Chapter

Brain-Inspired Spiking Neural Networks

Khadeer Ahmed

Abstract

Brain is a very efficient computing system. It performs very complex tasks while occupying about 2 liters of volume and consuming very little energy. The computation tasks are performed by special cells in the brain called neurons. They compute using electrical pulses and exchange information between them through chemicals called neurotransmitters. With this as inspiration, there are several compute models which exist today trying to exploit the inherent efficiencies demonstrated by nature. The compute models representing spiking neural networks (SNNs) are biologically plausible, hence are used to study and understand the workings of brain and nervous system. More importantly, they are used to solve a wide variety of problems in the field of artificial intelligence (AI). They are uniquely suited to model temporal and spatio-temporal data paradigms. This chapter explores the fundamental concepts of SNNs, few of the popular neuron models, how the information is represented, learning methodologies, and state of the art platforms for implementing and evaluating SNNs along with a discussion on their applications and broader role in the field of AI and data networks.

Keywords: spiking neural networks, spike timing dependent plasticity, neuomorphic computing, artificial intelligence, low power, supervised learning, unsupervised learning, spatio-temporal learning, neuron models, spike encoding, winner take all, stigmergy

1. Introduction

Nature has provided innumerable examples of very efficient solutions to complex problems with seemingly simple rules. With these as inspiration, many engineering problems are tackled using bioinspired techniques. A few of bioinspired techniques are evolutionary and genetic algorithms, stigmergy, hidden Markov models, belief networks, neural networks, etc. These are applicable in a wide variety of domains from robotics [1], communication systems, routing [2], building construction [3], scheduling, optimization, machine intelligence, etc. The brain is a very efficient computing element capable of performing complex tasks. This is possible due to massively parallel computation being performed by the vast number of cells called neurons in the brain while consuming very little energy. This has inspired a domain of algorithms and techniques called artificial intelligence (AI) where machines are programmed to learn and then solve complex tasks. The recent advances in high performance computing and theoretical advances into statistical learning methodologies have enabled a widespread use of AI techniques for tasks such as pattern recognition, natural language understanding, speech recognition, computer vision, odor recognition, machine translation, medical diagnosis, gaming, autonomous driving, path planning, autonomous robots, financial market modeling and the list goes on. Solving these kinds of problems with efficiency is not possible with the traditional computing paradigms. These algorithms are mimicking biology or are inspired from biology to tackle the above problems. For example, it is not humanly possible to have traditional software program coded to classify an image of a simple object such as a cup with reasonable accuracy, considering the innumerable variations available in terms of shape, size, color, etc. However, this is a trivial task for a human being as our brains learn to identify the salient features of an object. The inner working of the brains, especially the way it processes information is the inspiration behind a class of AI techniques called neural networks.

AI requires a large amount of compute power while churning through massive amounts of data. Today's real-world tasks require different sets of AI models with different modalities to interact with each other, hence needing a large pipeline with complex data dependencies. Training is time-consuming, while needing efficient multi-accelerator parallelization. Even with such advances we are nowhere close to the compute power or the efficiency of a human brain. Human brain is still a mystery and is a very actively researched topic. Several neuron models are proposed to mimic various aspects of how the brain works with the limited understand we have up till now.

Spiking neural networks (SNNs) are networks made up of interconnected computing elements called neurons. SNNs try to mimic biology to incorporate the efficiencies found in nature. These neurons use spikes to communicate with each other. SNNs are third generation of neural networks [4] and are gaining popularity due to its potential for very low energy dissipation due to their event-driven and asynchronous operation. SNNs are also interesting because of their ability learn in a distributed way using a technique called Spike Timing Dependent Plasticity (STDP) learning [5]. STDP relies on sparsely encoded spiking information among local neurons. SNNs are capable of learning rich spatio-temporal information [6]. In principle, SNNs can be fault tolerant due to its ability to re-learn and adapt the connections with other neurons, akin to how the brains learn. Also SNNs can natively interface with specialized hardware sensors which mimic biological vision (Dynamic Vision Sensor) and hearing (Dynamic Audio Sensor) [7] as they directly transduce sensory information to spikes.

In the rest of the chapter, a brief introduction on neuron biology and artificial neuron models is presented, followed by discussion on information representation as spikes, different learning methodologies, tools, and platforms available for simulating and implementing SNNs and finally few case studies as examples of SNN usage.

2. Neuron models

In this section, a brief overview of the biological neuron processes is provided to understand the inference and learning dynamics of SNNs. A few popular neuron models are discussed at a high level to make the reader aware of the diversity of such research and its use in SNNs.

2.1 Biological neuron

Complex living organisms have specialized cells called neurons, which are the fundamental unit of central nervous system. Neurons can transmit and receive signals in the form of electrical impulses. In a human brain, there are an estimated 200 billion neurons. Also, there are several different types of neurons in the body. In general, a neuron consists of a cell body or soma consisting of cell machinery, nucleus, dendrites, and an axon as shown in **Figure 1**.

The dendrites receive information from other neurons, and this causes a voltage buildup on the cell body. When this membrane potential reaches a certain threshold, an electrical impulse is generated, and the axon transmits this spike away from the cell body to other neurons. After a spike is generated, the neuron returns to a lower potential called resting potential. Also, immediately after a spike is generated, the neuron cannot generate another spike for a short duration called the refractory period. The axon terminates at axon terminals which interface with dendrites of other neurons; this is called as a synapse. A synapse is connection between a pre synaptic neuron (which generates electrical impulse) and a postsynaptic neuron (receives the spike information) as shown in **Figure 1**. The synapse is not a direct connection, instead it consists of a gap called synaptic cleft as shown in **Figure 2**. Discussion about astrocyte cells is presented later in Section 4.5.

When an electrical impulse reaches the synapse, the presynaptic neuron releases certain chemicals called neurotransmitters into the synaptic cleft. The postsynaptic neuron picks up these neurotransmitters eventually causing the postsynaptic neurons membrane potential to either increase or decrease. The brain learns by strengthening or weakening the existing synaptic connections or by making new synaptic connections or dissolving those which are no longer needed. In this way, the synapses make the brain plastic and provide the ability to learn. Also, the strength of the synapse also matters for learning as it can modulate the amount of neurotransmitters released in the synaptic cleft resulting in a stronger or weaker synapse and depending on the type

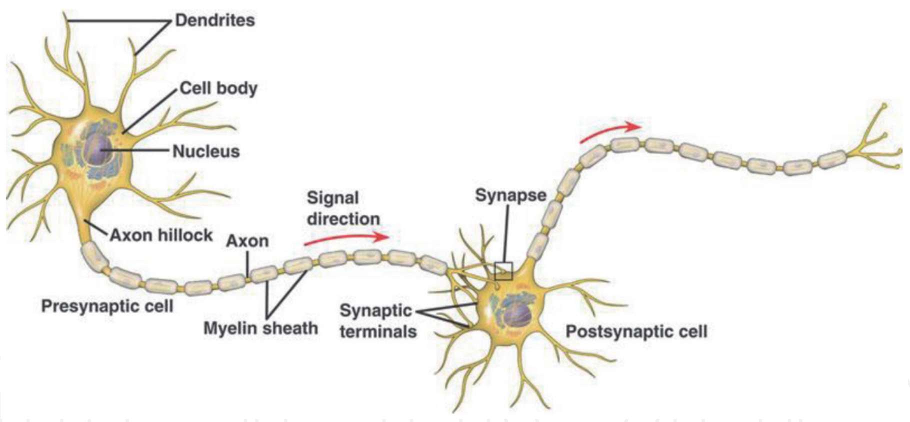

Figure 1.
Neurons (by unknown author, licensed under CC BY-SA https://creativecommons.Org/licenses/by-sa/3.0/).

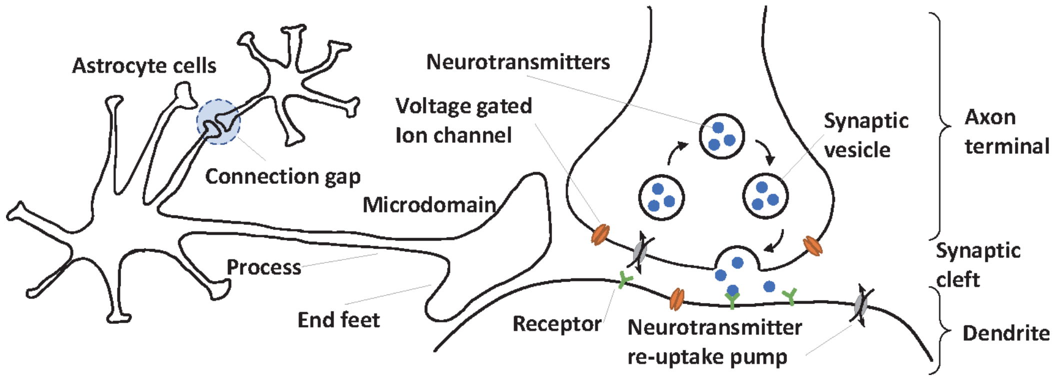

Figure 2.
Neuronal synapse along with astrocyte cells (author created).

of neurotransmitters released, the synapse can be excitatory or inhibitory. An excitatory synapse is one which would increase the membrane potential of the post synaptic neuron; conversely, an inhibitory synapse would decrease the membrane potential. Based on these fundamental concepts, several researchers have proposed various neuron models over the decades. We do not yet fully understand the inner workings of brains and is still an active field of research. New neuron models are being proposed frequently as our understanding of biology increases. A few neuron models are listed below, followed by an overview of select models.

