2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)

NEBULA: A Neuromorphic Spin-Based Ultra-Low Power Architecture for SNNs and ANNs

Sonali Singh, Anup Sarma, Nicholas Jao, Ashutosh Pattnaik, Sen Lu, Kezhou Yang, Abhronil Sengupta, Vijaykrishnan Narayanan, Chita R. Das
School of Electrical Engineering and Computer Science
The Pennsylvania State University
{sms821, avs6194, naj5075, szl5689, kzy74, sengupta, vxn9, cxd12}@psu.edu, ashutosh.pattnaik@hotmail.com

***Abstract*—Brain-inspired cognitive computing has so far followed two major approaches – one uses multi-layered artificial neural networks (ANNs) to perform pattern-recognition-related tasks, whereas the other uses spiking neural networks (SNNs) to emulate biological neurons in an attempt to be as efficient and fault-tolerant as the brain. While there has been considerable progress in the former area due to a combination of effective training algorithms and acceleration platforms, the latter is still in its infancy due to the lack of both. SNNs have a distinct advantage over their ANN counterparts in that they are capable of operating in an event-driven manner, thus consuming very low power. Several recent efforts have proposed various SNN hardware design alternatives, however, these designs still incur considerable energy overheads.**

In this context, this paper proposes a comprehensive design spanning across the device, circuit, architecture and algorithm levels to build an ultra low-power architecture for SNN and ANN inference. For this, we use spintronics-based magnetic tunnel junction (MTJ) devices that have been shown to function as both neuro-synaptic crossbars as well as thresholding neurons and can operate at ultra low voltage and current levels. Using this MTJ-based neuron model and synaptic connections, we design a low power chip that has the flexibility to be deployed for inference of SNNs, ANNs as well as a combination of SNN-ANN hybrid networks – a distinct advantage compared to prior works. We demonstrate the competitive performance and energy efficiency of the SNNs as well as hybrid models on a suite of workloads. Our evaluations show that the proposed design, NEBULA, is up to 7.9$\times$ more energy efficient than a state-of-the-art design, ISAAC, in the ANN mode. In the SNN mode, our design is about 45$\times$ more energy-efficient than a contemporary SNN architecture, INXS. Power comparison between NEBULA ANN and SNN modes indicates that the latter is at least 6.25$\times$ more power-efficient for the observed benchmarks.

***Index Terms*—Neural nets, low power design, domain-specific architectures, memory technologies**

I. Introduction

Machine learning based deep artificial neural networks (ANNs) have achieved phenomenal success at complex cognitive tasks such as image processing and speech and pattern recognition [23], [26]–[28], [43], [61], [81], [83], [87]. Although loosely modelled after the structure of the human brain, the energy cost of training and deploying these ANNs is quite high, exposing a wide gap between the two in terms of energy efficiency. Despite several hardware advancements with accelerators like GPUs [49], [60], [84], TPUs [39] and other custom ASICs [3], [53], [65], [67], [91], present day computing system is far from achieving the unparalleled power efficiency of a $\sim$20W human brain [19], [86]. Furthermore, the existing ANN algorithms lack the ability to efficiently process spatio-temporal information. Spiking Neural Networks (SNNs) [24], [40], [64], on the other hand, are a paradigm shift from conventional ANNs as they process information in an efficient, event-driven manner, closely emulating the human brain.

Broadly, SNNs can be considered as a special class of artificial neural networks in which neurons communicate with each other using asynchronous events, called spikes. The behavior of a network that is composed of spiking neurons is functionally similar to biological neurons, where signals are extremely sparse and are processed in a parallel fashion. As a result, SNN-based models have been used as a tool for studying various processes and subsystems of the brain including processing and storage of neural information, as well as various forms of plasticity. It has been shown that networks of spiking neurons are computationally more powerful than other non-spiking neural network models with sigmoidal gates [82]. Recent work has demonstrated that SNNs can achieve competitive accuracy at par with standard non-spiking networks (ANNs) for complex image recognition tasks [74]. When deployed on to low power neuromorphic hardware that is able to leverage their event-driven behavior, SNNs can exhibit an order of magnitude lower power consumption than an iso-network ANN [74].

Several prior works have attempted at designing scalable SNN platforms. Of these, SpiNNaker [22], TrueNorth [12], Loihi [16] follow a digital approach whereas BrainScaleS [69] and Stanford Neurogrid [6] take a mixed-signal approach. These designs vary in terms of programmability as well as the degree of biological complexity that they can handle. In spite of the significant advances, these systems are still quite power-hungry. For example, a TrueNorth chip [12] is composed of 4096, 256-neuron clusters or cores, totaling to $\sim$1.2 million neurons per chip. A human brain-scale emulation ($\sim$100B neurons) would require connecting several racks of such chips together, which would consume tens of kilo-Watts of power. Many recent approaches have focused on leveraging the emerging technologies such as ReRAM, PCRAM, FeFET

This research is supported in part by NSF grants #1317560, #1763681, #1629915, #1955815 and the Semiconductor Research Corporation CRISP and CBRIC centers.

along with their underlying physics to support more energy-efficient computation and communication primitives [4], [5], [34], [45], [51], [57], [88], [90], [92]. However, most of these technology-driven efforts have focused on small problem sizes, and their scalability to larger applications needs to be established.

While SNN-based architectures have shown promise in achieving brain-like energy efficiency compared to those based on ANN, they are yet to demonstrate their overall supremacy. This is attributed to several factors. First, for achieving brain-like energy efficiency, each computing primitive should be extremely power efficient. This is needed even at the lowest device level, where primarily CMOS or emerging technologies like RRAM/PCRAM have been used [48]. Second, ANNs have become popular and gradually more efficient and accurate due to significant advancements in general-purpose software stack [1], [13], [32], [62] as well as accelerators such as TPUs. In contrast, there is a lack of efficient HW-SW ecosystem that can support SNNs. Third, even algorithmic optimizations for improving the accuracy of SNN computations and learning are not fully explored, thereby limiting the efficacy of SNNs. Finally, there is a lack of application/benchmark suite for testing the capabilities of SNN hardware to their fullest potential. Most prior studies have either used simple applications or smaller sub-problems to demonstrate the effectiveness of SNNs. Thus, it is essential to consider a holistic approach embracing technology, architecture, and algorithm for designing a neuromorphic ecosystem.

In this context, this paper proposes a novel multi-modal architecture called NEBULA, that is distinct from all prior efforts in that it can support an SNN, an ANN as well as a hybrid model composed of SNN and ANN. To address the energy efficiency at the device level, NEBULA is designed using a spin-based ultra low-power device, called Magnetic Tunnel Junction (MTJ) [79], which is used to mimic the behavior of a single neuron (as a compute unit) as well as of a crossbar of synapses (as memory). MTJ-based primitives are known to be more power-efficient than their CMOS/PCRAM/MRAM/ReRAM based counterparts [71], thus laying the foundations of an ultra low power platform. The **main contributions** of this paper are as follows:

• It uses an ultra-low power MTJ-based synaptic device to design a *morphable* neuron core that forms the building block for a scalable architecture. In addition to performing in-memory analog computations for high parallelism, we minimize the use of power-hungry ADCs (Analog to Digital Converters) by aggregating partial sums of analog currents using a hierarchy of neurons, instead of converting them to digital values as in prior works [14], [56], [75], [78]. Furthermore, we avoid SRAM reads and writes needed to update neuron membrane potential by leveraging the inherent property of spin-based MTJ neurons to store this potential through domain wall movement.

• NEBULA can be deployed for an SNN, an ANN as well as a hybrid SNN-ANN network. SNNs need high evidence integration times to achieve accuracy comparable to ANN, which leads to high energy consumption. We show that a hybrid model can be leveraged to get the same accuracy as corresponding SNN in fewer timesteps, consequently lowering the energy consumption while retaining SNN's power efficiency. To the best of our knowledge, ours is the first work to design an architecture that can support spiking, non-spiking and hybrid models.

• We map several deep ANN and SNN workloads to the proposed architecture and perform an in-depth comparative study of accuracy, power and energy efficiency. Our evaluations show that, NEBULA, in the ANN mode is up to 7.9$\times$ more energy efficient than a state-of-the-art ANN accelerator, ISAAC [75]. In the SNN mode, it is $\approx$45$\times$ more energy-efficient than a contemporary SNN architecture, INXS [56]. Comparison of ANN and SNN modes of NEBULA shows that the latter is at least 6.25$\times$ more power-efficient.

II. Background

In this section, we provide an overview of SNNs, the functional properties of an MTJ-based synaptic device and and how these devices can be arranged to design a crossbar of spiking and non-spiking neurons for designing neural cores as the basic blocks in our NEBULA architecture.

A. ANN and SNN Primitives

The core primitives in a neural network are the neuron and the synapse. Let us first consider the traditional ANN computing framework, the de-facto standard for current deep learning algorithms. Each ANN neuron in a particular layer receives the weighted summation of neuron outputs from a previous layer. A non-linear transfer function is then applied to the weighted sum to get an activation value. Eq.1, below, shows this for a Rectified Linear (ReLU) transfer function:

$$y = \max(0, \sum_j w_j i_j), \tag{1}$$

where y is the neuron's activation value, i_j is the j-th neuron input, and w_j is the strength of the j-th incoming synapse.

As a significant shift from ANNs, SNNs draw inspiration from the fact that biological neurons process information temporally by means of sparse spiking signals. SNN computational models abstract the neuron/synapse functionality to a much higher degree of biofidelity. In this work, we focus on Linear-Integrate-Fire (IF) spiking neuron framework to model this bio-plausibility, as represented by Eq. 2 below:

$$u(t+1) = u(t) + \sum_j w_j i_j(t), \tag{2}$$

where, $u(t)$ is the model state variable (corresponding to the neuron membrane potential) and $i_j(t)$ is the input current from the j-th synaptic input at the current time-step t. The neuron *fires* or emits a spike when its membrane potential reaches a pre-defined value, v_{th}, called the firing threshold and then resets back to a resting potential v_{reset} until new inputs arrive and start accumulating at the membrane again.

