Neuromorphic Computing: Models, Hardware, and Applications

Keya Joy
Department of Electrical & Computer Engineering
University of Washington
Seattle, USA
kjoy@uw.edu

Koume Matsutori
Department of Electrical & Computer Engineering
University of Washington
Seattle, USA
koume7@uw.edu

I. INTRODUCTION

A. *Purpose and Features*

Neuromorphic computing is a type of computing system inspired by the efficiency of the human brain. It was developed partially in response to the current artificial intelligence systems based on von Neumann architecture, which have many limitations. Classic von Neumann architecture (Fig. 1) has a limited throughput of the data bus between the memory and the processor, and the big difference between the speed of the RAM and the processor causes latency, known as the von Neumann bottleneck. Furthermore, current artificial intelligence models use large amounts of energy and power. More specifically, "the cost to power the world's largest language models (LLMs) [has] ... an estimated annual electricity bill of 25 trillion dollars" [1]. This is due to the complex computational workload associated with the training and inference stages. The computational complexity is measured in floating point operations (FLOPs), and higher FLOPs correlate to greater electricity consumption [2]. Neuromorphic computing focuses on building circuits that mimic neurons and synapses, instead of traditional boolean logic gates. It has very low power consumption; its inspiration is the human brain, which only uses tens of watts. This is made possible due to various unique neuromorphic-specific properties.

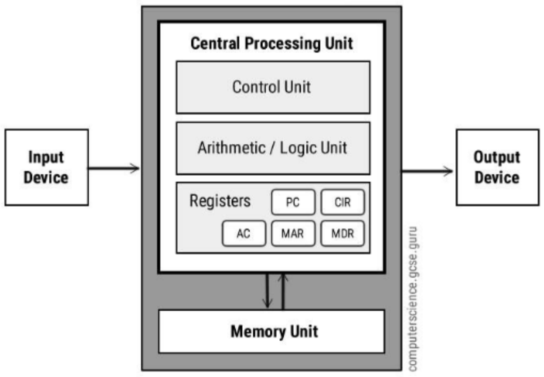

Fig. 1. Von Neumann architecture structure, depicting separation of central processing unit and memory unit. Von Neumann architecture consists of three main components – I/O device, CPU, and memory unit [3].

One key characteristic of neuromorphic computing is event-driven computation, meaning computing only takes place when data is available. More specifically, neurons and synapses in the system are only active when they receive or emit information in electric impulses, known as spikes. This characteristic contributes to energy efficiency as computations are only performed when necessary, rather than continuously like traditional systems [4]. Another property of neuromorphic computing is analog computing. Unlike digital computers which use binary digits (0s and 1s) to represent data, neuromorphic computing uses continuous, physical quantities to complete calculations. Analog neurons "can be 10,000 times faster and more energy efficient" than digital representation of information processing [5]. Additionally, analog computations use far less energy than digital computations because they use much fewer primitives [6]. To put this comparison in simpler terms, continuous quantities such as temperature and time are used in analog data, whilst digital data uses digital processing of binary digits. Neuromorphic hardware also utilizes grouped processing and memory, which helps to mitigate the von Neumann bottleneck. By performing computations and storing this data solely in the main memory, the process to reach the maximum possible throughput is sped up and leads to an overall lower energy usage due to the elimination of need to access data from the main memory. Finally, neuromorphic computing has a highly parallel operation where neurons and synapses can be operated simultaneously, which speed through massively parallel computation.

a) Existing neuromorphic projects

In recent years, neuromorphic computing has started to be implemented in various projects. Notably, there are a few leading companies and researchers that have made significant progress in the industry.

The TrueNorth project founded by IBM emulates the brain with a 4096 core chip packed with 1 million neurons and 256 million synapses, and consumes 70mW of energy when operating at a 1ms computation tick. This chip has no global clock, meaning the processing elements are interconnected through a completely asynchronous network, and operates solely on event-driven architecture [7]. The elimination of a global clock allows TrueNorth to operate with no clock skews unlike most VLSI digital circuits. TrueNorth also has a scalable design allowing for multiple cores to be combined. Using a TrueNorth chip to inspect the input from a DVS128 camera, the world's first "event-based gesture recognition system" which recognizes 11 hand gesture categories with 96.5% accuracy was developed [8]. Due to the asynchronous data representation, the camera operated at a much lower power than frame-based cameras, consuming less than 200mW (compared to 2W for frame-based) [8].

Another leader in the neuromorphic computing industry is Intel. Their latest neuromorphic chip, Loihi 2, offers real-time intelligent processing equipped with on-chip learning.

The Loihi 2 is comprised of "128 neural cores, 6 embedded processors and an asynchronous network" [9]. This chip supports a new deep neural network (DNN) implementation called Sigma-Delta Neural Network (SDNN) which increases speed and efficiency compared to traditional spiking neural network (SNN) approaches [10]. An application of Loihi 2 is Hala Point, a large-scale neuron neuromorphic system. Using 1,152 Loihi 2 processors, Hala Point supports up to 1.15 billion neurons and 128 billion synapses while only consuming 2,600 watts of power. A typical GPU rack with similar compute power would typically require 5-15 kW. Furthermore, the system can execute its full capacity at a rate of 20 times faster than the human brain [11].

The third major neuromorphic project is the Tianjic project, introduced by Tsinghua University. Unlike the above two projects, Tianjic enables hybrid networks by supporting both SNNs and artificial neural networks (ANNs). This chip contains 156 neural cores which simulate 40,000 neurons and 10 million synapses. The Tianjic chip has been used to produce an autonomous bicycle which has the ability to steer itself around obstacles, respond to voice commands, and make independent decisions [12].

B. *Training Neural Networks*

The human brain contains an estimated 10^{15} synapses, where roughly 10,000 synapses are associated with each neuron [13]. Neurons are the fundamental signaling units of the nervous system, responsible for processing and transmitting information that enables perception, learning, and memory. Each neuron consists of several key parts (Fig. 2): the soma (cell body), dendrites, axon, and presynaptic terminal. A synapse is a specialized junction where one neuron communicates with another. It consists of the presynaptic membrane, synaptic cleft, and postsynaptic membrane. Dendrites serve as the main apparatus for receiving incoming signals from other neurons, filtering and integrating this information to determine whether to generate an electrical response. The soma synthesizes neurotransmitters and integrates electrical signals from the dendrites to control the transmission of action potential. The axon acts as the neuron's transmitting element, carrying electrical impulses to other neurons, while the presynaptic terminal passes electrical signals across synapses to adjacent neurons [13].

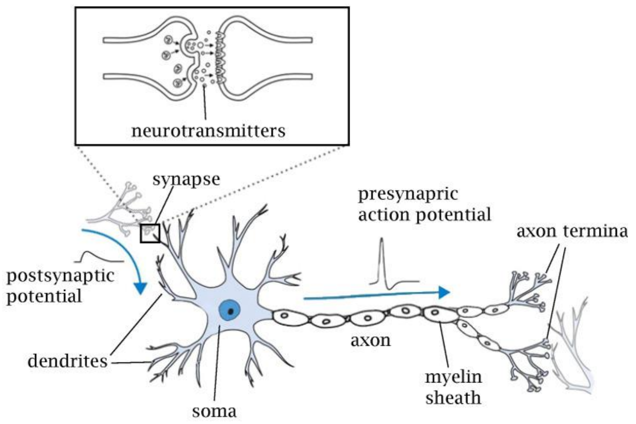

Fig. 2. Biological neuron structure with synapse junction showing the presynaptic membrane, synaptic cleft, and postsynaptic membrane in blow-up diagram [14].

There are two main mechanisms for synaptic transmission: electrical and chemical. Electrical synapses involve gap junctions that allow direct cell-to-cell transmission of potential charge through a very small synaptic cleft. These synapses are low in resistance and enable rapid, bidirectional signal propagation directly between cells. In contrast, chemical synapses rely on neurotransmitters to carry signals across the synaptic cleft, which makes transmission slower but capable of amplifying signals and enabling complex modulation of neural activity [13].

