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 possibility of using living neural networks *in vitro* as the basic computational elements for machine learning applications. A new supervised STDP-based learning algorithm is proposed in this work, which considers neuron engineering constraints. A 74.7% 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 states of the neurons and the connections among them, whereas SNN uses spike trains to represent inputs and outputs and 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.

In contrast, the proposed work explores whether biological living neurons *in vitro* can be directly used as basic computational elements to perform machine learning tasks. Living neurons can perform "computation" naturally by transferring spike information through synapses. The energy consumption is $100000\times$ more efficient than hardware evaluation [6]. Living neurons have small sizes (4 to 100 micrometers diameter) [7], and adapt to changes.

While precise control of living neural networks is challenging, recent advances in optogenetics, genetically encoded neural activity indicators, and cell-level micropatterning open up possibilities in this area [8] [9] [10]. 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 access to thousands of *in vitro* neurons. In addition, the invention of micropattern [11] enables modularized system design.

To the best of our knowledge, this work is the first to explore the possibility of using living neurons for machine learning applications. By considering neuron engineering design constraints, a new algorithm is proposed for easy training in future biological experiments. A fully connected spiking neural network is evaluated on MNIST dataset using NEURON simulator [12]. A 74.7% accuracy is obtained based on a biologically-plausible SNN model, which is a promising result that demonstrates the feasibility of using living neuron networks to compute.

2. METHODS

Spiking neural network is a model that closely represents biological neuron behavious. In biological neural network, neurons are connected through plastic synapses. A spike of a pre-synaptic neuron will change the membrane potential of a post-synaptic neuron. The impact of the spikes at different time from all of the pre-synaptic neurons will be accumulated at the post-synaptic neuron. Post synaptic spikes are generated when the membrane voltage of a neuron exceeds a certain threshold. In SNN, information is represented as a series of spikes, and the accumulative effect of the pre-synaptic spikes are also modeled.

This work models biological neural networks using the Hodgkin-Huxley (HH) neuron model [13] and spike-based data representation. One major difference between the proposed work and prior SNN based models [14] [15] [16] [17] [18] [19] 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. Neuron engineering design constraints lead 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 input neurons are connected to all output neurons through synapses (Fig. 1).

Due to bioengineering constraints, images from the MNIST dataset, which have 28×28 pixels, are compressed to 14×14 pixels. As a result of this simplification, the network has 196 input neurons, each corresponding to one pixel. Only black pixels generate spikes, and all input spikes occur 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 from 0 to 9, which can be defined artificially. The index of the group with 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.
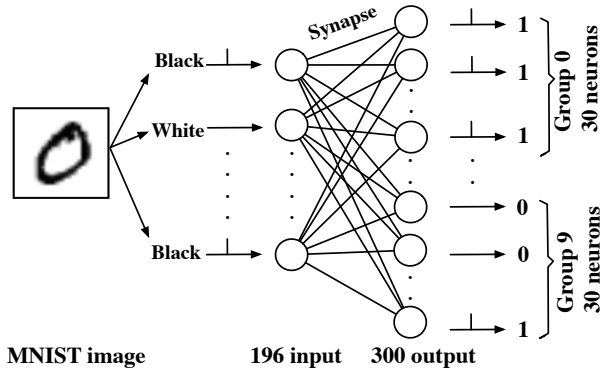

Fig. 1. Network topology.

2.2. Neuron model

To capture the realistic neuron dynamics, the Hodgkin-Huxley (HH) model [13] is used in the simulation, which models the electro-chemical information transmission of biological neurons with electrical circuit. HH model has been successfully verified by numerous biological experimental data and it is more biologically accurate than other simplified model such as Integrate and Fire model [20].

A 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 during which current pulse at the same amplitude cannot generate another spike, which is referred to as the refractory period [21].

2.3. Learning rule

The 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) [22].

