Image Recognition Using Spiking Neural Networks

Erik Sadovsky
Department of Multimedia and Information-Communication Technology, FEIT, University of Zilina,
Zilina, Slovak Republic
erik.sadovsky@feit.uniza.sk

Roman Jarina
Department of Multimedia and Information-Communication Technology, FEIT, University of Zilina,
Zilina, Slovak Republic
roman.jarina@feit.uniza.sk

Richard Orjesek
brainit.sk,
Zilina, Slovak Republic
richard.orjesek@brainit.sk

2021 31st International Conference Radioelektronika (RADIOELEKTRONIKA) | 978-1-6654-1474-6/20/$31.00 ©2021 IEEE | DOI: 10.1109/RADIOELEKTRONIKA52220.2021.9420192

***Abstract*—Spiking neural networks (SNNs), a successor to today's artificial neural networks (ANNs) represent a more realistic model of biological neuron functionality and is more computationally efficient. This predisposes it for efficient real-time pattern recognition and object detection tasks. While neurons in conventional ANNs communicate using a constant output value, neurons in SNNs communicate using spikes that are distributed in time. This functionality brings some problems in the process of encoding information into spike-like representation as well as in the SNN training. In this paper, we address some of these issues and introduced our ongoing work on SNN development. The proposed spiking multilayer perceptron and convolutional architectures were evaluated on the N-MNIST dataset for handwritten digit recognition task; the results show that the performance of the proposed solutions is comparable to the state-of-the-art and they even outperform some other related works under the comparison.**

Keywords—Spiking neural networks, N-MNIST, slayerPytorch, image recognition

I. Introduction

Spiking neural networks (SNNs) are considered to be a successor to the artificial neural networks (ANNs) that are used today because SNNs represent a more realistic model of biological neuron functionality. While neurons in today's conventional ANNs (either deep or shallow) communicate using a constant input/output values, neurons in the SNN structures communicate using spikes distributed in time. This predisposes it for efficient real-time pattern recognition and object detection tasks in various medical and surveillance applications including anomaly behaviour detection, etc. However, this essential SNN functionality brings some problems in the process of encoding the information into spike representation as well as with the training of networks of this type.

In this paper, we address some of these problems. First, we review a theory of biological neurons and related information coding, then we introduce mathematical models of spiking neurons and review some of the training algorithms. Also, we address some of the possibilities for creating databases encoded in spikes. Finally, we present our ongoing work on the development of SNNs for pattern recognition. We have already proposed and trained two types of spiking neural networks and applied them to a handwritten digit recognition task. The proposed spiking multilayer perceptron and convolutional architectures were evaluated on the N-MNIST dataset; the results show that the performance of the proposed solutions is comparable to the state-of-the-art and they even outperform some other related works under the comparison.

II. Theoretical Background

The brain is a very complex system. It consists of billions of neurons and an even bigger number of connections between them. The density of neurons is approximately 10^4 neurons and several kilometres of connections per one cubic millimetre. This density is different in other brain areas. Also, neurons in different parts of the brain have different properties. The biological neuron consists of dendrites (input), a soma (processing) and an axon (output). The connection with another neuron is called synapse. Neuron located before the synapse is called presynaptic and the neuron located after the synapse is called postsynaptic. The neuron has a membrane potential. When there are no input signals from presynaptic neurons, the membrane potential is at the resting value. When a signal is sent through the synapse it creates the postsynaptic potential (PSP). The PSPs of all synapses affect the membrane potential by either increasing it or decreasing it. When the membrane potential crosses a threshold, an output signal is generated [1], [2]. Signals used for communication between neurons are called action potentials (spikes). The action potentials are short voltage pulses with an amplitude of around 100 mV. An example spike is shown in Fig.1. At first, the membrane potential is at the resting state, but after the stimulation, the membrane potential increases and crosses the threshold (marked by the red dashed line), with the result of emitting a spike.