Neural plasticity – the ability of neurons and synapses to change their properties over time – underlies learning and memory formation in the brain [13]. Through mechanisms such as strengthening or weakening synaptic connections, the brain continuously adapts and reorganizes itself in response to experience. Mimicking the function of biological synapses, artificial synapses aim to synthesize synaptic plasticity.

1) *Spiking Neural Networks*

Spiking Neural Networks (SNNs) are referred to as the third generation of neural networks after artificial neural networks (ANNs) and deep neural networks (DNNs) [14]. ANNs require constant voltage signals between neurons, increasing power consumption. In comparison, within SNNs neurons communicate with each other using electrical signals called spikes. Real data can be encoded into these spikes via rate encoding – the average number of spikes over time, or temporal encoding, the time difference between spikes. SNNs embrace sparsity and a-synchronicity found in biology, making them more energy efficient than their predecessors [14].

One spiking neuron model is the Leaky Integrate and Fire (LIF) neuron which is a rudimentary but energy-efficient model showing in (1) [14].

$$\tau_m \frac{dV_{mem}}{dt} = -V_{mem} + I(t) \tag{1}$$

where V_{mem} is the post-synaptic membrane potential and τ_m is the time constant for membrane potential decay. $I(t)$ is the weighted summation of pre-spikes at each time step:

$$I(t) = \sum_{i=1}^{n}(w_i \sum_k \delta_i(t - t_k)) \tag{2}$$

where n is the number of pre-synaptic weights, w_i is the synaptic weight of the i-th synapse, $\delta_i(t - t_k)$ is a spike event from the i-th pre-neuron at time t_k modeled as a

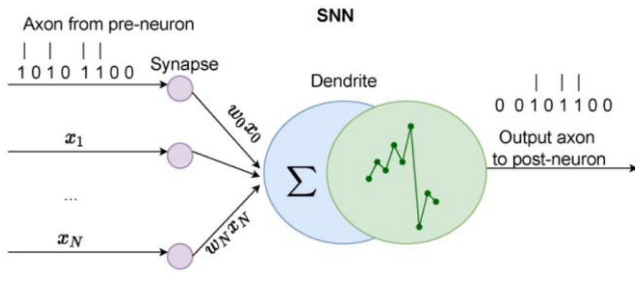

Fig. 3. Spiking neuron diagram. Each of the axons carry a signal x_i from a pre-neuron, which is multiplied by synaptic weight w_i and summed together and processed in the dendrite. Finally, the resulting signal is carried out through the output axon to other neurons [14].

Kronecker delta function. The pre-spike is modulated by the synaptic weight, so that the post-neuron receives a weighted current. When the membrane potential V_{mem} exceeds a threshold, the neuron will generate a spike and then reset the potential [15]. This weighted summation process can be seen in Fig. 3.

SNNs require much lower energy than DNNs and ANNs, but they are much more difficult to train because of complex neuron dynamics and spikes. They can be trained with several methods like supervised learning with gradient descent and spike backpropagation or reinforcement using reward modulated plasticity [14].

a) Backpropagation

Backpropagation (BP) is a machine-learning inspired method that works backwards from the output to calculate a specific set of values. The values are the gradient of the difference between the desired output and the neural network's actual output, also called a loss function. Training a network with BP requires a forward pass to make a rough prediction, a loss function, and BP of that error to calculate the partial derivatives of the function. Finally, an algorithm like gradient descent is used to update these values. Using the partial derivatives from the BP, gradient descent picks which direction is needed to update each parameter in order to reduce loss. How much each parameter is updated by is based on the learning rate, a tunable variable [16]. Spike-based backpropagation determines a set of desired firing timings of all output neurons given an input pattern [14].

b) Spike-time-dependent plasticity

Another method for training SNNs is reinforcement learning using plasticity. Spike-time-dependent plasticity (STDP) adjusts synaptic weight based on the temporal order of the pre and post synaptic spikes. If the pre-synaptic spike arrives before the post, the synaptic weight is increased, which is called long-term potential (LTP). If the pre-synaptic spike arrives after the post, the synaptic weight is decreased, which is called long-term depression (LTD). The amount the synaptic weight w is changed by is Δw, in (3).

$$\Delta w = \begin{cases} A_+ \exp\left(\frac{t_{pre}-t_{post}}{r_+}\right) & \text{if } t_{pre} \leq t_{post} \\ -A_- \exp\left(-\frac{t_{pre}-t_{post}}{r_-}\right) & \text{if } t_{pre} > t_{post} \end{cases} \quad (3)$$

A_+ and A_- can be made weight-dependent so that the change in weight is more biologically realistic. STDP is used to train the network with reinforcement learning using a reward signal. When a synapse's reward signal is positive, its weight increases. If the signal is negative, it is depressed [14]. The differences between this brain-inspired approach and backpropagation will be discussed in detail in section III.

II. IN-MEMORY COMPUTATIONS

In a typical computer, the architecture consists of several components that work together in a fetch-execute cycle. This is called von Neumann architecture (Fig. 4), and it suffices for most general-purpose computers. A CPU (central processing unit) receives an instruction, operates on data using an arithmetic unit, and sends back an output, where binary instructions and data are stored in main memory. This system requires data to constantly be sent between the CPU and memory, and data transfer is considered the main energy expenditure of AI runtime [17].

In AI computing, the large number of model weights creates an efficiency problem. The more weights are needed for a model means the larger the storage needed, which also means the storage is further away from the computing unit. A huge quantity of model weights will also have to be constantly discarded and reloaded. This creates a bottleneck, because unlike most workload latencies, the compute energy is only 10% of AI workload [18]. In-memory computing is a promising alternative that allows computation to be built directly into memory, eliminating the throughput between memory and CPU [17].

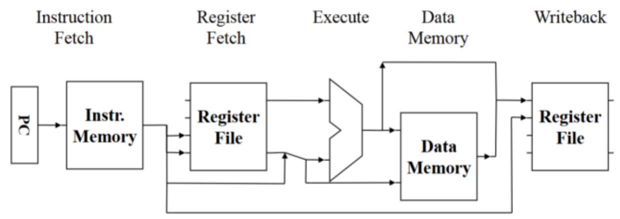

Fig. 4. Data path of von Neumann instruction cycle. Both the instruction memory and data memory exist in a separate location than the CPU, which contains the register file and execute units (Source: Primary).

A. Relevant Computation Elements

1) Memristor

A memristor device is useful for in-memory computing since it naturally integrates the computation into memory as a weighted summation. Made of a two-terminal metal-insulator-metal sandwich, this non-volatile device has promising qualities: scalable, fast operating speed, good integration density, and low power consumption. As for neuromorphic characteristics, its ability to exhibit analog conductance lends itself to displaying synaptic plasticity. Additionally, external applied voltage causes filaments in the memristor to be grown or dissolved over time creating LTP and LTD [17]. When there is no supply voltage, the memristor holds its resistance. As soon as the supply voltage across terminals surpasses its threshold, the device goes into low-resistance mode, shown by the diagonal red arrow in Fig. 5. Once the voltage resets, the memristor increases resistance, depicted as a shallow slope in black (Fig. 5).

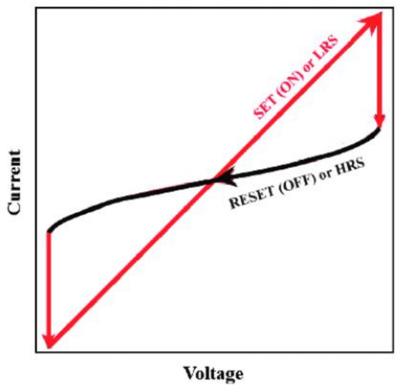

Fig. 5. Current-Voltage relationship of memristor device. The off position is a high resistance state, noted by the black shallow I-V curve, whereas the on position is a low resistance state noted by the steeper I-V curve [19].

