Spiking Neural Computing in Memristive Neuromorphic Platforms

Mahyar Shahsavari, Philippe Devienne, Pierre Boulet

Spiking Neural Computing in Memristive Neuromorphic Platforms

Mahyar Shahsavari, Philippe Devienne and Pierre Boulet

Abstract Neuromorphic computation using Spiking Neural Networks (SNN) is proposed as an alternative solution for future of computation to conquer the memory bottelneck issue in recent computer architecture. Different spike codings have been discussed to improve data transferring and data processing in neuro-inspired computation paradigms. Choosing the appropriate neural network topology could result in better performance of computation, recognition and classification. The model of the neuron is another important factor to design and implement SNN systems. The speed of simulation and implementation, ability of integration to the other elements of the network, and suitability for scalable networks are the factors to select a neuron model. The learning algorithms are significant consideration to train the neural network for weight modification. Improving learning in neuromorphic architecture is feasible by improving the quality of artificial synapse as well as learning algorithm such as STDP. In this chapter we proposed a new synapse box that can remember and forget. Furthermore, as the most frequent used unsupervised method for network training in SNN is STDP, we analyze and review the various methods of STDP. The sequential order of pre- or postsynaptic spikes occurring across a synapse in an interval of time leads to defining different STDP methods. Based on the importance of stability as well as Hebbian competition or anti-Hebbian competition the method will be used in weight modification. We survey the most significant projects that cause making neuromorphic platform. The advantages and disadvantages of each neuromorphic platform are introduced in this chapter.

AQ1 AQ2

1 Introduction

The mammalian nervous system is a network of extreme complexity which is able to perform cognitive computation in a parallel and power-efficient manner. Understanding the principles of the brain processing for computational modeling is one of the

M. Shahsavari (✉) · P. Devienne · P. Boulet
CRIStAL, Centre de Recherche en Informatique Signal et Automatique de Lille,
Université Lille, CNRS, Centrale Lille, UMR 9189, 59000 Lille, France
e-mail: mahyar.shahsavari@gmail.com

L. Chua et al. (eds.), *Handbook of Memristor Networks*,
https://doi.org/10.1007/978-3-319-76375-0_25

biggest challenges of the 21st century that led to the new branch of research e.g., neuromorphic computing. Neuromorphic engineering represents one of the promising fields for developing new computing paradigms complementing or even replacing current Von Neumann architecture [1].

The main remarkable difference between conventional Von Neumann architecture and neuromorphic systems is in their use of memory structures. The way of communication between memory and central processing unit (CPU) in conventional computing is not efficient. The memory and CPU communication suffers from what is called Von Neumann memory bottelneck. The CPUs access both data and program in memory using the same shared resources. CPUs spend most of their time idle because the speed of CPU is much more than memory due to the quality of materials applied to manufacturing the transistors in CPU and different memories.

If we want to apply better quality of memory such as SRAM, regarding to the high demands of memory usages the machine would be more expensive. To improve the efficiency of nowadays computation platforms, the applicable solution is what commonly known as the cache hierarchy; in other words, a limited amount of fast but costly memory sit closer to the processing unit, while most of the data would be stored in the cheaper but larger memory as it is shown in Fig. 1a. To execute computational tasks, instruction codes and data stored in the memory are fetched to the processor, and after execution, pushed back to the memory unit, via a memory bus. Subsequently, it would be operating system (OS) duty to manage the data around these different levels of memory to optimize the system speed by consisting frequently-used data to the closer memory with better quality and speed rate. On the other hand, the multi-core platforms are commonly used in the new hardwares and the memory hierarchy management would be more significant and difficult too. By proposing computing unit next to the local memory, neuromorphic brain-inspired computing paradigms offer an attractive solution for implementing alternative non von Neumann architectures, using advanced and emerging technologies such as memristor [2].

Neuromorphic systems are electronic implementations inspired from neural systems that is known as neuro-inspired computation system. The idea of creating circuit model for a neuron system refers back at least to 1907, where a neuron is modeled by a resistor and a capacitor [3]. However, the first neuromorphic term was coined by Carver Mead [4] using Very Large Scale Integration (VLSI) technology to propose an implementation of neural system hardware. Mahowald and Mead implemented the first silicon retina model with considering of adaptivity and energy efficiency by simulating retina functionalities [5]. Tobi Delbruck built on the idea of adaptive photoreceptor circuits developed in [6] and presented approaches for enhancing retinomorphic sensors consist of 128×128 pixel Dynamic Vision Sensor (DVS). DVS established a benchmark in neuromorphic vision domain with introducing Address Event Representation (AER) sensory data in which each individual pixel processed the normalized time derivative of the sensed light and provided an output in the form of spikes of the pixel addresses. In addition, vision sensory neuromorphic research, there are several neuromorphic studies using auditory and olfactory sensors [7–9] for review study in neuromorphic research using different sensory inputs, we refer the readers to [10].

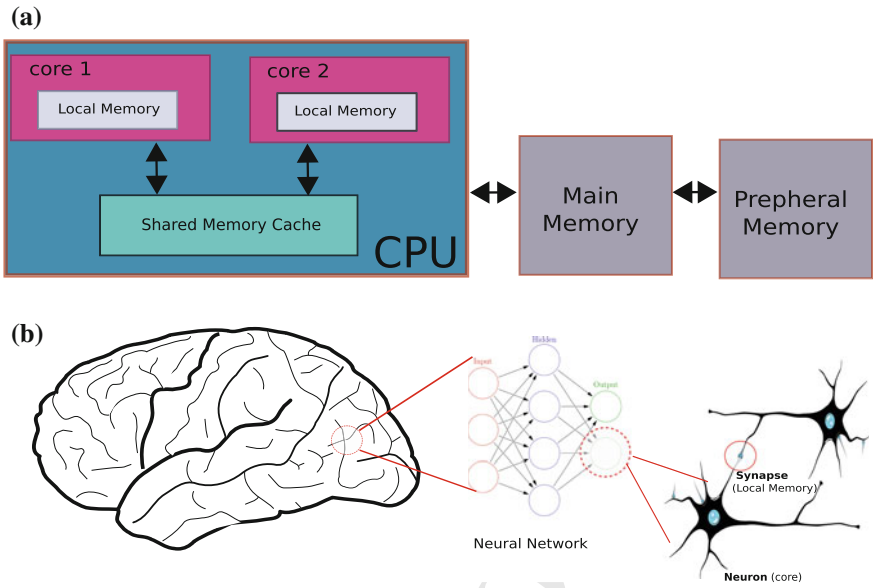

Fig. 1 Computational architecture **a** Von Neumman architecture, fast and costly memory are closer to cores in multiprocessor platforms as cashes and local memory as well as inexpensive and slower memory are in other layers close to magnetic memory to save the cost of CPU (memory hierarchy). **b** Neuromorphic architecture inspired from neural networks in the biological brain, capable to conquer Von neumann bottelneck issue, performing parallel and cognitive computing, as well as considering that the synapses are local memories connected to each neurons as computational cores

More close to our research, in 2014 two distinguished articles were published that increased the scientists attentions to the general neuromorphic platforms as novel computing architectures. Merolla et al. [11] in an IBM research was sponsored by DARPA, have demonstrated a computing hardware consist of the compact modular core for large-scale neuromorphic system architecture. The cores combine digital neurons with the large synaptic array. This general purpose neuromorphic processor was built using thousands of neurosynaptic cores are involved one million neurons and 256 million of reconfigurable synapses. The second notable work published in 2014 was Spiking Neural Network Architecture (SpiNNaker) project [12]. The SpiNNaker project is a decade old, comprehensive description of the project is announced in [12]. SpiNNaker project aims to deliver a massively parallel million core architectures whose interconnections are inspired by the connectivity properties of the mammalian brain. The hardware platform is suitable to model the large-scale spiking neural networks in biological real time. Neuromorphic and neuro-inspired computing is now being adapted by an increasing number of academic and industrial different research teams. In recent few years, there have been many valuable publications explaining the use of novel materials such as memristors are able to emulate some of the properties observed in biological synapses [2, 13–17].

Our work focuses on an alternative approach aimed at high performance computation to realize the compact, parallel, cognitive and energy-efficient architecture structure that emulate the style of computation of the biological brain, using the Spiking Neural Network (SNN) structure, modeling the neurons as computational cores next to memristive artificial synapses as local memories to skip memory delay bottelneck similar to what is shown in Fig. 1b. Therefore, it is necessary to define, analyze and verify the efficient models of network topology, neuron and synapse models based on state-of-the-art technologies besides choosing the optimized learning model adapted to our platform and devices. The structure of Chapter is followed by reviewing SNN and more significantly the functionality of various spike information codings. In the same section, we discuss different neural network topologies. Furthermore in the Sect. 3, different models of neuron is presented. Synapse and learning are explained in the Sect. 4 which various methods of spike-timing-dependent plasticity (STDP) [18, 19] are studied comprehensively. The state-of-the-art of the most important neuromorphic platforms and projects in the world is presented in Sect. 5. Lateral inhibition and Homeostasis have been discussed at the discussion part of this chapter.

2 Spiking Neural Networks

Artificial neural networks (ANN) can generally be categorized into three generations. The first generation of neural network consisted of McCulloch and Pitts neurons [20] that the output signals are limited to discrete ‘0’ or ‘1’ binary values. Perceptrons, Hopfield network, Boltzmann machine and multilayer networks with threshold units are ANN examples that are classified in first generation. The second generation of neural network, by using a continuous activation function such as sigmoid, polynomial or exponential functions, the output can take analog values between ‘0’ and ‘1’. Due to using analog output the network requires less neurons than the first generation class. Radial Basis Function (RBF) networks and Multi-Layer Perceptrons (MLP) are categorized under second generation class. The third generation of neural network model are networks which employ spiking neurons as computational units. In this model, the precise firing times of neurons are used for information coding. Spiking neural networks belong to the third generation of neural networks.

Indeed, artificial neural network in the first and second generation is a mathematical model of mammalian brain though, SNN is an electronic hardware neuromorphic model of the biological brain. Networks composed of spiking neurons are able to process significant amount of data using a relatively small number of spikes [21]. Due to the similarity between the biological neurons and spiking models functionality, SNNs provide powerful tools to emulate data processing in the brain, including neural information processing, plasticity and learning. Consequently, spiking networks offer solutions to a broad range of specific problems in applied engineering image detection, event detection, classification, speech recognition and many cognitive computation domain applications.

2.1 Spike Information Coding

Spike is the language that neurons communicate to each other in SNN architectures. One of the key unresolved questions in neuroscience is how information processed in the brain. The nature of the neural code is an unresolved topic of research in neuroscience. However, based on what is known from biology, a number of neural information encoding have been proposed:

1. **Rate coding**
   The rate of spikes in a time-window is counted for the information transmission. It is also called as frequency coding (Fig. 2a). As the intensity of a stimulus increases more, the firing rate of spikes increases more too. Rate encoding is motivated by the observation that biological neurons eager to fire more often for stronger stimuli. There are two types of rate coding namely spike-count rate and time-dependent firing rate. In spike-count rating by counting the number of spikes that are generated during a trial and dividing by the duration of the trial, we calculate the temporal average of rating. In independent firing rate, the average number of spikes over trial happens during a short interval between times t and t + Δt, divided by the duration of the interval. Brette [22] has compared these two approaches in rate information coding in more details.
2. **Latency coding**
   In this model, information is supposed to be contained in the exact timing of a set of spikes relative to each other as it is shown in Fig. 2b. It is already proved that precisely timed patterns of spikes have been postulated to play a significant role in the networks of neuron in different functions [23]. Precise spike timing is one of the important parameters that control variety forms of synaptic plasticity. Latency coding by using sequences of spikes are mainly observed in feed-forward networks since noise and dynamics of recurrent networks can disrupt spike timing precision, some attempts to harvest precise spiking timing in recurrent networks have been done for example by exploring the idea of reservoir computation [24].
3. **Phase coding**
   This model generates the times of emitted spikes based on the time point in a periodic signal. In this (Fig. 2c) method the spike trains can encode information in the phase of a pulse respecting to the background oscillations. Phase coding method has been used both in models and experimentally. Phase coding has been suggested for the hippocampus as well [25]. Spiking networks exploring the phase coding strategy have recently been applied in tasks as olfactory systems or robot navigation [26].
4. **Rank-coding (spike-order coding)**
   In this method of spike coding, information is encoded by the order of spikes in the activity of a group of neurons as it is depicted in Fig. 2d. Rank-coding approach has been suggested to describe ultra-fast categorization observed in the visual system. This model assumes that each neuron emits only a single spike during a presentation of the image. This method can be implemented in a feed-forward network with inhibitory feedback connections. Thorpe and others [27]

developed a spiking neural model that was able to classify static images with a processing speed comparable to that observed in humans one.

