}{\mu_0}\alpha_{\mathrm{ME}}\frac{V_{\mathrm{ME}}}{t_{\mathrm{ME}}}\right) \tag{3.2}$$

where α_{ME} is the ME constant, t_{ME} is the thickness of ME layer, and V_{ME} is the voltage across the ME layer. This ME field $\boldsymbol{H}_{\mathrm{ME}}$ is considered a component of the effective magnetic field $\boldsymbol{H}_{\mathrm{eff}}$ in Equation **2.11**. It is worth noting here that the resistance of P state is smaller than that of AP state. Hence, the spin current is larger in the P state than in the AP state with the same applied V_I. Thus, a small variation in V_I leads to a large change in spin current in P state. As a result, the V_I (V_{ME}) control knob dominates τ_{P} (τ_{AP}) variation. The asymmetric impact of each external voltage on τ_{P} and τ_{AP} enables the independent control of the device lifetimes by applying two independent external control voltages. Note that the device can be still used to perform rate encoding by not utilizing the V_{ME} control knob.

2.2. Device Design

The magneto-electric effect can therefore be exploited to envision three-terminal device structures shown in Figure **3-3**. The device consists of an MTJ stack lying on top of an ME oxide layer (for instance, $BaTiO_3$ or $BiFeO_3$). Sufficient voltage (V_{ME}) applied across the ME oxide induces an effective magnetic field on the nanomagnet lying on top. On the other hand, the voltage applied across the MTJ, V_I, controls the device lifetime τ_{P}. Typical device simulation parameters for a $2k_{\mathrm{B}}T$ barrier height magnet have been used from prior literature [125], [157], and are tabulated in Table **3-1**.

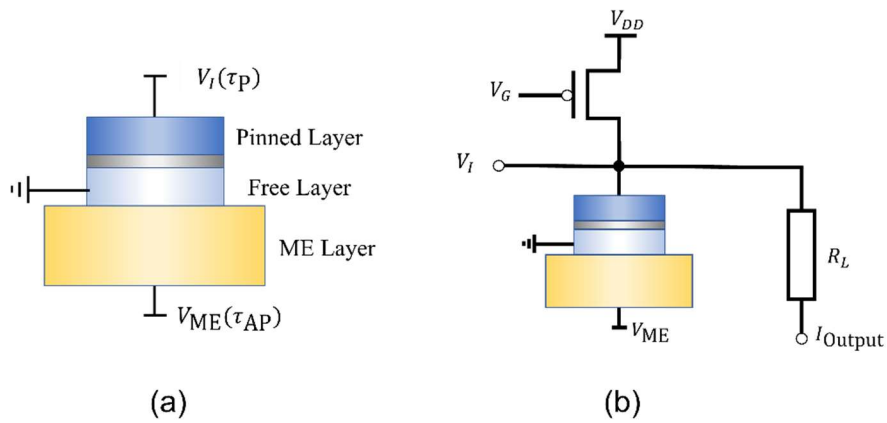

Figure **3-3**: The proposed ME-MTJ device and detect circuit are shown. (a) Concept of ME-MTJ device, driven by two independent inputs - (1) Voltage, V_{ME}, applied across the ME-oxide modulates lifetime τ_{AP}, (2) Voltage, V_I, applied across the MTJ modulates τ_P. (b) Circuit design to detect spikes. I_{Output} indicates the MTJ state.

Table **3-1**: Device parameters for LLG simulation are tabulated.

Parameter	Value
TMR	200%
Free-layer width, W_{MTJ} [125]	17nm
Free-layer length, L_{MTJ} [125]	42.5nm
Free-layer thickness, t_{MTJ} [125]	0.8nm
Saturation magnetization, M_S [125]	750kA/m
Gilbert-damping factor, α [125]	0.0122
Temperature, T	300K
ME constant, α_{ME} [157]	5×10^{-9}s/m
ME-layer thickness, t_{ME} [157]	5nm

The device state can be detected by a circuit shown in Figure **3-3** (b). The transistor working in saturation region provides a constant current, I_{Total}. V_I is the input voltage applied to the MTJ. The MTJ resistance modulates the current flowing through the MTJ, I_{MTJ}, leading to the control of current flowing through the load resistance R_L. As a result, the output current, $I_{Output} = I_{Total} - I_{MTJ}$, will be an indicator of the MTJ state.

The main distinguishing factors in this neuromimetic ME-MTJ design are as follows: (i) ME-MTJs have typically been considered to be switched by applying a voltage across the ME oxide

[158]. Here, the design in this work uses two independent inputs (V_{ME} and V_I). While the voltage applied across the ME oxide will produce the effect of an applied magnetic field (thereby modulating τ_{AP}), the external input current will be used to control τ_{P}. Independent control of these two parameters will enable the users to implement a stochastic nanoelectronic spiking neuron functionality that inherently performs temporal domain encoding of information, as explained previously. (ii) Most of the work on ME-MTJs is catered for usage of these devices in logic and memory applications [152], [156], [157], [159], [160]. This proposal involves utilizing ME for enabling neuromorphic applications, and in particular, for temporal-encoding of stochastic SNNs.

2.3. Independent Control of τ_{P} and τ_{AP}

Next, the device operation is characterized by varying the two external input voltages and measuring the average device lifetimes. It is worth noting here that from a system development perspective, the neurons will be interfaced with synaptic devices. Hence, achieving truly independent control of τ_{P} and τ_{AP} over a wide operating range of V_I and V_{ME} is crucial. We define a set of k factors (given in Equation **3.3**) to evaluate the impact of the two external bias signals on τ.

$$k_{\mathrm{AP(P)},\mathrm{ME}(I)} = \frac{\partial \tau_{\mathrm{AP(P)}}}{\partial V_{\mathrm{ME}(I)}} \tag{3.3}$$

The value of k depicts the amount of change in τ induced by a unit change in one of the biases (V_{ME} or V_I) with the other bias fixed. The total change of τ is expressed as

$$\Delta\tau_{\mathrm{AP(P)}} = k_{\mathrm{AP(P)},\mathrm{ME}}\Delta V_{\mathrm{ME}} + k_{\mathrm{AP(P)},I}\Delta V_I \tag{3.4}$$

To realize the independent control, the variation of s in one state should be dominated by only one of the biases, leading to the conditions:

$$\left|\frac{k_{\mathrm{AP,ME}}}{k_{\mathrm{AP},I}}\right| \gg 1, \qquad \left|\frac{k_{\mathrm{P},I}}{k_{\mathrm{P,ME}}}\right| \gg 1 \tag{3.5}$$

Equation **3.5** implies that τ_{AP} is dominated by V_{ME} and τ_{P} is dominated by V_I. It should be noted that the ratios in Equation **3.5** are related to the slope of contour lines of τ_{AP} and τ_{P} for varying V_{ME} and V_I. Figure **3-4** depicts the contour map of τ in the two states. The contour lines in P state have a small slope (note that $\left|\frac{k_{\mathrm{P},I}}{k_{\mathrm{P,ME}}}\right|$ is the reverse of the slope), indicating that the spin current has a dominant control on τ_{P}, while the slope of contour lines in AP state is large, which implies that ME is the leading control factor.

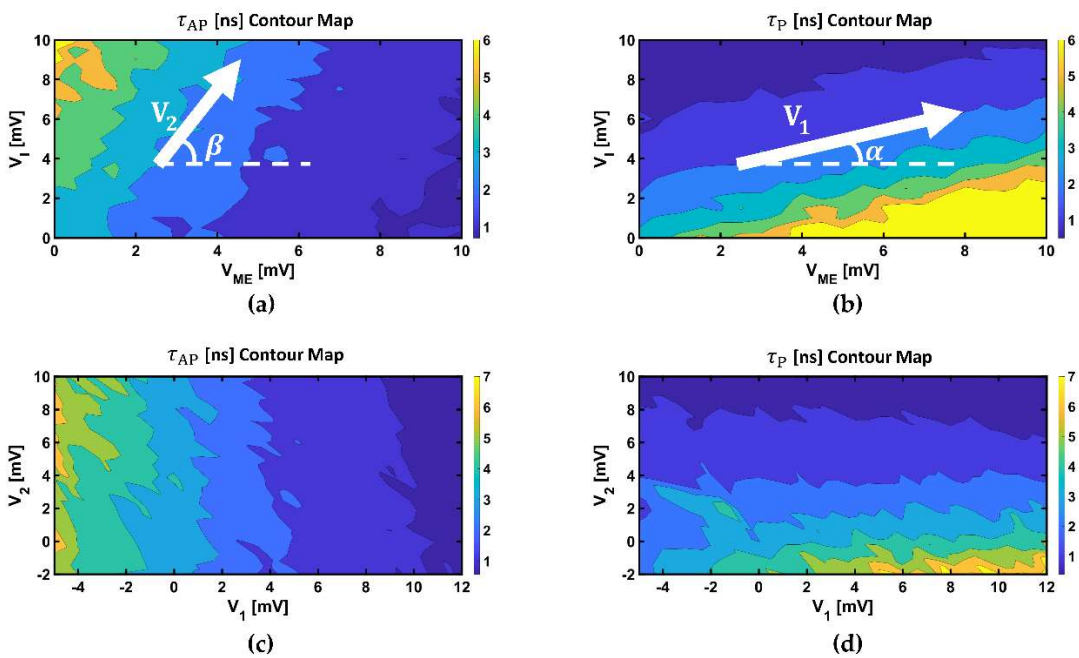

Figure **3-4**: Contour maps of τ- V relations are shown. (a) and (b) Contour map of τ_{P} and τ_{AP} vs V_I and V_{ME} and (c) and (d) contour map of τ_{P} and τ_{AP} vs V_1 and V_2.

However, in order to achieve truly independent control, the change of the dominant bias in one state should not make a prominent change on τ of the other state. For example, since V_{ME} dominates τ_{AP}, $\Delta\tau_{\mathrm{AP}}$ induced by ΔV_{ME} need to be much larger than $\Delta\tau_{\mathrm{P}}$ induced simultaneously. As a result, another condition for the independent control is

$$\left|\frac{k_{\mathrm{AP,ME}}}{k_{\mathrm{P,ME}}}\right| \gg 1, \qquad \left|\frac{k_{\mathrm{P},I}}{k_{\mathrm{AP},I}}\right| \gg 1 \tag{3.6}$$

Equation **3.6** indicates that V_{ME} has a much larger impact on τ_{AP} than on τ_{P}, and V_I has a larger impact on τ_{P} than on τ_{AP}. According to the definition of k, larger k values result in more rapid change of τ, leading to denser contour lines in the contour map. From this perspective, Equation **3.6** states that our device operating region has to restricted in an area where the contour lines should be much denser (the spacing between adjacent contour lines is smaller) in the AP state going along the V_{ME} axis, and concurrently the contour lines are much denser in the P state going along the V_I axis. As is shown in Figure **3-4**, in the map of AP state, contour lines are denser in the top-left while in the P state, the denser area is in the bottom-right portion of the plot. This opposite nature of k factor variation severely limits the operating region of the device toward the middle diagonal region of the plot to compromise between the restrictions imposed by Equation **3.6**.

Interestingly, it is observed that although the contour lines are not strictly horizontal or vertical, the slope of the lines is approximately constant throughout the range. Hence, to remove the limitation and expand the device operating region, one can introduce a set of new basis signals in the direction of the contour lines in Figure **3-4**, denoted as $< V_1,\ V_2 >$. No unwanted $\Delta\tau$ will be induced as s is fixed along the contour lines. The new basis signals can be mapped from the device inputs $< V_{\mathrm{ME}}\ ,\ V_I >$ through Equation **3.7**:

$$\begin{pmatrix} V_1 \\ V_2 \end{pmatrix} = \begin{pmatrix} \cos\alpha & \cos\beta \\ \sin\alpha & \sin\beta \end{pmatrix}^{-1} \begin{pmatrix} V_{\mathrm{ME}} \\ V_I \end{pmatrix} \tag{3.7}$$

where α, β are shown in Figure **3-4** (a) and (b). The contour map based on the new basis $< V_1,\ V_2 >$ is plotted in Figure **3-4** (c) and (d). The neuron functionality can now be conceptualized as being driven by external inputs V_1 and V_2. The actual inputs to the device, V_{ME} and V_I, are a linear combination of the two external inputs, V_1 and V_2, which can be easily implemented by voltage divider circuits. From a network perspective, these signals would be determined by current flowing though synaptic devices [120]. It is worth noting that such a simple transformation is made possible

due to the constant slope of contour lines throughout the plot. As observed in Figure **3-4** (c) and (d), the contour lines are approximately horizontal/vertical, thereby realizing independent control of device lifetimes over the entire operating region.

3. Building SNNs with ME-MTJs

3.1. Applying ME-MTJs as Spiking Neurons

Given such a continuously switching device is available where the precise temporal dynamics can be controlled, the high level question to be addressed next is: *Can we map the core device characteristics to compute primitives required in a functional stochastic SNN operation with temporal information encoding?* Let us consider a particular network where all the neurons are driven by the same voltage corresponding to input V_1 such that the average device lifetime in the AP state equals the duration of a "timestep" in the system. Note that the duration of timestep" will be determined by circuit and architecture level constraints for simulating the SNN. If we interpret the device AP state as the "spike" of the neuron, then the average time to fire for that neuron will be given by τ_{P}, which can be controlled by the external neuron input V_2. The spiking framework is illustrated in Figure **3-5**. For an SNN inferring data based on temporal encoding, this time to fire will dictate the winning neuron. The neuron which fires earliest will be interpreted as the winning class and is based on time-to-first-spike encoding. Note that the SNN can be turned off after the first spike, thereby resulting in significant sparsity and latency benefits. Such a fine-grained control of time to fire is not possible in case of stochastic magnetic devices driven by only a single external input signal since both the device lifetimes will be modulated together. It is also worth mentioning here that while our proposal is based on the ME-MTJ device, the formulation can be easily extended to experimentally demonstrated stochastic devices operating under the influence of external spin current and magnetic field [148], [149]. In order to train the network, let us assume that we set the

winning class neuron to fire at timestep t_1 while the other neurons target a firing time t_2. In order to infer with sufficient confidence margin, $\Delta t = t_2 - t_1$ should be reasonably high. Note that Δt, t_1 and t_2 are hyperparameters for our algorithm and user specified. In this work, we used a value of $t_1 = 1\text{ns}$ and $t_1 = 300\text{ns}$.

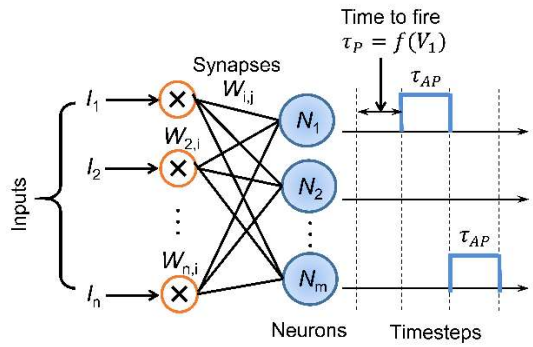

Figure **3-5**: Supervised algorithm for stochastic SNNs with temporal information encoding where neuron input, V_2, controls the time to fire is shown.

3.2. Algorithm Formulation

Fully connected neural network architectures with stochastic temporal encoding were trained on the MNIST dataset [161] based on algorithmic formulations described next. Since the real-time device lifetimes follow an exponential distribution in the low current regime [162], KL divergence is utilized to model the loss function. Assuming the target average device lifetime in the P state to be λ and the expected device lifetime due to the external input to be z, the KL divergence between the expected and target spike probability distributions is given by Equation **3.8**:

$$L = \sum_{a \in A} \frac{1}{\lambda} e^{-\frac{a}{\lambda}} \log\left(\frac{z}{\lambda} e^{a\left(\frac{1}{z} - \frac{1}{\lambda}\right)}\right) \quad \textbf{(3.8)}$$

where, A is the probability space. From a network perspective, each neuron receives the weighted summation of synaptic inputs ($\sum_i w_i I_i$) as the input voltage V_2 (see Figure **3-5**). Note that the output current in the spike detection circuit (see Figure **3-3**(b)) can be used to charge a capacitor till the input neuron device spikes, thereby converting the timing information to an analog voltage input for the next layer. Assuming the intrinsic device function mapping from the synaptic dot product to the average P state device lifetime to be $g(.)$ (which can be formulated by the exponential variation shown in Figure **3-6**),

$$z = g\left(\sum_i w_i I_i\right) = g(V_2) \tag{3.9}$$

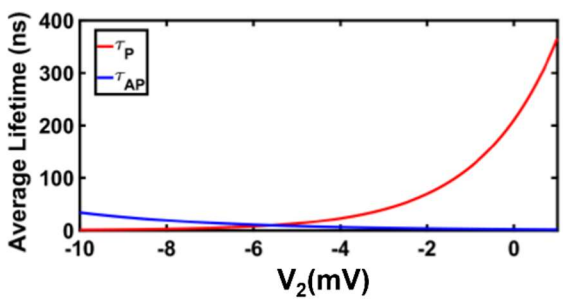

Figure **3-6**: Variation of the average device lifetimes as a function of the neuron input, V_2, which is equivalent to the weighted summation of synaptic inputs $\sum_i w_i I_i$. Device lifetime, τ_{AP}, remains roughly constant over the input voltage range while the exponential variation of τ_{P} with V_2 is considered to be the activation function of the neuron ($g(.)$ in Equation **3.9**).

It is worth mentioning here that the output z represents the average value of P-state device lifetime under the influence of V_2, although the real-time characteristics follow an exponential distribution [162]. The operating voltage range of the device is also chosen properly (Figure **3-6**) such that the change in τ_{P} is much larger than τ_{AP} (assumed constant equal to spike duration in the algorithm formulation) within this working range.

Using gradient descent, the weights of the network can be learnt through the following relations,

$$w = w - \alpha\left(\frac{\partial L}{\partial w}\right); \frac{\partial L}{\partial w} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial w} \tag{3.10}$$

where, α is the learning rate. The term $\frac{\partial z}{\partial w}$ can be obtained using Equation **3.9**, while the term $\frac{\partial L}{\partial z}$ can be derived from Equation **3.8** by algebraic manipulations as,

$$\frac{\partial L}{\partial z} = \sum_{a} \frac{1}{z\lambda} e^{-\frac{a}{\lambda}} - \sum_{a} \frac{a}{T^2\lambda} e^{-\frac{a}{\lambda}} \tag{3.11}$$

3.3. Network Performance

The activation function of the neurons, $g(.)$, given by the relationship between the P state lifetime, τ_{P}, and the applied voltage, V_2, is obtained from stochastic-LLG simulations of the superparamagnetic MTJ device with a $2k_{\mathrm{B}}T$ barrier height. A hybrid device-algorithm co-simulation framework calibrated to experimental measurements was used to evaluate the performance of the network. The 784×10 network therefore consisted of LLG simulations of 10 MTJ devices while the deeper $784 \times 400 \times 10$ network consisted of 400 MTJs in the hidden layer and 10 devices in the output layer. Figure **3-7** gives an example of prediction of 784×10 network based on the precise time to spike. Neuron 6 fires the first spike at $t = 23$ns while all neurons keep silent. In this case, the input figure is predicted to be digit 6.

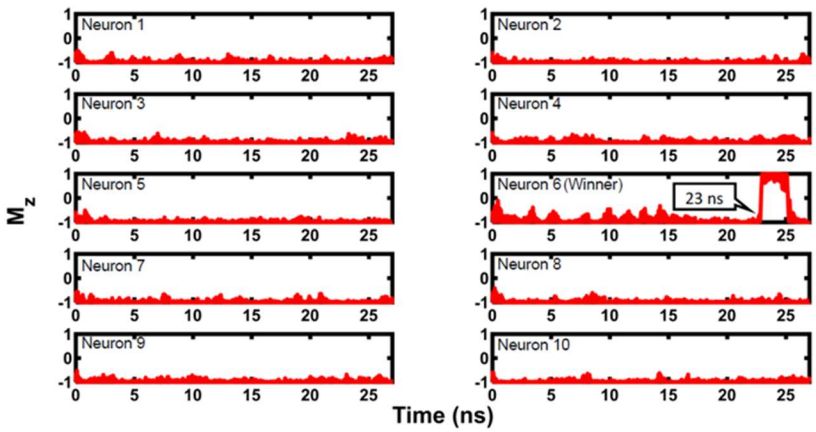

Figure **3-7**: The response of 10 neurons in the output layer for one input figure is shown.

A test accuracy of 90.88% was observed for a network architecture of 784×10 neurons. However, since the realtime device operation is stochastic with exponential lifetime characteristics, there might be image instances which are inferred incorrectly if the decision is solely based on the first spike, as is indicated by Figure **3-8**, where the trained τ_P of winning neuron and failing neuron(s) are close. In this case, even though the network is correctly trained, failing neuron may spike prior to winning neuron due to the stochastic firing of spikes, which leads to an error prediction.

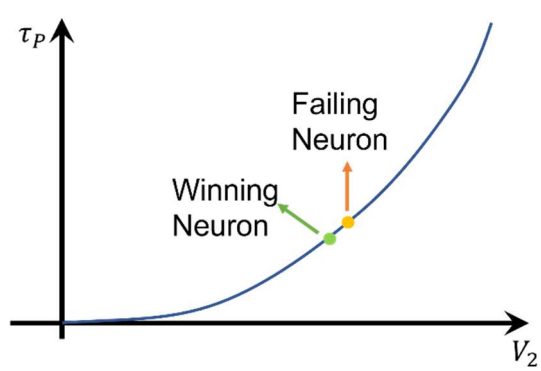

Figure **3-8**: Cause for stochasticity-related error is shown.

In that case, the robustness of the decision and the classification accuracy improves significantly if the inference process is based on the sum of multiple inter-spike intervals. As demonstrated in Figure **3-9**, the accuracy of the hardware network approaches the ideal baseline software accuracy with only a 2/3-spike confidence for the winning neuron, thereby resulting in a highly sparse firing behavior of the neurons due to temporal information encoding.

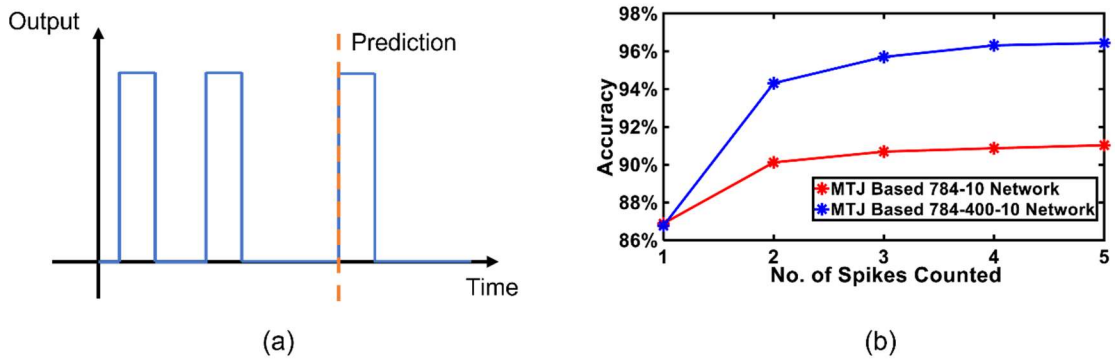

Figure **3-9**: Prediction process and result based on multiple spikes are shown. (a) The prediction is based on multiple inter-spike intervals of each neuron (interval of 3 spikes in the figure). (b)Prediction based on multiple spikes show improved accuracy. Accuracy close to ideal baseline can be achieved with only 2 or 3 spikes.