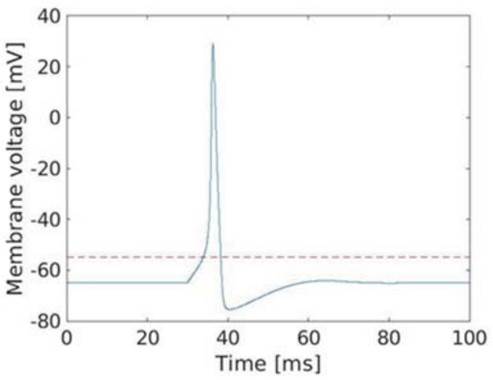

Fig. 1. An example of a spike. The blue line represents a membrane potential. The red dashed line represents the threshold

Spikes represent stimuli from some receptors or for example, the vision system. It is important to note that information is not carried in the shape of the spike, but either in the number of spikes or their timing. They are often sent in sequences, called spike trains. Several patterns were observed in spike trains. Some information is carried in the number of spikes, while other in the timing of spikes. Based on these observations, the two following main models of neural encoding have been created: Rate encoding and Temporal encoding.

The *Rate encoding* [1] is a neural encoding scheme where the information is carried in the number of spikes. The number of spikes is either averaged over a given duration or over several repeated experiments with the same input. Although this encoding scheme may be biologically plausible, sometimes there is just not enough time to wait for an average number of spikes to generate an appropriate response. An example described in [1] is that a fly can react to new stimuli from the environment and change the direction of flight in just 30 to 40 milliseconds. This reaction is often a response to only a few spikes. There is not enough time to average this input over several runs of the same stimulation or waiting a period of time to average the number of spikes.

The *Temporal encoding* (also called Pulse encoding [2]) is a scheme where the information is carried in the timing of spikes. Fewer spikes are used than with the rate encoding. The information can be carried in the timing of a single spike [3]. In the case of a periodic input stimulation, the spike can be fired when a phase change happens.

Both the encoding schemes can be applied either to a single neuron or a group of neurons [1]. For example, in the case of rate encoding, the average number of spikes can be calculated from a group of neurons. Similarly, when the temporal encoding is used, some patterns in firing times of a group of neurons can be observed based on the stimulation.

A. *Models of spiking neurons*

The models are used to simulate how a neuron (or a group of neurons) would react to certain spike trains. Some of these models are highly biologically plausible, which means they reproduce a behaviour that is very similar to a real neuron. But the more biologically plausible a model is, the more complex is the computation. In some cases, mathematical simplicity is preferred over complex computation. Therefore, some models were created as a simpler version of highly complex models.

The first mathematical model of a spiking neuron was created by Hodgkin and Huxley (H&H) [4]. In 1952, they observed a giant squid axon. Based on these observations they created an electrical scheme and a set of differential equations. This model is highly biologically plausible, but also difficult to compute. It consists of 4 nonlinear differential equations with several variables.

Simpler models were introduced to save computational resources. The passive membrane model [1] is the simplest model based on a parallel RC circuit scheme. The response to a stimulus is shaped like an exponential decay. This model is mathematically easy to compute. However, there is neither spike generating function included or the threshold functionality. This means the model is useful for modelling how a neuron reacts in the sub-threshold regions, but not for learning.

1) *Leaky integrate and fire (LIF)*

The leaky integrate and fire (LIF) model [5] is an improvement over the passive membrane model. This model has incorporated the threshold functionality ϑ as well as the spike generating function. The mathematical model is described in (1).

$$\tau \frac{du}{dt} = -(u - u_{rest}) + RI(t)$$
$$If\ u(t) = \vartheta \rightarrow fire\ a\ spike, and\ u = u_{reset} \tag{1}$$

LIF model belongs to the group of spiking neuron models called linear integrate and fire. In general, linear integrate and fire models can be described by equation (2), where *F(u)* represents a linear function.

$$\tau \frac{du}{dt} = F(u) + RI(t) \tag{2}$$

2) *Spike response model (SRM)*

The computation in the Spike response model [6], [7] is simpler because the previous models used differential equations to simulate the neuronal behaviour. The SRM parameters depend on the time of a previous spike.

The state of neurons (the membrane potential) is described by a variable *u(t)* [7].

$$u(t) = \sum_i w_i\, (\varepsilon * s_i)(t) + (\nu * s)(t) \tag{3}$$

The input spike train is represented by the variable s_i, and the output spike train is represented by the variable s. The spike response is created by convolving the spike response kernel ε with the input spike train. Each synapse has its index (*i*). The refractory response is calculated by convolving the refractory response kernel η with the output spike train.

