A SUPERVISED STDP-BASED TRAINING ALGORITHM FOR LIVING NEURAL NETWORKS

*Yuan Zeng** *Kevin Devincentis*† *Yao Xiao*‡ *Zubayer Ibne Ferdous**
*Xiaochen Guo** *Zhiyuan Yan** *Yevgeny Berdichevsky**§

Lehigh University, *Electrical and Computer Engineering Department §Bioengineering Department
†Carnegie Mellon University, Electrical and Computer Engineering Department
‡University of Science and Technology of China, School of Gifted Young

ABSTRACT

Neural networks have shown great potential in many applications like speech recognition, drug discovery, image classification, and object detection. Neural network models are inspired by biological neural networks, but they are optimized to perform machine learning tasks on digital computers. The proposed work explores the possibilities of using living neural networks *in vitro* as the basic computational elements for machine learning applications. A supervised STDP-based learning algorithm is proposed in this work, which considers neuron engineering constrains. A 75% accuracy is achieved on the MNIST benchmark for handwritten digit recognition.

***Index Terms*—** Spiking neural network, Spike timing dependent plasticity, Supervised learning, Biological neural network

1. INTRODUCTION AND MOTIVATION

Artificial Neural Network (ANN) and Spiking Neural Network (SNN) are two brain inspired computational models, which have shown promising capabilities for solving problems such as face detection [1] and image classification [2]. Computer programs based on neural networks can defeat professional players in the board game Go [3] [4]. ANN relies on numerical abstractions to represent both the inputs and the connections among neurons, whereas SNN uses spike trains to represent inputs and outputs, which mimics computations performed by neurons and synapses [5]. Both ANN and SNN are models extracted from biological neuron behaviors and optimized to perform machine learning tasks on digital computers. This work explores whether biological living neurons *in vitro* can be directly used as the fundamental computational elements to perform machine learning tasks.

While precise control of living neural networks is challenging, recent advancement on optogenetics, genetically encoded neural activity indicators, and cell-level micropatterning open up interesting possibilities in this area [6] [7] [8]. Optogenetics can label individual neurons with different types of optically controlled channels, which equip *in vitro* neural networks with optical interfaces. Patterned optical stimulation and high-speed optical detection allow simultaneous accesses to thousands of *in vitro* neurons. In addition, the invention of micropatterned [9] creates a unique opportunity to enable modularized system design.

This work is supported in part by a CORE grant from Lehigh University.

Living neurons can perform "computation" naturally to transfer spike information through synapses. The energy consumption is 100000× more efficient than hardware evaluation [10]. Living neuron has small size (4 to 100 micrometers diameter) [11], and it can adapt to changes.

This work explores the possibility of using living neurons for machine learning applications through simulation. By considering neuron engineering design constrains, a new learning algorithm is proposed for easy training in future biological experiments. A fully connected spiking neural network is simulated using on NEURON simulator [12].

2. METHODS

Spiking neural network is a model that closely represents biological neurons. In SNN, neurons are connected through plastic synapse, and information is represented as a series of spikes. Spikes are generated when the membrane voltage of a neuron exceeds a certain threshold.

This work models biological neural networks using the Hodgkin-Huxley (HH) neuron mode [13] and spike-based data representation. One major difference between the proposed work and prior SNN based models [18] [19] [20] [21] [22] [30] is that this work aims to explore the potential of using biological living neurons as the functional devices, while prior works focused on the computational capability of the neuron model. The use of biologically accurate neuron model leads to different design choice for input encoding, network topology, neuron model, learning rule, and model parameters.

2.1. Network topology

Neural connectivity in human brain is complex and has different types of topologies in different parts of the nervous system. To understand the network functionality, a simple network topology is built, where all of the input neurons are connected to all of the output neurons through synapses as shown in Fig. 1.