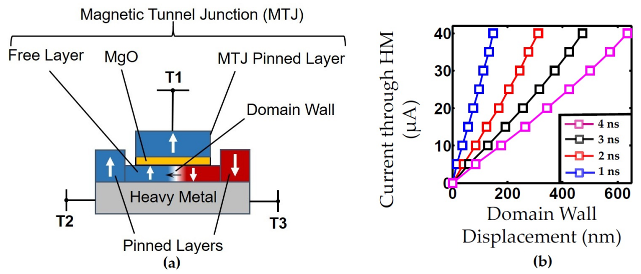

Fig. 1: Spin-based (a) DW-MTJ Synaptic device, (b) Device characteristics are shown for a $20 \times 0.6nm^3$ magnet calibrated to experimental measurements [20]. The device characteristics illustrate that programming current magnitude is directly proportional to the amount of domain wall displacement, i.e. MTJ conductance change [72].

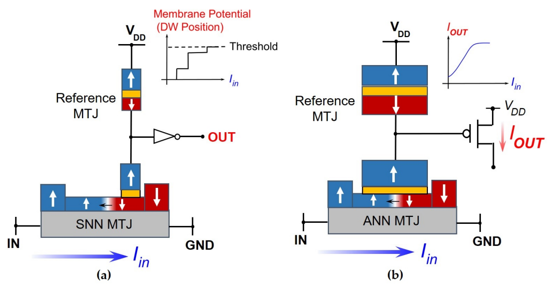

Fig. 2: Spin-based (a) Integrate-fire (IF) spiking neuron, (b) Non-spiking neuron [70], [72].

While a higher degree of biofidelity like modelling neuron membrane potential leak, homeostasis, refractory period or synaptic plasticity are exploration avenues from brain emulation perspectives [22], IF spiking neurons without any leak and refractory period have recently shown promise as a primitive for scalable training of SNNs [74]. However, our proposal can be easily extended to incorporate such additional characteristics. Inference using IF neurons is based on a rate-encoding framework – the neuron activation value is represented by the total number of spikes emitted by it over a given time window. A key advantage of using SNNs as a computational paradigm is attributed to the event-driven nature of computation. Since the neuron units transmit information as binary spike signals, neuromorphic hardware that leverage such event-driven behavior in computation and communication can operate at significantly reduced power levels. Also, as shown in Figure 4, the spiking activity (average number of spikes fired by a neuron per timestep) gradually decreases as we go deeper into the network, implying lower power consumption on event-driven hardware.

B. Spintronic Technology

1) Device Basics: Magnetic Tunnel Junction (MTJ) is the basic building block of our proposed "In-Memory" architecture. In an MTJ structure, there are two nanomagnets with a sandwiched insulator layer. The magnetization of one of the nanomagnets is pinned ("Pinned" layer), while the magnetization of the other nanomagnet ("Free" layer) is free to change its direction under an external stimulus, such as spin-transfer torque induced by a spin current [77]. The device possesses a high resistance in anti-parallel (AP) state (when the magnetizations of the two nanomagnet layers are in opposite direction) and a low resistance in parallel (P) state (when the magnetizations of the two layers are in the same direction).

2) Synaptic Device: In this work, a modified MTJ structure [72] is utilized to realize the functionality of a synapse primarily because MTJ devices can be operated at ultra-low power. The modified structure has an elongated "Free" layer to stabilize a domain wall (DW) between two magnetic domains with opposite magnetization directions. The domain wall can move continuously in the "Free" layer to change the proportion of the P and AP domains, resulting in continuous conductance change of the device, which enables multiple resistive states to be encoded. To reduce the critical current density required for domain-wall displacement and achieve decoupled "write" and "read" current paths, we consider spin-Hall effect (SHE) based magnet-heavy metal (HM) bilayers. The device structure, referred to as DW-MTJ, is shown in Figure 1(a), where a current flowing through the HM layer (between terminals T2 and T3) results in a conductance change between terminals T1 and T3 (Figure 1(b)). This device acts as the synapse primitive in our design and can be programmed to multi-level resistive states and is characterized by low switching current requirements and linear device behavior (device conductance change varies in proportion to magnitude of programming current). For inference, the device state is read through the MTJ structure between terminals T1 and T3. Experimentally, a multi-level DW motion based resistive device was recently shown to exhibit 15-20 intermediate resistive states [50]. DW-MTJ uses lower programming voltages ($\sim 100mV$) and energy ($\sim 100fJ$) compared to competing technologies such as Phase Change Memories (PCM) and Resistive Random Access Memories (RRAM) [36], [38], [44], which are characterized by much higher programming voltages (few V) and programming energies in the pJ range. They are also characterized by limited reliability and endurance in contrast to spintronics technologies [35].

3) Spiking/Non-spiking Neuronal Device: The displacement of the domain wall is proportional to the magnitude of input spikes through the heavy metal, which provides an intrinsic correspondence to the integrate-fire characteristics of a spiking neuron. The spiking neuron device structure is shown in Figure 2(a), where the MTJ is now located at the extreme edge and changes its state as the domain wall reaches the opposite edge of the ferromagnet. The SNN MTJ is interfaced with a reference MTJ and this resistive divider enables the inverter to generate an output spike [70]. Whenever there is an output spike, a current in the opposite direction is applied to reset the domain wall position to the left edge of the magnet. Additionally, for non-spiking networks, the device structure in Figure 1(a) can be used as a Saturating Rectified Linear neuron by interfacing with a reference MTJ and a transistor operating in the saturation region (instead of an inverter in the spiking scenario) [72] (Figure 2(b)). Note that the neuron threshold values are fixed in our design. In order to map different threshold values, the scaling factor can be accounted for by appropriately shifting the synaptic weight range driving the neurons. This can be either achieved by choosing an appropriate oxide thickness or scaling the synaptic read voltages.

C. All-Spin Neuromorphic Crossbar Array

Spintronic neurons and synapses can be arranged in a crossbar fashion, as shown in Figure 3, to realize an "All-Spin" crossbar array primitive [70]. RWL and WWL are the "read" and "write" control signals. In this work, since we focus only on the inference mode, the WWL control signals are turned off after programming weights at the synapse MTJs. The RWL signals control the read operations at the synapse MTJs and write operations at the neuron MTJs [72]. An appropriate magnitude of voltage applied along the BL can be used to program the domain wall position accordingly. Such an "In-Memory" computing primitive efficiently implements the parallel dot-product computing kernel by simple application of Kirchoff's law. The design of the interface neuron device and circuit as specified earlier can transform this to either an SNN or ANN. The DW-MTJ based crossbar design offers the following benefits. DW-MTJ based neurons have two properties that make it possible to operate large crossbar arrays of spintronic synapses at very low terminal voltages (typically $100mV$). Their magneto-metallic behavior ensures a low input resistance of the spin neurons, reducing the voltage drop across the device. In addition, the DW-MTJ can be switched with small currents. Since both the current and resistance are small, the input supply voltage to the entire array can be small.

Further, spintronic neurons are inherently current-driven and thereby, can be easily interfaced with the current outputs provided by the crossbar array unlike voltage-controlled devices, based on RRAM/PCM technologies. Hence, they do not require costly current to voltage converters unlike CMOS and other emerging technology-based implementations [54], [89]. Consequently, the proposed crossbar using DW-MTJ has significant potential for power and performance advantages over other competing technologies such as RRAM and PCM. Preliminary analysis has demonstrated the potential of ~100× improvement in energy consumption of "All-Spin" neuromorphic systems in comparison to a baseline CMOS implementation [73].

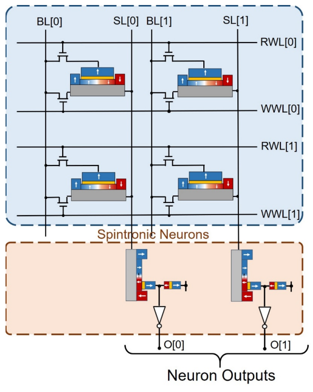

Fig. 3: All-spin Neuromorphic Crossbar Array.

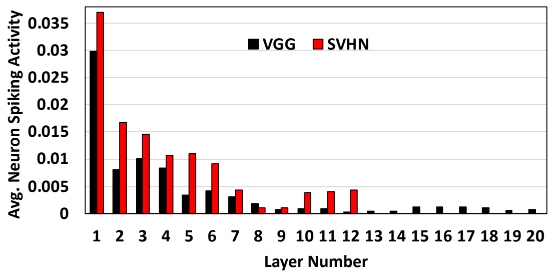

Fig. 4: Layerwise average neuron spiking activity.

III. Related Work

A significant body of research in the past decade has been dedicated to designing faster and accurate ANN models and hardware architectures. Due to the high degree of parallelism inherent in ANNs, processing-in-memory based solutions have been actively researched in order to efficiently run the large scale computations involved. ISAAC [75] and PRIME [14] use memristor-based crossbar arrays to perform in-situ multiplication in analog domain. Although such architectures are efficient in performing computation within a tile, their ADC operation in every cycle is a major power bottleneck. INXS [56], an SNN oriented architecture, performs weighted accumulation of incoming spikes to update the neuron membrane potential and a spike is generated when the membrane potential reaches its threshold value. Although the architecture emulates a spiking network, it suffers from the same fundamental disadvantage: the accumulated membrane potential at every time step has to be converted via ADC to a digital value. In addition, it does not consider explorations for the underlying hardware technology.

A large body of work has attempted to emulate the functionality of bio-inspired SNNs via SW and/or HW based solutions. For example, SpiNNaker [22] uses an ARM-based chip multiprocessor system that consumes ~1W of power to emulate thousands of neurons in a time-multiplexed fashion. Neurogrid [6] takes a mixed signal approach to simulate SNNs, where the constituent chips are connected in a tree topology. BrainScaleS [69] is a wafer-scale hardware system, which uses continuous time analog electronic circuitry to physically emulate neurons and synapse dynamics. IBM TrueNorth [12] uses digital neurosynaptic cores that mimic the operation of a cluster of neurons interconnected using a crossbar array. Intel's Loihi [16] architecture features synaptic delays and programmable synaptic learning rules in the primitive neuron model. The architecture has 128 neuromorphic cores organized in a mesh topology where each core realizes 1024 primitive spiking neural units. All of these architectures consume orders of magnitude more power than the ~20W target when scaled to billions of neurons and this is the primary motivation to look at other low-power design alternatives starting from the basic neuron model. In this context, a new flexible CMOS-based neuron model, Flexon was proposed for SNNs [47], which can simulate a wide variety of complex biological neuron models efficiently, as compared to a CPU or a GPU. As stated earlier, our MTJ-based neuron and synaptic model is much more power efficient than any CMOS based design. Besides these, Tianjic [63] is a multi-modal pattern recognition system, where SNNs are used for voice recognition and ANNs are used for object detection and tracking. It is a hybrid ANN-SNN system level design consisting of separate modules being used for different tasks. In contrast, our hybrid SNN-ANN design considers algorithmic modifications for a single network partitioned into spiking and non-spiking layers in the hybrid mode that preserves low-power benefits of SNNs, while reducing latency disadvantage of SNNs with respect to ANNs.

