frontiers | Frontiers in Neuroscience

TYPE Original Research
PUBLISHED 12 October 2022
DOI 10.3389/fnins.2022.991851

OPEN ACCESS

EDITED BY
Jie Yang,
Shanghai Jiao Tong University, China

REVIEWED BY
Fangxin Liu,
Shanghai Jiao Tong University, China
Youngeun Kim,
Yale University, United States

*CORRESPONDENCE
Yi Zeng
yi.zeng@ia.ac.cn

SPECIALTY SECTION
This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience

RECEIVED 12 July 2022
ACCEPTED 26 September 2022
PUBLISHED 12 October 2022

CITATION
Li Y, Zhao D and Zeng Y (2022) BSNN: Towards faster and better conversion of artificial neural networks to spiking neural networks with bistable neurons. *Front. Neurosci.* 16:991851. doi: 10.3389/fnins.2022.991851

BSNN: Towards faster and better conversion of artificial neural networks to spiking neural networks with bistable neurons

Yang Li[1,2], Dongcheng Zhao[1] and Yi Zeng[1,2,3,4]*

[1]Research Center for Brain-Inspired Intelligence, Institute of Automation, Chinese Academy of Sciences, Beijing, China, [2]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China, [3]Center for Excellence in Brain Science and Intelligence Technology, Chinese Academy of Sciences, Shanghai, China, [4]National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China

The spiking neural network (SNN) computes and communicates information through discrete binary events. Recent work has achieved essential progress on an excellent performance by converting ANN to SNN. Due to the difference in information processing, the converted deep SNN usually suffers serious performance loss and large time delay. In this paper, we analyze the reasons for the performance loss and propose a novel bistable spiking neural network (BSNN) that addresses the problem of the phase lead and phase lag. Also, we design synchronous neurons (SN) to help efficiently improve performance when ResNet structure-based ANNs are converted. BSNN significantly improves the performance of the converted SNN by enabling more accurate delivery of information to the next layer after one cycle. Experimental results show that the proposed method only needs 1/4–1/10 of the time steps compared to previous work to achieve nearly lossless conversion. We demonstrate better ANN-SNN conversion for VGG16, ResNet20, and ResNet34 on challenging datasets including CIFAR-10 (95.16% top-1), CIFAR-100 (78.12% top-1), and ImageNet (72.64% top-1).

KEYWORDS

spiking neural network, bistability, neuromorphic computing, image classification, conversion

1. Introduction

Deep learning (or Deep Neural Network, DNN) has made breakthroughs in many fields such as computer vision (Girshick, 2015; Liu et al., 2016; Redmon et al., 2016), natural language processing (Bahdanau et al., 2014; Devlin et al., 2018), and speech processing (Park et al., 2020), and has even surpassed humans in some specific fields. But many difficulties and challenges also need to be overcome in the development process of deep learning (Lake et al., 2015; Nguyen et al., 2015; Kemker et al., 2018; Yan et al., 2019). One concerning issue is that researchers pay more attention to higher computing power and better performance while ignoring the cost of energy consumption (Strubell et al., 2019). Taking natural language processing tasks as an example, the power consumption and carbon emissions of Transformer (Vaswani et al., 2017) model training

are very considerable. In recent years, the cost advantages and environmental advantages of low-energy AI have attracted the attention of researchers. They design compression algorithms (Wu et al., 2016; He and Cheng, 2018) to enable artificial neural networks (ANN) to significantly reduce network parameters and calculations while maintaining their original performance. Another part of the work focuses on computing architecture (Chen et al., 2014), less computational energy consumption can be achieved by designing hardware that is more suitable for the operational characteristics of neural network models. But the problem of the high computational complexity of deep neural networks still exists. Therefore, the spiking neural network, known as the third-generation artificial neural network (Maass, 1997), has received more and more attention (Bing et al., 2018; Illing et al., 2019; Jang et al., 2019; Tavanaei et al., 2019; Wang et al., 2020).

Spike neural networks (SNNs) process discrete spike signals through the dynamic characteristics of spiking neurons, rather than real values, and are considered to be more biologically plausible and more energy-efficient (Pfeiffer and Pfeil, 2018; Roy et al., 2019; Lobo et al., 2020). For the former, the event-type information transmitted by neurons in SNN is the spike, which is generated when the membrane potential reaches the neuron firing threshold. Thus, its information processing process is more in line with biological reality than traditional artificial neurons (Zhang et al., 2018; Fang H. et al., 2021; Liang and Zeng, 2021). For the latter, the information in SNN is based on the event, i.e., neurons that do not emit spikes do not participate in calculations, and the information integration of neurons is an accumulate (AC) operation, which is more energy-efficient than the multiply-accumulate (MAC) operations in ANN (Marian et al., 2002; Zhao et al., 2014). Therefore, researchers put forward the concept of neuromorphic computing (Burr et al., 2017; Davies, 2019; Song et al., 2020), which realizes the more biologically plausible SNN on hardware. It shows more significant progress in fast information processing and energy saving. But due to the non-differentiable characteristics of SNN, training SNN is still a challenging task. Because of the lack of the derivative of the output, the common backpropagation algorithm cannot be used directly. How to use SNN for effective reference has become a problem for researchers.

Taking inspiration from the brain, such as Spike-Timing Dependent Plasticity (STDP) (Bengio et al., 2015; Liu et al., 2021), lateral inhibition (Blakemore et al., 1970; Abbott and Nelson, 2000), Long-Term Potentiation (LTP) (Malenka, 2003), and Long-Term Depression (LTD) (Ito, 1989) are effective methods. By properly integrating different neural mechanisms in the brain (Zeng et al., 2017), SNN can be effectively trained. Because most of these methods are unsupervised, researchers often add SVM (Noble, 2006) or other classifiers for supervised learning (Hao et al., 2020; Wang et al., 2020) or directly do learning in an unsupervised manner (Diehl and Cook, 2015; Illing et al., 2019). All of them are of great importance for further enhancing the interpretability of SNN and exploring the working mechanism of the human brain. However, this optimization method that only uses local neural activities is challenging to achieve high performance and be applied to complex tasks. Some researchers try to train SNNs through approximated gradient algorithms (Fang et al., 2021a,b; Wu et al., 2021; Meng et al., 2022), where the backpropagation algorithm can be applied to the SNN by continuous the spike firing process of the neuron. However, this method suffers from difficulty in convergence and requires a lot of time in training procedure in the deep neural networks (DNN) because it is difficult to balance the whole firing rate. For the above two methods, they perform poorly in large networks and complex tasks and require a large amount of computing resources and memories. We believe that the inability to obtain an SNN with effective reference ability is a key issue in the development and application of SNN.