Similar observations were achieved when the network was scaled to a 3-layer architecture with $784 \times 400 \times 10$ neurons. The network had a test accuracy of 97.41%, at par with iso-architecture standard deterministic networks (a conventional non-spiking network with rectified linear neuron units with 400 hidden layer neurons was observed to have a test accuracy of 97.03% after 20 epochs of training). Interestingly, even for this deeper network, the testing accuracy achieved near-maximum values with only 2 to 3 spikes being considered for both the hidden and output layers. This is a significant improvement over rate encoding methods and substantiates the advantages of spiking sparsity enabled by

temporal encoding. In rate encoding, each layer triggers the next layer by the average firing rate and therefore the spiking activity increases exponentially with layer depth (for instance, the maximum firing activity per neuron can range between 5 to 10 in deep rate encoded SNN architectures like VGG and ResNet [92]). In contrast for temporal encoding, since information transmission from one layer to another does not depend on average firing rate but rather on the time of firing, there is no dependency of spiking activity on network scaling. While the stochasticity causes the number of spikes for inference to slightly increase above 1 to maintain minimal accuracy drop, it enables the usage of binary state-compressed scaled neuron devices to encode multi-bit information, instead of complex device structures exhibiting spin textures like domain walls, skyrmions, among others [70]. In order to perform a benchmarking analysis, we compared the sparsity levels in our network against an iso-accuracy rate-encoded stochastic MTJ network (implemented according to the proposal outlined in Ref. [60]). We observed 1.6 × reduction in spiking sparsity for the hidden layer and 3.77 × reduction in spiking sparsity for the output layer in the 784 × 400 × 10 neuron network. Scaling to deeper architectures is expected to improve the sparsity and latency benefits of such architectures along with providing accuracies at par with other implementations [46], [138].

4. Discussion and Outlook

This work presents a unique perspective of designing efficient stochastic neuromorphic systems with temporal information encoding driven by an interdisciplinary perspective from devices to brain-inspired algorithm development. The work provides algorithmic formulations to leverage the stochastic temporal device characteristics of superparamagnetic devices and provides proof-of-concept demonstrations through extensive simulations. Such an end-to-end co-design

effort to leverage unique properties of neuromorphic computing is an ideal fit for application drivers characterized by temporal information (for instance, sparse data collected by event-driven sensors [163], [164], among others).

Chapter 4

All-Spin Bayesian Neural Networks

Probabilistic machine learning enabled by the Bayesian formulation has recently gained significant attention in the domain of automated reasoning and decision making. While impressive strides have been recently made to scale up the performance of deep Bayesian neural networks, they have been primarily standalone software efforts without any regard to the underlying hardware implementation. In this chapter, an "all-spin" Bayesian neural network is proposed where the underlying spintronic hardware provides a better match to the Bayesian computing models. To the best of our knowledge, this is the first exploration of a Bayesian neural hardware accelerator enabled by emerging post-CMOS technologies. An experimentally calibrated device-circuit-algorithm simulation framework is developed and $24\times$ reduction in energy consumption against an iso-network CMOS baseline implementation is demonstrated.

1. Motivation

As is mentioned in previous chapters, probabilistic inference is at the core of decision-making in the brain, which can be realized by BNNs due to the capabilities of making predictions based on Bayes' theorem where probability distributions can be modeled by Gaussian distributions [165]. On the other hand, the development of spintronic devices shows a promising pathway towards a non-von Neumann hardware system, which avoids the bottleneck between memory and computing unit attributed to the in-memory computing framework and can be a hardware solution for BNN acceleration [70], [120], [141], [166], [167]. Moreover, scaled nanomagnetic devices operating at room temperature are characterized by thermal noise, resulting in probabilistic switching. These factors lead to the idea of leveraging the inherent device stochasticity of spintronic devices to generate the samples from Gaussian probability distributions by drawing insights from

a statistical central limit theorem. Furthermore, the work also elaborates on a cohesive design of a spintronic Bayesian processor that leverages the benefits of spin-based Gaussian random number generators (RNGs) and spintronic "in-memory" crossbar architectures to realize high-performance, energy-efficient hardware platforms. It is believed that the drastic reductions in circuit complexity (single devices emulating synaptic scaling operations, crossbar architectures implementing "in-memory" dot-product computing kernels and leveraging device stochasticity to sample from probability distributions), and low operating voltages of spintronic devices make them a promising path toward the realization of probabilistic machine learning enabled by the Bayesian formulation.

2. Hardware Design Space Concerns in Bayesian Neural Networks

As is introduced in chapter 2, Bayesian neural networks consider the weights of the network, $\boldsymbol{W}$, to be latent variables characterized by a probability distribution (shown in Figure **2-6**). During the inference process, weight values are sampled from a posterior distribution $P(\boldsymbol{W}|D)$. The sampled weight values join the dot-product calculation according to Equation **2.12**. In this way, the uncertainty in weight values results in probabilistic inference. However, posterior distribution given by Equation **2.8** according to Bayes' rule is intractable. For this reason, by adopting variational inference method, $P(\boldsymbol{W}|D)$ is approximated by a Gaussian distribution $q(\boldsymbol{W}, \theta)$. To summarize, the calculation process gives rise to the main hardware design space concerns in BNNs categorized as follows.

2.1. Gaussian Random Number Generation

Central to the entire framework, both in the learning as well as the inference process, is the random number generation corresponding to the synaptic weights. Given the current large model sizes characterized by over a million synapses, coupled with the fact that random draws need to

perform multiple times for each synaptic weight, RNG circuits would contribute significantly to the total area and power consumption of the hardware. Furthermore, the random numbers need to be sampled from a Gaussian distribution, thereby increasing the complexity of the circuit. The hardware costs for CMOS implementations of such Gaussian RNGs will be discussed in the following sections along with their limitations, followed by the proposal of nanomagnetic RNGs that can serve as the basic building blocks of such Bayesian neural networks.

2.2. Dot-Product Operation Between Inputs and Sampled Synaptic Weights

A common aspect of any standard deep-learning framework is the fact that forward propagation of information through the network involves a significant amount of memory-intensive operations. The dot-product operation between the synaptic weights and the inputs for inference involves the compute energy along with memory access and memory leakage components. For large-scale problems and correspondingly largescale models, CMOS memory access and memory leakage can be almost $\sim 50\%$ of the total energy consumption profile [168].

The situation is further worsened in a Bayesian deep network since each synaptic weight is characterized by two parameters (mean and variance of the probability distribution), thereby requiring double memory storage. However, the dot-product operation does not occur directly between the inputs and these parameters. In fact, for each inference operation, the synaptic weights (typically assumed constant during inference for non-probabilistic networks and implemented by memory elements in hardware) are repeatedly updated depending on sampled values from the Gaussian probability distribution. Hence, the direct utilization of crossbar-based "in-memory" computing platforms enabled by nonvolatile memory technologies (discussed in detail later) for alleviating the memory access and memory fetch bottlenecks is not possible and therefore requires a significant rethinking.

In the following sections, the work sequentially expand on each of these points and propose a spin-based neural processor that merges the deterministic and stochastic devices as a potential pathway to enable Bayesian deep learning that can be orders of magnitude more efficient in contrast to state-of-the-art CMOS implementations.

3. Spintronic Device Design

3.1. Magnetic Tunnel Junction—True Random Number Generator Design

The basic device structure under consideration is the MTJ introduced in chapter 2. Let us now consider the switching of the scaled magnet from one state to another by the application of an external current. The switching process is inherently stochastic at nonzero temperatures due to the thermal noise, as is shown in Figure **2-10** [126]. In the presence of an external current, the probability of switching from one state to the other is modulated depending on the magnitude and duration of the current. True RNG (TRNG) can be designed using such a device by biasing the magnet at the "write" current corresponding to a switching probability of 50%. Note that CMOS-based TRNGs suffer from high-energy consumption and circuit design complexity [169]. Proposals and experimental demonstrations of MTJ-based TRNG have been shown [144]. MTJ-based TRNGs are characterized by low area footprint and compatibility with CMOS technology.

In this work, a spin–orbit coupling-enabled device structure is considered (see Figure **4-1**). It consists of the MTJ stack lying on top of a heavy-metal (HM) underlayer. The device "read" is performed through the MTJ stack between terminals T1 and T3. However, the device "write" is performed by passing current through the HM underlayer between terminals T2 and T3. Input current flowing through the HM results in spin injection at the interface of the magnet and HM due to spin-Hall effect (SHE) [131] and thereby causes switching of the MTJ FL [170]. The device has the following advantages.

1. The decoupling of “write” and “read” current paths is advantageous from the perspective of peripheral circuit design to avoid “read”– “write” conflicts since the associated circuits can be optimized independently.
2. Such devices offer 1–2 orders of magnitude energy efficiency in comparison to standard spin-transfer torque MRAMs. This is due to the fact that in such spin–orbit coupling-based systems, every incoming electron in the “write” current path repeatedly scatters at the interface of the magnet and HM and transfers multiple units of spin angular momentum to the ferromagnet lying on top.

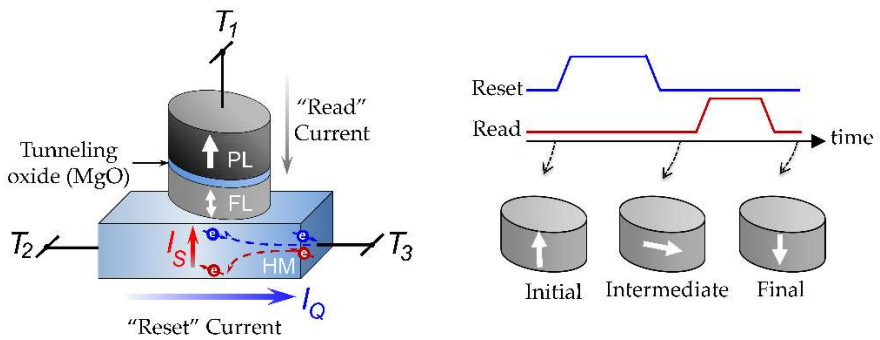

Figure **4-1**: TRNG device structure is shown. Reset current (I_Q) flowing through the HM results in in-plane spin current (I_S) injection for the MTJ FL. After switching to the in-plane metastable position, the magnet relaxes to either of the two stable states with 50% probability.

Usage of SHE-based switching enables us to use an alternative TRNG design [171], [172] that has the potential to produce high-quality random numbers in the presence of process, voltage, and temperature (PVT) variations. In the earlier scenario of a standard MTJ, device-to-device variations can result in deviations of the bias current required for 50% switching probability, thereby degrading the quality of the random number generation process. The scheme is shown in Figure **4-1**, where a magnet with perpendicular magnetic anisotropy (PMA) lies on top of the HM. The device operation is divided into three stages. During an initial “Reset” stage, a current flowing

through the HM results in in-plane spin injection in the magnet and orients it along the hard axis for a sufficient magnitude of the "reset" current. The magnet is then allowed to relax to either of the two stable states in the presence of thermal noise—the switching probability being 50% since the hard axis is a metastable orientation point for the magnet. In this case, device-to-device variations only cause change in the critical current required for biasing the magnet close to the metastable orientation and does not skew the probability distribution to a particular direction (as in the standard MTJ case). Hence, by maintaining a worst case critical value of the HM "reset" current, the quality of the random number generation process can be preserved even in the presence of PVT variations. Furthermore, the "reset" current does not flow through the tunneling oxide layer (unlike the standard MTJ case), and therefore, the reliability of the oxide layer is not a concern in this scenario [171], [172]. Note that our device operation is validated by recent experiments of holding the magnet to its metastable hard-axis orientation for performing Bennett clocking in the context of nanomagnetic logic [173]. SHE-based energy-efficient switching also results in the reduction of the energy consumption involved in the random number generation process.

The probabilistic switching characteristics of the MTJ can be analyzed by LLG equation (given in Equation **2.11**) with additional term to account for the spin–orbit torque generated by the SHE at the ferromagnet–HM interface [124]. The spin current from HM layer can be modeled as

$$\boldsymbol{I}_S = \theta_{SH}\left(\frac{A_{\mathrm{MTJ}}}{A_{\mathrm{HM}}}\right)\mathbf{I}_q \tag{4.1}$$

In Equation **4.1**, A_{MTJ} and A_{HM} are the MTJ and HM cross-sectional area. θ_{SH} is the spin-Hall angle, and $\mathbf{I}_q$ is the charge current flowing through the HM underlayer.

The device parameters are mentioned in Table **4-1**. Considering a worst case "reset" current of $140\mu\mathrm{A}$ for a duration of $1\mathrm{ns}$, the energy consumption involved in using a $20k_{\mathrm{B}}T$ barrier magnet (calibrated to experimental measurements reported in [174]) as a TRNG is $\sim 57\mathrm{fJ/bit}$ (I^2Rt energy consumption) [171], which is almost $2\times$ lower than the standard MTJ-based TRNG.

Table **4-1**: MTJ Device Simulation Parameters are tabulated.

Parameter	Value
Free-layer width	40nm
Heavy-metal thickness	2nm
Saturation magnetization, M_S [174]	1000kA/m
Spin-Hall angle, θ_{SH} [174]	0.3
Energy barrier, E_B	$20k_BT$
Temperature, T	300K

3.2. Domain-Wall Motion-Based Magnetic Devices— Multilevel Non-Volatile Memory Design

The DW motion-based MTJs (referred to as DW-MTJs) introduced in chapter 2 can be utilized as non-volatile memory devices since the device conductance is non-volatile and can be easily modified by applying charge current through HM layer, as is indicated in Figure **2-10**. It is worth noting here that the device structure in Figure **2-9** can be used as a neuron by interfacing with a reference MTJ [see Figure **4-2**] [120]. The resistive divider can drive a CMOS transistor where the output drive current would be a linear function of the input current flowing through the HM layer of the device, thereby mimicking the functionality of a saturated linear functionality by ensuring that the transistor operates in the saturation regime [120]. In this way, both memory and neurons are based on MTJs. The simulation parameters, provided in Table **4-2**, were used for the rest of this text for DW-MTJ unless otherwise stated. The parameters were obtained magnetometric measurements of CoFe–Pt nanostrips [121].

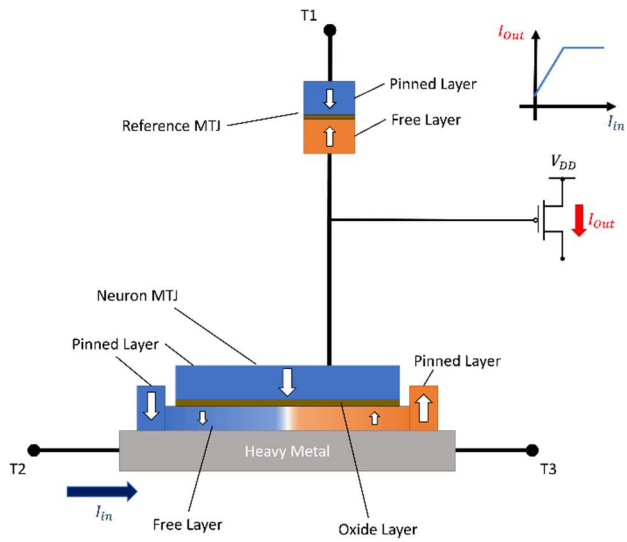

Figure **4-2**: DW-MTJs can be used as a neuron by interfacing with a reference MTJ. The current provided by the output transistor, I_{out}, is a saturated linear function of the input current, I_{in}.

Table **4-2:** DW-MTJ Device Simulation Parameters are tabulated.

Parameter	Value
Ferromagnetic thickness	0.6nm
Grid size	$4 \times 1 \times 0.6\text{nm}^3$
Heavy-metal thickness	3nm
Domain wall width	7.6nm
Saturation magnetization, M_S [121]	700kA/m
Spin-Hall angle, θ_{SH} [121]	0.07
Gilbert damping factor, α [121]	0.3
Exchange correlation constant, A [121]	$1 \times 10^{-11}\text{J/m}$
Perpendicular magnetic anisotropy, K_{u2} [121]	$4.8 \times 10^5\text{J/m}^3$
Effective DMI constant, D [121]	$-1.2 \times 10^{-3}\text{J/m}^2$

4. All-Spin Bayesian Neural Networks

4.1. Spin-Based Gaussian Random Number Generator

Gaussian random number generation task is a hardware-expensive process. CMOS-based designs for Gaussian RNGs would usually require a large number of registers, linear feedback circuits, and so on. For instance, a recent work for a CMOS-based Gaussian RNG implementation reports 1780 registers and $528.69 - \mathrm{mW}$ power consumption for a 64-parallel Gaussian RNG task [175].

Let us now discuss the proposal for spin-based Gaussian RNG. In last section, we discussed the design of spintronic TRNG. An array of TRNGs can be used for sampling from a uniform probability distribution. Note that each spin device can be considered to produce a sample from a Bernoulli distribution with a probability of 0.5. However, reading a particular row of the array provides a sample from a discrete uniform distribution. In order to generate a Gaussian probability distribution from a uniform one, we draw inspiration from the statistical central limit theorem, as discussed in Box 1. The key result of the central limit theorem that we utilize is that the sum of a large number of independent and identically distributed (i.i.d) random variables is approximately normal.

Box 1: Central Limit Theorem

Let $\{X_1, X_2, \dots, X_n\}$ be a random sample of n i.i.d random variables drawn from a distribution (which may not be normal) of mean μ and variance σ^2. Then, the probability density function of the sample average $S_n = (X_1 + X_2 + \cdots + X_n)/n$ approaches a normal distribution with mean μ and variance σ^2/n as n increases.

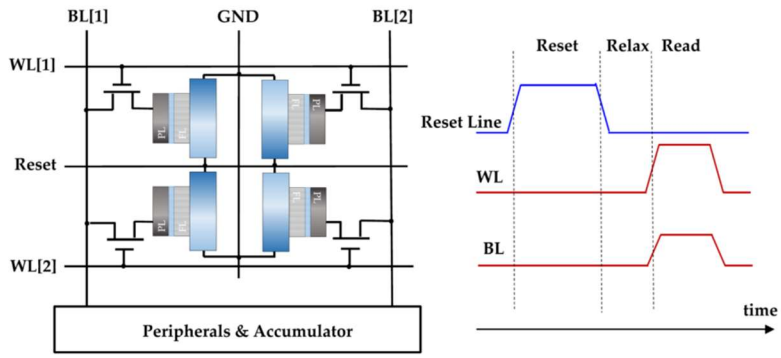

Figure **4-3**: Outline of a 2×2 array utilizing spin-based devices interfaced with an accumulator to implement a Gaussian RNG.

The proposed design is shown in Figure **4-3**, which depicts a possible array implementation [171] of our spin-based TRNGs. Each spin device is interfaced with an access transistor. Rows sharing a reset line can be driven simultaneously. Hence, random numbers can be generated in the entire array in parallel. The timing diagram is shown in Figure **4-3**. Each row can be read by asserting a particular wordline (WL) and sensing the bitline (BL) voltage. For an $m \times n$ array, each row read produces an n-bit number generated from a uniform probability distribution. By interfacing the array with an accumulator that averages all the generated random numbers, the array is able to produce random numbers drawn from a normal distribution. Note that the hardware overhead for this process would be high for applications that require precise sampling from Gaussian distributions since the convergence takes place only for infinite samples. However, for machine-learning workloads considered herein, the performance of such platforms is usually resilient to approximations in the underlying computations. For instance, Figure **4-4** shows that even with an 8-bit representation and three random variables drawn from the uniform probability distribution, an approximate Gaussian distribution can be achieved. While Gaussian probability distributions are primarily used in such algorithms, non-Gaussian weight distributions can also be designed by using the Gaussian function as a basis. Note that while Box 1 discussions are equally

valid for a CMOS-based TRNG, it will be an order of magnitude more area and power consuming than our proposed spin-based TRNG, as explained in previous section.

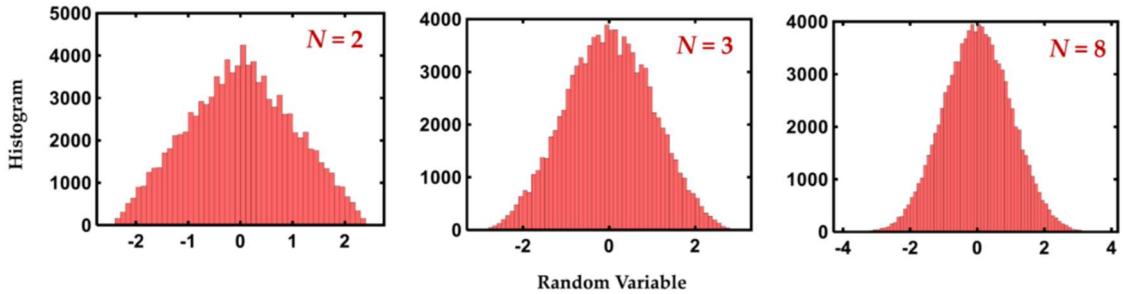

Figure **4-4**: The probability distributions of random numbers generated from such an array are shown in the extreme right by using a sum of N random variables (rows of the array). 8-bit representation and 100000 samples are used to plot the distribution.

4.2. Dot-Product Operation Between Inputs and Sampled Synaptic Weights

DW-MTJs can be implemented in crossbar arrays introduced in chapter 2 (see Figure **2-12**). Dot-products are calculated according to Kirchhoff's law given in Equation **2.12** and the resulting current output is sent to spintronic neurons. Consecutive "write" and "read" cycles of the spin neurons will implement multiple iterations of the Bayesian network. The analog output current provided by the spin neuron is then converted to a digital format using the analog-to-digital converters (ADCs). The digital outputs can be latched to provide the inputs for the fan-out crossbar arrays. The energy efficiency of the system stems mainly from two factors as follows.

1. The input write resistance of the spintronic neurons is low (being magnetometallic devices) and it inherently requires very low currents for switching. This enables the crossbar arrays of spintronic synapses to be operated at low terminal voltages (typically 100 mV). Furthermore, spintronic neurons are inherently current-driven and thereby do not require costly current to voltage converters, in contrast to CMOS and other emerging technology-

based (resistive random access memory and phase-change memory, among others) implementations [176].

2. Since spin devices are inherently nonvolatile technologies, the ability to perform the costly multiply-accumulate operations in the memory array itself enables us to address the issues of von-Neumann bottleneck.

However, in the context of Bayesian deep networks, even for the inference stage, the synaptic weights are not constant but are updated depending on sampled values from a Gaussian distribution. Assuming that we are able to generate samples from a normal distribution by using the device-circuit primitives proposed earlier, the computations in a Bayesian network can be partitioned in an appropriate fashion such that the benefits of spin-based "in-memory" computing can be still utilized. This is explained in Box 2.