IV. Overview of Nebula Design

A. Overall Design

The basic building block of NEBULA is the neural core (NC) as shown in Figure 6(a). A neural core consists of an eDRAM memory that receives inputs from the network, SRAM-based input and output buffers (IB/OB), which supply inputs to and store outputs from the crossbars, a super-tile composed of a 2×2 array of morphable tiles (Tile0 to Tile3), which function as the dot product engine, as well as some peripheral circuits. Multiple neural cores (denoted by S, A in Figure 6(b)) are tiled on a chip using a mesh NoC to create a highly scalable and parallel architecture. Figure 7(b) shows that the morphable tiles are further composed of 2×2 atomic crossbars, which can be configured using switches to realize different matrix sizes. The Neuron Units (NU) are attached along both the vertical extremities of the crossbar columns through switches and each NU consists of an array of neurons designed either for spiking or non-spiking modes. In every cycle, the input buffers supply inputs to the crossbars as analog voltages along the Bit-Lines (BLs), shown in Figure 3. These voltages are weighted by conductance values programmed at each synapse and the resulting current gets summed along the column Source-Lines (SLs), thus effectively performing several dot-products in parallel. The current accumulated along a SL passes through the dedicated spin neurons and moves their domain wall in proportion to its magnitude, thereby modulating the neuron output. In addition to these components, the augmented Routing Unit (RU), in conjunction with ADC in Figure 6(a) enable efficient reduction and activation of layers which span multiple NCs. The Accumulator Unit (AU) in Figure 6(b) supports hybrid mode operation.

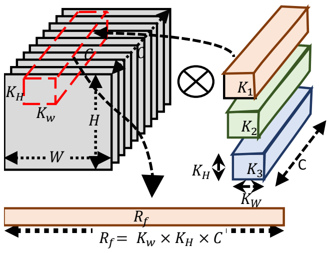

Fig. 5: Mapping a 3D kernel to a crossbar.

In the next subsections we explain the detailed design of each component on the chip.

B. Spin-based Neural Core Design

1) Dedicated Neural Cores for ANN and SNN: An important difference between ANN[1] and SNN inference is that in the former, the neurons are stateless whereas in the latter, they maintain a state. The input and output of a spiking neuron is in the form of a binary spike train. In contrast, the stateless ReLU neuron in an ANN is simply thresholded at 0 and has continuous valued (multi-bit) inputs and outputs. The hardware implication of these algorithmic differences is that we need to support multi-bit inputs to the atomic crossbars for the ANN operation and only single-bit inputs for the SNN operation. The multiple bits for the ANN can be supplied to the crossbar at once by using multi-level drivers or they can be fed serially to the crossbar using 1-bit DACs as in [75]. For our ANN NC, we choose the former design due to energy considerations. As a result, the MTJ neurons as well as the drivers in the ANN NC (denoted by A in Figure 6(b)) are larger and consume more power. SNN operation, on the other hand, does not need multi-bit drivers for binary inputs and therefore, is allocated to a dedicated low-power NC, denoted by S in Figure 6(b). Note that both the ANN and SNN NC designs are represented by Figure 6(a) as they are logically the same and only differ in the area and power consumption of their components.

2) Morphable Tiles for Varying Kernel Dimensions: Figure 5 shows how a kernel of size ($R_f = K_H \times K_W \times C$) is flattened to be mapped along the vertical dimension of a crossbar. R_f is the receptive field size of the kernel. A crossbar of size $N \times M$ can process M kernels of $R_f <= N$ in parallel. Although larger crossbar sizes provide dense synaptic connectivity and are ideal for mapping large receptive fields, they incur a high energy overhead due to the need to activate multiple rows in parallel and the large amount of read currents flowing through the SLs during MAC operations. Also, due to rigid array dimensions, sometimes many of the synaptic connections are unused. For example, the first layer of VGG-Net [76] will only use 27×64 of the synapses available in a 128×128 crossbar array. In contrast, smaller crossbars can help boost synapse utilization but will impose additional peripheral circuitry overhead due to kernel fragmentation across multiple crossbars and merge-summing. Consequently, we propose a morphable tile architecture that can better match the application requirements. Figure 7(b) shows a logical view of our tile architecture. The tile crossbar (shown as Tile2 in Figure 7(a)) is decomposable into a 2×2 array of smaller atomic crossbars (ACs) of size $M \times M$. Depending on the kernel size being mapped, the atomic crossbars can either function independently or in tandem with other ACs in a tile when they are connected via a programmable switch. Each AC has a dedicated NU which consists of M MTJ neurons, one along each SL, and M BLs, which allow it to function independently. On the other hand, all the 4 ACs can be connected using the central switches to get a fully functioning crossbar of size $2M \times 2M$. Figure 7(b) shows a possible crossbar configuration in which the right half

[1]We refer to feed-forward neural networks as ANNs - RNNs and RBMs are beyond the scope of this work.

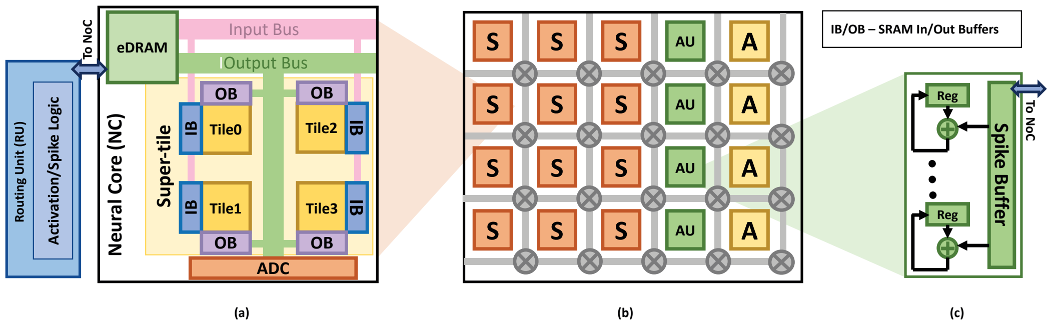

Fig. 6: The NEBULA architecture (a) A Neural Core (NC), (b) A NEBULA chip consisting of ANN and SNN NCs, (c) Accumulator Unit (AU) to support hybrid mode.

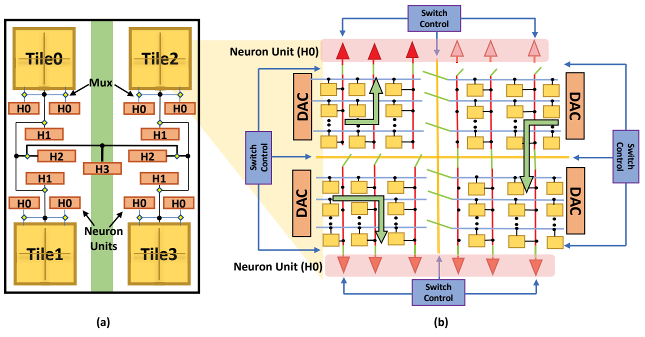

Fig. 7: (a) A *supertile* of 2×2 morphable tiles and neuron unit hierarchy, (b) A morphable tile of 2×2 atomic crossbars and neuron units at vertical extremities.

of the morphable tile supports a kernel size of up to $2M \times M$ by connecting the top and bottom ACs. The left half of the tile shows the ACs functioning independently.

3) Minimizing ADCs through Current Summation and a Hierarchy of Neuron Units: Large receptive field sizes of a layer generally overflow the crossbar dimensions and need to be mapped across multiple crossbars. In architectures such as [14], [75], [78], the partial sums of a dot product spanning multiple crossbars are first converted to digital signals using Sense Amplifiers (SAs) or ADCs and then aggregated using digital adders and stored in SRAM registers. As already noted in previous works, this operation is very power hungry due to the use of ADCs for analog to digital conversion. Also, in order to amortize the high power and area cost of ADCs, they are shared by multiple crossbars and time-multiplexed across their columns. This affects the overall throughput of the design. In order to mitigate the high ADC overheads for aggregating partial sums and improve computation throughput, we propose a novel *super-tile* design consisting of a hierarchy of NUs. The key idea is that we add the partial sums in the current domain itself by using simple Kirchoff's Current Law and activate the appropriate NU hierarchy level depending on the size of the currently mapped kernel. For example, for a kernel whose receptive field size is $R_f \leq M$ where the size of the most atomic crossbar is $M \times M$, we activate the NU arrays at hierarchy level 0 (denoted by H0 in Figure 7(a)), while the other NUs are turned off. The atomic crossbar (AC) can now process up to M kernels of size R_f in parallel. The ACs in this case will function independently and produce outputs through their dedicated NUs. If $2M < R_f \leq 4M$, we activate the central vertical switches for a crossbar size of $2M \times M$, and extend it further by summing the SL currents flowing from each half of the tile and activating only the NU at level H1 in the super-tile. As a result, a single tile of size $2M \times 2M$ can support kernel sizes of upto $4M \times M$. Further, if $4M < R_f \leq 8M$, the NUs at level H1 can be bypassed to sum SL currents coming from tiles 0 and 1. In this case, the NU at hierarchy level H2 will be active. In summary, by employing this unique design, a super-tile can support kernel sizes of upto $16M \times M$. So, if the R_f of a kernel is less than $16M$, its partial sums are aggregated within the NC, thus obviating the need for ADCs and sending multiple bits across the network fabric. If, however, $R_f > 16M$, the computations are spread out across multiple NCs whose partial sums are accumulated at RUs which are augmented with an adder and ReLU activation logic.