Recently, the conversion method has been proposed to convert the training result of ANN to SNN (Cao et al., 2015). The ANN-SNN conversion method maps the trained ANN parameters with ReLU activation function to SNN with the same topology as illustrated in Figure 1, which makes it possible for SNN to obtain extremely high performance at a very low computational cost. But direct mapping will lead to severe performance degradation (Yang et al., 2020). Diehl et al. (2015) propose the data-based normalization method, which scales the parameters with the maximum activation values of each layer in ANN, improving the performance of the converted SNN. Rueckauer et al. (2017) and Han et al. (2020) use integrate-and-fire (IF) neurons with soft reset to make SNN achieve performance comparable to ANN. Nonetheless, it usually takes more than 1,000–4,000 time steps to achieve better performance on complex datasets. And when converting ResNet (He et al., 2016) to SNN, researchers suffer from a certain performance loss (Hu et al., 2018; Sengupta et al., 2019; Xing et al., 2019) because the information received by the output neuron of the residual block is incomplete with the spikes on the shortcut path arriving earlier.

Bistability is a special activity form in biological neurons (Izhikevich, 2003). Neurons can switch between spike and non-spike states under the action of neuromodulating substances, thus exhibiting short-term memory function (Marder et al., 1996). Inspired from the bistability characteristic, we focus on improving the performance of SNN and propose a bistable spiking neural network (BSNN), which combines phase coding and the bistability mechanism that greatly improves the performance after conversion and reduces the time delay. For high-performance spiking ResNet, we propose synchronous neurons (SN), which can help spikes in the residual block synchronously reach the output neurons from input neurons through two paths. The information in BSNN takes one cycle to pass from one layer to another. Thus, the time steps required to achieve optimal performance in BSNN are

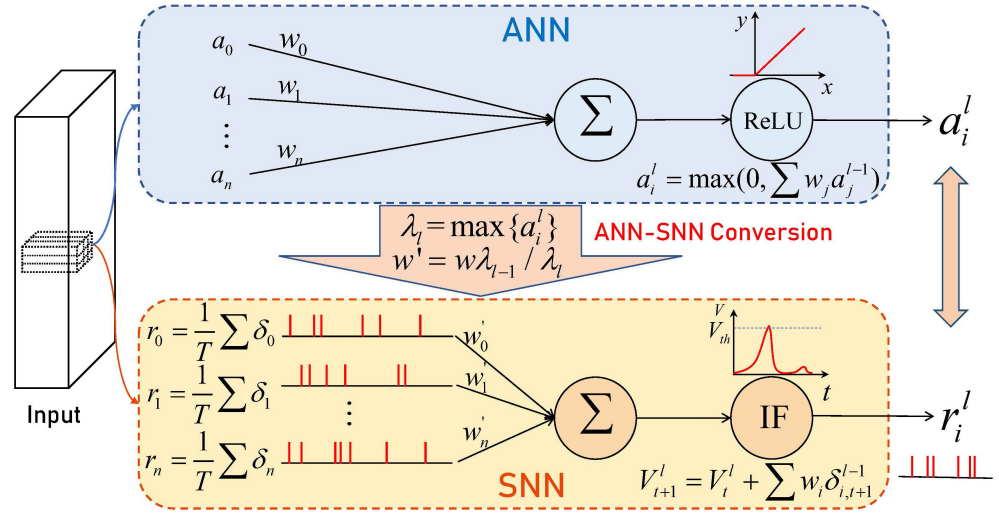

FIGURE 1
Illustration of ANN-SNN conversion.

significantly reduced compared to the methods of increasing the accuracy by continuously increasing the simulation time. The experimental results demonstrate they can help achieve nearly lossless conversion and state-of-the-art in MNIST, CIFAR-10, CIFAR-100, and ImageNet while significantly reduce time delay. Our contributions can be summarized as follows:

- We propose a novel BSNN that combines phase coding and bistability mechanism. It effectively solves the problem of SIN and greatly reduces the performance loss and time delay of the converted SNN.
- We propose synchronous neurons to solve the problem that information in the spiking ResNet cannot synchronously reach the output neurons from two paths.
- We achieve better performance on the MNIST, CIFAR-10, CIFAR-100, and ImageNet datasets, verifying the effectiveness of the proposed method.

2. Related work

Many conversion methods have been proposed in order to obtain high-performance SNN. According to the encoding method they can be divided into three kinds.

2.1. Temporal coding based conversion

Temporal coding uses neural firing time to encode the input to spike trains and approximate activations in ANN (Rueckauer and Liu, 2018). However, since neurons in the hidden layer need to accumulate membrane potential to spike, when the activation value is equal to the maximum, neurons in deep layers are difficult to spike immediately, making this method difficult to convert deep ANNs. Zhang et al. (2019) use ticking neurons to modify the method above, which transfers information layer by layer. Nevertheless, this method is less robust and difficult to be used in models with complex network structures like the residual block.

2.2. Rate coding based conversion

Unlike temporal coding, the rate coding-based conversion method uses the firing rates of spiking neurons to approximate the activation values in the ANN (Cao et al., 2015). Diehl et al. (2015) propose data-based and model-based normalization, which use the maximum activation value of neurons in each layer to normalize the weights. When disturbed by noise, the normalization parameter may be quite large, which will cause the weight smaller and the time to spike longer. Researchers propose to use the p-th largest value for normalization operation, thereby greatly improving robustness and reducing time delay (Rueckauer et al., 2017). Therefore, the conversion method based on rate coding has achieved better performance in ResNet (Hu et al., 2018) and Inception Networks (Sengupta et al., 2019; Xing et al., 2019). However, the processing speed of spikes on the paths with different processing units is different. The information received by the output neuron is delayed to various degrees when spreading on these wider networks. The difference between the firing rate and the activation value in the ANN will be greater. Therefore, the performance loss and the time delay of the SNN is more significant when converting these ANNs.

2.3. Phase coding based conversion

To overcome the large time delay of the converted SNN, researchers propose SNN with weighted spike, which assigns different weights to the spikes in different phases to pack more information in the spike (Kim et al., 2018). Nonetheless, when neurons do not spike in the expected phase, the spikes of neurons in hidden layers will deviate from the coding rules to a certain extent, resulting in poor performance on complex datasets and large networks. Phase coding and burst coding are combined to speed up the information transmission (Park et al., 2019), but still needs 3,000 simulation time on CIFAR-100 dataset.

3. Methods

In this section, we introduce the spiking neurons and encoding methods in detail, and then analyze the reasons for the loss of conversion performance based on the process of phase coding conversion methods. The detailed information of the model to reduce conversion loss and time delay is described. And we will introduce the effect of synchronized neurons in spiking ResNet.