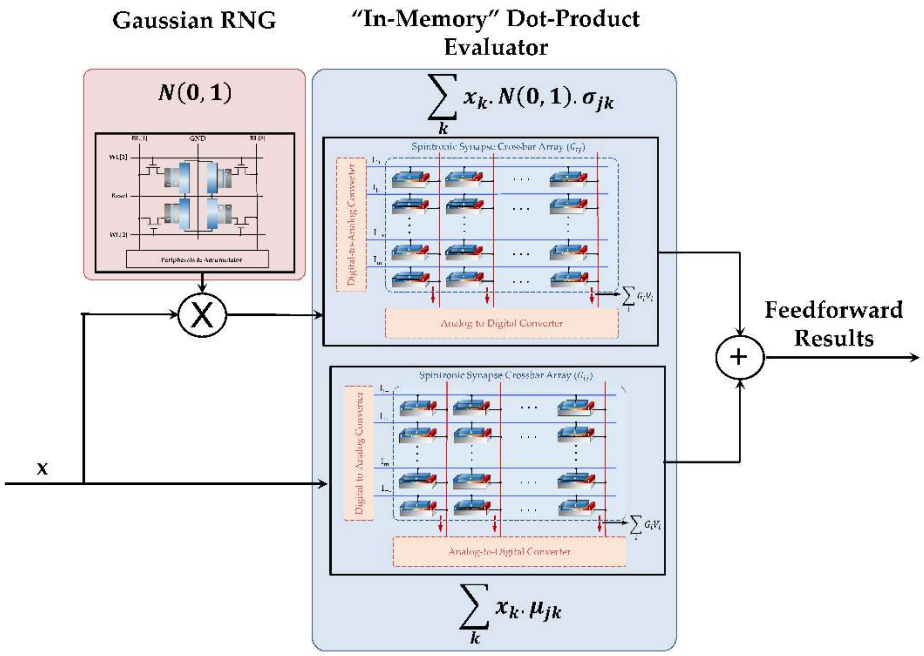

Figure **4-5**: All-spin Bayesian neural network implementation. The two crossbar arrays behave as "in-memory" computing kernels, whereas the RNG unit provides the sampling operation from the Gaussian RNGs.

Box 2: Computations Involved in Inference Operation

Once all the posterior distributions are learned (μ and σ parameters of the weight distributions), the network output corresponding to input, $\boldsymbol{x}$, should be obtained by averaging the outputs obtained by sampling from the posterior distribution of the weights, $\boldsymbol{W}$ [175]. The output of the network y is therefore given by Equation **4.2**.

$$y = \mathbb{E}_{P(\boldsymbol{W}|D)}[f(\boldsymbol{x},\boldsymbol{W})] \approx \mathbb{E}_{q(\boldsymbol{W},\theta)}[f(\boldsymbol{x},\boldsymbol{W})] \approx \frac{1}{S}\sum_{i=1}^{S} f\left(\boldsymbol{x},\boldsymbol{W}^i\right) \tag{4.2}$$

$f(\boldsymbol{x},\boldsymbol{W})$ is the network mapping for input $\boldsymbol{x}$ and weights, $\boldsymbol{W}$. Using the variational inference method introduced in chapter 2, we approximate the weight distribution by Gaussian functions. The approximation is performed over S independent Monte Carlo samples drawn from the Gaussian distribution $q(\boldsymbol{W},\theta)$.

Considering just a single layer and neglecting the neural transfer function, $f\left(\boldsymbol{x},\boldsymbol{W}^i\right)$ for the jth neuron can be decomposed into

$$\begin{aligned} f\left(\boldsymbol{x},\boldsymbol{W}_j^i\right) &= \sum_k x_k \cdot N\left(\mu_{jk},\sigma_{jk}\right) \\ &= \sum_k x_k \cdot \left(\mu_{jk} + \sigma_{jk} \cdot N(0,1)\right) \\ &= \sum_k x_k \cdot \mu_{jk} + \sum_k x_k \cdot N(0,1) \cdot \sigma_{jk} \end{aligned} \tag{4.3}$$

where k is the dimensionality of the input $\boldsymbol{x}$ and $N\left(\mu_{jk},\sigma_{jk}\right)$ represents a particular sample drawn from a normal probability distribution with mean μ_{jk} and variance σ_{jk}.

Realizing that a normal distribution with a particular mean and variance is equivalent to a scaled and shifted version of a normal distribution with zero mean and unit variance, we partition the inference equation, as shown in Equation **4.3**. The constant parameters μ_{jk} and σ_{jk} represent the mean and variance of the probability distribution of the corresponding synaptic weight and can,

therefore, be implemented by DW-MTJ-based memory devices from a hardware implementation perspective. The resultant system (see Figure **4-5**) consists of two crossbar arrays for storing the mean and variance parameters. While the inputs of a particular layer are directly applied to the crossbar array storing the mean values, they are scaled by the random numbers generated from the RNG unit (outputs normalized to provide random numbers with zero mean and unit variance) described previously for the crossbar array storing the variance values. Typical CMOS neuromorphic architectures are characterized by much higher movement of weight data than input data to compute the inference operation [177]. This proposal of computation partition, explained in Box 2, enables us to leverage the "in-memory" computing primitives for storing the probability distribution parameters while parallelly computing energy-efficient dot products in situ between inputs and stochastic weights. It is worth noting here that the crossbar column outputs are computed and read sequentially in order to ensure that the random numbers sampled for the synaptic weights of each column are independent.

5. Results and Discussion

A hybrid device-circuit-algorithm co-simulation framework was developed to evaluate the performance of the proposed all-spin Bayesian hardware. The magnetization switching characteristics of the monodomain and multidomain MTJ was simulated in MuMax3, a GPU accelerated micromagnetic simulation framework [178]. The nonequilibrium Green's function (NEGF)-based transport simulation framework [179] was used for modeling the MTJ resistance variation with oxide thickness and applied voltage. The obtained device characteristics from MuMax3 and SPICE simulation tools were used in an algorithm-level simulator, PyTorch, to evaluate the functionality of the circuit. The performance of this design was tested for a standard digit recognition problem on the MNIST data set [161]. A two-layer fully connected neural network was used, with each hidden layer having 200 neurons. The probability distributions were learned

using the "Bayes by Backprop" algorithm[1] [118], which learns the optimal Gaussian distribution by minimizing the Kullback–Leibler (KL) divergence[2] from the true probability distribution. The prior distribution on the weights used for training was a scaled mixture of two Gaussian functions. The network was trained offline to obtain the values of the mean and standard deviation of the probability distributions of the weights. Subsequently, they were mapped to the conductance values of the DW-MTJ devices. The baseline idealized software network was trained with an accuracy of 98.63% over the training set and 97.51% over the testing set (averaged over ten sampled networks).

The device parameters used in this article have been tabulated in Table **4-1** and Table **4-2**; $20k_{\mathrm{B}}T$ barrier height magnet was used in the Gaussian RNG unit. 4-bit representation in the DW-MTJ weights and 3-bit discretization in the neuron output are adopted. Note that the neuronal devices mimic a saturating linear functionality and our network was trained with such a transfer function itself, as is indicated in Figure **4-2**. Considering a minimum sensing and programming displacement of $20\mathrm{nm}$ for the DW location, the cross-point and neuronal devices were considered to be 320 and $160\mathrm{nm}$ in length. From the micromagnetic simulations, it is observed that the critical current required to switch the neuronal device from one edge to the other is $4\mu\mathrm{A}$ for a time duration of $10\mathrm{ns}$. The crossbar supply voltage was assumed to be $100\mathrm{mV}$ for evaluating the crossbar power consumption. The crossbar resistance ranges (which can be varied by the oxide thickness) were designed to provide the critical current requirement for the spin neurons. A TMR of 300% was adopted in the DW-MTJ conductance values of the crossbar array. Considering such device-level behavioral characteristics, nonidealities, and constraints, the test accuracy of the network was

[1]The related code can be found at https://github.com/nitarshan/bayes-bybackprop.

[2]The KL divergence is a measure of the difference between two probability distributions. In this case, the KL divergence is between the true posterior $P(\boldsymbol{W}|XD)$ and the approximated posterior $q(\boldsymbol{W},\theta)$. It can be shown that minimization of this difference function can be achieved by using the gradient descent method and iteratively updating the variational parameters, μ and σ [118]. This is referred to as the "Bayes by Backprop" algorithm.

96.98% (averaged over ten samples). Furthermore, nonideal DW programming can also impact the system accuracy. Five independent Monte Carlo runs of the network were performed with a 10% variation in each of the programmed crossbar device conductance values. The average accuracy degradation was observed to be insignificant − 96.74%.

In order to estimate the system-level energy consumption, the core RNG and crossbar energy consumption were considered along with peripheral circuitry, such as ADC and DAC[33]. The energy consumption was evaluated for a single-image inference and a particular network sample. The crossbar read latency was assumed to be 10ns (for each column read). During each 10-ns column read, the power consumption for the DAC and the corresponding crossbar column was considered. Subsequently, the neuron device state was read and converted to a digital value using an ADC. The neuron is reset before every operation. For the RNG, DAC, and ADC units, 8-bit precision was adopted and three variables were used for the accumulation process in the normal distribution sampling. 8-bit precision was assumed for the energy calculations in order to achieve a fair comparison with numbers reported in [175] for an iso network CMOS architecture. However, from a functional viewpoint, lower bit-precision ∼4 bits was observed to be sufficient. The total energy consumption of the proposed "all-spin" network was evaluated to be 790.2nJ per classification, which is 24 × energy efficient in contrast to the baseline CMOS implementation [175]. The energy consumption of the RNG unit, including peripherals for adding the random numbers generated per row, was estimated to be 446.8nJ. Energy consumption of the crossbar array, including DAC, ADC, and multiplier peripherals, was 343.3nJ. The system-level energy efficiency stems from both the RNG design and utilization of the "in-memory" computing units.

Note that resistive crossbars are usually characterized by limited fan-in—much smaller than neuron fan-in in typical deep networks due to nonidealities, parasitics, and sneak paths. Hence,

[3]The energy consumption for the peripheral circuitry was included from typical numbers considered in the literature [180], [181], and can be found at https://github.com/Aayush-Ankit/puma-simulator/blob/training/include/constants.py.

mapping a practically sized network requires mapping synapses of a neuron across multiple crossbars [180], [181]. Such architectural-level innovations can be easily integrated with the current proposal.

6. Summary

In summary, this work proposed the vision of an "all-spin" Bayesian neural processor that has the potential of enabling orders of magnitude hardware efficiency (area, power, and energy consumption) in contrast to state-of-the-art CMOS implementations. Computing frameworks, so far, have mainly segregated deterministic and stochastic computations. Standard deterministic deep-learning frameworks enabled by spintronic devices and other post-CMOS technologies have been explored. In such scenarios, device-level nonidealities are usually treated as a disadvantage. More recently, stochasticity inherent in such devices (for instance, probabilistic switching in the presence of thermal noise) has been exploited for computing to implement stochastic versions of their deterministic counterparts [60], [74]. Due to additional information encoding capacity in the switching probability, such devices can be scaled down to single bit instead of multibit representations. Device stochasticity has also been used in other unconventional computing platforms, such as Ising computing and combinatorial optimization problems, among others [142]. Note that prior work on using magnetic devices for Bayesian inference engines have been proposed [182], [183], which are mainly used for implementing Bayes' rule for simple prediction tasks in directed acyclic graphs and do not have relevance or overlap with Bayesian deep networks. Bayesian deep learning is a unique computing framework that necessitates the merger of both deterministic (dot-product evaluations of sampled weights and inputs) and stochastic computations (sampling weights from probability distributions), thereby requiring a significant rethinking of the design space across the stack from devices to circuits and algorithms.

Chapter 5

Leveraging Voltage-Controlled Magnetic Anisotropy to Solve Sneak Path Issues in Crossbar Arrays

In crossbar array structures, which serves as an "in-memory" compute engine for artificial intelligence (AI) hardware, write sneak path problem causes undesired switching of devices that degrades network accuracy. While custom crossbar programming schemes have been proposed, device-level innovations leveraging nonlinear switching characteristics of the cross-point devices are still under exploration to improve the energy efficiency of the write process. In this work, a spintronic device design based on magnetic tunnel junction (MTJ) exploiting the use of voltage-controlled magnetic anisotropy (VCMA) effect is proposed as a solution to the write sneak path problem. In addition, insights are provided regarding appropriate operating voltage conditions to preserve the robustness of the magnetization trajectory during switching, which is critical for proper switching probability manipulation.

1. Motivation

Crossbar array structure is the key computational primitive required in NN hardware acceleration. In a crossbar array, devices (i.e., synapses) are present at the junction points of the array, as is shown in Figure **2-12**. The crossbar in the figure receives input voltage signals along the horizontal lines and produces output current signals along the vertical lines. Following Kirchhoff's law, output current along the column is given by the Equation **2.12**. Though the crossbar array structure is intrinsically efficient in dot-product calculation, errors may occur due to undesired "sneak path" problems [184]–[186]. Sneak path issue refers to the situation where an applied voltage causes undesired current flowing through devices that are not supposed to be read/written, which results in an error in the reading/writing process. For example, in Figure **5-1**(a),

the blue current indicates the desired current, which passes through device D_S, while there may also be the undesired red current along the other row, because there can be a voltage drop across devices $D_{E,1}$, $D_{E,2}$, and $D_{E,3}$, since the other input and output terminals of the array are floating. While sneak path issue occurs both in the reading and writing process, write sneak path issue is a more challenging problem in "in-memory" dot-product calculation. To solve the sneak path issue, usually, a custom programming scheme is adopted for the write process [187]. As is shown in Figure **5-1**(b), instead of just applying a voltage signal to terminals linked to the device to switch, all terminals receive a voltage input, such that the voltage difference is controlled, and sneak path current is mitigated. The device to switch is noted as the "selected cell" (D_S) and applied a full set voltage, U_{Set}, and the devices noted as "half-selected cells" (D_{HS}) are under a half-set voltage, which is not sufficient enough for switching to occur. The remaining cells are "not-selected cells" (D_{NS}), which experience zero voltage drop. Though the sneak path issue can be reduced under this programming scheme, current flowing through half-selected cells contributes to unwanted energy consumption. For this reason, device-level innovations have been pursued to reduce the energy cost.

Several device designs based on resistive random access memory (RRAM) have been proposed [188]–[190]. A typical RRAM device is composed of two metal electrodes and an insulating layer in between. In the insulating layer, a conducting filament (CF) can be formed and modulated when voltage signals are applied across the electrodes. Device conductance is determined by the state of the CF. Previous works have mainly proposed cell designs consisting of multiple devices, such as 1T-1M [191]–[193], 1D-1M [194], [195], and 1S-1M [196], [197], where an additional transistor (T), diode (D), or selector (S) is used to reduce or block the undesired current for half-selected cells. While the energy consumption is reduced due to the mitigation of undesired current, such cross-point designs with multiple devices are not area-efficient and have been succeeded by single-device cell design proposals that leverage intrinsic nonlinear $I-V$

characteristics of the cross-point device itself. The cell design exhibiting similar $I-V$ characteristics as a 1S-1M cell is called a self-selective memristor [198]–[200], while that behaving similar to a 1D-1M cell is called a self-rectifying memristor [201], [202]. The nonlinear $I-V$ characteristics of the single-device cell design enable the reduction of undesired current in a similar manner as the multiple-device cell structure along with a higher area efficiency. However, such a bit-cell proposal exploiting nonlinear $I-V$ characteristics of crosspoint devices has not been explored before for spintronic crosspoint arrays.

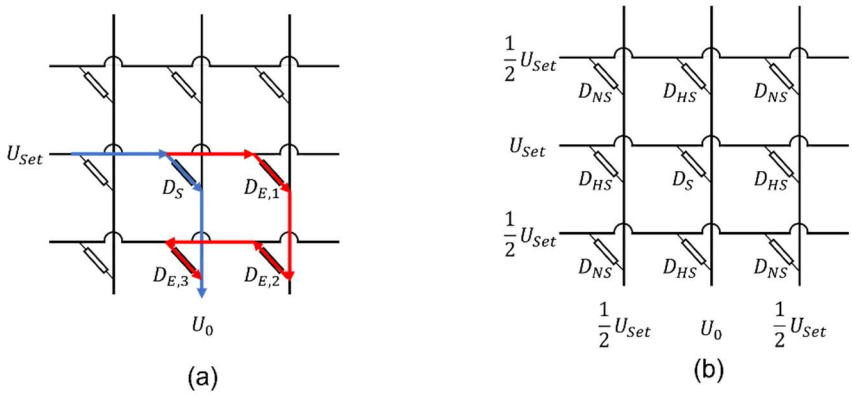

Figure **5-1**: (a) Sneak path current can be induced during reading and writing operations of the crossbar array. To read/write a selected cell D_S, a voltage signal U_{Set} is applied to the device ($U_0 = 0$), which leads to a read/write current (denoted by blue arrows). On the other hand, this voltage drop also results in sneak path current (denoted by red arrows) passing through other devices ($D_{E,1}$, $D_{E,2}$, and $D_{E,3}$), which causes errors in the reading and writing process. (b) Under the custom programming scheme, the selected device D_S is under a full set voltage U_{Set}. The half-selected devices D_{HS} are under $1/2U_{Set}$ voltage drop. The not-selected devices D_{NS} are under zero voltage drop.

Compared with other nonvolatile memory technologies, spintronic devices possess the advantage of lower operating voltage, which reduces energy consumption, faster read and write

processes, unlimited endurance, and compatibility to conventional CMOS-based systems, which makes it a promising choice to build the next generation hardware platform for neuromorphic computing systems [203]. However, the intrinsic nonlinear physics of emergent novel switching mechanisms of spintronic devices has not been leveraged before to mitigate the sneak path current issue in crossbar array-based systems. In this work, a single-device bit-cell solution leveraging voltage-controlled magnetic anisotropy (VCMA) effect is proposed. During the writing process, the full set voltage applied to selected cell switches the device via VCMA effect, while the switching of half-selected cells is still dominated by spin transfer torque (STT). Since the pulsewidth to achieve high switching probability by VCMA effect is much shorter, the pulsewidth of set voltage signal can be chosen properly, so that a high switching probability is achieved for selected cell, while the half-selected cells still have a low switching probability tending to zero. The sharp switching probability difference among half-selected cells and selected cells under the applied voltage signal enables a high write accuracy, since undesired switching of half-selected cells is restrained. More importantly, the sharp increase in switching probability is due to the change in the switching mechanism and is independent of the pulsewidth and can be achieved even with a short pulse in this proposed framework. Compared with the STT dominated mechanism, where the sharpness of switching probability increase with pulse amplitude is related to the pulsewidth [204], the proposed framework provides more design-time flexibility. Simultaneously, the proposed solution reduces the system-level energy consumption due to short pulse widths required by the VCMA effect for magnetic state switching.

2. Preliminaries

2.1. Device Physics

The device utilized in this work is based on MTJ introduced in chapter 2 (see Figure **2-7**). The resistance between P and AP state of MTJ enables information encoding. STT induced by spin current can be used to switch the device state. But, to achieve a high switching probability, a long current pulse is required, since the switching probability increases with pulsewidth. To reduce the energy consumption, it is necessary to reduce the pulsewidth of applied pulses. A recent work has shown that short switching pulses can be achieved through VCMA effect [205]. VCMA effect enables manipulation of device magnetocrystalline anisotropy energy (MAE, which is the magnetic energy difference between perpendicular and in-plane direction) by an applied voltage via spin–orbit interactions (which consists of two contributions, namely, angular momentum and magnetic dipole momentum [206], [207]). The orbital angular momentum dominates in strong ferromagnetic materials [208] and can be modulated by doping of charges with selective spin direction. On the other hand, magnetic dipole momentum modification (which results from intra-atomic electron redistribution) is the dominating mechanism in materials where spin–orbit interaction is not strong enough [209]. Such VCMA effect enables the change of magnetic anisotropy from perpendicular to in-plane direction.

During the transition of magnetic anisotropy from perpendicular direction to in-plane direction, the FL magnetization performs precession along the in-plane easy axis, as is shown in Figure **5-2**. Considering that the initial magnetization is at the south pole, the switching probability is high when the magnetization stops in the upper half of the unit sphere [region near point A in Figure **5-2**] and low in the lower half [region near point B in Figure **5-2**], such that the switching probability can be controlled by the pulsewidth. Prior work has reported that the VCMA-induced switching requires a shorter pulsewidth that STT-induced switching [205].

The difference in required pulsewidth between VCMA-induced switching and STT-induced switching leads to a possible solution to the sneak path problem based on the programming scheme illustrated in Figure **5-1**(b). If the set voltage is chosen appropriately, such that the selected cell operates via the VCMA-induced switching mechanism (although STT is present in this case, the switching process is dominated by the VCMA effect), while half-selected cells operate via the STT-induced switching mechanism, the pulsewidth applied to switch the selected cell will only result in near-zero switching probability to half-selected cells. In this way, a sharp difference in switching probability of selected and half-selected cells can be achieved.

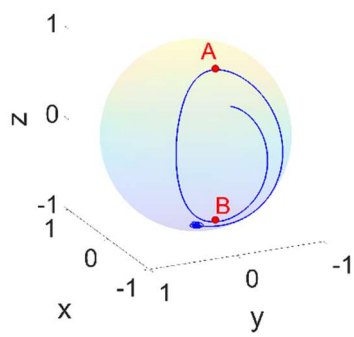

Figure **5-2**: Precession trajectory induced by VCMA effect along in-plane axis is shown. A high switching probability can be achieved if VCMA voltage pulse terminates when magnetization is at point A. The switching probability is low if VCMA pulse terminates when magnetization is at point B.

2.2. Simulation Method

The behavior of magnetization under an applied external voltage signal that causes STT and VCMA effect can be simulated by the LLG equation given in Equation **2.11**. To model the VCMA effect on MTJ, an additional effective magnetic field $\boldsymbol{H}_{\mathrm{VCMA}} = \left(\frac{2K_{\mathrm{ieff}}(U)}{\mu_0 M_S t_{\mathrm{FL}}}\right) m_z \hat{z}$ is added to

$\boldsymbol{H}_{\text{eff}}$ in Equation **2.11** [210], where t_{FL} is the FL thickness, and m_z is the z component of unit magnetization of FL, $\hat{m}$. $K_{\text{ieff}}(U) = K_i - \xi\left(\frac{U}{t_{\text{OX}}}\right)$ is the expression of effective energy density for interface perpendicular anisotropy, where K_i is the energy density of perpendicular anisotropy without applied voltage U, ξ is the VCMA coefficient, and t_{OX} is the oxide layer thickness. If not mentioned specifically, simulations are based on parameters mentioned in Table **5-1**. The electrical resistance of the MTJ in the P and AP states is obtained from the modeling framework [211] benchmarked to experimental data reported previously in work [205].

Table **5-1**: Device Simulation Parameters are tabulated.

