



**HAL**  
open science

# Spiking Neural Computing in Memristive Neuromorphic Platforms

Mahyar Shahsavari, Philippe Devienne, Pierre Boulet

► **To cite this version:**

Mahyar Shahsavari, Philippe Devienne, Pierre Boulet. Spiking Neural Computing in Memristive Neuromorphic Platforms. Springer Nature, Handbook of Memristor Networks, , inPress, 978-3-319-76375-0. 10.1007/978-3-319-76375-0\_25 . hal-02172472

**HAL Id: hal-02172472**

**<https://hal.science/hal-02172472>**

Submitted on 3 Jul 2019

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# 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 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 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](mailto:mahyar.shahsavari@gmail.com)

© Springer Nature Switzerland AG 2019  
L. Chua et al. (eds.), *Handbook of Memristor Networks*,  
[https://doi.org/10.1007/978-3-319-76375-0\\_25](https://doi.org/10.1007/978-3-319-76375-0_25)

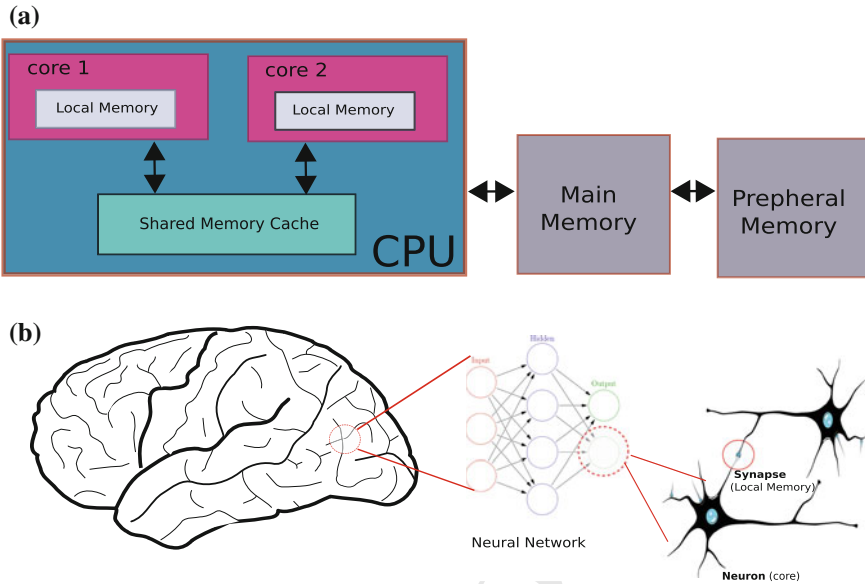
1

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

30 The main remarkable difference between conventional Von Neumann architec-  
31 ture and neuromorphic systems is in their use of memory structures. The way of  
32 communication between memory and central processing unit (CPU) in conventional  
33 computing is not efficient. The memory and CPU communication suffers from what  
34 is called Von Neumann memory bottleneck. The CPUs access both data and pro-  
35 gram in memory using the same shared resources. CPUs spend most of their time  
36 idle because the speed of CPU is much more than memory due to the quality of  
37 materials applied to manufacturing the transistors in CPU and different memories.

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

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



**Fig. 1** Computational architecture **a** Von Neumann architecture, fast and costly memory are closer to cores in multiprocessor platforms as caches 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 bottleneck issue, performing parallel and cognitive computing, as well as considering that the synapses are local memories connected to each neurons as computational cores

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

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

## 106 2 Spiking Neural Networks

107 Artificial neural networks (ANN) can generally be categorized into three generations.  
108 The first generation of neural network consisted of McCulloch and Pitts neurons [20]  
109 that the output signals are limited to discrete ‘0’ or ‘1’ binary values. Perceptrons,  
110 Hopfield network, Boltzmann machine and multilayer networks with threshold units  
111 are ANN examples that are classified in first generation. The second generation of  
112 neural network, by using a continuous activation function such as sigmoid, polyno-  
113 mial or exponential functions, the output can take analog values between ‘0’ and  
114 ‘1’. Due to using analog output the network requires less neurons than the first gen-  
115 eration class. Radial Basis Function (RBF) networks and Multi-Layer Perceptrons  
116 (MLP) are categorized under second generation class. The third generation of neural  
117 network model are networks which employ spiking neurons as computational units.  
118 In this model, the precise firing times of neurons are used for information coding.  
119 Spiking neural networks belong to the third generation of neural networks.

120 Indeed, artificial neural network in the first and second generation is a mathemati-  
121 cal model of mammalian brain though, SNN is an electronic hardware neuromorphic  
122 model of the biological brain. Networks composed of spiking neurons are able to  
123 process significant amount of data using a relatively small number of spikes [21].  
124 Due to the similarity between the biological neurons and spiking models function-  
125 ality, SNNs provide powerful tools to emulate data processing in the brain, including  
126 neural information processing, plasticity and learning. Consequently, spiking net-  
127 works offer solutions to a broad range of specific problems in applied engineering  
128 image detection, event detection, classification, speech recognition and many cogni-  
129 tive computation domain applications.

## 130 2.1 Spike Information Coding

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

### 136 1. Rate coding

137 The rate of spikes in a time-window is counted for the information transmission.  
138 It is also called as frequency coding (Fig. 2a). As the intensity of a stimulus  
139 increases more, the firing rate of spikes increases more too. Rate encoding is  
140 motivated by the observation that biological neurons eager to fire more often  
141 for stronger stimuli. There are two types of rate coding namely spike-count rate  
142 and time-dependent firing rate. In spike-count rating by counting the number of  
143 spikes that are generated during a trial and dividing by the duration of the trial, we  
144 calculate the temporal average of rating. In independent firing rate, the average  
145 number of spikes over trial happens during a short interval between times  $t$  and  $t$   
146  $+ \Delta t$ , divided by the duration of the interval. Brette [22] has compared these two  
147 approaches in rate information coding in more details.

### 148 2. Latency coding

149 In this model, information is supposed to be contained in the exact timing of a  
150 set of spikes relative to each other as it is shown in Fig. 2b. It is already proved  
151 that precisely timed patterns of spikes have been postulated to play a significant  
152 role in the networks of neuron in different functions [23]. Precise spike timing is  
153 one of the important parameters that control variety forms of synaptic plasticity.  
154 Latency coding by using sequences of spikes are mainly observed in feed-forward  
155 networks since noise and dynamics of recurrent networks can disrupt spike timing  
156 precision, some attempts to harvest precise spiking timing in recurrent networks  
157 have been done for example by exploring the idea of reservoir computation [24].

### 158 3. Phase coding

159 This model generates the times of emitted spikes based on the time point in a  
160 periodic signal. In this (Fig. 2c) method the spike trains can encode information  
161 in the phase of a pulse respecting to the background oscillations. Phase coding  
162 method has been used both in models and experimentally. Phase coding has been  
163 suggested for the hippocampus as well [25]. Spiking networks exploring the phase  
164 coding strategy have recently been applied in tasks as olfactory systems or robot  
165 navigation [26].

### 166 4. Rank-coding (spike-order coding)

167 In this method of spike coding, information is encoded by the order of spikes  
168 in the activity of a group of neurons as it is depicted in Fig. 2d. Rank-coding  
169 approach has been suggested to describe ultra-fast categorization observed in the  
170 visual system. This model assumes that each neuron emits only a single spike  
171 during a presentation of the image. This method can be implemented in a feed-  
172 forward network with inhibitory feedback connections. Thorpe and others [27]

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

## 175 5. Population coding

176 This coding model is a method to introduce stimuli by applying the joint activities  
 177 of the group of neurons. In population coding, each neuron has a distribution of  
 178 responses to the certain set of inputs, and the responses of group of neurons will be  
 179 combined to present a value for the inputs (Fig. 2e). During the last two decades,  
 180 the theory has focused on analyzing the methods in which different parameters  
 181 that characterize neuronal responses to external stimuli affect the information  
 182 content of these responses. Recent challenge in population coding is to develop a  
 183 theory that can generate predictions for specific readout mechanisms for example  
 184 for visual target information [28].

