Hindawi
Computational Intelligence and Neuroscience
Volume 2022, Article ID 1633946, 11 pages
https://doi.org/10.1155/2022/1633946

Research Article

Spike-Based Approximate Backpropagation Algorithm of Brain-Inspired Deep SNN for Sonar Target Classification

Yang Liu,[1,2,3] Meng Tian,[1,2,3] Ruijia Liu,[1,2,4] Kejing Cao,[1,2,3] Ruiyi Wang,[2,3,5] Yadi Wang,[1,2,3] Wei Zhao,[1,6] and Yi Zhou[3,7]

[1]*Henan Province Engineering Research Center of Spatial Information Processing, Kaifeng 475004, China*
[2]*College of Computer and Information Engineering, Henan University, Kaifeng 475004, China*
[3]*Shenzhen Research Institute, Henan University, Shenzhen 518000, China*
[4]*College of Software, Henan University, Kaifeng 475004, China*
[5]*Henan Key Laboratory of Big Data Analysis and Processing, Kaifeng 475004, China*
[6]*Miami College, Henan University, Kaifeng 475004, China*
[7]*College of Artificial Intelligence, Henan University, Zhengzhou 450046, China*

Correspondence should be addressed to Yang Liu; ly.sci.art@gmail.com and Wei Zhao; henuzhao@vip.henu.edu.cn

Received 1 December 2021; Revised 22 May 2022; Accepted 8 August 2022; Published 20 October 2022

Academic Editor: José Alfredo Hernández-Pérez

With the development of neuromorphic computing, more and more attention has been paid to a brain-inspired spiking neural network (SNN) because of its ultralow energy consumption and high-performance spatiotemporal information processing. Due to the discontinuity of the spiking neuronal activation function, it is still a difficult problem to train brain-inspired deep SNN directly, so SNN has not yet shown performance comparable to that of an artificial neural network. For this reason, the spike-based approximate backpropagation (SABP) algorithm and a general brain-inspired SNN framework are proposed in this paper. The combination of the two can be used for end-to-end direct training of brain-inspired deep SNN. Experiments show that compared with other spike-based methods of directly training SNN, the classification accuracy of this method is close to the best results on MNIST and CIFAR-10 datasets and achieves the best classification accuracy on sonar image target classification (SITC) of small sample datasets. Further analysis shows that compared with artificial neural networks, our brain-inspired SNN has great advantages in computational complexity and energy consumption in sonar target classification.

1. Introduction

Neural computing is the main driving force of the current development of artificial intelligence. In recent years, with the development of various deep learning technologies [1–5], artificial neural networks (ANNs) have been widely applied in many fields and achieved remarkable results (such as target detection, speech recognition, and character recognition). However, due to a large number of parameters, the massive training samples, and the huge energy consumption required by ANN training and deployment, ANN is still difficult to be applied to edge smart devices (such as smart watches, smart detectors, and other unmanned autonomous systems). High energy consumption and high-performance computing requirements have become the main bottlenecks for the continued development of neural networks for sonar target classification in unmanned underwater vehicles.

The brain-inspired spiking neural network (SNN) is usually sparse, and the calculation is driven by events. The high multiplication calculation costs of ANN can be avoided through discrete binary spike signals, showing ultralow power consumption [6]. Coordinate neuromorphic hardware [7–11] shows the prospect of realizing low-power artificial intelligence, which is expected to break through the current bottleneck of neural computing. The spiking neuron is inspired by biological neurons that efficiently process discrete spatiotemporal spikes, and the main neuronal models include the leaky integrate-and-fire (LIF) model [12], the Izhikevich model [13], and the Hodgkin–Huxley model [14]. LIF neurons greatly simplify the process of the action potential of biological neurons. The membrane potential integrates the input current over time, and when the membrane potential exceeds the threshold, the neuron will fire a spike. At present, there have been several hardware circuits [15] based on the LIF neuron model. Due to the discontinuity and nondifferentiability of spiking neuronal activation function, the traditional ANN backpropagation (BP) training algorithm based on gradient descent cannot be directly applied to brain-inspired SNN. The training of the brain-inspired SNN is still a great challenge. At present, the main work can be divided into two categories: the ANN-SNN conversion method and the brain-inspired SNN direct training method.

In the ANN-SNN conversion method, ANN is trained and then converted into SNN by a specific means to replace the training of SNN. Sengupta et al. [16] achieved 91.55% accuracy in the CIFAR-10 dataset with a loss of 0.15% conversion accuracy. After ANN-SNN conversion, Rathi et al. [17] fine-tuned the model for training and achieved 92.22% accuracy in the CIFAR-10 dataset and reduced the inference time. Stockl and Maass proposed a method for few spike conversion (FS-Conversion) [18], achieving 92.42% accuracy on the CIFAR-10 dataset. Because the state-of-the-art (SOTA) training methods of ANN are used, many current ANN-SNN conversion methods obtain SNN SOTA classification performance. However, the ANN-SNN conversion method usually imposes constraints on the original ANN, which will lead to a decline in performance. Moreover, most of the ANN-SNN conversion methods require hundreds to thousands of time steps to complete one inference, resulting in additional time delays and energy consumption contrary to the goal.

Direct training methods for SNN mainly include unsupervised learning and supervised learning. Unsupervised learning generally only involves local signals of synapses, such as the spike-timing-dependent plasticity (STDP) algorithm. Diehl and Cook [19] achieved a classification accuracy of 95% on the MNIST dataset using the STDP trained network. Kheradpisheh et al. [20] used STDP and the support vector machine (SVM) to achieve a classification accuracy of 98.4% on the MNIST dataset, but the accuracy was still far behind that of ANN. In order to narrow the gap, researchers put forward the spike-based BP rule. Jin et al. proposed the HM2-BP (hybrid macro/microlevel BP) [21] algorithm, and the error BP in SNNs was deconstructed into two processes: the change of postsynaptic potential caused by a single spike at the microlevel and the loss function defined by frequency coding at the macrolevel. Based on the approximate derivative of the spiking neuronal activation function, Wu et al. proposed a spatio-temporal backpropagation (STBP) [22] algorithm combining the spatial domain with the time domain in SNNs. Wu et al. also proposed a neural normalization technique (NeuNorm) [23], which achieved good results when combined with the STBP algorithm on CIFAR-10 datasets. Lee et al. [24] processed the derivative of LIF neurons as the approximate derivative of IF neurons and calculated the corresponding leak correction compensation. The authors in [25] proposed an SNN training algorithm that is capable of learning not only the synaptic weights but also the membrane time constants of SNNs. Literature [26] proposed a neuromorphic global-local synergic learning model. Compared with the single learning method, this method has much higher performance in few-shot learning. In addition, the authors in [27] proposed an SNN learning algorithm via proxy, which requires shorter simulation time than the converted SNNs. Previous SNN BP algorithms are more complex, but ANN BP algorithms are simple and effective.