4) Supporting SNN, ANN and Hybrid Modes: As explained in section IV-B1, the NEBULA architecture has dedicated NCs for processing ANN and SNN to leverage the maximum advantage from both modes (Figure 6(b)). In the SNN mode, current summation output from the SLs represents the membrane potential increment of the spiking neuron that gets applied to the MTJ device and causes a displacement of its domain wall. Note that the domain wall displacement represents the neuron membrane potential at the current timestep, which persists until the next potential increment. This obviates the need to store and fetch neuron membrane potential in every cycle. Once the domain wall reaches the opposite end, it fires a spike voltage through the inverter, which is written to the SRAM output buffer. The ANN circuit works in a similar way except that the threshold voltage of the neuron is set to 0 and the output is a continuous value. In the hybrid mode, we perform a major part of the computations using SNN NCs and a small part using the ANN cores. However, as the ANNs require multi-bit continuous inputs, we cannot directly send the spiking output of the SNN cores to the ANN cores. As a result, we include an array of Accumulator Units (AUs in

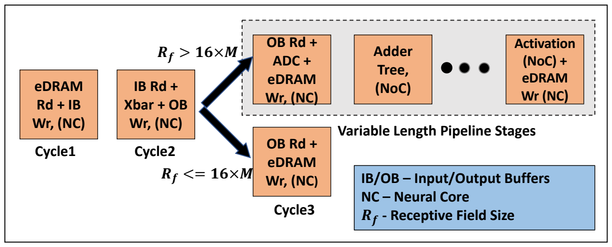
eDRAM Rd + IB Wr, (NC)
Cycle1
IB Rd + Xbar + OB Wr, (NC)
Cycle2
$R_f > 16\times M$
OB Rd + ADC + eDRAM Wr, (NC)
Adder Tree, (NoC)
Activation (NoC) + eDRAM Wr (NC)
Variable Length Pipeline Stages
$R_f <= 16\times M$
OB Rd + eDRAM Wr, (NC)
Cycle3
IB/OB – Input/Output Buffers
NC – Neural Core
R_f - Receptive Field Size

Fig. 8: Cycle-wise operations in NEBULA.

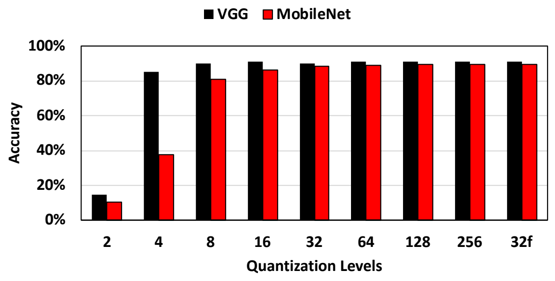

Fig. 9: Accuracy vs weight discretization levels. Activations are quantized to 4 Bits.

Figure 6(b)) in our design to accumulate the spikes over a time-window and then send it to the ANN cores. Figure 6(c) shows the AU architecture. In our hybrid mode evaluations, we have also accounted for the overhead incurred by this accumulation process.

5) NEBULA in Action: NEBULA is a pipelined architecture, where each stage in the pipeline is of 110 ns latency. The duration of a pipeline stage or cycle is dictated by the time required to switch the domain wall in the MTJ neurons. Figure 8 shows the operations taking place in each cycle in NEBULA. The pipeline starts with fetching data from the local eDRAM to the IB (cycle1). In cycle2, data is read from the IB and driven by peripheral circuitry to the crossbars. If the kernel fits onto the super-tile, the accumulated current, after traversing the neuron hierarchy, gets thresholded at the appropriate NU to yield a binary spike or a multi-bit value (depending on mode of operation), which gets written to the OB. In cycle3, the OB data is written back to eDRAM from where it is eventually released into the network ($R_f <= 16 \times M$). However, if the kernel overflows into multiple NCs (denoted by $R_f > 16 \times M$), there are additional stages in the pipeline to reduce the partial sum tree and then finally apply the activation function. In Figure 8, this is indicated by the dashed rectangle around the pipeline stages. Therefore, if the kernel MAC overflows the NC at the end of cycle-2, in cycle-3 it goes to the ADC, which sequentially digitizes the partial sums and sends it to the eDRAM. The partial sum operands are reduced by adders placed at the RUs. After one or more reduction hops, the final activation value is computed by the activation or spike logic at an RU, and is written to the eDRAM of a destination NC. Note that ADCs are used sparingly in our design – only when kernels are too large to fit on a super-tile. Also, the ADCs need to process at most 128 neurons in a 110ns cycle due to our unique mapping technique. This prevents the ADCs from being a bottleneck even though they are shared by multiple tiles in an NC. All synaptic weights are pre-programmed and control configurations are pre-computed and loaded at compile time using state machines onto the chip.

C. Precision Considerations in NEBULA

NEBULA supports ANN, SNN and hybrid models with weight and activation precision of 4 bits in inference. This is equivalent to 16 resolution levels that can be supported by the crossbar synaptic cells and the spiking MTJ neurons. Our preliminary evaluation of models trained on CIFAR and MNIST datasets shows that a weight and activation precision of 4-bits each is sufficient to achieve accuracy competitive to their floating-point counterparts. Additionally, considerable work done on quantization-aware training, post-training quantization and fine-tuning of deep neural networks [2] have demonstrated that deep networks can yield competitive accuracies during inference even at lower resolutions. As a result, we choose to adhere to 4-bits for this architecture, even though higher resolutions can be supported through additional hardware using the schemes proposed in the works such as PRIME [14], ISAAC [75] and CASCADE [15].

Since we train all our models with floating point precision, the weights and feature maps can have any value under the FP precision scope. However, the resource-constrained analog devices cannot support arbitrary magnitudes of current and precision levels, and therefore, we account for such limitations at the algorithmic level. This is done by clipping the ReLU activations at a certain percentile of the activation values. By passing a subset of the training dataset through the model, we fix the maximum activation values for every layer at a_{max}, beyond which all values would be clipped without affecting the accuracy. Next, we quantize the activation to 16 levels using a range based linear quantizer as described in [94].

Resistance-based crossbar synapse circuits impose an additional constraint on the resistance ranges that can be supported by the device. Specifically, the ratio of maximum to minimum device conductance (or equivalently synaptic weight) that can be supported in the underlying technology is limited. This translates to an additional constraint on the weight values that are programmed at the synapses. Consequently, we account for this limitation in our model simulations by clipping the kernel values to a certain range. This range was empirically decided for each layer to minimize accuracy loss. Finally, each of the weight ranges were quantized to 16 levels within maximum and minimum conductance levels to achieve a fully quantized model with negligible accuracy loss. Figure 9 reports the accuracy of VGG and MobileNet networks quantized using this technique. While limited MTJ resistance ratio between ON and OFF states is a design concern currently, $7\times$ ratio has been experimentally observed [31] and this is expected to increase to over $10\times$ in the future [29]. Alternatively, the computation can be distributed across multiple crossbars to account for limited conductance ranges [10]. Note that, since the weight and activation ranges vary from layer-to-layer,

ANN type	Dataset	% Accuracy		t steps	Depth
		ANN	SNN		
3-layer MLP	MNIST	96.81	95.75	50	3
Lenet5	MNIST	99.12	98.56	40	5
MobileNet-v1	CIFAR-10	91.00	81.08	500	29
VGG-13	CIFAR-10	91.60	90.05	300	20
MobileNet-v1	CIFAR-100	66.06	56.88	1000	29
VGG-13	CIFAR-100	71.50	68.32	1000	18
SVHN Network	SVHN	94.96	94.48	100	12
AlexNet	ImageNet	51	50	500	11

TABLE I: ANN-to-SNN conversion accuracy.

while the number of quantization levels remains constant, some signal scaling factors are needed at every layer – this is taken care of by the peripheral circuitry in our design.

D. Impact of Noise and Signal Variability

Signal noise and variability in analog computations are a concern for all emerging technologies like PCRAM, ReRAM and MTJ. However, neuromorphic applications are known to be resilient and, therefore, can tolerate non-idealities like device mismatch, noise and other variabilities [66] to a certain degree. In order to validate our hypothesis, we ran Monte-Carlo simulations with 10% variations in weights during inference for a 16-level fully quantized ANN and SNN and observed that the inference accuracy decreases only by 0.74% and 0.81%, respectively. The accuracy of the noise-injected quantized VGG-ANN is 90.31% and that of VGG-SNN is 89.41%, indicating resilience to minor imprecisions in the underlying hardware.

V. Evaluation Methodology

We developed several deep SNN benchmarks using popular ANNs that were trained using back-propagation on standard image recognition datasets such as MNIST [46], CIFAR-10, CIFAR-100 [42], SVHN [59] and ImageNet [17]. As reported in Table I, these SNNs yield comparable accuracy to their ANN counterparts. In this section, we briefly describe the method used for this conversion, which is mainly adapted from the works of [9], [18], [68]. We then provide motivation for hybrid SNN-ANN models followed by some details on device and circuit-level modeling of DW-MTJs.

A. ANN to SNN Conversion

For near lossless conversion from continuous-valued ANNs to SNNs, the original networks need to be trained with the following constraints:

Average Instead of Max-pooling for Pooling Layers: Since the feature maps are encoded with binary spikes, a max-pooling layer would either output a 1 or a 0 at every time-step, irrespective of the number of spikes present in its input map. As a result, information loss occurs due to the max-pooling from input to output feature maps. Also, max-pooling cannot be implemented directly using a crossbar architecture and requires additional circuitry. Replacing max-pooling with average pooling eases hardware mapping.

Rectified Linear Unit (ReLU) as the Sole Activation Function: This is because, the behavior of the rectified linear function

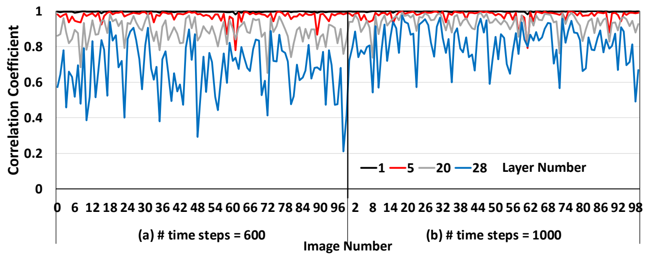

Fig. 10: Model: MobileNet-v1, Dataset: CIFAR-100. Correlation between ANN and SNN feature maps at layers 1, 5, 20 and 28.

can be well approximated by a spiking Linear-Integrate-and-Fire (IF) neuron in the absence of leak and refractory period [9], [18]. Specifically, the total number of spikes fired by the IF neuron is proportional to the activation value of the corresponding ReLU neuron.