2) *Vector Matrix Multiplication*

The main large-scale operation used for machine learning (ML) algorithms in AI computing is vector matrix multiplication (VMM). VMM multiplies an $m \times n$ matrix G by an n-element column vector V, resulting in an m-element output vector. In neural network computation, one use for VMM is to train the synaptic weights g_{ij} between two neurons. In this case, G can be trained with an algorithm like backpropagation to a data set. The time complexity of this operation can be worse case $O(n^3)$ or with some recent improvements $O(n^{2.37})$ [20]. With high runtimes, VMM is expensive, which is why memristors are being studied as an in-memory computing option [21].

3) *Multiply Accumulate Operation*

To perform a large VMM, many multiply-accumulate (MAC) operations (Fig. 6) are performed between each element v_j of the column vector V and a matrix element g_{ij}. This computation takes the product of two numbers v and g and adds it to an accumulator Z such that $Z = Z + v \times g$ [17]. This is a critical operation for AI computations at the chip level, such as natural language processing (NLP) and deep neural networks (DNNs). More than 99% of operations in most DNNs are MAC operations [22].

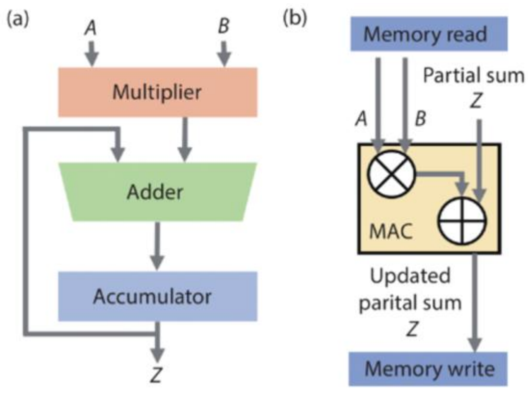

Fig. 6. MAC operation process diagram with multiplier, adder, accumulator that feeds back into the adder. A, B, and Z are read from memory, then $A \times B + Z$ is written back to Z [17].

B. *Crossbar Computations*

1) *Memristor Crossbar Array*

Following the synapse weight application, the data set would have three parts: training, testing, and validation. Each part has several tuples or vectors V. After each iteration, G is updated to $G = G + \alpha \Delta G$, where α is the learning rate and ΔG is the change after the current iteration [21]. Using memristors, these VMM operations can be done on crossbar memory arrays with notable increases in energy efficiency and time complexity.

Using a memristor crossbar array to map analog numerical matrices, MAC can be done in a parallel fashion. Fig. 7 depicts two 3-by-3 crossbar arrays of memristor nodes with conductance G_{rc}, both forward input vector (Fig. 7a), and backpropagation (Fig. 7b). In Fig. 7a, the input vector V is applied as voltage pulses V_r with differing amplitudes or pulse widths in each row. Since each memristor connects the voltage wire to the corresponding current wire, Kirchoff's Current Law tells us that the current in each column is the sum shown in (4) [17].

$$I_c = \sum_{r=1}^{3} I_{rc} \tag{4}$$

This is completed by Ohm's law, which tells us that the current going through a memristor is the product of the voltage across it and its conductance. Thus, the current I_c is collected in each column as a MAC operation between input voltage and conductance at each node such that the current in each column can be calculated with (5) [17].

$$I_c = \sum_{r=1}^{3} V_r \times G_{rc} \tag{5}$$

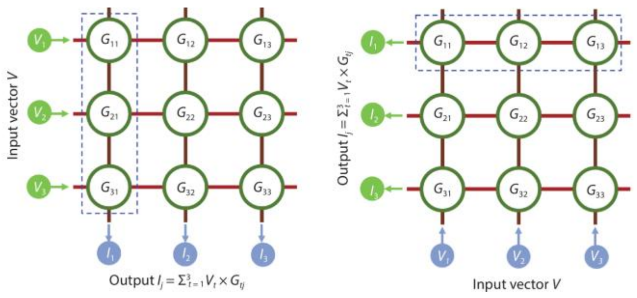

Fig. 7. Memristor crossbar arrays for (a) forward input and (b) backpropagation. The input vector contains voltages V_i that multiply with the conductances G_{ij} of the memristor nodes to get output current I_j [17].

This is also the case with backpropagation, such that both crossbar arrays can do VMM purely in the analog domain in $O(1)$ time complexity, compared to $O(n^3)$ using a processing unit. One study simulated both traditional logic gate and memristor-crossbar 2×2 MAC implementations. They found that traditional MAC consumed 19 µW with a 4.627 ns delay, whereas the memristor MAC used 3.9 µW with a 1.067 ns delay [23]. While these statistics are promising, many nuanced tradeoffs arise with using memristors in these hardware implementations, which will be discussed in the following section.

2) *RRAM Crossbar Array and Non-Ideality*

A type of memristive device called RRAM (resistive switching random access memory) can be used in a crossbar array to implement VMM. Unlike the memristor crossbar array in Fig. 7, a RRAM crossbar array has a voltage input and voltage output shown in (6) [24].

$$\begin{bmatrix} v_{o,1} \\ \vdots \\ v_{o,m} \end{bmatrix} = \begin{bmatrix} c_{1,1} & \cdots & c_{1,n} \\ \vdots & \ddots & \vdots \\ c_{m,1} & \cdots & c_{m,n} \end{bmatrix} \begin{bmatrix} v_{i,1} \\ \vdots \\ v_{i,n} \end{bmatrix} \tag{6}$$

where $(\vec{V_i})$ is the input voltage vector, $(\vec{V_o})$ is the output voltage vector. $c_{k,j}$ are the matrix parameters, shown in (7).

$$c_{k,j} = \frac{g_{k,j}}{g_s + \sum_{l=1}^{n} g_{k,l}} \tag{7}$$

where is $g_{k,j}$ the conductivity of the RRAM device, g_s is the conductivity of the load resistor (Fig. 8b).

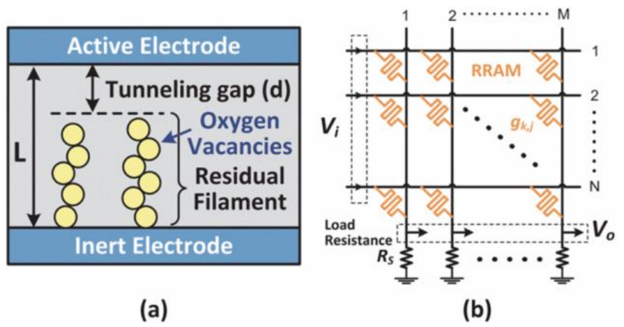

Fig. 8. Physical model of (a) RRAM and (b) RRAM crossbar array [24]. RRAM is a memristive device with an insulator center that can build up filaments, with tunneling gap d being the distance between the filament and active electrode. To build a RRAM array, each device connects an input line to an output line so that each RRAM node acts as a voltage divider with the load resistance, to get an output voltage.

Because the conductivities are positive, two arrays can be used to represent a matrix with positive and negative parameters. Another option is to use a single matrix, but voltages under a threshold (such as half the applied voltage) are considered negative. The first option requires twice the device components, but the second option could lead to more errors.

Additionally, RRAMs are often modeled as linear resistors, neglecting their nonideal factors. For example, their IV curve is nonlinear, there is interconnect resistance, and resistance state deviation [24]. The value d is the tunneling gap distance, the distance between the tip and filament in opposite electrodes of the RRAM device (Fig. 8a).

One study examined the impact of nonideal factors on the performance accuracy of the arrays [24]. Figure 9 shows how much the resistance deviates from an ideal RRAM with applied voltage and varying tunneling gap lengths. When the applied voltage is around 0V, the IV curve becomes linear. The deviation from linearity is dependent on both the applied voltage and the tunnel gap distance. For example, a voltage of 0.5V causes a 5% resistance deviation for $d = 0.2$nm. At $d = 1.9$nm, the voltage has to be constricted to 0.15V to keep within a 5% resistance deviation.