The main contributions of our work are as follows: First, we propose the spike-based approximate backpropagation (SABP) algorithm for SNN training, whose approximate derivative of the spike neuronal activation function is simple and efficient. In addition, we have built general deep SNNs, which can adopt the popular architectures such as VGG [2] and ResNet [5] to build deep SNNs technologies and can also use SNN-based dropout to increase its generalization ability to alleviate the overfitting phenomenon in the learning process. We will then demonstrate the effectiveness of our work on MNIST, CIFAR-10, and sonar image target classification (SITC) datasets. To the best of our knowledge, the classification accuracy of our method is close to the best results on MNIST and CIFAR-10 datasets and achieves the best classification accuracy on the SITC dataset. Finally, we further analyze the advantages of this method compared with ANN in terms of computational complexity and energy consumption.

The structure of the paper is as follows: In Section 2, we will introduce our brain-inspired deep spiking neural network architecture and spike-based approximate backpropagation algorithm. In Section 3, the experiment is described and the experimental results are analyzed to demonstrate the effectiveness of the proposed method on MNIST, CIFAR-10, and SITC datasets. In Section 4, we discuss the relevant methods in recent years and compare them with ours. Finally, the thesis is summarized in Section 5.

2. Materials and Methods

2.1. Brain-Inspired Deep Spiking Neural Network Architecture. As shown in Figure 1, a deep SNN has been defined, including the input layer, encoder, hidden layer, output layer, and decoder. It is a feed-forward neural network, and the weight of each layer of neuron synapse will be updated according to the SNN BP algorithm described in Section 2.2. In this section, the structure and function of the encoder, the hidden layers, and the decoder will be mainly illustrated.

2.1.1. Encoder. As shown in Figure 2, in the encoder, the input image is two-dimensional static image data, the pixel image is encoded as a Poisson distributed spike train with a certain time step, and the probability of spike generation is proportional to the pixel intensity.

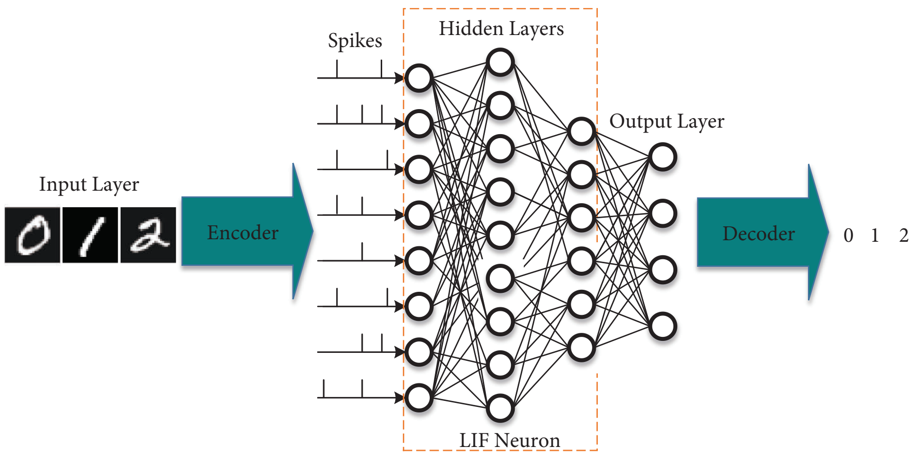

FIGURE 1: Overall architecture of brain-inspired SNNs. The encoder converts the input images into spikes, the hidden layers consist of LIF neurons, which can be any network structure (such as LeNet, VGG, and ResNet), and the decoder converts the output spikes into the corresponding classification results.

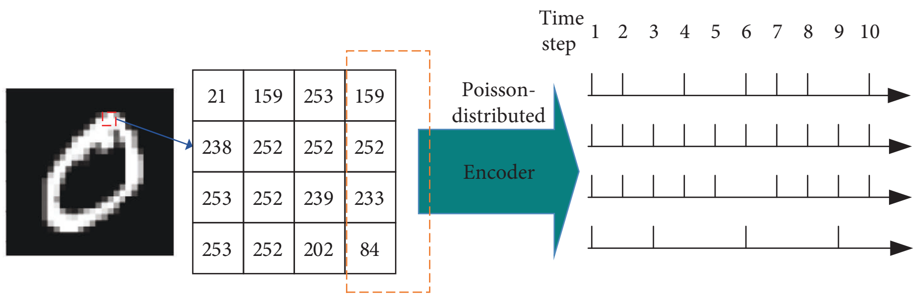

FIGURE 2: A simple example of the coding process. A 4×4 pixel image block is selected from the image, and the last column of the image block is transformed into the spike trains with a time step of 10, conforming to the Poisson distribution according to the pixel intensity.