3.1. Spiking neuron and encoding

The most commonly used spiking neuron model is the integrate-and-fire (IF) model. The IF neuron continuously receives spikes from the presynaptic neuron and dynamically changes its membrane potential. When it exceeds the threshold, the neuron spikes and the membrane potential is traditionally reset to zero. But it will cause a lot of information loss. We follow (Rueckauer et al., 2017) and use the soft reset to subtract the threshold from the membrane potential:

$$V_{i,t}^{l} = V_{i,t-1}^{l} + \sum_{j} w_{ij}\delta_{j,t}^{l-1}, \tag{1}$$

$$if \quad V_{i,t}^{l} \geq V_{th}, \quad \begin{cases} V_{i,t}^{l} = V_{i,t}^{l} - V_{th}, \\ \delta_{i,t}^{l} = 1. \end{cases} \tag{2}$$

where $V_{i,t}^{l}$ represents the membrane potential of neuron i in layer l at time t, w_{ij} is the weight connecting the neuron j and i, $\delta_{j,t}^{l-1}$ is the spike of neuron j in layer $(l-1)$ at time t.

The spike trains can be encoded by real values with different encoding methods. The real value is equal to the firing rate in rate coding, which is the number of spikes in a period, or the ratio of the difference between the total simulation time T and the spike time to T in temporal coding, which is:

$$a_{rate} = \frac{N}{T}, \quad a_{temporal} = 1 - \frac{t_{spike}}{T}, \tag{3}$$

where N denotes the number of spikes, t_{spike} is the time of the first spike. Previous work shows a considerable time delay with the use of rate and temporal coding. For example, they all need at least 1,000 time steps to represent 0.001 of input.

Therefore, we use phase coding (Kim et al., 2018) to encode activation values to spike trains. It can pack more information in one spike by assigning different weights to spikes and thresholds of each phase. Thus, phase coding is more energy efficient. Experiments show a shorter time is taken to accurately represent the real value when phase coding is used:

$$a_{j}^{l} = \frac{1}{n}\sum_{k=1}^{nK} S_k \delta_{j,k}^{l}, \quad V_{th,t} = S_t V_{th}, \tag{4}$$

where a_j^l is the activation value of neuron j in layer l,K is the number of the phase of a period, $n = \frac{T}{K}$ is the number of the period, the phase function S is represented by

$$S_t = 2^{-(1+ \mod(t,K))}. \tag{5}$$

3.2. Framework of ANN-SNN conversion

To make SNN work, we need to do some processing on ANN before conversion. We use $a_i^l = \max\{0, \sum_j w_{ij} a_j^{l-1} + b_i^l\}$ to denote the arbitrary activation value in the ANN, w_{ij} and b_i^l are weight and bias respectively. The maximum firing rate in SNN is one because neurons emit one spike at most at every time step. Thus, we normalize the weight and bias with the data-norm method (Rueckauer et al., 2017) by

$$\hat{w}_{ij}^{l} = w_{ij}^{l}\frac{\lambda_{l-1}}{\lambda_l}, \quad \hat{b}_i^l = \frac{b_i^l}{\lambda_l}, \tag{6}$$

where $\hat{w}_{ij}^{l}$ and $\hat{b}_i^l$ represent the weights and biases used in SNN, λ_l is the maximum activation value of the l-th layer. Then all activation values in ANN are at most 1.

As mentioned above, it is hard to perform max-pooling and batch normalization (BN) in SNN. We choose the spike of the neuron with the largest firing rate to output as the max-pooling operation in SNN. We follow (Rueckauer et al., 2017) and merge the convolutional layer and the subsequent BN layer to form a new convolutional layer. An input x is transformed into $BN[x] = \frac{\gamma}{\theta}(x-\mu)+\beta$, where μ and θ are mean and variance of batch, β and γ are two learned parameters during training. The parameters of the new convolutional layer which can be converted, are described by

$$\hat{w_{ij}} = \frac{\gamma_i}{\theta_i} w_{ij}, \quad \hat{b_i} = \frac{\gamma_i}{\theta_i}(b_i - \mu_i) + \beta_i. \tag{7}$$

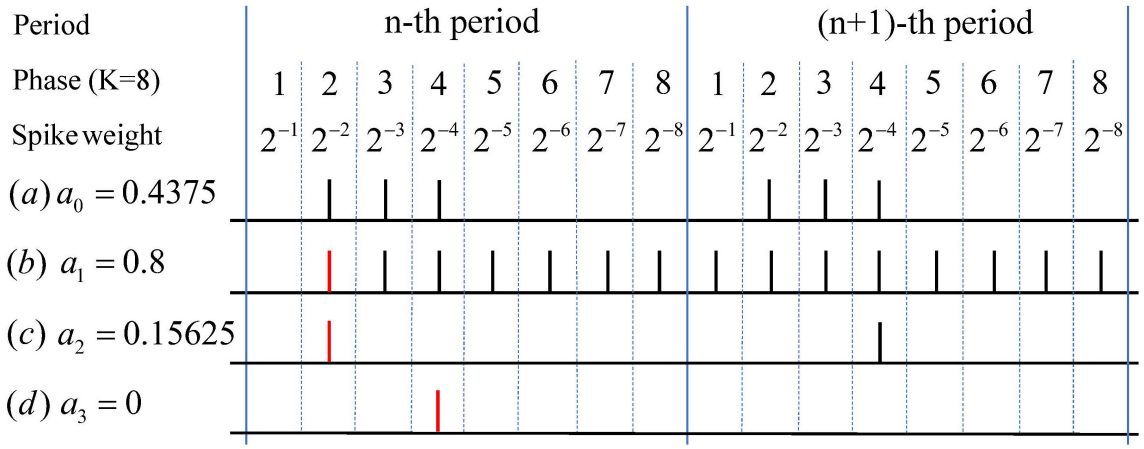

FIGURE 2
Examples of Phase Lag, Phase Lead, and SIN. Real value a_0 in (a) can be accurately represented with few spikes in phase coding. While due to the uncertainty of the postsynaptic current, phase lag problem like (b) occurs when the membrane potential in the current phase exceeds the threshold at the previous moment. When a current greater than one is suddenly received at a certain moment, phase lead problem like (c) will occur. Both of these problems will directly or indirectly lead to SIN in (d), where neurons corresponding to the activation value of zero will also spike.