The spike train s_i is equal to (4):

$$s_i(t) = \sum_f \delta(t - t_i^{(f)}) \tag{4}$$

The variable $t_i^{(f)}$ represents the firing time of a spike with index *f*, i is the synapse's index.

PSP is calculated by scaling each of the spike responses by the synaptic weight *w*. The membrane potential is then a sum of all PSP and refractory responses. When the membrane potential crosses a threshold value ϑ, the output spike is generated. When there is no input, *u*(*t*) has a value of a resting state.

B. *Training spiking neural networks*

In general, training of neural networks is done by using the backpropagation algorithm. However, there is a problem with using backpropagation and spiking neural networks. The function responsible for firing a spike is not differentiable because of the binary nature (there either is a spike or not). Thus, we cannot employ the commonly used ANN training methods. Another issue stated in [7] is that there is a problem with assigning errors to spikes because the effect of a spike on neuron membrane potential is not instant. There are several methods to overcome these issues [8]. In the following subsections we provide a brief overview of different learning algorithms.

1) *Unsupervised learning*

Spike Time Dependent Plasticity (STDP) [9] is an unsupervised learning method inspired by Hebb [10] and brain mechanics. This method optimizes weights based on the timing of incoming pre-synaptic spike and outcoming post-synaptic spike. When a pre-synaptic spike is received and then a post-synaptic spike is fired, there is a strong possibility that it is a result of that pre-synaptic spike. This is correct behaviour resulting in potentiating the weight of that synapse. On the other hand, if a neuron fires a spike before the receiving spikes form a presynaptic neuron, the synaptic weight is decreased. Neuron firing spontaneously is not considered a correct behaviour.

2) Supervised learning

SpikeProp [9] is one of the first supervised learning algorithms for SNN training. It is inspired by the backpropagation algorithm but has one downside of it. Each neuron can only spike once, which can lead to the neurons that are not spiking. This phenomenon is called the dead neuron problem.

The *ReSuMe* [9] algorithm is based on STDP. To prevent the dead neuron problem, this algorithm uses external teacher spikes.

The *Spike Layer Error Reassignment* (SLAYER) [7] is another supervised learning algorithm. Its authors tried to solve two problems with training SNNs. To tackle the non-differentiable spike generation function, they calculate a probability density function of a neuron emitting a spike which is done by increasing the soma potential by a small amount and calculating whether this increase caused the neuron to spike. This probability density function can be differentiated. The solution to the error reassignment problem is to calculate a correlation between the spike response kernel and the input spike train.

C. Datasets for training spiking neural networks

Since SNNs are processing spikes, we cannot use the same kind of databases that are used to train conventional ANNs [2]. In this part, we introduce some of the available solutions proposed by researchers.

There are several methods enabling using the standard benchmark MNIST database for SNN training. One of the methods is to convert MNIST into spikes by using the Poisson encoding [8].

Some devices are functioning similarly to human sensory systems. *Dynamic Vision Sensor* (DVS) [11] is a device representing a different approach to capturing image information. Today most widely used cameras are frame-based, in which the image information is captured and stored synchronously every time-frame. Thus, the intensity of every pixel is captured at precise time steps. For example, mobile phone cameras have frame rates ranging from 24 FPS up to 240 FPS, or even more. For a frame rate of 24 FPS, the sensor reads data approx. every 41.66 milliseconds, that means if a short event occurs between two successive readings this information might not be captured.

On the other hand, the DVS camera is considered an event-based system. The management of data capture is focused on each pixel individually. Information is not captured frame-by-frame, but when an intensity of light on a particular pixel reaches a certain threshold. If the intensity value rises and crosses the threshold, an *ON* event is saved with *xy* coordinates of the pixel, type of the event, and its timestamp. When the intensity drops down a threshold an *OFF* event is saved, also with the same data structure. Each pixel is writing data independently. This type of camera is able to capture events with latency in microseconds.

The DVS systems use a special file structure called *Address Event Representation* (AER) [11]. The data is encoded into streams with coordinates of each pixel, polarity (ON/OFF) and timestamp of an event (x,y,p,t).

