MethodsX 10 (2023) 102157

Contents lists available at ScienceDirect

MethodsX

journal homepage: www.elsevier.com/locate/mex

Models developed for spiking neural networks

Shahriar Rezghi Shirsavar[a,b], Abdol-Hossein Vahabie[a], Mohammad-Reza A. Dehaqani[a,b,*]

[a] School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran, Iran
[b] School of Cognitive Sciences, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran

ARTICLE INFO

Method name:
Literature review

Keywords:
Spiking neural network
Biological plausibility
STDP
R-STDP
Backpropagation
ANN-to-SNN

ABSTRACT

Emergence of deep neural networks (DNNs) has raised enormous attention towards artificial neural networks (ANNs) once again. They have become the state-of-the-art models and have won different machine learning challenges. Although these networks are inspired by the brain, they lack biological plausibility, and they have structural differences compared to the brain. Spiking neural networks (SNNs) have been around for a long time, and they have been investigated to understand the dynamics of the brain. However, their application in real-world and complicated machine learning tasks were limited. Recently, they have shown great potential in solving such tasks. Due to their energy efficiency and temporal dynamics there are many promises in their future development. In this work, we reviewed the structures and performances of SNNs on image classification tasks. The comparisons illustrate that these networks show great capabilities for more complicated problems. Furthermore, the simple learning rules developed for SNNs, such as STDP and R-STDP, can be a potential alternative to replace the backpropagation algorithm used in DNNs.

- Different building blocks of spiking neural networks are explained in this work.
- Developed models for SNNs are introduced based on their characteristics and building blocks.

Specifications table

Subject area:	Neuroscience
More specific subject area:	Brain-inspired Neural Networks
Name of the reviewed methodology:	Literature review
Keywords:	Spiking neural network, Biological Plausibility, STDP, R-STDP, Backpropagation, ANN-to-SNN
Resource availability:	N.A.
Review question:	What models have been developed for spiking neural networks? What are the differences between these models? What is the level of biological plausibility of these networks?

Abbreviations: STDP, Spike-timing-dependent plasticity; R-STDP, Reward-modulated Spike-timing-dependent plasticity; ANN, Artificial neural network; SNN, Spiking neural network; DNN, Deep neural network; SVM, Support vector machine; PCA, Principal component analysis.

* Corresponding author at: School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran, Iran.
E-mail addresses: shahriar.rezghi@ut.ac.ir, shahriar.rezghi.sh@gmail.com (S. Rezghi Shirsavar), h.vahabie@ut.ac.ir (A.-H. Vahabie), dehaqani@ut.ac.ir (M.-R. A. Dehaqani).

https://doi.org/10.1016/j.mex.2023.102157
Received 13 January 2023; Accepted 23 March 2023
Available online 24 March 2023

Introduction

Artificial neural networks (ANNs) have been around for a long time. The first generation of ANNs was made of McCulloch-Pitts [1] neurons. These neurons were the result of simple modeling of the biological neurons. These neurons work with binary signals by accumulating the input and firing a spike when their internal potential reaches a certain threshold. Despite their simplicity, these neurons are powerful and are used to create multi-layer perceptron networks. The second generation of ANNs used neuron models with continuous activation functions (sigmoid [2] and ReLU [3], for example). Feedforward and recurrent networks are a part of this generation. These networks work well with analog signals and are able to approximate analog functions quite well.

In recent years, the second generation of ANNs has had major success and has been able to outperform other methods in most areas [4–6]. Although these networks are able to achieve high accuracies, they are data- [7] and energy-hungry [8] and have lower resistance to noise and disturbance [9]. The human brain is a very capable neural network. It is able to learn with a few samples and has high generalizability. It is able to store a large amount of information and has amazing energy efficiency. Mentioned problems of DNNs have led researchers to come up with a biologically plausible structure that models the brain more closely compared to the second these networks. Spiking neural networks (SNNs) are the third generation of ANNs and have a higher biological plausibility compared to DNNs. These networks use spatio-temporal information, much like biological neurons and propagate binary spike trains instead of analog signals. Several electrophysiological studies emphasize the role of temporal dynamics in neural coding [10,11]. The binary signals passed through these networks can have a high sparsity rate that is directly related to energy efficiency. This energy efficiency has led to the implementation of these networks on neuromorphic and embedded hardware [12–14]. There have also been theoretical discussions about the noise invariance of SNNs [15].

The goal of this review is to introduce different models developed for spiking neural networks and to compare them. First, we will introduce the building blocks of SNNs in Section 2 and explain their components. Then, successful networks will be introduced in Section 3 and discussed. Finally, the review will be concluded in Section 4, and possible future directions will be explored.

Building blocks

SNNs have different building blocks. Each of these blocks model a specific functionality of the brain. Since different models have been proposed for each building block, a singly defined structure does not exist for SNNs. Instead, the choice of which component to use for a building block and the connection between different components is important to have a successful SNN model. These building blocks are explored in Sections 2.1, 2.2, and 2.3. A demonstration of an SNN can be seen in Fig. 1.

Models of biological neurons

Artificial neural networks are made of modeled neurons. They model biological neurons, and in order to have a biologically plausible network, the model has to follow the spatio-temporal dynamics of a real neuron. However, biological neuron has complex dynamics, and complicated models are computationally demanding. Since the simulated network can be made of millions of neurons, using simpler models can lead to an efficient simulation. Several models have been proposed that vary in their level of complexity and computational weight. Some of these models will be introduced in Sections 2.1.1, 2.1.2, and 2.1.3.

Leaky integrate-and-fire

Leaky integrate-and-fire (LIF) [16] is a widely used model. If we want to model the neuron in a simple manner, the inputs are aggregated, and once the potential passes a threshold, the neuron fires. It is assumed that the shape of the spikes is generally the same in the LIF model, and the shape of the spike does not contain information. Instead, the information is transmitted in the presence and absence of the spikes. This model does not try to reconstruct the shape of the spike and models the neuron with a combination of a capacitator and a resistor. When the internal potential passes a certain threshold, a spike is fired, and the potential of the capacitator reverts to the resting state. The differential equation of this model is shown in Eq. (1) where C is the capacitance, g_L is the resistance, and $I(t)$ is the input current.