## 185 6. Sparse coding

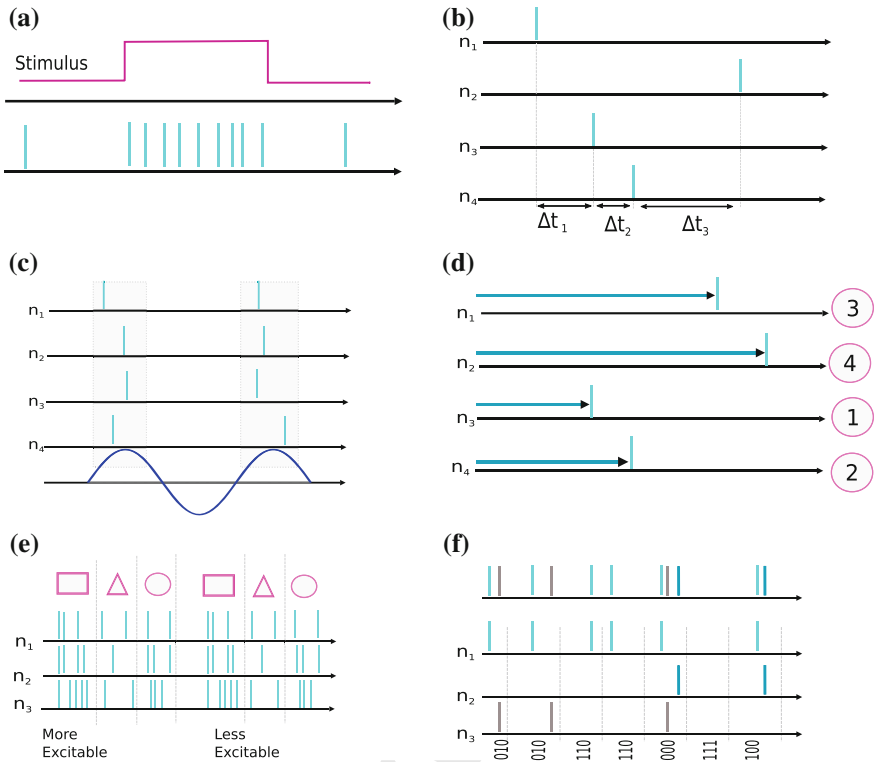
186 This model of coding generally refers to a representation where a few number  
 187 of neurons are active, with the majority of the neurons inactive or showing low  
 188 activity see Fig. 2f. Sparse coding has been suggested as a guiding principle in  
 189 neural representations of sensory input, specially in the visual sensory system.  
 190 It is also discussed that sparse coding offers a useful solution to the problem  
 191 of representing natural data because such a scheme allows the system to take  
 192 advantage of the sparse structure of the sensory environment. It is believed that  
 193 the natural environment is inherently sparse and codes that using this structure can  
 194 be both metabolically efficient and useful for learning. Sparseness can be defined  
 195 over a population of neurons at a specific point in time (population sparseness)  
 196 or it can be measured for a single neuron over a certain time-window [29].

## 197 2.2 Network Topology

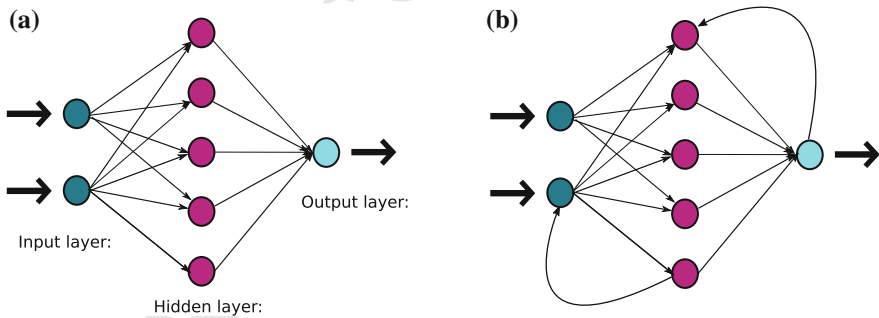
198 The interconnection structure of neurons in a network of neurons is called topology,  
 199 architecture or graph of an artificial neural network. The manner in which the inter-  
 200 connection is structured intimately is linked to the learning algorithms applied to  
 201 train the neural networks.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

### 205 2.2.1 Feed-Forward Neural Networks (FFNN)

206 The FFNN is divided into two different structure called single-layer FFNN and  
 207 multilayer FFNN. The single-layer is structured as an input and output layer which  
 208 is strictly a feed-forward or acyclic graph. We do not count the input layer because  
 209 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)



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

- 218 • **fully connected**

219 In this configuration, every node in each layer of the network is connected to  
 220 every other node in the next layer. In fact, we can call them globally connected  
 221 networks. The Restricted Boltzmann Machine (RBM) could be an example of fully  
 222 connected FFNN.

- 223 • **partially connected**

224 In this configuration, some communication links are missing. The convolutional  
 225 neural networks is a good example for the partially connected FFNN. Partially  
 226 connected topologies present a suitable alternative with a reduced degree of redund-  
 227 dancy and thus a potential for increased efficiency of neural networks.

## 228 2.2.2 Recurrent Neural Networks (RNN)

229 The RNN is distinguished from FFNN in that it has at least one *feedback* loop con-  
 230 nection. Recurrent neural networks can be single-layer or multilayer as well. Unlike  
 231 feed-forward neural networks, recurrent networks retain a state that can represent  
 232 information from an arbitrarily long context window. Although recurrent neural net-  
 233 works have traditionally been difficult to train, and often contain thousands of param-  
 234 eters, recent studies in network architectures, optimization techniques, and parallel  
 235 computation have enabled successful large-scale learning to use RNN [31]. *Hopfield*  
 236 [32] network is an example of the recurrent artificial neural network that is used to  
 237 store one or more stable vectors. The stable vectors can be considered as memories  
 238 that the network recalls them when provided with similar vectors that operate as a  
 239 queue to the network memory. Other example of RNN is *Elman* network [33] that  
 240 refers as a simple Recurrent Network is the special case of recurrent artificial neural  
 241 networks. This type of artificial neural network has the memory that allows it to both  
 242 detect and generate time-varying patterns.

## 243 2.2.3 Modern Neural Networks

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

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

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

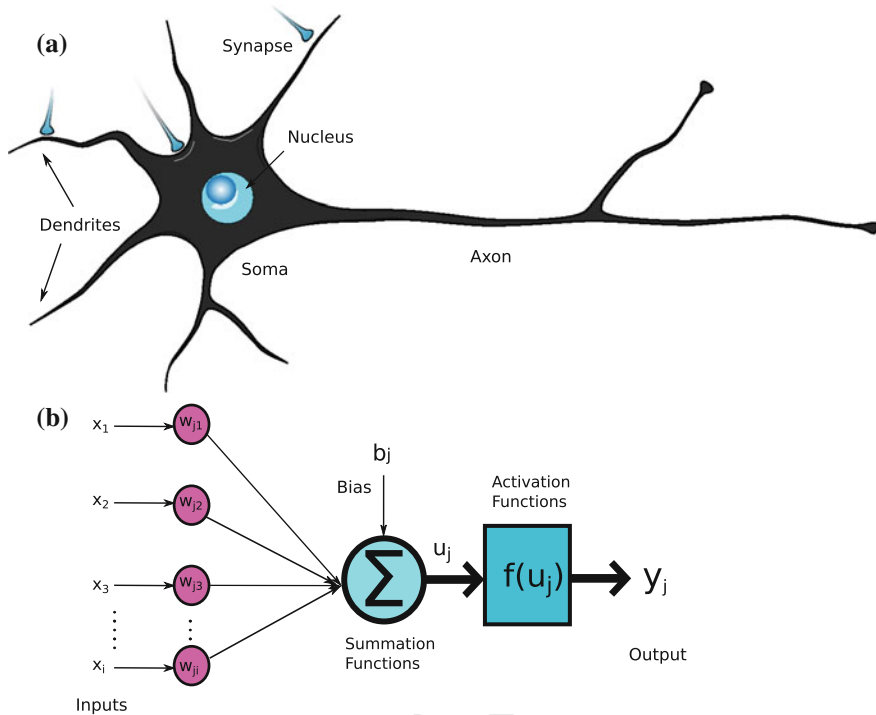
260 • **Deep Belief Networks (DBN)** Deep learning is currently an extremely active  
261 research area in machine learning and cognitive computing society. It has obtained  
262 many successes in a wide area of applications such as speech recognition, com-  
263 puter vision, and natural language processing.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

### 273 3 Spiking Neuron Model

274 The neuron is a dynamic element and processing unit that emits output pulses when-  
275 ever the excitation exceeds some threshold. The resulting sequence of pulses or  
276 “spikes” contains all the information that is transmitted from one neuron to the other  
277 one. In this section, we compare the biological, artificial and spiking neuron and  
278 furthermore, we explain various model of spiking neuron models.

#### 279 3.1 Biological, Artificial and Spiking Neuron

280 A biological neuron is an electrically excitable cell that processes and transmits infor-  
281 mation by electrochemical signals. Chemical signaling occurs via synapses, special-  
282 ized connections with other cells. A typical physiological neuron can be divided into  
283 three anatomical and functional parts, called *dendrites*, *soma* and *axon* as it is shown  
284 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

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

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

$$u_j = \sum_1^i w_{ji} x_i \quad (1)$$

$$y_j = f(u_j + b_j) \quad (2)$$

It is obvious in Fig. 4b that  $x_1, x_2, x_3, \dots, x_i$  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  $\theta$ . Here, we choose  $\theta = 0$  in the way we could perform a binary classification.

$$y_j = \begin{cases} 1 & \text{if } u_j \geq 0 \\ 0 & \text{if } u_j < 0 \end{cases} \quad (3)$$

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

$$u_j = \sum_1^i w_{ji} x_i + b_j \quad (4)$$

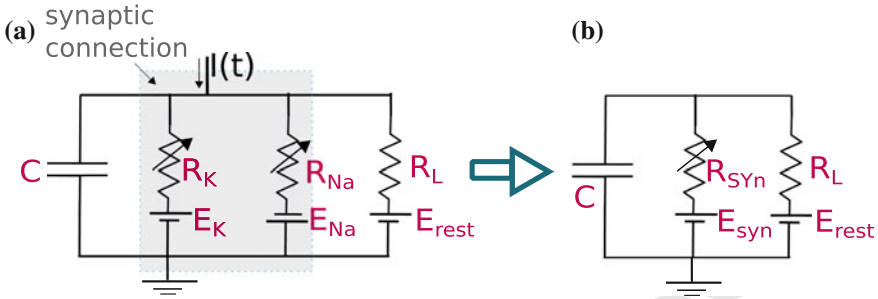
Such a model of neuron is referred to McCulloch and Pitts [20].