5. **Population coding**
   This coding model is a method to introduce stimuli by applying the joint activities of the group of neurons. In population coding, each neuron has a distribution of responses to the certain set of inputs, and the responses of group of neurons will be combined to present a value for the inputs (Fig. 2e). During the last two decades, the theory has focused on analyzing the methods in which different parameters that characterize neuronal responses to external stimuli affect the information content of these responses. Recent challenge in population coding is to develop a theory that can generate predictions for specific readout mechanisms for example for visual target information [28].
6. **Sparse coding**
   This model of coding generally refers to a representation where a few number of neurons are active, with the majority of the neurons inactive or showing low activity see Fig. 2f. Sparse coding has been suggested as a guiding principle in neural representations of sensory input, specially in the visual sensory system. It is also discussed that sparse coding offers a useful solution to the problem of representing natural data because such a scheme allows the system to take advantage of the sparse structure of the sensory environment. It is believed that the natural environment is inherently sparse and codes that using this structure can be both metabolically efficient and useful for learning. Sparseness can be defined over a population of neurons at a specific point in time (population sparseness) or it can be measured for a single neuron over a certain time-window [29].

2.2 Network Topology

The interconnection structure of neurons in a network of neurons is called topology, architecture or graph of an artificial neural network. The manner in which the interconnection is structured intimately is linked to the learning algorithms applied to train the neural networks. Indeed, the interconnection can be structured in numerous ways results in numerous possible topologies that are divided into two basic classes namely: Feed-Forward Neural Networks (FFNN) and Recurrent (or feedback) Neural Networks (RNN) depicted in Fig. 3.

2.2.1 Feed-Forward Neural Networks (FFNN)

The FFNN is divided into two different structure called single-layer FFNN and multilayer FFNN. The single-layer is structured as an input and output layer which is strictly a feed-forward or acyclic graph. We do not count the input layer because no calculation is performed in input nodes (neurons). The multilayer FFNN has one

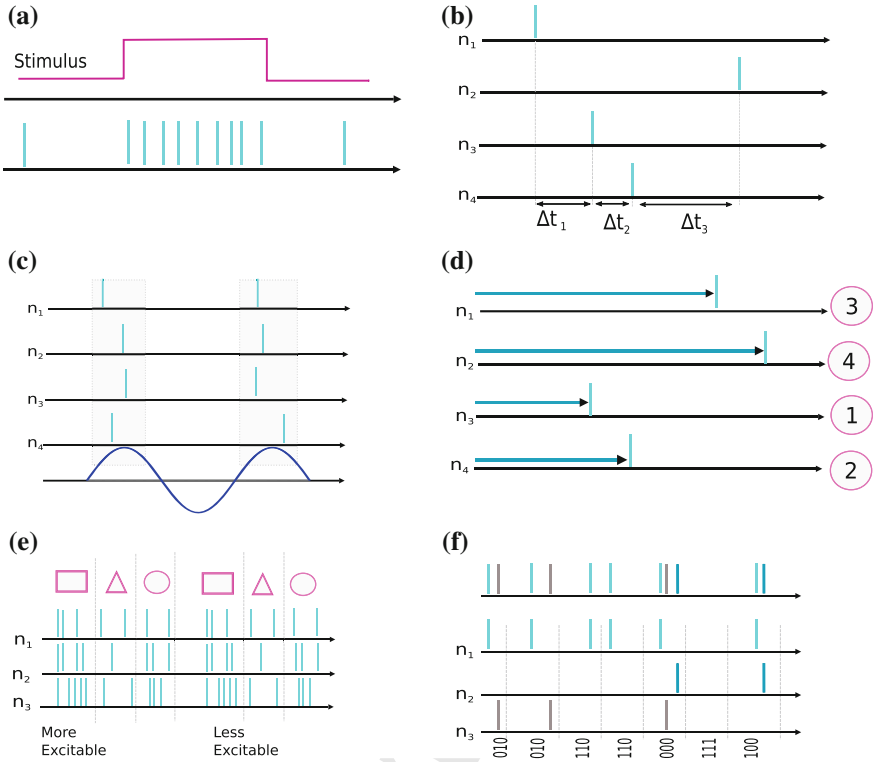

Fig. 2 Spike information coding strategies **a** Rate coding. **b** Latency coding. **c** Phase coding. **d** Rank-coding (spike-order coding). **e** Population coding. **f** Sparse coding

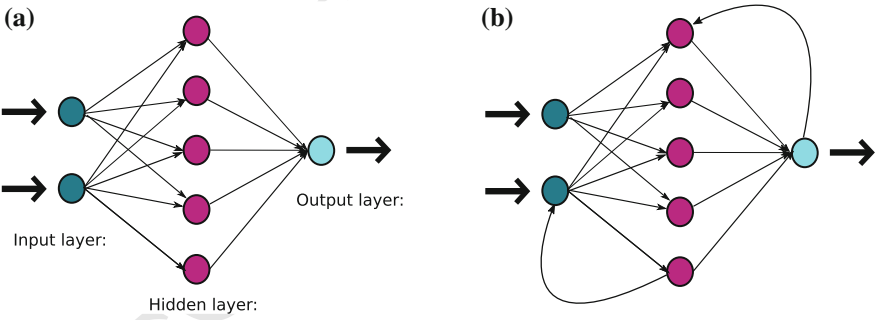

Fig. 3 Two main topologies of artificial neural network architectures **a** Feed-Forward Neural Networks (FFNN), **b** Recurrent Neural Networks (RNN)

or more hidden layers between input and output layers similar to Fig. 3a which has one hidden layer. By adding one or more hidden layers, the neural network can extract the higher-order statistics which is particularly valuable when the size of the input layer is large [30]. Among the known types of neural networks (NN), the feed-forward neural networks are the mostly used because of their simplicity, flexible structure, good qualities of representation, and their capability of universal approximation. Respecting to the way of interconnectivity of the nodes (neurons) there are two kinds of feed-forward architecture:

- **fully connected**
  In this configuration, every node in each layer of the network is connected to every other node in the next layer. In fact, we can call them globally connected networks. The Restricted Boltzmann Machine (RBM) could be an example of fully connected FFNN.
- **partially connected**
  In this configuration, some communication links are missing. The convolutional neural networks is a good example for the partially connected FFNN. Partially connected topologies present a suitable alternative with a reduced degree of redundancy and thus a potential for increased efficiency of neural networks.

2.2.2 Recurrent Neural Networks (RNN)

The RNN is distinguished from FFNN in that it has at least one *feedback* loop connection. Recurrent neural networks can be single-layer or multilayer as well. Unlike feed-forward neural networks, recurrent networks retain a state that can represent information from an arbitrarily long context window. Although recurrent neural networks have traditionally been difficult to train, and often contain thousands of parameters, recent studies in network architectures, optimization techniques, and parallel computation have enabled successful large-scale learning to use RNN [31]. *Hopfield* [32] network is an example of the recurrent artificial neural network that is used to store one or more stable vectors. The stable vectors can be considered as memories that the network recalls them when provided with similar vectors that operate as a queue to the network memory. Other example of RNN is *Elman* network [33] that refers as a simple Recurrent Network is the special case of recurrent artificial neural networks. This type of artificial neural network has the memory that allows it to both detect and generate time-varying patterns.

2.2.3 Modern Neural Networks

Here, we discuss recent feed-forward promising neural network which has been applied in different sensory computation applications.

- **Convolutional Neural Networks (CNN)** Convolutional network is a multi-layer feed-forward network architecture in which neurons in one layer receive inputs

from multiple neurons in the previous layer and produce an output which is a threshold or sigmoidal function of the weighted sum of its inputs. The connectivity pattern between the nodes of one layer and the node of the subsequent layer, responsible for the weighted sum operation forms the convolution kernel. Each layer mainly has one or few number of convolution kernels that link the activity of a set of neurons from one layer to the target neuron of the next layer [34]. Convolutional neural networks which have been explored intensively within the neuromorphic community for visual processing tasks [35]. They are normally implemented on CPUs and GPUs which consume a significant amount of power. In recent years, System-On-Chip (SOC) solutions and FPGA platforms have been used to implement these networks for increasing their performance while decreasing their power consumption.

- **Deep Belief Networks (DBN)** Deep learning is currently an extremely active research area in machine learning and cognitive computing society. It has obtained many successes in a wide area of applications such as speech recognition, computer vision, and natural language processing. Deep Belief Networks (DBNs), introduced by Hinton and his colleagues as a special type of deep neural networks with generative model properties [36]. This network is structured as interconnected pairs of Restricted Boltzmann Machines. An adaptation of the neural model to allow transfer of parameters to a 784-500-500-10 layer spiking DBN was described in [15] with good performance on the MNIST digit database. DBN architecture has been implemented on a Xilinx Spartan-6 LX150 FPGA [37] with very promising classification performance results (92%) on the same MNIST database. This FPGA implementation of the DBN (also called Minitaur) contains 32 parallel cores and 128 MB of DDR2 as main memory.

3 Spiking Neuron Model

The neuron is a dynamic element and processing unit that emits output pulses whenever the excitation exceeds some threshold. The resulting sequence of pulses or "spikes" contains all the information that is transmitted from one neuron to the other one. In this section, we compare the biological, artificial and spiking neuron and furthermore, we explain various model of spiking neuron models.

3.1 Biological, Artificial and Spiking Neuron

A biological neuron is an electrically excitable cell that processes and transmits information by electrochemical signals. Chemical signaling occurs via synapses, specialized connections with other cells. A typical physiological neuron can be divided into three anatomical and functional parts, called *dendrites, soma* and *axon* as it is shown in Fig. 4a. The soma is the central part of the neuron. It contains the nucleus of

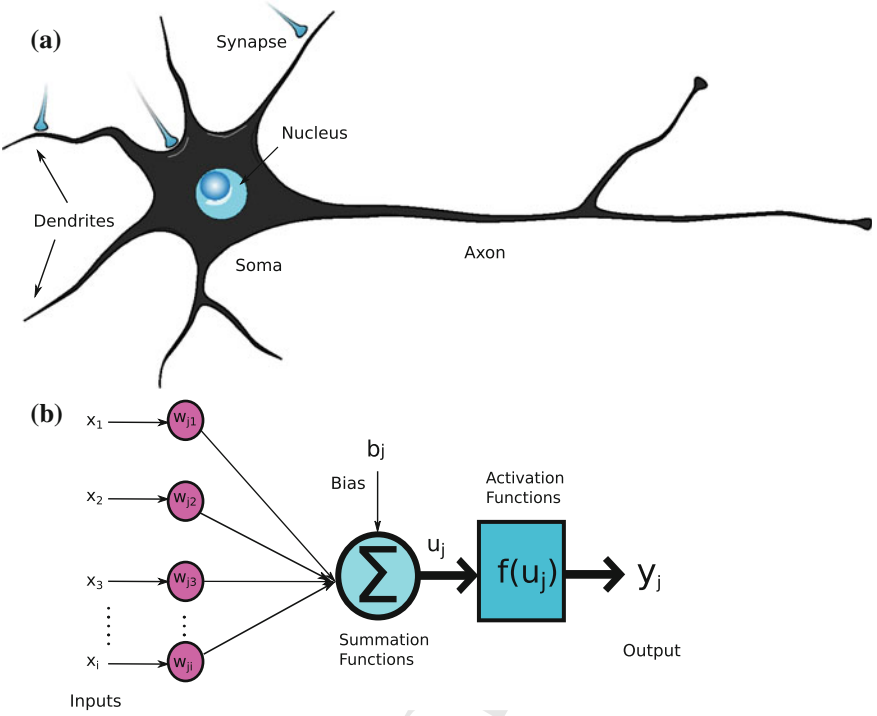

Fig. 4 The structure of a neuron **a** Physiological neuron. **b** Artificial neuron model

the cell, where most protein synthesis occurs. The soma is considered as a central processing unit that performs an important nonlinear processing. The dendrites of a neuron are cellular extensions with many branches. Dendrites typically are considered as inputs of the neuron. The axon carries nerve signals away from the soma and typically is considered as neuron output. Neurons have only one axon, but this axon may and will usually undergo extensive branching, enabling communication with many target cells. Another term which is necessary to know in the physiological neuron is *action potential* which is a short-lasting event in which the electrical membrane potential of a cell rapidly rises and falls. It plays a central role in cell-to-cell communication. Action potentials are also called "nerve impulses" or *spikes*, and the temporal sequence of them generated by a neuron is called *spike train*. A neuron that emits an action potential is said to fire.

The artificial model of the neuron is a mathematical model of the physiological neuron. The basic computational element (neuron) is often called a node, unit or *perceptron*. Each input has an associated weight w, which can be modified and react like a biological synapse. The unit computes the f function of the weighted sum of its inputs x_i:

$$u_j = \sum_{1}^{i} w_{ji} x_i \tag{1}$$

$$y_j = f(u_j + b_j) \tag{2}$$

It is obvious in Fig. 4b that x_1, x_2, x_3, …, x_i are neuron inputs, w_{ji} is the synaptic weights between neuron j and neuron i, b_j is bias, f is known as *activation function* or *transfer function* and y_j is output of the neuron. Based on the model and application of neural networks, there are several types of activation functions such as threshold or step function, linear function, and Non-linear (Sigmoid) function. Here to be able to Understand how neural network works we explain the functionality of neuron using threshold function. Respecting to the input connections in Fig. 4b, we can define a threshold for transfer function f by defining threshold θ. Here, we choose $\theta = 0$ in the way we could perform a binary classification.

$$y_j = \begin{cases} 1 \text{ if } u_j \geq 0 \\ 0 \text{ if } u_j < 0 \end{cases} \tag{3}$$

where u_j is the induced local field of the neuron; which is,

$$u_j = \sum_{1}^{i} w_{ji} x_i + b_j \tag{4}$$

Such a model of neuron is referred to McCulloch and Pitts [20].