$$C\frac{dV}{dt} = -g_L\left(V(t) - E_L\right) + I(t) \tag{1}$$

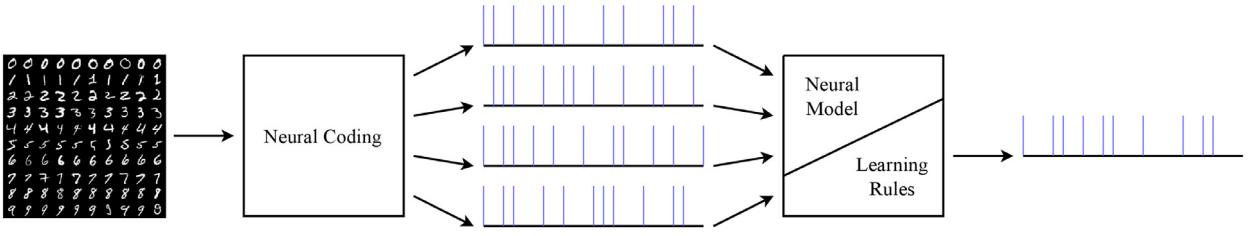

Fig. 1. A schematic for a spiking neural network process. The input signals are converted to spike trains using a coding scheme and then forwarded through a network that is made of models of the biological neurons. Learning happens in the network, and the network outputs are used for various tasks.

Spike-response model

Differential equations are often used in the modeling of biological neurons. However, in the spike-response model (SRM) [17], the parameters of the model are replaced with functions of time. This model is a generalized version of the LIF model. It can have both fixed and variable thresholds compared to the LIF model, which only has a fixed threshold. The modeling of the subthreshold activity of the SRM model is better compared to the LIF model. However, it has a simpler modeling of the spike compared to more complex models. The equation of this model is shown in Eq. (2) where $\hat{t}$ is the time of the previous spike, η function determines the shape of the action potential, and κ function determines the linear response to the delta function as input.

$$u(t) = \eta\left(t - \hat{t}\right) + \int_0^\infty \kappa\left(t - \hat{t}\right) I(t - s) ds \tag{2}$$

Izhikevich model

The Izhikevich model reproduces the spiking pattern of cortical neurons [18]. This model combines the biological plausibility of Hodgkin-Huxley models [19] (a complicated model of spiking neurons) and the computational efficiency of integrate-and-fire neurons [18]. Having such features makes this type of model suitable for SNNs. This model can be formulated in Eq. (3) with the auxiliary after-spike resetting shown in Eq. (4).

$$\begin{aligned} v' &= 0.04v^2 + 5v + 140 - u + I \\ u' &= a(bv - u) \end{aligned} \tag{3}$$

$$\text{if } v \geq 30mV, \text{ then} \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \tag{4}$$

where a, b, c, and d are dimensionless parameters, v is the membrane potential, and u is the membrane recovery variable, which accounts for the behavior of ion channels and provides negative feedback to v [18].

Neural coding

Input signals to spiking neural networks can be analog, much similar to the signals captured by human sensors (audio and images, for example) [20]. However, the modeled neurons use spikes as both input and output. In order for these networks to be able to process the analog input signals, we need to apply conversion schemes. Some of these coding schemes will be explained in Sections 2.2.1, 2.2.2, and 2.2.3.

Rate coding

Rate coding [21] is a frequently used coding mechanism, and the information is transmitted through the rate of firing of the neurons. This mechanism uses the intensity of the input signal (for example, in an image input, the value of a pixel is its intensity) for conversion, and the rate of firing is correlated to the intensity of the input. The Poisson processes can be used to model this coding scheme [22], and the mean of the distribution is the rate of firing.

Temporal coding

A lengthy window of time is needed to transfer information using rate coding, and the signal has low sparsity. Another suggested method of coding is temporal coding [23]. In this coding, the information is transmitted using the timing of the spikes. One form of this coding scheme is that the inputs that have higher values are translated into earlier firing times (see [24,25] for implementations in SNN). This coding is fast and sparse, and this sparsity can speed up SNN simulations. This coding has multiple forms, such as time-to-first-spike [26], rank order coding [27], and relative spike latency [28]. It has been suggested [29] that temporal coding might be more efficient in some situations. In rank order coding, the exact timing of the spikes is not considered, and the timings are only considered relative to each other.

Phase coding

Phase coding [30,31] is another coding scheme that encodes information in spike patterns that have phases correlating with internally generated background oscillation rhythms. In this coding, the neurons fire in different phases, and the phase is used to transmit information.

Learning

Learning strategies are an important part of a neural network. Certain learnable parameters of the network get tuned with the goal of learning robust and generalizable representations. Different learning rules for SNNs will be explored in Sections 2.3.1, 2.3.2, 2.3.3, and 2.3.4.

STDP

Although deep neural networks can achieve high accuracies with backpropagation [4–6,32], their convergence is slow, and they need a large number of labeled examples and energy in order to learn. But humans are able to learn with a few examples and without explicit labels and a small amount of energy. In the brain, learning occurs through changes in the strength of the connection between neurons. This change in the strength of connections is called synaptic plasticity [33]. Different learning rules have been developed that take advantage of synaptic plasticity. One widely recognized method is the spike-timing-dependent plasticity (STDP) learning rule [34]. This method works by adjusting synaptic weights using the timing of the spikes. In this rule, when a pre-synaptic neuron fires before a post-synaptic neuron, their connection is strengthened, and when a pre-synaptic neuron fires after a post-synaptic neuron, their connection is weakened. Changing these synaptic weights can change the spike flow of a network [35]. STDP allows the neurons to extract and learn features that are frequently seen in the input. STDP is an asymmetrical form of the Hebbian learning rule [36]. The Hebbian learning rule suggests that if a pre-synaptic neuron repeatedly or persistently takes part in the firing of the post-synaptic neuron, then their connection should be strengthened [36]. STDP takes a step further and adds a mechanism to weaken the connection, as explained earlier.

STDP is an unsupervised learning rule [34]. It can be formulated as Eq. (5) where A^+, A^-, τ^+, τ^- are the positive learning rate, negative learning rate, positive time constant, and negative time constant respectively, and $\Delta t = t_{post} - t_{pre}$.

$$\Delta W_{i,j} = \begin{cases} A^+ \exp\left(-\frac{\Delta t}{\tau^+}\right), t_{pre} < t_{post} \\ A^- \exp\left(+\frac{\Delta t}{\tau^-}\right), \ t_{pre} \geq t_{post} \end{cases} \tag{5}$$

R-STDP

There is also a reinforcement learning rule that uses STDP, and it is called reward-modulated STDP (R-STDP) [37]. Reinforcement learning helps machine learning models to learn and make a series of decisions in an environment. In this rule, the agent learns to achieve a defined goal in an uncertain and complex environment. The R-STDP learning rule changes the STDP method such that the neurons that have the correct response are rewarded, and the neurons that have the incorrect response are punished. Research suggests that when the input signals have frequent features that are not helpful in the decision-making process, the R-STDP rule can learn to ignore these features and improve the decision-making process [25].