2.2 Artificial neuron models

Some of the models proposed try to mimic biology for the purpose of understanding and modeling neuro-physiological processes and some models more oriented toward computing purposes. A few of neuron models to consider are McCulloch and Pitts [8], Hodgkin-Huxley [9], Perceptron [10], Izhikevich [11] Integrate and fire [12], Leaky integrate-and-fire [13], Quadratic integrate-and-fire [14], Exponential integrate-and-fire [15], Generalized integrate-and-fire [16], Time-varying integrate-and-fire model [17], Integrate-and-fire or burst [18], Resonate-and-fire [19], and Bayesian neuron model [20].

2.3 Hodgkin and Huxley neuron model

Hodgkin and Huxley [9] studied the giant axon of the squid and found currents induced by different types of ions namely sodium ions, potassium ions, and leakage current due to calcium ions. The cell consists of voltage-dependent ion channels which regulate the concentration of these ions across the cell membrane. For the sake of simplicity, at a high level, the total membrane current is the sum of current induced by membrane capacitance and the ion channel currents as shown in Eq. (1), where I_i is the ionic current density, V is the membrane potential, C_M is the membrane capacitence per unit area, t is time, and I_{Na}, I_K, I_l are the sodium, potasium, and leakage current induced by calcium and other ions.

$$I = I_c + I_i \tag{1}$$

$$I_c = C_M \frac{\mathrm{dV}}{\mathrm{d}t} \tag{2}$$

$$I_i = I_{Na} + I_K + I_l \tag{3}$$

They also describe gating variables to control the ion channels and the resting potential of the cell. When the membrane potential increases significantly above the resting potential, the gating variable activates and then deactivates the channels resulting in a spike. This is a very simplified model and has several limitations [21].

2.4 Izhikevich neuron model

Izhikevich neuron model [11] is more biologically plausible as shown in equations below.

$$\begin{aligned} &v' = 0.04v^2 + 5v + 140 - u + I \\ &u' = a(bv - u) \\ &If\ v \geq 30mV, then \begin{cases} v = c \\ u = u + d \end{cases} \end{aligned} \tag{4}$$

Where v is the membrane potential, u is the recovery variable, I is the current, and a, b, c and d are neuron parameters. Various biologically plausible firing patterns can be modeled using this model as shown in **Figure 3**.

Over time, if a biological neuron does not spike, then any potential builtup would dissipate. This phenomenon is modeled by several variations of Leaky Integrate and Fire (LIF) models. LIF neuron model is very popular due to its ease of implementation as a software model and for developing dedicated hardware models. Digital hardware implementation is more popular than the analog variants, again due to its simplicity of design, fabrication, and scalability.

2.5 Discrete leaky integrate and fire

A typical generic LIF model adapted for discrete implementation [22] is represented as:

Synaptic integration

$$V(t) = V(t-1) + \sum_{i=0}^{N-1} x_i(t) s_i \tag{5}$$

Leak integration

$$V(t) = V(t) - \lambda \tag{6}$$

Threshold, fire and reset

$$\text{If } V(t) \geq \alpha \text{ then Spike and } V(t) = R \tag{7}$$

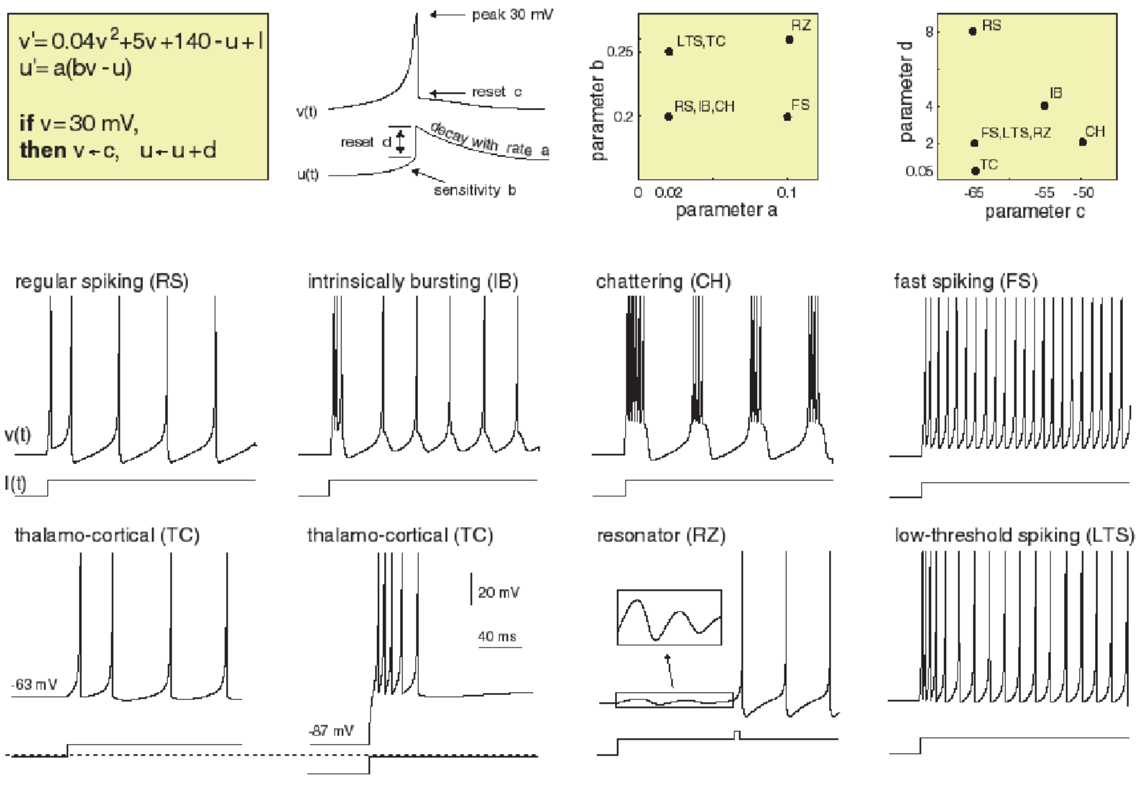

Figure 3.
Izhikevich neuron model [11].

Where $V(t)$ is the membrane potential, t is discrete time step, N is the number of synapses, $x_i(t)$ is the i^{th} synapse, s_i synaptic weight of i^{th} synapse, λ is leak, α is spiking threshold, and R is the resting potential. A spike value is 1, otherwise 0. Whenever, a spike occurs on a synapse $x(t)$ then the synaptic weight gets accumulated increasing the membrane potential. Every time step a leak is applied and finally when the membrane potential reaches a threshold α, the neuron spikes and the membrane potential is reset to a resting value R.

2.6 Bayesian neuron model

Bayesian neuron (BN) model is proposed in [20]. BN model is a stochastic neuron model. When the membrane potential reaches the threshold a BN model fires a spike stochastically. It generates a spike based on a Poisson process where neuron Z fires at time t with a probability proportional to its membrane potential at time t. The membrane potential $u(t)$ is computed as:

$$u(t) = w_0 + \sum\nolimits_{i=1}^{n} w_i y_i(t) \tag{8}$$

Where the weight of the synapse between i^{th} presynaptic neuron y_i and Z is w_i. If y_i fires a spike at time t, then $y_i(t)$ is 1. The intrinsic excitability is w_0. The firing probability of this stochastic neuron model depends exponentially on the membrane potential $u(t)$ as:

$$probability(Z\,fires\;at\;time\;t) \propto \exp\left(u(t)\right) \tag{9}$$

To generate a Poisson process with time-varying rate $\lambda(t)$, the *Time-Rescaling Theorem* is used. According to this theorem, when spike arrival times v_k follow a Poisson process of instantaneous rate $\lambda(t)$, the time-scaled random variable $\Lambda_k = \int_0^{v_k} \lambda(v)dv$ follows a homogeneous Poisson process with unit rate. Then the interarrival time τ_k satisfies exponential distribution with unit rate.

$$\tau_k = \Lambda_k - \Lambda_{k-1} = \int_{v_{k-1}}^{v_k} \lambda(v)dv \tag{10}$$

τ_k represents a generated random variable satisfying an exponential distribution with unit rate. v_k is the next time to spike. As shown in Eq. (10), the instantaneous rates from Eq. (8) is cumulated until the integral values is greater than or equal to τ_k. At this point of time, a spike is generated as it implies that the interspike interval has passed. Poisson spiking behavior is achieved in this way reflecting the state of the neuron. Other stochastic neuron behaviors can be easily constructed by stochastically varying different parameters of the model.

3. Information representation

SNNs understand the language of spikes, and it is necessary to decide what is the best possible way to represent real-world data to achieve best possible training of the network and efficient inference. Different coding techniques model different aspects of input spectrum. Some of the spike coding techniques are described below to get an intuition of signal representation using spikes.

3.1 Rate coding

With rate coded spike trains, the information is encoded in the number of spikes over a specified temporal window. The firing rate ν_k, over k trials is shown in Eq. (11), where n_k^{sp} is the number of spikes over k trials over a temporal window T and is the number of trials [23].

$$\nu_k = \frac{n_k^{sp}}{T} \tag{11}$$

Evidence of rate coding is experimentally shown in sensory and motor systems [24]. The number of spikes emitted by the receptor neuron increases with the force applied to the muscle.