3.3. Analysis of performance loss

Even though the ANN is processed, the converted SNN usually suffers performance loss. To simplify the analysis of performance loss, we assume that $a_i^l \geq 0$, $b_i^l = 0$ and the threshold V_{th} is 1. The neuron membrane potential is $V_{i,nK}^l$ at the end of the simulation. The total number of spikes of the neuron is numerically equal to the total received input minus the membrane potential at T:

$$N = \sum_{t}^{nK} S_t \sum_{j} w_{ij} \delta_{j,t}^{l-1} - V_{i,nK}^l. \tag{8}$$

Then the firing rate of neurons is approximately equal to the activation value in ANN when T is long enough:

$$\begin{aligned} r_{i,nK}^l &= \frac{N}{n} = \frac{1}{n} \sum_{t}^{nK} S_t \sum_{j} w_{ij} \delta_{j,t}^{l-1} - V_{i,nK}^l \\ &= \frac{1}{n} \left(\sum_{j}^{n} \sum w_{ij} \sum_{k=1}^{K} S_k \delta_{j,k}^{l-1} - V_{i,nK}^l \right) \\ &= \sum_{j} w_{ij} a_j^{l-1} - \frac{1}{n} V_{i,nK}^l. \end{aligned} \tag{9}$$

Note that the postsynaptic current at each moment is as follows:

$$I_{j,t} = \sum_{j} w_{ij} \delta_{j,t}^{l-1}. \tag{10}$$

As shown in Figure 2, once the neuron in hidden layers spikes earlier or later than the time directly encoded, which we call phase lead or phase lag, the neuron will transmit too much or too little information to the next layer. Suppose the total synaptic current received by the neuron at time T is equal to the product of the activation value and T, so we can get $\widetilde{I}_{j,T} = \sum_{t}^{T} I_{j,t} = T * a_j$, where $t \in (0, T]$. According to the instability of the synaptic current in Equation (10), if the total synaptic current received at time t $\widetilde{I}_{j,t}$ is less than the expected current $\frac{t}{T}\widetilde{I}_{j,T}$, the neuron will receive more current at a later time to send more spikes to make up for the shortage of the number of spikes in time step (0,t). Note that the neuron can only emit at most one spikes at each time step, so part of the information will be stored in the neuron in the form of membrane potential and cannot be released, if the number of spikes to be emitted exceeds T-t. However, if the total synaptic current received at time t is greater than the expected current, i.e., $\widetilde{I}_{j,t} > \frac{t}{T}\widetilde{I}_{j,T}$, for neurons with activation values greater than 0, this problem can be remedied by firing fewer spikes during time steps (t,T]. However, if the activation value is less than 0, due to the instability of the synaptic current, once it exceeds the threshold potential, the neuron will issue a spike, called spikes of inactivated neurons (SIN). SNN needs a long time to accumulate spikes to reduce the impact of these destructive spikes. Thus, the features corresponding to the network firing rate can be approximately equal and proportional to the ANN features, which is the reason for the large time delay of the converted SNN. When the problem of SIN is quite severe, e.g., a large number of features that should not be activated in the ANN are activated in the SNN, it cannot be solved by long-time simulation and causes serve performance loss. Note that the above analysis is also applicable to rate-based conversion methods.

3.4. Bistable SNN

The immediate response of the neuron to the received current is unreliable. How should the information propagate in the spiking neurons to make the spike trains conform to the encoding rules to avoid the SIN problem caused by phase lag and phase lead? We solve the problem by proposing a bistable IF neuron (BIF) combining the IF neuron and bistability mechanism. We model the process of spiking as a piecewise function according to the fact that the bistability is shown as the periodic change of spike and non-spike states. In the spike stage, neurons spike according to the membrane potential normally while can't spike in the non-spike phase:

$$\delta^l_{A,i,t} = \begin{cases} \mathcal{H}(V^l_{A,i,t} - V_{th,t}), & \mod(\lfloor \frac{t}{K} \rfloor, 2) = 1, \\ 0, \quad else. \end{cases}$$
$$\delta^l_{B,i,t} = \begin{cases} \mathcal{H}(V^l_{B,i,t} - V_{th,t}), & \mod(\lfloor \frac{t}{K} \rfloor, 2) = 0, \\ 0, \quad else. \end{cases} \tag{11}$$

where $\mathcal{H}(x)$ is unit step function, $\lfloor x \rfloor$ is the round-down operation. With periodic input, neurons do not have to respond to the input spikes all the time but accumulate spikes first and then respond and loop. Neurons respond accurately in each phase by accumulating spikes in the non-spike stage, which can effectively avoid the phase lead or lag mentioned above.

We use two BIF neurons as one unit to represent one activation value in the ANN, which is:

$$\delta^l_{i,t} = \delta^l_{B,i,t} + \delta^l_{A,i,t}. \tag{12}$$

One reason for using two BIF neurons is that the BIF neuron does not spike half the simulation time. The use of two neurons with complementary spike states can make the information be transmitted to the next layer in time and maintain the continuity of information transmission. One of the neurons in two adjacent layers is in the spike state to release memory information, and the other is in the non-spike state to accumulate spikes. Note that even if the neurons in the previous layer are in the non-spike state, its silence will not interfere with the neurons in the spike state connected to the next layer. Another reason is its powerful scalability. We can convert ANNs of various topologies without carefully designing the spike stage for each layer when converting deeper and wider ANNs. If only one BIF neuron is used in each layer, when the neuron is in a spike state, it cannot play the role of accumulation as described above.

As shown in Figure 3, there are two connections between the two units: neuron A of one unit is connected to neuron B of the other unit:

$$V^l_{A,i,t} = V^l_{A,i,t-1} + \sum_j w_{ij}\delta^{l-1}_{B,j,t}, \quad V^l_{B,i,t} = V^l_{B,i,t-1} + \sum_j w_{ij}\delta^{l-1}_{A,j,t}. \tag{13}$$

They share the same weight. When the presynaptic neuron is in the spike phase, the postsynaptic neuron in the non-spike phase accumulates spikes to respond accurately later. In fact, the information between the two adjacent layers is periodically switched between the red connection and the blue connection with the simulation time, which also reflects that our BSNN can convert any structure of ANN. Consider using real-valued input, $\delta^0_{i,t} = a^0_i$, the total synaptic current received by neurons in the last layer can be expressed as

$$\sum_{t=0}^{KL} I^L_{j,t} = \sum_{t=0}^{KL} S_t \sum w^L_{ij}\delta^{L-1}_{i,t} = \sum w_{ij}a^{L-1}_i \tag{14}$$

Bistable neurons combined with phase encoding can make the information in the network transmitted in the form of accumulation and then firing in each layer. Among them, the accumulation process can ensure that accurate information is transmitted to the next layer with a delay of one cycle, so as to avoid the influence of the immediate response of synaptic current on the conversion.