One way of formulating the R-STDP learning rule is shown in Eq. (6) where A_r^+, A_r^-, A_p^-, A_p^+ are reward positive, reward negative, punishment negative and punishment positive learning rates and $\Delta t = t_{post} - t_{pre}$, respectively.

$$\Delta W_{i,j} = \begin{cases} \begin{cases} A_r^+ \exp\left(-\frac{\Delta t}{\tau_r^+}\right), \ t_{pre} < t_{post} \\ A_r^- \exp\left(+\frac{\Delta t}{\tau_r^-}\right), \ t_{pre} \geq t_{post} \end{cases}, \ if \ rewarded \\ \begin{cases} A_p^- \exp\left(-\frac{\Delta t}{\tau_p^-}\right), \ t_{pre} < t_{post} \\ A_p^+ \exp\left(+\frac{\Delta t}{\tau_p^+}\right), \ t_{pre} \geq t_{post} \end{cases}, \ if \ punished \end{cases} \tag{6}$$

Backpropagation

Backpropagation is a learning rule that is widely used today in deep neural networks. However, there are some problems when one tries to apply this rule to spiking neural networks. Since spike trains are not differentiable, using backpropagation is not straightforward. Despite this difficulty, various supervised learning rules have been developed for SNNs that use backpropagation (see Section 3). SpikeProp [38] can be thought of as the first backpropagation algorithm developed for SNNs. This method works by propagating the temporal error at the output. It is able to overcome the discontinuity associated with thresholding by linearizing the relationship between the post-synaptic input and the output spike time. Backpropagation methods are generally thought of as having less biological plausibility.

ANN-to-SNN

The combination of gradient descent and error backpropagation is the primary learning rule in deep neural networks. This method has been quite successful. As mentioned, these methods have been used in spiking neural networks after some modifications. However, there is another way to use this method to have a trained network. One can train a deep network, convert it to an SNN, and then go on to perform inference operations using the SNN. It is critical to consider which layers to replace in the deep network and what changes to make (see Section 3.4) to not use accuracy in the resulting network (or have a very small drop of accuracy) [39].

Libraries and toolboxes

A number of high-performance and established libraries and tools have been developed for DNNs. Some examples of these tools are PyTorch [40], TensorFlow [41], and MXNet [42]. These tools have helped researchers work faster and reach better results. There are also a number of libraries developed for SNNs, but these libraries are not comparable to DNN simulation tools in terms of runtime performance, maintenance, and stability. Some of the simulation tools for SNNs are written on top of PyTorch. Examples of these

Table 1
Considered papers and some of their characteristics including accuracy on the MNIST dataset, learning method, coding scheme, and whether convolution is used or not.

Paper	Average Accuracy (%)	Method	Coding Scheme	Convolutional
[48]	99.42 (99.01)	STDP+SVM	Temporal	Yes
[24]	98.40	STDP+SVM	Temporal	Yes
[49]	98.10	Somato-dendritic+SVM	Rate	No
[25]	97.20	STDP+*R*-STDP	Temporal	Yes
[50]	95.20	R-STDP	–	No
[51]	95.07	STDP+Voting	Rate	Yes
[52]	91.64	STDP-Like	Rate	No
[53]	91.40	STDP	Rate	Yes
[54]	91.22	Tempotron+Voting	Multi-temporal	No
[55]	99.46	ANN-to-SNN	Rate	Yes
[56]	99.44	ANN-to-SNN	Rate	Yes
[39]	99.14 (98.68)	ANN-to-SNN	Rate	Yes
[57]	98.40	ANN-to-SNN	Rate	Both
[58]	99.67	Backpropagation	Rate	Yes
[59]	99.59	Backpropagation	Rate	Yes
[60]	99.58	Backpropagation	Rate	Yes
[61]	99.57	Backpropagation	Rate	Yes
[62]	99.50	Backpropagation	Rate	Yes
[63]	99.42 (98.89)	Backpropagation	Rate	Both
[64]	99.42	Backpropagation	Rate	Yes
[59]	99.31	Backpropagation	Rate	Yes
[65]	97.40	Backpropagation	Temporal	No
[66]	97.00	Backpropagation	Temporal	No

libraries are BindsNET [43], SpykeTorch [44], and Norse [45]. Others like Brian [46] and NEST [47] are written from scratch. One notable library written from the ground up is the Spyker[1] library. It uses highly optimized low-level backends on both CPU and GPU devices, has both C++ and Python interfaces, is multiple times faster compared to its predecessors, and has the ability to be integrated with commonly used tools like PyTorch and Numpy. This library supports rank order coding, rate coding, integrate-and-fire neurons, STDP, R-STDP, and backpropagation (experimental).

Developed models

The initial set of papers considered for this review with descent accuracy and structure are listed in Table 1. These papers are mainly gathered from ScienceDirect and PubMed databases. Since the accuracies of different models will be compared, the search needs to include papers that solve the same dataset. Since the MNIST dataset is a widely used benchmark dataset, it can be used to compare the performance of different models. The average accuracy can be computed by extracting the predicted labels for testing samples of the dataset form the network, and computing the ratio of correctly labeled samples by the network. This process can be repeated with different network starting points and the average of the accuracy for all the starting points can be reported. The review aims to find the best-performing models developed for SNNs and the models that perform better than others that have similar building blocks will be explored further in Sections 3.1, 3.2, 3.3, and 3.4. Furthermore, a detailed summary of the structure and the features of the selected papers is shown in Table 2. Please note that some learning rules cannot be directly used for classification. In these cases, the support vector machine (SVM) classifier is used. Please note that all of the models listed in Table 1 are evaluated on the MNIST dataset, and they can be directly applied to problems in the field of computer vision (tuning might be needed) and to problems in other fields of machine learning after structural changes.

STDP network

The Kheradpisheh et al. network [24] combines the STDP learning rule with an external classifier. The structure for the MNIST dataset is made of two convolutional layers followed by an SVM classifier (see schematics in Fig. 2). The input is filtered with an on- and an off-center DoG filter, and temporal coding is used. Each convolutional layer is followed by an IF activation function and a pooling layer. The layers are trained with the STDP learning rule, and lateral inhibition and WTA mechanisms are employed. The training of the layers is done in sequence, and after the network is trained, the network outputs for the training and the testing set are computed, and an SVM classifier is used to classify the outputs. This network reaches an impressive testing accuracy of 98.40%. This structure works well but has less biological plausibility compared to the Mozafari et al. network [25] (explained in Section 3.2) due to the usage of the SVM classifier, which has no biological roots.