## 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-Huxley 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

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

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

$$346 \quad I_L(t) = \frac{V_C(t) - E_{rest}}{R_L} \quad (5)$$

$$347 \quad I_{syn}(t) = C \frac{dV_C(t)}{dt} + \frac{V_C(t) - E_{rest}}{R_L} \quad (6)$$

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

$$350 \quad V_C(t) = v_\infty(1 - \exp(-\frac{t}{\tau})) + E_{rest} \quad (7)$$

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

### 354 3.2.2 Integrate-and-Fire (I&F) Neurons

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

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

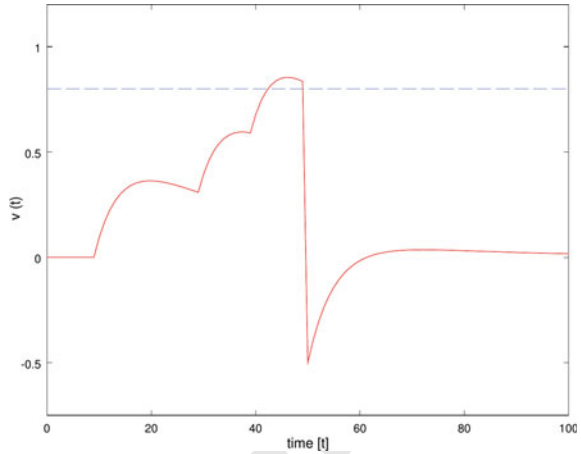
364 Neurons integrate the spike inputs from other neurons they are connected to. These  
 365 input spikes change the internal potential of the neuron, it is known as neuron's mem-  
 366 brane potential or state variable. When this membrane potential passes a threshold  
 367 voltage due to integrated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

$$369 \quad \tau_n \frac{dv}{dt} = -v(t) + RI_{syn}(t) \quad (8)$$

$$370 \quad I_{syn}(t) = \sum_j g_{ij} \sum_n \alpha(t - t_j^{(n)}) \quad (9)$$

372 where,  $v(t)$  represents the membrane potential at time  $t$ ,  $\tau_n = RC$  is the membrane  
 373 time constant and  $R$  is the membrane resistance. Equation 8 describes a simple  
 374 parallel resistor-capacitor (RC) circuit where the leakage term is due to the resistor  
 375 and the integration of  $I_{syn}(t)$  is due to the capacitor. The total input current,  $I_{syn}(t)$ ,  
 376 is generated by the activity of pre-synaptic neurons. In fact, each pre-synaptic spike  
 377 generates a post-synaptic current pulse. The total input current injected to a neuron  
 378 is the sum over all current pulses which is calculated in Eq. 9. Time  $t_j^{(n)}$  represents  
 379 the time of the  $n_{th}$  spike of post-synaptic neuron  $j$ , and  $g_{ij}$  is the conductance of  
 380 synaptic efficacy between neuron  $i$  and neuron  $j$ . Function  $\alpha(t) = q\delta(t)$ , where  $q$   
 381 is the injected charge to the artificial synapse and  $\delta(t)$  is the Dirac pulse function. If  
 382  $I_{syn}(t)$  is big enough where action potential can pass the threshold voltage, neuron  
 383 fires. It means there are enough input spikes in a short time window. When there is  
 384 no or only a few spikes in a time window, the neuron is in the leaky phase and the  
 385 state variable decreases exponentially. The duration of this time window depends on  
 386  $\tau_n = RC$ . The equation is analytically solvable and thus we use the answer of Eq. 8  
 387 in the network simulation when there is an input spike to improve the simulation  
 388 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



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

### 392 3.2.3 Izhikevich Neuron Model

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

$$398 \quad \frac{dV(t)}{dt} = 0.04V(t)^2 + 5V(t) + 140 - u(t) + I(t) \quad (10)$$

$$399 \quad \frac{du(t)}{dt} = a.(b.V(t) - u(t)) \quad (11)$$

$$401 \quad \text{if } V(t) \geq 30 \text{ mV, then } \begin{cases} V(t) \leftarrow c \\ u(t) \leftarrow u(t) + d \end{cases} \quad (12)$$

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

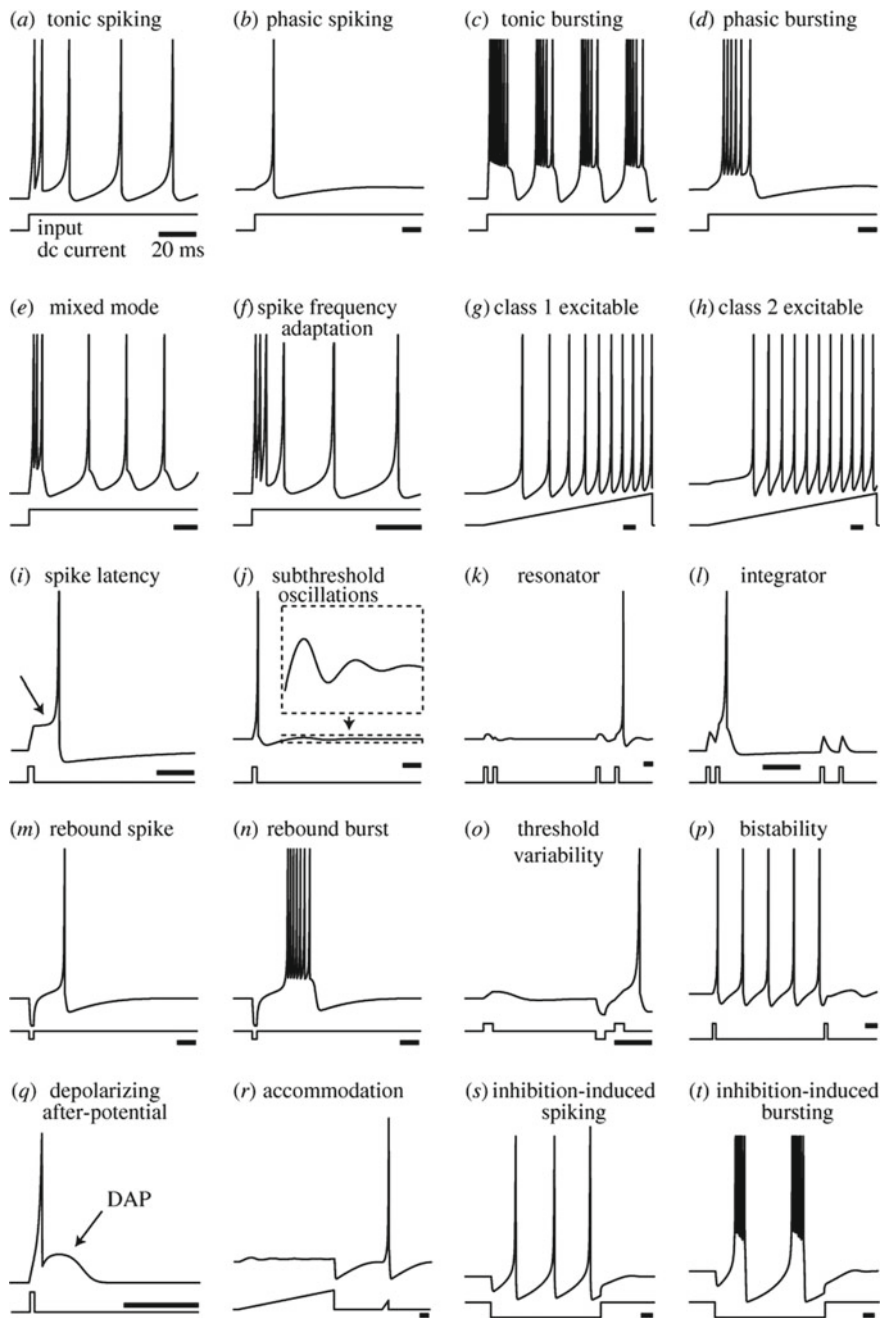
408  $u(t)$ . Smaller value results in slower recovery. The parameter  $b$  presents the sensi-  
409 tivity of the recovery variable  $u(t)$  to the subthreshold fluctuations of the membrane  
410 potential  $V(t)$ . Greater values couple  $V(t)$  and  $u(t)$  more strongly resulting in possible  
411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