Due to bioengineering constrains, original images from MNIST dataset, which have 28×28 pixels, are compressed to 14×14 pixels. As a result of this simplification, the network has 196 inputs, each corresponding to one input pixel. Only black pixels generate spikes, and all input spikes occurs simultaneously.

Output for the network is a vector of spiking state for output neurons, "1" represents a spike, "0" means no spike. Each output neuron is associated with a group index 0 to 9. Group index can be defined artificially. The group that has the largest number of spiking neurons will be considered as the network output. In our experiment, 300 output neurons are used. Every 30 consecutive neurons belong to one group, for example, the first 30 belong to group 0.

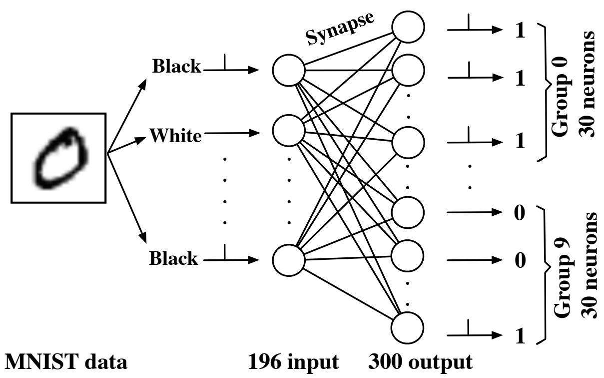

Fig. 1. Network topology.

2.2. Neuron model

To capture the realistic neuron dynamics, Hodgkin-Huxley (HH) model [13] is used in the simulation, which has been successfully verified by numerous biological experimental data. It models the electro-chemical information transmission of biological neurons with capacitors and resistors based electrical circuit. Membrane potential is given by Eq. (1), where g_{Na}, g_K, and g_L represent the conductance for sodium, potassium and leakage channels respectively. E_{Na}, E_K, and E_L denote the corresponding equilibrium potentials. m, h, and n are gating variables [13]. $I(t)$ is the input current injected into the cell.

$$C\frac{d_u}{d_t} = -g_{Na}m^3h(u - E_Na) - g_K n^4(u - E_k) - g_L(u - E_L) + I(t) \quad (1)$$

Spike will be generated if the membrane potential for a neuron exceeds a certain threshold. However, after a strong current pulse excites a spike, there will be a period that current pulse at the same amplitude cannot generate another spike, which is referred to as the refractory period [14].

2.3. Learning rule

Plasticity of synapses between neurons is important for learning. Connection strength changes based on precise timing between pre- and post-synaptic spikes. This phenomenon is called Spike Timing Dependent Plasticity (STDP) [15].

Eqs. (2)-(5) describe the STDP model [16] used in this work. In this model, the amount of weight changes is proportional to spike trace [17]. t is the time a spike arrives; τ_{LTD} and τ_{LTP} are time constants; $ALTD$ and $ALDP$ are respectively the amplitudes of trace updating for potentiation and depression; and $aLTP$ and $aLTD$ are respectively the potentiation and depression learning rate. Each pre-synaptic spike arrival will update the pre-synaptic trace P according to Eq. (2), post-synaptic spike changes the post-synaptic trace Q according to Eq. (3). If a pre-synaptic spike happens after a post-synaptic spike, weight will decrease δ_{Wq} based on (4). If a post-synaptic spike happens after a pre-synaptic spike, weight will increase δ_{Wp} based on (5).

$$P = P \times exp(\frac{t_{pre} - t}{\tau_{LTP}}) + ALTP \quad (2)$$

$$Q = Q \times exp(\frac{t_{post} - t}{\tau_{LTD}}) + ALTD \quad (3)$$

$$\delta_{Wq} = aLTD \times Q \times exp(\frac{t_{post} - t}{\tau_{LTD}}) \quad (4)$$

$$\delta_{Wp} = aLTP \times P \times exp(-\frac{t - t_{pre}}{\tau_{LTP}}) \quad (5)$$

STDP rule itself is not enough for learning. Both unsupervised [18] [19] [20] and supervised [21] [22] algorithm based on STDP rule have been proposed, which focus on computational aspect of the neuron model. These algorithms use Poisson based spike trains as input and include other bio-inspired mechanism like winner-take-all [23] and homoeostasis [24].