3.2 *Spiking Neuron*

The Spiking neural model is an electrical model of physiological neuron that can be implemented on the circuit using traditional devices or state-of-the-art technologies e.g., CMOS transistors or on hardware platforms e.g., FPGAs. In Spiking model the neurons communicate using spikes and the input spikes make an action potential firing if inside a neuron reaches to the desired threshold (can be compared to threshold activation function in the artificial model of the neuron). Different models of the spiking neuron are proposed that here we study the main models.

3.2.1 Hodgkin-Huxley Model

The first electrical model and in other words the first spiking model of neuron is Hodgkin-Huxley neuron model [38] which got the Nobel Prize in Physiology or Medicine. Hodgkin and Huxley performed experiments on the giant axon of the squid and found three different types of current: sodium, potassium and leak current. It was

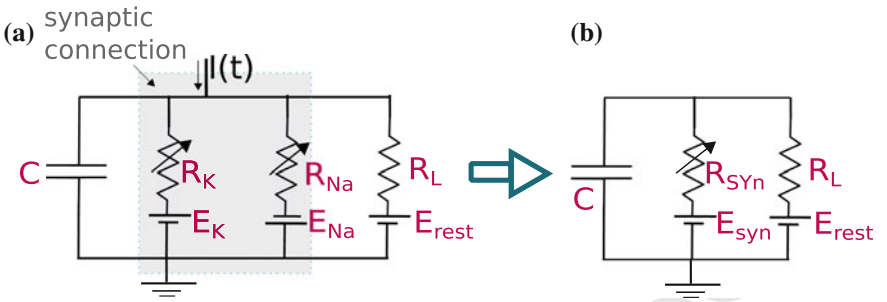

Fig. 5 Electrical circuit represents Hodgkin-Haxley model of the neuron. **a** Details circuit model of the neuron with sodium and potassium channels effects and leakage current. **b** Equivalent circuit for more simplicity in solving equations

demonstrated that the ionic permeability of the membrane can be highly dependent on the membrane potential. The schematic diagram of the Hodgkin-Huxley model is shown in Fig. 5 where E_{rest} is the membrane potential, C is the membrane capacitance, the leakage channel is described by an independent R and the conductance of this leakage is calculated $g_L = \frac{1}{R}$ the conductance the other ion channels ($g_{Na} = \frac{1}{R_{Na}}$ and $g_K = \frac{1}{R_K}$) is voltage and time dependent. The ionic current is divided into components carried by sodium and potassium ions. Each element of the ionic current is determined by a driving force which may easily be measured as an electrical potential, E_{rest} as resting membrane potential, E_{Na} and E_K sodium and potassium potentials respectively. Current can be carried through the membrane either by charging the membrane capacitance or by moving ions through the resistances in parallel with the capacitance.

The equivalent circuit of Hodgkin-Hulxey model is shown in the left side of Fig. 5 that by representing the Krichhoffs law and using this circuit we can write following equations:

$$I_L(t) = \frac{V_C(t) - E_{rest}}{R_L} \tag{5}$$

$$I_{syn}(t) = C\frac{dV_C(t)}{dt} + \frac{V_C(t) - E_{rest}}{R_L} \tag{6}$$

Solving the Eq. 6 leads to an exponential answer (Eq. 7) that can model the behavior of membrane potential.

$$V_C(t) = v_\infty(1 - \exp(-\frac{t}{\tau}) + E_{rest} \tag{7}$$

Respecting to the synaptic current charging if there is enough input current to membrane the neuron will fire. We note that $\tau = RC$ in Eq. 7 is the time constant for charging and discharging the membrane.

3.2.2 Integrate-and-Fire (I&F) Neurons

Integrate-and-Fire (I&F) neuron model are derived from the Hodgkin-Huxley neuron model. There is an important type of I&F neuron model which is named *Leaky-Integrate-and-Fire* (*LIF*). There are other types of I&F models such as Quadratic-Integrate-and-Fire (QIF). The Leaky-Integrate-and-Fire (LIF) neuron model is a well-studied model of the neuron. There are three reasons for using LIF in our platform.

- The fabricated model with recent CMOS technology is available [39, 40].
- LIF works effectively in spiking and event-based networks [41].
- LIF models are quite fast to simulate, and particularly attractive for large-scale network simulations [42].

Neurons integrate the spike inputs from other neurons they are connected to. These input spikes change the internal potential of the neuron, it is known as neuron's membrane potential or state variable. When this membrane potential passes a threshold voltage due to integrated inputs, the action potential occurs, in other words, the neuron fires. The model is described by the neuron membrane potential:

$$\tau_n \frac{\mathrm{d}v}{\mathrm{d}t} = -v(t) + RI_{syn}(t) \tag{8}$$

$$I_{syn}(t) = \sum_j g_{ij} \sum_n \alpha(t - t_j^{(n)}) \tag{9}$$

where, $v(t)$ represents the membrane potential at time t, $\tau_n = RC$ is the membrane time constant and R is the membrane resistance. Equation 8 describes a simple parallel resistor-capacitor (RC) circuit where the leakage term is due to the resistor and the integration of $I_{syn}(t)$ is due to the capacitor. The total input current, $I_{syn}(t)$, is generated by the activity of pre-synaptic neurons. In fact, each pre-synaptic spike generates a post-synaptic current pulse. The total input current injected to a neuron is the sum over all current pulses which is calculated in Eq. 9. Time $t_j^{(n)}$ represents the time of the n_{th} spike of post-synaptic neuron j, and g_{ij} is the conductance of synaptic efficacy between neuron i and neuron j. Function $\alpha(t) = q\delta(t)$, where q is the injected charge to the artificial synapse and $\delta(t)$ is the Dirac pulse function. If $I_{syn}(t)$ is big enough where action potential can pass the threshold voltage, neuron fires. It means there are enough input spikes in a short time window. When there is no or only a few spikes in a time window, the neuron is in the leaky phase and the state variable decreases exponentially. The duration of this time window depends on $\tau_n = RC$. The equation is analytically solvable and thus we use the answer of Eq. 8 in the network simulation when there is an input spike to improve the simulation performance. In Fig. 6, you can see the Matlab model of a single neuron. When the

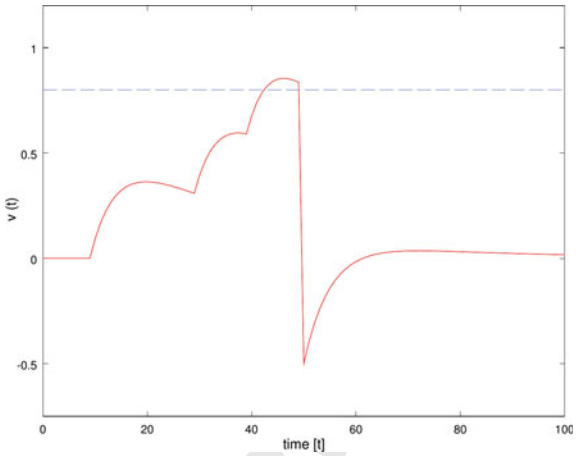

Fig. 6 Simulation of a single LIF neuron in Matlab, the input spikes are applied in t = [10, 30, 40, 50] ms. Between 10 and 30 there is more decrease than between 30 and 40

input voltage passes the threshold, the neuron fires and resets to resting state. The membrane potential stays for a definite period, which is called the *refractory* period, below the reset value.

3.2.3 Izhikevich Neuron Model

Izhikevich neuron model [43] combines the biological plausibility of Hodgkin-Huxley model and the computational efficiency of integrate-and-fire neurons. Using this model, we can simulate tens of thousands of spiking cortical neurons in real time. The model has two main characteristics it is computationally simple as well as capable of producing rich firing patterns that physiological neuron could produce.

$$\frac{\mathrm{d}V(t)}{\mathrm{d}t} = 0.04V(t)^2 + 5V(t) + 140 - u(t) + I(t) \tag{10}$$

$$\frac{\mathrm{d}u(t)}{\mathrm{d}t} = a.(b.V(t) - u(t)) \tag{11}$$

$$\text{if } V(t) \geq 30\,\text{mV}, \text{ then } \begin{cases} V(t) \leftarrow c \\ u(t) \leftarrow u(t) + d \end{cases} \tag{12}$$

where $V(t)$ and $u(t)$ are variables without any dimension, and a, b, c, and d are parameters without dimension. $V(t)$ represents the membrane potential of the neuron and $u(t)$ represents a membrane recovery variable, which accounts for the activation of K^+ ionic currents and inactivation of Na^+ ionic currents, and it provides negative feedback to $V(t)$. Synaptic currents or injected dc-currents are delivered via the variable $I(t)$. The parameter a describes the time scale of the recovery variable

$u(t)$. Smaller value results in slower recovery. The parameter b presents the sensitivity of the recovery variable $u(t)$ to the subthreshold fluctuations of the membrane potential $V(t)$. Greater values couple $V(t)$ and $u(t)$ more strongly resulting in possible subthreshold oscillations and low-threshold spiking dynamics. The parameter c represents the after-spike reset value of the membrane potential $V(t)$ caused by the fast high-threshold K^+ conductances. Finally, the parameter d describes after-spike reset of the recovery variable $u(t)$ caused by slow high-threshold Na^+ and K^+ conductance. Different firing behaviors can occur in biological spiking neurons and Izhikevich model can produce them is shown in Fig. 7.

4 Synapse and Learning

Synapse is a specialized structure with highly plastic characteristics enabling two neurons to exchange spike signals between themselves in other words, adjusting the connection strength between neurons. Thanks to the plasticity property of synapse, we can basically say the synapse is where the learning happens in neural network system. A physiological synapse connects the axon of a presynaptic neuron (the neuron before the synapse) to the dendrite of a postsynaptic neuron (the neuron after the synapse). Two behavioral types of biological synapses are defined:chemical and electrical.

The chemical synapse is the primary definition of neurotransmitters between presynaptic and postsynaptic neurons. A neurotransmitter through a chemical synapse consists of three parts. The axon potential causes the presynaptic neuron to release a chemical substance into the synaptic *cleft* which is an intracellular space between the two neurons. The neurotransmitter then diffuses through the synaptic cleft. Moreover, the neurotransmitter causes a change in the voltage of the membrane of the postsynaptic neuron. In biological neural system, a synapse is *excitatory* if the neurotransmitter causes an increase in the voltage of the postsynaptic neuron and *inhibitory* if it causes a reducing voltage in postsynaptic neuron. An electrical synapse consists of a group of *gap junctions* occurring close together. Gap junctions are tiny channels in the cell membrane that directly connect the cytoplasms of two cells [44]. The basic mechanism of synaptic transmission is well established. A presynaptic spike depolarizes the synaptic terminal, leading to a calcium flow through presynaptic calcium channels, causing vesicles of neurotransmitter to be released into the synaptic cleft. The neurotransmitter binds temporarily to postsynaptic channels, opening them and allowing ionic current to flow across the membrane. Modeling this complete electrochemical behavior is rather challenging. The purpose of our study is not to model the exact behavior of synapse suitable for neuroscience study. The purpose of our study is to design a neuromorphic system appropriate for hardware implementation. Therefore, the behavior of synapse, neuron and model of neuron are studied to compare with recent techniques in addition to recent alternative technologies for hardware implementations.

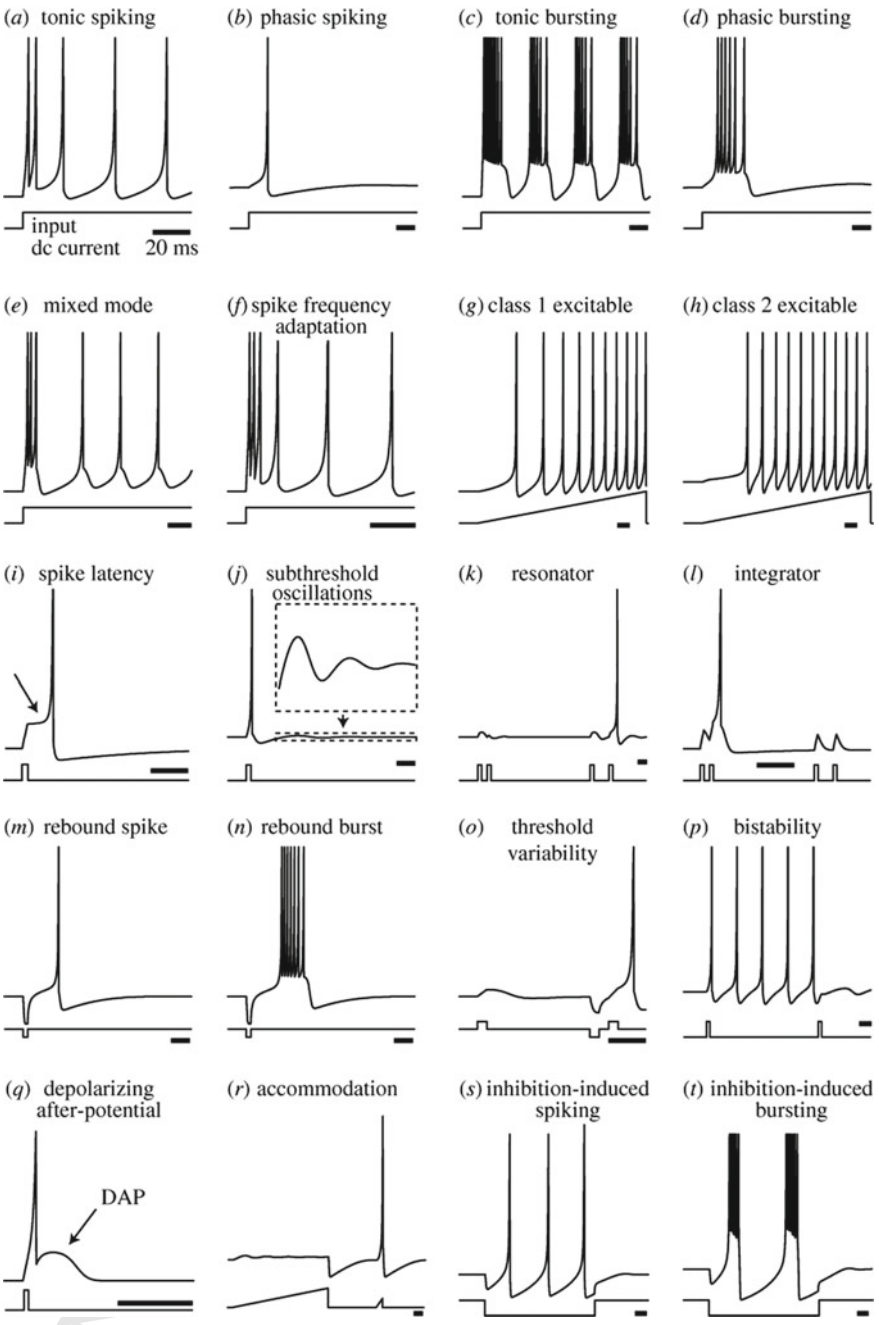

Fig. 7 Different Known types of neurons correspond to different values of the parameters a, b, c, and d could be reproduced by Izhikevich model From [43]