Eqs. (1)-(4) describe the STDP rule [23] used in this work. In this rule, the weight changes are proportional to spike trace [24]. t_{pre} and t_{post} denote the time a pre- and post-synaptic, respectively, spikes arrive; $t_{pre}\prime$ and $t_{post}\prime$ represent the arrival time of previous pre- and post-synaptic spikes respectively. $ALTP$ and $ALTD$ are the amplitudes of trace updating for potentiation and depression respectively; and $aLTP$ and $aLTD$ are the potentiation and depression

learning rate respectively. Each pre-synaptic spike arrival will update the pre-synaptic trace P according to Eq. (1), post-synaptic spike changes the post-synaptic trace Q according to Eq. (2). If a pre-synaptic spike happens after a post-synaptic spike, weight will decrease by δ_{Wq}, which is given by (3). If a post-synaptic spike occurs after a pre-synaptic spike, weight will increase by δ_{Wp} given by (4).

$$P = P \times exp\left(\frac{t_{pre}\prime - t_{pre}}{\tau_{LTP}}\right) + ALTP \tag{1}$$

$$Q = Q \times exp\left(\frac{t_{post}\prime - t_{post}}{\tau_{LTD}}\right) + ALTD \tag{2}$$

$$\delta_{Wq} = aLTD \times Q \times exp\left(\frac{t_{post} - t_{pre}}{\tau_{LTD}}\right), t_{post} < t_{pre} \tag{3}$$

$$\delta_{Wp} = aLTP \times P \times exp\left(\frac{t_{pre} - t_{post}}{\tau_{LTP}}\right), t_{pre} < t_{post} \tag{4}$$

The STDP rule itself is not enough for learning. Both unsupervised [14] [15] [16] and supervised [17] [18] training algorithms based on the 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 mechanisms like winner-take-all [25] and homoeostasis [26].

Since artificial stimuli can be precisely applied to optically stimulated neurons, synchronous inputs are used for the proposed work. Considering a feedforward network, the synchronous inputs 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 when all of the input neurons fire at the same time, no post-synaptic neuron will fire 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 training algorithm. Four basic operations in this algorithm are shown in Fig. 2 and discussed below.

In this algorithm, 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 and output neurons are shown in pink and blue respectively.

Without artificial output stimuli, an output neuron can reach its action potential and fire in response to the network inputs, which is shown in yellow. Because the output neuron spikes after the input stimuli, weights between those in-out pairs will be naturally potentiated. This is referred to as the network's "natural increase", with $t_{pre} - t_{post} = T2 - T4$.

Besides this natural increase, stimuli can be directly applied to the input and output neuron pairs to artificially change the weights. If an output stimulus is given at T3 and an input stimulus is given at T2, weight between this pair will be increased, which is referred to as "artificial increase". If an 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, groups of weights are
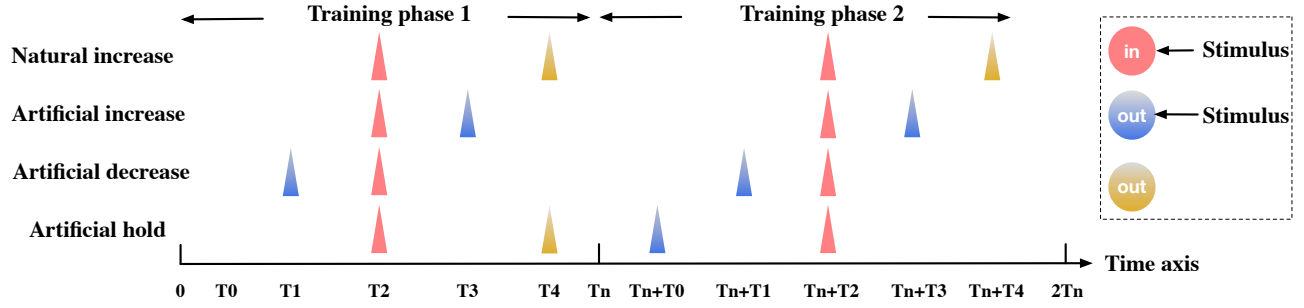
Fig. 2. Four mechanisms in the supervised STDP training.

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 a 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, each training step 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 neurons 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 matches. Through this approach, the weight can be kept roughly the same. This process is referred to as "artificial hold".