If the rate ν is defined via a spike count over a temporal window of duration T, the exact firing time of a spike does not matter [23]. We can define it as a Poisson process where spikes events are stochastic and independent of each other with an instantaneous firing rate ν. In a homogeneous Poisson process, the probability to find a spike in a short interval Δt is

$$P_F(t; t + \Delta t) = \nu \Delta t \tag{12}$$

Therefore, the instantaneous firing rate is

$$\nu = \lim_{\Delta t \to 0} \frac{P_F(t; t + \Delta t)}{\Delta t} \tag{13}$$

The expected number of spikes for the temporal window T is

$$\langle n^{sp} \rangle = \nu T \tag{14}$$

To summarize, the experimental procedure of counting spikes over a time T and dividing by T gives an empirical estimate of the rate ν of the Poisson process. When recording an experiment over several trials, the spike response can be represented via a Peri-Stimulus-Time Histogram (PSTH) with bin width Δt as shown in **Figure 4**. The number of spikes $n_k(t; t + \Delta t)$ summed over all repetitions K of the experiment is a measure of the typical activity of the neuron between time t and $t + \Delta t$. Therefore, the spike density can be represented as shown in Eq. (15).

$$\rho = \frac{1}{\Delta t} \frac{n_k(t; t + \Delta t)}{K} \tag{15}$$

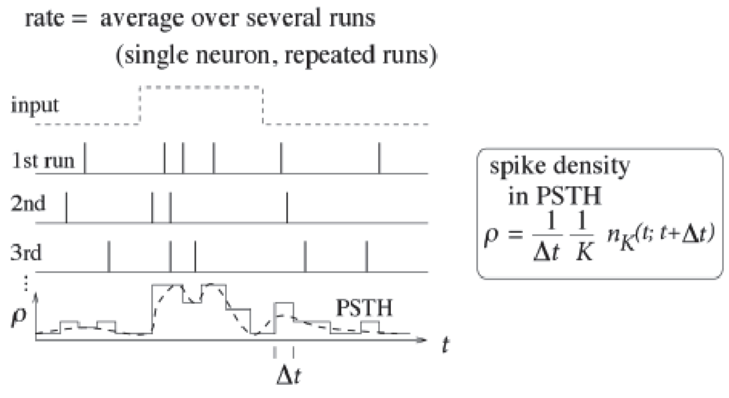
rate = average over several runs
(single neuron, repeated runs)
input
1st run
2nd
3rd
ρ
PSTH
t
Δt
spike density in PSTH
$\rho = \frac{1}{\Delta t}\frac{1}{K} n_K(t; t+\Delta t)$

Figure 4.
The Peri-stimulus-time histogram and the average time-dependent firing rate [23].

A spike train $S(t)$ is a sum of δ functions with a spike occurring at t_s.

$$S(t) = \sum_s \delta(t - t_s) \tag{16}$$

The instantaneous firing rate is the expectation over trials.

$$v(t) = \langle s(t) \rangle \tag{17}$$

An empirical estimate of the instantaneous firing rate can be deduced as shown in Eq. (18). It implies that the PSTH as described above represents the instantaneous firing rate.

$$\nu(t) = \frac{1}{K\Delta t} \sum_{k=1}^{K} n_k^{sp}(t) \tag{18}$$

The average firing rate can be computed for a single neuron, or for a population of neurons representing a class over a single run or over several trials. Rate coding over a time window is suitable for representing the strength of stimulation. On the other hand, population-based rate coding could convey the same information by employing several neurons in a shorter temporal window. The latter trades quick response over a number of neurons. There is evidence of Purkinje neurons demonstrating information coding which is not just firing rate but also the timing and duration of nonfiring, quiescent periods [25, 26].

3.2 Temporal coding

If the time of spike occurrence in a temporal window carries information, then such coding is referred to as temporal coding. In such coding schemes the quiescent periods and the spiking time both carry information. There are several evidences in biology demonstrating this behavior [27, 28]. A typical temporal code is shown in **Figure 5A**, where the time interval of spike to start of stimulus caries information. These are sometimes referred to as pulse codes. Another variation is Rank Order Coding, which uses the relative timing of spikes across a population of cells. Rank order codes look at time to spike across the neuron population and a rank order can

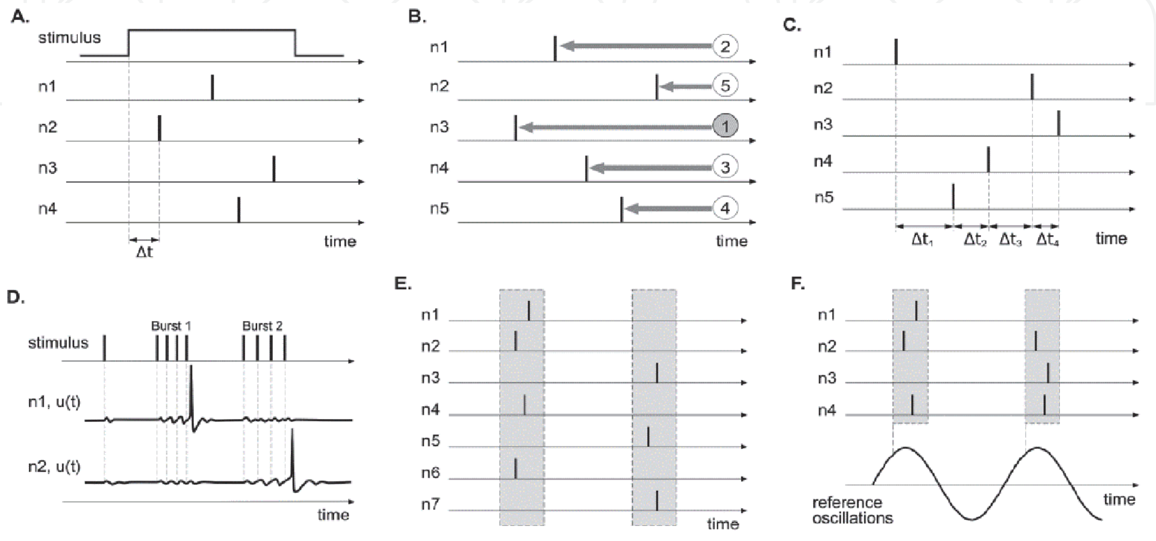

Figure 5.
Different strategies for information coding with spikes (refer to [29] for details). (A) Time to first spike coding (B) rank order coding (C) latency coding (D) resonant burst coding (E) synchrony coding (F) phase coding.

be implied from the firing order among the neurons in the population as described in **Figure 5B**.

There is evidence suggesting that simple temporal averaging of firing rate is too simplistic to model neuronal circuits in the brain [30]. To address some of the shortcomings, several derivations of coding schemes based on different combinations of above concepts are widely used. Few of the common schemes and some task specific coding schemes are Rate code, Time to spike code, Time-to-first-spike: Latency code [31], Reverse time to spike code, Weighted spike code [32], Burst code [33], Population code, Population rate, Rank order code [34], Phase-of-firing code [35, 36], Place code [37], etc. **Figure 5** summarizes a few coding strategies. These coding schemes require appropriate algorithms for converting real-world data to spikes and vice versa. A few common conversion techniques are discussed in the next section.

3.3 Spike transduction

SNNs understand the language of spikes; therefore, we must transform the real-world data to appropriate spike representation and subsequently transform the output spikes to real-world formats for human consumption. There are several encoding and decoding algorithms available to achieve this goal. Several heuristics are also employed. Some of the coding techniques mentioned above infer a specific coding/decoding scheme. Based on the nature of application (such as images, audio, video, financial data, user activity data), one must choose which is the best approach.

Image pixel values are binned and proportional firing rates are assigned to different neurons in the receptive fields for each pixel neuron, hence generating random process with rate coding [38]. Since spikes have no polarity positive and negative spike, subchannels can be used to represent richer encoding of data. In threshold-based schemes, a spike is generated when the input signal intensity crosses a threshold. Real numbers are compared against different thresholds, and positive and negative spikes are produced accordingly which are rate coded [39]. BSA algorithm for encoding and decoding [40] is used for modeling brain-machine interfaces and neurological processes in the brain. The work presented by the authors of [41] provides details on step-forward (SF), and moving-window (MW) encoding schemes. In SF scheme, a baseline $B(t)$ intensity for the input signal is set and a positive spike is generated if the intensity is above the baseline by the threshold $B(t) + Th$ amount and the baseline is updated to this new value $B(t) = B(t-1) + Th$. Conversely a negative spike is generated if the signal intensity is below $B(t) - Th$ and the new baseline is adjusted as $B(t) = B(t-1) - Th$. MW scheme is like SF scheme except that the baseline is set based on the mean of signal intensities. These schemes are suitable for encoding continuous value signals. The above examples are only a limited set of algorithms out of a vast majority of methods to convert diverse signal formats to spikes.

4. Learning principles for SNN

Hebb postulated that synaptic efficacy increases from a presynaptic neuron if it repeatedly assists the post synaptic neuron [42]. This forms the fundamentals of STDP rule for learning. STDP mimics biology where a synapse is strengthened when a presynaptic spike occurs before a post synaptic spike in close intervals, this is called Long-Term Potentiation (LTP). On the other hand, the synapse is weakened if the post synaptic neuron fires before the presynaptic neuron in close intervals. This is called as Long-Term Depression (LTD). In biology neurons are highly selective due to lateral inhibition. This allows for them to learn discriminatory and unique features in an unsupervised manner leading to an emergent Winner Take All (WTA) behavior. Apart from this the biological system demonstrates homeostasis to maintain overall stability. These are key principles in SNN modeling. There are several ways to achieve WTA and homeostasis behavior, some directly modify the neuron state, others use neural circuits. One such example with a scalable neural circuit [43] is shown in **Figure 6**. A WTA network consists of inhibitor neurons suppressing the activation of other lateral symbol neurons as shown in **Figure 6(a)**. To assist in homeostasis a normalization of the excitations of one neural circuit compared to others can be achieved using a Normalized Winner Take All (NWTA) network as shown in **Figure 6(b)**. Where an upper limit (UL) neuron uniformly inhibits all symbol neurons if they are firing beyond a desirable high threshold. On the contrary if the symbol neurons are firing below a desired low threshold, then the lower limit (LL) neuron triggers an excitor (Ex) neuron to uniformly boost the firing rate of all symbol neurons. In this manner all independent neural circuits within an SNN fire in the dynamic range of excitations of the overall network. Both hard and soft WTA behavior can be achieved based on the amount of inhibition generated. In Hard WTA only one symbol neuron is active whereas in soft WTA more than one symbol neuron is active providing richer context.