1) Neuromorphic MNIST (N-MNIST)

N-MNIST [12] is a dataset derived from the original MNIST dataset. What makes N-MNIST unique is the process that was used to create this database. A DVS camera was used to capture each sample of MNIST. The camera was mounted on a special device moving in saccadic motion. Each sample of the MNIST database was displayed on a computer display, while the camera base did 3 saccades which consists of 100 ms for each saccade. Each saccade was done in a different direction. All the samples from the MNIST database were converted using this system, and also the original distribution of the train and test samples was preserved. This means that the N-MNIST database contains 60 000 training samples and 10 000 testing samples. The goal of the authors was to create a database that can be used as a benchmark for measuring the accuracy of spiking neural networks [12].

According to [13] this dataset can be considered as DVS-converted, which means the original database was created using a frame-based system and only converted to the event-based system using artificial motion.

There are also DVS-captured databases, that were originally captured using a DVS camera. An example is *DVS-Gestures* [13].

In the audio domain, there are several event-based databases, mainly *Spiking Heidelberg Digits* and *Spiking Speech Commands*. Both databases were created using the algorithm described in [14]. The raw audio signal is first fed into a Basilar membrane model, that uses calculation similar to a Mel spectrogram. The output of the basilar membrane model is then fed into a biologically inspired hair cell model, followed by a layer of bushy cells.

DAS Audio Digits [15] is another audio database suitable for SNN training . It was recorded using a special device called *Dynamic Audio Sensor* [16]*,* which has a binaural input with a silicon cochlea connected to 64 bandpass filters. Spikes are then generated from each of the filters.

III. Experiments

For all experiments, we used Python programming language and *PyTorch* framework with CUDA acceleration in conjunction with the *slayerPytorch* [7] framework. This framework is an implementation of the *SLAYER* algorithm (see III. B.) in Python programming language within the *PyTorch* framework.

A strong positive is that with *slayerPytorch* the neural network is spiking from the start. With some frameworks such as the *SNNToolbox* [17], a classic ANN needs to be trained first and then converted to SNN.

We used supervised learning on the N-MNIST database to create two SNN models. The first implemented model is a spiking feed-forward network (also called the Multilayer perceptron), and the second one is a spiking convolutional neural network (SCNN).

All experiments had the same configuration in terms of a loss function and neuron properties. The *slayerPytorch* framework includes the loss function that represents the rate as a spike count encoding scheme. The predicted class is assigned to the neuron in the output layer that spikes the most during the simulation time. We configured the target spike rate to 60 spikes for the correct class. However, to prevent the dead neuron situation where no training is possible without spikes, there are also false spikes from other neurons by the number of 10.

A. *Spiking multilayer perceptron (SMLP)*

For the feedforward network, we flattened the input layer from 34x34x2 dimension into 2312x1 dimension. The hidden layers were populated with 2000, 1500, 1000, 500, and 100 neurons. The output layer has 10 neurons, which is the same number as the number of classes of N-MNIST. We trained this model with the *ADAM* optimizer. The hyperparameters were set to these values: learning rate to 0.001, batch size to 8. The trained model was able to achieve an accuracy of 98.3% on the testing part of the dataset.

B. *Spiking convolutional neural network (SCNN)*

After creating a SMLP model we trained a spiking convolutional neural network. The architecture consisted of 3 convolutional blocks that includes a convolutional layer and a pooling layer. The output from these 3 convolutional blocks is fed to a fully connected output layer. In the last convolutional block, we exchanged the pooling layer for a dropout layer. We used the same loss function and batch size as in the case of SMLP, the model was optimized with an *ADAM* optimizer with the learning rate 0.001. The accuracy of this model on the training set was 97.5%.

After analysing the accuracy of SCNN, we decided to modify its architecture. Our new architecture consists of two convolutional blocks and two hidden fully connected layers. The number of neurons in the output layer is 10. Both convolutional blocks consist of the convolutional layer followed by average pooling (AP) and batch normalization (BN). We also included dropout layers after fully connected layers *Fc1* and *Fc2*. The whole architecture is summarized in Table I.

TABLE I. SCNN ARCHITECTURE

Layer	Shape
Input	34x34x2
ConvBlock1	32c5+AP+BN
ConvBlock2	16c3+AP+BN
Fc1	250
Fc2	100
Output	10

The ADAM optimizer was used with starting learning rate of 0.01. We also implemented a step learning rate scheduler that modified the learning rate after 30 epochs. This model was able to achieve an accuracy of 98.50% on the testing set. Table II. summarizes the results and compares them with some of the related works. Note, all the results were achieved by training SNNs on the N-MNIST dataset.