Parameter	Value
Free-layer width, W_{MTJ}	40nm
Free-layer length, L_{MTJ}	70nm
Free-layer thickness, t_{MTJ}	0.9nm
Oxide layer thickness, t_{OX}	1.3nm
Saturation magnetization, M_{S} [211]	1257.3kA/m
Gilbert damping factor, α	0.075
Temperature, T	300K
VCMA coefficient, ξ	200fJ/V·m
Interfacial perpendicular anisotropy, K_i [211]	0.9267mJ/m^2

3. Proposal

3.1. Simulation Results

Figure **5-3**(a) shows the relation between MTJ switching probability and pulsewidth for 0.7-V voltage pulses. The fluctuation in switching probability results from the precession along the in-plane axis. The peaks (valleys) are according to the cases where the voltage pulse terminates when the magnetization rotates to the top (bottom) positions of the trajectory. The peaks and valleys tend to be 50% switching probability, as the magnetization gradually rotates to the in-plane direction with increasing pulsewidth, which is the new easy axis under VCMA effect. The

switching probability is stable at 50% when the FL magnetization remains in the in-plane direction. The 92.1% switching probability at the first peak implies the viability to switch the device with a set voltage pulse as short as 1.8ns. Figure **5-3**(b) shows the relation between switching probability and pulse amplitude for 1.8-ns wide switching pulses. The switching probability remains low for pulses with small amplitude (region A). The reason is that the VCMA effect induced by such low amplitude pulses is not strong enough. In this region, the STT dominates the switching event, which requires a longer pulse for high switching probability. The sharp increase of switching probability at 0.65V indicates that the VCMA effect induced by 0.65-V pulse is strong enough to change the magnetic anisotropy from perpendicular direction to in-plane direction. The 1.8-ns pulsewidth enables the magnetization to reach the top half of the trajectory (see Figure **5-2**), resulting in a switching probability of 92.1% (region B). The switching probability again drops when the pulse amplitude is larger than 0.8V (region C). This can be explained by the magnetization trajectory robustness, which will be discussed in the next section.

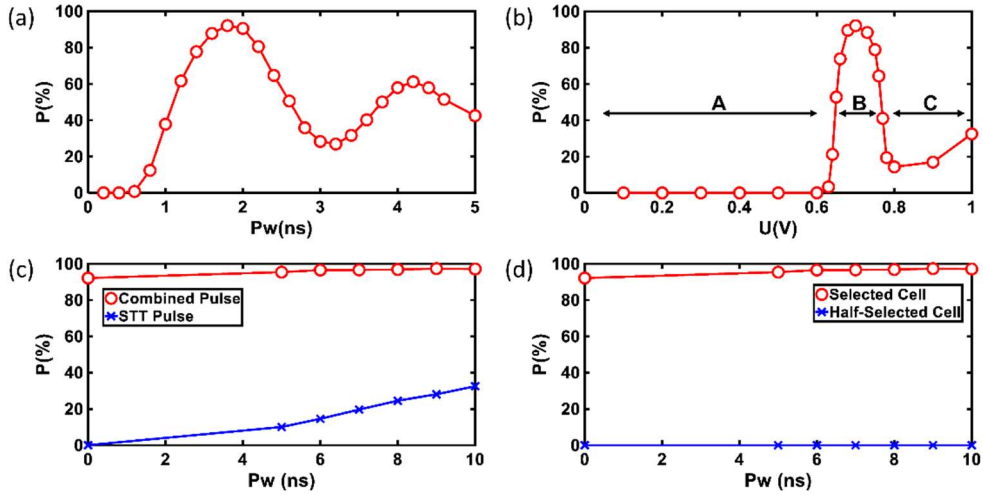

Figure **5-3**: (a) Switching probability (P) changes with applied pulsewidth (Pw) for pulse magnitude $U = 0.7$V. The precession of device magnetization results in the peaks and valleys. The highest peak depicts a switching probability of 92.1% for a pulsewidth of 1.8ns. (b) Variation of

switching probability with pulse amplitude is shown. The pulsewidth is fixed to be 1.8ns. (c) Switching probability—pulsewidth (following STT pulse) variation for combined pulsing scheme is given by the red curve, and that of a pure STT pulse is given by the blue curve. (d) Variation of selected cell (under U_{Set} voltage corresponding to VCMA-STT pulsing scheme) switching probability with the following STT pulsewidth is given by the red curve. Half-selected cell (under $1/2U_{\mathrm{Set}}$ voltage corresponding to VCMA-STT pulsing scheme) switching probability variation with the following STT pulsewidth is given by the blue curve. U_{Set} is a VCMA-STT combined pulse (0.7-V, 1.8-ns VCMA pulse followed by 0.6-V STT pulse).

To further increase the switching probability, another STT pulse can be applied after the VCMA pulse, as is proposed in prior literature [205]. In this work, the VCMA-STT combined pulse consists of a short VCMA pulse (0.7V, 1.8ns) and a following STT pulse (0.6V). The pulsewidth of the STT pulse can be fixed in accordance with the desired switching probability of the selected cell. In order to justify the contribution of VCMA switching to the proposal, we also compare against a pure STT pulsing scheme consisting of a first STT pulse (0.6V, 1.8ns) and a following STT pulse (0.6V). Figure **5-3**(c) shows that the switching probability of the selected cell under VCMA-STT combined switching increases with the pulsewidth of the following STT pulse. The switching probability reaches 97% when the following STT pulse is longer than 9ns. On the other hand, the switching probability for the pure STT pulse scenario is much lower than that of VCMA-STT combined pulse, which indicates that the VCMA-STT combined pulse can be much shorter (and, therefore, much more energy-efficient) than pure STT pulse to reach a high switching probability for the selected cell.

On the other hand, in the programming scheme shown in Figure **5-1**(b), there are also half-selected cells that experience half-set voltage during the switching process. To avoid undesired switching, such half-selected devices should exhibit near-zero switching probability. As is shown in Figure **5-3**(d), the switching probability remains near zero for the half-selected cell even when

the selected cell experiences a switching probability over 97%. The sharp difference in switching probability resulting from the VCMA effect makes it possible to ensure a high switching probability for selected cells and a near-zero switching probability for half-selected cells and, therefore, provides a solution to the sneak path problem for MTJ-based spintronic cross-point arrays.

Unlike logic applications, neuromorphic computing applications are resilient to minor imprecision in hardware operation. While the maximum switching probability shown for AP to P switching was $\approx 97\%$ (note that P to AP switching will have a slightly reduced switching probability, since VCMA pulse is always of the same polarity, and STT pulse varies in polarity in the two cases), this did not have any significant impact at the system level. On-chip learning simulations were performed for a 784×10 network on the MNIST dataset. The ideal software accuracy was evaluated to be 91.13% (five epochs). The weight values in the network were implemented using 10-bit resolution. The weight discretized network (considering 100% switching probability in the devices) had an accuracy of 90.35% (five epochs), while the hardware-realistic simulation with slightly reduced switching probabilities had an accuracy of 89.34% (five epochs, averaged for five independent runs of the training process), which is only $\approx 1.01\%$ lower than the network with no switching error. The training convergence time is not affected due to the hardware nonidealities and constraints (see Figure **5-4**).

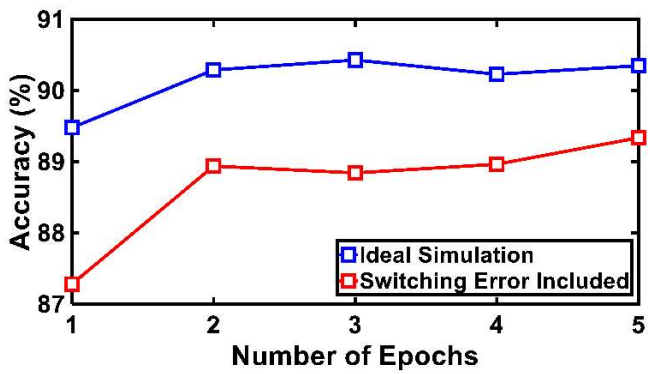

Figure **5-4**: Accuracy of network with and without switching error has been obtained for different training epochs. Switching error only causes a reduction of 1.01% in accuracy after five epochs of training.

3.2. Robustness

It is observed that even when the magnetization motion is dominated by VCMA effect, switching probability may still be low, as is shown in Figure **5-3**(b) (region C). This is due to the loss of magnetization trajectory robustness. The robustness refers to the uniqueness of the route of magnetization precession. The high switching probability results from the fact that every time the voltage pulse ends, the magnetization is right at the top of the trajectory, which only happens when the magnetization follows the same trajectory. If the magnetization precession trajectory is random, there is no determined relation between pulsewidth and final position of magnetization. In other words, a high switching probability can be ensured only when there is a certain magnetization trajectory (i.e., the robustness is preserved).

In order to figure out how the robustness is preserved, it is necessary to study the motion of FL magnetization, $\hat{m}$. The FL magnetization motion can be characterized by the motion "velocity" on the unit sphere, $\frac{d\hat{m}}{dt}$, which is given by the LLG equation in Equation **2.11**. The precession is mainly related to the first term, $-\hat{m} \times \boldsymbol{H}_{\text{eff}}$, where VCMA effect contributes by

adding an effective field $\boldsymbol{H}_{\mathrm{VCMA}}$ in the $\hat{z}$-direction. Denoting the total magnetic field along $\hat{x}$ (short axis), $\hat{y}$ (long axis), and $\hat{z}$ (perpendicular axis) directions as $\boldsymbol{H}_x$, $\boldsymbol{H}_y$, and $\boldsymbol{H}_z$ respectively, Figure **5-5** shows the direction of the vector field $-\hat{m} \times \boldsymbol{H}_x$, $-\hat{m} \times \boldsymbol{H}_y$, and $-\hat{m} \times \boldsymbol{H}_z$ under $U = 0.7$V. Any component of H_i $(i \in x, y, z)$ forms a precession along axis i solely. Note that since $\hat{m} \cdot \boldsymbol{H}_i < 0$, there is also a component repelling $\hat{m}$ from axis i. Direction of the total field depends on the magnitude relation among $\boldsymbol{H}_x$, $\boldsymbol{H}_y$, and $\boldsymbol{H}_z$. For example, at the position of point A, where $\boldsymbol{H}_y$ is small and can be neglected, total field direction will be in the same direction as $\boldsymbol{H}_x$ ($\boldsymbol{H}_z$) if $|\boldsymbol{H}_x| > |\boldsymbol{H}_z|$ ($|\boldsymbol{H}_x| < |\boldsymbol{H}_z|$). For the same reason, total field at point B is dominated by the larger component between $\boldsymbol{H}_y$ and $\boldsymbol{H}_z$. As a result, competition among $|\boldsymbol{H}_x|$, $|\boldsymbol{H}_y|$, and $|\boldsymbol{H}_z|$ leads to a different total field.

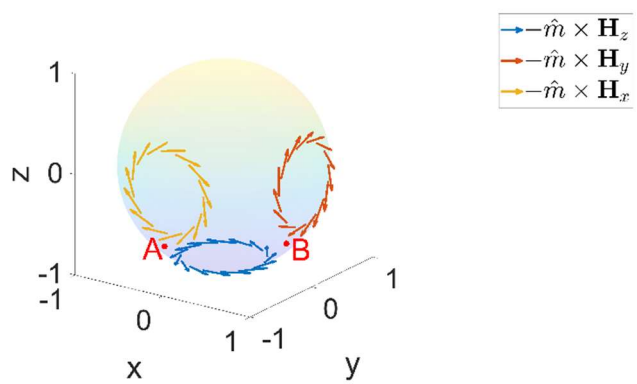

Figure **5-5**: Field vectors $-\hat{m} \times \boldsymbol{H}_x$, $-\hat{m} \times \boldsymbol{H}_y$, and $-\hat{m} \times \boldsymbol{H}_z$ under $U = 0.7$V are plotted on the unit sphere. Each of the fields leads to a precession of the magnetization along the corresponding axis, with a repelling component. Since the field vectors have components along opposite directions in the adjacent region between any two pairs of the three fields, the direction of the total field depends on the relative magnitude of $\boldsymbol{H}_x$, $\boldsymbol{H}_y$, and $\boldsymbol{H}_z$.

Figure **5-6**(a) shows the total $\frac{d\hat{m}}{dt}$ field for applied voltage $U = 0.7\text{V}$ around the south pole, which is the initial location of the device magnetization. In this situation, the magnitude of the magnetic field satisfies $|\boldsymbol{H}_x| < |\boldsymbol{H}_z| < |\boldsymbol{H}_y|$, which results in two symmetric exit windows. Magnetization trajectory robustness can be preserved since the precession starts from one of the two symmetric exit windows every time. Figure **5-6**(b) shows the magnetization trajectories of ten LLG runs for $U = 0.7\text{V}$. Magnetization leaves the north pole from one of the two exits every time and follows a certain trajectory. In this case, the switching probability can be controlled by the applied pulsewidth since the relation between the position of magnetization and pulsewidth is determined by the fixed trajectory. On the other hand, for larger applied voltage, the magnitude of magnetic field satisfies $|\boldsymbol{H}_x| < |\boldsymbol{H}_y| < |\boldsymbol{H}_z|$. Figure **5-6**(c) shows the total $\frac{d\hat{m}}{dt}$ field for applied voltage $U = 0.8\text{V}$. There is no definite exit points in the pole region in this situation, as is shown in Figure **5-6**(d). The trajectories are random, and the switching probability cannot be controlled by the applied pulsewidth. On the other hand, although the magnetization trajectories are random in this case, the switching probability is still increasing with pulse amplitude, as is shown in Figure **5-3**(b). The reason is that the magnetic energy in the $\hat{z}$-direction increases with applied pulse amplitude due to increasing $|\boldsymbol{H}_z|$ caused by the VCMA effect. Due to larger magnetic energy, for larger pulse amplitude, the magnetization is more likely to leave the pole and switch to the other side when the pulse ends, leading to a higher switching probability.

As a result, to enable a high switching probability, the applied voltage U has to be in a range determined by device parameters. The relation is given by Equation **5.1**.

$$\frac{t_{\text{OX}}}{\xi}\left(\frac{(N_{xx} - N_{zz})M_S^2\mu_0 t_{\text{MTJ}}}{2} + K_i\right) < U < \frac{t_{\text{OX}}}{\xi}\left(\frac{(N_{yy} - N_{zz})M_S^2\mu_0 t_{\text{MTJ}}}{2} + K_i\right) \qquad \textbf{(5.1)}$$

In the equation, N_{xx}, N_{yy}, and N_{zz} are demagnetization factors determined by the device shape. Other parameters are the same as the ones introduced in Table **5-1**. Since VCMA is a surface effect occurring at the interface between the FL and the oxide layer, variations in FL thickness play a

critical role in ensuring that a given operating voltage range is robust enough to device variations. The robustness of the operating voltage range to device parameter variations implies that there should be an overlapping region in the operating voltage ranges of all devices in the system, such that they can be programed at the same voltage. To verify the operating voltage range robustness, the operating window of 1000 devices was calculated according to Equation **5.1** and $6\sigma = 1.5\%$ [210] variation in the FL thickness was considered. It was found that over 99% of all 1000 devices can work at the same applied voltage in the operating voltage range between 0.68 and 0.73V. It is worth mentioning here that other device, circuit, and system-level parameters, such as Gilbert's damping ratio, input pulse shape waveform variations can also influence the magnetization reversal in the time domain [212].

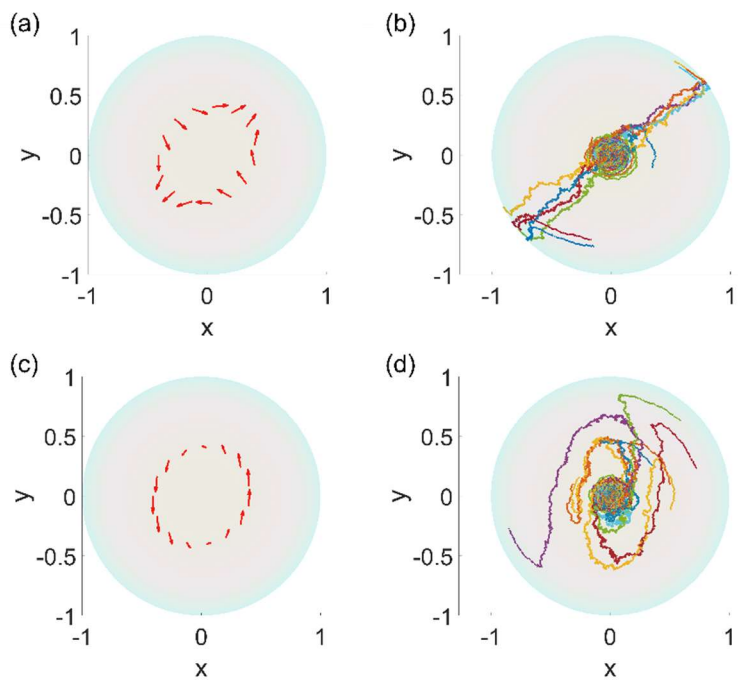

Figure **5-6**: (a) Total $\frac{d\hat{m}}{dt}$ field under applied voltage $U = 0.7$V in the region around the south pole of the unit sphere. The relative magnitude of $\boldsymbol{H}_x$, $\boldsymbol{H}_y$, and $\boldsymbol{H}_z$ results in two symmetric exit windows along diagonal directions in the XY plane. (b) Trajectories of ten LLG simulations

leave the pole area from the exit windows, which enables stable magnetization motion. (c) Total $\frac{d\hat{m}}{dt}$ field under applied voltage $U = 0.8\text{V}$ in the region around the south pole. In this situation, $\boldsymbol{H}_z$ dominates, and the total field does not form definite exit windows unlike the $U = 0.7$-V case. (d) Trajectories of ten LLG simulations for applied voltage $U = 0.8\text{V}$ are almost random.

4. Conclusion

In this work, a spintronic device utilizing VCMA effect-induced switching scheme is proposed as a solution to the sneak path problem in neuromorphic non-volatile cross-point arrays. The required pulsewidth difference between VCMA-induced switching and STT switching leads to a sharp difference in switching probability (over 97% for VCMA-induced switching and $\approx 0\%$ for STT switching) and, thereby, enables a potentially energy efficient solution to the write sneak path problem. In addition, it is also observed that ensuring a specific operating voltage range is critical for the VCMA effect to ensure high switching probability of selected cells, such that the effective magnetic field $\boldsymbol{H}_z$ does not exceed $\boldsymbol{H}_x$, which leads to the loss of FL magnetization trajectory robustness.

Chapter 6

Hardware in Loop Learning with Spin Stochastic Neurons

Despite the promise of superior efficiency and scalability, real-world deployment of emerging nanoelectronic platforms for brain-inspired computing have been limited thus far, primarily because of inter-device variations and intrinsic non-idealities. In this work, it is demonstrated that mitigating these issues by performing learning directly on practical devices through a hardware-in-loop approach, utilizing stochastic neurons based on heavy metal/ferromagnetic spin-orbit torque heterostructures. The probabilistic switching and device-to-device variability of our fabricated devices of various sizes are characterized to showcase the effect of device dimension on the neuronal dynamics and its consequent impact on network-level performance. The efficacy of the hardware-in-loop scheme is illustrated in a deep learning scenario achieving equivalent software performance. This work paves the way for future large-scale implementations of neuromorphic hardware and realization of truly autonomous edge-intelligent devices.

1. Motivation

Interest in bio-plausible devices and systems stems from the brain's unique ability to process real-world information effectively and efficiently. In recent times, although deep artificial neural networks have been able to come close and even surpass human-level performance in some cases, the energy and area cost associated with such systems are still many-folds over their biological counterparts [213]. Thus, in-memory computing paradigms akin to the brain designed with specialized electronics for intrinsic emulation of neuronal and synaptic functionalities have emerged as alternatives to the traditional von-Neumann architecture and complementary metal oxide semiconductor (CMOS) technologies [69], [214]–[218]. However, their adoption in large-

scale neuromorphic hardware is scarce[219], relying rather on CMOS [25], [55]. A key factor behind this has been device-to-device variations and inherent non-idealities of these emerging devices [69], [220]–[222], requiring additional peripheral circuitry for reliable operation, eroding their area and energy advantage.

While there have been efforts in literature to characterize device non-idealities and their impact on learning for artificial synapses of varied technologies [69], [223], [224], such studies for artificial neurons, specifically stochastic ones, have been rather limited [60]. The robust computational ability of the brain is largely attributed to its noisy probabilistic nature [225]. Additionally, to reach brain-like memory densities, continual scaling of these devices is needed. Although the energy and area advantages of scaling in CMOS is evident, it is not so straightforward for neuromorphic devices; non-idealities can become an insurmountable issue. Thus, for large-scale integration, it is pivotal to understand the interplay between device size and its impact on the variation and stochasticity of the neuronal dynamics. A systematic analysis through characterization of realistic devices identifying their categorical effect on network-level performance for deep learning applications has been missing. Similarly, various schemes for variation compensation exist, however, their potency has not been tested in experimental demonstrations for deep learning tasks such as pattern recognition.

Spintronic devices with their nanosecond response capabilities, and compatibility with existing nanoelectronics are a great prospect for realizing neuromorphic frameworks [70]. Compared to other emerging technologies such as phase change, and resistive memories [69], [226], spintronic devices are more compact and require less operating energy. Prior works looking at the device-to-device variations of spin-based devices have either looked at them for inference only [227] or have not exploited the device stochasticity [228]. Additionally, previous stochastic hardware implementations and hardware-in-loop learning have focused more on probabilistic computing applications [229], [230] and associative learning [227], [230]. The more powerful and widely used neural networks employing backpropagation for training have only been demonstrated

in software [60], [231]. Moreover, these demonstrations have all lacked comprehensive characterization analysis of device properties. Our work tries to overcome all these issues through extensive experimental characterization of devices of varied dimensions and hardware demonstration of a cognitive task, namely recognition of handwritten digits, to present the true potential of stochastic spintronic devices.

In this work, such an investigation has been demonstrated for spintronic stochastic neurons based on a heavy metal/ferromagnetic spin-orbit torque (SOT) hall-bar heterostructures [204] along with a proof-of-concept demonstration of hardware-in-loop learning offsetting intrinsic hardware variations for deep learning applications. The sigmoidal stochastic switching of the SOT devices and its variances for a wide range of device sizes, ranging from 5μm to 300nm have been studied. It has been found that the necessary bias current for switching decreases with decreasing size while the slope of the probabilistic switching characteristics increases, highlighting an intertwined inverse relationship between power consumption, accuracy and robustness in neural network scenarios. Finally, results of our experimental hardware-in-loop setup have been extrapolated to draw insights for large-scale neuromorphic implementations through a hardware-software co-analysis.

2. Materials and Methods

2.1. Materials and Devices Information

In this work, the stochastic neurons displaying non-linear dependence to input current were realized with spintronic devices employing spin-orbit torque. The core device structure adopted is Hall bar (see Figure **2-13**) fabricated through photo-/Ebeam-lithography. Details of fabrication process are described in chapter 2, section 5.1 and 5.2. The fabricated device and material stack is shown in Figure **6-1**(a). A Hall bar structure is used, so that the device magnetization can be probed out using the anomalous Hall effect, where a voltage difference occurs across terminals

perpendicular to the flow of current by accumulation of electrons with different spin directions [132]. The characterization setup has been discussed in chapter 2, section 5.3. A magnetic field in out-of-plane direction is applied to obtain the hysteresis loop shown in Figure **6-1**(b). The rectangle hysteresis loop indicates perpendicular magnetic anisotropy (PMA).