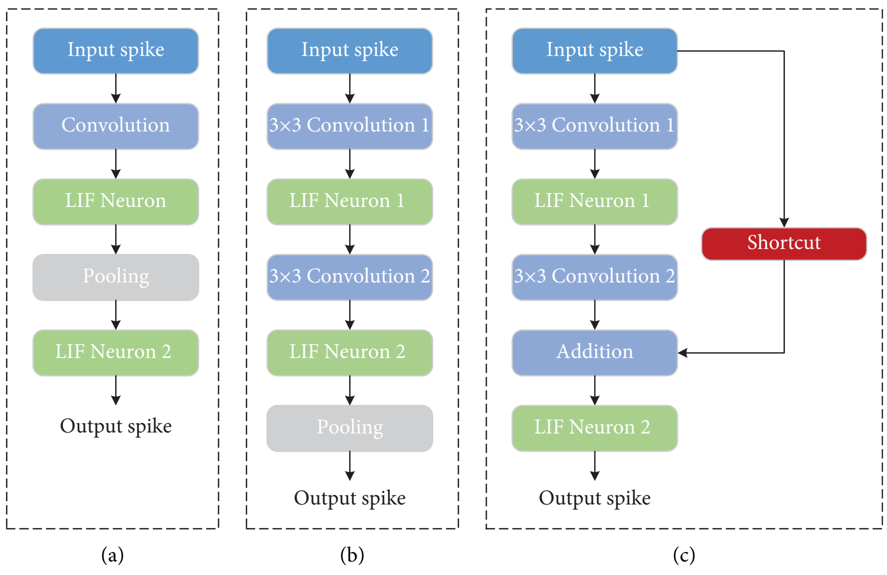

FIGURE 3: The basic architecture of the brain-inspired SNN blocks. (a) Spiking general structure. (b) Spiking VGG block. (c) Spiking ResNet block.

2.1.2. Hidden Layers. As shown in Figure 3, the hidden layers are composed of convolution layers, pooling layers, and fully connected (FC) layers. The features of spikes are extracted by the convolution layer and pooling layer, and then the one-dimensional vectors are generated in the FC layers and input to the decoder. The hidden layers are

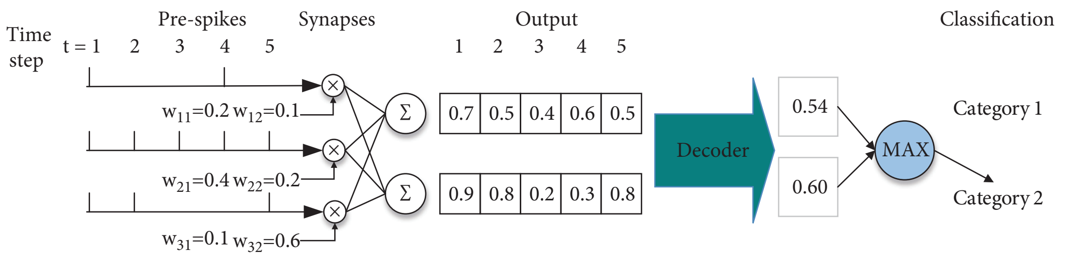

Figure 4: A simple example of the decoding process. In each time step, the value of the output layer is the result of multiplying the input spike train of the preneuron by the synaptic weight of the output layer. In the decoder, the values of the output layer in all time steps are accumulated and divided by the total time steps. Finally, the classification results are predicted by comparing the values of neurons in the decoder.

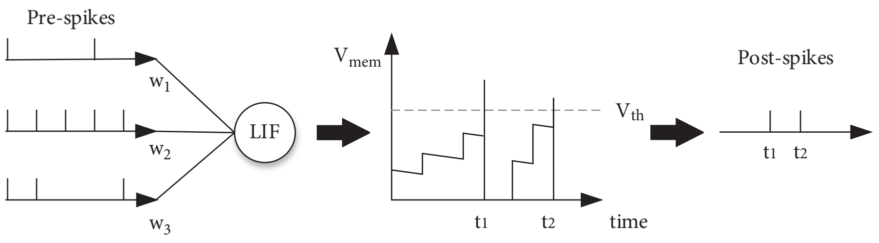

Figure 5: The process of leaky, integrate, and fire characteristics of LIF neurons. After integrating the current of the preneuron, the membrane potential of the postneuron begins to accumulate and decreases exponentially with time. Until the accumulation of the membrane potential exceeds the firing threshold, the LIF neuron fires a spike backward and resets the membrane potential.

composed of neurons based on the LIF model. The leaking and firing processes of synapses in the convolutional layer and the pooling layer will be discussed in detail in Section 2.2.

2.1.3. Decoder. As shown in Figure 4, the decoder accumulates the one-dimensional vector features extracted from the output layer for final classification. This accumulation is the sum of the output spike of each time step multiplied by the weight of the corresponding output layer. The number of neurons in this layer is the same as the number of categories to be classified.

2.2. Spike-Based Approximate Backpropagation Algorithm

2.2.1. Neuron Model. As shown in Figure 5, the leakyintegrate-–and-fire (LIF) neuron model greatly simplifies the action potential process and retains the three key characteristics (leaky, integrate, and fire) of the neuron membrane potential, and its formula can be expressed as follows:

$$\tau_m \frac{\mathrm{d}V_{mem}}{\mathrm{d}t} = V_{rest} - V_{mem} + R_m I(t), \tag{1}$$

where τ_m is the time constant of membrane potential decays, V_{mem} is the postneuron membrane potential, V_{rest} is the resting potential, R_m is the impedance of the membrane, and $I(t)$ is the input current.

2.2.2. Spike Forward Propagation. In forward propagation of spikes, the pixel value of the image is converted into a Poisson distributed spike train and transmitted to the network, and the input spike is multiplied by the synaptic weight to generate an input current. The input current accumulates in the membrane potential of the postneuron. When the membrane potential exceeds the firing threshold of the neuron, the postneuron generates an output spike and resets. If no spike is generated, the membrane potential decays (the membrane potential of the pooling layer neuron does not decay) exponentially over time. As shown in Figure 6, the neurons in each layer of the hidden layer will carry out this process in turn according to the input current received by the previous layer. As time goes by, the weighted sum of the spikes of the neurons can be formulated as follows:

$$net_j^{l+1}(t) = \sum_{i=1}^{n^l} w_{ij}^l \times x_i^l(t), \tag{2}$$

where $net_j^{l+1}(t)$ represents the total current inflow of the membrane potential accumulated in neurons j in the layer $l+1$ over time t, n^l represents the total number of neurons in the layer l, $w_{ij}^l(t)$ represents the weight of the connection synapse from the neuron i in the layer l to the neuron j in the layer $l+1$, and $x_i^l(t)$ is the sum of the spike events of neurons in the layer l over time t, which can be formulated as follows:

$$x_i^l(t) = \sum_{k=1}^{t} f_i^l(t-t_k), \tag{3}$$

where $f_i^l(t-t_k)$ represents the moment when the neuron in the layer l generates a spike at the time t_k, which can be expressed as follows:

$$f_i^l(t-t_k) = \begin{cases} 1, & \text{if fire,} \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