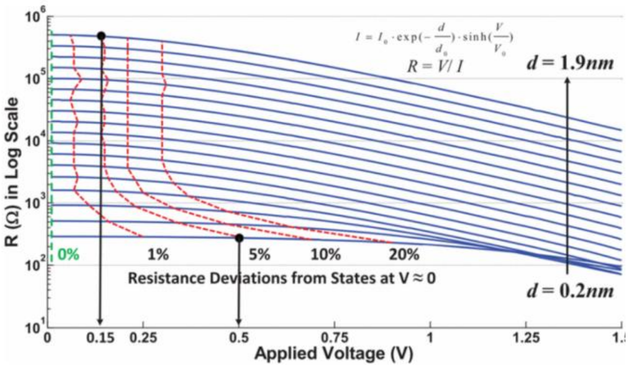

Fig. 9. Resistance vs. applied voltage for various tunneling gaps d for a hafnium oxide RRAM. The resistance state is calculated by dividing the voltage by current, which is calculated using the tunneling gap and the hyperbolic sine property dependence on voltage [24].

Interconnect resistance is another consideration for crossbar arrays. Interconnect resistance is the parasitic resistance between two neighboring RRAM devices. It impacts the error rate of the calculations in an array like Fig. 8b. If interconnect resistance is ignored, the error rate will be reported as lower than it actually is (Fig. 10). Fig. 10 represents the worst-case scenario error rate vs. crossbar size for different size technology nodes. Technology nodes are standardized nodes with specific design rules and manufacturing process. The error rate is calculated using a SPICE simulation that compares the measured and actual value of the output voltage furthest from the input ports, when all the input ports have the same voltage, and all RRAM devices are in LRS.

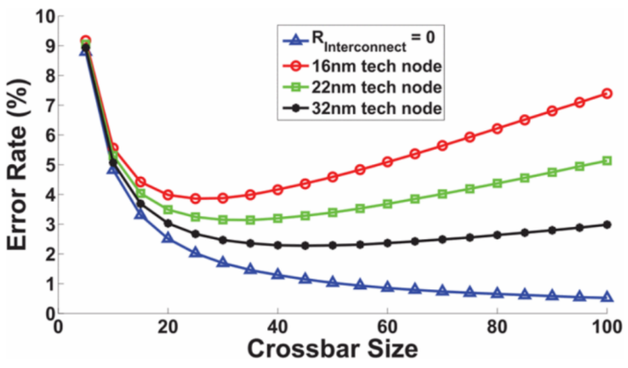

Fig. 10. Error rate vs. crossbar size for different RRAM nodes [24]. Ignoring the interconnect resistance, error rate decreases with crossbar size n of an $n \times n$ – element array. However, when accounting for accumulating interconnect resistance, the error rate begins to increase again with crossbar size.

Generally, the error rate decreases as the crossbar size increases, since lower applied voltage means more linearity. However, as the size keeps increasing, the error rate goes back up because of the interconnect resistance. Therefore, there is an optimal crossbar size for each technology node. Purely positive conductance, resistance state deviation, and interconnect resistance are just a few of the nuances of crossbar array computations that complicate neuromorphic computing. The next section will discuss more nuances and comparisons between using BP and using STDP.

III. IMPLEMENTATION AND CHALLENGES

A. *Neuromorphic Accelerators*

With the recent advancement of technology, sophisticated means of processing real-world input data such as image and audio are becoming more and more crucial. Neural network accelerators are one of the most attractive architectures for this task and can be separated into two domains – "machine-learning and neuroscience" [25]. Spiking Neural Network with Spike-Timing Dependent Plasticity (STDP) are neural network accelerators inspired by neuroscience, whereas Multi-Layer Perceptron (MLP) with BP are accelerators inspired by machine learning. Currently, FPGAs or GPUs are popular accelerators, but the underlying issue is the amount of energy these accelerators use. Therefore, these neuromorphic accelerators inspired by neuroscience and machine learning are rising as accelerators with high energy efficiency and a broad application scope. Both accelerators hold different benefits, and since they are developed by two different fields (neuroscience and machine learning), quantitative comparisons have been hard to make.

B. MLP and BP

Machine learning accelerators are gaining attraction due to its high capability of processing real-world inputs (especially video and audio), as well as the characteristic of Deep Neural Networks (DNN) – only a few variations of the same algorithm need to be implemented for a very broad application scope. However, a main downfall is that DNNs are too large to be mapped in hardware with countless synapses and neurons, so recent research has been working towards implementing a few Convolutional Neural Networks (CNNs, a variant of DNNs), of a small, full sized MLP.

MLPs contain an input, multiple hidden, and an output layer. The inputs are n-bit values, the output neurons are connected to all neurons in the hidden layer, and the hidden neurons are connected to all inputs. A neuron, n, in layer l performs (8), where s_n^l is calculated using (9), w_{ni} is the synaptic weight between neuron i in layer $l-1$ and neuron n in layer l, N_l is the number of neurons in layer l, and f is the activation function (10) [25].

$$y_n^l = f\left(s_n^l\right) \tag{8}$$

$$s_n^l = \Sigma_{i=0}^{N_{l-1}} w_{ni}^l y_i^{l-1} \tag{9}$$

$$f(x) = \frac{1}{1+e^{-x}} \tag{10}$$

BP is a learning method where an output is produced using the feed-forward path given a presented input. This iterative method computes the error between the output calculated by the network and the known output, then propagates it back through the network to update the weights. This is repeated until the target error is achieved or an allocated learning time has elapsed. The weights are updated using (11), where t is the learning iteration, η is the learning rate, and δ_n^l is the direction of error [25].

$$w_{ni}^l(t+1) = w_{ni}^l(t) + \eta \delta_n^l(t) y_i^{l-1}(t) \tag{11}$$

The direction of error is calculated using (12) in the output layer and using (13) in the hidden layer, where e_n^l is the difference between the network and expected output and f' is the derivative of f [25].

$$\delta_n^l(t) = f'\left(s_n^l(t)\right) \times e_n^l(t) \tag{12}$$

$$\delta_n^l(t) = f'(s_n^l(t)) \times \Sigma_{k=0}^{N_{l+1}} \delta_k^{l+1}(t) w_{kj}(t) \tag{13}$$

C. SNN and STDP

Accelerators inspired by neuroscience more directly emulate large-scale biological neural networks and have already successfully completed machine-learning tasks such as face and handwriting detection. Furthermore, information is coded with SNNs in these systems which lead to a significant reduction in energy usage, and STDP allows for simultaneous continuous learning while the SNN is being used.

In an SNN, the inputs and outputs of neurons are represented as spikes, as shown in Fig. 11. Additionally, when one neuron fires, all other neurons are inhibited, where it enters a refractory period and incoming spikes have no impact.

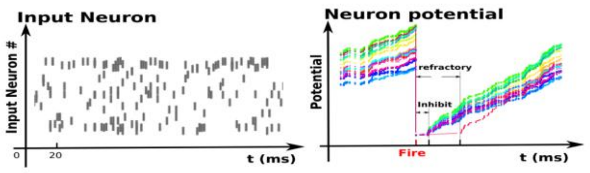

Fig. 11. Spike Coding in SNNs. (a) each line represents all spikes of 300 neurons for one MNIST presentation image. (b) the potential of the different neurons increase as they receive input spikes (one color line per neuron), until one fires; the firing neuron enters a refractory period for 20 ms, and the other neurons are inhibited for 5 ms by the firing neuron [25].

Unlike BP, STDP is an unsupervised learning principle, where each neuron progressively converges to one of the most important pieces of information [25]. If a neuron fires soon after an input spike is received, it suggests that synapse plays a role in the firing and thus should be reinforced (LTP), whereas if a neuron fires a long time after receiving an input spike, it is concluded that the synapse has no impact, and thus is decreased (LTD). A process called homeostasis is also implemented, where neurons that fire too frequently are punished by increasing their firing threshold and neurons which fire infrequently are encouraged by decreasing the firing threshold, to balance the overall information among neurons. This process improves the SNN's accuracy by ~5% [25].