The artificial stimuli can be precisely applied to optically stimulated neurons, therefore, deterministic inputs are used in the model. Considering a feedforward network, the deterministic input will lead to a problem. There will be no weight decrease for the network based on the basic STDP rule and all of the neurons will fire eventually. That is because all of the input neurons fire at the same time. Therefore, post-synaptic neuron never fires before any pre-synaptic spike.

In order to solve this problem and make the living neural network easy to train, we propose a new supervised STDP algorithm. Basic operations in this algorithm are shown in Fig. 2, and the pseudo code is presented in Algorithm 1.

In this scheme, stimuli can be applied to both input and output neurons artificially to generate a spike. In Fig. 2, external stimuli that generate a spike for input neurons are shown in pink and external stimuli that generate a spike for output neurons are shown in blue.

Without external stimuli, an output neuron can reach its action potential and fire. These spikes of output neurons generated as nature responds to the network inputs are shown in yellow. Because the output neuron spikes after the input stimulus, weights between those in-out pairs will be naturally potentiated. This is referred to as the network's "natural increase", with interval between t_{pre} and t_{post} equal to T4-T2.

Besides this natural increase, stimuli can be directly applied to the input and output neuron pairs to artificially change the weights. If the output stimulus is given at T3 and the input stimulus is given at T2, weight between this pair will be increased, which is referred to as the "artificial increase". If the output stimulus is given at T1, which is before an input stimulus, "artificial decrease" will happen and weight will decrease.

In this work, these timing-based rules of applying external stimuli are the essential mechanisms to change synaptic weights. For the digit recognition task evaluated work,

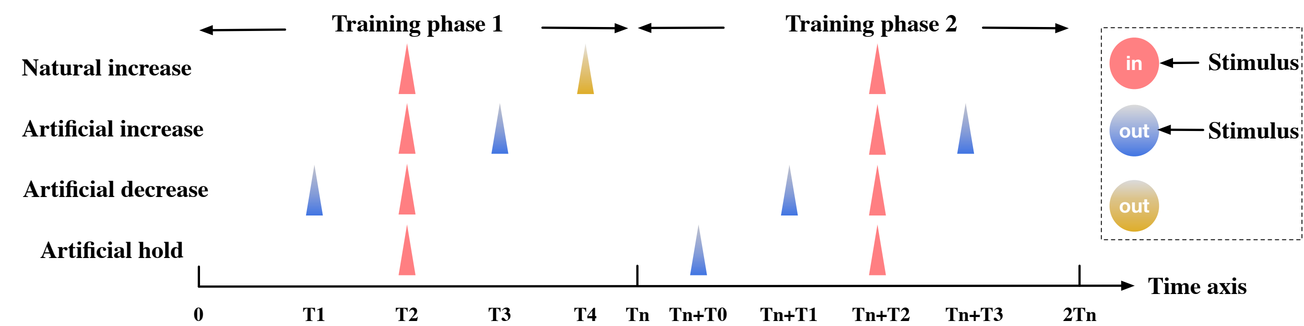

Fig. 2. Four mechanisms in the supervised STDP training.

groups of weights are increased or decreased to make the network converge. Some weights that are already reaching convergence need to be kept same. However, if the weight of synapse is large enough to make the output neuron fire, the weight will increase naturally during the training process, which will move the network away from convergence. Therefore, the training process is separated into two phases, and the input stimuli corresponding to the input image are given once for each phase. During the first phase, if a weight that should be kept the same actually increased, a stimulus will be added to the corresponding output at Tn+T0, which is before the input stimulus at Tn+T2 for the second phase to decrease the increased weight. Because of the refractory period, there will not be a natural increase at the second phase. The time interval between input and the stimuli (T2-T0) is adjusted with the time interval between natural output spike and input stimuli (T4-T2) at the first phase, so that the amount of decrease and increase are same. Through this approach, the weight can be held the same. This process is referred to as the "artificial hold".