The new STDP training algorithm is described in Algorithm 1. 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 3-5). The index of the group that contain the largest number of spiking neurons is the predicted result (if two groups have the same number of spikes, the smaller group index is chosen). To train the network, the 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 (lines 7-14). Based on the selected neurons, stimuli are applied to the network to update weights (lines 16-22).

Three tunable parameters can be set for the training process. *trainStep* is the number of training steps the network goes through for one image. A larger trainStep means higher effective learning rate. In order for the correct group to have more firing neurons than the other groups, *inTarget* and *deTarget* are targets of the number of firing neurons in each group. *inTarget* represents the desired number of firing neurons to be observed in the output group that matches the correct label. *deTarget* represents the desired number of firing neurons to be observed in the incorrect groups. *numSpike[id]* is the number of spiking neurons in group *id*. When *id* matches *ID*, all spiking neurons in this group will be added to the *holdList* to keep their weights since they respond correctly

(line 8). If the number of firing neurons is less than *inTarget*, *inTarget-numSpike[id]* neurons will be randomly chosen among the non-spiking ones and added to the *inList* (lines 9-11). For other output groups, if the number of firing neurons is more than *deTarget*, *numSpike[id]-deTarget* neurons will be randomly chosen among the spiking ones and added to the *deList* (lines 12-14). After selecting *inList*, *deList*, and *holdList*, corresponding stimuli are applied in time sequence shown in Fig. 2 for each training step (lines 16-22).

Algorithm 1 Supervised STDP Training

1: // **Tunable parameters**: trainStep, inTarget, deTarget
2: **for** each image **do**
3:     Clear $inList$, $deList$ and $holdList$
4:     Apply stimuli to input neurons based on pixel values
5:     Record spike pattern for output neurons
6:     **for** each output group (id = 0 to 9) **do**
7:         **if** $id==ID$ **then**
8:             Add all spiking neurons to $holdList$
9:             **if** $numSpike[id]<inTarget$ **then**
10:                 $x = inTarget - numSpike[id]$
11:                 Add x non-spiking neurons to $inList$
12:         **if** $id \neq ID$ **and** $numSpike[id]>deTarget$ **then**
13:             $y = numSpike[id] - deTarget$
14:             Add y spiking neurons to $deList$
15:     **for** each trainStep **do**
16:         Apply stimuli to the $deList$ at T1
17:         Apply stimuli to input neurons at T2
18:         Apply stimuli to the $inList$ at T3
19:         Apply stimuli to the $holdList$ at Tn+T0
20:         Apply stimuli to the $deList$ at Tn+T1
21:         Apply stimuli to input neurons at Tn+T2
22:         Apply stimuli to the $inList$ at Tn+T3

---

3. SIMULATION

3.1. Parameters

Parameters used for the network are listed in Table 1. Timing parameters in Fig. 2 impact the learning rate for each *trainStep*. Small time interval between pre- and post-synaptic neuron spikes (*e.g.*, T3-T2) leads to greater weight changes. However, the optical stimuli cannot be spaced arbitrarily close to each other. In this work, a fixed 5 ms interval is used between pre- and post-synaptic spikes for artificial increase and decrease. The pre- and post-synaptic spike interval for the
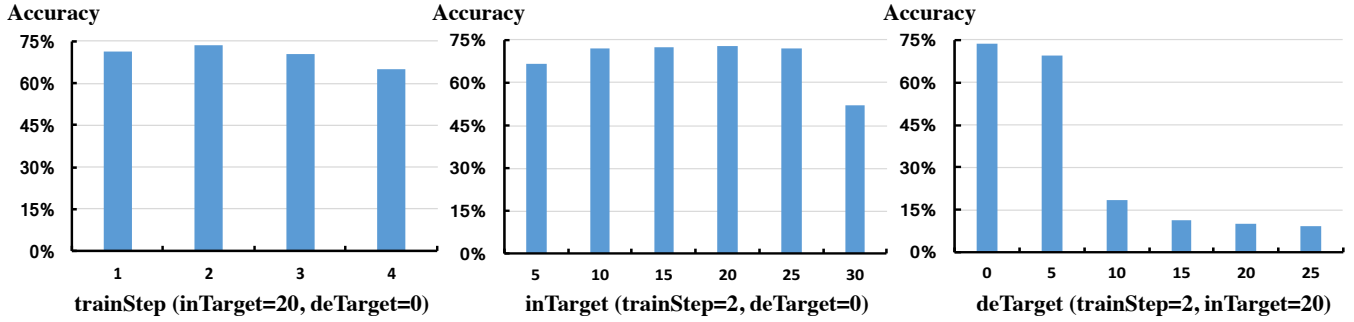
Fig. 3. Sensitivity of the image recognition accuracy to different parameters.

hold mechanism matches the natural increase interval, which is 10 ms according to empirical data.