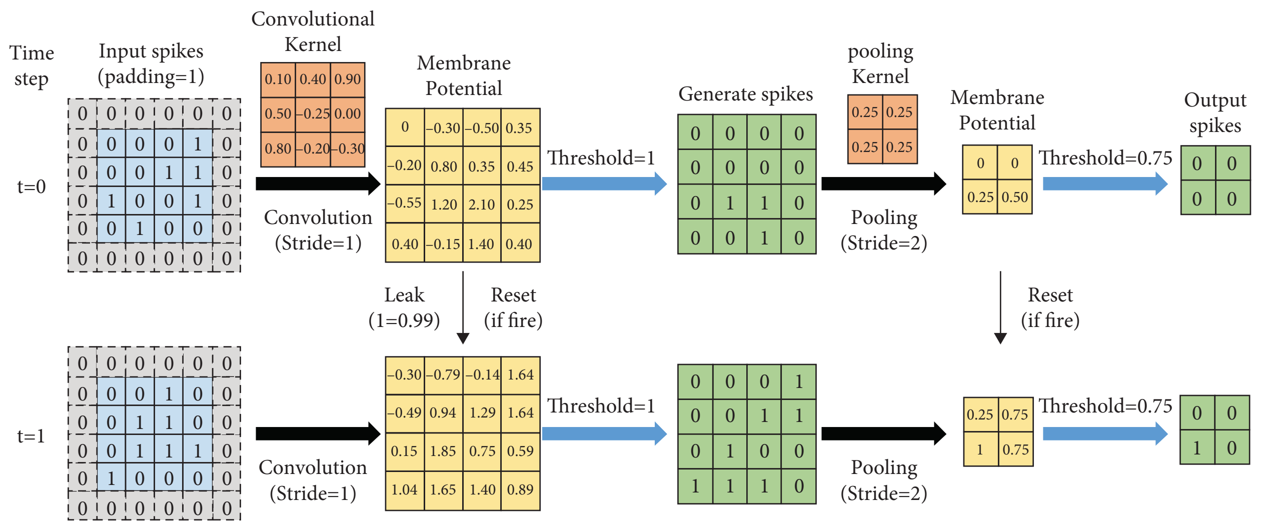

Figure 6: Illustration of a simplified operational example of the convolutional layer and the average pooling layer over two time steps. At each time step, when the membrane potential exceeds the firing threshold of the neuron, the postneuron generates an output spike and resets. If no spike is generated, the membrane potential will leak over time. In the average pooling layer, the membrane potential will not leak over time.

In time t, the sum of the spike trains generated by the neuron j in the layer $l+1$ can be formulated by $a_j^{l+1}(t)$ as follows:

$$a_j^{l+1}(t) = \sum_{k=1}^{t} f_j^{l+1}(t - t_k). \tag{5}$$

It can be seen that the sum of the spike trains produced by the neuron depends on the total amount of input current received. The neurons of the output layer will not generate spikes; instead, the membrane potential of the output layer will accumulate in the decoder at each time step, and the membrane potential of the decoder will not decay with time. At the last time step, the decoder will divide the accumulated membrane potential by the total time steps T to calculate the final result, which can be expressed as follows:

$$\text{result} = \frac{V_{mem}}{T}. \tag{6}$$

2.2.3. Error Backpropagation and Weight Update. After the forward propagation of a spike, the loss function is the difference between the predicted output of the decoder and the value of the label. At the decoder, the partial derivative of the loss function is calculated and propagated back to each previous layer using the chain rule, and the weight of each layer is updated according to the corresponding partial derivative obtained. The loss function can be expressed as follows:

$$L = \frac{1}{2}\sum_{j=1}^{n^l}\left(\text{prediction}_j - \text{label}_j\right)2, \tag{7}$$

where L represents the final loss and n represents the total number of neurons in the decoder. The leak characteristic of LIF neurons is taken as noise; then, the partial derivative of the difference with respect to the weight parameter of the hidden layer can be expressed by the following formula:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial net}\frac{\partial net}{\partial w}, \tag{8}$$

where w represents the corresponding weight to be updated, *net* represents the input current of the preneuron, and a represents the output current of the neuron after activation. Different from ANN, the activation function in SNN represents the relationship between the weighted summation of the preneuronal input and the postneuronal output over time. If the membrane potential does not exceed the threshold, the neuron output is 0, and if the membrane potential exceeds the threshold, the neuron output is 1. Similar to the ReLU function, here, we only differentiate positive values in the network. Since the activation function of SNNs is discontinuous and nondifferentiable, it is necessary to find an approximate derivative for $\partial a/\partial net$. If the membrane potential of the LIF neuron does not exceed the firing threshold, no spike will be generated, and the derivative of the neuron activation function is set to 0. If the neuronal membrane potential exceeds the threshold, a spike will be generated, and the membrane potential of the postneuron at a given time instant t can be expressed by the following formula:

$$V_{mem}(t) \approx \sum_{i=1}^{n}\left(w_i x_i(t)\right) - V_{th}\, a(t), \tag{9}$$

where n represents the number of preneurons, $x_i(t)$ represents the sum of the spikes of the pre-neuron i over time t, and $a(t)$ represents the sum of the spikes after the activation of the postneuron over time t. Since $V_{mem}(t)$ does not reach the firing threshold, $V_{mem}(t)$ will be ignored, and then, the derivative of the activation function can be approximated as a linear function; it can be directly estimated as $1/V_{th}$ [24], and the formula is deduced as follows:

$$a(t) \approx \frac{1}{V_{th}} \sum_{i=1}^{n} \left(w_i x_i(t)\right). \tag{10}$$

According to formula (2), it can be obtained as follows:

$$a(t) \approx \frac{1}{V_{th}} \operatorname{net}(t), \tag{11}$$

$$\frac{\partial a}{\partial net} = \frac{1}{Vth}. \tag{12}$$

Then, the partial derivative of the difference with respect to the weight can be approximated as follows:

$$\frac{\partial a}{\partial w} = \frac{1}{Vth} \frac{\partial L}{\partial a} \frac{\partial net}{\partial w}. \tag{13}$$

For simplicity, V_{th} is set to 1 in this paper so that the formula can be simplified as follows:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial a} \frac{\partial net}{\partial w}. \tag{14}$$

2.2.4. Dropout in the Spiking Neural Network. Dropout [3] is a very popular regularization technology for training ANN. In the training process, given a probability p that obeys the Bernoulli distribution, some neurons are randomly disconnected by p to avoid the occurrence of overfitting phenomenon.