The work of Kheradpisheh et al. can be further improved, and the aim of the Shirsavar et al. network [48] is to do so. This work increases the accuracy of previous works while improving the speed of processing by orders of magnitude (a visualization of this

[1] https://github.com/ShahriarRezghi/Spyker

Table 2
In this table, explained papers and their highlighted features are summarized and their accuracies, biological plausibility, and sparsity levels are mentioned.

Paper	Description	Average Accuracy(%)	Biological Plausibility	Sparsity
[25]	DoG feature enhancement Temporal coding Three convolution layers with IF and lateral inhibition STDP in first two layers, R-STDP in third Classification with assigning channels to labels	97.20	High	High
[24]	DoG feature enhancement Temporal coding Two convolution layers with IF and lateral inhibition STDP learning rule Classification using SVM	98.40	Medium	High
[48]	Based on previous networks Neuron firing times as output PCA before SVM to improve runtime speed Network weight quantization	99.42 (99.01)	Medium	High
[39]	Convert ANN to SNN ReLU and dropout in ANN Normalizing of weights to close accuracy gap Two convolution Layers, one fully connected layer	99.14	Low	Low
[55]	Train weight normalization parameters Replace ReLU with rate norm layer (RNL) Add a parameter to the loss function	99.46	Low	Low
[67]	Train SNN with backpropagation LIF with lateral inhibition Treat spikes as continuous signals Normalizing weights to stabilize gradients Weight and threshold regularization Adam optimizer and Data augmentation two convolution layers, two fully connected layers	99.31	Low	Low
[58]	Use average firing rate to determine output label Train with mean squared error Only calculate the gradient at the moment of spike Add residual connections between spikes in the backward path	99.67	Low	Low

network can be seen in Fig. 3). It is based on previous networks and makes some changes. The first change happens in the output of the network. The outputs are changed from binary indication of spikes to the firing time of the output neurons. The following change is to introduce a dimension reduction before the SVM classification process. This dimension reduction (principal component analysis or PCA) has little to no effect on the accuracy, while it improves the runtime significantly. The final change consists of an improved training process. With better control of learning parameters and quantizing the weights afterward, the training iterations can be significantly reduced (only one iteration for the MNIST dataset). This network reaches $99.011 \pm 0.068\%$ accuracy with a linear classifier and $99.421 \pm 0.033\%$ with a non-linear classifier. The biological plausibility of this network is much like the Kheradpisheh et al. Network.

R-STDP network

STDP is an unsupervised learning rule, and it alone is not enough to solve classification tasks. The Mozafari et al. [25] network solves this problem by combining it with the R-STDP learning rule. It proposes a network structure that has several components (see schematics in Fig. 4). The structure includes three on- and off-center Difference of Gaussian (DoG) filters on the inputs for feature enhancement. The network uses a temporal coding scheme and has three convolutional layers, each with an integrate-and-fire (IF) activation function and a pooling operation. The training of the layers is done in sequence (one layer after another). Lateral inhibition is used in this network and is an essential part of the learning process. In this mechanism, when a neuron fires in a feature map, it stops other neurons in the same position belonging to other feature maps from firing. After inhibition, a number of neurons are selected using a winner-take-all (WTA) mechanism, and the STDP learning rule is applied to them. Lateral inhibition and WTA force the neurons to compete with each other in order to learn. The first and the second layers use the STDP learning rule. However, the third layer uses the R-STDP learning rule, and its activation function has an infinite threshold. The infinite threshold stops the neurons from firing, and the internal potentials of the neurons are used instead. In this layer, output channels are assigned to different output classes, and the neuron with the highest potential is selected. If the mapped class label of the channel that the neuron belongs to is the same as the class label of the input image, then the neuron will be rewarded (see Section 2.3.2). If the labels are not the same, it will be punished. This structure has a high biological plausibility due to the fact that it uses DoG filters, STDP and R-STDP learning rules and does not use a classifier with no biological roots (SVM, for example). It achieves a good test accuracy of 97.20%.

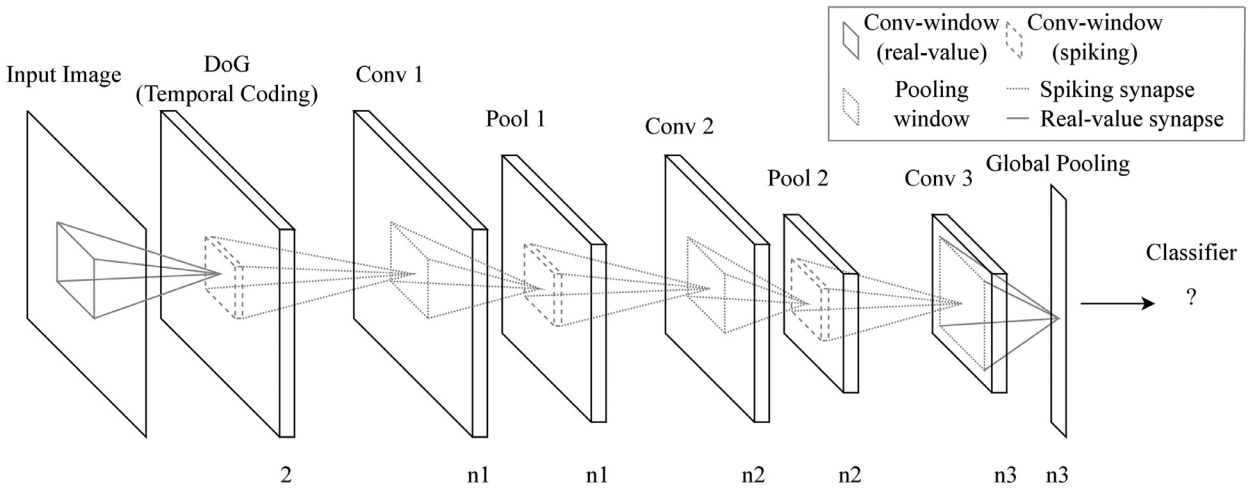

Fig. 2. Schematics for the Kheradpisheh et al. network. This network uses an on-center and an off-center DoG filters on the inputs with rank order coding afterwards, has two convolutional layers, each followed by a pooling layer, and an SVM classifier at the end. Please note that the third convolutional layer is not used on the MNIST dataset.