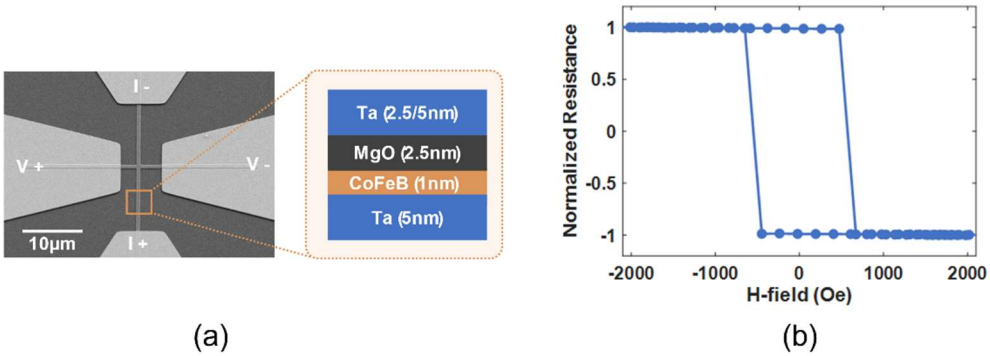

Figure **6-1**: (a) SEM image of the device structure is shown. Connections for measurement are annotated. The figure inset shows the material stack used for the device. (d) Magnetic hysteresis loop of a representative device with out-of-plane magnetic field (H field, Oe) is shown. The rectangular shape of the loop indicated perpendicular magnetic anisotropy (PMA) of the devices.

Other than the hysteresis loop, the magnetic anisotropy field is also estimated for different Hall bars. In-plane magnetic field sweeping measurements can be conducted. The device magnetization gently tilts under the applied in-plane field, which causes a small deviation in Hall resistance. The deviating Hall resistance results in a bending hysteresis loop, which can be used to estimate magnetic anisotropy field. Details have been discussed in chapter 2, section 5.4. The estimated magnetic anisotropy field is uniform for Hall bars of different sizes (see Figure **6-2**), which is reasonable since magnetic anisotropy field is a material-level characteristic.

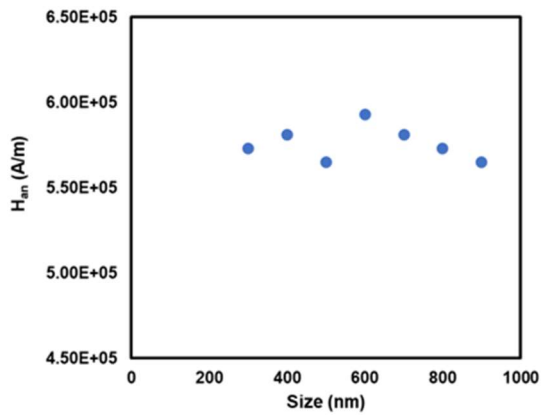

Figure **6-2**: Magnetic anisotropy field of bars with different sizes is shown.

2.2. Hardware-in-Loop Training Methodology

For the hardware-in-loop training, the images are converted to Poisson spike trains of length 100 time-steps. The feedforward network structure consists of 784×4 weight connections, which are randomly initialized. After modulation by the network, the impulses are scaled and biased according to the device characteristics obtained during characterization. A LabVIEW interface handles the generation of the proper pulsing scheme for the neuron devices with the help of the pulse current generator, Keithley 6221. The resultant Hall resistance is measured using a multimeter, Keithley 2000. Based on the resistance values after the reset and programming pulses, the interface counts the number of switching events for a single image. This is used to calculate the activation, by dividing it with the total time-steps, 100. The activation is then used for error and gradient calculations, followed by backpropagation through the network to ultimately calculate the weight updates for the feedforward network in software. The step is repeated for all the training images. During inference, the feedforward network is used, and the activations indicate the network's ability to accurately identify the patterns. The confidence is calculated from the normalized inputs to the neurons from the network and the neuron with the highest confidence is assigned to that class.

3. Results

In biological neural networks, neurons serve as the key computing unit. Inputs are propagated from pre-synaptic neurons to post-synaptic neurons, which integrate them and fire output spikes once a certain threshold is crossed. This kind of neurons are known as integrate-and-fire neurons (see chapter 2, section 2.1.1). On the other hand, neurons, particularly in the cortex, have been observed exhibiting stochastic firing with nonlinear dependence on the resultant post-synaptic current input to the neuron [232]. These are known as stochastic neurons (see chapter 2, section 2.1.2). Thanks to the underlying physics of the FM/HM materials (see chapter 2, section 5.1), the stochastic firing behavior (given by Equation **2.6)** can be obtained with FM/HM Hall bars when pulse current is applied. Previously, it has been shown that the average firing activity of such a stochastic neuron over time linearly approximates the sigmoid function used for computation in traditional neural networks [60]. Going beyond inference, reference [233] featured such a stochastic sigmoid used directly in training neural networks to eradicate the need for evaluating the network over time.

3.1. Effect of dimension on device characteristics

In order to characterize the switching and quantify the effect of dimension reduction, experiments are conducted on devices with bar widths of 5.0μm, 2.5μm, 2.0μm, 1.5μm, 1.0μm, 0.9μm, 0.7μm, 0.5μm and 0.3μm. For each size, we studied the switching behavior of 4 different devices to identify the device-to-device variability. The measurement setup is shown in Figure **2-13**. A 200Oe in-plane magnetic field along the direction of current flow (x-direction) is applied. The characterization begins by confirming the SOT induced magnetization switching of the devices by applying gradually increasing pulses. Again, the state of the device is probed by measuring the anomalous Hall resistance, R_{AHE}, which is defined in Equation **2.13**. It is observed that as the device

width is reduced, the hysteresis loop shrinks accordingly, i.e., the devices switch at lower magnitudes of pulse currents. Note, the current switching behavior is stochastic and thus next this behavior is characterized. For this, devices are applied with 100 iterations of reset-set cycles. In each iteration, the devices are first applied with a reset pulse with a pulse width of 100μs, to initialize the device state. The reset pulse must be large so that the device is in the '-1' state. This is confirmed by reading out the R_{AHE}. Afterwards, a set or write current pulse, I_{write}, with pulse width 100μs is applied to switch the device. The magnitude of I_{write} is increased to figure out the switching dynamics of the devices. As expected, the switching probability of devices shows sigmoid relation with pulse amplitude. The process of obtaining this relation is detailed in Figure **6-3**. In Figure **6-6**(c-k), the switching dynamics of 4 individual devices of each of the 9 widths considered is shown. In order to fit the dynamics to that of an ideal sigmoid, the neuronal switching due to current can be thought of having two components:

$$I_{\mathrm{write}} = I_{\mathrm{bias}} + I_{\mathrm{syn}} \tag{5.1}$$

Here, I_{bias} is the necessary current to the HM layer of the spin neuron to bias it at 50% probability, whereas I_{syn} is the resultant input synaptic current to the neuron. Note, I_{syn} needs to be normalized by a factor I_0, which encodes the degree of dispersion of the neurons' sigmoidal characteristics. Generally, it is found that smaller width corresponds to smaller dispersion or programming window. Additionally, it is also found that the dynamics of each individual spin neuron is quite stable over time (Figure **6-4**). Figure **6-5**(a, b) summarizes the characterization results. It is observed that both the switching bias current, I_{bias} and the programming window, which is defined by the pulse amplitude range between 0.01% and 99.9% switching probability, increase linearly with bar width.

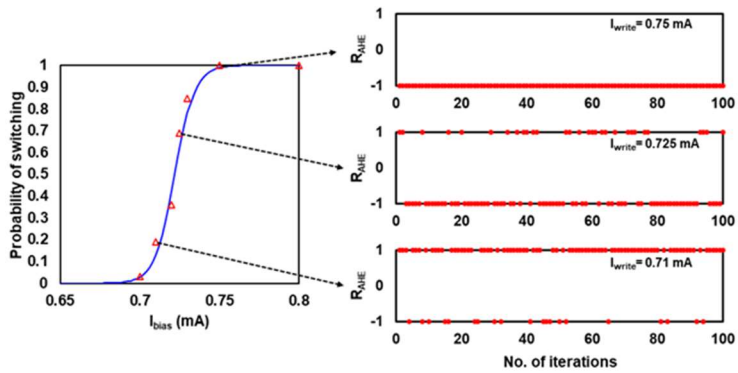

Figure **6-3**: The process of obtaining the sigmoidal characteristics of the neuronal devices is shown. The devices are given a specific write current for 100 iterations. Before each write current pulse, a reset current pulse is applied to reset the device to the high resistance state, as indicated by '+1'. After the write pulse is given, the state of the device is read again. If the device stays in the high resistive state ('+1'), there is no switching. If it goes to the low resistive state (indicated by '-1') then the device is considered to be switched. The total number of switches in the 100 iterations is counted to calculate the probability of switching of the devices.

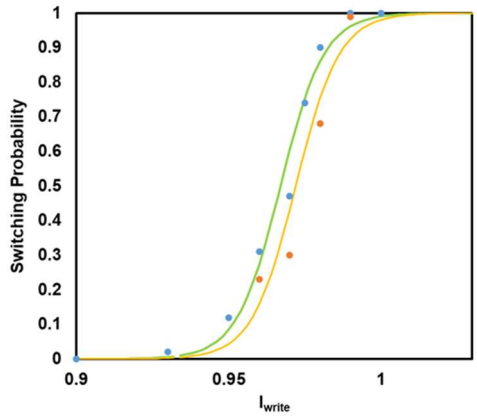

Figure **6-4**: Persistence of neuronal dynamics is shown. The neuronal dynamics of the same device was measured after a week, and it showed similar switching characteristics, with no significant variation (~0.5%) in the bias switching current.

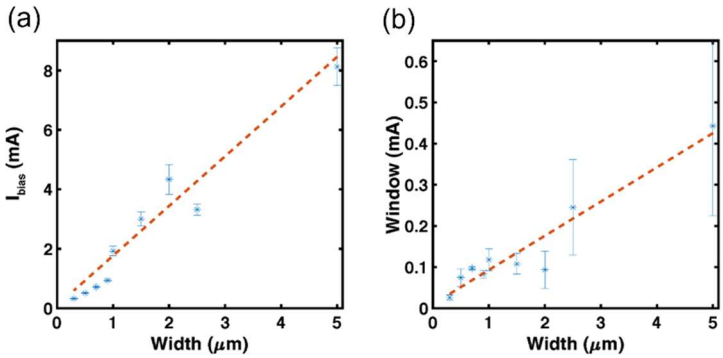

Figure **6-5**: Impact of device dimension on neural dynamics is shown. (a) Relationship between the bias current, I_{bias} and device width is shown. Switching current increases linearly with hall bar width. (b) Relationship between programming window and bar width is shown. Again, with decreasing bar size, it is observed that the programming window also decreases. Device-to-device variation of input-bias current. It is also observed that for the different sizes of the hall bars, we can have up to 25% variation from one device to another.

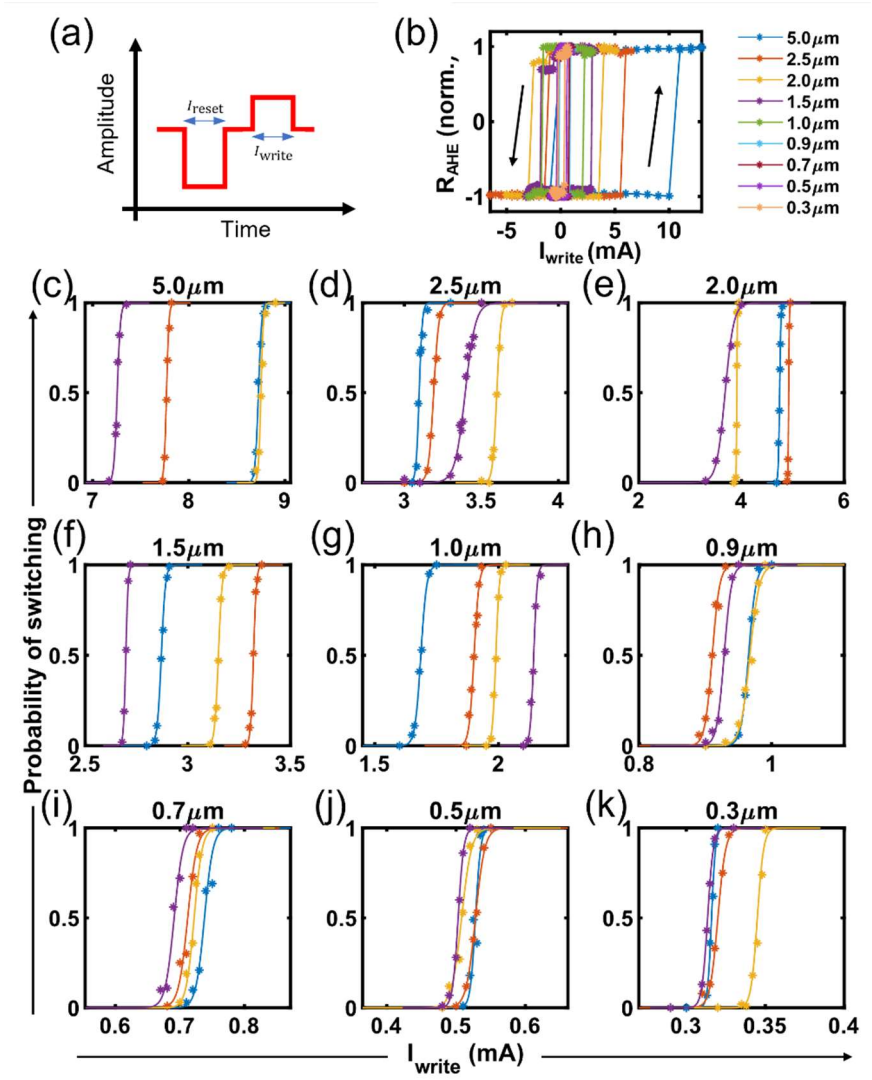

Figure **6-6**: Neuronal dynamics characterization results are shown. (a) Programming pulse is shown. In each iteration, a 100μs reset pulse and a 100μs write pulse are applied. Read pulses of 500ms and 50μA are applied after each programming pulse to read the device state. The interval between pulses is 2s. (b) Normalized Hall resistance of the various sized devices as the write current is gradually swept. It is found that for sufficiently high switching current, the device switches from "-1" state to the "+1" state abruptly. It is found that the hysteresis loop became larger with increasing device size. (c-k) The experimental results of 4 devices each of different sizes of

spin neuron devices (5μm, 2.5μm, 2μm, 1.5μm, 1μm, 0.9μm, 0.7μm, 0.5μm, 0.3μm). Each device's switching dynamics is fitted to that of a sigmoid, showing close resemblance.

3.2. Proof-of-concept hardware-in-loop training

The conspicuous impact of device-to-device variation was highlighted in the previous section with smaller devices being affected more than their larger counterparts. However, scaling is a highly desirable feature as it lowers the total energy cost significantly. Thus, it is essential to overcome these issues. Numerous approaches for repressing these issues have been outlined on the device-level [230], [234], [235] and network-level [192], [236], [237]. Here, proof-of-concept demonstration of a network-level approach is presented. The neuronal devices are included in the training process of a handwritten digit recognition problem, allowing the network to learn the desired patterns with the effect of device-to-device variations included. This is performed through a hardware-in-loop scheme, shown in Figure **6-8**(a), where the incoming training images are converted to temporal spikes and are fed into the neuron hardware after modulation through a feedforward network. Based on the input, the spin neurons switch, which is used to calculate the activations and the subsequent errors, gradients and weight update in software and update the synaptic weights of the network. Further details are provided in the Methods section.

For the hardware-in-loop learning, we use four spin neuron devices of size 0.5μm. The neuronal dynamics of the four devices are shown in Figure **6-8**(b), along with the fitted sigmoid. As four neurons are used as the output, the network is trained on four classes from the MNIST dataset ("0", "2", "4" and "6"), with 4 images from each group. The network architecture is given in Figure **6-7**. For testing, a single sample from each class was used. The network's training loss was tracked during the training process, as illustrated in Figure **6-8**(c). It can be observed that the loss gradually decreases. After training, the network was used for inference on the 4 test images and observe the network input for the 4 hardware neurons. As can be seen from Figure **6-8**(d), the

network is able to differentiate between the classes and correctly identify 3 out of the 4 images. The failure to recognize the class "2" image can be attributed to the small size of training set and the apparent dissimilarity between the training samples and test sample of class "2" (Figure **6-7**). To compare the hardware-in-loop performance with performance without hardware-in-loop training, the same network was trained in software with the same learning parameters and then used the spin stochastic neurons for inference only. In this scenario, we found that the network can identify only 1 sample correctly (Figure **6-8**). This showcases the efficacy of the hardware-in-loop learning scheme. Additionally, as such neuromorphic hardware systems are expected to be employed in resource-constrained environments, we perform the same experiment with a single neuronal device, time-multiplexed to serve the function of two neurons. Here, we again see that such a network can achieve ideal accuracies, further corroborating the need for including hardware in training for edge intelligence applications (Figure **6-9**).

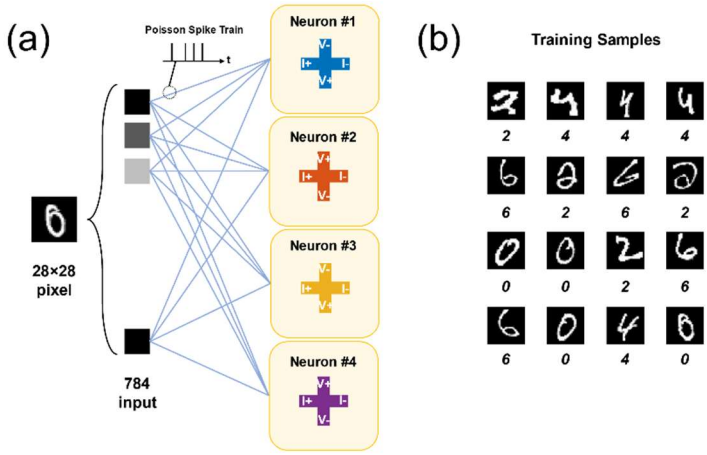

Figure **6-7**: Training for the Hardware-in-loop scheme is shown. (a) The network architecture used for training in the hardware-in-loop scheme is shown. (b) Training images used for hardware-in-loop learning is shown. 4 classes ("0", "2", "4" and "6") with 4 images of each class were used.

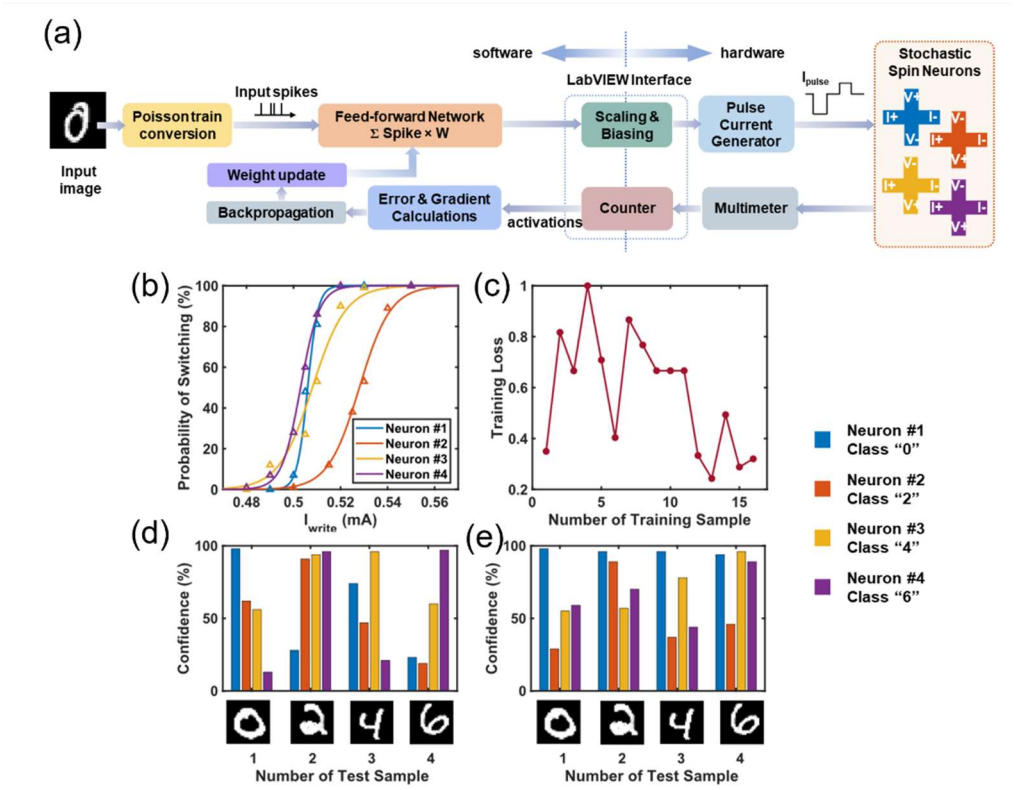

Figure **6-8**: Hardware-in-loop learning for the SOT Stochastic Spin Neuron Devices is shown. (a) The schematic of the hardware-in-loop (HIL) learning setup used for the experiments is shown. (b) Neuronal dynamics of the four-spin stochastic neuron used for HIL training is shown. As can be seen, the four neurons have differing biases and operation windows. Note, each neuron represents a different class and is represented with a different color. (c) The training loss of the network over the 16 training images from (4 each from four classes, '0', '2', '4', '6') the MNIST Handwritten Digit Dataset is shown. The training loss becomes low indicating the network is gradually learning. (d) The testing results are shown on 4 test images from the MNIST dataset when HIL is used. The network is able to successfully classify 3 out of the 4 digits. The confidence is calculated from the normalized inputs to the neurons from the network and the neuron with the highest confidence is assigned that corresponding class. (e) The testing results on 4 test images from the MNIST dataset when HIL is not used are shown. The software-trained network is only able to classify one image correctly, highlighting the need for including the hardware in training.

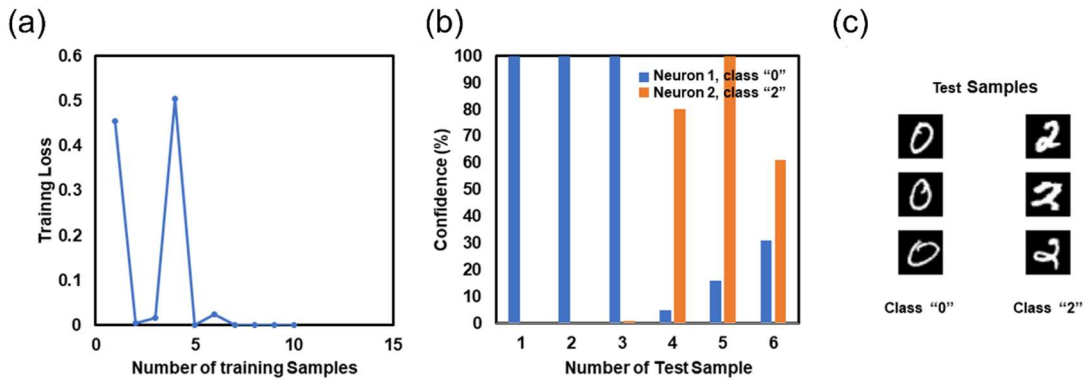

Figure **6-9**: The results of training a single device with time-multiplexing to emulate 2 neurons through hardware-in-loop scheme are shown. (a) The training loss of the network over the training images is shown. (b) The testing results on 6 test images from the MNIST dataset are shown. The network achieves an accuracy of 100%. (c) Test samples used for the inference of the two classes ("0" and "2") are shown.