In SNN, the use of dropout is slightly different from that of ANN. In the training process of ANN, data are divided into several batches, each iteration has only one forward propagation, and a number of connections between neurons and networks are disconnected randomly according to the probability p. However, in SNN, there will be multiple forward propagations in each iteration, which depends on the time step set. The error is back propagated, and the network parameters are updated only at the last time step. Therefore, the consistency of disconnection neurons in each time step must be guaranteed in an iteration. In ANN, the typical dropout probability p is set to 0.5. Since the activation of neurons is sparser in the forward propagation of SNN, the set of probability p in SNN is generally smaller than that of ANN. In our work, it is generally set to somewhere between 0.1 and 0.25.

3. Experiments and Results

3.1. Dataset Description. In order to measure the effectiveness of the SNN BP training algorithm in the brain-inspired SNN model, MNIST, CIFAR-10, and SITC datasets are selected for experiments. The MNIST handwritten dataset is composed of grayscale images with an image size of 28 × 28. There are 60,000 training samples and 10,000 test samples, including 10 digital categories of 0–9. CIFAR-10 is composed of color images with an image size of 32 × 32. There are 50,000 training samples and 10,000 test samples, including 10 categories of animals and vehicles. SITC is a small-sample sonar image dataset. The SITC dataset is based on 62 aircraft images and 385 shipwreck images publicly provided by the SeabedObjects-KLSG [28] dataset, and 289 seabed images and 18 drowning images are collected and combined into an experimental dataset for underwater target classification. All target images are directly cropped from the original sonar image. A total of 18 drowning images, 62 aircraft images, 289 seabed images, and 385 shipwreck images are included.

Table 1: Experiments datasets.

Dataset	Image	Train sets	Test sets	Category
MNIST	28 × 28 gray	60000	10000	10
CIFAR-10	32 × 32 × 3 color	50000	10000	10
SITC	Unfixed gray	529	225	4

Table 1 lists the detailed information about the experiments' datasets. The training samples of the three datasets are used for deep SNN training, and then, the trained SNN is used to predict and classify the test samples of the dataset to obtain classification accuracy. There is no intersection between training samples and test samples.

3.2. Experimental Setup. A custom simulation the SNN framework is developed using the PyTorch deep learning framework, which is available in Python 3.6.12 and PyTorch 1.1.0. For training, as described in Subsection 2.2.3, synaptic weights are trained with a mini-batch approximate BP algorithm in an end-to-end manner. A stochastic gradient descent (SGD) optimization algorithm is used to optimize the training parameters, and each experiment has 150 training epochs.

3.3. Network Topologies. As described in Section 2.1, the hidden layer of SNN can contain the current popular network architecture. The appropriate SNN architecture is selected according to the complexity of the experimental dataset. For the convenience of comparison, a network architecture similar to the LeNet5 model is used on the MNIST and SITC datasets, which contain two sets of convolutional layer pooling layers and two FC layers. For the CIFAR-10 dataset, a deeper network architecture is chosen, with a structure similar to ResNet11. The hidden layer is composed of residual blocks, and there are 11 layers of trainable parameters. The detailed structure is shown in Table 2.

3.4. Encoding Scheme. For the MNIST dataset, the pixel value of the grayscale image is scaled between 0 and 1 and then transformed into a spike event stream with a certain number of time steps conforming to the Poisson distribution according to the pixel intensity. For the CIFA-10 dataset, we first preprocessed the color image by horizontal flipping, scaling the pixel value between −1 and 1 and then transformed it into a spike event stream with a certain number of time steps in line with the Poisson distribution according to the pixel intensity. For the SITC dataset, the pixel value of the gray image is scaled to between 0 and 1 and then transformed into a spike event stream with a certain number of time steps conforming to the Poisson distribution according to the pixel intensity.

Table 2: LeNet5 and ResNet11 network architecture.

LeNet5				ResNet11			
Layer	Kernel	Channel	Stride	Layer	Kernel	Channel	Stride
Conv	$1 \times 5 \times 5$	20	1	Conv	$3 \times 3 \times 3$	64	1
Avgpool	2×2	20	2	Avgpool	2×2	20	2
Conv	$20 \times 5 \times 5$	50	1	Conv	$64 \times 3 \times 3$	128	1
Avgpool	2×2	50	2	Conv	$128 \times 3 \times 3$	128	1
				Shortcut	$64 \times 1 \times 1$	128	1
				Conv	$128 \times 3 \times 3$	256	1
				Conv	$256 \times 3 \times 3$	256	2
				Shortcut	$128 \times 1 \times 1$	256	2
				Conv	$256 \times 3 \times 3$	512	1
				Conv	$512 \times 3 \times 3$	512	1
				Shortcut	$256 \times 1 \times 1$	512	1
				Conv	$512 \times 3 \times 3$	512	1
				Conv	$512 \times 3 \times 3$	512	2
				Shortcut	$512 \times 1 \times 1$	512	2
FC		200		FC		1024	
Output		10		Output		10	

Table 3: The classification accuracy of SNN on the MNIST dataset.