C. *Related works under performance comparison*

Shrestha and Orchard [7] are the authors of the SLAYER algorithm. To demonstrate its capabilities, they trained multiple SNN architectures including a SCNN on NMNIST and DVS Gestures datasets.

He et al. [13] trained SMLP with LIF neurons on the N-MNIST dataset for comparison with Recurrent neural networks. They preprocessed N-MNIST by merging multiple timesteps of input spikes to shorten the samples.

Cohen et al [18] trained SNN using the synaptic kernel inverse method that is modelled by the biological neurons and utilizes nonlinear kernels in each synapse. They trained a SNN on N-MNIST with 10 000 neurons in the hidden layer.

Iranmehr et al [19] used an ionic liquid space as a form of connecting neurons. The weights are computed based on the ion density.

Ramesh et al. [20] proposed an event-based descriptor that can be used for multiple computer vision problems such as object tracking and classification using. They used this descriptor for the classification of various datasets including N-MNIST.

Wu et al. [21] created a training algorithm that used approximate functions to cope with the inability to differentiate the spike activity and trained SMLP on the N-MNIST dataset. In [22], they created SNN that included an output layer with a population of neurons resulting in a transition from rate encoding to a neural group encoding. They trained SCNN on N-MNIST with modified LIF neurons.

TABLE II. ACCURACY ACHIEVED IN PRIOR WORKS

Work	SNN-Type	Accuracy
Cohen et al. [18]	SMLP	92.87%
Iranmehr et al. [19]	ILS-Based	97.69%
Ramesh et al. [20]	DART	97.95%
He et al. [13]	SMLP	98.28%
Wu et al. [21]	SMLP	98.78%
Shrestha, Orchard [7]	SCNN	99.2%
Wu et al. [22]	SCNN	99.53%
This work	SMLP	98.30%
This work	SCNN	98.50%

IV. CONCLUSION

In this paper we described the functionality of Spiking neural networks. This type of neural networks is considered to be more biologically similar to a brain. In the first parts of the paper, we described the basic functionality of SNNs, introduced mathematical models and some of the obstacles that are relevant to training. In the next part we described the databases that are suitable for SNN training. We also trained two SNNs with feedforward and convolutional architectures on N-MNIST database and achieved the accuracy of 98.30% and 98.50%. As is shown, the proposed architectures have performance comparable with the state-of-the-art works and even outperformed some related works.

ACKNOWLEDGMENTS

This work was supported by the Slovak Research and Development Agency under the project PP-COVID-20-0100: DOLORES.AI: The pandemic guard system.

REFERENCES

[1] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal dynamics: from single neurons to networks and models of cognition.* Cambridge, United Kingdom: Cambridge University Press, 2014.

[2] S. Ghosh-Dastidar and H. Adeli, “SPIKING NEURAL NETWORKS,” *Int. J. Neural Syst.*, vol. 19, no. 04, pp. 295–308, Aug. 2009, doi: 10.1142/S0129065709002002.

[3] W. Maass, “Networks of spiking neurons: The third generation of neural network models,” *Neural Netw.*, vol. 10, no. 9, pp. 1659–1671, Dec. 1997, doi: 10.1016/S0893-6080(97)00011-7.

[4] A. L. Hodgkin and A. F. Huxley, “A quantitative description of membrane current and its application to conduction and excitation in nerve,” *J. Physiol.*, vol. 117, no. 4, pp. 500–544, Aug. 1952, doi: 10.1113/jphysiol.1952.sp004764.

[5] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*, 1st ed. Cambridge University Press, 2002.

[6] R. Jolivet, T. J., and W. Gerstner, “The Spike Response Model: A Framework to Predict Neuronal Spike Trains,” in *Artificial Neural Networks and Neural Information Processing — ICANN/ICONIP 2003*, vol. 2714, O. Kaynak, E. Alpaydin, E. Oja, and L. Xu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 846–853.

[7] S. B. Shrestha and G. Orchard, “SLAYER: Spike Layer Error Reassignment in Time,” *Adv. Neural Inf. Process. Syst.*, vol. 31, pp. 1412–1421, 2018.