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	30ms

Table 1. Simulation parameters, [27][28][29][14].

3.2. Results and analysis

Three tunable parameters for the network are: *trainStep*, *inTarget* and *deTarget*. *trainStep* is kept at 2, *inTarget* is set at 20, and *deTarget* is configured as 0 as the base line. 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. Training time for each trainStep is 60 ms. Simulation results are shown in Fig. 3.

When *trainStep* increases from 1 to 2, the 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 it is too large, overshooting will happen when moving towards the global optimum point, which leads to oscillations and hurts the performance.

For *inTarget*, better results are achieved in the middle range, which shows that, training nearly half of the neurons to fire can provide enough information while avoid divergence. For example, if two images have a large set of overlapping firing neurons in the inputs but have different labels, strengthening the connections between all of the firing inputs and the corresponding outputs for one image will likely lead to mis-prediction for the other image. The best performance is achieved at *inTarget*=20.

Decreasing weights associated with all firing neurons in all incorrect groups (*deTarget*=0) achieves the best performance. When *deTarget* is larger than 10, performance drops dramatically. This is because 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. However, training 2 steps cannot guarantee both *inTarget* and *deTarget* are reached. The best accuracy for the sensitivity study is 72.7%.

A larger dataset (10000 images from MNIST) is evaluated based on the best parameters: *trainStep*=2, *inTarget*=20 and *deTarget*=0. The accuracy for this larger dataset is 74.7%.

Compared to a single-layer fully connected ANN, which achieves 88% [30] accuracy on MNIST dataset, the proposed supervised STDP-based SNN still has an accuracy gap. Unlike the ANN, where weights and inputs can be directly used in the prediction, most SNN models only rely on the output spikes other than the exact membrane potential to make a prediction. The loss of information leads to the accuracy drops. The only single-layer SNN work in our knowledge [31] derives an extra mathematical function to extract more informations through the timing relationships for output spikes, which does not consider biological properties of neurons and synapses and hence is an unrealistic scheme for living neuron experiment.

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 a MNIST dataset [18]. That is similar to the proposed single-layer network in this paper. There are other SNN works that have better results on MNIST [14] [15] [16] [17] [19]. However, those networks have at least two layers and a larger number of neurons (e.g. 71,026 in [19]). Some works also preprocess the input images to achieve better accuracy [17]. The major difference between the proposed work and prior works is that prior works are optimized for solid state computers. For the proposed supervised scheme, bioengineering constraints are considered. Input data are compressed and applied as synchronous spike trains, HH model are used instead of the integrate and fire model. The biological limitations on maintaining the synaptic weights lead to the design of two training phases and multiple training steps.

4. CONCLUSION

To explore the possibility of using living neuron for machine learning tasks, a new supervised STDP training algorithm has been proposed and simulated on a fully-connected neural network based on the HH model. A 74.7% accuracy is achieved on digit recognition task for the MNIST dataset. This result demonstrates the feasibility of using living neurons as 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] P. King, "How is the human brain so energy-efficient?," *https://www.quora.com/How-is-the-human-brain-so-energy-efficient*, 2012.

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

[8] 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.

[9] 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.

[10] 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.

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

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] James P Keener, FC Hoppensteadt, and J Rinzel, "Integrate-and-fire models of nerve membrane response to oscillatory input," *SIAM Journal on Applied Mathematics*, vol. 41, no. 3, pp. 503–517, 1981.

[21] 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.

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

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

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

[25] 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.

[26] 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.

[27] 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.

[28] 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.

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

[30] 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.

[31] 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.