Author	Method	Accuracy (%)
Diehl and Cook [19]	STDP	95.00
Kheradpisheh et al. [20]	STDP	98.40
Hunsberger and Eliasmith [29]	Conversion	98.37
Diehl et al. [30]	Conversion	99.10
Rueckauer et al. [31]	Conversion	99.44
Lee et al. [32]	Spike-based BP	99.31
Jin et al. [21]	HM2-BP	99.49
Wu et al. [22]	Spike-based BP	99.42
Lee et al. [33]	STDP + spike-based BP	99.28
Lee et al. [24]	Spike-based BP	99.59
Fang et al. [25]	Surrogate gradient	99.72
Wu et al. [26]	Spike-based HP	99.50
This work	**SABP**	**99.62**

In SNN, the spike event for each time step can only be 0 or 1, and the time steps provide additional time dimension information; they can be considered as the actual precision of neuronal activation. If the time steps are too less, SNN will not be able to get enough information for inference. If the time steps are too long, the randomness of SNN will be destroyed, and the additional inference time and energy consumption will be generated. In our experiments, the time steps of the relatively simple MNIST dataset are 50, the time steps of the more complex CIFAR-10 dataset are 100, and the time steps of the SITC dataset are 70.

3.5. Experimental Results. At present, MNIST and CIFAR-10 are mostly used to verify SNN inference ability. Tables 3–5, respectively, list the classification results of SNN on MNIST, CIFAR-10, and SITC datasets in recent years, and the results of our algorithm SABP are bold in thetable.

As shown in Tables 3–5, the accuracy of the SABP algorithm is 99.62%, 91.03%, and 91.11%, respectively. For MNIST and CIFAR-10 datasets, the SABP algorithm is very close to the highest accuracy of surrogate gradient [25] (Tables 3 and 4). For the SITC dataset of the small sample, after image enhancement by style transfer and weighted random sampling, the classification accuracy is significantly higher than that of surrogate gradient [25] (Table 5).

4. Discussion

We propose the general brain-inspired SNN framework and the SABP algorithm. The SABP algorithm derives the approximate derivative of an LIF neuron by the relationship between the input current of the LIF neuron and the output of the neuron after activation, and in this process, we treat the leak characteristics of LIF neurons as noise. In the weight update stage, we use the approximate derivative to realize the error backpropagation. The approximate derivative of the SABP algorithm is very concise because we believe that an SNN training algorithm must be concise and effective.

4.1. Comparison with Relevant Works. Unlike the HM2-BP algorithm proposed by Jin et al. [21] and the STBP algorithm by Wu et al. [22], our work is similar to that of the approximate derivative algorithm of IF neurons by Lee et al. [24], which considers only the activation of spikes to naturally train the deep brain-inspired SNN. In addition, in the process of training, the method by Lee et al. in each neuron is to save a complex leakage compensation value for error backpropagation. However, our LIF neuron activation function approximate derivative is very simple (active 1 and inactive 0) and without additional storage required, which saves the memory and improves the training speed. Therefore, the work in this paper can advance the spike-based BP algorithm for training deep SNN.

4.2. Calculation of Energy Consumption. Different from ANN, which requires multiply-accumulate computation (MAC) at each layer, the spikes of SNN are sparse and only

Table 4: The classification accuracy of SNN on the CIFAR-10 dataset.

Author	Method	Accuracy (%)
Hunsberger and Eliasmith [29]	Conversion	82.95
Esser et al. [34]	Conversion	89.32
Rueckauer et al. [31]	Conversion	88.82
Sengupta et al. [16]	Conversion	91.55
Rathi et al. [17]	Conversion + STDB	92.22
Stöckl and Maass [18]	FS-conversion	92.42
Wu et al. [23]	Spike-based BP	90.53
Lee et al. [24]	Spike-based BP	90.95
Fang et al. [25]	Surrogate gradient	93.50%
Saeed Kheradpisheh and Maryam [27]	Proxy	93.11%
Wu et al. [26]	Spike-based HP	91.08%
This work	SABP	91.03

Table 5: The classification accuracy of SNN on the SITC dataset.

Author	Method	Accuracy (%)
Fang et al. [25]	Surrogate gradient	86.67%
This work	SABP	91.11

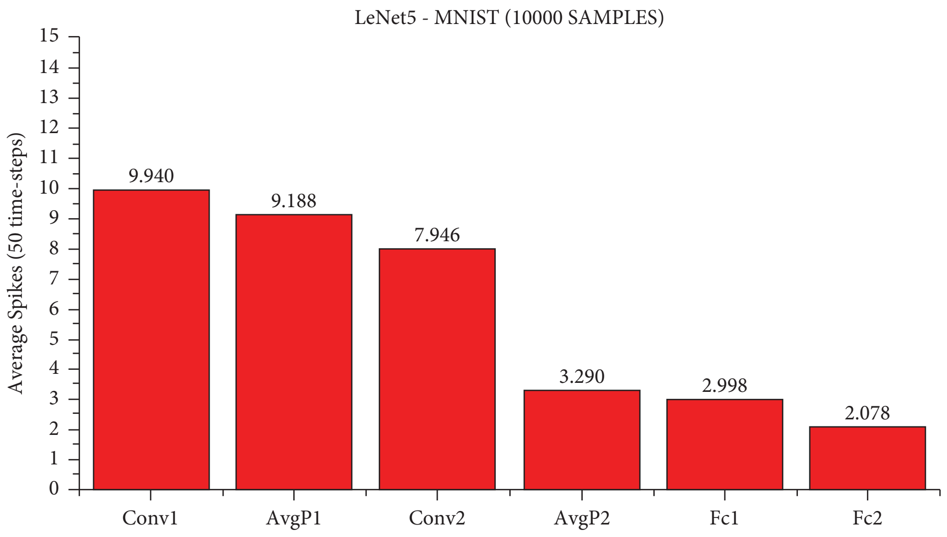

Figure 7: The average number of spike activations at each layer of LeNet5. On all test sets of MNIST samples, when using 50 time steps, the accuracy of this method is 99.62%, and the average number of spikes activated per layer under the LeNet5 network architecture is obtained.

have accumulate computation (AC) at each layer, which enables SNN to reduce computational complexity and save energy consumption. If both SNN and ANN use conventional hardware, this advantage will be lost, and even the computational complexity will be tens to hundreds of times that of ANN. Fortunately, special hardware is available to implement event-based operations, and SNN can take advantage of this mechanism to compare with ANN.

At present, although it is difficult for us to directly estimate the actual energy consumption of SNN and ANN, we can still make a rough estimate of the synaptic operands of each network layer to compare the computational complexity of SNN and ANN. Suppose an SNN has l layers, each layer has n_l synaptic connections, the total time steps are T_s, and the average activation rate of synapses is a_l, then the AC operand of an SNN is $\sum_l n_l \times a_l \times T_s$, and $\sum_l n_l \times a_l$ is the MAC operand of an ANN of the same structure. The number of spike activations at each layer of LeNet5 and ResNet11 is shown in Figures 7–9.