4. Conclusion

This work presented a detailed analysis of SOT-based spintronic stochastic neurons – how their dynamics evolves with device dimension and how device-to-device variations impact performance on the network level for deep learning applications. The importance of hardware-software co-design for neuromorphic hardware is demonstrated via the interplay between accuracy, and robustness for 36 devices of 9 varying sizes, ranging from 5μm to 0.3μm. In total, the co-design analysis is corroborated by conducting over 20,000 different measurement steps for the various characterizations. How these variations can be compensated by in situ learning through a hardware-in-loop learning scheme is demonstrated for a model handwritten digit recognition problem. This shows how edge intelligence applications can be enabled by scaled hardware performing training natively.

Chapter 7

Conclusions and Future Work

1. Conclusions

As discussed in the previous chapters, spintronic devices can be a promising option for building a post-CMOS neuromorphic computing hardware platform. Under this motivation, this dissertation has contributed to the field in the following points:

In chapter 3, a 2-terminal ME-based scaled MTJ device design is proposed as stochastic neuron. The device design enables independent control of lifetime of the two states of MTJ (i.e. τ_{P} and τ_{AP}). Conditions for realizing independent control were also explored. The proposed device design was used to build SNNs (784×10 and $784 \times 400 \times 10$). The network achieved a test accuracy of 90.88% for 784×10 network and 97.41% for $784 \times 400 \times 10$ network with only 2/3 spikes used in each layer for inference. It was observed that the spike sparsity is reduced by $1.6 \times$ for hidden layer and $3.77 \times$ for output layer in $784 \times 400 \times 10$ network compared to its rate-encoded counterpart.

In chapter 4, a Bayesian Neural Network hardware accelerator design based on spintronic devices was proposed. DW-MTJs were used as programmable synapses and scaled MTJs, of which the intrinsic stochasticity due to thermal noise is utilized, were used to build the Gaussian random number generator. The network achieved a test accuracy of 96.98% with no device non-idealities and 96.74% with 10% variation in the programmed conductance values. This indicates the proposed network is able to compensate for the programming error due to device non-idealities. Compared to the CMOS-based network, the proposed design achieves a $24 \times$ energy efficiency.

In chapter 5, a solution for write sneak path problem in neuromorphic nonvolatile cross-point arrays based on VCMA-MTJ devices was proposed. The pulsewidth difference between VMCA- and STT-induced switching mechanisms enables sharp difference in switching probability

(over 97% for VCMA-induced switching and ≈ 0% for STT switching). This offers an energy efficient solution for sneak path problem. The condition for applied voltage to ensure magnetization trajectory robustness was also explored. This is crucial to a high switching probability.

In chapter 6, an illustration of hardware-in-loop training was demonstrated. The devices are Hall bar structures of different sizes (5μm, 2.5μm, 2μm, 1.5μm, 1μm, 0.9μm, 0.7μm, 0.5μm, 0.3μm) fabricated on Si/SiO2(300nm)/Ta (5nm)/CoFeB (1nm)/MgO (2.5nm)/Ta (2.5nm /5nm) material stack. The relation between device characteristics and sizes was explored. It was observed that the bias current amplitude and size of programming window reduce with bar size. The devices are used to build networks for training and inferencing purposes (on MNIST dataset). It was found that the network is able to successfully classify 3 out of the total 4 test images, which is better compared to the software network where HIL is not included.

2. Discussions

2.1. Device Temperature

The proof-of-principle simulations in this dissertation are conducted for room temperature cases. However, when current passes through devices, there is Joule heat generated during the process and device temperature rises consequently. The change in temperature affects the switching behavior of devices, as the thermal noise plays an important role [238]. For a further study, a comprehensive model for temperature change caused by current through the devices is required and in this way the temperature can be included in simulations in a real-time manner, which provides a more realistic result.

2.2. Crossbar Array Scale

The work in this dissertation mainly focuses on device-level design and improvement. For this reason, circuit-level analysis was not considered in its entirety. However, as reported in references, in large-scale crossbar arrays, which is the typical situation for neuromorphic computation for complex problems, wire resistance becomes non-negligible and distorts the output signals [239], [240]. As a result, the read (write) margin, which is the available current/voltage operating window for an accurate read (write) operation is affected and the power consumption increases. Many approaches have been proposed on both hardware and algorithm sides to compensate for the problem [240]–[242]. Further studies on circuit level implications for novel device designs explored in this work need to be considered.

3. Future Work

Future work can be at 3 levels: 1) The bottom stack involves the material level, which means to work more closely with materials research community to explore novel device physics that can be beneficial to neuromorphic computing. 2) The second level can be the device-network level. This dissertation stays in this level, which studies how discovered physics can be beneficial to the network performance and what adjustment is required in algorithms to comply with the device physics. 3) The highest level is the circuit and architecture level. This level discusses the peripheral circuits and system organization that are required for the device network to work with the remaining units of the computing system.

Reference

[1] G. E. Moore, "Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff.," *IEEE Solid-State Circuits Society Newsletter*, vol. 11, no. 3, pp. 33–35, Feb. 2009, doi: 10.1109/n-ssc.2006.4785860.

[2] I. Ferain, C. A. Colinge, and J. P. Colinge, "Multigate transistors as the future of classical metal-oxide-semiconductor field-effect transistors," *Nature*, vol. 479, no. 7373. pp. 310–316, Nov. 17, 2011. doi: 10.1038/nature10676.

[3] R. Quhe *et al.*, "Sub-10 nm two-dimensional transistors: Theory and experiment," *Physics Reports*, vol. 938. Elsevier B.V., pp. 1–72, Nov. 25, 2021. doi: 10.1016/j.physrep.2021.07.006.

[4] A. D. Franklin, "Nanomaterials in transistors: From high-performance to thin-film applications," *Science*, vol. 349, no. 6249. American Association for the Advancement of Science, Aug. 14, 2015. doi: 10.1126/science.aab2750.

[5] J. Robertson, "High dielectric constant oxides," *EPJ Applied Physics*, vol. 28, no. 3. pp. 265–291, Dec. 2004. doi: 10.1051/epjap:2004206.

[6] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671. Nature Publishing Group, pp. 195–202, Sep. 13, 2017. doi: 10.1038/nature23474.

[7] S. S. Gill *et al.*, "Quantum computing: A taxonomy, systematic review and future directions," *Softw Pract Exp*, vol. 52, no. 1, pp. 66–114, Jan. 2022, doi: 10.1002/spe.3039.

[8] W. Maass, "Noise as a resource for computation and learning in networks of spiking neurons," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 860–880, 2014, doi: 10.1109/JPROC.2014.2310593.

[9] Y. Liu, S. Liu, Y. Wang, F. Lombardi, and J. Han, "A Survey of Stochastic Computing Neural Networks for Machine Learning Applications," *IEEE Trans Neural Netw Learn Syst*, vol. 32, no. 7, pp. 2809–2824, Jul. 2021, doi: 10.1109/TNNLS.2020.3009047.

[10] N. Guo *et al.*, "Energy-Efficient Hybrid Analog/Digital Approximate Computation in Continuous Time," *IEEE J Solid-State Circuits*, vol. 51, no. 7, pp. 1514–1524, Jul. 2016, doi: 10.1109/JSSC.2016.2543729.

[11] C. Mead, "Neuromorphic Electronic Systems," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629–1636, 1990, doi: 10.1109/5.58356.

[12] R. Haeb-Umbach *et al.*, "Speech Processing for Digital Home Assistants: Combining signal processing with deep-learning techniques," *IEEE Signal Processing Magazine*, vol. 36, no. 6. Institute of Electrical and Electronics Engineers Inc., pp. 111–124, Nov. 01, 2019. doi: 10.1109/MSP.2019.2918706.

[13] H. Purwins, B. Li, T. Virtanen, J. Schlüter, S. Y. Chang, and T. Sainath, "Deep Learning for Audio Signal Processing," *IEEE Journal on Selected Topics in Signal Processing*, vol. 13, no. 2, pp. 206–219, May 2019, doi: 10.1109/JSTSP.2019.2908700.

[14] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "SlowFast Networks for Video Recognition." [Online]. Available: https://github.com/

[15] A. Roy, J. Sun, R. Mahoney, L. Alonzi, S. Adams, and P. Beling, "Deep learning detecting fraud in credit card transactions," in *2018 Systems and Information Engineering Design Symposium, SIEDS 2018*, Institute of Electrical and Electronics Engineers Inc., Jun. 2018, pp. 129–134. doi: 10.1109/SIEDS.2018.8374722.

[16] D. Silver *et al.*, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017, doi: 10.1038/nature24270.

[17] D. Ivanov, A. Chezhegov, M. Kiselev, A. Grunin, and D. Larionov, "Neuromorphic artificial intelligence systems," *Front Neurosci*, vol. 16, 2022, doi: 10.3389/fnins.2022.959626.

[18] M. Horowitz, "1.1 Computing's energy problem (and what we can do about it)," in *Digest of Technical Papers - IEEE International Solid-State Circuits Conference*, 2014, pp. 10–14. doi: 10.1109/ISSCC.2014.6757323.

[19] A. Čolaković and M. Hadžialić, "Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues," *Computer Networks*, vol. 144. Elsevier B.V., pp. 17–39, Oct. 24, 2018. doi: 10.1016/j.comnet.2018.07.017.

[20] S. Seneviratne *et al.*, "A Survey of Wearable Devices and Challenges," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 4. Institute of Electrical and Electronics Engineers Inc., pp. 2573–2620, Oct. 01, 2017. doi: 10.1109/COMST.2017.2731979.

[21] N. Verma *et al.*, "In-memorycomputing advances and prospects," *IEEE Solid-State Circuits Magazine*, vol. 11, no. 3. Institute of Electrical and Electronics Engineers Inc., pp. 43–55, Jun. 01, 2019. doi: 10.1109/MSSC.2019.2922889.

[22] T. Gokmen and Y. Vlasov, "Acceleration of deep neural network training with resistive cross-point devices: Design considerations," *Front Neurosci*, vol. 10, no. JUL, 2016, doi: 10.3389/fnins.2016.00333.

[23] C. Mayr, S. Hoeppner, and S. Furber, "SpiNNaker 2: A 10 Million Core Processor System for Brain Simulation and Machine Learning," Nov. 2019, [Online]. Available: http://arxiv.org/abs/1911.02385

[24] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The SpiNNaker project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, 2014, doi: 10.1109/JPROC.2014.2304638.

[25] M. Davies *et al.*, "Loihi: A Neuromorphic Manycore Processor with On-Chip Learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan. 2018, doi: 10.1109/MM.2018.112130359.

[26] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. USA: Cambridge University Press, 2014.

[27] A. Amir *et al.*, "A Low Power, Fully Event-Based Gesture Recognition System."

[28] A. A. Faisal, L. P. J. Selen, and D. M. Wolpert, "Noise in the nervous system," *Nature Reviews Neuroscience*, vol. 9, no. 4. pp. 292–303, Apr. 2008. doi: 10.1038/nrn2258.

[29] M. Al-Shedivat, R. Naous, G. Cauwenberghs, and K. N. Salama, "Memristors empower spiking neurons with stochasticity," *IEEE J Emerg Sel Top Circuits Syst*, vol. 5, no. 2, pp. 242–253, Jun. 2015, doi: 10.1109/JETCAS.2015.2435512.

[30] E. Neftci, "Stochastic neuromorphic learning machines for weakly labeled data," in *2016 IEEE 34th International Conference on Computer Design (ICCD)*, 2016, pp. 670–673. doi: 10.1109/ICCD.2016.7753355.

[31] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002. doi: 10.1017/CBO9780511815706.

[32] W. Maass, "Networks of Spiking Neurons: The Third Generation of Neural Network Models," 1997.

[33] E. D. Adrian and Y. Zotterman, "The impulses produced by sensory nerve endings: Part 3. Impulses set up by Touch and Pressure," *J Physiol*, vol. 61, no. 4, pp. 465–483, Aug. 1926, doi: 10.1113/jphysiol.1926.sp002308.

[34] T. Gollisch and M. Meister, "Rapid Neural Coding in the Retina with Relative Spike Latencies," 2008. [Online]. Available: https://www.science.org

[35] R. S. Johansson and I. Birznieks, "First spikes in ensembles of human tactile afferents code complex spatial fingertip events," *Nat Neurosci*, vol. 7, no. 2, pp. 170–177, Feb. 2004, doi: 10.1038/nn1177.

[36] J. Gautrais and S. Thorpe, "Rate coding versus temporal order coding: a theoretical approach," *Biosystems*, vol. 48, no. 1, pp. 57–65, 1998, doi: https://doi.org/10.1016/S0303-2647(98)00050-1.

[37] Y. Cao, Y. Chen, and D. Khosla, "Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition," *Int J Comput Vis*, vol. 113, no. 1, pp. 54–66, May 2015, doi: 10.1007/s11263-014-0788-3.

[38] E. Hunsberger and C. Eliasmith, "Spiking Deep Networks with LIF Neurons," Oct. 2015, [Online]. Available: http://arxiv.org/abs/1510.08829

[39] P. U. Diehl, D. Neil, J. Binas, M. Cook, S. C. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *Proceedings of the International Joint Conference on Neural Networks*, Institute of Electrical and Electronics Engineers Inc., Sep. 2015. doi: 10.1109/IJCNN.2015.7280696.

[40] H. Zheng, Y. Wu, L. Deng, Y. Hu, and G. Li, "Going Deeper With Directly-Trained Larger Spiking Neural Networks," 2021. [Online]. Available: www.aaai.org

[41] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, and Y. Tian, "Incorporating Learnable Membrane Time Constant to Enhance Learning of Spiking Neural Networks." [Online]. Available: https://github.com/fangw

[42] R. Kempter, W. Gerstner, and J. Leo Van Hemmen, "Hebbian learning and spiking neurons."

[43] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Front Comput Neurosci*, vol. 9, no. AUGUST, Aug. 2015, doi: 10.3389/fncom.2015.00099.

[44] G. Srinivasan and K. Roy, "ReStoCNet: Residual Stochastic Binary Convolutional Spiking Neural Network for Memory-Efficient Neuromorphic Computing," *Front Neurosci*, vol. 13, 2019, doi: 10.3389/fnins.2019.00189.

[45] Y. Xing, G. Di Caterina, and J. Soraghan, "A New Spiking Convolutional Recurrent Neural Network (SCRNN) With Applications to Event-Based Hand Gesture Recognition," *Front Neurosci*, vol. 14, 2020, doi: 10.3389/fnins.2020.590164.

[46] S. B. Shrestha and G. Orchard, "SLAYER: Spike Layer Error Reassignment in Time," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/file/82f2b308c3b01637c607ce05f52a2fed-Paper.pdf

[47] J. J. Hopfield and C. D. Brody, "What is a moment? Transient synchrony as a collective mechanism for spatiotemporal integration," *Proceedings of the National Academy of Sciences*, vol. 98, no. 3, pp. 1282–1287, 2001, doi: 10.1073/pnas.98.3.1282.

[48] J. J. Hopfield and C. D. Brody, "What is a moment? 'Cortical' sensory integration over a brief interval," *Proceedings of the National Academy of Sciences*, vol. 97, no. 25, pp. 13919–13924, 2000, doi: 10.1073/pnas.250483697.

[49] I. Kiral-Kornek *et al.*, "TrueNorth-enabled real-time classification of EEG data for brain-computer interfacing," in *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2017, pp. 1648–1651. doi: 10.1109/EMBC.2017.8037156.

[50] Z. Yan, J. Zhou, and W.-F. Wong, "Energy efficient ECG classification with spiking neural network," *Biomed Signal Process Control*, vol. 63, p. 102170, 2021, doi: https://doi.org/10.1016/j.bspc.2020.102170.

[51] J. L. Hennessy and D. A. Patterson, *Computer Architecture A Quantitative Approach*, 5th ed. Amsterdam: Morgan Kaufmann, 2012.

[52] N. P. Jouppi, C. Young, N. Patil, and D. Patterson, "A domain-specific architecture for deep neural networks," *Commun ACM*, vol. 61, no. 9, pp. 50–59, Sep. 2018, doi: 10.1145/3154484.

[53] M. Mahowald, “The Silicon Retina,” in *An Analog VLSI System for Stereoscopic Vision*, Boston, MA: Springer US, 1994, pp. 4–65. doi: 10.1007/978-1-4615-2724-4_2.

[54] M. Mahowald and R. Douglas, “A silicon neuron,” *Nature*, vol. 354, no. 6354, pp. 515–518, 1991, doi: 10.1038/354515a0.

[55] P. A. Merolla *et al.*, “A million spiking-neuron integrated circuit with a scalable communication network and interface,” *Science (1979)*, vol. 345, no. 6197, pp. 668–673, 2014, doi: 10.1126/science.1254642.

[56] J. Pei *et al.*, “Towards artificial general intelligence with hybrid Tianjic chip architecture,” *Nature*, vol. 572, no. 7767, pp. 106–111, Aug. 2019, doi: 10.1038/s41586-019-1424-8.

[57] S. Höppner *et al.*, “The SpiNNaker 2 Processing Element Architecture for Hybrid Digital Neuromorphic Computing,” Mar. 2021, [Online]. Available: http://arxiv.org/abs/2103.08392

[58] A. Grübl, S. Billaudelle, B. Cramer, V. Karasenko, and J. Schemmel, “Verification and Design Methods for the BrainScaleS Neuromorphic Hardware System,” *J Signal Process Syst*, vol. 92, no. 11, pp. 1277–1292, Nov. 2020, doi: 10.1007/s11265-020-01558-7.

[59] O. Moreira *et al.*, “NeuronFlow: A Hybrid Neuromorphic – Dataflow Processor Architecture for AI Workloads,” in *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2020, pp. 1–5. doi: 10.1109/AICAS48895.2020.9073999.

[60] A. Sengupta, M. Parsa, B. Han, and K. Roy, “Probabilistic Deep Spiking Neural Systems Enabled by Magnetic Tunnel Junction,” *IEEE Trans Electron Devices*, vol. 63, no. 7, pp. 2963–2970, Jul. 2016, doi: 10.1109/TED.2016.2568762.

[61] G. Palma, M. Suri, D. Querlioz, E. Vianello, and B. De Salvo, “Stochastic neuron design using conductive bridge RAM,” in *2013 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, 2013, pp. 95–100. doi: 10.1109/NanoArch.2013.6623051.

[62] D. Kuzum, R. G. D. Jeyasingh, B. Lee, and H.-S. P. Wong, “Nanoelectronic Programmable Synapses Based on Phase Change Materials for Brain-Inspired Computing,” *Nano Lett*, vol. 12, no. 5, pp. 2179–2186, May 2012, doi: 10.1021/nl201040y.

[63] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, “Nanoscale Memristor Device as Synapse in Neuromorphic Systems,” *Nano Lett*, vol. 10, no. 4, pp. 1297–1301, Apr. 2010, doi: 10.1021/nl904092h.

[64] T.-Y. Liu *et al.*, “A 130.7mm2 2-layer 32Gb ReRAM memory device in 24nm technology,” in *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, 2013, pp. 210–211. doi: 10.1109/ISSCC.2013.6487703.

[65] D. Kau *et al.*, “A stackable cross point Phase Change Memory,” in *2009 IEEE International Electron Devices Meeting (IEDM)*, 2009, pp. 1–4. doi: 10.1109/IEDM.2009.5424263.

[66] A. Prakash, D. Deleruyelle, J. Song, M. Bocquet, and H. Hwang, “Resistance controllability and variability improvement in a TaOx-based resistive memory for multilevel storage application,” *Appl Phys Lett*, vol. 106, no. 23, Jun. 2015, doi: 10.1063/1.4922446.

[67] Y.-T. Su *et al.*, “A Method to Reduce Forming Voltage Without Degrading Device Performance in Hafnium Oxide-Based 1T1R Resistive Random Access Memory,” *IEEE Journal of the Electron Devices Society*, vol. 6, pp. 341–345, 2018, doi: 10.1109/JEDS.2018.2805285.

[68] Y. J. Huang, T. H. Shen, L. H. Lee, C. Y. Wen, and S. C. Lee, “Low-power resistive random access memory by confining the formation of conducting filaments,” *AIP Adv*, vol. 6, no. 6, Jun. 2016, doi: 10.1063/1.4954974.

[69] S. R. Nandakumar, M. Le Gallo, I. Boybat, B. Rajendran, A. Sebastian, and E. Eleftheriou, “A phase-change memory model for neuromorphic computing,” *J Appl Phys*, vol. 124, no. 15, Oct. 2018, doi: 10.1063/1.5042408.

[70] A. Sengupta and K. Roy, “Encoding neural and synaptic functionalities in electron spin: A pathway to efficient neuromorphic computing,” *Applied Physics Reviews*, vol. 4, no. 4. American Institute of Physics Inc., Dec. 01, 2017. doi: 10.1063/1.5012763.

[71] A. Sengupta, P. Panda, P. Wijesinghe, Y. Kim, and K. Roy, “Magnetic tunnel junction mimics stochastic cortical spiking neurons,” *Sci Rep*, vol. 6, Jul. 2016, doi: 10.1038/srep30039.

[72] A. Sengupta and K. Roy, “A Vision for All-Spin Neural Networks: A Device to System Perspective,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 12, pp. 2267–2277, Dec. 2016, doi: 10.1109/TCSI.2016.2615312.

[73] A. Sengupta, A. Banerjee, and K. Roy, “Hybrid Spintronic-CMOS Spiking Neural Network with On-Chip Learning: Devices, Circuits, and Systems,” *Phys Rev Appl*, vol. 6, no. 6, Dec. 2016, doi: 10.1103/PhysRevApplied.6.064003.

[74] G. Srinivasan, A. Sengupta, and K. Roy, “Magnetic Tunnel Junction Based Long-Term Short-Term Stochastic Synapse for a Spiking Neural Network with On-Chip STDP Learning,” *Sci Rep*, vol. 6, Jul. 2016, doi: 10.1038/srep29545.

[75] M. Sharad, C. Augustine, G. Panagopoulos, and K. Roy, “Spin-based neuron model with domain-wall magnets as synapse,” *IEEE Trans Nanotechnol*, vol. 11, no. 4, pp. 843–853, 2012, doi: 10.1109/TNANO.2012.2202125.

[76] M. Sharad, D. Fan, and K. Roy, “Spin-neurons: A possible path to energy-efficient neuromorphic computers,” *J Appl Phys*, vol. 114, no. 23, Dec. 2013, doi: 10.1063/1.4838096.

[77] A. Sengupta, S. H. Choday, Y. Kim, and K. Roy, “Spin orbit torque based electronic neuron,” *Appl Phys Lett*, vol. 106, no. 14, p. 143701, Apr. 2015, doi: 10.1063/1.4917011.

[78] E. Goan and C. Fookes, “Bayesian Neural Networks: An Introduction and Survey,” in *Case Studies in Applied Bayesian Data Science*, Springer International Publishing, 2020, pp. 45–87. doi: 10.1007/978-3-030-42553-1_3.

[79] Tishby, Levin, and Solla, “Consistent inference of probabilities in layered networks: predictions and generalizations,” in *International 1989 Joint Conference on Neural Networks*, 1989, pp. 403–409 vol.2. doi: 10.1109/IJCNN.1989.118274.

[80] D. J. C. MacKay, “Bayesian Interpolation,” *Neural Comput*, vol. 4, no. 3, pp. 415–447, May 1992, doi: 10.1162/neco.1992.4.3.415.