Also, the accuracy loss is negligible when networks are trained with average instead of max-pooling layers [8] and ReLU activations are widely used in state-of-the-art CNNs. As a result, most of the popular ANNs are potential candidates for such ANN-to-SNN conversion. The spiking network (with IF neurons) is initialized with the same topology and weights as the pre-trained network and then the following modifications are made to it:

1) Each input pixel intensity is approximated by a rate-encoded poisson spike train [41] to lend a temporal aspect to the SNN. The presence and absence of a spike in a time step are represented by binary 1 and 0, respectively.
2) All the ReLU neurons in the ANN are replaced with IF neurons in the SNN. We also add an additional layer of IF neurons after every pooling layer to make the entire network amenable to SNN hardware.
3) Next, we set the thresholds for all IF neurons in the SNN using a data-based normalization approach outlined in [18] and [68].

Handling Batch-Normalization Layers: A Batch Normalization (BN) [33] layer is a part of many state-of-the-art ANNs [30] [80] [23] and is used to normalize the activation values of intermediate layers to zero mean and unit variance, thereby enabling higher learning rates and faster network convergence [7] during training. We apply the technique discussed in [68] to 'fold' back the BN layer into the weights and biases of the previous layer resulting in a BN-free ANN that can be efficiently converted to an SNN and mapped to crossbars.

B. Motivation for Hybrid SNN-ANN Models

As shown in Table I, the SNN integration times needed to yield accuracy comparable to the corresponding ANN is a function of the network depth and complexity of the dataset on which it is trained. The evidence integration time is particularly sensitive to the network depth as corroborated by Figure 10, in which we plot layer-wise correlation values between the feature maps of the ANN and the converted SNN

VGG			SVHN		
Mode	**t-step**	**Acc (%)**	**Mode**	**t-step**	**Acc (%)**
SNN	-	90.05	SNN	-	94.48
Hyb-1	250	90.10	Hyb-1	80	94.46
Hyb-2	200	85.57	Hyb-1	70	94.18
Hyb-2	150	78.90	Hyb-2	60	94.37
Hyb-2	100	59.23	Hyb-3	50	93.39
Hyb-3	100	62	Hyb-3	40	93.29

TABLE II: Hybrid SNN-ANN model accuracy. E.g. Hyb-2 has 2 non-spiking layers.

for MobileNet-v1 on CIFAR-100 for 100 images. Figure 10 indicates that as the layer depth increases from 1 to 28, the correlation between ANN and SNN feature maps drops. Also, this drop in correlation is higher for the case in which the network is simulated for 600 timesteps (shown on left), as opposed to when it is simulated for 1000 timesteps (shown on right). As a result, the former yields higher error than the latter. The long evidence integration time can be attributed to the temporal sparsity (or low spiking rate) inherent in SNNs, due to which it takes a certain amount of time to propagate enough information to the later layers for accurate prediction.

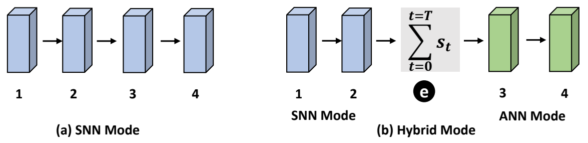

Fig. 11: Logical view of fully spiking vs hybrid models.

Longer integration times not only increase inference latency, but also lead to higher energy consumption in SNN hardware (refer to Figure 17 (first and last bars above the x-axis)). In order to reduce the integration times without loss of accuracy, we propose a hybrid approach in which, we split a deep neural network into two parts – the first part (closer to the input) is in the spiking domain and is converted using the method described in section V-A. The second part consists of layers which are in the ANN domain and accept continuous input values. The inputs to this layer are aggregated from the spikes collected from the previous layer as shown in Figure 11. The output spikes from an intermediate layer are aggregated over time (shown in white e in the figure) and scaled by a constant factor. The values obtained after such aggregation represent the activation values from the previous layer in the continuous domain, which are then fed as input to the next layer. This helps to prevent loss of information from one layer to the next due to transmission of spikes. The layers shown in green accept continuous inputs and hence, are considered to be non-spiking.

C. Power, Area and Performance Modelling

In order to drive the architectural level analysis, a device-circuit co-simulation framework was used to characterize the delay, power and area of our “All-Spin” neuromorphic crossbar array, while accounting for algorithm level constraints like bit-precision of neurons/synapses, neuron thresholds, etc. The core magnetization dynamics of the domain-wall based ferromagnetic strip is simulated in MuMax [85]. The device simulation framework was calibrated to experimental measurements reported in [20]. The MTJ resistance is simulated using a modified version of Non-Equilibrium Green’s Function (NEGF) based transport simulation framework reported in [21] and its layout area is extracted based on [37]. The device level characteristics were incorporated in the circuit level HSPICE simulator and the crossbar was designed including the synaptic device cell, interconnect parasitics and peripheral components. The interconnect parasitics are derived based on the layout dimensions of the cell and the crossbar size. The CMOS transistors, used to construct the crossbar peripherals, are evaluated with Predictive Technology Model (PTM) for a 32nm process node [93]. The layout design of the peripheral circuitry is pitch-matched to a single synaptic cell such that each row of the crossbar connects to its assigned driver, e.g. input spike drivers for the SNN and multi-voltage DAC-based driver for the ANN.

Design Tradeoffs: Next, we explain the design tradeoffs involved in the device-circuit layers of the stack. The device length was determined by the number of programmable states required in the neuron/synapse units. We considered $20nm$ to be the minimum programmable resolution for DW pinning [73], resulting in a device length of $320nm$ to represent 16 resistance states. For the synapse, the device dimensions causes the cell area of the crossbar to be large. The neuron device width, crossbar supply voltage, crossbar resistance ranges (can be varied by MTJ oxide thickness) and neuron “write” latency served as important tradeoff parameters between energy consumption and network accuracy. Since the crossbar is interfaced with the neuronal spin devices, the crossbar output current flows through the heavy-metal resistance. In order to ensure proper dot-product evaluation and consequently low neuron device resistance, the neuron device width was scaled to $200nm$. For a given neuron programming duration, the crossbar supply voltage determines the required crossbar conductance range (to ensure critical current supply to the neuron). The crossbar resistances have to be sufficiently high to ensure maximal input voltage drop across the crossbar array. Increasing the supply voltage/lowering the crossbar conductances reduces the dot-product evaluation non-ideality while increasing the energy consumption. The synaptic device width was scaled to ensure that the maximum “read” current (“write” current for neuron) does not disturb the programmed DW position in the synapse. For an extensive discussion of such device-circuit design tradeoffs, please refer to [70]. We further optimized our design based on the ANN/SNN operating mode. Due to high activation sparsity in SNN, the number of active rows for a particular crossbar is usually much lower than ANN, thereby allowing us to relax the constraints for ensuring near-ideal dot-product evaluation, while simultaneously reducing the energy consumption. Note that our ANN design (and the neuron, synapse devices), unlike the bit-splitting techniques explored in [75], has been optimized for multi-bit processing. This has been done in order to leverage the core neuron device

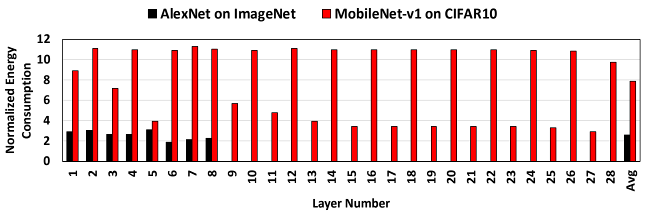

Fig. 12: Layerwise energy consumption of ISAAC [75] normalized to NEBULA-ANN for AlexNet and MobileNet.

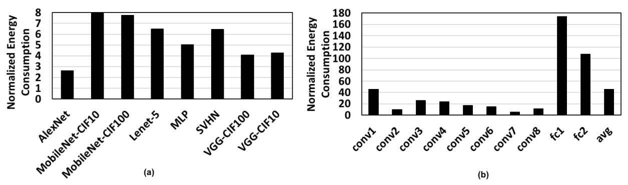

Fig. 13: (a) Average energy consumption of ISAAC [75] normalized to NEBULA-ANN, (b) Layerwise energy consumption of INXS [56] normalized to NEBULA-SNN for VGGNet.

thresholding functionality directly in a single crossbar evaluation. Hence, while our core ANN crossbar power consumption is increased due to less sparsity of ANN inputs, the number of cycles required per inference is greatly reduced in contrast to architectures utilizing bit-splitting which, in turn, reduces the resultant energy consumption per inference.

Neural Core (NC)				
Component	**Param**	**Spec**	**Power**	**Area (mm^2)**
eDRAM [25]	size	32 KB	9.55 mW	0.02523
ADC [11]	resolution	4 bits	0.43 mW	0.005
ANN Super-Tile	size	128 KB	98.87 mW	0.4247
SNN Super-Tile	size	128 KB	8.46 mW	0.3822
ANN Input Buffer	size	16 KB	4.36 mW	0.06462
SNN Input Buffer	size	4 KB	1.08 mW	0.01615
ANN Output Buffer	size	2 KB	0.545 mW	0.00808
SNN Output Buffer	size	0.5 KB	0.136 mW	0.00202
Core Total	**ANN** **SNN**		**113.8 mW** **19.66 mW**	**0.528** **0.431**
Super-Tile Components				
ANN DAC	count voltage resolution	16x128 0.75 V 4 bits	26.56 mW	0.04848
ANN Crossbar	count array size bits/cell total size	16 128x128 4 128 KB	72.16 mW	0.376
SNN Driver	count voltage resolution	16x128 0.25 V 1 bit	0.904 mW	0.00606
SNN Crossbar	count array size bits/cell total size	16 128x128 4 128 KB	7.4 mW	0.376
Neuron Unit (NU)	count	23x128	0.151 mW	0.000189
Super-Tile Total	**ANN** **SNN**		**98.87 mW** **8.46 mW**	**0.4247** **0.3822**
Digital Accumulator Unit (AU)				
Adder	count width	1024 8 bit	0.355 mW	0.00588
Register	count width total size	1024 16 bit 2 KB	0.545 mW	0.00808
Accumulator Total			**0.9 mW**	**0.0669**
Major Components Total, Operating frequency 1.2 GHz				
ANN Cores	count	14x1	1.593 W	7.392
SNN Cores	count	14x13	3.578 W	78.4
Accumulators	count	14x1	12.6 mW	0.937
Total			**5.2 W**	**86.729**

TABLE III: Component Specifications for NEBULA.