We use the calculation method described above, under the LeNet5 architecture, the AC operands of SNN are about 7.4x of the MAC operand of ANN. Under the ResNet11 architecture, the AC operands of SNN are about 2.2x as many as the MAC operands of ANN. However, according to the work by Han et al. [35], the energy of MAC operation is usually one order of magnitude higher than that of AC operation. For example, in the 45 nm technology node, the energy consumed by 32-bit floating-point MAC operation is about 4.6PJ and that of AC operation is about 0.9PJ. The power consumption for 32-bit integer MAC operation is

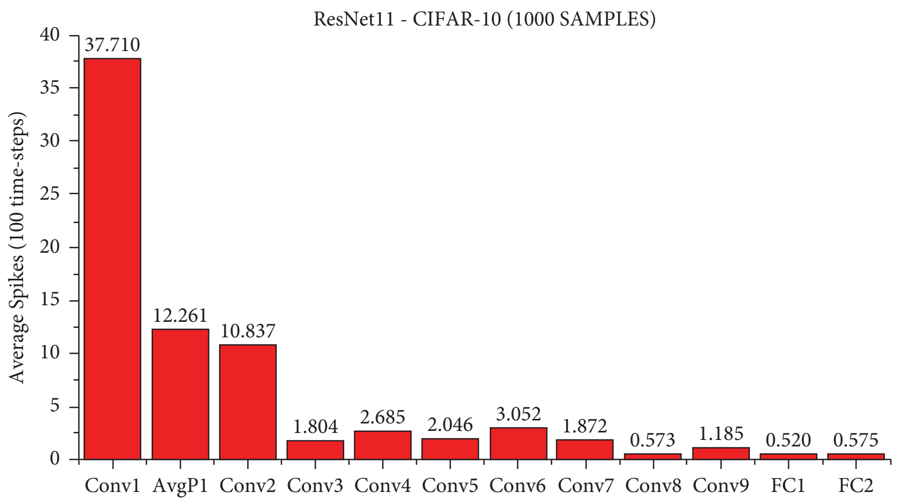

Figure 8: The average number of spike activations at each layer of ResNet11. When using 100 time steps, 1000 samples are randomly selected from the test samples of the CIFAR-10 dataset, and the accuracy of this method is 92.00%; the average number of spikes activated per layer under the ResNet11 network architecture is obtained.

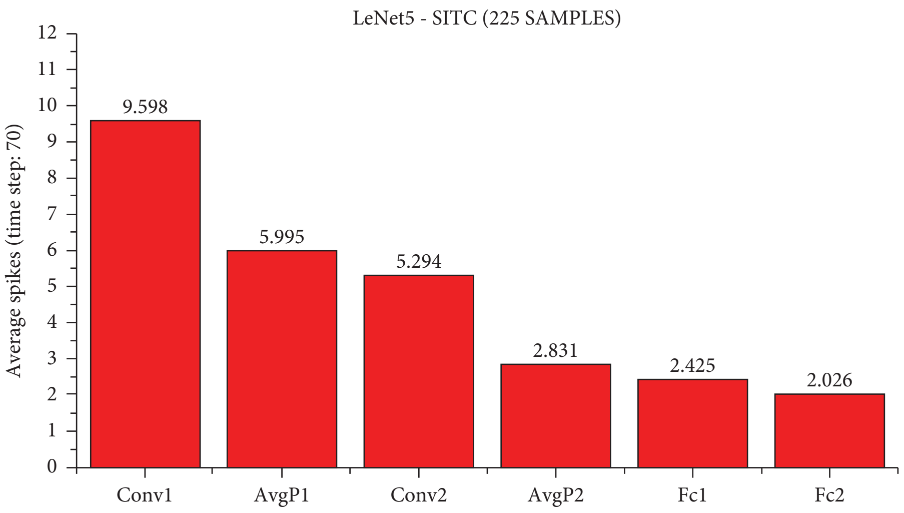

Figure 9: On all test sets of MNIST samples, when using 70 time steps, the accuracy of this method is 91.11%, and the average number of spikes activated per layer under the LeNet5 network architecture is obtained.

about 3.2 PJ, and for AC operation, it is about 0.1 PJ. It has been reported [36] that 32-bit floating-point computing can be replaced by fixed-point computing using integer MAC and AC units with almost no loss. According to the calculation method mentioned above, for the MNIST dataset, the calculation energy efficiency of SNN is about 4.32x that of ANN, for the CIFAR-10 dataset, the computational energy efficiency of SNN is about 14.55x that of ANN, and for the SITC dataset, the computational energy efficiency of SNN is about 6.2x that of ANN. It is worth noting that the average activation rate of neurons in each layer decreases with the deepening of the SNN network, and the activated neurons become more and more sparse. Although the actual energy consumption of SNN and ANN is not completely consistent with the synaptic operand and the actual energy consumption may be affected by other factors, the advantages of brain-inspired SNN in computational complexity and energy consumption can still be observed.

5. Conclusion

Aiming at the problem that CNN is difficult to deploy on edge smart devices due to high energy consumption, this paper proposes a general brain-inspired SNN architecture and a simple and effective SABP algorithm and constructs different network architectures according to the characteristics of different datasets. Experiments show that the method has a good performance on brain-inspired deep SNN. Energy consumption analysis proves the superiority of SNN in energy consumption, which is more suitable for deploying edge smart devices than CNN. So far, compared with other SNN training methods, the accuracy of our experiment is very close to the highest accuracy on MNIST and CIFAR-10 datasets, and the highest accuracy is achieved on small-sample SITC datasets. When applied to suitable neural mimicry hardware, our proposed approach can significantly reduce computational complexity and energy consumption for sonar target classification in unmanned underwater vehicles.

In the future, we will strive to reduce the time step required for SNN classification to reduce the delay and deploy the algorithm on an unmanned underwater vehicle equipped with neuromorphic hardware for further research.