435574_1_En_25_Chapter ☑ TYPESET ☐ DISK ☐ LE ☑ CP Disp.:28/2/2019 Pages: 38 Layout: T1-Standard

4.1 Synapse Model

Emerging devices in nano-scale have demonstrated novel properties for making new memories and unconventional processing units. One of those is the memristor that was hypothetically presented by Leon Chua in 1971 [45] and after a few decades, HP was the first to announce the successful memristor fabrication [46]. The unique properties in memristor nano-devices such as, extreme scalability, flexibility because of analog behavior, and ability to remember the last state make the memristor a very promising candidate to apply it as a synapse in Spiking Neural Network (SNN) [47].

In the recent years, there have been several research works using non-volatile resistive nanodevice as a synapse to build a SNN hardware [11, 47, 48]. Forgetting in the biological brain is an important key of adaptive computation, as without forgetting the biological memory soon becomes overwhelmed by the details of every piece of information ever experienced. Consequently, some studies have been done using volatile memory as a synapse in brain-like computing [49–51].

We combine both volatile and non-volatile types of artificial synapses. It leads to make a synapse which can forget if the information is not important as well as remember if it is significant data. Due to the demonstrated potential of NOMFET (Nanoparticle Organic Memory Field-Effect Transistor) [49, 50] to play the role of a synapse, we use it as a volatile synapse in the synapse box. The non-volatile device could be any solid-state memristor. We have chose here the resistive memory presented in [52] as non-volatile memory. Resistive RAM is modeled in our previous work [53] and is used here as a nonvolatile memristor in the synapse box. As it is shown in Fig. 8b by changing the doped-undoped regions of device, the conductance will be changed. Bigger doped region leads to more conductivity. Therefore by controlling this boundary between two regions, the conductivity is controlled. The behavior of memristor can be modeled as follows [46]:

$$v(t) = R_m i(t) \tag{13}$$

$$R_m = R_{ON}\frac{w(t)}{D} + R_{OFF}\left(1 - \frac{w(t)}{D}\right) \tag{14}$$

where R_m is the variable resistance of memristor, $w(t)$ is the width of the doped region, D is the overall thickness of device, R_{ON} and R_{OFF} are device resistances while the active region is completely doped ($w = D$) and mostly undoped ($w \to 0$) respectively (Fig. 8b). To model the changing of the conductance, we use the model extracted from Eq. 14 and introduced in [54] by considering $g_{max} = \frac{1}{R_{ON}}$ and $g_{min} = \frac{1}{R_{OFF}}$ as the maximum and minimum device conductance respectively.

NOMFET is designed particularly for neuro-inspired computing architectures [50]. NOMFET uses charge trapping/detrapping in an array of gold nanoparticles (NPs) with the SiO_2/pentacene interface designed to mimic dynamic plasticity of a biological synapse as depicted in Fig. 8 [50]. The NOMFET is used as a two-terminal device by connecting drain (D) and gate (G) together and using this terminal as an

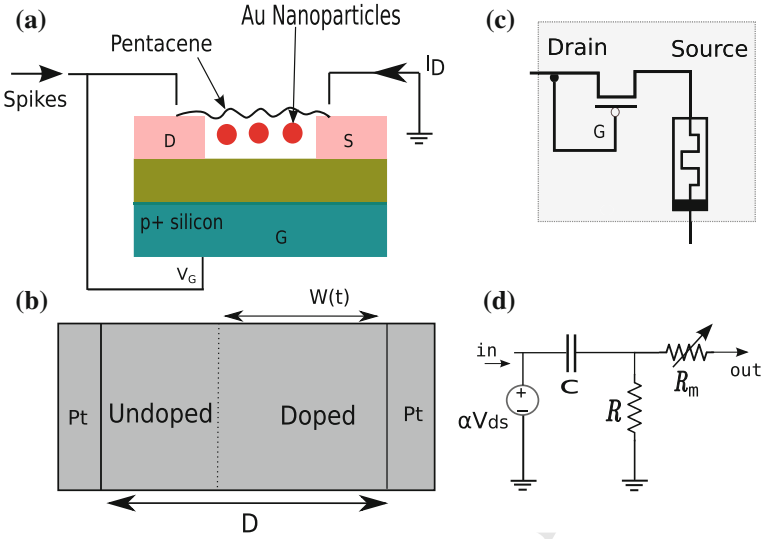

Fig. 8 Artificial synapse: **a** schematic view of the NOMFET as a volatile memory, **b** TiO_2 based nonvolatile memory, **c** synapse box schematic, **d** equivalent circuit with simple elements

input. The source (S) is used as output of the device. Equation 15 shows the behavior of NOMFET as a memristor:

$$i_{ds}(t) = g(q_{np}(t), v_{ds}(t), t)v_{ds} \tag{15}$$

where g is the conductance of the device, $v_{ds}(t)$ is the applied voltage and q_{np} is the charges trapped in the NP. For more details of physical structure and behavior of NOMFET refer to [50, 51].

Figure 8c is the synapse box schematic that we apply in our simulation platform to take the advantages of both nonvolatile and volatile artificial synapses. The equivalent circuit of transistor is depicted in Fig. 8d. Actually, weight modification follows the Short-term potentiation (STP) rule until reaching the Long-term pote potentiation (LTP) threshold in NOMFET. The modification of nonvolatile device is based on STDP learning. Indeed the NOMFET reacts similar to a high-pass filter (HPF). The stimuli spikes with low frequency are not qualified to pass in forgetting Phase. In LTP , stimuli spikes which have more frequency pass to interfere in learning phase (Fig. 6). This synapse box is an approach to improve the quality of synapse for better learning in SNN that have demonstrated better learning performance in SNN rather than nonvolatile memristive synapse [55].

4.2 Learning and Plasticity

To be able to model a proper synapse to contribute in learning process in an efficient way in neural system, we need to analyze how learning happens in synapse. Neurons and synapses are the two basic computational units in the brain. The human brain consists of 10^{11} neurons and an extremely large number of synapses, 10^{15}, which act as a highly complex interconnection network among the neurons.

Subsequently, each neuron is connected to 1000–10000 synapses [56]. Neuron computation is performed by integrating the inputs coming from other neurons and producing spikes as based on variety of the connections. The synapses contribute to the computation by modifying their connection strength as a result of neuronal activity, which is known as the synaptic plasticity. This synaptic plasticity is believed as the basis of adaptation and learning, even in traditional neural network models where several synaptic weight updating rules are based on Hebb's law [57, 58].

4.2.1 Classes of Learning Algorithms

The primary significance of any type of neural networks is the property of learning from the environment to improve the performance of neural network. There are several types of *learning algorithms*. Although interconnection configuration of neural network is important in learning however, learning algorithms generally differ from each other in the way in which they adjust synapse weights. Simon Haykin, mentioned five different basic algorithms for learning in his book [30] namely memory-based, Hebbian, error-correction, competitive, and Boltzmann learning. Memory-based learning functionality is based on memorizing the training data explicitly. Hebbian and competitive learning are inspired by neurobiology. Error-correction is working using optimum filtering rule and Boltzmann learning is based on ideas borrowed from statistical mechanics. In general, learning algorithms can be divided into *supervised or with teacher* learning, *semi-supervised* learning, and *unsupervised or without teacher* learning algorithms.

- Supervised algorithms
  Teacher has the knowledge of environment and this knowledge will be shared with the network as some examples of inputs and their corresponding outputs. The supervision is continued letting a modification rule adjust the synapses until the desired computation emerges as a consequence of the training process. Then the supervision process is stopped and network must have the similar outputs with the specific inputs while the supervision was working. Error-correction algorithms which include the back-propagation using *gradient descent* is an example of supervised algorithms, other well-known supervised algorithms are support vector machines (SVM) and Bayesian type of learning algorithms. In fact, we put label on the data in training and check those labels in testing. This type of algorithms are used for regression and classifications.

- Semi-supervised algorithms
  Semi-supervised learning falls between supervised learning and unsupervised learning. Labeled data are often difficult, expensive, and time consuming to obtain, as they require the efforts of experienced human annotators. Meanwhile unlabeled data may be relatively easy to collect. Semi-supervised uses large amount of unlabeled data, together with the labeled data, to build better classifiers. Intuitively, in semi-supervised learning we can consider the learning problem as an exam and labeled data as the few example problems that the teacher solved in the course. The teacher also provides a set of unsolved problems. Semi-supervised learning requires less human effort and gives higher accuracy, therefore it is of great interest both in theory and in practical application.
- Unsupervised algorithms
  There is no teacher and environment is unknown for the network too. There is no labeled data output in unsupervised learning. Unsupervised learning can be thought of as finding patterns in the data above and beyond what is considered as pure unstructured noise. One very simple classic example of unsupervised learning is clustering. Hebbian plasticity is a form of unsupervised learning, which is useful for clustering input data but less appropriate when a desired outcome for the network is known in advance.

4.2.2 Short-Term and Long-Term Plasticity

Physiological synapses have an inherent dynamics, that controls how the pattern of amplitudes of postsynaptic responses depends on the temporal pattern of the incoming spike train. Indeed, each effective spike evokes a spike response in the postsynaptic neuron that is fewer (depression) or bigger (facilitation or potentiation) than the previous one. The strength of synaptic connections or weights are caused by memorizing events, underling the ability of the brain to memorize. In the biological brain, short-term plasticity refers to a number of phenomena that affect the probability that a presynaptic action potential opens postsynaptic channels and that takes from milliseconds to tens of seconds. Short-term plasticity is achieved through the temporal enhancement of a synaptic connection, which then quickly decays to its initial state. Short-term plasticity depends on the sequence of presynaptic spikes Fig. 9.

In local learning process, iteration of stimulation leads to a more stable change in the connection to achieve long-term plasticity. Long-term plasticity is sensitive to the presynaptic firing rate over a time scale of tens or hundreds of seconds [59]. In general, synapses can exhibit potentiation and depression over a variety of time scales, and multiple components of short- or long-term plasticity. Thus, four combination are possible from short and long term plasticity: Short-term potentiation (STP), short-term depression (STD), Long-term potantiation (LTP) and long-term depression (LTD) [60].

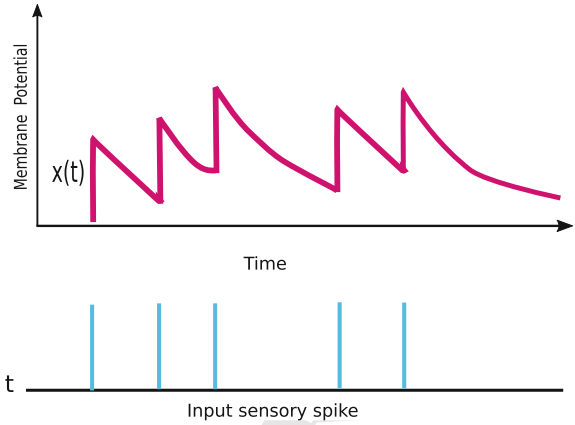

Fig. 9 Implementation of plasticity by local variables which each spike contributes to a trace x(t). The update of the trace depends on the sequence of presynaptic spikes

4.2.3 Spike-Timing Dependent Plasticity (STDP)

Most of the plasticity models employed in the neuroscience and neuromorphic approach were inspired by Hebb's (1949) postulate that explains the way that synapse connection weight should be modified: *When an axon of cell A is near enough to excite cell B or repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.*

Local learning rules aim to deal with information encoded by precise spike timing in local synaptic memory. One of the most commonly studied and used rules is spike-timing-dependent plasticity (STDP) [18, 19] that can be considered as a spike-based producing of Hebbian learning. Based on the STDP modification rule, the synaptic changing is reinforced while both the pre- and post-synaptic neurons are active, nothing prevents the synapses from strengthening themselves boundlessly, which causes the post-synaptic activity to explode [61]. Indeed, the plasticity depends on the time intervals between pre- and postsynaptic spikes or in the other words, the concept of timing-LTP/LTD. The basic mechanisms of plasticity in STDP is derived from the long term potentiation (LTP) and the long term depression (LTD). Pre-synaptic spikes that precede post-synaptic action potentials produce long-term potentiation (LTP), and pre-synaptic spikes that proceed post-synaptic action potentials generate long-term depression (LTD).