D. Neuroscience vs. Machine Learning Accelerators

University of Washington researchers compared the accuracy of a MLP+BP and a SNN+STDP model using the machine-learning benchmark (MNIST) corresponding to handwritten digits (Fig. 12), a database of 60,000 examples of handwritten digits.

Fig. 12. Visualization of the MNIST data set – a data set consisting of 60,000 examples of handwritten digits. A subset of the larger NIST data set [26].

The highest recorded accuracy achieved by a neural network composed of MLPs, is 98.4% [25]. In comparison to this, the researchers used a small MLP with 100 neurons in hidden layers and trained this model to produce an accuracy of 97.65% [25]. On the other hand, an SNN model was trained to produce an accuracy of 91.82%, which was within 1.68% of the best reported result using SNN+STDP [25]. Therefore, it can be concluded that the MLP accelerator is 4.15-5.83% more accurate than the SNN accelerator. Although this is a significant difference in life-or-death situations, it is very minute in benign application such as identifying objects in a photo. Additionally, researchers concluded that most of the loss in accuracy in SNN+STDP was due to the STDP. This was found when the accuracy jumped to 95.40% when an SNN was trained to use BP instead [25].

Another key comparison is the cost in hardware. SNNs can be significantly cheaper in terms of area and power when fully expanded, but for a folded design, the MLP approach is cheaper in area/energy. In a folded design, the hardware components are shared by several logic units that use it at different times, which can work well for MLP. Although SNNs hold advantages such as online learning capability, VLS spatially expanded implementations, and its closer relationship to the brain, MLP+BP would be the more beneficial choice for typical embedded hardware in terms of cost, energy, and accuracy tradeoffs [25].

E. Bottlenecks

Revisiting the von Neumann bottleneck, AI computing uses simple, predictable operations, the conventional processor ends up working far below its full capacity [26]. Since computation efficiency progressed drastically whilst data transfer efficiency did not, processors are left in an idle state while data moves across the bottleneck. The longer the distance is to the memory in a processor, the more energy is used for data transfers. Although the time and energy cost of each data transfer is low, billions of weights are needed to be loaded for data to propagate through a large language model (LLM).

A few solutions to this bottleneck have arisen in recent years. The first is to close the distance to memory and improving data localization. In February of 2025, IBM announced a “polymer optical waveguide for co-packaged optics”, which “brings the speed and bandwidth of optics all the way to the edge of chips” [27] to reduce model training time and energy costs. In terms of in-memory computing solutions, one specific approach is phase-change memory (PCM). PCM is a non-volatile memory that stores data using silicon-chalcogenide glass, a phase-change semiconductor. PCM has high durability, fast data transfers, and lower power consumption compared to flash memory [28]. Finally, a processor called the AIU NorthPole has been introduced, which stores memory in a digital SRAM, and each core has access to local memory, making it an extreme example of near memory computing. In recent tests, it was found that NorthPole was 47x faster than the next most energy efficient GPI and 73x more energy efficient that the next lowest latency GPU [18].

F. Challenges in Neuromorphic Computing

Figure 13 showcases the current applications of neuromorphic computing. Although neuromorphic chips are

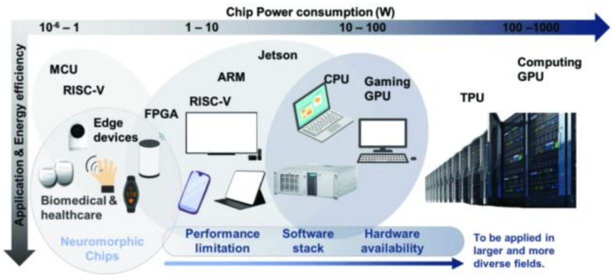

Fig. 13. Graphic showing neuromorphic applications and yet to be explored fields. The x-axis represents the power consumption of each application in Watts, and the y-axis corresponds to each application’s energy efficiency [29].

gaining significant attraction in the biomedical and healthcare domains, neuromorphic computing has yet to gain sufficient visibility and opportunities to impact larger consumer markets. Unlike deep learning used in hardware like GPUs, neuromorphic computing lacks in well-developed software for training and deployment. This lack of toolsets makes neuromorphic chips exceedingly costly and time-consuming to integrate into practical applications. Furthermore, the lack of standardized simulation and training of neuromorphic hardware limits researchers and developers from experimenting and/or implementing neuromorphic computing.

Another limitation to neuromorphic computing is its interdisciplinary complexity. Neuromorphic computing requires expertise in a vast range of fields such as neuroscience, computer science, electrical engineering, computer engineering, and materials science. This therefore requires a team consisting of researchers specializing in each field, which can be time consuming to form, and difficult to maintain.

Thirdly, neuromorphic computing software is quite complex. As explained in the “Bottlenecks” section of this paper, many neuromorphic hardware uses in-memory computing, or keeps the memory and processor in the same unit to counteract the von Neumann bottleneck. Although this is a workaround to the bottleneck, it also means that the software becomes significantly more complex and difficult to develop, increasing in the time, energy, and costs associated with developing hardware. All of this research, materials, and algorithms that go into the development of neuromorphic architecture leads to extremely expensive systems. Furthermore, since production of neuromorphic chips are not yet optimized for large scale manufacturing, the cost per chip also increases.

Lastly is the discussion of binary vs. analog SNNs. Because of the nonlinearity in memristive devices discussed in section II, additional circuitry needed can be both difficult to implement and not energy efficient. When using binary values to encode spikes, the computing becomes especially low-power, because using 16-bit floating point values require additional memory. Ternary DLNs, where the weights can only be -1, 0, or 1, work well on large-scale classification tasks, because there is less impact from cycle-to-cycle variation that RRAM analog conductance displays [30].

IV. Applications

A. Current Usages

In recent years, there has been a significant growth in neuromorphic computing as discussed in the introduction of this paper. In addition to the few leading projects already explored in this paper, this section will discuss current usages of neuromorphic computing in various fields.

1) Human Brain Modeling

One of the earliest applications of neuromorphic was to simply produce a model of the human brain to understand the different brain regions. Abstract models of spiking neurons and synapses can model the electrical activities of the brain regions. Hardware such as the SpiNNaker system consists of up to a million ARM cores, and GPU-enabled simulation framework, GeNN, consists of over 70,000 neurons. "Neurological dynamics, ... episodic memories, and different neuroscience models of cognition are being studied by modeling and simulating them as SNNs" [31].

2) Transportation

Another big sector in neuromorphic computing is mobility. Currently, sensors used in vehicles and traffic lights utilize convolutional neural network (CNN)-based approaches. Despite having a limited electrical capacity primarily used for vehicle movement, cars use a wide variety of sensors (cameras, lidar, radar, cellular communication, etc.). In fact, the computation required to control autonomous vehicles dominates from 40% - 80% of power consumption in an autonomous control system [32]. Therefore, this naturally points towards neuromorphic computing, a much more energy efficient solution. Studies have already been able to successfully automate a small car using neuromorphic computing in a small-scale, controlled environment. Furthermore, neuromorphic computing has high performance and orders of gain in energy efficiency, meaning a quicker course of correction and improved collision avoidance is made possible. Continuing in the scope of automation, smart home and manufacturing automation hold significant potential for increased energy efficiency. Similarly to cars, both smart automations heavily utilize cameras, which currently use CNN-based approaches. SNN training and development based on evolutionary optimization can develop reinforced learning and control, and advanced sensors can be used to perform sensor fusion and event detection or prediction.

3) Health Sciences

In the field of health sciences, CNN-based computing is used in image analysis such as pathology and radiology. As found in automation, there are many opportunities within health sciences where real time processing and low SWaP (size, weight, and power) requirements could significantly improve patient care due to the low power requirements of neuromorphic devices. One study used deep learning to analyze pathology images used for cancer research. It was found that there was an approximate 16x speedup in processing pathology images in real time compared to a baseline inception network [33]. Other opportunities in the healthcare field that are being focused on include sensors to monitor patient vitals to identify patterns that may not be visible with current devices. These devices could be embedded within a bed, a wristband, or a smart watch. However, there is currently a lack of data sets for training and developing SNNs, which is the biggest obstacle in developing more neuromorphic devices.