VI. Experimental Results

The benchmark networks evaluated in this work are based on popular image recognition datasets such as MNIST [46], CIFAR-10, CIFAR-100 [42], SVHN [59] and ImageNet [17]. We use both the ANN and SNN versions of the networks to evaluate the ANN and SNN hardware modes of NEBULA. Table I lists the individual benchmark's characteristics (accuracy, depth, timesteps etc.) in their ANN and SNN versions. Further, as a proof of concept, we also evaluate a hybrid version of AlexNet [43], VGGNet [76] and SVHN on our hybrid architecture and the hybrid model accuracies are reported in Table II. We map the networks to our architecture and use an analytical model to estimate the system-level area, power, energy and throughput for the benchmarks. We also compare our ANN accelerator design with the state-of-the-art CNN accelerator, ISAAC [75]. For this, we estimate their energy consumption using the component parameters reported in the paper. Since ISAAC is designed for 16-bit computations, while NEBULA is optimized for 4-bits, we adapt the ISAAC design to cater to 4-bit computations for a fair and direct comparison. As a result, we reduce the number of cycles required for 16-bit bit-serial computations from 16 cycles to 4 cycles in ISAAC, and also, scale down the ADC power accordingly. On the other hand, to provide a fair comparison with the SNN mode on NEBULA, we choose the SNN accelerator, INXS [56]. Table III reports the power, area and count of all the components used in NEBULA. Finally, we analyze the NEBULA architecture in ANN, SNN and hybrid modes and elucidate the merits of each.

A. *Comparison with ISAAC*

Figure 12 shows the layer-wise energy consumption of MobileNet-v1 and AlexNet on ISAAC [75], normalized to the energy consumption of NEBULA. On an average, the energy consumption of NEBULA, while running MobileNet-v1, is $\approx$7.9$\times$ lower when compared to executing on ISAAC. Similarly, AlexNet shows a $\approx$2.8$\times$ improvement in energy efficiency over ISAAC. One of the reasons for this is, the bit-serial computations in ISAAC lead to a deep pipeline and consequently a higher latency. Intra-tile pipeline in ISAAC for computing the dot-product one bit at a time, combined with layer replication, causes many components including the memristive crossbars and power-hungry ADCs to be active across multiple cycles. While our design also replicates kernels to gain higher parallelism, our intra-NC pipeline has fewer stages. This is enabled by the all-spin DW-MTJ crossbar arrays, which perform the dot-product in a single cycle after which its dynamic power is zero, if there are no more inputs. Additionally, since the weights are programmed on a single synaptic device as opposed to being sliced across multiple devices in the crossbar, the ADC and shift-add operations needed to aggregate different weight slices and the associated overhead are also avoided. This helps to save energy in layers with smaller receptive fields. For layers with large receptive fields, NEBULA attempts to contain the merge-summing within a super-tile by virtue of its hierarchical NUs. As a result, it is able to save energy for all layers with various receptive field sizes.

We further notice in Figure 12 that the energy savings in the even-numbered layers 2, 4, 6, 8 etc. that correspond to depthwise-separable convolutions in MobileNet-v1 are generally higher as compared to the savings in the odd-numbered layers, which perform point-wise convolutions. This is because the crossbar utilization is low due to smaller receptive field size of separable convolutions, leading to lower power consumption. As opposed to MobileNet-v1, the AlexNet network has only dense convolutions, resulting in higher crossbar utilization. Its large convolution kernels sometimes occupy multiple NCs, which can lead to the use of ADC and thereby, increase the energy consumption of NEBULA. Figure 13(a) shows the overall energy consumption of ISAAC with respect to NEBULA for different ANN benchmarks. The proposed architecture is energy-efficient across a variety of networks with different layer sizes and depths. Energy savings are highest for MobileNet because it consists of light-weight convolution layers that have small R_f sizes, which therefore, can be contained within an NC.

B. Comparison with INXS

As NEBULA is specifically designed and optimized for SNNs, we see major benefits of our design when compared to a contemporary SNN accelerator, INXS [56]. Figure 13(b) shows the layer-wise energy consumption of VGG-SNN on INXS normalized to that of NEBULA to be $\approx 45\times$ higher on average. The reasons for such high energy savings in NEBULA are two-fold: firstly, the thresholding logic is localized to a neural core and is performed in-situ by the MTJ neuron units by virtue of their domain wall movement. In INXS, the membrane potential increments from the crossbars are first converted to digital signals using an ADC, and after summation of partial sums are sent on the network to the neuron unit. Secondly, at the neuron unit in INXS, an SRAM read is performed to retrieve the previous membrane potential, which gets added to the current potential increment and is written back into the register. This operation happens at every algorithmic timestep of the SNN leading to high energy overhead. The NEBULA-SNN architecture avoids these multiple loads and stores otherwise needed for SNN operation and localizes the thresholding logic within the NC for most cases. We also observe from Figure 13(b) that the fully connected (FC) layers show greater savings than the convolution (conv) layers. This is because (for this case) the FC layers have smaller R_f sizes compared to the conv layers, which helps to avoid ADC operations. In addition, as we go deeper into the network, the low spiking rate of neurons also helps to reduce the dynamic power consumption of those layers.

C. Analyzing NEBULA

1) Power Considerations in NEBULA: ANN-SNN conversion relies on the fact that a ReLU unit behaves in a similar manner to an IF spiking neuron. In other words, it is an alternative form of information encoding, where the multi-bit ANN neuron output is encoded as a set of binary signals distributed over time. The $0, 1$ binary spike representation can be leveraged to reduce the peak and average power consumption of a system, by turning off computations in the absence of spikes. Figure 14 shows the layer-wise peak power consumption of NEBULA-ANN relative to NEBULA-SNN for different networks. As we can see, the ANN peak power consumption can be as high as $\approx 50\times$ compared to SNN. Instantaneous peak power consumption has significant impact on the chip power grid design and the cost of packaging, especially for low-cost edge devices. SNNs provide a knob to redistribute the higher instantaneous power of ANNs over multiple clock cycles. Figure 17 (bottom axes) shows the average power consumption of NEBULA-SNN with respect to NEBULA-ANN for different benchmarks. Results from all our benchmarks show that the SNN mode is $\approx 7\times$ more power-efficient than the ANN mode. Note that while other circuit-level/architecture optimizations could be potentially explored to reduce the peak power consumption, ANN-SNN conversion provides a mathematically consistent framework to automatically distribute ANN computations over time by utilizing the same hardware architecture on NEBULA. The number of iterations for SNN can be increased to reduce the instantaneous peak power and reach ANN accuracy.

2) Energy Consumption in NEBULA: Figures 15(a) and (b) show relative energy consumption of the major components in NEBULA in the SNN and ANN modes for VGGNet, respectively. Due to the low power design of spiking NCs, particularly of MTJ crossbars and DACs (refer to Table III), they make a small contribution ($\approx$23%) to the overall energy as compared to their energy contribution in the ANN cores (65.5%). As a result, the SRAM energy becomes more significant in the SNN mode (due to higher static power of SRAMs) and contributes to 36.6% of the total on average. Although there is only one ADC per NC, its energy contribution in SNN mode is higher ($\approx$12%) compared to that in ANN mode due to the large number of time-steps for which it needs to be active. We also observe that the relative energy consumption of the deeper layers is higher in ANN mode as compared to SNN. This can be attributed to the high activation sparsity of SNNs in later layers, which is also corroborated by Figure 4. Figure 16 reports the relative energy breakdown of eight different SNNs and ANNs on NEBULA, which shows similar trends as observed on VGGNet (Figure 15) for other network models. Overall, in the SNN mode, SRAM memories and crossbars followed by eDRAM mostly dominate the energy consumption, whereas in the ANN mode, crossbars and DACs are the major energy consumers.

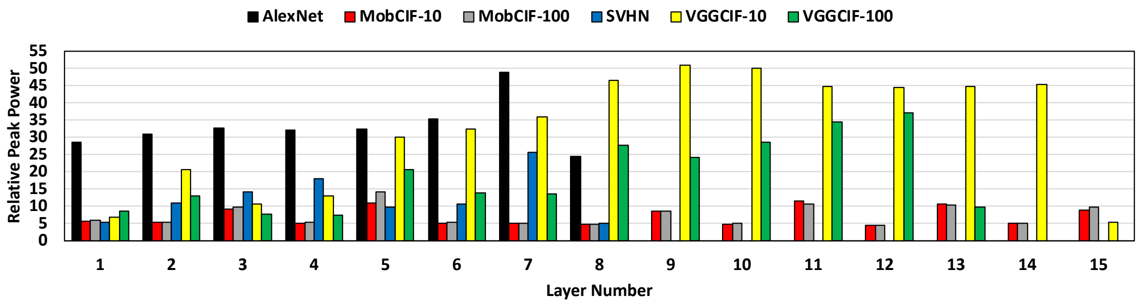

Fig. 14: Layerwise peak power of ANN compared to SNN on NEBULA for various models.

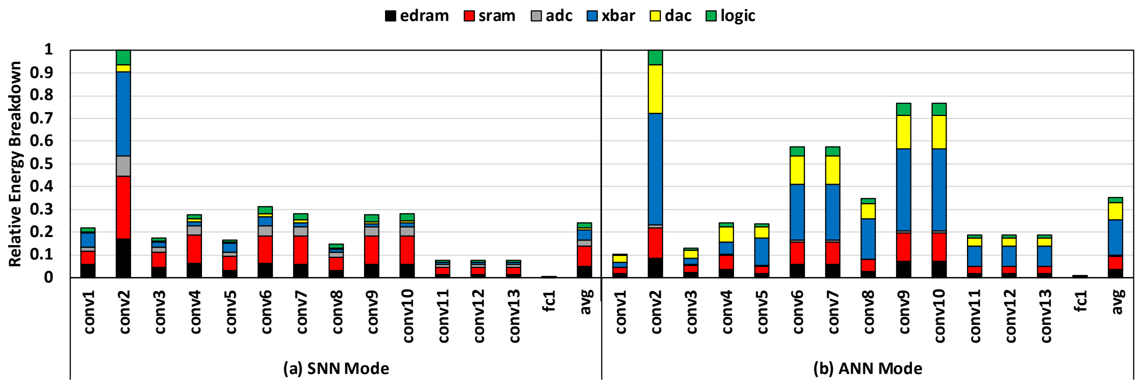

Fig. 15: Component-wise energy breakdown of VGGNet on NEBULA in (a) SNN and (b) ANN modes.