The basic configuration of STDP learning is depicted in Fig. 10. The rate of weight changing Δw_{ji} of a synapse from a presynaptic neuron j to postsynaptic neuron i depends on the relative timing between presynaptic spike and postsynaptic spikes. Let us name the presynaptic spike arrival times at synapse j by t_j^{pre} where $pre = 1, 2, 3, \ldots$ counts the presynaptic spikes. Similarly, t_i^{post} with $post = 1, 2, 3, \ldots$ labels the firing times of the postsynaptic neuron. The total weight change w_{ji} induced by Eq. 16 is then [18]

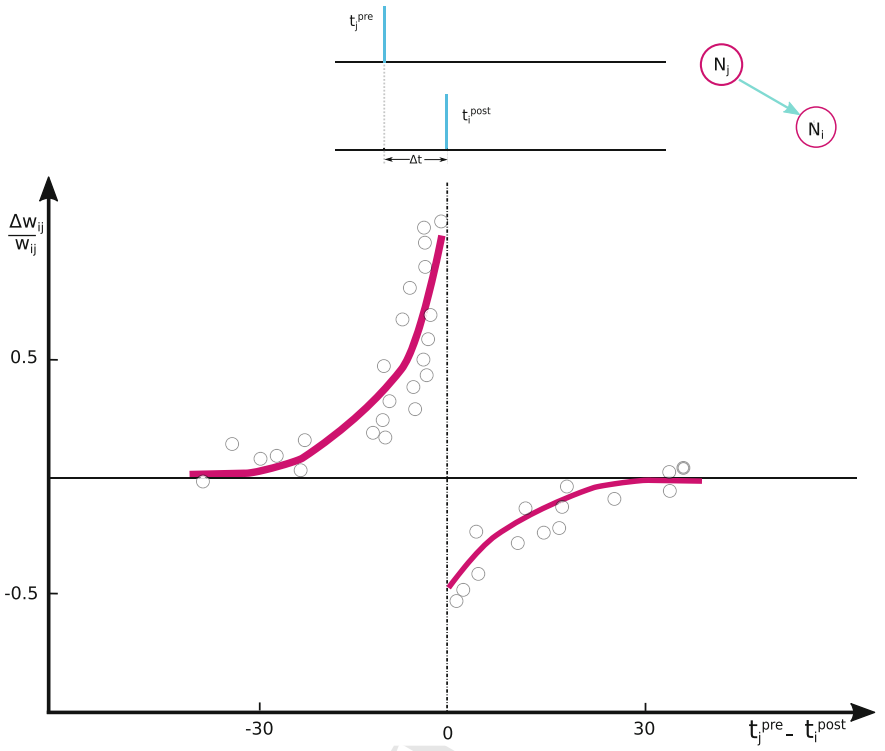

Fig. 10 Basic of spike-timing-dependent plasticity. The STDP function expresses the change of synaptic weight as a function of the relative timing of pre- and post-synaptic spikes

$$\Delta w = \sum_{pre=1}^{n} \sum_{post=1}^{m} W(x)(t_i^{post} - t_j^{pre}) \tag{16}$$

where W(x) is called a STDP learning function. Based on Zhang et al. [62] in their experimental work presented W(x) as:

$$W(x) = \begin{cases} A_+ e^{(\frac{-x}{\tau_+})} & \text{if } x \geq 0 \\ -A_- e^{(\frac{x}{\tau_-})} & \text{if } x < 0 \end{cases} \tag{17}$$

where the parameters A_+ and A_- depend on the current value of the synaptic weight w_{ij}. The time constants τ_+ and τ_- are on the order of 10 ms.

4.2.4 Different Models for STDP Learning

Multiple pre- or postsynaptic spikes occurring across a synapse in an interval of time, the plasticity modification depends on their timing in a more complex manner. For instance, pair-based STDP models present "pre-post-pre" and "post-pre-pos" triplets of spikes with the same pairwise intervals should induce the same plasticity, however experimental studies demonstrated that these two triplet patterns have different effects [63, 64].

- **Pair-based STDP**
  In this model of spike counting in the STDP interpret the biological evidence in terms of a pair-based update rule, i.e. the modification of a synaptic weight depends on the temporal difference between pairs of pre- and postsynaptic spikes:

$$\begin{cases} W_{inc}(x) & = F_{inc}(w).e^{\left(-\frac{|\Delta t|}{\tau_+}\right)} \text{ if } \Delta t > 0 \\ W_{dec}(x) & = -F_{dec}(w).e^{\left(-\frac{|\Delta t|}{\tau_-}\right)} \text{ if } \Delta t < 0 \end{cases} \tag{18}$$

In Eq. 18, $\Delta t = t_i^{post} - t_j^{pre}$ is the temporal difference between the post- and the presynaptic spikes, and $F_{inc}(w)/F_{dec}(w)$ presents the dependence of the update on the current synaptic weight. A pair-based model is fully specified by defining the form of $F_{inc}(w)/F_{dec}(w)$ as well as determining which pairs are taken into account to perform a new modification. A pair-based weight modification rule can be implemented using two local variables: one for a low-pass filtered version of the presynaptic spike train and another one for the postsynaptic spike train as it is shown in Fig. 11. Let us suppose that each spike from presynaptic neuron j contributes to a trace $x_j(t)$ at the synapse weight then we can write:

$$\frac{dx_j(t)}{dt} = -\frac{x_j(t)}{\tau_{pre}} + \sum_{t_j^{pre}} \delta(t - t_j^{pre}) \tag{19}$$

where t_j^{pre} represents the history of the firing times of the presynaptic neuron. In particular, the variable is increased by an amount of one at the arrival time of a presynaptic spike and reduces exponentially with time constant τ_{pre} afterwards. Similarly, each spike from postsynaptic neuron i contributes to a trace $x_i(t)$:

$$\frac{dx_i(t)}{dt} = -\frac{x_i(t)}{\tau_{post}} + \sum_{t_i^{post}} \delta(t - t_i^{post}) \tag{20}$$

where t_i^{post} presents the firing times of the postsynaptic neuron. Similar to presynaptic spike, a decrease of the weight is induced proportionally to the momentary value of the postsynaptic trace $x_i(t)$. The steady-state average for synaptic strength in pair-based STDP has a stable nontrivial mean if the depression window is larger than the potentiation window [64]. This fixed point is unique, so the mean of the

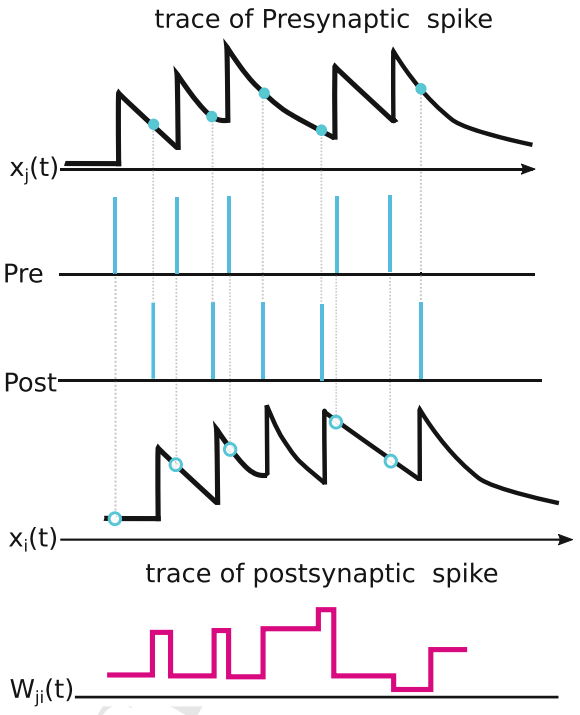

Fig. 11 Pair-based STDP using local variables. The spikes of presynaptic neuron j leave a trace $x_j(t)$ and the spikes of the postsynaptic neuron i leave a trace $x_i(t)$. The update of the weight W_{ji} at the moment of a postsynaptic spike is proportional to the momentary value of the trace $x_j(t)$ (filled circles). This gives the amount of potentiation due to pre-before-post pairings. Analogously, the update of W_{ji} on the occurrence of a presynaptic spike is proportional to the momentary value of the trace $x_i(t)$ (unfilled circles), which gives the amount of depression due to post-before-pre pairings

steady-state distribution of synaptic weights converges to this value regardless of its initial value. The stability of the mean is not a sufficient condition for the steady-state distribution of synaptic strengths to be fully stable, each synapse must also have a stable deviation from the mean. The connection strength of a particular synapse can be presented as $w = w + \delta w$, where δw is the deviation of the synapse from the mean. If the deviation is going to grow over time, the synapses will drift away from the mean and the distribution will be partially stable. If the deviation tends to decrease, the synapses will cluster around the mean and the distribution will be stable.

- **The triplet model**

The standard pair-based STDP models predict that if the repetition frequency is increased, the strength of the depressing interaction becomes greater, leading to less network potentiation. The frequency-dependence of STDP experiments can be accounted for if one assumes that the basic building block of potentiation during STDP experiments is not only a pair-wise interaction but also could be a triplet interaction between two postsynaptic spikes and one presynaptic spike. Pfister and Gerstner [65] to propose the triplet model, which takes into account interactions of spikes beyond pre-post pairings. This model is based on sets of three spikes, one presynaptic and two postsynaptic. For a pre-post-pre triplet, the first presynaptic spike enforces extra depression on the synapse, additionally for a post-pre-post

triplet the first postsynaptic spike enforces extra potentiation. The triplet model sums the contributions of all previous pre- and postsynaptic spikes as well as all pre-post pairings. Pfister and Gerstner [65] also provided a version of the triplet model based only on nearest neighboring spikes, but the qualitative behavior of both all to all and nearest neighboring versions is similar.

Similarly to pair-based rules, each spike from presynaptic neuron j contributes to a trace $x_j(t)$ at the synapse:

$$\frac{dx_j(t)}{dt} = -\frac{x_j(t)}{\tau_{pre}} + \sum_{t_j^{pre}} \delta(t - t_j^{pre}) \tag{21}$$

where t_j^{pre} presents the firing times of the presynaptic neuron. In contrast with pair-based STDP, each spike from postsynaptic neuron i contributes to a fast trace $x_i(t)$ and a slow trace $x_i'(t)$ at the synapse:

$$\frac{dx_i(t)}{dt} = -\frac{x_i(t)}{\tau_{1post}} + \sum_{t_i^{post}} \delta(t - t_i^{post}) \tag{22}$$

$$\frac{dx_i'(t)}{dt} = -\frac{x_i'(t)}{\tau_{2post}} + \sum_{t_i^{post}} \delta(t - t_i^{post}) \tag{23}$$

where $\tau_{1post} < \tau_{2post}$, how the triplet model works is depicted in Fig. 12. In this model, LTD is induced as in the standard STDP pair model in Eq. 18, i.e. the weight change is proportional to the value of the fast postsynaptic trace $x_i(t)$ evaluated at the arrival of a presynaptic spike. The new feature of the rule is that LTP is pursued by a triplet effect: the weight change is proportional to the value of the presynaptic trace $x_j(t)$ evaluated at the arrival time of a postsynaptic spike as well as to the slow postsynaptic trace $x_i'(t)$ from previous postsynaptic spike. The main functional advantage of a triplet STDP rule is that it can be mapped to a Bienenstock-Cooper-Munro learning rule [66]. It means if we assume that the pre- and postsynaptic spike trains are managed by Poisson statistics, the triplet rule presents depression for low postsynaptic firing rates and potentiation for high postsynaptic firing rates.

- **Suppression model**

Plasticity experiments using triplets of spikes demonstrated different effects than the hippocampal results. In the synapses of the visual cortex of rats, pre-post-pre triplets induce potentiation while post-pre-post triplets induce depression. These results led Froemke et al. [67] to develop the suppression model, in which STDP is induced by nearest neighbor pre- and postsynaptic spikes. In this model of STDP, the plasticity is computed from the standard pair-based STDP curve, however the impact of the presynaptic spike in each pair is suppressed by previous presynaptic spikes and, similarly, the plasticity induced by the postsynaptic spike in each pair is suppressed by previous postsynaptic spikes as it is shown in Fig. 13.