For each new image observed by the network, a prediction is made by applying the input stimuli to the network and checking the natural response of the outputs (Algorithm 1 lines 2-4). The index of the group that contain the largest number of spiking neurons is the predicted result. To train the network, correct label (*ID*) for the input image and the actual spike pattern for output neurons are compared to generate the control signals to select neurons into different lists that require external stimuli (Algorithm 1 lines 5-12). Based on the selected neurons, stimuli are applied to the network to update weights (Algorithm 1 lines 13-20).

Three tunable parameters can be set for the training process. *trainStep* is the number of times the network get trained for one image. Larger trainStep means higher effective learning rate. *inTarget* represents the minimum number of firing neurons must be observed in the output group that matches the correct label. *deNumber* represents the maximum number of firing neurons to be decreased in the incorrect groups. *numSpike[id]* is the number of spiking neuron in group *id*. When *id* matches *ID* and the number of firing neurons is less than the *inTarget*, *inTarget-numSpike[id]* number of neurons will be randomly chosen among non-spiking ones and added to the *increaseList* (Algorithm 1 lines 6-8). Spiking neurons in this group will be added to the *holdList* to keep the weight the same since they respond correctly (Algorithm 1 line 9). For other output groups, if more than *deNumber* of output neurons are firing, *deNumber* neurons will be randomly chosen among the firing ones and be added to the *decreaseList*. If the number of firing neurons is less than *deNumber*, all the firing neurons will be added to the *decreaseList* (Algorithm 1 lines 10-12). After selecting the *increaseList*, *decreaseList*, and *holdcreaseList*, corresponding stimuli are applied in time sequence shown in Fig. 2 for each training step. (Algorithm 1 lines 13-20)

Algorithm 1 Supervised STDP Training

1: // **Tunable parameters**: trainStep, inTarget, deNumber
2: **for** each image **do**
3: Apply stimuli to input neurons base on pixel values
4: Record spike pattern for output neurons
5: **for** each output group (id = 0 to 9) **do**
6: **if** $id{=}{=}ID$ **and** $numSpike[id]{<}inTarget$ **then**
7: $x = inTarget - numSpike[id]$
8: Add x non-spiking neurons to $increaseList$
9: Add all spiking neurons to $holdList$
10: **if** $id \neq ID$ **then**
11: $y = min(numSpike[id], deNumber)$
12: Add y spiking neurons to $decreaseList$
13: **for** each trainStep **do**
14: Apply stimuli to the $decreaseList$ at T1
15: Apply stimuli to input neurons at T2
16: Apply stimuli to the $increaseList$ at T3
17: Apply stimuli to the $holdList$ at T4
18: Apply stimuli to the $decreaseList$ at Tn+T1
19: Apply stimuli to input neurons at Tn+T2
20: Apply stimuli to the $increaseList$ at Tn+T3

3. SIMULATION

3.1. Parameters

Parameters used for the network are listed in Table 1.

Three tunable parameters for the network are: *trainStep*, *inTarget* and *deNumber*. In the fully connected network, 300 output neurons are separated into 10 groups with 30 neurons each. *trainStep* kept at 1, *inTarget* is set at 20, and *deNumber* configured as 20 when they are not the variable parameters. Timing parameters in Fig. 2 also have an influence on the learning rate for each *trainStep*. Small time interval between pre- and post-synaptic neuron spikes (*e.g.*, T3-T2) leads to