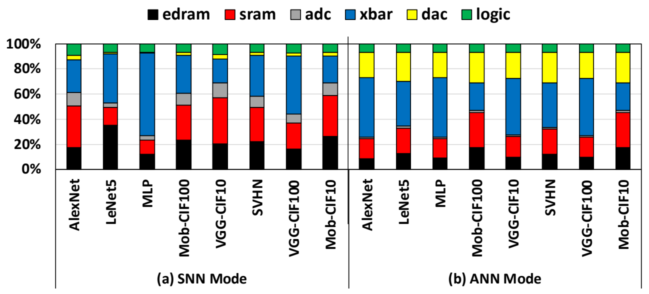

Fig. 16: Component-wise relative energy breakdown of various models on NEBULA in (a) SNN and (b) ANN modes.

3) NEBULA Hybrid SNN-ANN Architecture: Figures 17(a), (b) and (c) show a comparative behavior of SNN, hybrid and ANN models on NEBULA. The top axes show energy consumption of hybrid and ANN models with respect to SNN. The x-axis displays the number of time-steps for which the models were simulated. Starting from a pure SNN, we progressively add more ANN layers to the hybrid model from left to right (number of ANN layers in hybrid model are annotated in yellow). The rightmost bar shows pure ANN. The figures indicate that the energy consumption of SNN is ≈5-10× higher compared to that of the ANN and ≈1.1-2.5× higher than the hybrid models. The higher energy consumption of SNN is an artifact of distributing the computations in time. Since both the ANN and SNN hardware in NEBULA have the same synchronization clock, higher algorithmic time steps directly translates to higher energy consumption. This could potentially be mitigated by employing more efficient temporal codes at the algorithm level and is a field of active research [52], [55], [58].

We attempt to counter the high energy consumption of SNNs by performing a small part of the computations in the ANN domain. This serves the dual purpose of keeping both latency and energy in check, while also maintaining higher accuracy compared to pure SNNs. The dotted lines in Figure 17 indicate iso-accuracy points of the SNN and hybrid models. The numbers next to the red arrows above the hybrid bars indicate the difference in accuracy of the hybrid model compared to pure SNNs for the same number of time steps. For example, the hybrid bar consisting of 3 ANN layers at t=100 time steps in Figure 17(b) has ≈14% higher accuracy compared to a pure SNN simulated for 100 time steps.

The bottom axes in Figures 17(a), (b) and (c) show the power consumption of SNN and hybrid models normalized to the ANN power consumption on NEBULA for AlexNet, VGGNet and SVHN networks. The ANN power consumption is ≈6.25× higher than SNN for SVHN and ≈10× higher for AlexNet and VGGNet models. Compared to the hybrid models, the ANN power consumption is at least 3.6× and up to ≈9.3× higher. This establishes the importance of SNN-ANN hybrid models as a trade-off between energy/latency and power. Hybrid models can benefit from the low power requirement of SNNs, while also reducing the high latency and energy consumption of SNNs. The normalized power curve

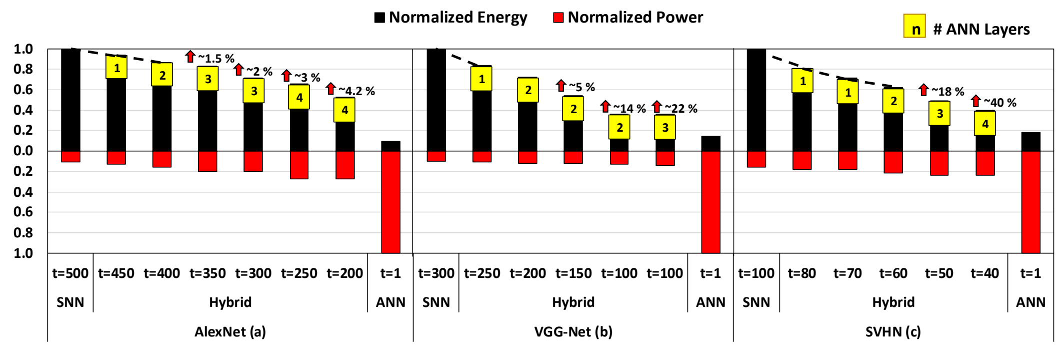

Fig. 17: SNN vs Hybrid vs ANN: Power and energy comparisons on NEBULA for (a) AlexNet, (b) VGGNet, (c) SVHN models.

shows that the power consumption of the hardware increases as more ANN layers are added to the hybrid model and in the worst case, where all but one layers are in the ANN mode, it will approach the ANN power consumption. Also, the network layer size (number of neurons) increases as we go from the bottom-up, requiring more AUs to be active, thereby increasing power overhead. As a result, it is important to have few ANN layers in the hybrid model to minimize this overhead.

VII. Conclusions

In the quest for designing more power-efficient SNN architectures to solve complex problems akin to a human brain, this paper presents a spin-based ultra-low power architecture, called NEBULA. In contrast to the prior CMOS/PCRAM/RRAM based designs, the proposed MTJ-based design is inherently more power-efficient because it operates in mV programming voltage range compared to V range for other devices. Using this MTJ-based basic neuron and synapse model, we design a morphable neuron core that has the flexibility to operate as an SNN, ANN or a hybrid network for inference – a distinct advantage compared to prior art. In addition, the neural cores need minimal number of power hungry ADCs and DACs, required in other designs. Using the neural cores, we show how the design can be scaled to integrate a large number of tiles for solving complex problems.

We demonstrate the competitive performance and energy-efficiency of the SNNs as well as the hybrid models on a suite of workloads. Comparison of the NEBULA design with state-of-the-art ISAAC [75] design in the ANN mode shows that our MTJ-based architecture is $\approx$2.8 to $\approx$7.9$\times$ more energy-efficient than the former. Similarly, in the SNN mode, NEBULA is $\approx$45$\times$ more energy efficient than the INXS design [56]. Power comparison between NEBULA ANN and SNN modes indicate that the latter is $\approx$6.25 to 10$\times$ more power-efficient for the observed benchmarks.

References

[1] M. Abadi *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015. [Online]. Available: https://www.tensorflow.org/
[2] M. Alizadeh *et al.*, "A Systematic Study of Binary Neural Networks' Optimisation," in *ICLR*, 2019.
[3] M. Alwani *et al.*, "Fused-layer CNN accelerators," in *MICRO*, 2016.
[4] A. Ankit *et al.*, "RESPARC: A Reconfigurable and Energy-Efficient Architecture with Memristive Crossbars for Deep Spiking Neural Networks," in *DAC*, 2017.
[5] A. Ankit *et al.*, "PUMA: A Programmable Ultra-efficient Memristor-based Accelerator for Machine Learning Inference," in *ASPLOS*, 2019.
[6] B. Benjamin *et al.*, "Neurogrid: A Mixed-Analog-Digital Multichip System for Large-scale Neural Simulations," *Proc. IEEE*, 2014.
[7] J. Bjorck *et al.*, "Understanding Batch Normalization," in *NeurIPS*, 2018.
[8] Y. Boureau *et al.*, "A Theoretical Analysis of Feature Pooling in Visual Recognition," in *ICML*, 2010.
[9] Y. Cao *et al.*, "Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition," *IJCV*, 2015.
[10] M. Chen *et al.*, "Magnetic Skyrmion as a Spintronic Deep Learning Spiking Neuron Processor," *IEEE Trans. Magn.*, 2018.
[11] V. Chen *et al.*, "An 8.5mW 5GS/s 6b flash ADC with dynamic offset calibration in 32nm CMOS SOI," in *VLSIC*, 2013.
[12] H.-P. Cheng *et al.*, "Exploring the Optimal Learning Technique for IBM TrueNorth Platform to Overcome Quantization Loss," in *NANOARCH*, 2016.
[13] S. Chetlur *et al.*, "cuDNN: Efficient Primitives for Deep Learning," 2014.
[14] P. Chi *et al.*, "PRIME: A Novel Processing-in-Memory Architecture for Neural Network Computation in ReRAM-Based Main Memory," in *ISCA*, 2016.
[15] T. Chou *et al.*, "CASCADE: Connecting RRAMs to Extend Analog Dataflow In An End-To-End In-Memory Processing Paradigm," in *MICRO*, 2019.
[16] M. Davies *et al.*, "Loihi: A Neuromorphic Manycore Processor with On-chip Learning," *IEEE Micro*, 2018.
[17] J. Deng *et al.*, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR*, 2009.
[18] P. U. Diehl *et al.*, "Fast-classifying, High-accuracy Spiking Deep Networks through Weight and Threshold Balancing," in *IJCNN*, 2015.
[19] D. Drubach, *The Brain Explained*. Prentice Hall, 2000.
[20] S. Emori *et al.*, "Spin Hall Torque Magnetometry of Dzyaloshinskii Domain Walls," *Phys. Rev. B*, 2014.
[21] X. Fong *et al.*, "KNACK: A Hybrid Spin-charge Mixed-mode Simulator for Evaluating Different Genres of Spin-transfer Torque MRAM Bit-cells," in *SISPAD*, 2011.
[22] S. B. Furber *et al.*, "The Spinnaker Project," *Proc. IEEE*, 2014.
[23] H. G. *et al.*, "Densely Connected Convolutional Networks," *CVPR*, 2016.
[24] S. Ghosh-Dastidar *et al.*, "Spiking Neural Networks," *IJNS*, 2009.
[25] F. Hamzaoglu *et al.*, "A 1 Gb 2 GHz 128 GB/s Bandwidth Embedded DRAM in 22 nm Tri-Gate CMOS Technology," *IEEE J. Solid-State Circuits*, 2015.
[26] K. J. Han *et al.*, "State-of-the-Art Speech Recognition Using Multi-Stream Self-Attention with Dilated 1D Convolutions," *ASRU Workshop*, 2019.
[27] J. Hauswald *et al.*, "Sirius: An Open End-to-End Voice and Vision Personal Assistant and its Implications for Future Warehouse Scale Computers," in *ASPLOS*, 2015.
[28] K. He *et al.*, "Deep Residual Learning for Image Recognition," in *CVPR*, 2016.