## 417 4 Synapse and Learning

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

426 The chemical synapse is the primary definition of neurotransmitters between  
427 presynaptic and postsynaptic neurons. A neurotransmitter through a chemical synapse  
428 consists of three parts. The axon potential causes the presynaptic neuron to release a  
429 chemical substance into the synaptic *cleft* which is an intracellular space between the  
430 two neurons. The neurotransmitter then diffuses through the synaptic cleft. Moreover,  
431 the neurotransmitter causes a change in the voltage of the membrane of the postsynap-  
432 tic neuron. In biological neural system, a synapse is *excitatory* if the neurotransmitter  
433 causes an increase in the voltage of the postsynaptic neuron and *inhibitory* if it causes  
434 a reducing voltage in postsynaptic neuron. An electrical synapse consists of a group  
435 of *gap junctions* occurring close together. Gap junctions are tiny channels in the cell  
436 membrane that directly connect the cytoplasm of two cells [44]. The basic mecha-  
437 nism of synaptic transmission is well established. A presynaptic spike depolarizes the  
438 synaptic terminal, leading to a calcium flow through presynaptic calcium channels,  
439 causing vesicles of neurotransmitter to be released into the synaptic cleft. The neuro-  
440 transmitter binds temporarily to postsynaptic channels, opening them and allowing  
441 ionic current to flow across the membrane. Modeling this complete electrochemical  
442 behavior is rather challenging. The purpose of our study is not to model the exact  
443 behavior of synapse suitable for neuroscience study. The purpose of our study is  
444 to design a neuromorphic system appropriate for hardware implementation. There-  
445 fore, the behavior of synapse, neuron and model of neuron are studied to compare  
446 with recent techniques in addition to recent alternative technologies for hardware  
447 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]

## 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].

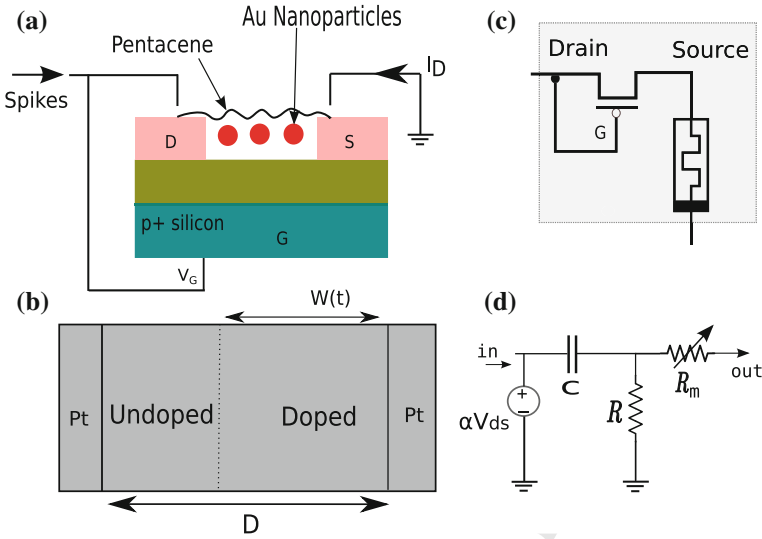
We combine both volatile and non-volatile types of artificial synapses. It leads to make a synapse which can forget if the information is not important as well as remember if it is significant data. Due to the demonstrated potential of NOMFET (Nanoparticle Organic Memory Field-Effect Transistor) [49, 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]:

$$v(t) = R_m i(t) \quad (13)$$

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

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 ( $w \rightarrow 0$ ) 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<sub>2</sub>/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<sub>2</sub> based nonvolatile memory, **c** synapse box schematic, **d** equivalent circuit with simple elements

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

$$489 \quad i_{ds}(t) = g(q_{np}(t), v_{ds}(t), t)v_{ds} \quad (15)$$

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

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

## 504 4.2 Learning and Plasticity

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

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

### 517 4.2.1 Classes of Learning Algorithms

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

- 531 • Supervised algorithms

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

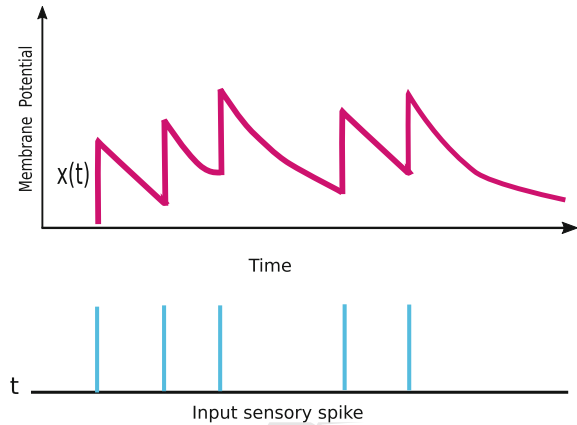
- 543 ● Semi-supervised algorithms  
 544 Semi-supervised learning falls between supervised learning and unsupervised  
 545 learning. Labeled data are often difficult, expensive, and time consuming to obtain,  
 546 as they require the efforts of experienced human annotators. Meanwhile unlabeled  
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

#### 562 4.2.2 Short-Term and Long-Term Plasticity

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

575 In local learning process, iteration of stimulation leads to a more stable change  
 576 in the connection to achieve long-term plasticity. Long-term plasticity is sensitive  
 577 to the presynaptic firing rate over a time scale of tens or hundreds of seconds [59].  
 578 In general, synapses can exhibit potentiation and depression over a variety of time  
 579 scales, and multiple components of short- or long-term plasticity. Thus, four combina-  
 580 tion are possible from short and long term plasticity: Short-term potentiation  
 581 (STP), short-term depression (STD), Long-term potentiation (LTP) and long-term  
 582 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

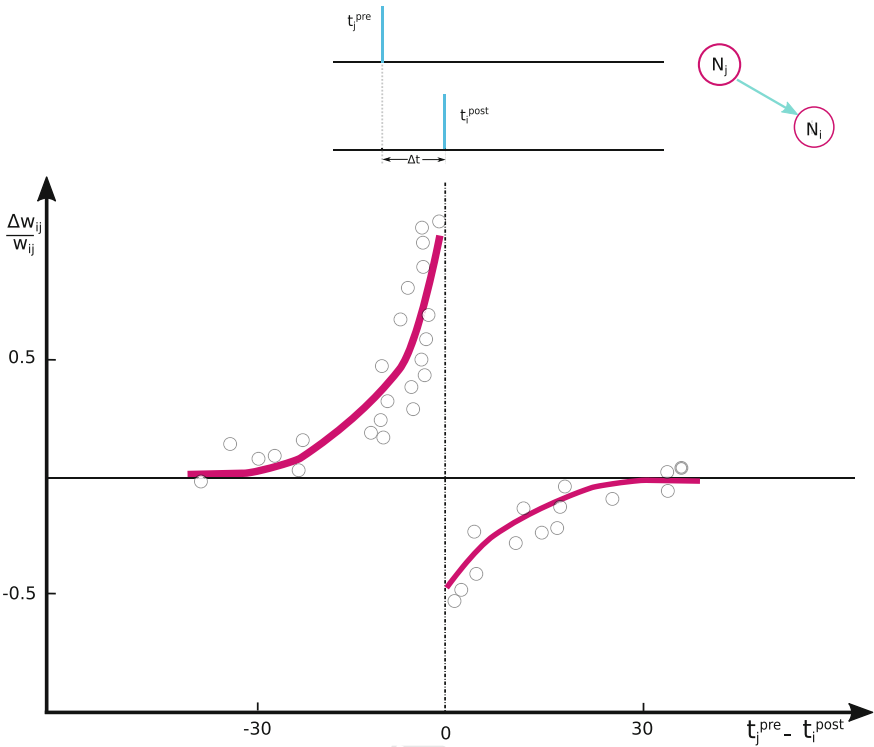


### 4.2.3 Spike-Timing Dependent Plasticity (STDP)

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  $\Delta w_{ji}$  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  $pre = 1, 2, 3, \dots$  counts the presynaptic spikes. Similarly,  $t_i^{post}$  with  $post = 1, 2, 3, \dots$  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

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

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

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

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

#### 616 4.2.4 Different Models for STDP Learning

617 Multiple pre- or postsynaptic spikes occurring across a synapse in an interval of time,  
 618 the plasticity modification depends on their timing in a more complex manner. For  
 619 instance, pair-based STDP models present “pre-post-pre” and “post-pre-pos” triplets  
 620 of spikes with the same pairwise intervals should induce the same plasticity, however  
 621 experimental studies demonstrated that these two triplet patterns have different effects  
 622 [63, 64].