[81] J. Singh and R. Banerjee, “A Study on Single and Multi-layer Perceptron Neural Network,” in *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, 2019, pp. 35–40. doi: 10.1109/ICCMC.2019.8819775.

[82] S. Jeyanthi and M. Subadra, “Implementation of single neuron using various activation functions with FPGA,” in *Proceedings of 2014 IEEE International Conference on Advanced Communication, Control and Computing Technologies, ICACCCT 2014*, Institute of Electrical and Electronics Engineers Inc., Jan. 2015, pp. 1126–1131. doi: 10.1109/ICACCCT.2014.7019273.

[83] I. Del Campo, R. Finker, J. Echanobe, and K. Basterretxea, “Controlled accuracy approximation of sigmoid function for efficient FPGA-based implementation of artificial neurons,” *Electron Lett*, vol. 49, no. 25, pp. 1598–1600, Dec. 2013, doi: 10.1049/el.2013.3098.

[84] M. Bañuelos Saucedo *et al.*, “Implementation of a Neuron Using FPGAS,” *Journal of Applied Research and Technology*, vol. 1, pp. 248–255, May 2003, doi: 10.22201/icat.16656423.2003.1.03.611.

[85] X. Glorot, A. Bordes, and Y. Bengio, “Deep Sparse Rectifier Neural Networks,” in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, G. Gordon, D. Dunson, and M. Dudík, Eds., in Proceedings of Machine Learning Research, vol. 15. Fort Lauderdale, FL, USA: PMLR, May 2011, pp. 315–323. [Online]. Available: https://proceedings.mlr.press/v15/glorot11a.html

[86] D. Baptista and F. Morgado-Dias, “Low-resource hardware implementation of the hyperbolic tangent for artificial neural networks,” *Neural Comput Appl*, vol. 23, no. 3–4, pp. 601–607, Sep. 2013, doi: 10.1007/s00521-013-1407-x.

[87] A. L. Hodgkin and A. F. Huxley, “A quantitative description of membrane current and its application to conduction and excitation in nerve,” *J Physiol*, vol. 117, no. 4, pp. 500–544, 1952, doi: https://doi.org/10.1113/jphysiol.1952.sp004764.

[88] F. Castaños and A. Franci, “Implementing robust neuromodulation in neuromorphic circuits,” *Neurocomputing*, vol. 233, pp. 3–13, Apr. 2017, doi: 10.1016/j.neucom.2016.08.099.

[89] Q. Ma, M. R. Haider, V. L. Shrestha, and Y. Massoud, “Bursting Hodgkin-Huxley model-based ultra-low-power neuromimetic silicon neuron,” in *Analog Integrated Circuits and Signal Processing*, Oct. 2012, pp. 329–337. doi: 10.1007/s10470-012-9888-6.

[90] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *Bull Math Biophys*, vol. 5, no. 4, pp. 115–133, 1943, doi: 10.1007/BF02478259.

[91] D. Querlioz, O. Bichler, P. Dollfus, and C. Gamrat, “Immunity to device variations in a spiking neural network with memristive nanodevices,” *IEEE Trans Nanotechnol*, vol. 12, no. 3, pp. 288–295, 2013, doi: 10.1109/TNANO.2013.2250995.

[92] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, “Going Deeper in Spiking Neural Networks: VGG and Residual Architectures,” *Front Neurosci*, vol. 13, Mar. 2019, doi: 10.3389/fnins.2019.00095.

[93] E. Wallace, M. Benayoun, W. van Drongelen, and J. D. Cowan, “Emergent oscillations in networks of stochastic spiking neurons,” *PLoS One*, vol. 6, no. 5, 2011, doi: 10.1371/journal.pone.0014804.

[94] C. D. Schuman *et al.*, “A Survey of Neuromorphic Computing and Neural Networks in Hardware,” May 2017, [Online]. Available: http://arxiv.org/abs/1705.06963

[95] G. Bi and M. Poo, “Synaptic Modification by Correlated Activity: Hebb’s Postulate Revisited,” *Annu Rev Neurosci*, vol. 24, no. 1, pp. 139–166, 2001, doi: 10.1146/annurev.neuro.24.1.139.

[96] R. Kempter, W. Gerstner, and J. L. van Hemmen, “Hebbian learning and spiking neurons,” *Phys Rev E*, vol. 59, no. 4, pp. 4498–4514, Apr. 1999, doi: 10.1103/PhysRevE.59.4498.

[97] M. Kiselev, “Rate coding vs. temporal coding - Is optimum between?,” in *Proceedings of the International Joint Conference on Neural Networks*, Institute of Electrical and Electronics Engineers Inc., Oct. 2016, pp. 1355–1359. doi: 10.1109/IJCNN.2016.7727355.

[98] D.-A. Nguyen, X.-T. Tran, and F. Iacopi, “A Review of Algorithms and Hardware Implementations for Spiking Neural Networks,” *Journal of Low Power Electronics and Applications*, vol. 11, no. 2, 2021, doi: 10.3390/jlpea11020023.

[99] S. Denève and C. K. Machens, “Efficient codes and balanced networks,” *Nature Neuroscience*, vol. 19, no. 3. Nature Publishing Group, pp. 375–382, Feb. 23, 2016. doi: 10.1038/nn.4243.

[100] C. Eliasmith and C. H. Anderson, *Neural Engineering (Computational Neuroscience Series): Computational, Representation, and Dynamics in Neurobiological Systems*. Cambridge, MA, USA: MIT Press, 2002.

[101] H. Mostafa, B. U. Pedroni, S. Sheik, and G. Cauwenberghs, "Fast classification using sparsely active spiking networks," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017, pp. 1–4. doi: 10.1109/ISCAS.2017.8050527.

[102] B. Rueckauer and S.-C. Liu, "Conversion of analog to spiking neural networks using sparse temporal coding," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018, pp. 1–5. doi: 10.1109/ISCAS.2018.8351295.

[103] A. M. M. Oswald, B. Doiron, and L. Maler, "Interval coding. I. Burst interspike intervals as indicators of stimulus intensity," *J Neurophysiol*, vol. 97, no. 4, pp. 2731–2743, Apr. 2007, doi: 10.1152/jn.00987.2006.

[104] D. Auge, J. Hille, E. Mueller, and A. Knoll, "A Survey of Encoding Techniques for Signal Processing in Spiking Neural Networks," *Neural Processing Letters*, vol. 53, no. 6. Springer, pp. 4693–4710, Dec. 01, 2021. doi: 10.1007/s11063-021-10562-2.

[105] B. Rueckauer, I. A. Lungu, Y. Hu, M. Pfeiffer, and S. C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Front Neurosci*, vol. 11, no. DEC, Dec. 2017, doi: 10.3389/fnins.2017.00682.

[106] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Front Neurosci*, vol. 10, no. NOV, 2016, doi: 10.3389/fnins.2016.00508.

[107] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-based optimization to spiking neural networks," *IEEE Signal Process Mag*, vol. 36, no. 6, pp. 51–63, Nov. 2019, doi: 10.1109/MSP.2019.2931595.

[108] H. Mostafa, "Supervised Learning Based on Temporal Coding in Spiking Neural Networks," *IEEE Trans Neural Netw Learn Syst*, vol. 29, no. 7, pp. 3227–3235, 2018, doi: 10.1109/TNNLS.2017.2726060.

[109] N. Zheng and P. Mazumder, "Online Supervised Learning for Hardware-Based Multilayer Spiking Neural Networks Through the Modulation of Weight-Dependent Spike-Timing-Dependent Plasticity," *IEEE Trans Neural Netw Learn Syst*, vol. 29, no. 9, pp. 4287–4302, 2018, doi: 10.1109/TNNLS.2017.2761335.

[110] H. Markram, J. Lübke, M. Frotscher, and B. Sakmann, "Regulation of Synaptic Efficacy by Coincidence of Postsynaptic APs and EPSPs," *Science (1979)*, vol. 275, no. 5297, pp. 213–215, 1997, doi: 10.1126/science.275.5297.213.

[111] Y. Dan and M.-M. Poo, "Spike Timing-Dependent Plasticity: From Synapse to Perception," *Physiol Rev*, vol. 86, no. 3, pp. 1033–1048, 2006, doi: 10.1152/physrev.00030.2005.

[112] P. Ferré, F. Mamalet, and S. J. Thorpe, "Unsupervised feature learning with winner-takes-all based STDP," *Front Comput Neurosci*, vol. 12, Apr. 2018, doi: 10.3389/fncom.2018.00024.

[113] C. Lee, G. Srinivasan, P. Panda, and K. Roy, "Deep Spiking Convolutional Neural Network Trained with Unsupervised Spike-Timing-Dependent Plasticity," *IEEE Trans Cogn Dev Syst*, vol. 11, no. 3, pp. 384–394, Sep. 2019, doi: 10.1109/TCDS.2018.2833071.

[114] A. Tavanaei, T. Masquelier, and A. S. Maida, "Acquisition of visual features through probabilistic spike-timing-dependent plasticity," in *2016 International Joint Conference on Neural Networks (IJCNN)*, 2016, pp. 307–314. doi: 10.1109/IJCNN.2016.7727213.

[115] R. Houthooft *et al.*, "VIME: Variational Information Maximizing Exploration," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2016/file/abd815286ba1007abfbb8415b83ae2cf-Paper.pdf

[116] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan, "An Introduction to MCMC for Machine Learning," *Mach Learn*, vol. 50, no. 1, pp. 5–43, 2003, doi: 10.1023/A:1020281327116.

[117] Z. Ghahramani and M. Beal, "Propagation Algorithms for Variational Bayesian Learning," in *Advances in Neural Information Processing Systems*, T. Leen, T. Dietterich, and V. Tresp, Eds., MIT Press, 2000. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2000/file/77369e37b2aa1404f416275183ab055f-Paper.pdf

[118] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight Uncertainty in Neural Network," in *Proceedings of the 32nd International Conference on Machine Learning*, F. Bach and D. Blei, Eds., in Proceedings of Machine Learning Research, vol. 37. Lille, France: PMLR, May 2015, pp. 1613–1622. [Online]. Available: https://proceedings.mlr.press/v37/blundell15.html

[119] M. Julliere, "Tunneling between ferromagnetic films," *Phys Lett A*, vol. 54, no. 3, pp. 225–226, 1975, doi: https://doi.org/10.1016/0375-9601(75)90174-7.

[120] A. Sengupta, Y. Shim, and K. Roy, "Proposal for an all-spin artificial neural network: Emulating neural and synaptic functionalities through domain wall motion in ferromagnets," *IEEE Trans Biomed Circuits Syst*, vol. 10, no. 6, pp. 1152–1160, Dec. 2016, doi: 10.1109/TBCAS.2016.2525823.

[121] S. Emori *et al.*, "Spin Hall torque magnetometry of Dzyaloshinskii domain walls," *Phys Rev B Condens Matter Mater Phys*, vol. 90, no. 18, Nov. 2014, doi: 10.1103/PhysRevB.90.184427.

[122] S. Emori, U. Bauer, S. M. Ahn, E. Martinez, and G. S. D. Beach, "Current-driven dynamics of chiral ferromagnetic domain walls," *Nat Mater*, vol. 12, no. 7, pp. 611–616, Jul. 2013, doi: 10.1038/nmat3675.

[123] S. Lequeux *et al.*, "A magnetic synapse: Multilevel spin-torque memristor with perpendicular anisotropy," *Sci Rep*, vol. 6, Aug. 2016, doi: 10.1038/srep31510.

[124] J. C. Slonczewski, “Conductance and exchange coupling of two ferromagnets separated by a tunneling barrier,” *Phys Rev B*, vol. 39, no. 10, pp. 6995–7002, Apr. 1989, doi: 10.1103/PhysRevB.39.6995.

[125] C. M. Liyanagedera, A. Sengupta, A. Jaiswal, and K. Roy, “Stochastic Spiking Neural Networks Enabled by Magnetic Tunnel Junctions: From Nontelegraphic to Telegraphic Switching Regimes,” *Phys Rev Appl*, vol. 8, no. 6, Dec. 2017, doi: 10.1103/PhysRevApplied.8.064017.

[126] W. Scholz, T. Schrefl, and J. Fidler, “Micromagnetic simulation of thermally activated switching in fine particles,” *J Magn Magn Mater*, vol. 233, no. 3, pp. 296–304, 2001, doi: https://doi.org/10.1016/S0304-8853(01)00032-4.

[127] A. Mondal and A. Srivastava, “In Situ Stochastic Training of MTJ Crossbars With Machine Learning Algorithms,” *J. Emerg. Technol. Comput. Syst.*, vol. 15, no. 2, Mar. 2019, doi: 10.1145/3309880.

[128] T. Sharma, C. Wang, A. Agrawal, and K. Roy, “Enabling Robust SOT-MTJ Crossbars for Machine Learning using Sparsity-Aware Device-Circuit Co-design,” in *2021 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2021, pp. 1–6. doi: 10.1109/ISLPED52811.2021.9502492.

[129] I. M. Miron *et al.*, “Perpendicular switching of a single ferromagnetic layer induced by in-plane current injection,” *Nature*, vol. 476, no. 7359, pp. 189–193, 2011, doi: 10.1038/nature10309.

[130] C. Song *et al.*, “Spin-orbit torques: Materials, mechanisms, performances, and potential applications,” *Prog Mater Sci*, vol. 118, p. 100761, 2021, doi: https://doi.org/10.1016/j.pmatsci.2020.100761.

[131] J. E. Hirsch, “Spin Hall Effect,” *Phys Rev Lett*, vol. 83, no. 9, pp. 1834–1837, Aug. 1999, doi: 10.1103/PhysRevLett.83.1834.

[132] J. Sinova, S. O. Valenzuela, J. Wunderlich, C. H. Back, and T. Jungwirth, “Spin Hall effects,” *Rev Mod Phys*, vol. 87, no. 4, pp. 1213–1260, Oct. 2015, doi: 10.1103/RevModPhys.87.1213.

[133] Yu. A. Bychkov and É. I. Rashba, “Properties of a 2D electron gas with lifted spectral degeneracy,” *Soviet Journal of Experimental and Theoretical Physics Letters*, vol. 39, p. 78, Jan. 1984.

[134] G. Dresselhaus, “Spin-Orbit Coupling Effects in Zinc Blende Structures,” *Physical Review*, vol. 100, no. 2, pp. 580–586, Oct. 1955, doi: 10.1103/PhysRev.100.580.

[135] A. Manchon, H. C. Koo, J. Nitta, S. M. Frolov, and R. A. Duine, “New perspectives for Rashba spin-orbit coupling,” *Nat Mater*, vol. 14, no. 9, pp. 871–882, Aug. 2015, doi: 10.1038/nmat4360.

[136] K. Garello *et al.*, “Ultrafast magnetization switching by spin-orbit torques,” *Appl Phys Lett*, vol. 105, no. 21, Nov. 2014, doi: 10.1063/1.4902443.

[137] M. Davies *et al.*, "Advancing Neuromorphic Computing with Loihi: A Survey of Results and Outlook," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 911–934, May 2021, doi: 10.1109/JPROC.2021.3067593.

[138] W. Severa, C. M. Vineyard, R. Dellana, S. J. Verzi, and J. B. Aimone, "Training deep neural networks for binary communication with the Whetstone method," *Nat Mach Intell*, vol. 1, no. 2, pp. 86–94, 2019, doi: 10.1038/s42256-018-0015-y.

[139] W. Guo, M. E. Fouda, A. M. Eltawil, and K. N. Salama, "Neural Coding in Spiking Neural Networks: A Comparative Study for Robust Neuromorphic Systems," *Front Neurosci*, vol. 15, Mar. 2021, doi: 10.3389/fnins.2021.638474.

[140] K. Y. Camsari, R. Faria, B. M. Sutton, and S. Datta, "Stochastic p-bits for invertible logic," *Phys Rev X*, vol. 7, no. 3, Jul. 2017, doi: 10.1103/PhysRevX.7.031014.

[141] A. Sengupta, G. Srinivasan, D. Roy, and K. Roy, "Stochastic Inference and Learning Enabled by Magnetic Tunnel Junctions," in *2018 IEEE International Electron Devices Meeting (IEDM)*, 2018, pp. 15.6.1-15.6.4. doi: 10.1109/IEDM.2018.8614616.

[142] K. Roy, A. Sengupta, and Y. Shim, "Perspective: Stochastic magnetic devices for cognitive computing," *J Appl Phys*, vol. 123, no. 21, Jun. 2018, doi: 10.1063/1.5020168.

[143] B. Behin-Aein, V. Diep, and S. Datta, "A building block for hardware belief networks," *Sci Rep*, vol. 6, Jul. 2016, doi: 10.1038/srep29893.

[144] D. Vodenicarevic *et al.*, "Low-Energy Truly Random Number Generation with Superparamagnetic Tunnel Junctions for Unconventional Computing," *Phys Rev Appl*, vol. 8, no. 5, Nov. 2017, doi: 10.1103/PhysRevApplied.8.054045.

[145] A. Sengupta, C. M. Liyanagedera, B. Jung, and K. Roy, "Magnetic Tunnel Junction as an On-Chip Temperature Sensor," *Sci Rep*, vol. 7, no. 1, Dec. 2017, doi: 10.1038/s41598-017-11476-7.

[146] Y. Shim, A. Sengupta, and K. Roy, "Biased Random Walk Using Stochastic Switching of Nanomagnets: Application to SAT Solver," *IEEE Trans Electron Devices*, vol. 65, no. 4, pp. 1617–1624, Apr. 2018, doi: 10.1109/TED.2018.2808232.

[147] K. Y. Camsari, B. M. Sutton, and S. Datta, "p-Bits for Probabilistic Spin Logic," *Appl Phys Rev*, vol. 6, no. 1, Sep. 2019, doi: 10.1063/1.5055860.

[148] B. R. Zink, Y. Lv, and J. P. Wang, "Independent Control of Antiparallel-and Parallel-State Thermal Stability Factors in Magnetic Tunnel Junctions for Telegraphic Signals with Two Degrees of Tunability," *IEEE Trans Electron Devices*, vol. 66, no. 12, pp. 5353–5359, Dec. 2019, doi: 10.1109/TED.2019.2948218.

[149] B. R. Zink, Y. Lv, and J. P. Wang, "Telegraphic switching signals by magnet tunnel junctions for neural spiking signals with high information capacity," *J Appl Phys*, vol. 124, no. 15, Oct. 2018, doi: 10.1063/1.5042444.

[150] Y. Cheng, B. Peng, Z. Hu, Z. Zhou, and M. Liu, "Recent development and status of magnetoelectric materials and devices," *Physics Letters, Section A: General, Atomic and Solid State Physics*, vol. 382, no. 41, pp. 3018–3025, Oct. 2018, doi: 10.1016/j.physleta.2018.07.014.

[151] M. Fiebig, "Revival of the magnetoelectric effect," *J Phys D Appl Phys*, vol. 38, no. 8, p. R123, 2005, doi: 10.1088/0022-3727/38/8/R01.

[152] D. E. Nikonov and I. A. Young, "Benchmarking spintronic logic devices based on magnetoelectric oxides," *J Mater Res*, vol. 29, no. 18, pp. 2109–2115, 2014, doi: DOI: 10.1557/jmr.2014.243.

[153] I. Dzyaloshinsky, "A thermodynamic theory of 'weak' ferromagnetism of antiferromagnetics," *Journal of Physics and Chemistry of Solids*, vol. 4, no. 4, pp. 241–255, 1958, doi: https://doi.org/10.1016/0022-3697(58)90076-3.

[154] T. Moriya, "Anisotropic Superexchange Interaction and Weak Ferromagnetism," *Physical Review*, vol. 120, no. 1, pp. 91–98, Oct. 1960, doi: 10.1103/PhysRev.120.91.

[155] J. T. Heron *et al.*, "Deterministic switching of ferromagnetism at room temperature using an electric field," *Nature*, vol. 516, no. 7531, pp. 370–373, Dec. 2014, doi: 10.1038/nature14004.

[156] A. Jaiswal and K. Roy, "MESL: Proposal for a Non-volatile Cascadable Magneto-Electric Spin Logic," *Sci Rep*, vol. 7, no. 1, p. 39793, 2017, doi: 10.1038/srep39793.

[157] I. Chakraborty, A. Agrawal, and K. Roy, "Design of a Low-Voltage Analog-to-Digital Converter Using Voltage-Controlled Stochastic Switching of Low Barrier Nanomagnets," *IEEE Magn Lett*, vol. 9, pp. 1–5, 2018, doi: 10.1109/LMAG.2018.2839097.

[158] A. Jaiswal, S. Roy, G. Srinivasan, and K. Roy, "Proposal for a Leaky-Integrate-Fire Spiking Neuron Based on Magnetoelectric Switching of Ferromagnets," *IEEE Trans Electron Devices*, vol. 64, no. 4, pp. 1818–1824, Apr. 2017, doi: 10.1109/TED.2017.2671353.

[159] S. Manipatruni, D. E. Nikonov, and I. A. Young, "Beyond CMOS computing with spin and polarization," *Nat Phys*, vol. 14, no. 4, pp. 338–343, Apr. 2018, doi: 10.1038/s41567-018-0101-4.

[160] A. Jaiswal, I. Chakraborty, and K. Roy, "Energy-Efficient Memory Using Magneto-Electric Switching of Ferromagnets," *IEEE Magn Lett*, vol. 8, pp. 1–5, 2017, doi: 10.1109/LMAG.2017.2712685.

[161] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998, doi: 10.1109/5.726791.

[162] A. F. Vincent, N. Locatelli, J. O. Klein, W. S. Zhao, S. Galdin-Retailleau, and D. Querlioz, "Analytical macrospin modeling of the stochastic switching time of spin-transfer

torque devices," *IEEE Trans Electron Devices*, vol. 62, no. 1, pp. 164–170, Jan. 2015, doi: 10.1109/TED.2014.2372475.

[163] S. Singh, A. Sarma, S. Lu, A. Sengupta, V. Narayanan, and C. R. Das, "Gesture-SNN: Co-optimizing accuracy, latency and energy of SNNs for neuromorphic vision sensors," in *2021 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2021, pp. 1–6. doi: 10.1109/ISLPED52811.2021.9502506.

[164] K. Mahapatra, S. Lu, A. Sengupta, and N. R. Chaudhuri, "Power System Disturbance Classification with Online Event-Driven Neuromorphic Computing," *IEEE Trans Smart Grid*, vol. 12, no. 3, pp. 2343–2354, May 2021, doi: 10.1109/TSG.2020.3043782.

[165] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," in *Proceedings of The 33rd International Conference on Machine Learning*, M. F. Balcan and K. Q. Weinberger, Eds., in Proceedings of Machine Learning Research, vol. 48. New York, New York, USA: PMLR, May 2016, pp. 1050–1059. [Online]. Available: https://proceedings.mlr.press/v48/gal16.html

[166] M. Romera *et al.*, "Vowel recognition with four coupled spin-torque nano-oscillators," *Nature*, vol. 563, no. 7730, pp. 230–234, Nov. 2018, doi: 10.1038/s41586-018-0632-y.