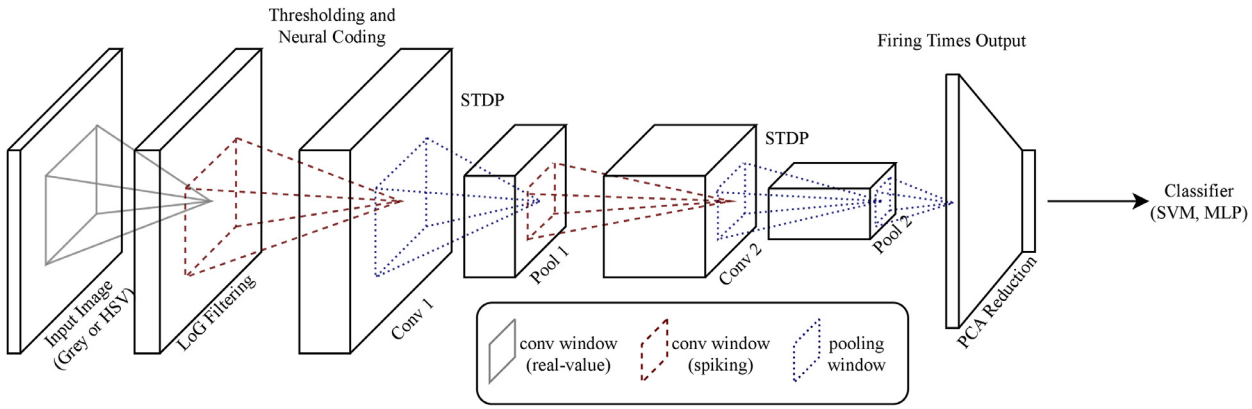

Fig. 3. Overview of Shirsavar et al. Network. This network uses three LoG filters with rank order coding. It has two convolutional layers, each followed by a pooling layer. Finally, PCA dimension reduction is used on the network outputs, and the SVM classifier is run.

Backpropagation network

Backpropagation has been the primary learning rule in DNNs, and one approach that is researched is to apply backpropagation directly to an SNN network. Since spikes passed through the network are not differentiable because of discontinuous step jumps, applying backpropagation is not as straightforward, but Lee et al. [67] propose a method to solve this problem. In this work, leaky integrate-and-fire neural model and lateral inhibition are used to model biological neurons and introduce competition in the network architecture, respectively. This method ignores fluctuations in the spikes and treats them as continuous signals which helps it derive the necessary error gradients for the backpropagation. Weight normalization is also used to avoid vanishing or exploding gradients as the network becomes deeper (has more layers). Weights are uniformly initialized, and the threshold is calculated with respect to the number of synapses of each neuron. Weight decay and threshold regularization are used in this model to improve stability and generalizability. Adam optimization [68] is employed here, and the network has two convolutional layers, each following a pooling operation. The network also uses two fully connected layers to compute the output of the model. This structure achieves a high testing accuracy of 99.31%. However, data augmentation is used to improve the results, and the accuracy without the augmentation is not reported. Since backpropagation is used, the network lacks biological plausibility.

Another work that reaches state-of-the-art results is the Shen et al. network [58]. This work uses the average firing rate of the last layer to determine the output label and trains the network with mean squared error. Previous works based on surrogate-gradient methods calculate the gradients around the threshold. This method does not account for the fact that the neurons not emiting spikes in the forward path will participate in the parameter update process. Also, earlier spike moments will have a larger influence on the weight update compared to later ones. This conflicts with the neurophysiology rules [58]. This work proposes the biologically plausible spatial adjustment (BSPA) that only calculates the gradient of the neuron at the moment of spiking to the membrane

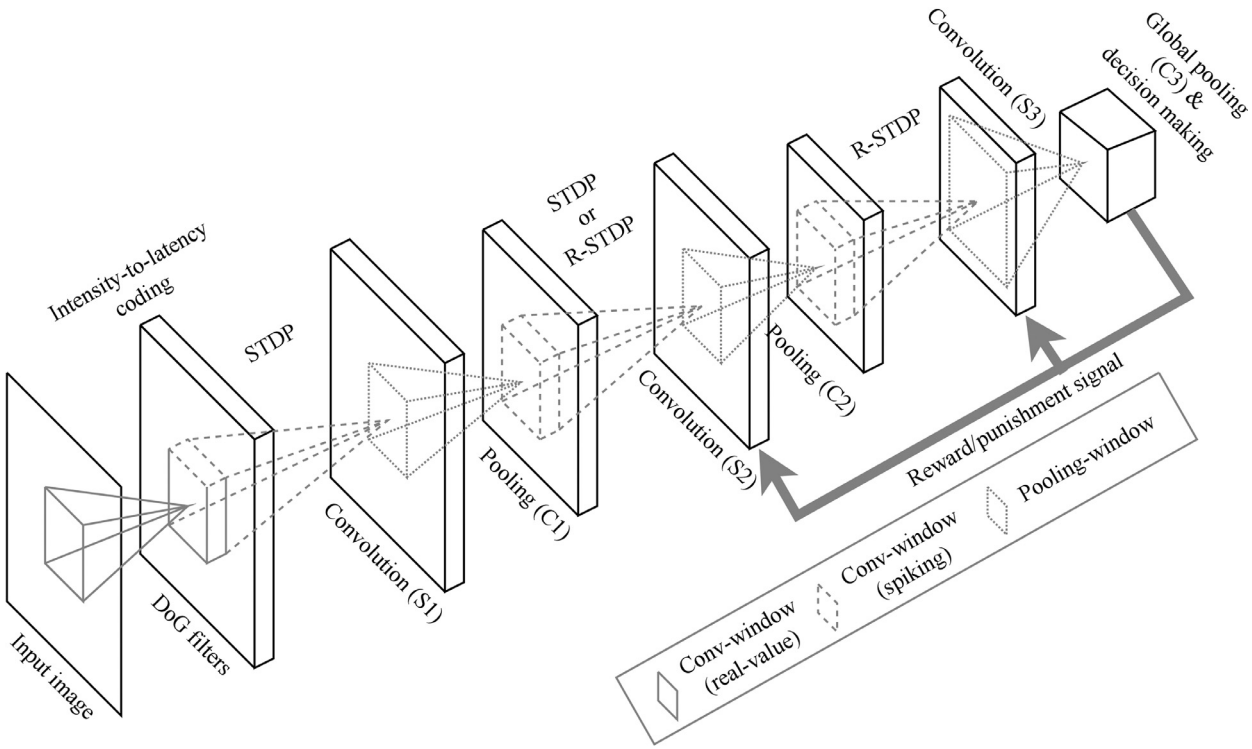

Fig. 4. Schematics for the Mozafari et al. network. The network uses three on-center and three off-center DoG filters on inputs with rank order coding afterward, has three convolutional layers, each followed by a pooling layer, and the STDP and R-STDP learning rules are used to train the network. The network has a native classifier.

potential. It also proposes the biologically plausible temporal adjustment (BSTA). Spikes that occur affect the spikes that will occur in the future, and the backpropagation algorithm does not consider the influence between the spikes [58]. This work improves this process by adding a residual connection between the spikes during the backward path.