##### 623 • Pair-based STDP

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

$$627 \begin{cases} W_{inc}(x) = F_{inc}(w) \cdot e^{\left(-\frac{\Delta t}{\tau_+}\right)} & \text{if } \Delta t > 0 \\ W_{dec}(x) = -F_{dec}(w) \cdot e^{\left(-\frac{|\Delta t|}{\tau_-}\right)} & \text{if } \Delta t < 0 \end{cases} \quad (18)$$

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

$$637 \frac{dx_j(t)}{dt} = -\frac{x_j(t)}{\tau_{pre}} + \sum_{t_j^{pre}} \delta(t - t_j^{pre}) \quad (19)$$

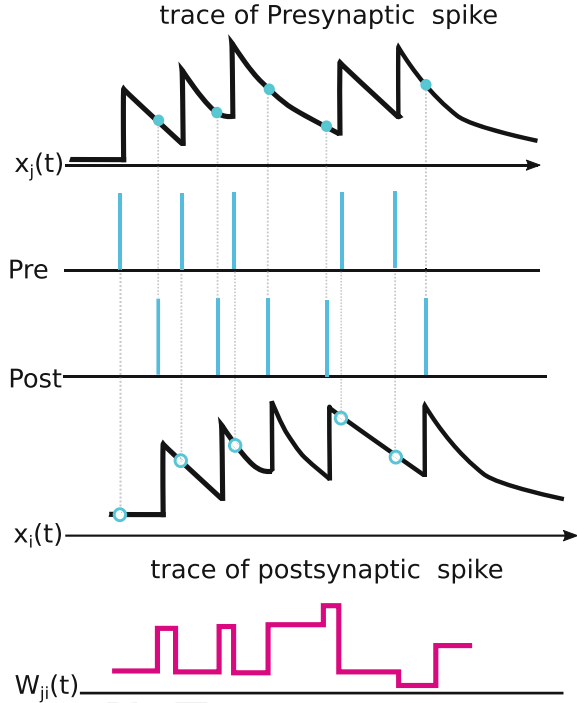
638 where  $t_j^{pre}$  represents the history of the firing times of the presynaptic neuron. In  
 639 particular, the variable is increased by an amount of one at the arrival time of a  
 640 presynaptic spike and reduces exponentially with time constant  $\tau_{pre}$  afterwards.  
 641 Similarly,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

$$642 \frac{dx_i(t)}{dt} = -\frac{x_i(t)}{\tau_{post}} + \sum_{t_i^{post}} \delta(t - t_i^{post}) \quad (20)$$

643 where  $t_i^{post}$  presents the firing times of the postsynaptic neuron. Similar to presy-  
 644 naptic spike, a decrease of the weight is induced proportionally to the momentary  
 645 value of the postsynaptic trace  $x_i(t)$ . The steady-state average for synaptic strength  
 646 in pair-based STDP has a stable nontrivial mean if the depression window is larger  
 647 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



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

#### 657 • The triplet model

658 The standard pair-based STDP models predict that if the repetition frequency is  
 659 increased, the strength of the depressing interaction becomes greater, leading to  
 660 less network potentiation. The frequency-dependence of STDP experiments can  
 661 be accounted for if one assumes that the basic building block of potentiation during  
 662 STDP experiments is not only a pair-wise interaction but also could be a triplet  
 663 interaction between two postsynaptic spikes and one presynaptic spike. Pfister and  
 664 Gerstner [65] to propose the triplet model, which takes into account interactions of  
 665 spikes beyond pre-post pairings. This model is based on sets of three spikes, one  
 666 presynaptic and two postsynaptic. For a pre-post-pre triplet, the first presynaptic  
 667 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:

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

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:

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

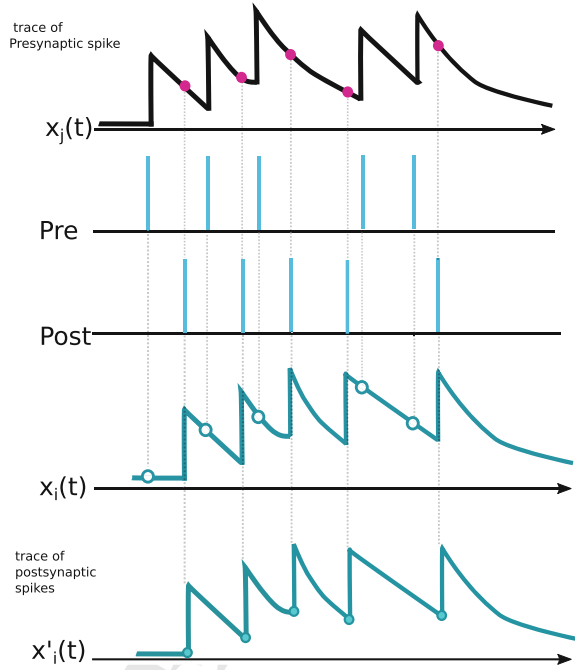
$$\frac{dx'_i(t)}{dt} = -\frac{x'_i(t)}{\tau_{2post}} + \sum_{t_i^{post}} \delta(t - t_i^{post}) \quad (23)$$

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.

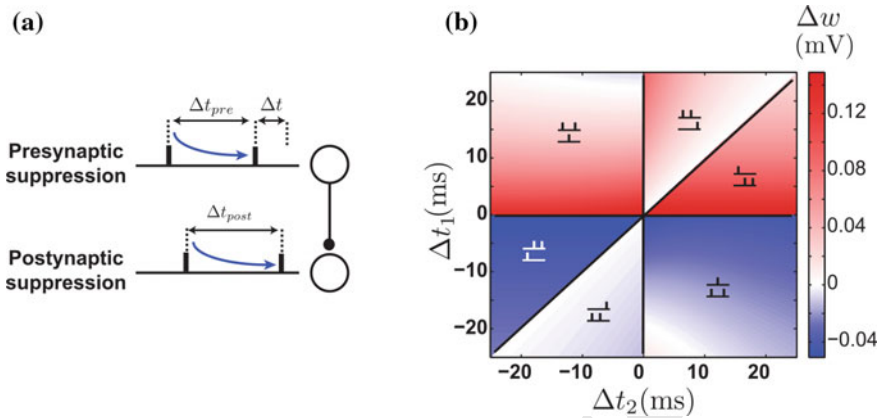
**Fig. 12** Triplet STDP model using local variables. The spikes of presynaptic neuron  $j$  contribute to a trace  $x_j(t)$ , the spikes of postsynaptic neuron  $i$  contribute to a fast trace  $x_i(t)$  and a slow trace  $x'_i(t)$ . The update of the weight  $W_{ji}$  at the arrival of a presynaptic 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)



704 The suppression is maximal after each pre- or postsynaptic spike, and it decreases  
 705 exponentially as the interval between consecutive pre- or postsynaptic spike  
 706 increases. In a post-pre-post sequence of spikes, the timing of the first post-pre  
 707 pairing was the best predictor for the synaptic weight modification. Moreover, in a  
 708 pre-post-pre sequence of spikes, the first pre-post pair induces potentiation, never-  
 709 theless the amount of depression induced by the second post-pre pair is suppressed  
 710 by the first presynaptic spike. In the suppression STDP model, synaptic weight  
 711 modification is presented by

$$712 \quad \Delta w = (1 - e^{-\frac{\Delta t_{pre}}{\tau_{pre}}})(1 - e^{-\frac{\Delta t_{post}}{\tau_{post}}}) \times \begin{cases} A_{inc} \cdot e^{-\frac{\Delta t}{\tau_{inc}}} & \text{if } \Delta t \geq 0 \\ -A_{dec} \cdot e^{-\frac{\Delta t}{\tau_{dec}}} & \text{if } \Delta t < 0 \end{cases} \quad (24)$$

713 where  $\Delta t_{pre}$  is the interval between the presynaptic spike in the pair and its pre-  
 714 ceding presynaptic spike, and  $\Delta t_{post}$  is the interval between the postsynaptic spike  
 715 and its preceding spike. This model introduces a proper fit to triplet and quadru-  
 716 plet protocols particularly in the visual cortex, and also represents a much better  
 717 prediction for synaptic changing due to natural spike trains [67]. Nonetheless, it  
 718 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 receptors. It is a description for the main STDP experiments and resemble both the triplet and suppression models and and it is sensitive to spike

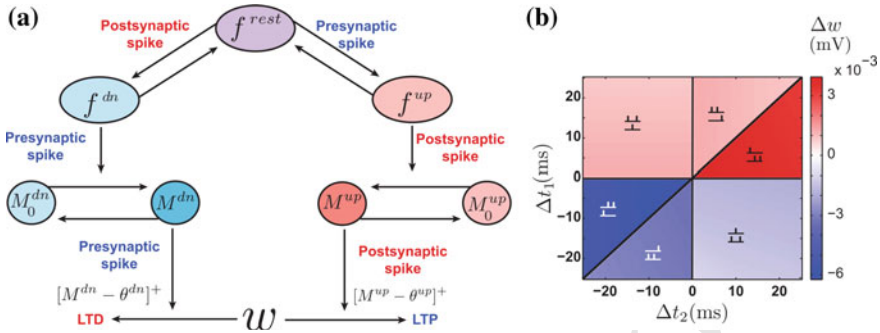
744 interactions beyond pre-post pairings. The NMDAR-based model is proposed to  
 745 have three states, rest, up and down. Every presynaptic spike moves a portion of the  
 746 NMDARs in the rest state into the up state, and every postsynaptic spike transitions  
 747 a portion of the rest-state into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

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

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

#### 768 • Other methods

769 In addition to the reviewed methods above, there are other types of STDP models  
 770 for learning such as supervised [70] and reinforcement learning [71]. However,  
 771 due to the unsupervised nature of STDP learning that is interesting for neuro-  
 772 inspired computation, we do not focus on them in this study. Pair-based STDP  
 773 models can be categorized into three classes: weight dependence, spike-pairing  
 774 scheme and delay partition. Choosing each category should be made consciously  
 775 and take into account the relevant available experimental findings. The recent  
 776 available evidences shows that both potentiation and depression are dependent on  
 777 the weight. Accordingly it is recommended to begin with very simplified models.  
 778 Moreover, we know that STDP models which assume some weight dependence  
 779 generate different behavior from the additive model. The pair-based and triplet  
 780 models are partially stable and use Hebbian competition. The Suppression and  
 781 NMDAR-based have more stability but they use anti-Hebbian competition. The  
 782 main challenge in this domain is to perform analytical and simulation studies that  
 783 are able to identify and characterize their composite effects, and investigate their  
 784 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.