4) Facial Recognition

Loihi chips, which were briefly touched on in the introduction, have been used to conduct real-time facial expression recognition. When compared to AI accelerators, Loihi reduced power dissipation in approximately two orders of magnitude and increased energy savings in order of magnitude. Additionally, these reductions in power and energy were achieved whilst maintaining a comparable accuracy. Figure 14 shows the results from five images tested on Loihi, with the 14a being the raw, greyscale image, and 14b being the edge-detected images [34].

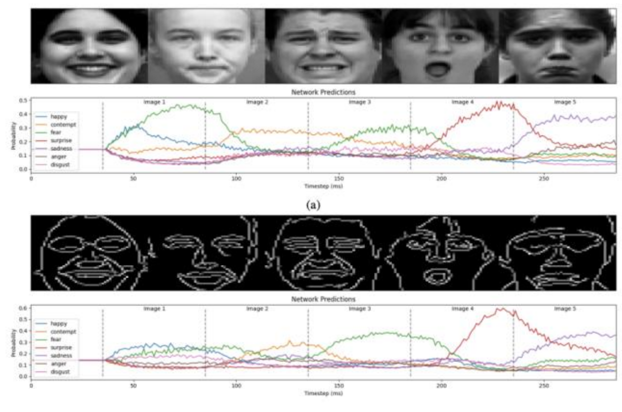

Fig. 14. Inference results from five sample photos./images tested on Loihi: (a) grayscale image of each facial expression, and (b) edge-detected images [34].

5) Bio-Inspired Hardware Systems

As discussed, research efforts for bio-inspired hardware systems are gaining more and more interest due to the problems encountered during the complex VLSI circuits manufacturing process. Internationally, there are two main bio-inspired hardware systems – a two-level embryonic structure developed by University of York, England; and a four-level embryonic structured developed by the Swiss Federal Institute of Technology.

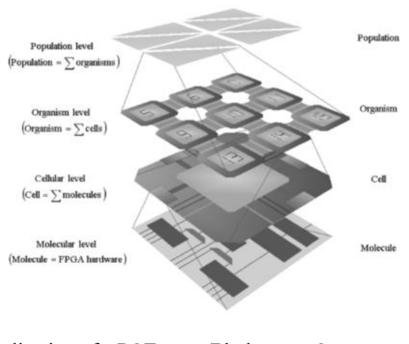

Fig. 15. Visualization of a POE-type (Phylogeny, Ontogeny, and Epigenesis) embryonic system developed in analogy with the evolutionary process of biological systems. Consists of four levels [35].

The architecture developed by the Swiss Federal Institute of Technology has four levels of organizing, as shown in Fig. 15. As a POE (Phylogeny, Ontogeny and Epigenesis) type, this embryonic system was developed in analogy with the evolutionary process of biological systems. Furthermore, the system derives from the multi-cellular structure of living organisms which have strong hierarchical organization from molecular to population levels.

Although artificial cell terminology is frequently used in international research, these are not mathematically founded models, nor do they define the cell structure in detail. Therefore, there have been efforts to develop an FPGA-based artificial cell model with a generalized character to reproduce the complex interaction rules inside bio-inspired VLSI hardware systems. According to this model, each FPGA circuit is represented by an artificial cell with four lattices fully capable of operating in a network structure using defined data and control buses. Each lattice of the cell has the same number of input/output bits, configuration structure, and operation functions as shown in Fig. 16a. For network efficiency, these cells are organized in groups of nine artificial cells, as showcased in Fig. 16b.

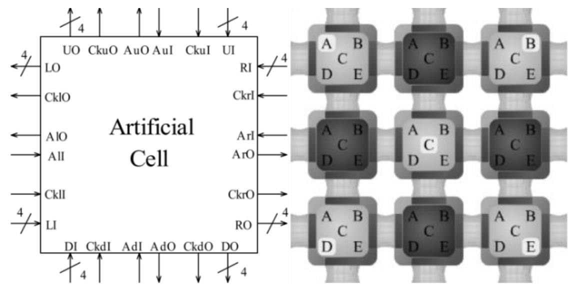

Fig. 16. (a) hardware model of a single FPGA-based artificial cell. Holds four 12-bit lattices capable for full operation in a network structure. (b) model of cell network - macro groups of artificial cells to increase network efficiency [35].

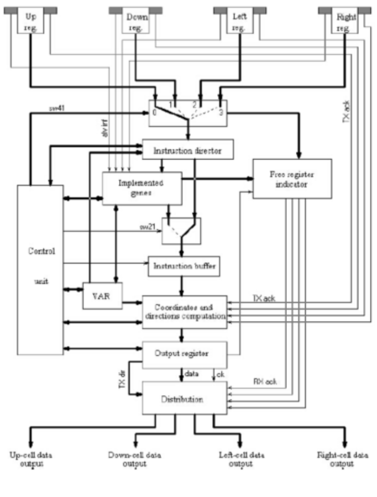

Fig. 17. Top-level block diagram showing the proposed hardware for the artificial cell internal structure. Each artificial cell is connected in a bi-dimensional network, and receive information from four neighboring cells (up, down, left, right). This information is then stored in 4 input registers (shown as Up reg., Down reg., Left reg., Right reg.). Each register is loaded with one instruction from a neighbor cell, and this instruction contains all necessary information from the neighbor cell such as the instruction code, cluster and cell coordinates, and the implemented gene [35].

Each cell contains all genes of the organism, and depending on the situation, only one gene from a cell is active. Figure 17 demonstrates this visually. Each cell has genes A-E, and only the highlighted gene in each cell is active. To produce a network configuration, each artificial cell is connected in a bi-dimensional network so it can receive information from its four neighbor cells (up, down, left, and right). This is then stored in a four-input register, and each register is loaded with one instruction from the adequate neighbor cell. Therefore, each register has a 8x4 configuration which allows full network communication with no restrictions [35]. The registers are read by an instruction director which recognizes the instruction type and decodes the instruction to decide which cell to store the instruction in. If the instruction is addressed to another cell, the instruction is directed to the output register and distribution unit. Otherwise, the instruction is processed inside the current cell and through the implemented genes unit to generate a new instruction for another cell in the network. Figure 16 shows the overall block diagram and connections in the hardware.

Overall, neuromorphic computing is a very versatile hardware that has an incredibly wide applicability. In addition to the use cases discussed in this section, there are many other implementations including:

1) *Cybersecurity*: neuromorphic systems can help detect unusual patterns that could signify cyberattacks or breaches, and the low latency helps minimize these threats quickly [36].

2) *Edge AI*: the low power consumption of neuromorphic architecture helps with the short battery life of devices like smartphones and wearable technology. Adaptability and event-driven nature fit the information processing methods of remote sensors, drones, and other IoT devices.

3) *Robotics*: neuromorphic computing is an adaptable technology, so it can enhance a robot's real-time learning and decision-making skills. It has been used to recognize objects, navigate factory layouts, and operate faster in an assembly line [37].

There are endless opportunities in the world of neuromorphic computing, and the next section discusses potential applications in the near future.

B. *Future Opportunities*

1) Devices

One way to improve upon the current progress of neuromorphic computing is to study alternative devices and their properties. Phase-change memory (PCM) is a device structure that also stores information as conductance, but in a unique way. It alters the atomic configuration of a material like $Ge_2Sb_2Te_5$; by switching between amorphous and crystalline phases, the material's conductance changes [38]. For example, the number of crystallization pulses can slowly update the analog conductance, making it applicable to MAC operations and crossbar array training. It also can retain data for up to 10 years and has a fast write speed.