ANN-to-SNN network

Since deep neural networks have had major success, and algorithms and tools for their training and evaluation are developed and established in recent years, some researchers have looked for a way to bring this knowledge to spiking neural networks. More specifically, their aim has been to convert trained DNN models to SNNs for inference. Initially, attempts at this conversion resulted in a big accuracy loss. However, this accuracy gap has been filled since by developing new methods. The Diehl et al. paper [39] proposes a conversion scheme. To have minimal conversion loss, the architecture of the DNN network must have some conditions. First, the closest mechanism to the integrate-and-fire activation function in SNNs is rectified linear unit (ReLU) function. Using a ReLU activation function in the DNN structure helps avoid challenges related to negative values and biases [39]. Dropout [69] layers can be used to reduce problems related to overfitting and the conversion accuracy loss following it.

The paper introduces a weight normalization method that can be used to have near-lossless and faster convergence. Two methods are proposed in the Diehl et al. paper for weight normalization. The first method considers all positive activations that could occur as input to a layer, and all weights are rescaled by the maximum possible positive input. It is trivial to see this method requires a considerable amount of time to complete. The second method records all the activation values in the ReLU layers for the training set, and after the training is completed, the maximum value is extracted. This method can be faster but makes the decision solely based on the training set. The network used in this work is made of two convolutional layers, each followed by a pooling operation and a fully connected layer at the end. This work achieves an excellent testing accuracy of 99.14%, but the conversion scheme eliminates biological plausibility.

Another work that improved the ANN-to-SNN process is the work of Ding et al. [55]. The problem with previous methods was that the weight normalization process relied on empirical methods and could only happen after the training process. This work proposes a layer called Rate Norm Layer (RNL) to replace the ReLU layer in the ANN. This layer uses a clip function with a trainable upper bound to output the simulated firing rate [55]. These trainable parameters cannot be learned through the original loss function. Thus, a new term is added to the original loss function of the network. This network reaches a testing accuracy of 99.67%.

Conclusion

Deep neural networks have advanced greatly in recent years. They have achieved amazing results and have become state-of-the-art models in some fields of machine learning. However, these networks have shortcomings, such as high energy usage and requiring a large amount of data to learn. The human brain does not have these shortcomings and is able to solve complicated tasks. Although deep neural networks try to model the brain, there are differences between the two. Spiking neural networks are the next generation of neural networks that model the brain better compared to deep neural networks. These networks process spike trains, use more plausible learning rules and model biological neurons better. More attention has been paid to these networks, and researchers have developed several models and structures for these networks. The proposed structures vary in their building blocks, learning rules, and structures.

In this paper, we explained the neural models, coding schemes, learning rules, and simulation tools of spiking neural networks. We listed recent and successful proposed structures for these networks and explored the better-performing ones further.

Having a network that processes binary input has some advantages. These networks have the potential to be highly energy efficient, and this efficiency makes these networks suitable to be implemented on specialized hardware. Furthermore, replacing a good percentage of floating-point operations with integer ones can yield faster runtimes.

Despite the recent progress and developments in the field of spiking neural networks, these networks are not yet comparable to deep neural networks in terms of accuracy. These networks have been able to solve a subset of machine learning problems well, but this success must be spread to other areas and harder problems. This can happen by studying the brain and finding what makes it so great, introducing more complicated dynamics to the networks, and finding structures that perform better.

Ethics statements

The authors have followed MethodsX ethical guidelines. This work does not involve human subjects, animal experiments, or data collected from social media.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Shahriar Rezghi Shirsavar: Methodology, Writing – original draft. **Abdol-Hossein Vahabie:** Conceptualization, Writing – review & editing. **Mohammad-Reza A. Dehaqani:** Supervision, Writing – review & editing.

Data availability

No data was used for the research described in the article.

Acknowledgments

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

[1] W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5 (1943) 115–133, doi:10.1007/BF02478259.
[2] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (1986) 533–536, doi:10.1038/323533a0.
[3] V. Nair, G.E. Hinton, Rectified linear units improve restricted boltzmann machines, in: Proceedings of the 27th International Conference on International Conference on Machine Learning, Omnipress, Madison, WI, USA, 2010, pp. 807–814.
[4] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natural language processing (Almost) from scratch, J. Mach. Learn. Res. 12 (2011) 2493–2537.
[5] A. Graves, J. Schmidhuber, Framewise phoneme classification with bidirectional LSTM and other neural network architectures, Neural Netw. 18 (2005) 602–610, doi:10.1016/j.neunet.2005.06.042.
[6] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, Commun. ACM 60 (2017) 84–90, doi:10.1145/3065386.
[7] C. Sun, A. Shrivastava, S. Singh, A. Gupta, Revisiting Unreasonable Effectiveness Of Data In Deep Learning Era, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV, 2017, pp. 843–852, doi:10.1109/ICCV.2017.97.
[8] D. Li, X. Chen, M. Becchi, Z. Zong, Evaluating the energy efficiency of deep convolutional neural networks on CPUs and GPUs, in: Proceedings of the IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom), 2016, pp. 477–484, doi:10.1109/BDCloud-SocialCom-SustainCom.2016.76.
[9] M.A. Alcorn, Q. Li, Z. Gong, C. Wang, L. Mai, W.S. Ku, A. Nguyen, Strike (With) a pose: neural networks are easily fooled by strange poses of familiar objects, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 4840–4849, doi:10.1109/CVPR.2019.00498.
[10] M.-R.A. Dehaqani, A.-.H. Vahabie, R. Kiani, M.N. Ahmadabadi, B.N. Araabi, H. Esteky, Temporal dynamics of visual category representation in the macaque inferior temporal cortex, J. Neurophysiol. 116 (2016) 587–601, doi:10.1152/jn.00018.2016.
[11] M.-R.A. Dehaqani, A.-.H. Vahabie, M. Parsa, B. Noudoost, A. Soltani, Selective changes in noise correlations contribute to an enhanced representation of saccadic targets in prefrontal neuronal ensembles, Cereb. Cortex 28 (2018) 3046–3063, doi:10.1093/cercor/bhy141.