SNNs can learn in both unsupervised and supervised modes. WTA concepts are essential part of unsupervised learning as the neuron with highest excitation inhibits the lateral neurons the strongest hence enabling it to preferentially pick up unique features. Unsupervised learning is possible by employing a teacher signal which excites the specific neurons to fire thereby allowing it to learn the features represented by the input signal. STDP based learning has its advantages of being able to model spatio-tempotal dynamics. Where the spatial component refers to localized activity/learning and temporal component refers to additional information representation by the spike intervals along the time axis. With the constant advances in SNN research, native STDP based rules are catching up to the more popular backpropagation-based learning methods used in Artificial Neural Networks (ANN). STDP lends itself for efficient localized and distributed learning, which is a huge advantage over other learning methods. Also SNNs can be adapted to model memories in the form of Long Short-Term Memory networks [39] which shows that recurrent learning behavior is also possible. The following sub-sections discus few learning rules used in training SNNs along with a brief introduced to backpropagation-based learning.

4.1 Classic STDP rule

A classic STDP rule [44] is shown in **Figure 7**. The STDP curve tries to approximate experimentally observed behavior.

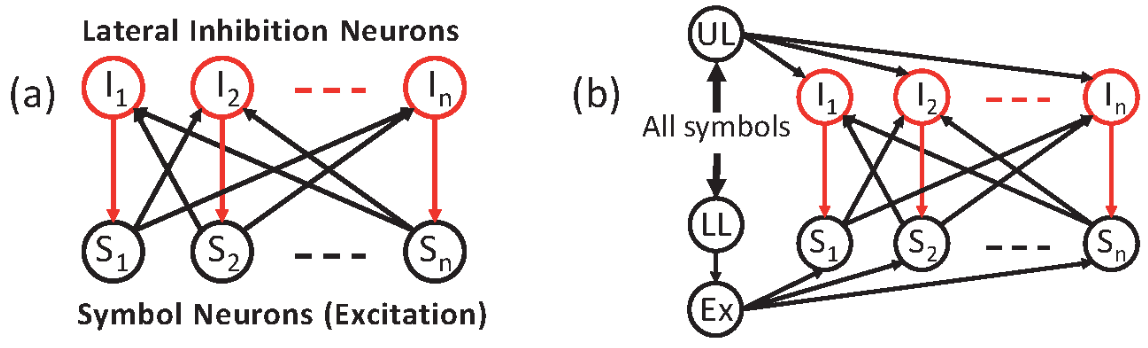

Figure 6.
(a) Winner take all network (b) normalized winner take all network [43].

Here ΔW is the weight update plotted against $\Delta t = t_{pre} - t_{post}$ representing the interval between the presynaptic and post synaptic spike. This approximation is represented in Eq. (19)

$$\Delta W = \begin{cases} a^{+} \exp\left(\dfrac{t_{pre} - t_{post}}{\tau^{+}}\right) if\ t_{pre} \le t_{post}\ (LTP) \\ -a^{-} \exp\left(-\dfrac{t_{pre} - t_{post}}{\tau^{-}}\right) if\ t_{pre} > t_{post}\ (LTD) \end{cases} \tag{19}$$

Where a^{+}, a^{-} are the learning rates and τ^{+}, τ^{-} are the time constants for LTP and LTD, respectively. There are several variations of the STDP curves available in the literature and the reader is encouraged to explore this topic further.

4.2 Simplified stable STDP rule with efficient hardware model

There are two broad categorizations of STDP rules, additive and multiplicative STDP [38]. Multiplicative rule tends to be more stable than additive rule. In additive rules the weight changes are independent of current weight and requires additional constraints to keep the values in operating bounds. These weight changes however produce bimodal distribution resulting in strong competition. In multiplicative rule presented in [38], the weight change is inversely proportional to the current weight making it inherently stable and resulting in a unimodal distribution. This distribution lacks synaptic competition which is desirable for learning discriminative features. For such rules, competition must be introduced in a different method. The stable multiplicative rule is further explored below and simplified for efficient implementation. Here the STDP rule is modeled such that weight change of a synapse has an exponential dependence on its current weight as shown in **Figure 8** (a). Update for the weight w_i of ith synapse of the neuron is calculated as below.

If

$$t_{post} - t_{pre} < \tau_{LTP} \tag{20}$$

then,

$$\Delta w_i = \eta_{LTP} e^{-w_i}, w_i = w_i + \Delta w_i$$

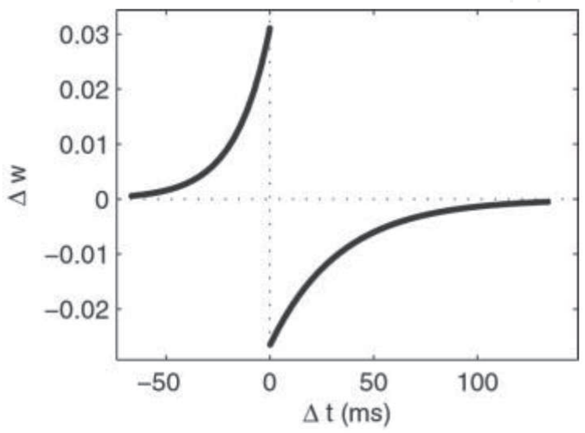

Figure 7.
Classic STDP curve [44].

If

$$t_{post} - t_{pre} > \tau_{LTP} \text{ or } t_{pre} - t_{post} < \tau_{LTD}$$

then,

$$\Delta \mathrm{w_i} = \eta_{\mathrm{LTD}} e^{\mathrm{w_i}}, \mathrm{w_i} = \mathrm{w_i} - \Delta \mathrm{w_i} \tag{21}$$

Where t_{post} and t_{pre} are the pre and post-synaptic neuron spiking time steps, τ_{LTP} and τ_{LTD} are the LTP and LTD window and η_{LTP} and η_{LTD} are the LTP and LTD learning rates respectively. Plasticity is implemented with LTP and LTD windows as shown in **Figure 8** (b). This rule is called as Exp rule.

The Exp STDP rule requires an exponential and a multiplication operation for both LTP and LTD for each synapse. From the perspective of efficient digital hardware implementation these are expensive operations in terms of circuit area and computation time. Quantized 2-power shift rule (Q2PS), which approximates the Exp rule in Eq. (20) and Eq. (21) by removing both multiplication and exponential. The approximation is summarized in Eq. (22) and Eq. (23).

If

$$t_{post} - t_{pre} < \tau_{LTP}$$
$$\Delta w_i = \eta_{LTP} 2^{-w_i} = 2^{\eta'_{LTP} - w_i} \tag{22}$$

If

$$t_{post} - t_{pre} > \tau_{LTP} \text{ or } t_{pre} - t_{post} < \tau_{LTD}$$
$$\Delta w_i = \eta_{LTD} 2^{w_i} = 2^{\eta'_{LTD} + w_i} \tag{23}$$

where $\eta'_{LTP} = \log_2 \eta_{LTP}$ and $\eta'_{LTD} = \log_2 \eta_{LTD}$. Let $Q = \eta'_{LTP} - w_i$ for LTP and $Q = \eta'_{LTD} + w_i$ for LTD. Let $\overline{Q}$ be the quantization of Q through priority encoding. Priority encoding compresses a binary representation of a number to value with only the most significant bit being active as rest of the active bits have no priority. For example, the binary representation of $Q = 13$ is 1101 and the priority encoded value is 1000, hence $\overline{Q} = 8$. Based on this quantization method, the synaptic weight change can be easily computed by left shifting 1 by $\overline{Q}$ or right shifting if negative as shown in Eq. (24).

$$\Delta w_i = \begin{cases} 1 \ll |\overline{Q}|, \text{if } \overline{Q} > 0 \\ 1 \gg |\overline{Q}|, \text{if } \overline{Q} < 0 \end{cases} \tag{24}$$

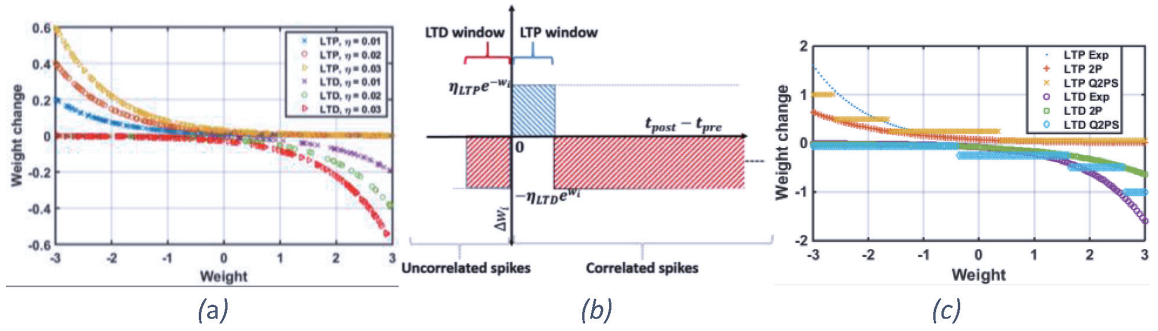

Figure 8.
(a) Current weight vs weight change for learning rates (b) STDP windows (c) Comparison of Exp, 2P and Q2PS STDP rules [38].

where $\ll$ and $\gg$ represent binary shift left and shift right operations, respectively. This approximation allows implementation of the STDP rule presented in Eq. (20) and Eq. (21) on digital hardware by using a priority encoder, negligibly small lookup to determine $|\overline{Q}|$ from the encoded value, barrel shifter and an adder circuit. Please note that, based on Eq. (22) and Eq. (23), Δw_i should be calculated as $2^{\overline{Q}}$, which can be obtained by shifting value 1 by $\overline{Q}$. **Figure 8** (c) compares the Δw_i calculated using the Exp, 2P and Q2PS rules, with a learning rate of 0.08 for all the cases. Here 2P rule is same as Q2PS rule except that 2 is raised to the power of Q. As we can see, the Q2PS rule provides multi-level quantization, which enables similar quality of trained weights even with approximations when compared to Exp rule.