The residual block of ResNet has two information paths, in which shortcut path connects input and output directly or through a convolution operation. The convolutional layer and the BN layer are merged to facilitate the conversion. When converting ResNet, two key problems need to be addressed:

- **The information of the two paths cannot be scaled synchronously.** The information of two paths received by output neurons of the residual block is not proportional to the activation values. Because it is impossible to normalize the shortcut path which has no convolutional layer.
- **The information of the two paths cannot reach the output neuron synchronously.** The shortcut path is one less ReLU operation, which corresponds to two BIF neurons in the SNN, than the convolution path. Since neurons need time to accumulate membrane potential to spike, the information of the shortcut path reaches the output neuron faster.

3.5. Synchronous neurons for spiking ResNet

For the first problem, we determine the scale parameters according to the maximum activation value of the input and output so that the sum of the information of the two paths received by the output is proportional to the activation value:

$$scale = \frac{\lambda_{in}}{\lambda_{out}}. \tag{15}$$

To solve the second problem, we add synchronous neurons, which are two BIF neurons, in the shortcut path. It is

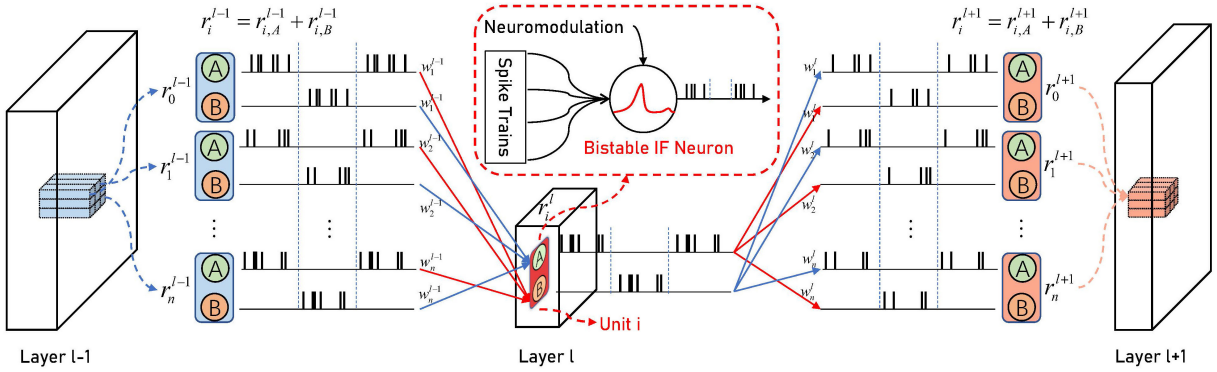

FIGURE 3
The Architecture of BSNN. Neuron A in the unit, which corresponds to an ANN activation value, only has synaptic connections with neuron B in the adjacent layer and vice versa. The weights of two connections in the same unit are the same. When neuron A in the l-th layer is in the spike stage, neuron B in the $(l-1)$-th layer is in the non-spike stage. It will not cause interference to spikes of neuron A in layer l, and neuron B of the $(l+1)$-th layer can integrate the spikes until the end of the period.

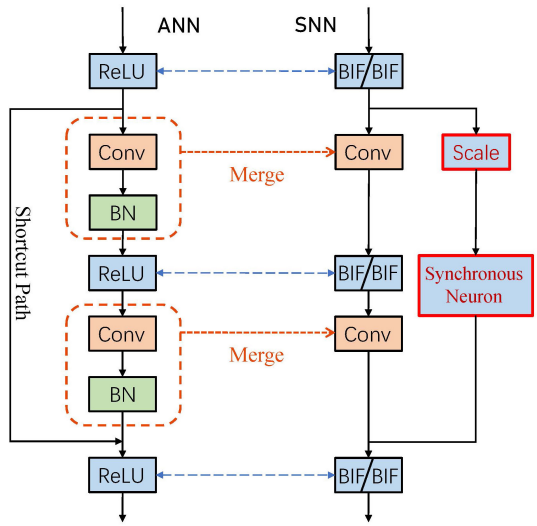

FIGURE 4
Synchronous neurons for spiking ResNet.

equivalent to adding a ReLU function to the head of the shortcut path in ANN. Figure 4 shows the conversion process of the residual block. The information reaches the output of the residual block through the synchronous neurons. Since the input of the shortcut path is all non-negative, the transmission in ANN will not have any impact. In SNN, due to the existence of synchronous neurons, the output of the shortcut path and the convolutional path will reach the output neuron at the same time, thereby eliminating the phase lead and lag and SIN problems in spiking ResNet. The entire conversion process summarized in Algorithm 1 where the SNNs transmit information with BIF neurons.

Input: Training and test set, simulation time T, trained ANN
Output: Performance of the SNN

Let $V_{th}=1, \lambda_l=0$ for $l=1,\cdots,L$ to save the maximum activation value of each ANN layer.
Merge the convolutional layer and BN layer according to Equation (7).
for $l=1$ to L **do**
  $a^l \leftarrow$ layer-wise activation value
  $\lambda_l = \max\{a_i^l\}$
end for
for $l=1$ to L **do**
  $\hat{w}_{ij}^l = w_{ij}^l \frac{\lambda_{l-1}}{\lambda_l}, \quad \hat{b}_i^l = \frac{b_i^l}{\lambda_l}$
end for
Map the processed parameters to the SNN.
for $s=1$ to # of test set **do**
  for $t=1$ to L **do**
    do inference according to Equations (11), (12), (13)
  end for
end for
return performance of the SNN

Algorithm 1. ANN-SNN conversion with BIF neurons.

4. Experiment

In this section, various experiments are conducted to evaluate the performance of our proposed conversion algorithm.

We also test the effect of the synchronous neurons and compare our BSNN with various advanced conversion algorithms.

4.1. Dataset

The MNIST (LeCun et al., 1998), CIFAR-10, CiFAR-100 (Krizhevsky et al., 2009), and ImageNet (Deng et al., 2009) datasets are used to test the performance of our proposed BSNN.

The MNIST dataset is the most commonly used dataset and benchmark for classification tasks. It contains 60,000 handwritten digital images from 0 to 9, 50,000 images for the training set, and 10,000 images for the test set. Each image contains 28x28 pixels, which are represented in the form of 8-bit gray values. Note that we do not perform any preprocessing on the MNIST dataset.

The CIFAR-10 dataset is the color image dataset closer to universal objects and a benchmark test set of the CNN architecture. It contains 60,000 images of 10 classes. 50,000 images for the training sets, and 10,000 images for the test sets. It is a 3-channel color RGB image, whose size of each image is 32x32. Unlike MNIST, we normalize the dataset to make the CIFAR-10 obey a standard normal distribution.