[12] P.K. Huynh, M.L. Varshika, A. Paul, M. Isik, A. Balaji, A. Das, Implementing spiking neural networks on neuromorphic architectures: a review, (2022). 10.48550/arXiv.2202.08897.
[13] K. Akbarzadeh-Sherbaf, S. Safari, A.-.H. Vahabie, A digital hardware implementation of spiking neural networks with binary FORCE training, Neurocomputing 412 (2020) 129–142, doi:10.1016/j.neucom.2020.05.044.
[14] K. Akbarzadeh-Sherbaf, B. Abdoli, S. Safari, A.-.H. Vahabie, A scalable FPGA architecture for randomly connected networks of hodgkin-huxley neurons, Front. Neurosci. 12 (2018) https://www.frontiersin.org/articles/10.3389/fnins.2018.00698. accessed December 18, 2022.
[15] M. Beer, J. Urenda, O. Kosheleva, V. Kreinovich, Why spiking neural networks are efficient: a theorem, in: Information Processing and Management of Uncertainty in Knowledge-Based Systems, Springer International Publishing, Cham, 2020, pp. 59–69, doi:10.1007/978-3-030-50146-4_5.
[16] L.F. Abbott, Lapicque's introduction of the integrate-and-fire model neuron (1907), Brain Res. Bull. 50 (1999) 303–304, doi:10.1016/s0361-9230(99)00161-6.
[17] R. Jolivet, J. T, W. Gerstner, The spike response model: a Framework to predict neuronal spike trains, in: Artificial Neural Networks and Neural Information Processing — ICANN/ICONIP 2003, Springer, Berlin, Heidelberg, 2003, pp. 846–853, doi:10.1007/3-540-44989-2_101.
[18] E.M. Izhikevich, Simple model of spiking neurons, IEEE Trans. Neural Netw. 14 (2003) 1569–1572, doi:10.1109/TNN.2003.820440.
[19] A.L. Hodgkin, A.F. Huxley, A quantitative description of membrane current and its application to conduction and excitation in nerve, J. Physiol. 117 (1952) 500–544.
[20] V. Kostakos, J. Rogstadius, D. Ferreira, S. Hosio, J. Goncalves, et al., Human Sensors, in: Participatory Sensing, Opinions and Collective Awareness, Springer International Publishing, Cham, 2017, pp. 69–92, doi:10.1007/978-3-319-25658-0_4.
[21] E.D. Adrian, Y. Zotterman, The impulses produced by sensory nerve endings: part 3. impulses set up by touch and pressure, J. Physiol. 61 (1926) 465–483, doi:10.1113/jphysiol.1926.sp002308.
[22] M.C. Wiener, B.J. Richmond, Decoding spike trains instant by instant using order statistics and the mixture-of-Poissons model, J. Neurosci. 23 (2003) 2394–2406, doi:10.1523/JNEUROSCI.23-06-02394.2003.
[23] G. Buzsáki, Rhythms of the Brain, Oxford University Press, New York, 2006, doi:10.1093/acprof:oso/9780195301069.001.0001.
[24] S.R. Kheradpisheh, M. Ganjtabesh, S.J. Thorpe, T. Masquelier, STDP-based spiking deep convolutional neural networks for object recognition, Neural Netw. 99 (2018) 56–67, doi:10.1016/j.neunet.2017.12.005.
[25] M. Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, S.J. Thorpe, T. Masquelier, Bio-inspired digit recognition using reward-modulated spike-timing-dependent plasticity in deep convolutional networks, Pattern Recognit. 94 (2019) 87–95, doi:10.1016/j.patcog.2019.05.015.
[26] R.S. Johansson, I. Birznieks, First spikes in ensembles of human tactile afferents code complex spatial fingertip events, Nat. Neurosci. 7 (2004) 170–177, doi:10.1038/nn1177.
[27] S. Thorpe, J. Gautrais, Rank Order Coding (1998) 113–118, doi:10.1007/978-1-4615-4831-7_19.
[28] T. Gollisch, M. Meister, Rapid neural coding in the retina with relative spike latencies, Science 319 (2008) 1108–1111, doi:10.1126/science.1149639.
[29] J. Gautrais, S. Thorpe, Rate coding versus temporal order coding: a theoretical approach, Biosystems 48 (1998) 57–65, doi:10.1016/s0303-2647(98)00050-1.
[30] J. O'Keefe, M.L. Recce, Phase relationship between hippocampal place units and the EEG theta rhythm, Hippocampus 3 (1993) 317–330, doi:10.1002/hipo.450030307.
[31] G. Laurent, Dynamical representation of odors by oscillating and evolving neural assemblies, Trends Neurosci. 19 (1996) 489–496, doi:10.1016/S0166-2236(96)10054-0.
[32] M. Tan, Q.V. Le, EfficientNet: rethinking model scaling for convolutional neural networks, (2020). 10.48550/arXiv.1905.11946.
[33] A. Citri, R.C. Malenka, Synaptic plasticity: multiple forms, functions, and mechanisms, Neuropsychopharmacol 33 (2008) 18–41, doi:10.1038/sj.npp.1301559.
[34] H. Markram, J. Lübke, M. Frotscher, B. Sakmann, Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs, Science 275 (1997) 213–215, doi:10.1126/science.275.5297.213.
[35] J. Vreeken, Spiking neural networks, an introduction, Undefined. (2003). /paper/Spiking-neural-networks/2C-an-introduction-Vreeken/4ced5f507d2ca65c35580bcd1709969650171a55.
[36] D.O. Hebb, The Organization of Behavior; a Neuropsychological Theory, Wiley, Oxford, England, 1949.
[37] N. Frémaux, W. Gerstner, Neuromodulated Spike-Timing-Dependent Plasticity, and Theory of Three-Factor Learning Rules, Front. Neural Circuits 9 (2016) https://www.frontiersin.org/article/10.3389/fncir.2015.00085.
[38] S. Bohte, J. Kok, J. Poutré, SpikeProp: backpropagation for networks of spiking neurons., in: 2000: pp. 419–424.
[39] P.U. Diehl, D. Neil, J. Binas, M. Cook, S.-.C. Liu, M. Pfeiffer, Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing, in: Proceedings of the 2015 International Joint Conference on Neural Networks (IJCNN), 2015, pp. 1–8, doi:10.1109/IJCNN.2015.7280696.
[40] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: an imperative style, high-performance deep learning library, Advances in Neural Information Processing Systems, Curran Associates, Inc., 2019 https://papers.nips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.
[41] T. Developers, TensorFlow, (2022). 10.5281/zenodo.6574269.
[42] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, Z. Zhang, MXNet: a flexible and efficient machine learning library for heterogeneous distributed systems, ArXiv:1512.01274 [Cs]. (2015). https://arxiv.org/abs/1512.01274.
[43] H. Hazan, D.J. Saunders, H. Khan, D. Patel, D.T. Sanghavi, H.T. Siegelmann, R. Kozma, BindsNET: a machine learning-oriented spiking neural networks library in python, Front. Neuroinform. 12 (2018), doi:10.3389/fninf.2018.00089.
[44] M. Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, T. Masquelier, SpykeTorch: efficient simulation of convolutional spiking neural networks with at most one spike per neuron, Front. Neurosci. 13 (2019), doi:10.3389/fnins.2019.00625.
[45] C.-.G. Pehle, J.E. Pedersen, Norse - A deep learning library for spiking neural networks, (2021). doi:10.5281/zenodo.4422025.
[46] M. Stimberg, R. Brette, D.F. Goodman, Brian 2, an intuitive and efficient neural simulator, eLife 8 (2019) e47314, doi:10.7554/eLife.47314.
[47] M.-.O. Gewaltig, M. Diesmann, NEST (NEural Simulation Tool), Scholarpedia 2 (2007) 1430, doi:10.4249/scholarpedia.1430.
[48] S.R. Shirsavar, M.R.A. Dehaqani, A faster approach to spiking deep convolutional neural networks, (2022). 10.48550/arXiv.2210.17442.
[49] D. Drix, V.V. Hafner, M. Schmuker, Sparse coding with a somato-dendritic rule, Neural Netw. 131 (2020) 37–49, doi:10.1016/j.neunet.2020.06.007.
[50] Z. Xu, S. Skorheim, M. Tu, V. Berisha, S. Yu, J. Seo, M. Bazhenov, Y. Cao, Improving efficiency in sparse learning with the feedforward inhibitory motif, Neurocomputing 267 (2017) 141–151, doi:10.1016/j.neucom.2017.05.016.
[51] D.J. Saunders, D. Patel, H. Hazan, H.T. Siegelmann, R. Kozma, Locally connected spiking neural networks for unsupervised feature learning, Neural Netw. 119 (2019) 332–340, doi:10.1016/j.neunet.2019.08.016.
[52] M. Beyeler, N.D. Dutt, J.L. Krichmar, Categorization and decision-making in a neurobiologically plausible spiking network using a STDP-like learning rule, Neural Netw. 48 (2013) 109–124, doi:10.1016/j.neunet.2013.07.012.
[53] Q. Xu, J. Peng, J. Shen, H. Tang, G. Pan, Deep CovDenseSNN: a hierarchical event-driven dynamic framework with spiking neurons in noisy environment, Neural Netw. 121 (2020) 512–519, doi:10.1016/j.neunet.2019.08.034.
[54] T. Wang, C. Shi, X. Zhou, Y. Lin, J. He, P. Gan, P. Li, Y. Wang, L. Liu, N. Wu, G. Luo, CompSNN: a lightweight spiking neural network based on spatiotemporally compressive spike features, Neurocomputing 425 (2021) 96–106, doi:10.1016/j.neucom.2020.10.100.
[55] J. Ding, Z. Yu, Y. Tian, T. Huang, Optimal ANN-SNN conversion for fast and accurate inference in deep spiking neural networks, (2021). doi:10.48550/arXiv.2105.11654.
[56] B. Rueckauer, I.A. Lungu, Y. Hu, M. Pfeiffer, S.C. Liu, Conversion of continuous-valued deep networks to efficient event-driven networks for image classification, Front. Neurosci. 11 (2017) https://www.frontiersin.org/articles/10.3389/fnins.2017.00682. accessed November 30, 2022.
[57] A. Zhang, H. Zhou, X. Li, W. Zhu, Fast and robust learning in spiking feed-forward neural networks based on intrinsic plasticity mechanism, Neurocomputing 365 (2019) 102–112, doi:10.1016/j.neucom.2019.07.009.
[58] G. Shen, D. Zhao, Y. Zeng, Backpropagation with biologically plausible spatiotemporal adjustment for training deep spiking neural networks, Patterns 3 (2022) 100522, doi:10.1016/j.patter.2022.100522.