[29] A. Hirohata *et al.*, "Roadmap for Emerging Materials for Spintronic Device Applications," *IEEE Trans. Magn.*, 2015.

[30] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *ArXiv*, 2017.

[31] S. Ikeda *et al.*, "Tunnel Magnetoresistance of 604% at 300 K by Suppression of Ta Diffusion in Co Fe B/ Mg O/ Co Fe B Pseudo-spin-valves Annealed at High Temperature," *Appl. Phys. Lett.*, 2008.

[32] "Intel Math Kernel Library. Reference Manual," https://software.intel.com/en-us/mkl-developer-reference-c, Intel Corporation.

[33] S. Ioffe *et al.*, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *ArXiv*, 2015.

[34] R. Islam *et al.*, "Device and Materials Requirements for Neuromorphic Computing," *J Phys D Appl Phys*, 2018.

[35] A. S. Iyengar, "Energy-Efficient and Secure Designs Of Spintronic Memory: Techniques and Applications," Ph.D. dissertation, The Pennsylvania State University, 2018.

[36] B. L. Jackson *et al.*, "Nanoscale Electronic Synapses Using Phase Change Devices," *JETC*, 2013.

[37] S. Jain *et al.*, "RxNN: A Framework for Evaluating Deep Neural Networks on Resistive Crossbars," *ArXiv*, 2018.

[38] S. H. Jo *et al.*, "Nanoscale Memristor Device as Synapse in Neuromorphic Systems," *Nano Lett.*, 2010.

[39] N. P. Jouppi *et al.*, "In-datacenter Performance Analysis of a Tensor Processing Unit," in *ISCA*, 2017.

[40] E. R. Kandel *et al.*, *Principles of Neural Science*. McGraw-hill New York, 2000.

[41] R. E. Kass *et al.*, "A Spike-Train Probability Model," *Neural Comput.*, 2001.

[42] A. Krizhevsky *et al.*, "Learning Multiple Layers of Features from Tiny Images," , 2009.

[43] A. Krizhevsky *et al.*, "Imagenet Classification With Deep Convolutional Neural Networks," in *NeurIPS*, 2012.

[44] D. Kuzum *et al.*, "Nanoelectronic Programmable Synapses Based on Phase Change Materials for Brain-Inspired Computing," *Nano Lett.*, 2011.

[45] A. F. Laguna *et al.*, "Ferroelectric fet based in-memory computing for few-shot learning," in *GLSVLSI*, 2019.

[46] Y. LeCun *et al.*, "MNIST Handwritten Digit Database," http://yann.lecun.com/exdb/mnist/, 2010.

[47] D. Lee *et al.*, "Flexon: A Flexible Digital Neuron for Efficient Spiking Neural Network Simulations," in *ISCA*, 2018.

[48] J.-J. Lee *et al.*, "Integrated Neuron Circuit for Implementing Neuromorphic System with Synaptic Device," *Solid-State Electron.*, 2018.

[49] K. Lee, "Introducing Big Basin: Our Next-generation AI Hardware," "https://engineering.fb.com/data-center-engineering/introducing-big-basin-our-next-generation-ai-hardware".

[50] S. Lequeux *et al.*, "A Magnetic Synapse: Multilevel Spin-torque Memristor with Perpendicular Anisotropy," *Sci. Rep.*, 2016.

[51] H. Li *et al.*, "Neuro-inspired Computing with Emerging Memories: where Device Physics Meets Learning Algorithms," in *Spintronics XII*, 2019.

[52] T. Liu *et al.*, "PT-spike: A Precise-time-dependent Single Spike Neuromorphic Architecture with Efficient Supervised Learning," in *ASP-DAC*, 2018.

[53] T. Luo *et al.*, "Dadiannao: A Machine-Learning Supercomputer," in *MICRO*, 2014.

[54] R. Mochida *et al.*, "A 4M Synapses integrated Analog ReRAM based 66.5 TOPS/W Neural-Network Processor with Cell Current Controlled Writing and Flexible Network Architecture," in *IEEE Symposium on VLSI Technology*, 2018.

[55] H. Mostafa, "Supervised Learning Based on Temporal Coding in Spiking Neural Networks," *IEEE Trans. Neural Netw. Learn. Syst.*, 2016.

[56] S. Narayanan *et al.*, "INXS: Bridging the Throughput and Energy Gap for Spiking Neural Networks," in *IJCNN*, 2017.

[57] V. Narayanan *et al.*, "Video Analytics Using Beyond CMOS Devices," in *DATE*, 2014.

[58] E. O. Neftci *et al.*, "Surrogate Gradient Learning in Spiking Neural Networks," 2019.

[59] Y. Netzer *et al.*, "Reading Digits in Natural Images with Unsupervised Feature Learning," *NeurIPS*, 2011.

[60] Nvidia Corporation, "GPU-Based Deep Learning Inference," "https://www.nvidia.com/content/tegra/embedded-systems/pdf/jetson_tx1_whitepaper.pdf".

[61] D. S. Park *et al.*, "Specaugment: A Simple Data Augmentation Method for Automatic Speech Recognition," *ArXiv*, 2019.

[62] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *NeurIPS*, 2019.

[63] J. Pei *et al.*, "Towards Artificial General Intelligence with Hybrid Tianjic chip Architecture," *Nature*, 2019.

[64] M. Piccolino, "Luigi Galvani's Path to Animal Electricity," *Comptes Rendus Biologies*, 2006.

[65] M. Putic *et al.*, "Dyhard-DNN: Even More DNN Acceleration with Dynamic Hardware Reconfiguration," in *DAC*, 2018.

[66] D. Querlioz *et al.*, "Immunity to Device Variations in a Spiking Neural Network With Memristive Nanodevices," *IEEE Trans. Nanotechnol.*, 2013.

[67] B. Reagen *et al.*, "Minerva: Enabling Low-power, Highly-accurate Deep Neural Network Accelerators," in *ISCA*, 2016.

[68] B. Rueckauer *et al.*, "Conversion of Continuous-Valued Deep Networks to Efficient Event-Driven Networks for Image Classification," *Front. Neurosci.*, 2017.

[69] S. Schmitt *et al.*, "Neuromorphic hardware in the loop: Training a deep spiking network on the brainscales wafer-scale system," in *IJCNN*, 2017.

[70] A. Sengupta *et al.*, "A Vision for All-spin Neural Networks: A Device to System Perspective," *IIEEE Trans. Circuits Syst. I, Reg. Papers*, 2016.

[71] A. Sengupta *et al.*, "Probabilistic Deep Spiking Neural Systems Enabled by Magnetic Tunnel Junction," *IEEE Trans. Electron Devices*, 2016.

[72] A. Sengupta *et al.*, "Proposal for an All-spin Artificial Neural Network: Emulating Neural and Synaptic Functionalities through Domain Wall Motion in Ferromagnets," *IEEE Trans. Biomed. Circuits Syst.*, 2016.

[73] A. Sengupta *et al.*, "Performance Analysis and Benchmarking of All-spin Spiking Neural Networks (special session paper)," in *IJCNN*, 2017.

[74] A. Sengupta *et al.*, "Going Deeper in Spiking Neural Networks: VGG and Residual Architectures," *Front. Neurosci.*, 2019.

[75] A. Shafiee *et al.*, "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars," *ISCA*, 2016.

[76] K. Simonyan *et al.*, "Very Deep Convolutional Networks for Large-scale Image Recognition," in *ArXiv*, 2014.

[77] J. C. Slonczewski, "Currents, Torques, and Polarization Factors in Magnetic Tunnel Junctions," *Phys. Rev. B*, 2005.

[78] L. Song *et al.*, "Pipelayer: A Pipelined Reram-based Accelerator for Deep Learning," in *HPCA*, 2017.

[79] H. J. M. Swagten *et al.*, "Magnetic Tunnel Junctions," in *Encyclopedia of Materials: Science and Technology*. Elsevier, 2010.

[80] C. Szegedy *et al.*, "Going Deeper with Convolutions," in *CVPR*, 2015.

[81] M. Tan *et al.*, "Efficientnet: Rethinking Model Scaling for Convolutional Neural Networks," *ArXiv*, 2019.

[82] D. L. Tennenhouse *et al.*, "A Survey of Active Network Research," *IEEE Commun. Mag.*, 1997.

[83] H. Touvron *et al.*, "Fixing the Train-Test Resolution Discrepancy," in *NeurIPS*, 2019.

[84] S.-Z. Ueng *et al.*, "Cuda-lite: Reducing gpu programming complexity," in *LCPC*, 2008.

[85] A. Vansteenkiste *et al.*, "MuMax: a New High-performance Micromagnetic Simulation Tool," *J. Magn. Magn. Mater.*, 2011.

[86] M. Versace *et al.*, "The Brain of a New Machine," *IEEE Spectr.*, 2010.

[87] N. Wang *et al.*, "Learning a Deep Compact Image Representation for Visual Tracking," in *NeurIPS*, 2013.

[88] F. Wu *et al.*, "Regulating Firing Rates in a Neural Circuit by Activating Memristive Synapse with Magnetic Coupling," *Nonlinear Dynamics Psychol. Life Sci.*, 2019.

[89] C. Xue *et al.*, "24.1 A 1Mb Multibit ReRAM Computing-In-Memory Macro with 14.6ns Parallel MAC Computing Time for CNN Based AI Edge Processors," in *ISSCC*, 2019.

[90] B. Yan *et al.*, "Understanding the Trade-offs of Device, Circuit and Application in ReRAM-based Neuromorphic Computing Systems," in *IEDM*, 2017.

[91] C. Zhang *et al.*, "Optimizing FPGA-Based Accelerator Design for Deep Convolutional Neural Networks," in *FPGA*, 2015.

[92] T. Zhang *et al.*, "Memristive Devices and Networks for Brain-Inspired Computing," *Phys. Status Solidi-R*, 2019.

[93] W. Zhao *et al.*, "New Generation of Predictive Technology Model for sub-45 nm Early Design Exploration," *IEEE Trans. Electron Devices*, 2006.

[94] N. Zmora *et al.*, "Neural Network Distiller: A Python Package For DNN Compression Research," https://arxiv.org/abs/1910.12232, 2019.