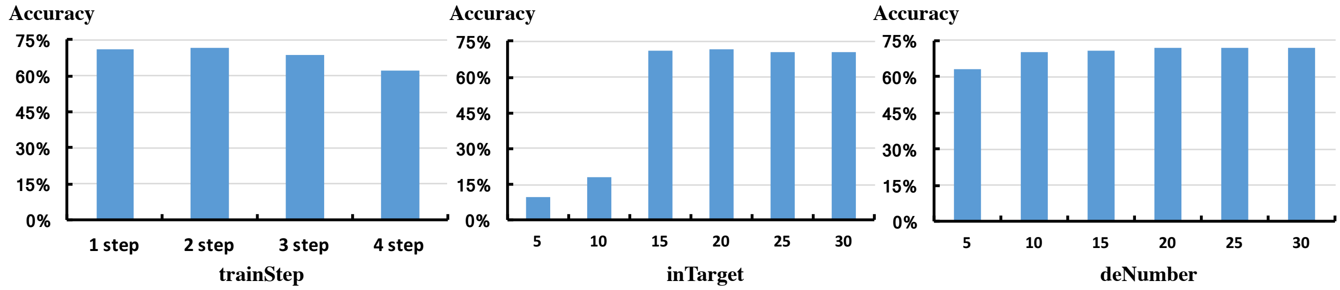

Fig. 3. Sensitivity of the image recognition accuracy to different parameters.

greater weight changes. However, the optical stimuli can not be spaced arbitrarily close to each other. In this work, a fixed 5 ms interval is used for pre- and post-synaptic spike for artificial increase and decrease. The pre- and post-synaptic spike interval for the hold mechanism is set to be same as the natural increase interval, which is 10 ms according to empirical data. A sensitivity study is done by tuning one parameter at a time. For this sensitivity study, 1000 images from the MNIST dataset are used for training, another 1000 images are used for testing. Simulation results are shown in Fig. 3.

Parameter	Value
τ_{LTP} / τ_{LTD}	20ms / 20ms
ALTP / ALTD	1 / -1
aLTP / aLTD	6×10^{-5} / 6.3×10^{-5}
max weight	0.02
refractory period	25ms
T3-T2 / T2-T1	5ms / 5ms
T4-T2 / T2-T0	10ms / 10ms
Tn	25ms

Table 1. Simulation parameters.[25][26][27][18]

3.2. Results and analysis

When *trainStep* increases from 1 to 2, prediction accuracy increases. However, when learning rate is too large (beyond 3 steps), the accuracy drops. This is because a larger effective learning rate may lead to fast convergence, but if the step is too large, overshooting will happen when moving towards the global optimum point in the error surface, which leads to oscillations and hurts the performance.

For *inTarget* and *deNumber*, a better performance can be achieved if both values are more than half of the number of the neurons in a group. For increase, the best performance is achieved at *inTarget*=20, which shows that, training half of the neurons to fire can provide enough information, while the untrained neuron can provide fault tolerance.

Decrease the number of firing neurons by 20 in each of the incorrect groups achieves the best performance. The accuracy exhibits noticeable degradation when deNumber is below 10. This is because the inTarget is 20 for this set of results. The number of firing neurons in the incorrect group needs to be below 20 to make sure that the correct group has the greatest number of firing neurons. Best accuracy result for the sensitivity study is 72%.

A much larger dataset (10000 images from MNIST) is tested based on the best parameters: trainStep=2, inTarget=20 and deNumber=20. Accuracy can be improved to 75%.

Compared to a single-layer fully connected ANN, which achieves 88% [28] accuracy on MNIST dataset, the proposed supervised STDP-based SNN still has an accuracy gap. Unlike the ANN, where weights are visible values that could be used in the prediction, synaptic weights are invisible at the outputs of an SNN. Predictions are based on the spikes in an SNN system, which depend on the firing threshold and the absolute accumulated inputs through corresponding synapses. The loss of information leads to the accuracy drops. The only single-layer SNN work in our knowledge [29] derives an extra mathematical function to extract more informations through the timing relationships for output spikes, which is an unrealistic scheme for living neuron experiment. This work [29] does not considering biological properties of neurons and synapses.