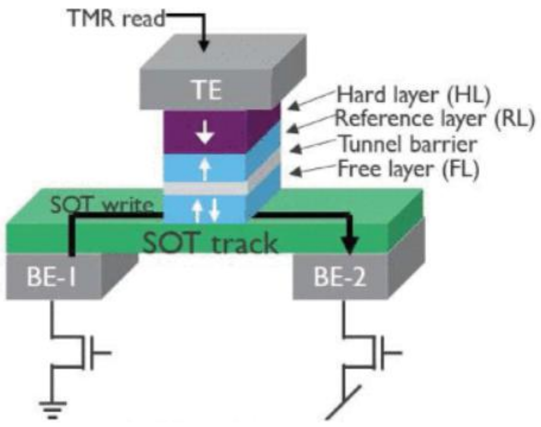

Fig. 18. Spin-orbit-torque (SOT) Device layout [39]. This spintronic has heavy metal terminals with a free layer, tunnel barrier (insulating layer), and reference layer. The free layer switches given a current through the heavy metal, and read out using the tunnel magneto-resistance effect.

Another type of memory is MRAM (magneto resistive RAM). Currently, spin-transfer torque MRAM (STT-MRAM) is already in mass production, but it has a shared read and write path. This means the device needs a higher switching current, limiting the lifespan of the devices. Spin-orbit-torque (SOT)-MRAM (Fig. 18) is an alternative device that has promising qualities. The free layer switches by passing a charge current through the heavy metal layers, and the value is read out using the tunnel magneto-resistance (TMR) effect. This means the SOT MRAM has a different read and write path, and an under 300 ps switching speed [39].

Lastly, the ferroelectric tunnel junction (FTJ) is a device structure consisting of two metal electrodes and a thin ferroelectric film in between. The tunneling conductance through the FTJ has two states: high and low resistance. The resistance can be changed by adjusting the polarization state of the film, done by applying an electric field. In a few studies, ferroelectric transistors have been shown to work in SNNs successfully with STDP [40].

2) *Some Novel Applications*

a) Stochastic learning

A stochastic algorithm for learning involves a visual pattern alternated with random noise. In a study, input data was submitted into an SNN at equal time intervals [41]. At each interval, either a full pattern or random noise with a similar density was submitted. Each pre-synapse that produces a spike had a period in which it cannot produce another spike, like a down time. This guarantees that each synapse must have noise/randomness in between patterns. Originally, the synaptic weights have random amounts of resistance, but after 500 periods, the synaptic weights potentiated the patterns, and the 'background' synapses were depressed. The noise is important because it allows the SNN to 'forget' previous patterns by repressing the background synapses. τ_{learn} is the learning time to potentiate the desired pattern. In Fig. 19, τ_{learn} decreases as the noise density increases because the probability of background noise depression increases with noise density. The trade-off is that the probability of error also increases with noise density, so a

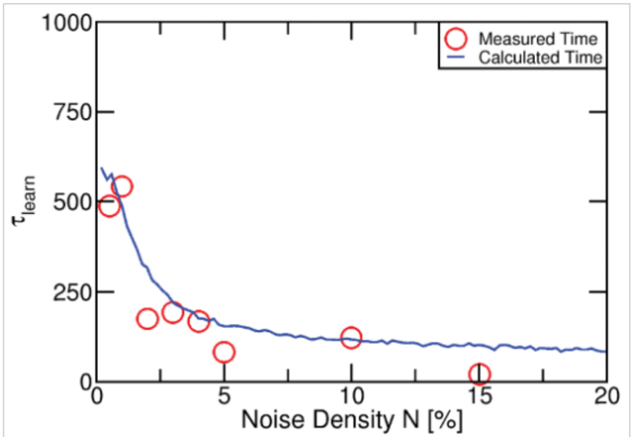

Fig. 19. Learning time vs. noise density for training SNN [41]. While the noise density increases, the learning time decreases exponentially because the probability of background depression increases.

5% noise density is most ideal for optimizing τ_{learn} while limiting error probability [41].

b) Epidemic spread

Some research has been done on using SNNs to model the spread of diseases with more efficiency [42]. The Susceptible-Infected-Removed (SIR) model is a system of differential equations that determines the evolution of the susceptible, infected and recovered/removed populations. In this experiment, the SNN is set up so that each vertex has a voltage that represents the susceptibility of an individual to becoming infected, with a down time representing immunity. The weighted synapses represent social interactions, and therefore determine the rate of the spread of disease. Neurons that have never fired are susceptible, but once their voltage exceeds a spike threshold value, the firing neurons are infected. With a weak social interaction/synaptic weight, a firing weight will not cause someone else to get sick, but a strong social interaction will cause another neuron to fire. Sample tests were run with several factors in mind, like immunity, infectious period, and average contact rate. The results highlight the similarities between a neural network and the spread of disease, which could be used in the future to estimate disease spread given a small sample size or previous data [42]. Newer applications of neuromorphic computing show how the properties of neural networks – like down time or weighted interactions between vertices – can be utilized in several fields.

V. Conclusion

Inspired by the sparse firing neurons of the human brain, neuromorphic computing increases computing efficiency especially in AI applications. By using in-memory computing hardware to train neural networks, the von Neumann bottleneck of data transfer can be avoided. This new architecture is made possible by memristive devices like RRAM, PCM, and newer alternatives. Several complications exist, both on the device level, and with scaling. Despite this, there is a huge effort to research neuromorphic computing because of how energy efficient these systems can be. Other than increasing efficiency for AI computations, additional applications include cybersecurity, stochastic learning, and modeling epidemic spread.

ACKNOWLEDGMENT

K. J. and K. M. thank Arpan De for the helpful feedback, suggestions, and resources.

REFERENCES

[1] K. Dickman, “The future of AI”, LANL, July 2025. [Online]. Available: https://www.lanl.gov/media/publications/1663/1269-neuromorphic-computing

[2] I. Lorina, and N. France, “From computation to consumption: exploring the compute-energy link for training and testing neural networks for SED dystems”, October 2024, arXiv:2409.05080 [Online]. Available: https://arxiv.org/html/2409.05080v1

[3] ComputerScience.GCSE.GURU, “Von Neumann Architecture”, April 2019. [Online]. Available: https://www.computerscience.gcse.guru/theory/von-neumann-architecture

[4] B.Jung, M. Kalcher, M. Marinova, P. Powell, and E. Sakalli, “Neuromorphic Computing - An Overview”, October 2025, arXiv:2510.06721v2 [Online]. Available: https://arxiv.org/html/2510.06721v2#biba.bibx18

[5] D. Ivanov, A. Chexhegov, M.Kiselev, A. Grunin, and D. Larionov, “Neuromorphic artifical intelligence systems”, September 2022, National Library of Medicine, DOI: 10.3389/fnins.2022.959626.

[6] K. Boahen, "A neuromorph's prospectus," in *Computing in Science & Engineering*, vol. 19, no. 2, pp. 14-28, Mar.-Apr. 2017, DOI: 10.1109/MCSE.2017.33.

[7] F. Ottati, “TrueNorth: A Deep Dive Into IBM’s Neuromorphic Chip Design”, March 2023. [Online]. Available: https://open-neuromorphic.org/blog/truenorth-deep-dive-ibm-neuromorphic-chip-design/

[8] A. Amir *et al*., "A Low Power, Fully Event-Based Gesture Recognition System," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017, pp. 7388-7397, doi: 10.1109/CVPR.2017.781.