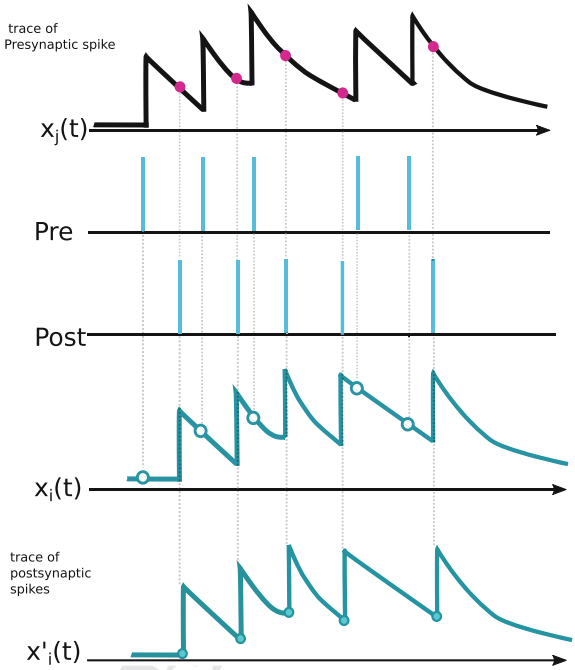

Fig. 12 Triplet STDP model using local variables. The spikes of presynaptic neuron j contribute to a trace $x_j(t)$, the spikes of postsynaptic neuron i contribute to a fast trace $x_i(t)$ and a slow trace $x'_i(t)$. The update of the weight W_{ji} at the arrival of a presynaptic spike is proportional value of the fast trace $x_i(t)$ (green unfilled circles), as in the pair-based model. The update of the weight W_{ji} at the arrival of a postsynaptic spike is proportional to the value of the trace $x_j(t)$ (red filled circles) and the value of the slow trace $x'_i(t)$ just before the spike (green filled circles)

The suppression is maximal after each pre- or postsynaptic spike, and it decreases exponentially as the interval between consecutive pre- or postsynaptic spike increases. In a post-pre-post sequence of spikes, the timing of the first post-pre pairing was the best predictor for the synaptic weight modification. Moreover, in a pre-post-pre sequence of spikes, the first pre-post pair induces potentiation, nevertheless the amount of depression induced by the second post-pre pair is suppressed by the first presynaptic spike. In the suppression STDP model, synaptic weight modification is presented by

$$\Delta w = (1 - e^{-\frac{\Delta t_{pre}}{\tau_{pre}}})(1 - e^{-\frac{\Delta t_{post}}{\tau_{post}}}) \times \begin{cases} A_{inc}.e^{(-\frac{\Delta t}{\tau_{inc}})} & \text{if } \Delta t \geq 0 \\ -A_{dec}.e^{(\frac{\Delta t}{\tau_{dec}})} & \text{if } \Delta t < 0 \end{cases} \quad (24)$$

where Δt_{pre} is the interval between the presynaptic spike in the pair and its preceding presynaptic spike, and Δt_{post} is the interval between the postsynaptic spike and its preceding spike. This model introduces a proper fit to triplet and quadruplet protocols particularly in the visual cortex, and also represents a much better prediction for synaptic changing due to natural spike trains [67]. Nonetheless, it does not predict the increase of LTP with the repetition frequency.

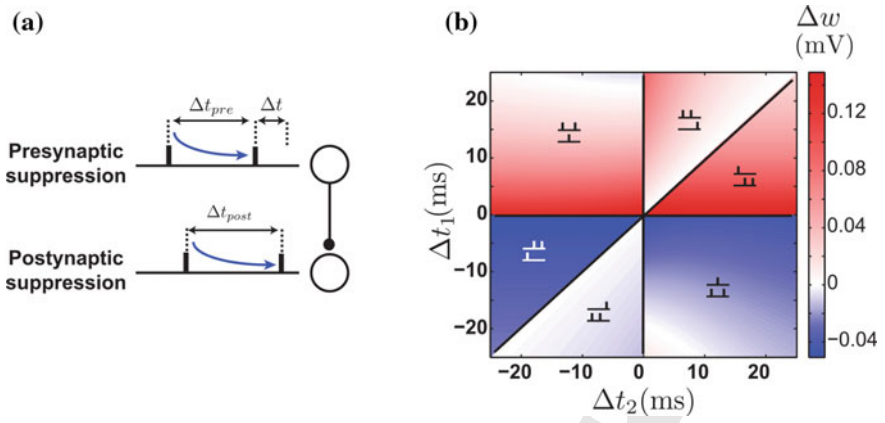

Fig. 13 The suppression STDP model. **a** Spike interactions in the suppression model, in which the impact of the presynaptic spike in a pair is suppressed by a previous presynaptic spike (top), and the impact of the postsynaptic spike is suppressed by a previous postsynaptic spike (bottom). **b** Plasticity in the suppression model induced by triplets of spikes: pre-post-pre triplets induce potentiation (top left), and post-pre-post triplets induce depression (bottom right), From [64]

- **Voltage dependence model**
  Experimental model of Spike-Timing Dependent Plasticity recommends that synaptic weight modifications are caused by the tight temporal correlations between pre- and post- synaptic spikes. However, other experimental protocols where presynaptic spikes are paired with a fixed depolarization of the postsynaptic neuron (e.g. under voltage clamp) show that postsynaptic spikes are not necessary to induce long-term potentiation and depression of the synapse [68].
  It has been discussed whether the voltage dependence is more fundamental than the dependence on postsynaptic spike. In fact, voltage dependence alone can produce a behavior similar to STDP learning, as the membrane potential reacts in a particular manner in the vicinity of a spike it means high shortly before a spike, and low shortly after. Alternatively, a dependence on the slope of the postsynaptic membrane potential has been shown to regenerate the properties of STDP weight change curve. The voltage effects caused by back-propagating spikes is implicitly contained in the mechanistic formulation of STDP models outlined above. In particular, the fast postsynaptic trace $x_i(t)$ in the triplet model can be considered as an approximation of a backpropagating action potential. In contrast, a standalone STDP rule does not automatically generate a voltage dependence. Furthermore, synaptic effects caused by subthreshold depolarization in the absence of postsynaptic firing cannot be modeled by standard STDP or triplet models.
- **The NMDAR-based model**
  The NMDAR-based model was proposed for the first time in [69] and "NMDAR-based model", is phenomenologically based on the kinetics of the N-Methyl-D-Aspartate receptoras. It is a description for the main STDP experiments and resemble both the triplet and suppression models and and it is sensitive to spike

interactions beyond pre-post pairings. The NMDAR-based model is proposed to have three states, rest, up and down. Every presynaptic spike moves a portion of the NMDARs in the rest state into the up state, and every postsynaptic spike transitions a portion of the rest-state into the down state.The NMDAR goes exponentially back to the rest state while there is no spike.

This model introduce two second messengers called "up" and "down" messengers, which cause to potentiation and depression, respectively which can be in active or inactive states. The arrival of presynaptic spike causes to a fraction of the inactive down messengers a transition to the active state. Similarly, when a postsynaptic spike arrives in the synapse, it shifts a portion of the inactive up messengers into their active state. The messengers go back to their inactive states when there is no spike. Subsequently, when a presynaptic spike arrives, the synapse is depressed proportionally to the value of active down messenger, provided that this is greater than a threshold θ^{dn}. Similarly, each postsynaptic spike leads synapse to potentiate proportionally to the amount of active up messenger provided that it is greater than a threshold θ^{up}. Therefore, the presynaptic spike has three roles in this model: it transmits resting NMDARs into the up state, it activates the down messenger, and it induces depression. The postsynaptic spike also plays three roles: it movement resting NMDARs into the down state, it activates the up messenger, as well as it induces potentiation see Fig. 14.

Shortly, the specific property of the NMDAR-based learning model compared to the pair-based model is the possibility of a stable synaptic distribution and anti-Hebbian competition when the maximum depression is significantly larger than the maximum potentiation.

- **Other methods**

In addition to the reviewed methods above, there are other types of STDP models for learning such as supervised [70] and reinforcement learning [71]. However, due to the unsupervised nature of STDP learning that is interesting for neuro-inspired computation, we do not focus on them in this study. Pair-based STDP models can be categorized into three classes: weight dependence, spike-pairing scheme and delay partition. Choosing each category should be made consciously and take into account the relevant available experimental findings. The recent available evidences shows that both potentiation and depression are dependent on the weight. Accordingly it is recommended to begin with very simplified models. Moreover, we know that STDP models which assume some weight dependence generate different behavior from the additive model. The pair-based and triplet models are partially stable and use Hebbian competition. The Suppression and NMDAR-based have more stability but they use anti-Hebbian competition. The main challenge in this domain is to perform analytical and simulation studies that are able to identify and characterize their composite effects, and investigate their functional consequences.

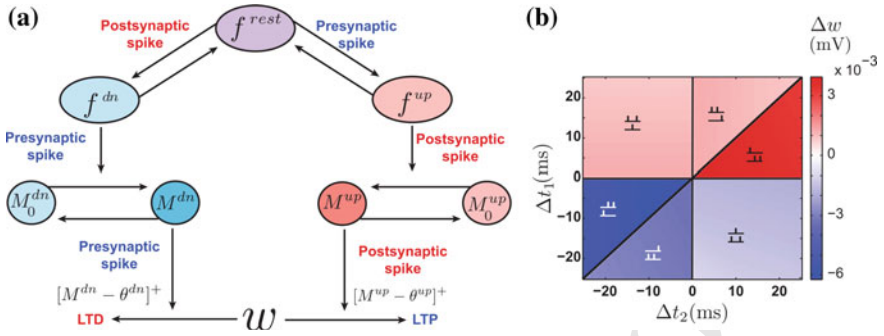

Fig. 14 The NMDAR-based model. **a** Schematic illustration of spike interactions in the NMDAR-based model. The presynaptic spike up-regulates f rest, activates M dn and depresses the synapse. The postsynaptic spike down-regulates f_{rest}, activates M_{up} and potentiates the synapse. **b** The effect is asymmetric, with pre-post-pre triplets inducing potentiation (top left) and post-pre-post depression (bottom right), From [64]

5 Hardware Spiking Neural Network Systems

Specific application domains such as Big Data classification, visual processing, pattern recognition and in general sensory input data, require information processing systems which are able to classify the data and to learn from the patterns in the data. Such systems should be power-efficient. Thus researchers have developed brain-inspired architectures such as spiking neural networks. For large scale brain-like computing on neuromorphic hardware, there are four approaches:

1. Microprocessor based approaches where the system can read the codes to execute and model the behavior of neural systems and cognitive computation such as the SpiNNaker machine [12].
2. Fully digital custom circuits where the neural system components are modeled in circuit using state-of-the-art CMOS technology e.g., IBM TrueNorth machine [11].
3. Analog/digital mixed-signal systems that model the behavior of biological neural systems, e.g. the NeuroGrid [17] and BrainScales [72] projects.
4. Memristor crossbar array based systems where the analog behavior of the memristors emulate the synapses of a spiking neural network.

In the following, we give some details about these approaches and compare their performance.

SpiNNaker is a massively parallel and processor-based (ARM processor) system with the purpose of building large scale spiking neural networks simulations. It is highly scalable and capable to simulate a network from thousands to millions of neurons with varying degree of connectivity. It proposes to integrate 57,600 custom VLSI chips based on the AER (Address Event Representation) communication protocol [73]. Each chip contains 18 fixed-point advanced RISC ARM968 process-

ing cores next to the custom routing infrastructure circuits which is dedicated 96 kB of local memory besides 128 MB of shared Dynamic Random Access Memory (DRAM) as it is depicted in Fig. 15a. The router memory consists of a three-state 1024×32 bits Content Addressable Memory (CAM) and a 1024×24 bits Random Access Memory (RAM). Going more to the details, each ARM core has a local 32 kB instruction memory and 64 kB data memory. Regarding to the architecture and design properties, SpiNNaker offers very fast simulation of large scale neural networks. It has a remarkable flexibility for arbitrary connectivity for network architecture and various neurons, synapses and learning algorithms. However, the system still uses von Neumann architecture with a large extent of memory hierarchies found in conventional computers with memory wall bottleneck issues. Although using low-power ARM processors dedicated to power-efficient platforms used in training and robotic applications with four to 48 nodes, SpiNNaker consumes a relatively small amount of power. However, the largest machine with the ability to simulate of one percent of a human brain and incorporating over a million ARM processor cores, still requires up to 75 kW of electrical power.

IBM designed a scalable, flexible and non-von Neumann full custom spiking neural network named "TrueNorth". Although TrueNorth uses transistors as digital gates, they use event-driven method to communicate in fully asynchronous manner. The structure of TrueNorth consists of 5.4 billion transistors to build 4096 neurosynaptic cores. Each core includes 256 digital LIF neurons, 256×256 binary programmable synapses, and asynchronous encoding/decoding and routing circuits. Each synapse has a binary behavior that can be individually turned on or off and can be assigned to model one type of inhibitory and two types of excitatory synapse with different weights. Neuron dynamics has a global 1 kHz clock and so is discretized into 1 ms time steps. Regarding to the synaptic matrix, each neuron can be connected to one up to 256 neurons of a destination core. The routing in TrueNorth is less flexible than in SpiNNaker, however TrueNorth can distribute the system memory includes core synaptic matrix and routing table entries (Fig. 15b) The architecture thus supports dynamics of connectivity that includes feed-forward, recurrent, and lateral connections. The power consumption is 20 mW/cm^2, though the traditional central processing unit (CPU) is 50–100 W/cm^2. In this platform the synapses do not implement any plasticity mechanism, therefore they are not able to perform on-line learning.