The CIFAR-100 dataset has the same image format as CIFAR-10. We also perform the same normalization operation on it, with different normalization parameters. The difference with CIFAR-10 is that CIFAR-100 contains 100 categories instead of 10. Each category contains 500 training images and 100 test images.

ImageNet is currently the world's largest image recognition large-scale labeled image database organized according to the wordnet structure, and it is also the most challenging classification dataset for SNN. Among them, the training set is 1281167 pictures, and the verification set is 50,000 pictures, including 1,000 different categories and 3-channel natural images. The normalization process is also performed to obtain a sufficiently high classification performance.

4.2. Experimental setup

Our experiments are implemented on the Pytroch framework and NVIDIA A100. We convert CNN with 12c5-2s-64c5-2s-10 architecture (Kim et al., 2018) on MNIST. 12c5 means a convolutional layer with 12 output channels and kernel size of 5 and 2s refers to non-overlapping pooling layer with kernel size of 2. We use VGG16, ResNet18, ResNet20 architecture on CIFAR-10 and CIFAR-100, while ResNet18 and ResNet34 are used for experiments on ImageNet. Their structures are the same as that of Pytorch's built-in model. We train the ANN for 100 or 300 epochs by using the stochastic gradient descent algorithm. The initial learning rate is 0.01, and the learning rate is scaled by 0.1 at the training epoch of [180, 240, 270]. We use real-value input in SNNs for better performance. We use data augmentation on the datasets except for MNIST. We set the padding to 4 and crop the training data to 32*32. We also use other data augmentation, such as random horizontal flip, Cutout, and AutoAugment. For CIFAR10 and CIFAR100, we use stochastic gradient descent (SGD) as the optimizer with an initial learning rate of 0.1. The cosine decay strategy is used. Our batch size is 128, and the total epochs of training are 300.

4.3. Performance and comparison with other methods

Then we compare the performance of our model and other conversion methods on MNIST, CIFAR-10, CIFAR-100, and ImageNet, as shown in Table 1. The time step is the simulation time required to achieve the best performance. We choose rate-based methods including p-Norm (Rueckauer et al., 2017), Spike-Norm (Sengupta et al., 2019), RMP-SNN (Han et al., 2020), Opt. (Deng and Gu, 2021), SpikeConverter (Liu et al., 2022), etc., phase-based Weighted Spikes (Kim et al., 2018) method, temporal coding-based TSC (Han and Roy, 2020) method, and other advanced methods such as CQ trained (Yan et al., 2021), Hybrid training (Rathi et al., 2020), etc. for comparison. The biggest difference between BSNN and these methods is that information is passed from layer to layer in a cycle K, thus avoiding the immediate response of neurons to synaptic currents.

Here we do not compare the BSNN with algorithms based on biological rules and backpropagation. Because the former focuses on the biological interpretability of the network, while the latter focuses on exploring the temporal and spatial representation of features. The training cost of both is particularly high because of the information processing method similar to RNN in the training process. It is difficult to apply them to complex networks such as VGG16 and ResNet34, Thus, their performance significantly lags behind advanced conversion-based methods.

We first focus on the performance loss of the conversion method. The phase-based method is usually better than other methods because it combines the advantages of rate coding and temporal coding. The time information expressed in phase and the rate information expressed in period improve the information expressing ability of the spike. Based on this, our BSNN improves the information propagation of SNN based on BIF neurons and reduces the phase lead and lag problems in the Weighted Spike method, thus minimizing the performance loss. We achieved 99.31% performance on MNIST, 94.12% (VGG16), and 95.02% (ResNet20) performance on CIFAR-10, 73.41% (VGG16) and 78.12% (ResNet20) performance on CIFAR-100, and 69.65% (ResNet18) performance on ImageNet, which are better than other conversion method. To continue testing the ability of our method to convert deep networks, we conduct

TABLE 1 Top-1 classification accuracy on MNIST, CIFAR-10, CIFAR-100, and ImageNet for our converted SNNs, compared to the original ANNs, and compared to other conversion methods.

Dataset	Method	Network	Encoding	ANN (%)	SNN (%)	Loss (%)	Time steps
MNIST	p-Norm (Rueckauer et al., 2017)	CNN	Rate	99.44	99.44	0.00	-
	Weighted Spikes (Kim et al., 2018)	CNN	Phase	99.20	99.20	0.00	16
	BSNN	**CNN**	**Phase**	**99.30**	**99.31**	**−0.01**	**35**
CIFAR-10	p-Norm (Rueckauer et al., 2017)	VGG16	Rate	91.91	91.85	0.06	35
	Spike-Norm (Sengupta et al., 2019)	VGG16	Rate	91.70	91.55	0.15	-
	Hybrid Training (Rathi et al., 2020)	VGG16	Rate	92.81	91.13	1.68	100
	RMP-SNN (Han et al., 2020)	VGG16	Rate	93.63	93.63	0.00	1536
	TSC (Han and Roy, 2020)	VGG16	Temporal	93.63	93.63	0.00	2048
	CQ Trained (Yan et al., 2021)	VGG16	Rate	92.56	92.48	0.08	600
	Opt. (Deng and Gu, 2021)	VGG16	Rate	92.34	92.24	0.10	128
	BSNN	**VGG16**	**Phase**	**94.11**	**94.12**	**−0.01**	**166**
	Weighted Spikes (Kim et al., 2018)	ResNet20	Phase	91.40	91.40	0.00	-
	Hybrid Training (Rathi et al., 2020)	ResNet20	Rate	93.15	92.22	0.93	250
	RMP-SNN (Han et al., 2020)	ResNet20	Rate	91.47	91.36	0.11	-
	TSC (Han and Roy, 2020)	ResNet20	Temporal	91.47	91.42	0.05	1536
	Opt. (Deng and Gu, 2021)	ResNet20	Rate	93.61	93.56	0.05	128
	BSNN	**ResNet20**	**Phase**	**95.02**	**95.16**	**−0.14**	**206**
CIFAR-100	Hybrid Training (Rathi et al., 2020)	VGG11	Rate	71.21	67.87	3.34	125
	RMP-SNN (Han et al., 2020)	VGG16	Rate	71.22	70.93	0.29	2048
	TSC (Han and Roy, 2020)	VGG16	Temporal	71.22	70.97	0.25	1024
	CQ Trained (Yan et al., 2021)	VGG	Rate	71.84	71.84	0.00	300
	Opt. (Deng and Gu, 2021)	VGG16	Rate	70.49	70.47	0.02	128
	BSNN	**VGG16**	**Phase**	**73.26**	**73.41**	**−0.15**	**242**
	Spiking ResNet (Hu et al., 2018)	ResNet44	Rate	70.18	68.56	1.62	-
	Weighted Spikes (Kim et al., 2018)	ResNet32	Phase	66.10	66.20	-0.10	-
	RMP-SNN (Han et al., 2020)	ResNet20	Rate	68.72	67.82	0.90	2048
	TSC (Han and Roy, 2020)	ResNet	Temporal	68.72	68.18	0.54	2048
	Opt. (Deng and Gu, 2021)	ResNet20	Rate	69.80	69.49	0.31	128
	BSNN	**ResNet20**	**Phase**	**77.97**	**78.12**	**−0.15**	**265**
ImageNet	Spike-Norm (Sengupta et al., 2019)	ResNet20	Rate	70.52	69.39	1.13	-
	BSNN	**ResNet18**	**Phase**	**69.65**	**69.65**	**0.00**	**200**
	Hybrid Training (Rathi et al., 2020)	ResNet34	Rate	70.20	61.48	8.72	250
	RMP-SNN (Han et al., 2020)	ResNet34	Rate	70.64	69.89	0.75	4096
	SpikeConverter (Liu et al., 2022)	ResNet34	Rate	70.64	70.57	0.07	16
	BSNN	**ResNet34**	**Phase**	**73.27**	**72.64**	**0.63**	**989**