[59] C. Lee, S.S. Sarwar, P. Panda, G. Srinivasan, K. Roy, Enabling spike-based backpropagation for training deep neural network architectures, Front. Neurosci. 14 (2020) https://www.frontiersin.org/articles/10.3389/fnins.2020.00119. accessed November 30, 2022.
[60] D. Zhao, Y. Zeng, Y. Li, BackEISNN: a deep spiking neural network with adaptive self-feedback and balanced excitatory-inhibitory neurons, (2021). doi:10.48550/arXiv.2105.13004.
[61] W. Zhang, P. Li, Spike-train level backpropagation for training deep recurrent spiking neural networks, (2019). doi:10.48550/arXiv.1908.06378.
[62] W. Zhang, P. Li, Temporal spike sequence learning via backpropagation for deep spiking neural networks, in: Advances in Neural Information Processing Systems, Curran Associates, Inc., 2020, pp. 12022–12033. https://proceedings.neurips.cc/paper/2020/hash/8bdb5058376143fa358981954e7626b8-Abstract.html.
[63] Y. Wu, L. Deng, G. Li, J. Zhu, L. Shi, Spatio-temporal backpropagation for training high-performance spiking neural networks, Front. Neurosci. 12 (2018) https://www.frontiersin.org/articles/10.3389/fnins.2018.00331. accessed November 30, 2022.
[64] X. Cheng, Y. Hao, J. Xu, B. Xu, LISNN: improving spiking neural networks with lateral interactions for robust object recognition, in: 2020: pp. 1519–1525. doi:10.24963/ijcai.2020/211.
[65] S.R. Kheradpisheh, T. Masquelier, Temporal backpropagation for spiking neural networks with one spike per neuron, Int. J. Neur. Syst. 30 (2020) 2050027, doi:10.1142/S0129065720500276.
[66] S.R. Kheradpisheh, M. Mirsadeghi, T. Masquelier, BS4NN: binarized spiking neural networks with temporal coding and learning, Neural Process Lett. 54 (2022) 1255–1273, doi:10.1007/s11063-021-10680-x.
[67] J.H. Lee, T. Delbruck, M. Pfeiffer, Training deep spiking neural networks using backpropagation, Front. Neurosci. (2016) 10, doi:10.3389/fnins.2016.00508.
[68] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, (2017). doi:10.48550/arXiv.1412.6980.
[69] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (2014) 1929–1958.