For SNN works based on neuron science simulations, a three layer design with supervised STDP achieved an accuracy of 75.93% for 10 digit recognition task on MNIST dataset [22]. Which is similar with the proposed single-layer network. There are other SNN works that have better results on MNIST [18] [19] [20] [21] [30]. However, those networks have at least two layers with a larger number of neurons. Some work also preprocess the input images to achieve better accuracy [21]. The major difference between the proposed work and prior work is that prior work is optimized for solid state computers. For the proposed supervised scheme, bioengineering constrains are considered. Input data are compressed and applied as deterministic spike trains. The biological limitations on maintaining the synaptic weights lead to the design of two training phases and multiple training steps.

75% accuracy is obtained on biologically-plausible SNN model, which is a promising result that demonstrates the feasibility of using living neuron network to compute.

4. CONCLUSION

To explore the possibility of using living neuron for machine learning task, a new supervised STDP training algorithm has been proposed and simulated on a fully-connected neural networks based on HH model. A 75% accuracy is achieved on digit recognition task for the MNIST dataset. This result demonstrates the feasibility of using living neurons as the fundamental computation elements for machine learning tasks.

5. REFERENCES

[1] Haoxiang Li, Zhe Lin, Xiaohui Shen, Jonathan Brandt, and Gang Hua, "A convolutional neural network cascade for face detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5325–5334.

[2] Taras Iakymchuk, Alfredo Rosado-Muñoz, Juan F Guerrero-Martínez, Manuel Bataller-Mompeán, and Jose V Francés-Víllora, "Simplified spiking neural network architecture and stdp learning algorithm applied to image classification," *EURASIP Journal on Image and Video Processing*, vol. 2015, no. 1, pp. 4, 2015.

[3] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[4] Karen Simonyan Ioannis Antonoglou Aja Huang Arthur Guez Thomas Hubert Lucas Baker Matthew Lai Adrian Bolton Yutian Chen Timothy Lillicrap Fan Hui Laurent Sifre George van den Driessche Thore Graepel Demis Hassabis David Silver, Julian Schrittwieser, "Mastering the game of go without human knowledge," *Nature*, 2017.

[5] Zidong Du, Daniel D. Ben-Dayan Rubin, Yunji Chen, Liqiang He, Tianshi Chen, Lei Zhang, Chengyong Wu, and Olivier Temam, "Neuromorphic accelerators: A comparison between neuroscience and machine-learning approaches," pp. 494–507, 2015.

[6] Yevgeny Berdichevsky, Helen Sabolek, John B Levine, Kevin J Staley, and Martin L Yarmush, "Microfluidics and multielectrode array-compatible organotypic slice culture method," *Journal of neuroscience methods*, vol. 178, no. 1, pp. 59–64, 2009.

[7] Yevgeny Berdichevsky, Kevin J Staley, and Martin L Yarmush, "Building and manipulating neural pathways with microfluidics," *Lab on a Chip*, vol. 10, no. 8, pp. 999–1004, 2010.

[8] Erkin Seker, Yevgeny Berdichevsky, Matthew R Begley, Michael L Reed, Kevin J Staley, and Martin L Yarmush, "The fabrication of low-impedance nanoporous gold multiple-electrode arrays for neuralelectrophysiology studies," *Nanotechnology*, vol. 21, no. 12, pp. 125504, 2010.

[9] Md. Fayad Hasan and Yevgeny Berdichevsky, "Neural circuits on a chip," *Micromachines*, vol. 7, no. 9, 2016.

[10] P. King, "How is the human brain so energy-efficient?," *https://www.quora.com/How-is-the-human-brain-so-energy-efficient*, 2012.

[11] Melissa Davies, "The neuron: Size comparison," *Neuroscience: A journey through the brain.*, 2002.