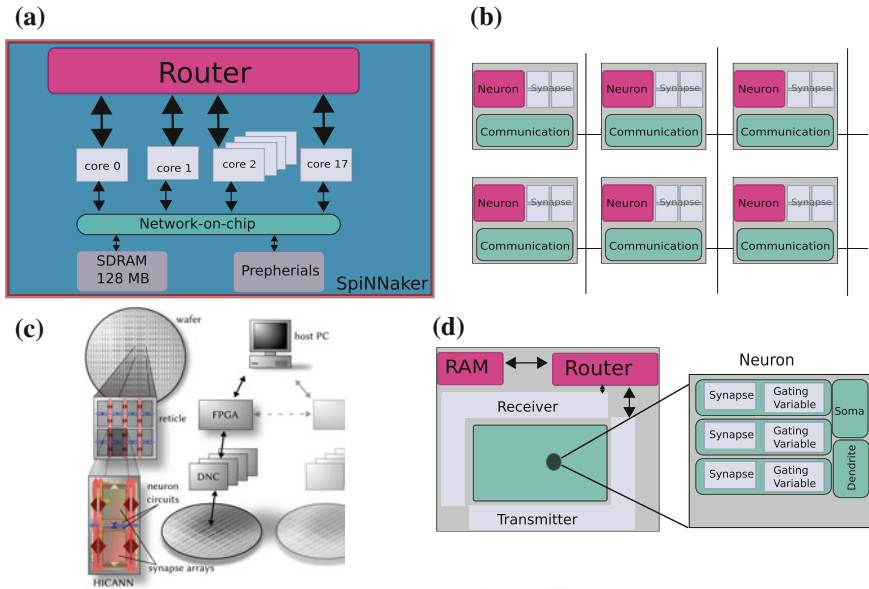
In the following, we give some details about these approaches and compare their performance.

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  
 $1024 \times 32$  bits Content Addressable Memory (CAM) and a  $1024 \times 24$  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 con-  
 ventional 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 neu-  
 ral 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,  $256 \times 256$  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 differ-  
 ent 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  $20 \text{ mW/cm}^2$ , though the traditional  
 central processing unit (CPU) is  $50\text{--}100 \text{ W/cm}^2$ . 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 accu-  
 racy. This computational neuroscience model is provided by neuro-scientists, and  
 reproduces the results obtained from numerical simulations executed on conven-  
 tional computers. The Heidelberg University BrainScales project (HICANN chip)  
 aims to produce a wafer-scale neural simulation platform, in which each 8 inch sil-  
 icon wafer integrates  $50 \times 106$  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  $256 \times 256$  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

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

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



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

881 An alternative to these architectures, that has been proposed by several authors [47–  
 882 49, 75, 76], is to use memristive devices as synapses in neuromorphic circuits. This  
 883 has the potential to lower the energy consumption by a large proportion. It has also  
 884 been showed that the memristors can emulate the STDP learning rule, and thus lead  
 885 to unsupervised learning circuits. We have thus chosen to study this kind of archi-  
 886 tecture and, in particular, to check how some parameters of the architecture or of the  
 887 devices influence the learning capabilities of the circuit.

## 888 6 Discussion

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

### 895 6.1 Homeostasis

AQ3 896 Homeostasis addresses a general principle that safeguards the stability of natural and  
 897 artificial neural systems, where stability is understood in its more classical sense of  
 898 robustness against external perturbations. Homeostasis is a fundamental concept in  
 899 neuropsychology, psychophysiology and neuroscience. Homeostasis will be defined  
 900 as negative feedback control. In physiological neural systems, the synaptic input of a  
 901 neuron is changing over time due to the external neural drive and learning results of  
 902 synaptic plasticity. From a perspective of metabolic cost, a restricted activity range  
 903 of a neuron is really meaningful.

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

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

$$911 \quad \frac{dV_{th}}{dx} = \gamma(fr_{mean} - fr_{target}) \quad (25)$$

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

## 916 6.2 Winner-Take-All

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

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

$$935 \quad b_i = \begin{cases} 1 & \text{if } x_i > x_j \text{ for all } j \neq i \\ 0 & \text{if } x_i < x_j \text{ for some } j \neq i \end{cases} \quad (26)$$

936 Therefore in the case of inputs  $x_1, \dots, x_n$  a single output  $b_i$  has values 1 that marks  
937 the position of the biggest input  $x_i$ . Wolfgang Maass [80] introduced two types of  
938 WTA namely  $k$ -WTA and *soft*-WTA. In  $k$ -WTA,  $b_i$  has value 1 if and only if  $x_i$  is  
939 among the  $k$  largest inputs. In soft-WTA the  $i_{th}$  output is an analog variable  $r_i$  whose  
940 value reflects the rank of  $x_i$  among the input variables. We use WTA in our research  
941 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

- Indiveri, G., Liu, S.C.: Proc. IEEE **103**(8), 1379 (2015). <https://doi.org/10.1109/JPROC.2015.2444094>. ArXiv: 1506.03264
- Rajendran, B., Alibart, F.: IEEE J. Emerg. Sel. Top. Circuits Syst. **6**(2), 198 (2016). <https://doi.org/10.1109/JETCAS.2016.2533298>
- Lapicque, L.: J. Physiol. Pathol. Gen. **9**, 620 (1907). <http://www.pubmedcentral.nih.gov/tocrender.fcgi?journal=484&action=archive>
- Mead, C.: Proc. IEEE **78**(10), 1629 (1990). <https://doi.org/10.1109/5.58356>
- Mahowald, M.A., Mead, C.: Sci. Am. **264**(5), 76 (1991)
- 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>
- 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>
- 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>
- 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>
- 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/>