[8] L. R. Iyer, Y. Chua, and H. Li, “Is Neuromorphic MNIST neuromorphic? Analyzing the discriminative power of neuromorphic datasets in the time domain,” *ArXiv180701013 Cs*, Jul. 2018, Accessed: Jan. 26, 2021. [Online]. Available: http://arxiv.org/abs/1807.01013.

[9] A. Taherkhani, A. Belatreche, Y. Li, G. Cosma, L. P. Maguire, and T. M. McGinnity, “A review of learning in biologically plausible spiking neural networks,” *Neural Netw.*, vol. 122, pp. 253–272, Feb. 2020, doi: 10.1016/j.neunet.2019.09.036.

[10] R. G. M. Morris, “D.O. Hebb: The Organization of Behavior, Wiley: New York; 1949,” *Brain Res. Bull.*, vol. 50, no. 5–6, p. 437, Nov. 1999, doi: 10.1016/S0361-9230(99)00182-3.

[11] G. Gallego *et al.*, “Event-based Vision: A Survey,” *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 1–1, 2020, doi: 10.1109/TPAMI.2020.3008413.

[12] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, “Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades,” *Front. Neurosci.*, vol. 9, Nov. 2015, doi: 10.3389/fnins.2015.00437.

[13] W. He *et al.*, “Comparing SNNs and RNNs on neuromorphic vision datasets: Similarities and differences,” *Neural Netw.*, vol. 132, pp. 108–120, Dec. 2020, doi: 10.1016/j.neunet.2020.08.001.

[14] B. Cramer, Y. Stradmann, J. Schemmel, and F. Zenke, “The Heidelberg Spiking Data Sets for the Systematic Evaluation of Spiking Neural Networks,” *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–14, 2020, doi: 10.1109/TNNLS.2020.3044364.

[15] J. Anumula, D. Neil, T. Delbruck, and S.-C. Liu, “Feature Representations for Neuromorphic Audio Spike Streams,” *Front. Neurosci.*, vol. 12, p. 23, Feb. 2018, doi: 10.3389/fnins.2018.00023.

[16] Shih-Chii Liu, A. van Schaik, B. A. Minch, and T. Delbruck, “Asynchronous Binaural Spatial Audition Sensor With 2$\,\times\,$64$\,\times\,$4 Channel Output,” *IEEE Trans. Biomed. Circuits Syst.*, vol. 8, no. 4, pp. 453–464, Aug. 2014, doi: 10.1109/TBCAS.2013.2281834.

[17] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, “Conversion of Continuous-Valued Deep Networks to Efficient Event-Driven Networks for Image Classification,” *Front. Neurosci.*, vol. 11, p. 682, Dec. 2017, doi: 10.3389/fnins.2017.00682.

[18] G. K. Cohen, G. Orchard, S.-H. Leng, J. Tapson, R. B. Benosman, and A. van Schaik, “Skimming Digits: Neuromorphic Classification of Spike-Encoded Images,” *Front. Neurosci.*, vol. 10, 2016, doi: 10.3389/fnins.2016.00184.

[19] E. Iranmehr, S. Baghri Shouraki, and M. M. Faraji, “ILS-based Reservoir Computing for Handwritten Digits Recognition,” in *2020 8th Iranian Joint Congress on Fuzzy and intelligent Systems (CFIS)*, Mashhad, Iran, Sep. 2020, pp. 1–6, doi: 10.1109/CFIS49607.2020.9238722.

[20] B. Ramesh, H. Yang, G. Orchard, N. A. L. Thi, S. Zhang, and C. Xiang, “DART: Distribution Aware Retinal Transform for Event-Based Cameras,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 11, pp. 2767–2780, Nov. 2020, doi: 10.1109/TPAMI.2019.2919301.

[21] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, “Spatio-Temporal Backpropagation for Training High-Performance Spiking Neural Networks,” *Front. Neurosci.*, vol. 12, 2018, doi: 10.3389/fnins.2018.00331.

[22] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, and L. Shi, “Direct Training for Spiking Neural Networks: Faster, Larger, Better,” *Proc. AAAI Conf. Artif. Intell.*, vol. 33, no. 01, Art. no. 01, Jul. 2019, doi: 10.1609/aaai.v33i01.33011311.