4.3 Overview of learning in artificial neural networks

With the tremendous advances in the field of ANNs, a growing body of research is available on various statistical learning algorithms. ANNs are inspired by biology but they do not mimic it. ANNs are made up of artificial neuron models specifically tuned for compute purposes and model a biological neuron at a very abstract level. An artificial neuron computes weighted sum of input signals and then an activation function computes the neuron output. In these networks' neurons transmit signals as real numbers. ANNs compute inference by transmitting the neuron signals in the forward direction. The learning happens usually via a method called Backpropagation. This algorithm computes the gradients based on the error signal produced by a cost function and propagates it back for each layer of neurons in the neural network. The weight updates are usually made using gradient descent algorithms. There are many flavors of gradient descent algorithms available in the literature. For back propagation to work the activation function must be differentiable. Unlike SNNs, where a spike is not differentiable. In general, ANNs have proven to be very effective in tackling a wide variety of problems. Using these algorithms as inspiration several modified STDP rules have been researched, one among them is discussed below. This overview is a very high-level introduction to some of the terminology required to understand the following section. The reader is encouraged to explore further on this topic.

4.4 Backpropagation-STDP

The Backpropagation-STDP (BP STDP) [45] algorithm uses the number of spikes in a spike trains as an approximation for the real value of an artificial neurons excitation. They also divide the time interval into sub-intervals such that each sub-interval contains zero or one spike.

In supervised training, the weight adjustment is governed by the STDP model shown in Eq. (25) and Eq. (26), in conjunction with a teacher signal. The teacher signal when applied to target neurons undergo weight change based on STDP and non-target neurons undergo weight changes based on anti-STDP. Anti-STDP is the opposite of STDP where LTP and LTD equations are swapped. Target neurons are identified by spike trains with maximum spike frequency (β) and non-target neurons are silent. The expected output spike trains z, are tagged with their input labels. Eq. (25) represents the weight change for a desired spike pattern $z_i(t)$ for the output layer neurons.

$$\Delta w_{ih}(t) = \mu \xi_i(t) \sum_{t'=t-\epsilon}^{t} s_h(t') \tag{25}$$

$$\xi_i(t) = \begin{cases} 1, z_i(t) = 1, r_i \neq 1 \ [t-\epsilon, t] \\ -1, z_i(t) = 0, r_i = 1 \ [t-\epsilon, t] \\ 0, otherwise \end{cases} \tag{26}$$

A target neuron would generate a spike $z_i(t) = 1$ and non-target neurons would remain silent $z_i(t) = 0$. Based on the expected output spike train target neuron should fire within the short STDP window $[t - \epsilon, t]$. Based on the presynaptic activity usually zero or one spike in the STDP window, the synaptic weights are increased proportionally. The presynaptic activity is the count of spikes in the $[t - \epsilon, t]$ interval denoted as $\sum_{t'=t-\epsilon}^{t} s_h(t')$. On the other hand, the non-target neurons upon firing undergo weight depression in the same way. The difference between the desired spike pattern and output spike pattern is used as the guide for identifying target neurons and non-target neurons as the backpropagation rule. Same methodology is used for each layer while back propagating. Among the several learning methods inspired by ANN algorithms a few use strategies where the ANN is trained in its native form and tuned based on a shadow SNN and finally use those adapted weights on SNN for inference.

4.5 Stigmergy assisted learning

Stigmergy is a methodology where several independent agents produce an emergent behavior through indirect interaction among themselves. This is facilitated with the help of asynchronous communication through traces left in the environment by individual agents. Stigmergy has been observed in nature and widely researched upon especially in insect colonies, these principles have been applied towards solving various engineering problems. Recent advances in neuroscience have shown evidence of another type of cells called astrocytes working in tandem with neurons to regulate the behavior of the central nervous system [46]. Astrocytes are star shaped cells with several branches called as processes. The end of these branches called as end feet interface with a synapse by wrapping around it creating a region around the synaptic cleft called as microdomain as shown in **Figure 2**. Astrocytes also interface the neurons apart from the synapse providing a closed loop feedback mechanism. They also interface with other astrocytes like a synapse, instead this is called as a gap junction. Gap junction facilitates communication between astrocyte cells only through chemical means. Astrocytes are functionally very diverse and play a very important role, only a high-level concept with limited detail is introduced for understanding of relevant discussion. With the help of calcium ions as a signaling mechanism along with the help of neurotransmitters the astrocytes help regulate the efficiency of synaptic transmission. These cells play a critical role in maintaining homeostasis, modulating LTP, LTD and structural plasticity in the brain.

Spiking activity results in release of neurotransmitters and change in concentration among different ions in the microdomain and extra cellular space. These

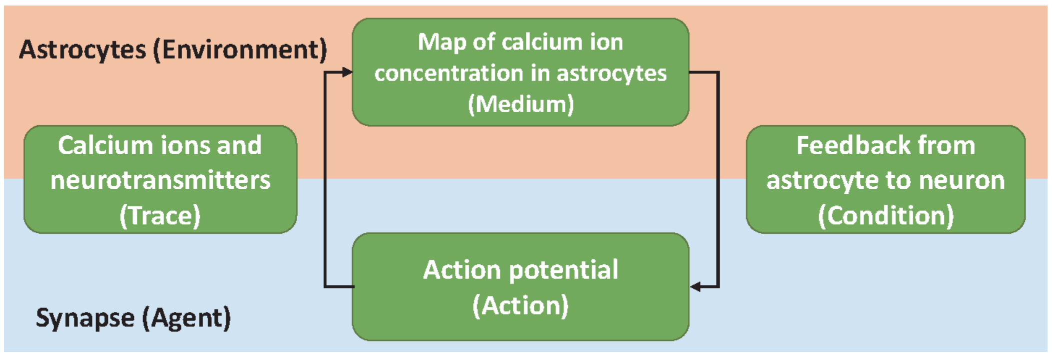
Astrocytes (Environment)
Map of calcium ion concentration in astrocytes (Medium)
Calcium ions and neurotransmitters (Trace)
Feedback from astrocyte to neuron (Condition)
Action potential (Action)
Synapse (Agent)

Figure 9.
Stigmergic interactions between astrocytes and neurons (modified from [47]).

changes are monitored as traces for indirect communication by astrocytes. Astrocytes themselves behaving like an environment with calcium ion concentration gradients within the cell acting as a medium for other neuron agents to indirectly infer these changes. This interaction creates a feedback mechanism in an asynchronous and distributed manner [47]. **Figure 9** shows the emergent stigmergy pattern in the brain. Short term activity and long-term activity gets communicated over a distance to other synapses over a spatial domain. Greater the distance, lower would be the influence. The details about the stigmergy based brain plasticity is presented in [47], interested readers are encouraged to explore further. This is a relatively new discovery and extensive research is underway to understand the role of astrocytes in overall brain mechanics.

5. SNN simulation tools and hardware accelerators

There are several spiking neural network simulation tools available which support biologically realistic neuron models for large scale networks. Some of the popular ones are:

Brian [48], is a free, open source simulator for spiking neural networks. This simulator is capable of running on several different platforms and is implemented in python making it extendable and easy to use.

NEST [49] is another simulator focusing on the dynamics, size and structure of neural systems both large and small. This tool is not intended for modeling the intricate biological details of a neuron.

NEURON [50] is simulation environment best suited for modeling individual neurons and their networks. This is popular among neuroscientists for its ability to handle complex models in a computationally efficient manner. Unlike above simulator, NEURON can handle morphological details of a neuron and is used to validate theoretical models with experimental data.

The above tools are commonly used in modeling biologically realistic neuron modes. They have their own unique interfaces and low-level semantics. An effort is made to smooth things out with a tool independent API package developed on Python programming language called PyNN [51]. The PyNN framework provides API support to model SNNs at a high level of abstraction of all aspects of neuron modeling and SNN representation, including populations of neurons, connections, layers etc. Though this provides high level abstraction, it also provides the ability to program at a low level such as adjusting individual parameters at the neuron and synapse level. To make things easy PyNN provides a set of library implementation for neurons, synapses, STDP models etc. They also provide easy interfaces to model various connectivity patterns among neurons like; all-to-all, small-world, random distance-dependent etc. These APIs are simulator independent making the code portable across different supported simulation tools and neuromorphic hardware platforms. It is relatively straightforward to add support to any custom simulation tool. PyNN officially supports BRIAN, NEST and NEURON SNN simulation tools. It is also supported on SpiNNaker [52] and BrainScaleS-2 [53] neuromorphic hardware systems. There are several more simulation tools which work with PyNN.

Cypress [54] is a C++ based SNN Simulation tool. This provides a C++ wrapper around PyNN APIs. Hence, extending the multi-platform reach of Cypress using C++ interface. It is also capable of executing networks remotely on neuromorphic compute platforms.

The BrainScaleS-2 [53] is a mixed-signal accelerated neuromorphic system with analog neural core, digital connectivity along with embedded SIMD microprocessor. It is efficient for emulations of neurons, synapses, plasticity models etc. This hardware based system is capable of evaluating models up to ten thousand times faster than real time.

The SpiNNaker [52] is another neuromorphic system custom built with digital multicore ARM processors. The SpiNNaker system (NM-MC-1) consists of custom chips each with eighteen cores sharing a local 128 MB RAM. The overall system scales to more than a million cores.

Apart from the above tools and platforms the are many custom SNN tools available to model SNNs easily for machine learning purposes. ANNarchy (Artificial Neural Networks architect) [55] is a custom simulator for evaluating SNNs. This is implemented in C++ language, along with acceleration support provided using OpenMP/CUDA. The network definitions are provided using python interface.