- 985 11. Merolla, P.A., Arthur, J.V., Alvarez-Icaza, R., Cassidy, A.S., Sawada, J., Akopyan, F., Jackson,  
986 B.L., Imam, N., Guo, C., Nakamura, Y., Brezzo, B., Vo, I., Esser, S.K., Appuswamy, R., Taba,  
987 B., Amir, A., Flickner, M.D., Risk, W.P., Manohar, R., Modha, D.S.: *Science* **345**(6197), 668  
988 (2014). <https://doi.org/10.1126/science.1254642>.
- 989 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>
- 990  
991 13. Jo, S.H., Chang, T., Ebong, I., Bhadviya, B.B., Mazumder, P., Lu, W.: *Nano Lett.* **10**(4), 1297  
992 (2010). <https://doi.org/10.1021/nl904092h>
- 993 14. Shi, L., Pei, J., Deng, N., Wang, D., Deng, L., Wang, Y., Zhang, Y., Chen, F., Zhao, M., Song,  
994 S., Zeng, F., Li, G., Li, H., Ma, C.: In: 2015 IEEE International Electron Devices Meeting  
995 (IEDM), pp. 4.3.1–4.3.4 (2015)
- 996 15. O'Connor, P., Neil, D., Liu, S.C., Delbruck, T., Pfeiffer, M.: *Neuromorphic Eng.* **7**, 178 (2013).  
997 <https://doi.org/10.3389/fnins.2013.00178>
- 998 16. Prezioso, M., Merrih-Bayat, F., Hoskins, B., Adam, G., Likharev, K.K., Strukov, D.B.: *Nature*  
999 **521**(7550), 61 (2015). <https://doi.org/10.1038/nature14441>. ArXiv:1412.0611
- 1000 17. Benjamin, B.V., Gao, P., McQuinn, E., Choudhary, S., Chandrasekaran, A.R., Bussat, J.M.,  
1001 Alvarez-Icaza, R., Arthur, J.V., Merolla, P.A., Boahen, K.: *Proc. IEEE* **102**(5), 699 (2014).  
1002 <https://doi.org/10.1109/JPROC.2014.2313565>
- 1003 18. Kempter, R., Gerstner, W., van Hemmen, J.L.: *Phys. Rev. E* **59**(4), 4498 (1999). <https://doi.org/10.1103/PhysRevE.59.4498>
- 1004  
1005 19. Song, S., Miller, K.D., Abbott, L.F.: *Nat. Neurosci.* **3**(9), 919 (2000). <https://doi.org/10.1038/78829>
- 1006  
1007 20. McCulloch, W.S., Pitts, W.: *The bulletin of mathematical biophysics* **5**(4), 115 (1943). <https://doi.org/10.1007/BF02478259>
- 1008  
1009 21. VanRullen, R., Guyonneau, R., Thorpe, S.J.: *Trends Neurosci.* **28**(1), 1 (2005).  
1010 <https://doi.org/10.1016/j.tins.2004.10.010>. <http://www.sciencedirect.com/science/article/pii/S0166223604003546>
- 1011  
1012 22. Brette, R.: *Front. Syst. Neurosci.* **151** (2015). <https://doi.org/10.3389/fnsys.2015.00151>
- 1013 23. Bohte, S.M.: *Nat. Comput.* **3**(2), 195 (2004). <https://doi.org/10.1023/B:NACO.0000027755.02868.60>
- 1014  
1015 24. Ponulak, F., Kasiski, A.: *Neural Comput.* **22**(2), 467 (2010). <https://doi.org/10.1162/neco.2009.11-08-901>
- 1016  
1017 25. Masquelier, T., Hugues, E., Deco, G., Thorpe, S.: *J. Neurosci.* **29**(43), 13484 (2009). <https://doi.org/10.1523/JNEUROSCI.2207-09.2009>
- 1018  
1019 26. Chen, H.T., Ng, K.T., Bermak, A., Law, M.K., Martinez, D.: *IEEE Trans. Biomed. Circuits*  
1020 *Syst.* **5**(2), 160 (2011). <https://doi.org/10.1109/TBCAS.2010.2075928>
- 1021 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](https://doi.org/10.1016/S0893-6080(01)00083-1). <http://www.sciencedirect.com/science/article/pii/S0893608001000831>
- 1022  
1023  
1024 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>
- 1025  
1026 29. Olshausen, B.A., Field, D.J.: *Nature* **381**(6583), 607 (1996). <https://doi.org/10.1038/381607a0>.  
1027 <http://www.nature.com/nature/journal/v381/n6583/abs/381607a0.html>
- 1028 30. Haykin, S.: *Neural Networks: A Comprehensive Foundation*, 2nd edn. Prentice Hall PTR,  
1029 Upper Saddle River, NJ, USA (1998)
- 1030 31. Lipton, Z.C., Berkowitz, J., Elkan, C.: arXiv:1506.00019 [cs] (2015). ArXiv: 1506.00019
- 1031 32. Hopfield, J.J.: *Proc. Natl. Acad. Sci.* **79**(8), 2554 (1982). <http://www.pnas.org/content/79/8/2554>
- 1032  
1033 33. Elman, J.L.: *Cogn. Sci.* **14**(2), 179 (1990). <https://doi.org/10.1207/s15516709cog1402>
- 1034 34. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: *Proc. IEEE* **86**(11), 2278 (1998). <https://doi.org/10.1109/5.726791>
- 1035  
1036 35. Krizhevsky, A., Sutskever, I., Hinton, G.: **2**, 1097–1105 (2012)
- 1037 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>
- 1038

- 1039 37. Neil, D., Liu, S.C.: IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **22**(12), 2621 (2014).  
 1040 <https://doi.org/10.1109/TVLSI.2013.2294916>
- 1041 38. Hodgkin, A.L., Huxley, A.F.: J. Physiol. **117**(4), 500 (1952). <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1392413/>
- 1042 39. Chicca, E., Badoni, D., Dante, V., D'Andreagiovanni, M., Salina, G., Carota, L., Fusi, S.,  
 1043 Giudice, P.D.: IEEE Trans. Neural Netw. **14**(5), 1297 (2003). [https://doi.org/10.1109/TNN.](https://doi.org/10.1109/TNN.2003.816367)  
 1044 [2003.816367](https://doi.org/10.1109/TNN.2003.816367)
- 1045 40. Liu, S.C., Douglas, R.: IEEE Trans. Neural Netw. **15**(5), 1305 (2004). [https://doi.org/10.1109/](https://doi.org/10.1109/TNN.2004.832725)  
 1046 [TNN.2004.832725](https://doi.org/10.1109/TNN.2004.832725)
- 1047 41. Maass, W., Bishop, C.M. (eds.): Pulsed Neural Networks. MIT Press, Cambridge, MA, USA  
 1048 (1999)
- 1049 42. Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J.M., Diesmann, M.,  
 1050 Morrison, A., Goodman, P.H., Harris, F.C., Zirpe, M., Natschlger, T., Pecevski, D., Ermentrout,  
 1051 B., Djurfeldt, M., Lansner, A., Rochel, O., Vieville, T., Muller, E., Davison, A.P., El Boustani,  
 1052 S., Destexhe, A.: J. Comput. Neurosci. **23**(3), 349 (2007). [https://doi.org/10.1007/s10827-007-](https://doi.org/10.1007/s10827-007-0038-6)  
 1053 [0038-6](https://doi.org/10.1007/s10827-007-0038-6)
- 1054 43. Izhikevich, E.M.: IEEE Trans. Neural Netw./a publication of the IEEE Neural Networks Coun-  
 1055 cil **14**(6), 1569 (2003). <https://doi.org/10.1109/TNN.2003.820440>
- 1056 44. Hu, E.H., Bloomfield, S.A.: J. Neurosci.: Official J. Soc. Neurosci. **23**(17), 6768 (2003)
- 1057 45. Chua, L.: IEEE Trans. Circuit Theory **18**(5), 507 (1971). [https://doi.org/10.1109/TCT.1971.](https://doi.org/10.1109/TCT.1971.1083337)  
 1058 [1083337](https://doi.org/10.1109/TCT.1971.1083337)
- 1059 46. Strukov, D.B., Snider, G.S., Stewart, D.R., Williams, R.S.: Nature **453**(7191), 80 (2008).  
 1060 <https://doi.org/10.1038/nature06932>. [http://www.nature.com/nature/journal/v453/n7191/full/](http://www.nature.com/nature/journal/v453/n7191/full/nature06932.html)  
 1061 [nature06932.html](http://www.nature.com/nature/journal/v453/n7191/full/nature06932.html)
- 1062 47. Jo, S.H., Chang, T., Ebong, I., Bhadviya, B.B., Mazumder, P., Lu, W.: Nano Lett. **10**(4), 1297  
 1063 (2010). <https://doi.org/10.1021/nl904092h>
- 1064 48. Indiveri, G., Linares-Barranco, B., Legenstein, R., Deligeorgis, G., Prodromakis, T.: Nanotech-  
 1065 nology **24**(38), 384010 (2013). <https://doi.org/10.1088/0957-4484/24/38/384010>. ArXiv:  
 1066 [1302.7007](https://doi.org/10.1088/0957-4484/24/38/384010)
- 1067 49. Alibart, F., Pleutin, S., Gurin, D., Novembre, C., Lenfant, S., Lmimouni, K., Gamrat, C.,  
 1068 Vuillaume, D.: Adv. Funct. Mater. **20**(2), 330 (2010). <https://doi.org/10.1002/adfm.200901335>
- 1069 50. Alibart, F., Pleutin, S., Bichler, O., Gamrat, C., Serrano-Gotarredona, T., Linares-Barranco, B.,  
 1070 Vuillaume, D.: Adv. Funct. Mater. **22**(3), 609 (2012). <https://doi.org/10.1002/adfm.201101935>.  
 1071 <http://www.sciencedirect.com/science/article/pii/S1566119915000786>
- 1072 51. Desbief, S., Kyndiah, A., Gurin, D., Gentili, D., Murgia, M., Lenfant, S., Alibart, F., Cramer,  
 1073 T., Biscarini, F., Vuillaume, D.: Org. Electron. **21**, 47 (2015). [https://doi.org/10.1016/j.orgel.](https://doi.org/10.1016/j.orgel.2015.02.021)  
 1074 [2015.02.021](https://doi.org/10.1016/j.orgel.2015.02.021). <http://www.sciencedirect.com/science/article/pii/S1566119915000786>
- 1075 52. Querlioz, D., Bichler, O., Dollfus, P., Gamrat, C.: IEEE Trans. Nanotechnol. **12**(3), 288 (2013).  
 1076 <https://doi.org/10.1109/TNANO.2013.2250995>
- 1077 53. Shahsavari, M., Faisal Nadeem, M., Arash Ostadzadeh, S., Devienne, P., Boulet, P.: Phys. Status  
 1078 Solidi (c) **12**(1–2), 222 (2015). <https://doi.org/10.1002/pssc.201400069>
- 1079 54. Querlioz, D., Dollfus, P., Bichler, O., Gamrat, C.: In: 2011 IEEE/ACM International Symposi-  
 1080 um on Nanoscale Architectures, pp. 150–156 (2011). [https://doi.org/10.1109/NANOARCH.](https://doi.org/10.1109/NANOARCH.2011.5941497)  
 1081 [2011.5941497](https://doi.org/10.1109/NANOARCH.2011.5941497)
- 1082 55. Shahsavari, M., Falez, P., Boulet, P.: In: 12th ACM/IEEE International Symposium on  
 1083 Nanoscale Architectures (Nanoarch 2016), Beijing, China (2016)
- 1084 56. Drachman, D.A.: Neurology **64**(12), 2004 (2005). [https://doi.org/10.1212/01.WNL.](https://doi.org/10.1212/01.WNL.0000166914.38327.BB)  
 1085 [0000166914.38327.BB](https://doi.org/10.1212/01.WNL.0000166914.38327.BB)
- 1086 57. Morris, R.G.: Brain Res. Bull. **50**(5–6), 437 (1999)
- 1087 58. Hebb, D.: Wiley, New York (3) (1949)
- 1088 59. Morrison, A., Diesmann, M., Gerstner, W.: Biol. Cybern. **98**(6), 459 (2008). [https://doi.org/](https://doi.org/10.1007/s00422-008-0233-1)  
 1089 [10.1007/s00422-008-0233-1](https://doi.org/10.1007/s00422-008-0233-1). <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2799003/>
- 1090 60. Dayan, P., Abbott, L.F.: Theoretical Neuroscience: Computational and Mathematical Modeling  
 1091 of Neural Systems. The MIT Press (2005)
- 1092