Bold values represents experimental results.

experiments on ResNet34. The results show that BSNN only needs less than 1,000 time steps to achieve the performance of 72.64% with only 0.63% performance loss. As far as we know, this is also the highest performance that SNN can achieve.

In addition to the excellence in accuracy, our model has also achieved outstanding performance in time steps. The conversion method based on rate and timing naturally takes a long time to accurately represent the information and therefore requires a longer time step. The Hybrid Training method sacrifices part of the performance in exchange for shorter simulation time. We analyze above that the reason why the conversion method requires a long simulation time is that SNN needs enough spikes to compensate for the destruction of the proportional relationship caused by spikes of inactivated neurons. BSNN uses the bistable mechanism to accumulate and release spikes, thus the SIN problem is significantly improved. As shown in the Table 1, on complex datasets such as CIFAR-10 and ImageNet, BSNN only needs a time step of 1/4 to 1/10 to achieve the performance of other advanced algorithms. hence, BSNN can save at least 25% of calculation loss and energy consumption to a certain extent, which plays an important role in the development and application of SNN.

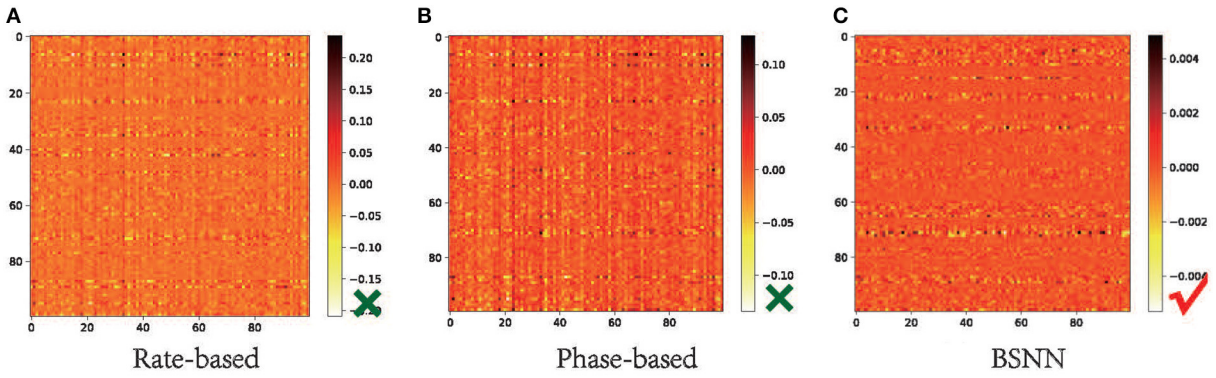

FIGURE 5
Output difference between ANN and converted SNN on CIFAR-100. We compare the output rate of 100 samples of VGG16 with the output firing rate of three conversion methods to verify that the reason for BSNN to improve network performance lies in better approximation. **(A)** Rate-based. **(B)** Phase-based. **(C)** BSNN.

4.4. Effect of bistable neuron

To obtain a high-performance SNN, the firing rate of the converted SNN should be similar or equal to the activation value of ANN, which is consistent with the conversion principle. We check the output difference of 100 samples of CIFAR-100 between the firing rate of converted SNN and the corresponding activation value of the ANN with architecture of VGG16. Ideally, due to the weight normalization, the output of the ANN is proportional to the firing frequency of the SNN output, and the multiple is the maximum value of the ANN output layer. We multiply the output of the SNN with the multiple for comparison. As we can see from Figure 5, the difference between the output of the selected 100 samples and the output of the ANN is mostly near 0. However, although the rate-based conversion method is widely used, it can be seen from the output of the network that the performance loss is that SNN cannot approach the activation value of ANN very well. The method based on phase encoding reduces the difference between the outputs by increasing the amount of information contained in the spikes, however, the problem of inaccurate approximation is still not solved. As can be seen in Figure 5C, the output of BSNN is at most 0.005 different from the corresponding activation value of ANN. This indicates that the improvement of performance with BSNN comes from the accurate approximation to ANN activation values.

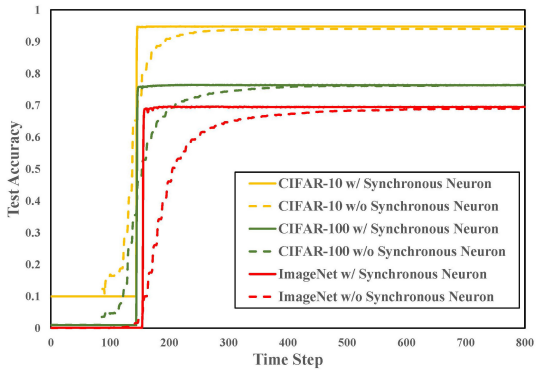

FIGURE 6
The performance of spiking ResNet18 on three datasets.

TABLE 2 The results of adding synchronous neurons on ResNet18.