NeuCube [6] is a development environment for creation of Brain-Like Artificial Intelligence. The computational architecture is suited for modeling SNN applications across several domain areas. This tool supports the latest neural network models for AI purpose. It supports PyNN interface, hence extending its versatility. This tool can run on CPU, GPU and SpiNNaker platforms, also a cloud version of the tool is available.

TrueNorth [56] is another neuromorphic platform capable of evaluating SNNs at faster than real time and at very low power. They demonstrate running state of the art neural networks on the hardware platform scaling up to 64 million neurons and 16 billion synapses while the system consumes only 70 W of power out of which only 15 W is consumed by the neuromorphic hardware components. The hardware supports inference only, with learning performed off chip.

Loihi [57] is the latest offering in the neuromorphic SNN hardware. This hardware approach gets rid of crossbar architecture, which is prevalent in most previous neuromorphic implementations, lending itself to greater amount of flexibility. Loihi is also capable of on-chip learning which is a huge advantage in terms of online learning of synapses.

Other simulators capable of modeling software based models and models for custom neuromorphic hardware are presented in [20, 58–60]. This is still an ongoing field of research and there are several more accelerator-based simulators available hence the reader is encouraged to explore further. Neuromorphic hardware using more exotic hardware devices like memristors and phase change memories are also an active area of research, they are yet to make it to mainstream consumption hence they are only mentioned here.

6. Case studies

In this section few case studies are presented to bolster the concepts discussed in this chapter. The topics covered here include STDP learning dynamics, probabilistic graphical models as SNNs, SNN with BP-STDP based learning and SNNs on Neuromorphic Hardware.

6.1 STDP learning dynamics

A SNN is trained [38] to classify handwritten digits from the MNIST dataset using the STDP based learning rules Exp, Q2PS and 2P presented in Section 4.2. The authors build a three-layer SNN as shown in **Figure 10**. The MNIST images are of 28x28 pixel dimensions, hence the input layer contains 784 neurons, one per image pixel. The second/hidden layer contains neurons for learning the features of the input images. The number of neurons in this layer is varied over different trials to evaluate the effectiveness of the learning rule. Finally, the third layer consists of 10

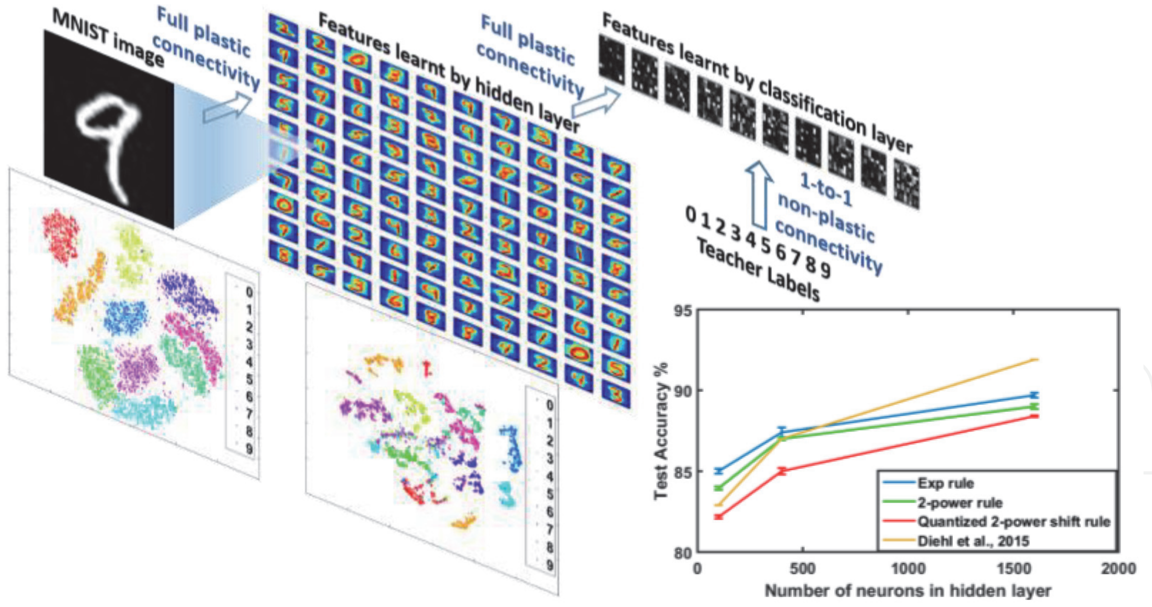
MNIST image
Full plastic connectivity
Features learnt by hidden layer
Full plastic connectivity
Features learnt by classification layer
1-to-1 non-plastic connectivity
0 1 2 3 4 5 6 7 8 9
Teacher Labels
Test Accuracy %
Number of neurons in hidden layer
Exp rule
2-power rule
Quantized 2-power shift rule
Diehl et al., 2015

Figure 10.
MNIST SNN architecture showing connectivity, input, learnt features, labels and t-SNE visualizations, along with accuracy results [38].

neurons for classifying the input with one neuron per class. The input layer encodes the pixel intensities with varying firing rate in the range of 0 Hz – 300 Hz. Each input neuron is fully connected to the hidden layer neurons similarly each hidden layer neuron is fully connected to the output/classification layer neurons. In this network all synapses are plastic with soft WTA connectivity implemented between input layer and hidden layer neurons to facilitate different neurons to pick up shared features. On the other hand, a hard WTA connectivity exists between hidden layer and the classification layer.

A qualitative analysis of the learning rule is depicted by the t-distributed stochastic neighbor embedding (t-SNE) [61] visualizations in **Figure 10**. The t-SNE algorithm maps high dimensional data points lying on different but related low-dimensional manifolds to lower dimensions by capturing local structure present in high dimensional data. The input layer firing rate visualizations show the clustering of digit classes in 2 dimensions based on raw pixel data which has 784 dimensions. Similarly, the second visualization is made using the firing rate based on the learnt features of hidden layer as input to the t-SNE algorithm with 100 dimensions. It can be clearly seen that the STDP rule produces tight clustering of input space which is projected on to the feature space. The classification layer further groups these features to its respective classes. Networks with different number of hidden layer neurons are experimented with and the results are shown in the bottom right side of **Figure 10**. The robustness of the learning method is also demonstrated with experiments yielding similar accuracies with additive Gaussian white noise along with the use of NWTA network.

6.2 Probabilistic graphical models as SNNs

An inference network based on a probabilistic graphical model for sentence construction is created using Bayesian neurons. It consists of lexicons representing words and phrases. Here each lexicon is a WTA sub network.

The network consists of two functional sections: word sub network and phrase sub network. Each symbol neuron in word sub network represents a possible word occurrence and each symbol neuron in phrase sub network represents a possible pair of words co-occurring. The synapses between the symbol neurons represent the log conditional probabilities of words and phrases co-occurring. This network is initialized to have same intrinsic potential across all symbol neurons resulting in same initial firing rate. Based on the synaptic weights the strongly connected neurons resonate and enhance each other while laterally inhibiting other symbol neurons within the lexicon WTA network. These winning neurons proportionally excite other symbol neurons across different lexicons. In this manner the network settles on a steady state firing rate which represents a contextually correct behavior. From each lexicon of the word sub network a symbol neuron is picked with highest firing rate representing a grammatically correct semantically meaningful sentence. The WTA connections in this network perform soft WTA action there by the facilitating the retention of contextual information. **Figure 11** (a) shows the network topology. For the experiments, random documents images are picked, and fuzzy character recognition is performed. Due to the fuzzy nature, each character position will result in several possible matches hence, multiple possible matches for each word position is possible as described in [62]. An example of lexicon set is [{we, wo, fe, fo, ne, no, ns, us} {must, musk, oust, onst, ahab, bust, chat} {now, noa, non, new, how, hew, hen, heu} {find, rind, tina} {the, fac, fro, kho} {other, ether}]. The SNN after evaluating the lexicons settles on a grammatically correct sentence as [we must now find the other] as seen in **Figure 11** (b).

6.3 SNN with Backpropagation-STDP based learning

Using the learning rule presented in Section 4.4, the authors of [45] train SNNs to evaluate BP-STDP rule on the XOR problem, the iris dataset and the MNIST dataset. They show that the network can model the linearly inseparable XOR problem using an SNN with 2 input, 20 hidden and 2 output neurons. For the iris dataset they create a SNN with 4 input, 30 hidden and 3 output neurons. With this network they were able to achieve 96% accuracy which is comparable to ANN trained with traditional backpropagation with an accuracy of 96.7%. The SNN for MNIST dataset