- 1093 61. Yger, P., Gilson, M.: *Front. Comput. Neurosci.* **138** (2015). [https://doi.org/10.3389/fncom.](https://doi.org/10.3389/fncom.2015.00138)  
1094 [2015.00138](https://doi.org/10.3389/fncom.2015.00138)
- 1095 62. Zhang, L.I., Tao, H.W., Holt, C.E., Harris, W.A., Poo, M.M.: *Nature* **395**(6697), 37  
1096 (1998). <https://doi.org/10.1038/25665>. [http://www.nature.com/nature/journal/v395/n6697/](http://www.nature.com/nature/journal/v395/n6697/abs/395037a0.html)  
1097 [abs/395037a0.html](http://www.nature.com/nature/journal/v395/n6697/abs/395037a0.html)
- 1098 63. Wang, H.X., Gerkin, R.C., Nauen, D.W., Bi, G.Q.: *Nat. Neurosci.* **8**(2), 187 (2005). [https://doi.](https://doi.org/10.1038/nn1387)  
1099 [org/10.1038/nn1387](https://doi.org/10.1038/nn1387). <http://www.nature.com/neuro/journal/v8/n2/abs/nn1387.html>
- 1100 64. Babadi, B., Abbott, L.F.: *PLOS Comput. Biol.* **12**(3), e1004750 (2016). [https://doi.org/10.1371/](https://doi.org/10.1371/journal.pcbi.1004750)  
1101 [journal.pcbi.1004750](https://doi.org/10.1371/journal.pcbi.1004750). [http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.](http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004750)  
1102 [1004750](http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004750)
- 1103 65. Pfister, J.P., Gerstner, W.: *J. Neurosci.* **26**(38), 9673 (2006). [https://doi.org/10.1523/](https://doi.org/10.1523/JNEUROSCI.1425-06.2006)  
1104 [JNEUROSCI.1425-06.2006](https://doi.org/10.1523/JNEUROSCI.1425-06.2006). <http://www.jneurosci.org/content/26/38/9673>
- 1105 66. Bienenstock, E.L., Cooper, L.N., Munro, P.W.: *J. Neurosci.: Official J. Soc. Neurosci.* **2**(1), 32  
1106 (1982)
- 1107 67. Froemke, R.C., Dan, Y.: *Nature* **416**(6879), 433 (2002). [https://doi.org/10.1038/416433a.](https://doi.org/10.1038/416433a)  
1108 <http://www.nature.com/nature/journal/v416/n6879/abs/416433a.html>
- 1109 68. Sjstrm, P.J., Turrigiano, G.G., Nelson, S.B.: *J. Neurophysiol.* **92**(6), 3338 (2004). [https://doi.](https://doi.org/10.1152/jn.00376.2004)  
1110 [org/10.1152/jn.00376.2004](https://doi.org/10.1152/jn.00376.2004)
- 1111 69. Senn, W., Markram, H., Tsodyks, M.: *Neural Comput.* **13**(1), 35 (2001)
- 1112 70. Frgnac, Y., Shulz, D.E.: *J. Neurobiol.* **41**(1), 69 (1999)
- 1113 71. Pfister, J.P., Toyozumi, T., Barber, D., Gerstner, W.: *Neural Comput.* **18**(6), 1318 (2006).  
1114 <https://doi.org/10.1162/neco.2006.18.6.1318>
- 1115 72. Schemmel, J., Briiderle, D., Gribbl, A., Hock, M., Meier, K., Millner, S.: *IEEE*, pp. 1947–  
1116 1950 (2010). <https://doi.org/10.1109/ISCAS.2010.5536970>. [http://ieeexplore.ieee.org/lpdocs/](http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5536970)  
1117 [epic03/wrapper.htm?arnumber=5536970](http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5536970)
- 1118 73. Boahen, K.: *IEEE Trans. Circuits Syst. II: Analog Digit. Sig. Process.* **47**(5), 416  
1119 (2000). <https://doi.org/10.1109/82.842110>. [http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.](http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=842110)  
1120 [htm?arnumber=842110](http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=842110)
- 1121 74. Wendt, K., Ehrlich, M., Schffny, R.: In: *Proceedings of the 2nd WSEAS International Confer-*  
1122 *ence on Computer Engineering and Applications. World Scientific and Engineering Academy*  
1123 *and Society (WSEAS), Stevens Point, Wisconsin, USA, 2008, CEA'08*, pp. 189–194. [http://](http://dl.acm.org/citation.cfm?id=1373936.1373969)  
1124 [dl.acm.org/citation.cfm?id=1373936.1373969](http://dl.acm.org/citation.cfm?id=1373936.1373969)
- 1125 75. Yang, J., Pickett, M., Li, X., Ohlberg, D., Stewart, D., Williams, R.: *Nat. Nanotech-*  
1126 *mol.* **3**(7), 429 (2008). [http://www.scopus.com/inward/record.url?eid=2-s2.0-46749093701&](http://www.scopus.com/inward/record.url?eid=2-s2.0-46749093701&partnerID=40&md5=f2a7152ab8e0922c0eabc0c44f89ee7b)  
1127 [partnerID=40&md5=f2a7152ab8e0922c0eabc0c44f89ee7b](http://www.scopus.com/inward/record.url?eid=2-s2.0-46749093701&partnerID=40&md5=f2a7152ab8e0922c0eabc0c44f89ee7b)
- 1128 76. Borghetti, J., Li, Z., Straznicky, J., Li, X., Ohlberg, D.A.A., Wu, W., Stewart, D.R., Williams,  
1129 R.S.: *Proc. Natl. Acad. Sci.* **106**(6), 1699 (2009). <https://doi.org/10.1073/pnas.0806642106>
- 1130 77. Marder, E., Goaillard, J.M.: *Nat. Rev. Neurosci.* **7**(7), 563 (2006). [https://doi.org/10.1038/](https://doi.org/10.1038/nrn1949)  
1131 [nrn1949](https://doi.org/10.1038/nrn1949). <http://www.nature.com/nrn/journal/v7/n7/full/nrn1949.html>
- 1132 78. Li, C., Li, Y.: *IEEE Access* **4**, 119 (2016). <https://doi.org/10.1109/ACCESS.2015.2509005>
- 1133 79. Grossberg, S.: *Biol. Cybern.* **23**(3), 121 (1976). <https://doi.org/10.1007/BF00344744>
- 1134 80. Maass, W.: *Neural Comput.* **12**(11), 2519 (2000). [https://doi.org/10.1162/](https://doi.org/10.1162/089976600300014827)  
1135 [089976600300014827](https://doi.org/10.1162/089976600300014827)
- 1136 81. Jin, D.Z., Seung, H.S.: *Phys. Rev. E Stat. Nonlin. Soft Matter Phys.* **65**(5 Pt 1), 051922 (2002).  
1137 <https://doi.org/10.1103/PhysRevE.65.051922>
- 1138 82. Oster, M., Liu, S.C.: In: *Proceedings of the 2004 11th IEEE International Conference on*  
1139 *Electronics, Circuits and Systems, 2004. ICECS 2004*, pp. 203–206 (2004). [https://doi.org/10.](https://doi.org/10.1109/ICECS.2004.1399650)  
1140 [1109/ICECS.2004.1399650](https://doi.org/10.1109/ICECS.2004.1399650)
- 1141 83. Hafziger, P.: *IEEE Trans. Neural Netw.* **18**(2), 551 (2007). [https://doi.org/10.1109/TNN.2006.](https://doi.org/10.1109/TNN.2006.884676)  
1142 [884676](https://doi.org/10.1109/TNN.2006.884676)