[9] Open Neuromorphic, “Loihi 2 - Intel”, N.D. [Online]. Available: https://open-neuromorphic.org/neuromorphic-computing/hardware/loihi-2-intel/

[10] Intel, “Taking Neuromorphic Computing to the Next Level with Loihi 2”, https://download.intel.com/newsroom/2021/new-technologies/neuromorphic-computing-loihi-2-brief.pdf

[11] Intel, “Intel Builds World’s Largest Neuromorphic System to Enable More Sustainable AI”, April 2024. [Online]. Available: https://newsroom.intel.com/artificial-intelligence/intel-builds-worlds-largest-neuromorphic-system-to-enable-more-sustainable-ai

[12] L. Myers, “Chinese researchers develop a riderless autonomous bicycle”, February, 2021. [Online]. Available: https://www.designboom.com/technology/chinese-researchers-riderless-autonomous-bicycle-02-15-2021/

[13] H. Chen, H. Li, T. Ma, S. Han, and Q. Zhao, “Biological function simulation in neuromorphic devices: from synapse and neuron to behavior”, March 2023, National Library of Medicine, doi: 10.1080/14686996.2023.2183712

[14] K. Yamazaki, V. Vo-Ho, D. Bulsara, and N. Le, “Spiking neural networks and their applications: A Review”, June 2022, National Library of Medicine, doi: 10.3390/brainsci12070863

[15] C. Lee, S. S. Sarwar, P. Panda, G. Srinivasan, and K. Roy, “Enabling spike-based backpropagation for training deep neural network architectures”, March 2020, arXiv:1903.06379 [Online]. Available: https://arxiv.org/pdf/1903.06379

[16] M. Dampfhoffer, T. Mesquida, A. Valentian and L. Anghel, "Backpropagation-based learning techniques for deep spiking neural networks: a survey," in *IEEE Transactions on Neural Networks andLearning Systems*, vol. 35, no. 9, pp. 11906-11921, Sept. 2024, doi: 10.1109/TNNLS.2023.3263008

[17] J. Chen, J. Li, Y. Li, and X. Miao, “Multiply accumulate operations in memristor crossbar arrays for analog computing”, Chinese Institute of Electronics, vol.42, no. 1, doi: 10.1088/1674-4926/42/1/013104

[18] P. Hess, “Why a decades old architecture decision is impeding the power of AI computing”, February 2025. [Online]. Available: https://research.ibm.com/blog/why-von-neumann-architecture-is-impeding-the-power-of-ai-computing

[19] A. Yesil, F. Gül, and Y. Babacan, ‘Emulator Circuits and Resistive Switching Parameters of Memristor’, Memristor and Memristive Neural Networks. InTech, Apr. 04, 2018. doi: 10.5772/intechopen.71903.

[20] J. Alman, R. Duan, V. V. Williams, Y. Xu, Z. Zu, and R. Zhou, “More assymetry yields faster matrix multiplication”, October 2024, arXiv:2404.16349 [Online]. Available: https://arxiv.org/pdf/2404.16349

[21] H. Assaf, Y. Savaria and M. Sawan, "Vector matrix multiplication using crossbar arrays: a comparative analysis," *2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Bordeaux, France, 2018, pp. 609-612, doi: 10.1109/ICECS.2018.8617942.

[22] S. Zhang, J. Gu, S. Yin, L. Liu and S. Wei, "A multiple-precision multiply and accumulation Design with Multiply-Add Merged Strategy for AI Accelerating," *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*, Tokyo, Japan, 2021, pp. 229-234.

[23] A. Mohammaden, M. Ghoneim, R. Hesham and A. Soltan, "Current-mode multiplier accumulator design using a memristor-transistor crossbar architecture," *2021 3rd Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, Giza, Egypt, 2021, pp. 67-70, doi: 10.1109/NILES53778.2021.9600483.

[24] Peng Gu *et al*., "Technological exploration of RRAM crossbar array for matrix-vector multiplication," *The 20th Asia and South Pacific Design Automation Conference*, Chiba, Japan, 2015, pp. 106-111, doi: 10.1109/ASPDAC.2015.7058989.

[25] Z. Du *et al*., "Neuromorphic accelerators: A comparison between neuroscience and machine-learning approaches," *2015 48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Waikiki, HI, USA, 2015, pp. 494-507, doi: 10.1145/2830772.2830789.

[26] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

[27] P. Hess, “Co-packaged optics deliver high-speed connectivity to supercharge generative AI computing”, December 2024. [Online]. Available:https://research.ibm.com/blog/co-packaged-optics-to-supercharge-generative-ai-computing

[28] S. Raoux and T. J. Ibm, “Chapter 5 - Phase change memory (PCM) materials and devices,” Woodhead Publishing, 2014, pp. 161–199, ISBN: 978-0-85709-803-0

[29] J. Yang and M. Sawan, "Neuromorphic computing chips: challenges and trends," *2025 IEEE International Symposium on Circuits and Systems (ISCAS)*, London, United Kingdom, 2025, pp. 1-5, doi: 10.1109/ISCAS56072.2025.11043979.

[30] K. Roy, A. Jaiswal, P. Panda, “Towards spike-based machine intelligence with neuromorphic computing,” *Nature*, 2019, pp. 607-617, doi: doi.org/10.1038/s41586-019-1677-2 .

[31] R. Patton *et al*., "Neuromorphic computing for scientific Applications," *2022 IEEE/ACM Redefining Scalability for Diversely Heterogeneous Architectures Workshop (RSDHA)*, Dallas, TX, USA, 2022, pp. 22-28, doi: 10.1109/RSDHA56811.2022.00008.

[32] C. Schuman *et al.,* “Evolutionary vs imitation learning for neuromorphic control at the edge”, vol. 2, no.1, January 2022, doi: 10.1088/2634-4386/ac45e7

[33] R. M. Patton et al., "Exascale deep learning to accelerate cancer research," in 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 2019, pp. 1488-1496, doi: 10.1109/BigData47090.2019.9006467.

[34] H. Smith, J. Seekings, M. Mohammadi and R. Zand, "Realtime facial expression recognition: neuromorphic hardware vs. edge AI accelerators," *2023 International Conference on Machine Learning and Applications (ICMLA)*, Jacksonville, FL, USA, 2023, pp. 1547-1552, doi: 10.1109/ICMLA58977.2023.00233.

[35] C. Szasz and V. Chindris, "Bio-inspired hardware systems development and implementation with FPGA-based artificial cell network," *2008 IEEE International Conference on Automation, Quality and Testing, Robotics*, Cluj-Napoca, Romania, 2008, pp. 109-114, doi: 10.1109/AQTR.2008.4588717.

[36] A. L. Eyo, “Neuromorphic computing in cyber security: A paradigm shift towards enhanced threat detection and mitigation medium,” January 2024. [Online]. Available: analystlevy.medium.com/neuromorphic-computing-in-cyber-security-a-paradigm-shift-towards-enhanced-threat-detection-and-c8cad9ea6b7e.

[37] Yang, Yi & Bartolozzi, Chiara & Zhang, Haiyan & Nawrocki, Robert. (2023). Neuromorphic electronics for robotic perception, navigation and control: A survey. Engineering Applications of Artificial Intelligence. 126 Part A. 106838. 10.1016/j.engappai.2023.106838.

[38] G. S. Syed, M. L. Gallo, and A. Sebastian, “Phase-change memory for in-memory computing”, Chemical Reviews, vol. 125, iss. 11, May 2025, doi: 10.1021/acs.chemrev.4c00670.

[39] S. Rao *et al*., "Spin-orbit torque MRAM for ultrafast cache and neuromorphic computing applications," *2023 IEEE International Memory Workshop (IMW)*, Monterey, CA, USA, 2023, pp. 1-4, doi: 10.1109/IMW56887.2023.10145991.

[40] G. Verma, S. Soni, A. Nisar, and B. Kaushik, “Multi-bit MRAM based high performance neuromorphic accelerator for image classification”, vol. 4 no. 1, March 2024, doi: 10.1088/2634-4386/ad2afa

[41] G. Pedretti *et al*., "Stochastic learning in neuromorphic hardware via spike timing dependent plasticity with RRAM synapses," in *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 1, pp. 77-85, March 2018, doi: 10.1109/JETCAS.2017.2773124.

[42] K. Hamilton, P. Date, B. Kay, C. Schuman, “Modeling epidemic spread with spike-based models,” *International Conference on Neuromorphic Systems 2020,* Tennessee, USA, 2020, doi: https://doi.org/10.1145/3407197.3407219.