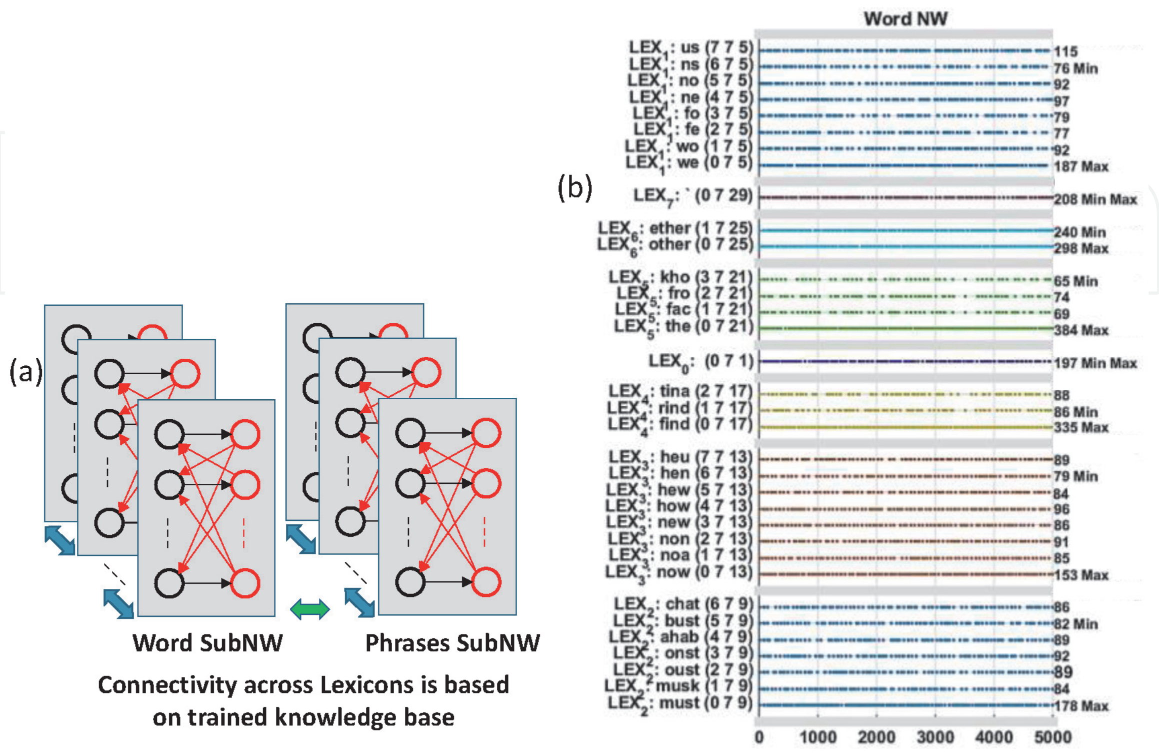
(a)
Word SubNW
Phrases SubNW
Connectivity across Lexicons is based on trained knowledge base
(b)
Word NW
0
1000
2000
3000
4000
5000

Figure 11.
(a) Sentence confabulation network, (b) confabulation results spike plot [62].

consists of 784 input neurons, 100 through 1500 hidden neurons and 10 output neurons. With this network they were able to achieve 97.2% classification accuracy.

6.4 SNNs on Neuromorphic hardware

Deep networks achieve higher accuracy in recognition tasks and in some cases outperform humans. Eedn framework is proposed in [63], which enables SNNs to be trained using backpropagation with batch normalization [64] and implement them on TrueNorth neuromorphic hardware. The Eedn trained networks are capable of achieving state-of-the-art accuracy across eight standard datasets of vision and speech. In this implementation the inference on hardware can be run at up to 2600 frames/s which is faster than real time while consuming very low power of at most 275 mW across their experiments. The network uses low precision ternary weights +1, 0 and − 1 for its synapses. A binary activation function with an approximate derivative is modeled to enable backpropagation. A hysteresis parameter is introduced in the weight update rule to avoid rapid oscillations of weights during learning. The input images are transduced by applying 12 different convolutional filter operators with binary outputs to get 12 channel input to the network as shown in **Figure 12**.

Experiments were performed on eight datasets using five different network sizes spanning across several TrueNorth chips. The results of the experiments are summarized in **Figure 13**.

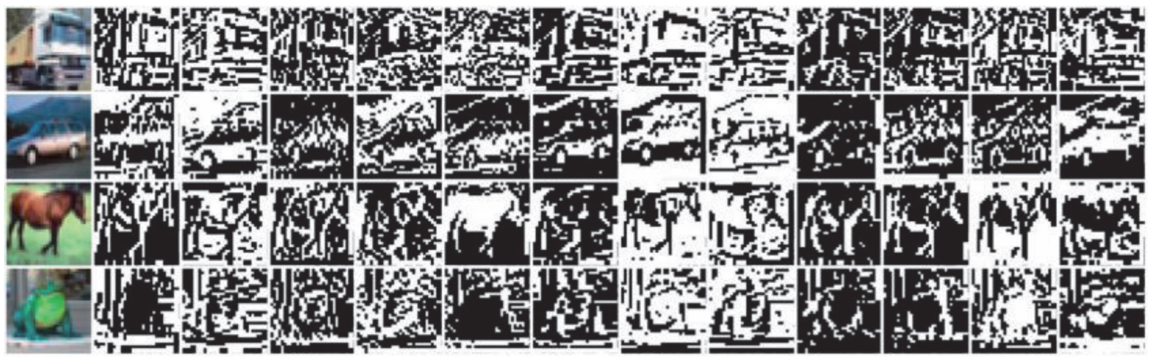

Figure 12.
Example image from CIFAR10 (column 1) and the corresponding output of 12 typical transduction filters (columns 2–13) [63].

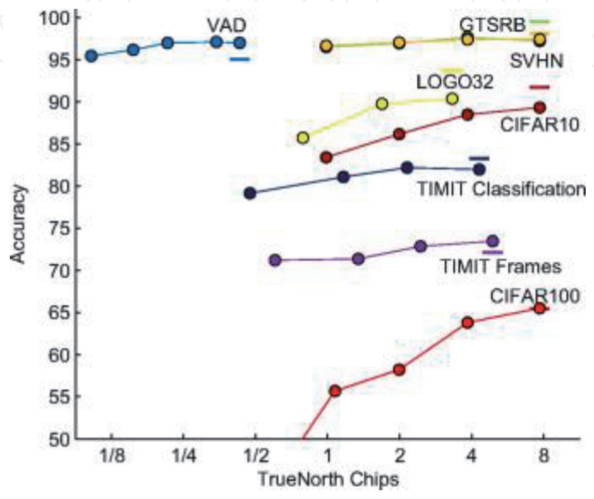

Figure 13.
Accuracy of different sized networks on eight datasets. For comparison, accuracy of state-of-the-art unconstrained approaches are shown as bold horizontal lines [63].

7. Conclusion

This chapter discussed several concepts and techniques, all of which are bio inspired. The case studies presented provide a strong basis to grasp the immense potential these algorithms provide in tackling the very complex problems of today, which were unimaginable without the advances in this field. This chapter specifically provided a beginner's guide to the field of spiking neural networks. It presented a brief overview of neuron biology and notes on popular artificial neuron models. Information representation as spikes and how to transduce real world data to spikes and vice-versa was discussed which is similar to how brain represents information. Several tools for spiking neural network modeling and evaluation were provided for wholistic understanding and for experimental evaluation of one's network models. A few case study examples are presented to understand the presented concepts and the scope of information presented in this chapter. This is an ongoing research and a very hot topic with substantially new concepts and discoveries being published every week. The motivation being the ability for machines to autonomously and efficiently perform tasks which were previously delegated to humans only along every aspect of our lives. This is a paradigm shift and research will continue to not only develop machine intelligence but also to understand the inner workings of our brains, our thoughts and advance the field of neuroscience.

Acknowledgements

This chapter represents fundamental knowledge for understanding spiking neural networks. Some of the text and images are adopted from the available research literature. Rest of the work represents authors original contributions along with the co-authors of the following research contributions [20, 38, 39, 43, 60, 62]. I am thankful for the support of Dr. Qinru Qiu from Syracuse University and her research group members specifically Amar Shrestha in contributing during the original research.

List of Abbreviations

AI	artificial intelligence
ANN	artificial neural networks
BN	Bayesian neuron
BP STDP	backpropagation-STDP
Ex	excitor neuron
LL	lower limit neuron
LTD	long-term depression
LTP	long-term potentiation
MW	moving-window
NWTA	normalized winner take all
PSTH	peri-stimulus-time histogram
SF	step-forward
SNN	spiking neural network
STDP	spike timing dependent plasticity
t-SNE	t-distributed stochastic neighbor embedding
UL	upper limit neuron
WTA	winner take all

Author details

Khadeer Ahmed
Synopsys Inc, Mountain View, USA

*Address all correspondence to: khadeer.ah@gmail.com

References

[1] Guoqing Z, Tao L. Bio-inspired autonomous navigation system for logistics mobile robots with inertial AHRS. In: 2017 IEEE 3rd Information Technology and Mechatronics Engineering Conference (ITOEC). New York City, NY, USA: IEEE; 2017. pp. 971-975

[2] Chengetanai G, O'Reilly GB. Review of swarm intelligence routing algorithms in wireless mobile ad hoc networks. In: 2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO). New York City, NY, USA: IEEE; 2015. pp. 1-7

[3] Bermejo-Busto J, Martin-Gomez C, Zuazua-Ros A, Ibanez-Puy M, Miranda-Ferreiro R, Baquero-Martin E. Improvement of a Peltier HVAC System Integrated into Building Envelopes Implementing Beehive Strategies: A Theory-Based Approach. Federacion Asociaciones Ingenieros Industriales Espana Alameda De Mazarredo. Bilbao Spain: DYNA Publishing; 2016

[4] Maass W. Networks of spiking neurons: The third generation of neural network models. Neural Networks. 1997;**10**(9):1659-1671

[5] Markram H, Gerstner W, Sjöström PJ. Spike-timing-dependent plasticity: A comprehensive overview. Frontiers in Synaptic Neuroscience. 2012;**4**:2. DOI: 10.3389/fnsyn.2012.00002. ISSN: 1663-3563. Available from: https://www.frontiersin.org/article/10.3389/fnsyn.2012.00002

[6] Kasabov NK. NeuCube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data. Neural Networks. 2014;**52**:62-76

[7] iniLabs [Online]. Available: https://inilabs.com

[8] McCulloch WS, Pitts W. A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biology. 1990;**52**(1–2):99-115

[9] Hodgkin AL, Huxley AF. A quantitative description of membrane current and its application to conduction and excitation in nerve. The Journal of Physiology. 1952;**117**(4):500-544

[10] Rosenblatt F. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review. 1958;**65**(6):386

[11] Izhikevich EM. Simple model of spiking neurons. IEEE Transactions on Neural Networks. 2003;**14**(6):1569-1572

[12] Abbott LF. Lapicque's introduction of the integrate-and-fire model neuron (1907). Brain Research Bulletin. 1999;**50**(5–6):303-304

[13] Stein RB. A theoretical analysis of neuronal variability. Biophysical Journal. 1965;**5**(2):173-194

[14] Ermentrout GB, Kopell N. Parabolic bursting in an excitable system coupled with a slow oscillation. SIAM Journal on Applied Mathematics. 1986;**46**(2):233-253

[15] Fourcaud-Trocmé N, Hansel D, Van Vreeswijk C, Brunel N. How spike generation mechanisms determine the neuronal response to fluctuating inputs. The Journal of Neuroscience. 2003;**23**(37):11628-11640

[16] Jolivet R, Lewis TJ, Gerstner W. Generalized integrate-and-fire models of neuronal activity approximate spike trains of a detailed model to a high degree of accuracy. Journal of Neurophysiology. 2004;**92**(2):959-976

[17] Stevens CF, Zador AM. Novel Integrate-and-re-Like Model of Repetitive Firing in Cortical Neurons.