The BrainScales project (Brain-inspired multiscale computation in neuromorphic hybrid systems) is the successor of FACETS [74] project. This project proposes the design and implementation of a custom analog/digital mixed-signal simulation engine that is able to implement the differential equations with an acceptable accuracy. This computational neuroscience model is provided by neuro-scientists, and reproduces the results obtained from numerical simulations executed on conventional computers. The Heidelberg University BrainScales project (HICANN chip) aims to produce a wafer-scale neural simulation platform, in which each 8 inch silicon wafer integrates 50×106 plastic synapses and 200,000 biologically realistic neuron circuits (see Fig. 15c). In order to have a scalable size with maximum number of processors on the wafer, relatively small capacitors have been applied for model-

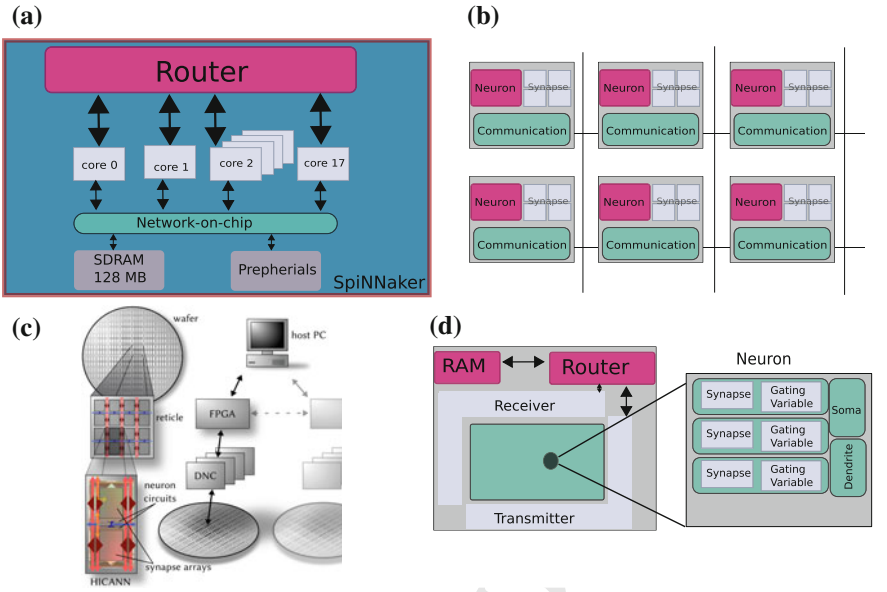

Fig. 15 Large scale spiking neural network systems, **a** Principal architectural parts of a SpiNNaker processing node. **b** In TrueNorth, conceptual blueprint of an architecture like the brain, tightly integrates memory, computation, and communication in distributed modules that operate in parallel and communicate via an event-driven platform. **c** Schematic of HICANN board in BrainScales project. **d** The chip comprises a 256×256 array of neuron elements, an asynchronous digital transmitter for sending the events generated by the neurons, a receiver block for accepting events from other sources, a router block for communicating packets among chips, and a memory blocks for supporting different network configurations

ing the synapses and neurons. Accordingly, using the large currents generated by the above-threshold circuit and the small capacitors, the BrainScales circuits are not able to achieve the long time-constants required for interacting with real-time environments. However, the speed of network components operations compared to biological elements reactions is accelerated by a factor of 10^3 or 10^4 which can reduce the simulation time dramatically. Furthermore, it needs large bandwidth and fast switching and still high-power circuit for propagating spikes across the network [1].

NeuroGrid is another big project developed at Stanford University that emulates neuromorphic engineering vision, sub-threshold network components circuits and uses analog/digital mixed-signal to model continuous time for network elements. This meuromorphic platform simulates a million neurons with billions of synaptic connections in real-time. Such as TrueNorth and BrainScales the architecture of Neurogrid is non-von Neumann. Neurogrid emulates four network elements: axon, dendrite, soma and synapse. Only the axon circuit is digital and the other elements are modeled in the analog circuits due to the better energy efficiency. NeuroGrid consists of 16 standard CMOS "NeuroCores" (see Fig. 15d) integrated on a board that works using 3 W of power energy connected in a tree network, with each NeuroCore con-

sisting of a 256 × 256 array of two-compartmental neurons. The synaptic circuits are shared among the neurons while different spikes can be assigned to the same synapse. The main goal of neuromorphic systems is to interact with real physical environments and process the natural signals with physiological time-scales, Neurogrid has long time constants in the range of tens of milliseconds. Consequently, this long time constants limitation causes difficulty in using typical VLSI for design and implementation. Neurogrid and BrainScales similarly use the temporal dynamic of memory elements to store the state of the network. Accordingly, these two projects have the capability of local learning using the STDP learning rule.

An alternative to these architectures, that has been proposed by several authors [47–49, 75, 76], is to use memristive devices as synapses in neuromorphic circuits. This has the potential to lower the energy consumption by a large proportion. It has also been showed that the memristors can emulate the STDP learning rule, and thus lead to unsupervised learning circuits. We have thus chosen to study this kind of architecture and, in particular, to check how some parameters of the architecture or of the devices influence the learning capabilities of the circuit.

6 Discussion

Still for a network simulation and implementation of neuromorphic spiking system, we need more techniques such as homeostasis and lateral inhibition to support learning process for an optimized system. Homeostasis is used in the SNN to adapt the threshold level of neurons to learning in SNN. Another consideration is lateral inhibition while we are using unsupervised learning methods such as STDP. Here we discuss Winner Take-All (WTA) method.

6.1 Homeostasis

AQ3 Homeostasis addresses a general principle that safeguards the stability of natural and artificial neural systems, where stability is understood in its more classical sense of robustness against external perturbations. Homeostasis is a fundamental concept in neuropsychology, psychophysiology and neuroscience. Homeostasis will be defined as negative feedback control. In physiological neural systems, the synaptic input of a neuron is changing over time due to the external neural drive and learning results of synaptic plasticity. From a perspective of metabolic cost, a restricted activity range of a neuron is really meaningful.

In STDP learning, the synaptic input of a neuron may strongly increase or decrease for a long time and as a result the neural activity will be drifting to an extremely high or low level. Homeostasis is a neuron property that regulates the firing threshold to prevent a neuron to be hyperactive [77]. The idea is to use an adaptive threshold for the membrane potential. If the neuron is too much active in a short time window the

threshold grows gradually; likewise, when a neuron is not active in a certain time window the threshold is reduced slightly.

$$\frac{dV_{th}}{dx} = \gamma(fr_{mean} - fr_{target}) \tag{25}$$

where fr_{mean} is the mean activity (or firing rate) of a neuron, fr_{target} is the target activity, and γ is a multiplicative positive constant. Consequently, the activity of the neuron is bounded in a homeostatic range to encode the synaptic input more effectively to improve the STDP learning [78].

6.2 Winner-Take-All

In a winner-take-all (WTA) network, in output layer or partially output layers, neurons compete with each other based on their output activities, which leads to an adaptation only of the weights of the neuron with the highest output activity [79]. In unsupervised learning using spike coding and plasticity learning. Without competition, all the neurons would behave alike and no specialization takes place in the neurons. The theoretical analysis shows that winner-take-all is a surprisingly powerful computational method compared with threshold gate (McCulloch-Pitts neuron) and sigmoidal gate [80]. There have been many implementations of winner take all (WTA) computations in recurrent networks in the literature [81, 82]. Also there have been many analog VLSI implementations of these circuit [82, 83]. In WTA, after the competition, only one neuron will be the most active for some inputs and the rest of the neurons will eventually become inactive for those inputs. Physiologically plausible learning methods can be mainly classified as dense, local, or sparse. Competitive learning such as WTA is a local learning rule as it activates only the unit that fits the input pattern best and suppresses the others through fixed inhibitory connections.

The simplest competitive computational model is a hard WTA that computes a function $f_{WTA}:\mathbb{R}^n \rightarrow \{0, 1\}^n$ whose output $\langle b_1, \ldots, b_n\rangle = f_{WTA}(x_1, \ldots, x_n)$ satisfies

$$b_i = \begin{cases} 1 \text{ if } x_i > x_j \quad \text{for all} \quad j \neq i \\ 0 \text{ if } x_i < x_j \text{ for some } j \neq i \end{cases} \tag{26}$$

Therefore in the case of inputs $x_1, \ldots, x_n$ a single output b_i has values 1 that marks the position of the biggest input x_i. Wolfgang Maass [80] introduced two types of WTA namely *k-WTA* and *soft-WTA*. In k-WTA, b_i has value 1 if and only if x_i is among the k largest inputs. In soft-WTA the i_{th} output is an analog variable r_i whose value reflects the rank of x_i among the input variables. We use WTA in our research that will be presented in the next sections.

7 Conclusion

Neuromorphic computation using Spiking Neural Networks (SNN) is proposed as an alternative solution for future of computation to conquer the memory bottleneck issue in recent computer architecture. Different spike codings have been discussed to improve data transferring and data processing in neuro-inspired computation paradigms. Choosing the appropriate neural network topology could result in better performance of computation, recognition and classification. The model of the neuron is another important factor to design and implement SNN systems. The speed of simulation and implementation, ability of integration to the other elements of the network, and suitability for scalable networks are the factors to select a neuron model. The learning algorithms are significant consideration to train the neural network for weight modification. Improving learning in neuromorphic architecture is feasible by improving the quality of artificial synapse as well as learning algorithm such as STDP. In this chapter we proposed a new synapse box that can remember and forget. Furthermore, as the most frequent used unsupervised method for network training in SNN is STDP, we analyzed and reviewed the various methods of STDP. The sequential order of pre- or postsynaptic spikes occurring across a synapse in an interval of time leads to defining different STDP methods. Based on the importance of stability as well as Hebbian competition or anti-Hebbian competition the method will be used in weight modification. We surveyed the most significant projects that cause making neuromorphic platform. The advantages and disadvantages of each neuromorphic platform have been introduced in this chapter.

References

1. Indiveri, G., Liu, S.C.: Proc. IEEE **103**(8), 1379 (2015). https://doi.org/10.1109/JPROC.2015.2444094. ArXiv: 1506.03264
2. Rajendran, B., Alibart, F.: IEEE J. Emerg. Sel. Top. Circuits Syst. **6**(2), 198 (2016). https://doi.org/10.1109/JETCAS.2016.2533298
3. Lapicque, L.: J. Physiol. Pathol. Gen. **9**, 620 (1907). http://www.pubmedcentral.nih.gov/tocrender.fcgi?journal=484&action=archive
4. Mead, C.: Proc. IEEE **78**(10), 1629 (1990). https://doi.org/10.1109/5.58356
5. Mahowald, M.A., Mead, C.: Sci. Am. **264**(5), 76 (1991)
6. Delbruck, T., Mead, C.A.: In: Proceedings of IEEE International Symposium on Circuits and Systems—ISCAS '94, vol. 4, pp. 339–342 (1994). https://doi.org/10.1109/ISCAS.1994.409266
7. Sarpeshkar, R., Lyon, R.F., Mead, C.: In: Lande, T.S. (ed.) Neuromorphic Systems Engineering: Neural Networks in Silicon, pp. 49–103. Kluwer Academic, Boston, MA (1998). http://resolver.caltech.edu/CaltechAUTHORS:20150112-105156628
8. Chiu, S.W., Tang, K.T.: Sensors **13**(10), 14214 (2013). https://doi.org/10.3390/s131014214. http://www.mdpi.com/1424-8220/13/10/14214
9. Liu, S.C., Schaik, A.V., Minch, B.A., Delbruck, T.: IEEE Trans. Biomed. Circuits Syst. **8**(4), 453 (2014). https://doi.org/10.1109/TBCAS.2013.2281834
10. Vanarse, A., Osseiran, A., Rassau, A.: Front. Neurosci. **10**, (2016). https://doi.org/10.3389/fnins.2016.00115. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4809886/