Data Availability

The datasets used to support the findings of this study are available at MNIST DATASET (http://yann.lecun.com/exdb/mnist/) CIFAR10 DATASET (https://tensorflow.google.cn/datasets/catalog/cifar10).

Conflicts of Interest

The authors declare that they have no conflicts of interests.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Nos. 62176087, 62176088, 62106066), the Shenzhen Special Foundation of Central Government to Guide Local Science and Technology Development (Nos. 2021Szvup032, 2021Szvup029), the Major Project of Science and Technology of Henan Province (No. 201400210300), the Postgraduate Education Reform and Quality Province Improvement Project of Henan Province (No. YJS2022JC33), and the Education Reform Research and Practice Project of Henan University (No. HDXJJG2020-109).

References

[1] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, no. 2, pp. 1097–1105, 2012.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, https://arxiv.org/abs/1409.1556.

[3] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[4] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," 2015, https://arxiv.org/abs/1502.03167.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, June 2016.

[6] L. Deng, Y. Wu, X. Hu et al., "Rethinking the performance comparison between SNNS and ANNS," *Neural Networks*, vol. 121, pp. 294–307, 2020.

[7] E. Painkras, L. A. Plana, J. Garside et al., "SpiNNaker: a 1-W 18-core system-on-chip for massively-parallel neural network simulation," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 8, pp. 1943–1953, 2013.

[8] B. V. Benjamin, P. Gao, E. Mcquinn et al., "Neurogrid: a mixed-analog-digital multichip system for large-scale neural simulations," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 699–716, 2014.

[9] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The SpiNNaker Project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, 2014.

[10] F. Akopyan, J. Sawada, A. Cassidy et al., "TrueNorth: design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537–1557, 2015.

[11] M. Davies, N. Srinivasa, T. H. Lin et al., "Loihi: a neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[12] S. M. Bohte, J. N. Kok, and H. La Poutré, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, vol. 48, no. 1-4, pp. 17–37, 2002.

[13] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[14] A. L. Hodgkin and A. 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.

[15] V. Kornijcuk, H. Lim, J. Y. Seok et al., "Leaky integrate-and-fire neuron circuit based on floating-gate integrator," *Frontiers in Neuroscience*, vol. 10, no. 212, p. 212, 2016.

[16] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: VGG and residual architectures," *Frontiers in Neuroscience*, vol. 13, no. 95, p. 95, 2019.

[17] N. Rathi, G. Srinivasan, P. Panda, and K. Roy, "Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation," 2020, https://arxiv.org/abs/2005.01807.

[18] C. Stockl and W. Maass, "Optimized spiking neurons can classify images with high accuracy through temporal coding with two spikes," *Nature Machine Intelligence*, vol. 3, no. 3, pp. 230–238, 2021.

[19] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in Computational Neuroscience*, vol. 9, no. 99, 2015.

[20] S. R. Kheradpisheh, M. Ganjtabesh, and T. Masquelier, "Bio-inspired unsupervised learning of visual features leads to robust invariant object recognition," *Neurocomputing*, vol. 205, pp. 382–392, 2016.

[21] Y. Jin, P. Li, and W. Zhang, "Hybrid macro/micro level backpropagation for training deep spiking neural networks," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 7005–7015, January 2019.

[22] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *Frontiers in Neuroscience*, vol. 12, no. 331, p. 331, 2018.

[23] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, and L. Shi, "Direct training for spiking neural networks: faster, larger, better," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 1311–1318, 2018.

[24] C. Lee, S. S. Sarwar, P. Panda, G. Srinivasan, and K. Roy, "Enabling spike-based backpropagation for training deep

neural network architectures," *Frontiers in Neuroscience*, vol. 14, p. 119, 2020.

[25] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, and Y. Tian, "Incorporating Learnable Membrane Time Constant to Enhance Learning of Spiking Neural Networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2641–2651, May 2021.

[26] Y. Wu, R. Zhao, J. Zhu et al., "Brain-inspired global-local learning incorporated with neuromorphic computing," *Nature Communications*, vol. 13, no. 1, p. 65, 2022.

[27] R. Saeed Kheradpisheh and M. T. Maryam, "Spiking neural networks trained via proxy," 2021, https://arxiv.org/abs/2109.13208#:~:text=We%20propose%20a%20new%20learning,architectures%20and%20shared%20synaptic%20weights.

[28] G. Huo, Z. Wu, and J. Li, "Underwater object classification in sidescan sonar images using deep transfer learning and semisynthetic training data," *IEEE Access*, vol. 8, pp. 47407–47418, 2020.

[29] E. Hunsberger and C. Eliasmith, "Spiking deep networks with LIF neurons," 2015, https://arxiv.org/abs/1510.08829.

[30] P. U. Diehl, G. Zarrella, A. Cassidy, B. U. Pedroni, and E. Neftci, "Conversion of Artificial Recurrent Neural Networks to Spiking Neural Networks for Low-Power Neuromorphic Hardware," in *Proceedings of the IEEE International Conference on Rebooting Computing (ICRC)*, pp. 1–8, Year.-Bedford, MA, USA, July 2016.

[31] 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," *Frontiers in Neuroscience*, vol. 11, p. 682, 2017.

[32] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers in Neuroscience*, vol. 10, no. 508, p. 508, 2016.

[33] C. Lee, P. Panda, G. Srinivasan, and K. Roy, "Training deep spiking convolutional neural networks with STDP-based unsupervised pre-training followed by supervised fine-tuning," *Frontiers in Neuroscience*, vol. 12, p. 435, 2018.

[34] S. K. Esser, P. A. Merolla, J. V. Arthur et al., "Convolutional networks for fast, energy-efficient neuromorphic computing," *Proceedings of the National Academy of Sciences*, vol. 113, no. 41, p. 11441, 2016.

[35] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural network," in *Proceedings of the Advances in Neural Information Processing Systems (Montréal, QC)*, pp. 1135–1143, Cambridge, MA, USA, December 2015.

[36] D. D. Lin, S. S. Talathi, and V. Sreekanth Annapureddy, "Fixed point quantization of deep convolutional networks," in *Proceedings of the International Conference on Machine Learning*, pp. 2849–2858, June 2016.