Rockville, MD, USA: American Physiological Society; 1998

[18] Smith GD, Cox CL, Sherman SM, Rinzel J. Fourier analysis of sinusoidally driven thalamocortical relay neurons and a minimal integrate-and-fire-or-burst model. Journal of Neurophysiology. 2000;**83**(1):588-610

[19] Izhikevich EM. Resonate-and-fire neurons. Neural Networks. 2001;**14** (6–7):883-894

[20] Ahmed K, Shrestha A, Qiu Q. Simulation of bayesian learning and inference on distributed stochastic spiking neural networks. In: 2016 International Joint Conference on Neural Networks (IJCNN). New York City, NY, USA: IEEE; 2016. pp. 1044-1051

[21] Meunier C, Segev I. Playing the Devil's advocate: Is the Hodgkin–Huxley model useful? Trends in Neurosciences. 2002;**25**(11):558-563

[22] Cassidy AS et al. Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores. In: The 2013 International Joint Conference on Neural Networks (IJCNN). New York City, NY, USA: IEEE; 2013. pp. 1-10

[23] Gerstner W, Kistler WM, Naud R, Paninski L. Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition. Cambridge, England, UK: Cambridge University Press; 2014

[24] Adrian ED, Zotterman Y. The impulses produced by sensory nerve-endings: Part II. The response of a single end-organ. The Journal of Physiology. 1926;**61**(2):151-171

[25] Forrest MD. The sodium-potassium pump is an information processing element in brain computation. Frontiers in Physiology. 2014;**5**:472. DOI: 10.3389/fphys.2014.00472

[26] Forrest MD. Intracellular calcium dynamics permit a Purkinje neuron model to perform toggle and gain computations upon its inputs. Frontiers in Computational Neuroscience. 2014;**8**:86. DOI: 10.3389/fncom.2014.00086

[27] Lestienne R. Determination of the precision of spike timing in the visual cortex of anaesthetised cats. Biological Cybernetics. 1996;**74**(1):55-61

[28] Mainen ZF, Sejnowski TJ. Reliability of spike timing in neocortical neurons. Science. 1995;**268**(5216):1503-1506

[29] Ponulak F, Kasinski A. Introduction to spiking neural networks: Information processing, learning and applications. Acta Neurobiologiae Experimentalis (Wars). 2011;**71**(4):409-433

[30] Stein RB, Gossen ER, Jones KE. Neuronal variability: Noise or part of the signal? Nature Reviews. Neuroscience. 2005;**6**(5):389-397

[31] Zohar O, Shamir M. A readout mechanism for latency codes. Frontiers in Computational Neuroscience. 2016; **10**:107

[32] Kim J, Kim H, Huh S, Lee J, Choi K. Deep neural networks with weighted spikes. Neurocomputing. 2018;**311**: 373-386

[33] Zeldenrust F, Wadman WJ, Englitz B. Neural coding with bursts—Current state and future perspectives. Frontiers in Computational Neuroscience. 2018;**12**:48. DOI: 10.3389/fncom.2018.00048

[34] Thorpe S, Gautrais J. Rank order coding. In: Computational neuroscience. Boston, MA, USA: Springer; 1998. pp. 113-118

[35] Cattani A, Einevoll G, Panzeri S. Phase-of-Firing Code. Ithaca, NY, USA: arXiv.org, Cornell University; 2015

[36] Montemurro MA, Rasch MJ, Murayama Y, Logothetis NK, Panzeri S. Phase-of-firing coding of natural visual stimuli in primary visual cortex. Current Biology. 2008;**18**(5):375-380

[37] Danielson NB, Zaremba JD, Kaifosh P, Bowler J, Ladow M, Losonczy A. Sublayer-specific coding dynamics during spatial navigation and learning in hippocampal area CA1. Neuron. 2016;**91**(3):652-665

[38] Shrestha A, Ahmed K, Wang Y, Qiu Q. Stable spike-timing dependent plasticity rule for multilayer unsupervised and supervised learning. In: 2017 International Joint Conference on Neural Networks (IJCNN). New York City, NY, USA: IEEE; 2017. pp. 1999-2006

[39] Shrestha A et al. A spike-based long short-term memory on a neurosynaptic processor. In: 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). New York City, NY, USA: IEEE; 2017. pp. 631-637

[40] Schrauwen B, Campenhout J. BSA, a fast and accurate spike train encoding scheme. In: Proceedings of the International Joint Conference on Neural Networks. Vol. 4. New York City, NY, USA: IEEE; 2003. pp. 2825-2830. DOI: 10.1109/IJCNN.2003.1224019

[41] Kasabov N et al. Design methodology and selected applications of evolving spatio-temporal data machines in the NeuCube neuromorphic framework. Neural Networks. 2016;**78**:1-14

[42] Hebb DO. The Organization of Behavior: A neuropsychological Theory. Abingdon, England, UK: Taylor & Francis; 1949

[43] Ahmed K, Shrestha A, Qiu Q, Wu Q. Probabilistic inference using stochastic spiking neural networks on a neurosynaptic processor. In: 2016 International Joint Conference on Neural Networks (IJCNN). New York City, NY, USA: IEEE; 2016. pp. 4286-4293

[44] Masquelier T, Guyonneau R, Thorpe SJ. Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains. PLoS One. 2008;**3**(1)

[45] Tavanaei A, Maida A. BP-STDP: Approximating backpropagation using spike timing dependent plasticity. Neurocomputing. 2019;**330**:39-47

[46] Haydon PG, Carmignoto G. Astrocyte control of synaptic transmission and neurovascular coupling. Physiological Reviews. 2006;**86**(3):1009-1031

[47] Xu X, Zhao Z, Li R, Zhang H. Brain-inspired Stigmergy learning. IEEE Access. 2019;**7**:54410-54424

[48] Stimberg M, Brette R, Goodman DFM. Brian 2, an intuitive and efficient neural simulator. eLife. 2019;**8**:e47314. DOI: 10.7554/eLife.47314

[49] Gewaltig M-O, Diesmann M. Nest (neural simulation tool). Scholarpedia. 2007;**2**(4):1430

[50] Hines ML, Carnevale NT. The NEURON simulation environment. Neural Computation. 1997;**9**(6):1179-1209

[51] Davison AP et al. PyNN: A common interface for neuronal network simulators. Frontiers in Neuroinformatics. 2009;**2**:11

[52] Furber SB et al. Overview of the spinnaker system architecture. IEEE Transactions on Computers. 2012;**62**(12):2454-2467

[53] Grübl A, Billaudelle S, Cramer B, Karasenko V, Schemmel J. Verification and Design Methods for the BrainScaleS

Neuromorphic Hardware System. arXiv Prepr. arXiv2003.11455. 2020

[54] Stöckel A. Cypress: C++ Spiking Neural Network Simulator Framework [Online]. Available from: https://github.com/hbp-unibi/cypress

[55] Vitay J, Dinkelbach HÜ, Hamker FH. ANNarchy: A code generation approach to neural simulations on parallel hardware. Frontiers in Neuroinformatics. 2015;**9**:19

[56] DeBole MV et al. TrueNorth: Accelerating from zero to 64 million neurons in 10 years. Computer (Long. Beach. Calif). 2019;**52**(5):20-29

[57] Davies M et al. Loihi: A neuromorphic manycore processor with on-chip learning. IEEE Micro. 2018; **38**(1):82-99

[58] Chou T-S et al. CARLsim 4: An open source library for large scale, biologically detailed spiking neural network simulation using heterogeneous clusters. In: 2018 International Joint Conference on Neural Networks (IJCNN). New York City, NY, USA: IEEE; 2018. pp. 1-8

[59] Catania V, Mineo A, Monteleone S, Palesi M, Patti D. Noxim: An open, extensible and cycle-accurate network on chip simulator. In: 2015 IEEE 26th International Conference on Application-Specific Systems, Architectures and Processors (ASAP). New York City, NY, USA: IEEE; 2015. pp. 162-163

[60] Ahmed K, Shrestha A, Wang Y, Qiu Q. System design for in-hardware stdp learning and spiking based probablistic inference. In: 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). New York City, NY, USA: IEEE; 2016. pp. 272-277

[61] van der Maaten L, Hinton G. Visualizing data using t-SNE. Journal of Machine Learning Research. 2008;**9** (Nov):2579-2605

[62] Qiu Q, Li Z, Ahmed K, Li HH, Hu M. Neuromorphic acceleration for context aware text image recognition. In: 2014 IEEE Workshop on Signal Processing Systems (SiPS). New York City, NY, USA: IEEE; 2014. pp. 1-6

[63] Esser S, et al. Convolutional Networks for Fast, Energy-Efficient Neuromorphic Computing. arXiv. arXiv Prepr. arXiv1603.08270. 2016

[64] Ioffe S, Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv Prepr. arXiv1502.03167. 2015