11. Merolla, P.A., Arthur, J.V., Alvarez-Icaza, R., Cassidy, A.S., Sawada, J., Akopyan, F., Jackson, B.L., Imam, N., Guo, C., Nakamura, Y., Brezzo, B., Vo, I., Esser, S.K., Appuswamy, R., Taba, B., Amir, A., Flickner, M.D., Risk, W.P., Manohar, R., Modha, D.S.: Science **345**(6197), 668 (2014). https://doi.org/10.1126/science.1254642.
12. Furber, S.B., Galluppi, F., Temple, S., Plana, L.A.: Proc. IEEE **102**(5), 652 (2014). https://doi.org/10.1109/JPROC.2014.2304638
13. Jo, S.H., Chang, T., Ebong, I., Bhadviya, B.B., Mazumder, P., Lu, W.: Nano Lett. **10**(4), 1297 (2010). https://doi.org/10.1021/nl904092h
14. Shi, L., Pei, J., Deng, N., Wang, D., Deng, L., Wang, Y., Zhang, Y., Chen, F., Zhao, M., Song, S., Zeng, F., Li, G., Li, H., Ma, C.: In: 2015 IEEE International Electron Devices Meeting (IEDM), pp. 4.3.1–4.3.4 (2015)
15. O'Connor, P., Neil, D., Liu, S.C., Delbruck, T., Pfeiffer, M.: Neuromorphic Eng. **7**, 178 (2013). https://doi.org/10.3389/fnins.2013.00178
16. Prezioso, M., Merrikh-Bayat, F., Hoskins, B., Adam, G., Likharev, K.K., Strukov, D.B.: Nature **521**(7550), 61 (2015). https://doi.org/10.1038/nature14441. ArXiv:1412.0611
17. Benjamin, B.V., Gao, P., McQuinn, E., Choudhary, S., Chandrasekaran, A.R., Bussat, J.M., Alvarez-Icaza, R., Arthur, J.V., Merolla, P.A., Boahen, K.: Proc. IEEE **102**(5), 699 (2014). https://doi.org/10.1109/JPROC.2014.2313565
18. Kempter, R., Gerstner, W., van Hemmen, J.L.: Phys. Rev. E **59**(4), 4498 (1999). https://doi.org/10.1103/PhysRevE.59.4498
19. Song, S., Miller, K.D., Abbott, L.F.: Nat. Neurosci. **3**(9), 919 (2000). https://doi.org/10.1038/78829
20. McCulloch, W.S., Pitts, W.: The bulletin of mathematical biophysics **5**(4), 115 (1943). https://doi.org/10.1007/BF02478259
21. VanRullen, R., Guyonneau, R., Thorpe, S.J.: Trends Neurosci. **28**(1), 1 (2005). https://doi.org/10.1016/j.tins.2004.10.010. http://www.sciencedirect.com/science/article/pii/S0166223604003546
22. Brette, R.: Front. Syst. Neurosci. 151 (2015). https://doi.org/10.3389/fnsys.2015.00151
23. Bohte, S.M.: Nat. Comput. **3**(2), 195 (2004). https://doi.org/10.1023/B:NACO.0000027755.02868.60
24. Ponulak, F., Kasiski, A.: Neural Comput. **22**(2), 467 (2010). https://doi.org/10.1162/neco.2009.11-08-901
25. Masquelier, T., Hugues, E., Deco, G., Thorpe, S.: J. Neurosci. **29**(43), 13484 (2009). https://doi.org/10.1523/JNEUROSCI.2207-09.2009
26. Chen, H.T., Ng, K.T., Bermak, A., Law, M.K., Martinez, D.: IEEE Trans. Biomed. Circuits Syst. **5**(2), 160 (2011). https://doi.org/10.1109/TBCAS.2010.2075928
27. Thorpe, S., Delorme, A., Van Rullen, R.: Neural Netw. **14**(6–7), 715 (2001). https://doi.org/10.1016/S0893-6080(01)00083-1. http://www.sciencedirect.com/science/article/pii/S0893608001000831
28. Shamir, M.: Curr. Opin. Neurobiol. **25**, 140 (2014). https://doi.org/10.1016/j.conb.2014.01.002. http://www.sciencedirect.com/science/article/pii/S0959438814000105
29. Olshausen, B.A., Field, D.J.: Nature **381**(6583), 607 (1996). https://doi.org/10.1038/381607a0. http://www.nature.com/nature/journal/v381/n6583/abs/381607a0.html
30. Haykin, S.: Neural Networks: A Comprehensive Foundation, 2nd edn. Prentice Hall PTR, Upper Saddle River, NJ, USA (1998)
31. Lipton, Z.C., Berkowitz, J., Elkan, C.: arXiv:1506.00019 [cs] (2015). ArXiv: 1506.00019
32. Hopfield, J.J.: Proc. Natl. Acad. Sci. **79**(8), 2554 (1982). http://www.pnas.org/content/79/8/2554
33. Elman, J.L.: Cogn. Sci. **14**(2), 179 (1990). https://doi.org/10.1207/s15516709cog1402
34. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Proc. IEEE **86**(11), 2278 (1998). https://doi.org/10.1109/5.726791
35. Krizhevsky, A., Sutskever, I., Hinton, G.: **2**, 1097–1105 (2012)
36. Hinton, G.E., Osindero, S., Teh, Y.W.: Neural Comput. **18**(7), 1527 (2006). https://doi.org/10.1162/neco.2006.18.7.1527

37. Neil, D., Liu, S.C.: IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **22**(12), 2621 (2014). https://doi.org/10.1109/TVLSI.2013.2294916
38. Hodgkin, A.L., Huxley, A.F.: J. Physiol. **117**(4), 500 (1952). http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1392413/
39. Chicca, E., Badoni, D., Dante, V., D'Andreagiovanni, M., Salina, G., Carota, L., Fusi, S., Giudice, P.D.: IEEE Trans. Neural Netw. **14**(5), 1297 (2003). https://doi.org/10.1109/TNN.2003.816367
40. Liu, S.C., Douglas, R.: IEEE Trans. Neural Netw. **15**(5), 1305 (2004). https://doi.org/10.1109/TNN.2004.832725
41. Maass, W., Bishop, C.M. (eds.): Pulsed Neural Networks. MIT Press, Cambridge, MA, USA (1999)
42. Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J.M., Diesmann, M., Morrison, A., Goodman, P.H., Harris, F.C., Zirpe, M., Natschlger, T., Pecevski, D., Ermentrout, B., Djurfeldt, M., Lansner, A., Rochel, O., Vieville, T., Muller, E., Davison, A.P., El Boustani, S., Destexhe, A.: J. Comput. Neurosci. **23**(3), 349 (2007). https://doi.org/10.1007/s10827-007-0038-6
43. Izhikevich, E.M.: IEEE Trans. Neural Netw./a publication of the IEEE Neural Networks Council **14**(6), 1569 (2003). https://doi.org/10.1109/TNN.2003.820440
44. Hu, E.H., Bloomfield, S.A.: J. Neurosci.: Official J. Soc. Neurosci. **23**(17), 6768 (2003)
45. Chua, L.: IEEE Trans. Circuit Theory **18**(5), 507 (1971). https://doi.org/10.1109/TCT.1971.1083337
46. Strukov, D.B., Snider, G.S., Stewart, D.R., Williams, R.S.: Nature **453**(7191), 80 (2008). https://doi.org/10.1038/nature06932. http://www.nature.com/nature/journal/v453/n7191/full/nature06932.html
47. Jo, S.H., Chang, T., Ebong, I., Bhadviya, B.B., Mazumder, P., Lu, W.: Nano Lett. **10**(4), 1297 (2010). https://doi.org/10.1021/nl904092h
48. Indiveri, G., Linares-Barranco, B., Legenstein, R., Deligeorgis, G., Prodromakis, T.: Nanotechnology **24**(38), 384010 (2013). https://doi.org/10.1088/0957-4484/24/38/384010. ArXiv: 1302.7007
49. Alibart, F., Pleutin, S., Gurin, D., Novembre, C., Lenfant, S., Lmimouni, K., Gamrat, C., Vuillaume, D.: Adv. Funct. Mater. **20**(2), 330 (2010). https://doi.org/10.1002/adfm.200901335
50. Alibart, F., Pleutin, S., Bichler, O., Gamrat, C., Serrano-Gotarredona, T., Linares-Barranco, B., Vuillaume, D.: Adv. Funct. Mater. **22**(3), 609 (2012). https://doi.org/10.1002/adfm.201101935. http://www.sciencedirect.com/science/article/pii/S1566119915000786
51. Desbief, S., Kyndiah, A., Gurin, D., Gentili, D., Murgia, M., Lenfant, S., Alibart, F., Cramer, T., Biscarini, F., Vuillaume, D.: Org. Electron. **21**, 47 (2015). https://doi.org/10.1016/j.orgel.2015.02.021. http://www.sciencedirect.com/science/article/pii/S1566119915000786
52. Querlioz, D., Bichler, O., Dollfus, P., Gamrat, C.: IEEE Trans. Nanotechnol. **12**(3), 288 (2013). https://doi.org/10.1109/TNANO.2013.2250995
53. Shahsavari, M., Faisal Nadeem, M., Arash Ostadzadeh, S., Devienne, P., Boulet, P.: Phys. Status Solidi (c) **12**(1–2), 222 (2015). https://doi.org/10.1002/pssc.201400069
54. Querlioz, D., Dollfus, P., Bichler, O., Gamrat, C.: In: 2011 IEEE/ACM International Symposium on Nanoscale Architectures, pp. 150–156 (2011). https://doi.org/10.1109/NANOARCH.2011.5941497
55. Shahsavari, M., Falez, P., Boulet, P.: In: 12th ACM/IEEE International Symposium on Nanoscale Architectures (Nanoarch 2016), Beijing, China (2016)
56. Drachman, D.A.: Neurology **64**(12), 2004 (2005). https://doi.org/10.1212/01.WNL.0000166914.38327.BB
57. Morris, R.G.: Brain Res. Bull. **50**(5–6), 437 (1999)
58. Hebb, D.: Wiley, New York (3) (1949)
59. Morrison, A., Diesmann, M., Gerstner, W.: Biol. Cybern. **98**(6), 459 (2008). https://doi.org/10.1007/s00422-008-0233-1. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2799003/
60. Dayan, P., Abbott, L.F.: Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems. The MIT Press (2005)

61. Yger, P., Gilson, M.: Front. Comput. Neurosci. 138 (2015). https://doi.org/10.3389/fncom.2015.00138
62. Zhang, L.I., Tao, H.W., Holt, C.E., Harris, W.A., Poo, M.M.: Nature **395**(6697), 37 (1998). https://doi.org/10.1038/25665. http://www.nature.com/nature/journal/v395/n6697/abs/395037a0.html
63. Wang, H.X., Gerkin, R.C., Nauen, D.W., Bi, G.Q.: Nat. Neurosci. **8**(2), 187 (2005). https://doi.org/10.1038/nn1387. http://www.nature.com/neuro/journal/v8/n2/abs/nn1387.html
64. Babadi, B., Abbott, L.F.: PLOS Comput. Biol. **12**(3), e1004750 (2016). https://doi.org/10.1371/journal.pcbi.1004750. http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004750
65. Pfister, J.P., Gerstner, W.: J. Neurosci. **26**(38), 9673 (2006). https://doi.org/10.1523/JNEUROSCI.1425-06.2006. http://www.jneurosci.org/content/26/38/9673
66. Bienenstock, E.L., Cooper, L.N., Munro, P.W.: J. Neurosci.: Official J. Soc. Neurosci. **2**(1), 32 (1982)
67. Froemke, R.C., Dan, Y.: Nature **416**(6879), 433 (2002). https://doi.org/10.1038/416433a. http://www.nature.com/nature/journal/v416/n6879/abs/416433a.html
68. Sjstrm, P.J., Turrigiano, G.G., Nelson, S.B.: J. Neurophysiol. **92**(6), 3338 (2004). https://doi.org/10.1152/jn.00376.2004
69. Senn, W., Markram, H., Tsodyks, M.: Neural Comput. **13**(1), 35 (2001)
70. Frgnac, Y., Shulz, D.E.: J. Neurobiol. **41**(1), 69 (1999)
71. Pfister, J.P., Toyoizumi, T., Barber, D., Gerstner, W.: Neural Comput. **18**(6), 1318 (2006). https://doi.org/10.1162/neco.2006.18.6.1318
72. Schemmel, J., Briiderle, D., Griibl, A., Hock, M., Meier, K., Millner, S.: IEEE, pp. 1947–1950 (2010). https://doi.org/10.1109/ISCAS.2010.5536970. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5536970
73. Boahen, K.: IEEE Trans. Circuits Syst. II: Analog Digit. Sig. Process. **47**(5), 416 (2000). https://doi.org/10.1109/82.842110. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=842110
74. Wendt, K., Ehrlich, M., Schffny, R.: In: Proceedings of the 2nd WSEAS International Conference on Computer Engineering and Applications. World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, 2008, CEA'08, pp. 189–194. http://dl.acm.org/citation.cfm?id=1373936.1373969
75. Yang, J., Pickett, M., Li, X., Ohlberg, D., Stewart, D., Williams, R.: Nat. Nanotechnol. **3**(7), 429 (2008). http://www.scopus.com/inward/record.url?eid=2-s2.0-46749093701&partnerID=40&md5=f2a7152ab8e0922c0eabc0c44f89ee7b
76. Borghetti, J., Li, Z., Straznicky, J., Li, X., Ohlberg, D.A.A., Wu, W., Stewart, D.R., Williams, R.S.: Proc. Natl. Acad. Sci. **106**(6), 1699 (2009). https://doi.org/10.1073/pnas.0806642106
77. Marder, E., Goaillard, J.M.: Nat. Rev. Neurosci. **7**(7), 563 (2006). https://doi.org/10.1038/nrn1949. http://www.nature.com/nrn/journal/v7/n7/full/nrn1949.html
78. Li, C., Li, Y.: IEEE Access **4**, 119 (2016). https://doi.org/10.1109/ACCESS.2015.2509005
79. Grossberg, S.: Biol. Cybern. **23**(3), 121 (1976). https://doi.org/10.1007/BF00344744
80. Maass, W.: Neural Comput. **12**(11), 2519 (2000). https://doi.org/10.1162/089976600300014827
81. Jin, D.Z., Seung, H.S.: Phys. Rev. E Stat. Nonlin. Soft Matter Phys. **65**(5 Pt 1), 051922 (2002). https://doi.org/10.1103/PhysRevE.65.051922
82. Oster, M., Liu, S.C.: In: Proceedings of the 2004 11th IEEE International Conference on Electronics, Circuits and Systems, 2004. ICECS 2004, pp. 203–206 (2004). https://doi.org/10.1109/ICECS.2004.1399650
83. Hafliger, P.: IEEE Trans. Neural Netw. **18**(2), 551 (2007). https://doi.org/10.1109/TNN.2006.884676