[167] A. Sengupta, A. Ankit, and K. Roy, "Performance analysis and benchmarking of all-spin spiking neural networks (Special session paper)," in *Proceedings of the International Joint Conference on Neural Networks*, Institute of Electrical and Electronics Engineers Inc., Jun. 2017, pp. 4557–4563. doi: 10.1109/IJCNN.2017.7966434.

[168] A. Ankit, A. Sengupta, P. Panda, and K. Roy, "RESPARC: A Reconfigurable and Energy-Efficient Architecture with Memristive Crossbars for Deep Spiking Neural Networks," in *Proceedings - Design Automation Conference*, Institute of Electrical and Electronics Engineers Inc., Jun. 2017. doi: 10.1145/3061639.3062311.

[169] K. Yang, D. Fick, M. B. Henry, Y. Lee, D. Blaauw, and D. Sylvester, "A 23Mb/s 23pJ/b fully synthesized true-random-number generator in 28nm and 65nm CMOS," in *Digest of Technical Papers - IEEE International Solid-State Circuits Conference*, Institute of Electrical and Electronics Engineers Inc., 2014, pp. 280–281. doi: 10.1109/ISSCC.2014.6757434.

[170] L. Liu, C.-F. Pai, Y. Li, H. W. Tseng, D. C. Ralph, and R. A. Buhrman, "Spin-Torque Switching with the Giant Spin Hall Effect of Tantalum," *Science (1979)*, vol. 336, no. 6081, pp. 555–558, May 2012, doi: 10.1126/science.1218197.

[171] Y. Kim, X. Fong, and K. Roy, "Spin-Orbit-Torque-Based Spin-Dice: A True Random-Number Generator," *IEEE Magn Lett*, vol. 6, pp. 1–4, 2015, doi: 10.1109/LMAG.2015.2496548.

[172] A. Sengupta, Z. Al Azim, X. Fong, and K. Roy, "Spin-orbit torque induced spike-timing dependent plasticity," *Appl Phys Lett*, vol. 106, no. 9, Mar. 2015, doi: 10.1063/1.4914111.

[173] D. Bhowmik, L. You, and S. Salahuddin, “Spin hall effect clocking of nanomagnetic logic without a magnetic field,” *Nat Nanotechnol*, vol. 9, no. 1, pp. 59–63, 2014, doi: 10.1038/nnano.2013.241.

[174] C. F. Pai, L. Liu, Y. Li, H. W. Tseng, D. C. Ralph, and R. A. Buhrman, “Spin transfer torque devices utilizing the giant spin Hall effect of tungsten,” *Appl Phys Lett*, vol. 101, no. 12, Sep. 2012, doi: 10.1063/1.4753947.

[175] R. Cai *et al.*, “VIBNN: Hardware acceleration of Bayesian neural networks,” in *ACM SIGPLAN Notices*, Association for Computing Machinery, Mar. 2018, pp. 476–488. doi: 10.1145/3173162.3173212.

[176] P. Wijesinghe, A. Ankit, A. Sengupta, and K. Roy, “An All-Memristor Deep Spiking Neural Computing System: A Step Toward Realizing the Low-Power Stochastic Brain,” *IEEE Trans Emerg Top Comput Intell*, vol. 2, no. 5, pp. 345–358, Oct. 2018, doi: 10.1109/TETCI.2018.2829924.

[177] Y. Chen *et al.*, “DaDianNao: A Machine-Learning Supercomputer,” in *Proceedings of the Annual International Symposium on Microarchitecture, MICRO*, IEEE Computer Society, Jan. 2015, pp. 609–622. doi: 10.1109/MICRO.2014.58.

[178] A. Vansteenkiste, J. Leliaert, M. Dvornik, M. Helsen, F. Garcia-Sanchez, and B. Van Waeyenberge, “The design and verification of MuMax3,” *AIP Adv*, vol. 4, no. 10, Oct. 2014, doi: 10.1063/1.4899186.

[179] X. Fong, S. K. Gupta, N. N. Mojumder, S. H. Choday, C. Augustine, and K. Roy, “KNACK: A hybrid spin-charge mixed-mode simulator for evaluating different genres of spin-transfer torque MRAM bit-cells,” in *International Conference on Simulation of Semiconductor Processes and Devices, SISPAD*, 2011, pp. 51–54. doi: 10.1109/SISPAD.2011.6035047.

[180] A. Ankit *et al.*, “PUMA: A Programmable Ultra-efficient Memristor-based Accelerator for Machine Learning Inference,” in *International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS*, Association for Computing Machinery, Apr. 2019, pp. 715–731. doi: 10.1145/3297858.3304049.

[181] A. Shafiee *et al.*, “ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars,” in *Proceedings - 2016 43rd International Symposium on Computer Architecture, ISCA 2016*, Institute of Electrical and Electronics Engineers Inc., Aug. 2016, pp. 14–26. doi: 10.1109/ISCA.2016.12.

[182] R. Faria, K. Y. Camsari, and S. Datta, “Implementing Bayesian networks with embedded stochastic MRAM,” *AIP Adv*, vol. 8, no. 4, Apr. 2018, doi: 10.1063/1.5021332.

[183] Y. Shim, S. Chen, A. Sengupta, and K. Roy, “Stochastic Spin-Orbit Torque Devices as Elements for Bayesian Inference,” *Sci Rep*, vol. 7, no. 1, Dec. 2017, doi: 10.1038/s41598-017-14240-z.

[184] Y. Cassuto, S. Kvatinsky, and E. Yaakobi, “Sneak-path constraints in memristor crossbar arrays,” in *2013 IEEE International Symposium on Information Theory*, 2013, pp. 156–160. doi: 10.1109/ISIT.2013.6620207.

[185] L. Shi, G. Zheng, B. Tian, B. Dkhil, and C. Duan, “Research progress on solutions to the sneak path issue in memristor crossbar arrays,” *Nanoscale Adv.*, vol. 2, no. 5, pp. 1811–1827, 2020, doi: 10.1039/D0NA00100G.

[186] Y. Cassuto, S. Kvatinsky, and E. Yaakobi, “Write sneak-path constraints avoiding disturbs in memristor crossbar arrays,” in *2016 IEEE International Symposium on Information Theory (ISIT)*, 2016, pp. 950–954. doi: 10.1109/ISIT.2016.7541439.

[187] Yong Chen *et al.*, “Nanoscale molecular-switch crossbar circuits,” *Nanotechnology*, vol. 14, no. 4, p. 462, 2003, doi: 10.1088/0957-4484/14/4/311.

[188] F. Zahoor, T. Z. Azni Zulkifli, and F. A. Khanday, “Resistive Random Access Memory (RRAM): an Overview of Materials, Switching Mechanism, Performance, Multilevel Cell (mlc) Storage, Modeling, and Applications,” *Nanoscale Research Letters*, vol. 15, no. 1. Springer, 2020. doi: 10.1186/s11671-020-03299-9.

[189] D. Ielmini, “Resistive switching memories based on metal oxides: Mechanisms, reliability and scaling,” *Semiconductor Science and Technology*, vol. 31, no. 6. Institute of Physics Publishing, May 16, 2016. doi: 10.1088/0268-1242/31/6/063002.

[190] T.-C. Chang, K.-C. Chang, T.-M. Tsai, T.-J. Chu, and S. M. Sze, “Resistance random access memory,” *Materials Today*, vol. 19, no. 5, pp. 254–264, 2016, doi: https://doi.org/10.1016/j.mattod.2015.11.009.

[191] M. Hu *et al.*, “Memristor-Based Analog Computation and Neural Network Classification with a Dot Product Engine,” *Advanced Materials*, vol. 30, no. 9, Mar. 2018, doi: 10.1002/adma.201705914.

[192] C. Li *et al.*, “Efficient and self-adaptive in-situ learning in multilayer memristor neural networks,” *Nat Commun*, vol. 9, no. 1, Dec. 2018, doi: 10.1038/s41467-018-04484-2.

[193] S. Yu *et al.*, “Binary neural network with 16 Mb RRAM macro chip for classification and online training,” in *Technical Digest - International Electron Devices Meeting, IEDM*, Institute of Electrical and Electronics Engineers Inc., Jan. 2017, pp. 16.2.1-16.2.4. doi: 10.1109/IEDM.2016.7838429.

[194] M. Song *et al.*, “Improved Distribution of Resistance Switching Through Localized Ti-Doped NiO Layer With InZnOx/CuOx Oxide Diode,” *IEEE Journal of the Electron Devices Society*, vol. 6, pp. 905–909, 2018, doi: 10.1109/JEDS.2018.2864180.

[195] D. K. Lee, G. H. Kim, H. Sohn, and M. K. Yang, “Positive effects of a Schottky-type diode on unidirectional resistive switching devices,” *Appl Phys Lett*, vol. 115, no. 26, Dec. 2019, doi: 10.1063/1.5133868.

[196] D. Kumar, R. Aluguri, U. Chand, and T. Y. Tseng, “One bipolar selector-one resistor for flexible crossbar memory applications,” *IEEE Trans Electron Devices*, vol. 66, no. 3, pp. 1296–1301, Mar. 2019, doi: 10.1109/TED.2019.2895416.

[197] Y. C. Bae *et al.*, “All oxide semiconductor-based bidirectional vertical p-n-p selectors for 3D stackable crossbar-array electronics,” *Sci Rep*, vol. 5, no. 1, p. 13362, 2015, doi: 10.1038/srep13362.

[198] C. H. Huang, T. S. Chou, J. S. Huang, S. M. Lin, and Y. L. Chueh, “Self-selecting resistive switching scheme using TiO2 nanorod arrays,” *Sci Rep*, vol. 7, no. 1, Dec. 2017, doi: 10.1038/s41598-017-01354-7.

[199] J. Woo *et al.*, “Selector-less RRAM with non-linearity of device for cross-point array applications,” *Microelectron Eng*, vol. 109, pp. 360–363, 2013, doi: 10.1016/j.mee.2013.03.130.

[200] M. Son *et al.*, “Self-selective characteristics of nanoscale VO x devices for high-density ReRAM applications,” *IEEE Electron Device Letters*, vol. 33, no. 5, pp. 718–720, May 2012, doi: 10.1109/LED.2012.2188989.

[201] K. M. Kim *et al.*, “Low-Power, Self-Rectifying, and Forming-Free Memristor with an Asymmetric Programing Voltage for a High-Density Crossbar Application,” *Nano Lett*, vol. 16, no. 11, pp. 6724–6732, Nov. 2016, doi: 10.1021/acs.nanolett.6b01781.

[202] C. Wu, T. W. Kim, H. Y. Choi, D. B. Strukov, and J. J. Yang, “Flexible three-dimensional artificial synapse networks with correlated learning and trainable memory capability,” *Nat Commun*, vol. 8, no. 1, p. 752, 2017, doi: 10.1038/s41467-017-00803-1.

[203] G. Verma, A. Prajapati, N. Bindal, and B. K. Kaushik, “Comparative analysis of spin based memories for neuromorphic computing,” in *Proc.SPIE*, Aug. 2020, p. 1147013. doi: 10.1117/12.2568378.

[204] K. Garello *et al.*, “SOT-MRAM 300MM Integration for Low Power and Ultrafast Embedded Memories,” in *2018 IEEE Symposium on VLSI Circuits*, 2018, pp. 81–82. doi: 10.1109/VLSIC.2018.8502269.

[205] S. Kanai *et al.*, “Magnetization switching in a CoFeB/MgO magnetic tunnel junction by combining spin-transfer torque and electric field-effect,” *Appl Phys Lett*, vol. 104, no. 21, May 2014, doi: 10.1063/1.4880720.

[206] T. Nozaki *et al.*, “Recent Progress in the Voltage-Controlled Magnetic Anisotropy Effect and the Challenges Faced in Developing Voltage-Torque MRAM,” *Micromachines (Basel)*, vol. 10, no. 5, 2019, doi: 10.3390/mi10050327.

[207] Y. Suzuki and S. Miwa, “Magnetic anisotropy of ferromagnetic metals in low-symmetry systems,” *Physics Letters, Section A: General, Atomic and Solid State Physics*, vol. 383, no. 11, pp. 1203–1206, Mar. 2019, doi: 10.1016/j.physleta.2019.01.020.

[208] T. Kawabe *et al.*, “Electric-field-induced changes of magnetic moments and magnetocrystalline anisotropy in ultrathin cobalt films,” *Phys Rev B*, vol. 96, no. 22, Dec. 2017, doi: 10.1103/PhysRevB.96.220412.

[209] S. Miwa *et al.*, “Voltage controlled interfacial magnetism through platinum orbits,” *Nat Commun*, vol. 8, no. 1, p. 15848, 2017, doi: 10.1038/ncomms15848.

[210] S. Sharmin, A. Jaiswal, and K. Roy, “Modeling and Design Space Exploration for Bit-Cells Based on Voltage-Assisted Switching of Magnetic Tunnel Junctions,” *IEEE Trans Electron Devices*, vol. 63, no. 9, pp. 3493–3500, Sep. 2016, doi: 10.1109/TED.2016.2587734.

[211] A. Sengupta, A. Jaiswal, and K. Roy, “True random number generation using voltage controlled spin-dice,” in *2016 74th Annual Device Research Conference (DRC)*, 2016, pp. 1–2. doi: 10.1109/DRC.2016.7548436.

[212] K. Ikegami *et al.*, “Voltage-controlled magnetic tunnel junction based MRAM for replacing high density DRAM circuits corresponding to 2X nm generation,” in *2017 IEEE International Magnetics Conference (INTERMAG)*, 2017, pp. 1–2. doi: 10.1109/INTMAG.2017.8007706.

[213] A. S. Cassidy, J. Georgiou, and A. G. Andreou, “Design of silicon brains in the nano-CMOS era: Spiking neurons, learning synapses and neural architecture optimization,” *Neural Networks*, vol. 45, pp. 4–26, 2013, doi: https://doi.org/10.1016/j.neunet.2013.05.011.

[214] G. Indiveri, “A low-power adaptive integrate-and-fire neuron circuit,” in *Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS ’03.*, 2003, pp. IV–IV. doi: 10.1109/ISCAS.2003.1206342.

[215] B. Rajendran *et al.*, “Specifications of nanoscale devices and circuits for neuromorphic computational systems,” *IEEE Trans Electron Devices*, vol. 60, no. 1, pp. 246–253, 2013, doi: 10.1109/TED.2012.2227969.

[216] D. Kuzum, S. Yu, and H. S. Philip Wong, “Synaptic electronics: Materials, devices and applications,” *Nanotechnology*, vol. 24, no. 38. Sep. 27, 2013. doi: 10.1088/0957-4484/24/38/382001.

[217] A. Kurenkov, S. DuttaGupta, C. Zhang, S. Fukami, Y. Horio, and H. Ohno, “Artificial Neuron and Synapse Realized in an Antiferromagnet/Ferromagnet Heterostructure Using Dynamics of Spin–Orbit Torque Switching,” *Advanced Materials*, vol. 31, no. 23, Jun. 2019, doi: 10.1002/adma.201900636.

[218] A. Saha, A. N. M. N. Islam, Z. Zhao, S. Deng, K. Ni, and A. Sengupta, “Intrinsic synaptic plasticity of ferroelectric field effect transistors for online learning,” *Appl Phys Lett*, vol. 119, no. 13, Sep. 2021, doi: 10.1063/5.0064860.

[219] Q. Xia and J. J. Yang, "Memristive crossbar arrays for brain-inspired computing," *Nature Materials*, vol. 18, no. 4. Nature Publishing Group, pp. 309–323, Apr. 01, 2019. doi: 10.1038/s41563-019-0291-x.

[220] R. De Rose *et al.*, "Variability-Aware Analysis of Hybrid MTJ/CMOS Circuits by a Micromagnetic-Based Simulation Framework," *IEEE Trans Nanotechnol*, vol. 16, no. 2, pp. 160–168, Mar. 2017, doi: 10.1109/TNANO.2016.2641681.

[221] J. Kang *et al.*, "Time-dependent variability in RRAM-based analog neuromorphic system for pattern recognition," in *2017 IEEE International Electron Devices Meeting (IEDM)*, 2017, pp. 6.4.1-6.4.4. doi: 10.1109/IEDM.2017.8268340.

[222] J. Grollier, D. Querlioz, K. Y. Camsari, K. Everschor-Sitte, S. Fukami, and M. D. Stiles, "Neuromorphic spintronics," *Nature Electronics*, vol. 3, no. 7. Nature Research, pp. 360–370, Jul. 01, 2020. doi: 10.1038/s41928-019-0360-9.

[223] A. Kurenkov, C. Zhang, S. DuttaGupta, S. Fukami, and H. Ohno, "Device-size dependence of field-free spin-orbit torque induced magnetization switching in antiferromagnet/ferromagnet structures," *Appl Phys Lett*, vol. 110, no. 9, Feb. 2017, doi: 10.1063/1.4977838.

[224] N. E. Miller, Z. Wang, S. Dash, A. I. Khan, and S. Mukhopadhyay, "Characterization of Drain Current Variations in FeFETs for PIM-based DNN Accelerators," in *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2021, pp. 1–4. doi: 10.1109/AICAS51828.2021.9458437.

[225] B. Naundorf, F. Wolf, and M. Volgushev, "Unique features of action potential initiation in cortical neurons," *Nature*, vol. 440, no. 7087, pp. 1060–1063, Apr. 2006, doi: 10.1038/nature04610.

[226] Z. Sun, G. Pedretti, E. Ambrosi, A. Bricalli, W. Wang, and D. Ielmini, "Solving matrix equations in one step with cross-point resistive arrays," *Proc Natl Acad Sci U S A*, vol. 116, no. 10, pp. 4123–4128, 2019, doi: 10.1073/pnas.1815682116.

[227] J. M. Goodwill *et al.*, "Implementation of a Binary Neural Network on a Passive Array of Magnetic Tunnel Junctions," *Phys Rev Appl*, vol. 18, no. 1, p. 14039, Jul. 2022, doi: 10.1103/PhysRevApplied.18.014039.

[228] W. A. Borders *et al.*, "Analogue spin–orbit torque device for artificial-neural-network-based associative memory operation," *Applied Physics Express*, vol. 10, no. 1, p. 13007, Dec. 2016, doi: 10.7567/APEX.10.013007.

[229] J. Kaiser, W. A. Borders, K. Y. Camsari, S. Fukami, H. Ohno, and S. Datta, "Hardware-Aware In Situ Learning Based on Stochastic Magnetic Tunnel Junctions," *Phys Rev Appl*, vol. 17, no. 1, p. 14016, Jan. 2022, doi: 10.1103/PhysRevApplied.17.014016.

[230] W. A. Borders, A. Z. Pervaiz, S. Fukami, K. Y. Camsari, H. Ohno, and S. Datta, "Integer factorization using stochastic magnetic tunnel junctions," *Nature*, vol. 573, no. 7774, pp. 390–393, Sep. 2019, doi: 10.1038/s41586-019-1557-9.

[231] Q. Yang *et al.*, “Spintronic Integrate-Fire-Reset Neuron with Stochasticity for Neuromorphic Computing,” *Nano Lett*, vol. 22, no. 21, pp. 8437–8444, Nov. 2022, doi: 10.1021/acs.nanolett.2c02409.

[232] M. A. N. D. B. L. A. N. D. M. W. Nessler Bernhard AND Pfeiffer, “Bayesian Computation Emerges in Generic Cortical Microcircuits through Spike-Timing-Dependent Plasticity,” *PLoS Comput Biol*, vol. 9, no. 4, pp. 1–30, May 2013, doi: 10.1371/journal.pcbi.1003037.

[233] D. Roy, I. Chakraborty, and K. Roy, “Scaling Deep Spiking Neural Networks with Binary Stochastic Activations,” in *Proceedings - 2019 IEEE International Conference on Cognitive Computing, ICCC 2019 - Part of the 2019 IEEE World Congress on Services*, Institute of Electrical and Electronics Engineers Inc., Jul. 2019, pp. 50–58. doi: 10.1109/ICCC.2019.00020.

[234] Y. Lv, R. P. Bloom, and J.-P. Wang, “Experimental Demonstration of Probabilistic Spin Logic by Magnetic Tunnel Junctions,” *IEEE Magn Lett*, vol. 10, pp. 1–5, 2019, doi: 10.1109/LMAG.2019.2957258.

[235] Y. Qu *et al.*, “Variation-Resilient True Random Number Generators Based on Multiple STT-MTJs,” *IEEE Trans Nanotechnol*, vol. 17, no. 6, pp. 1270–1281, Nov. 2018, doi: 10.1109/TNANO.2018.2873970.

[236] T. Dalgaty, N. Castellani, C. Turck, K.-E. Harabi, D. Querlioz, and E. Vianello, “In situ learning using intrinsic memristor variability via Markov chain Monte Carlo sampling,” *Nat Electron*, vol. 4, no. 2, pp. 151–161, 2021, doi: 10.1038/s41928-020-00523-3.

[237] B. Kiraly, E. J. Knol, W. M. J. van Weerdenburg, H. J. Kappen, and A. A. Khajetoorians, “An atomic Boltzmann machine capable of self-adaption,” *Nat Nanotechnol*, vol. 16, no. 4, pp. 414–420, Apr. 2021, doi: 10.1038/s41565-020-00838-4.

[238] Y. Takeuchi, H. Sato, S. Fukami, F. Matsukura, and H. Ohno, “Temperature dependence of energy barrier in CoFeB-MgO magnetic tunnel junctions with perpendicular easy axis,” *Appl Phys Lett*, vol. 107, no. 15, p. 152405, Oct. 2015, doi: 10.1063/1.4933256.

[239] J. Kim, H. C. Woo, T. Jeong, J.-H. Choi, and C. S. Hwang, “In-Depth Analysis of One Selector–One Resistor Crossbar Array for Its Writing and Reading Operations for Hardware Neural Network with Finite Wire Resistance,” *Advanced Intelligent Systems*, vol. 4, no. 4, p. 2100174, 2022, doi: https://doi.org/10.1002/aisy.202100174.

[240] S. N. Truong, “Compensating Circuit to Reduce the Impact of Wire Resistance in a Memristor Crossbar-Based Perceptron Neural Network,” *Micromachines (Basel)*, vol. 10, no. 10, 2019, doi: 10.3390/mi10100671.

[241] M. Hu, J. P. Strachan, Z. Li, R. Stanley, and Williams, “Dot-product engine as computing memory to accelerate machine learning algorithms,” in *2016 17th International Symposium on Quality Electronic Design (ISQED)*, 2016, pp. 374–379. doi: 10.1109/ISQED.2016.7479230.

[242] C. Li *et al.*, “Long short-term memory networks in memristor crossbar arrays,” *Nat Mach Intell*, vol. 1, no. 1, pp. 49–57, 2019, doi: 10.1038/s42256-018-0001-4.

VITA

Kezhou Yang

Education

- **Pennsylvania State University, University Park, USA**
  Ph.D in Materials Science and Engineering | Aug.2019 – Aug.2023

- **Pennsylvania State University, University Park, USA**
  M.Sc in Materials Science and Engineering | Aug.2018 – Aug.2019

- **Tongji University, Shanghai, China**
  B.Sc in Applied Physics | Sept.2013 – Jun.2018

Work Experience

- **Seagate Technology LLC, Normandale, USA**
  Intern - Reader Characterization | May.2022 – Aug.2022

- **Pennsylvania State University, University Park, USA**
  Graduate Research Assistant | Aug.2019 – Aug.2023