[12] Gordon M Shepherd, Jason S Mirsky, Matthew D Healy, Michael S Singer, Emmanouil Skoufos, Michael S Hines, Prakash M Nadkarni, and Perry L Miller, "The human brain project: neuroinformatics tools for integrating, searching and modeling multidisciplinary neuroscience data," *Trends in neurosciences*, vol. 21, no. 11, pp. 460–468, 1998.

[13] Alan L Hodgkin and Andrew F Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, no. 4, pp. 500–544, 1952.

[14] David E Meyer and David E Kieras, "A computational theory of executive cognitive processes and multiple-task performance: Part 2. accounts of psychological refractory-period phenomena.," *Psychological review*, vol. 104, no. 4, pp. 749, 1997.

[15] Natalia Caporale and Yang Dan, "Spike timing–dependent plasticity: a hebbian learning rule," *Annu. Rev. Neurosci.*, vol. 31, pp. 25–46, 2008.

[16] Andrew P. Davison, "Modelling stdp in the neuron simulator," *http://andrewdavison.info/notes/modelling-stdp-neuron-simulator/*, 2007.

[17] Jesper Sjöström and Wulfram Gerstner, "Spike-timing dependent plasticity," *Spike-timing dependent plasticity*, vol. 35, 2010.

[18] Peter U Diehl and Matthew Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in computational neuroscience*, vol. 9, 2015.

[19] Jason M Allred and Kaushik Roy, "Unsupervised incremental stdp learning using forced firing of dormant or idle neurons," in *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE, 2016, pp. 2492–2499.

[20] Damien Querlioz, Olivier Bichler, Philippe Dollfus, and Christian Gamrat, "Immunity to device variations in a spiking neural network with memristive nanodevices," *IEEE Transactions on Nanotechnology*, vol. 12, no. 3, pp. 288–295, 2013.

[21] Joseph M Brader, Walter Senn, and Stefano Fusi, "Learning real-world stimuli in a neural network with spike-driven synaptic dynamics," *Neural computation*, vol. 19, no. 11, pp. 2881–2912, 2007.

[22] Amirhossein Tavanaei and Anthony S Maida, "A minimal spiking neural network to rapidly train and classify handwritten digits in binary and 10-digit tasks," *International Journal of Advanced Research in Artificial Intelligence*, vol. 4, no. 7, pp. 1–8, 2015.

[23] Bernhard Nessler, Michael Pfeiffer, and Wolfgang Maass, "Stdp enables spiking neurons to detect hidden causes of their inputs," in *Advances in neural information processing systems*, 2009, pp. 1357–1365.

[24] Mark CW Van Rossum, Guo Qiang Bi, and Gina G Turrigiano, "Stable hebbian learning from spike timing-dependent plasticity," *Journal of neuroscience*, vol. 20, no. 23, pp. 8812–8821, 2000.

[25] Sen Song, Kenneth D Miller, and Larry F Abbott, "Competitive hebbian learning through spike-timing-dependent synaptic plasticity," *Nature neuroscience*, vol. 3, no. 9, pp. 919–926, 2000.

[26] Rebecca Lewis, Katie E Asplin, Gareth Bruce, Caroline Dart, Ali Mobasheri, and Richard Barrett-Jolley, "The role of the membrane potential in chondrocyte volume regulation," *Journal of cellular physiology*, vol. 226, no. 11, pp. 2979–2986, 2011.

[27] Hélene Paugam-Moisy and Sander Bohte, "Computing with spiking neuron networks," in *Handbook of natural computing*, pp. 335–376. Springer, 2012.

[28] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[29] Wenyu Yang, Jie Yang, and Wei Wu, "A modified one-layer spiking neural network involves derivative of the state function at firing time," *Advances in Neural Networks–ISNN 2012*, pp. 149–158, 2012.

[30] Michael Beyeler, Nikil D Dutt, and Jeffrey L Krichmar, "Categorization and decision-making in a neurobiologically plausible spiking network using a stdp-like learning rule," *Neural Networks*, vol. 48, pp. 109–124, 2013.