	Dataset	SNN (%)	Loss (%)	Time steps
IF w/ SN	CIFAR-10	94.37	0.41	394
	CIFAR-100	76.35	0.05	775
	ImageNet	69.22	0.42	1,000
BIF w/out SN	CIFAR-10	94.04	0.74	395
	CIFAR-100	76.37	0.03	741
	ImageNet	69.32	0.32	996
BIF w/ SN	CIFAR-10	94.83	−0.05	218
	CIFAR-100	76.48	−0.08	237
	ImageNet	69.64	0.00	200

4.5. Effect of synchronous neuron

In order to verify the effectiveness of the proposed synchronous neuron in converting ResNet, we convert ResNet18 on multiple datasets. As shown in Figure 6, since neurons are not always in the spike state but switch between spike and non-spike states, BSNN doesn't work in the early simulation but completes the high-precision conversion with a small time delay. The detailed results are listed in Table 2. The loss means the

accuracy difference ($acc_{ANN} - acc_{SNN}$) between the source ANN and the converted SNN. The experimental results show that the performance of the spiking ResNet using synchronous neurons exceeds the SNNs without synchronous neurons on CIFAR-10, CIFAR-100, and ImageNet datasets. It achieves the same performance as the ANN with 200–800 time-steps reduction. The use of synchronous neurons on ResNet conversion can ensure that the information of two paths reaches the output neuron of the residual block synchronously, which significantly improves the conversion accuracy and reduces the time delay. We can see form Table 2 that SN does not play a significant role in other methods that support ResNet, such as RMP-SNN (Han et al., 2020), because their information is not periodically accumulated and released.

Note that previous work like Spike-Norm (Sengupta et al., 2019) uses average pooling and dropout instead of max-pooling and BN, limiting the performance of the converted SNN to a certain extent. The results show that our work can be adapted to various types of ANNs, and achieve almost lossless conversion with less time delay. Experimental results on complex datasets like CIFAR-100 and deep networks like ResNet34 show that BSNN can solve the difficulty in approximating features in deep layers to ANN by cooperating two BIF neurons of each unit to accumulate and emit spikes periodically. It means that we can achieve the same effect as current deep learning with a more biologically plausible network structure, less computational cost and energy consumption.

5. Conclusion

In this paper, we analyze the reasons for the performance loss and large time delay in the conversion method. Our analysis reveals that the immediate response of neurons to the received current is unreliable in converted SNNs. It can bring the problem of SIN, which makes the firing rate in the deep layer cannot approximate the activation values in ANNs. Based on these analysis and observation, we propose a novel Bistable SNN which combines phase coding and the bistability mechanism, and design synchronous neurons to improve energy-efficiency, performance, and inference speed. Our experiments demonstrate that the BSNNs could significantly reduce performance loss and time delay. The efficiency and efficacy of our proposed BSNN could thus be of great importance for fast and energy-efficiency spike-based neuromorphic computing.

Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

Author contributions

YL wrote the code, performed the experiments, and wrote the manuscript. DZ and YL analyzed the data. DZ revised the manuscript. YZ proposed and supervised the project and contributed to writing the manuscript. All authors contributed to the article and approved the submitted version.

Funding

This work was supported by the National Key Research and Development Program (2020AAA0107800), and the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDB32070100).

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

Abbott, L. F., and Nelson, S. B. (2000). Synaptic plasticity: taming the beast. *Nat. Neurosci.* 3, 1178–1183. doi: 10.1038/81453

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv: 1409.0473*.

Bengio, Y., Mesnard, T., Fischer, A., Zhang, S., and Wu, Y. (2015). STDP as presynaptic activity times rate of change of postsynaptic activity. *arXiv preprint arXiv:1509.05936*.

Bing, Z., Meschede, C., Röhrbein, F., Huang, K., and Knoll, A. C. (2018). A survey of robotics control based on learning-inspired spiking neural networks. *Front. Neurorobot.* 12, 35. doi: 10.3389/fnbot.2018.00035

Blakemore, C., Carpenter, R. H., and Georgeson, M. A. (1970). Lateral inhibition between orientation detectors in the human visual system. *Nature* 228, 37–39. doi: 10.1038/228037a0

Burr, G. W., Shelby, R. M., Sebastian, A., Kim, S., Kim, S., Sidler, S., et al. (2017). Neuromorphic computing using non-volatile memory. *Adv. Phys: X* 2, 89–124. doi: 10.1080/23746149.2016.1259585

Cao, Y., Chen, Y., and Khosla, D. (2015). Spiking deep convolutional neural networks for energy-efficient object recognition. *Int. J. Comput. Vision* 113, 54–66. doi: 10.1007/s11263-014-0788-3

Chen, T., Du, Z., Sun, N., Wang, J., Wu, C., Chen, Y., et al. (2014). Diannao: a small-footprint high-throughput accelerator for ubiquitous machine-learning. *ACM SIGARCH Comput. Arch. News* 42, 269–284. doi: 10.1145/2654822.2541967

Davies, M. (2019). Benchmarks for progress in neuromorphic computing. *Nat. Mach. Intell.* 1, 386–388. doi: 10.1038/s42256-019-0097-1

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). "ImageNet: a large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition* (IEEE), 248–255. doi: 10.1109/CVPR.2009.5206848

Deng, S., and Gu, S. (2021). Optimal conversion of conventional artificial neural networks to spiking neural networks. *arXiv preprint arXiv:2103.00476*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Diehl, P. U., and Cook, M. (2015). Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Front. Comput. Neurosci.* 9, 99. doi: 10.3389/fncom.2015.00099

Diehl, P. U., Neil, D., Binas, J., Cook, M., Liu, S.-C., and Pfeiffer, M. (2015). "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *2015 International Joint Conference on Neural Networks (IJCNN)* (IEEE), 1–8. doi: 10.1109/IJCNN.2015.7280696

Fang, H., Zeng, Y., and Zhao, F. (2021). Brain inspired sequences production by spiking neural networks with reward-modulated STDP. *Front. Comput. Neurosci.* 15, 8. doi: 10.3389/fncom.2021.612041

Fang, W., Yu, Z., Chen, Y., Huang, T., Masquelier, T., and Tia, Y. (2021a). Deep residual learning in spiking neural networks. *arXiv [Preprint]*. arXiv: 2102.04159. Available online at: https://proceedings.neurips.cc/paper/2021/file/afe434653a898da20044041262b3ac74-Paper.pdf

Fang, W., Yu, Z., Chen, Y., Masquelier, T., Huang, T., and Tian, Y. (2021b). "Incorporating learnable membrane time constant to enhance learning of spiking neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2661–2671. doi: 10.1109/ICCV48922.2021.00266

Girshick, R. (2015). "Fast r-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, 1440–1448. doi: 10.1109/ICCV.2015.169

Han, B., and Roy, K. (2020). "Deep spiking neural network: energy efficiency through time based coding," in *Proc. IEEE Eur. Conf. Comput. Vis. (ECCV)*, 388–404. doi: 10.1007/978-3-030-58607-2_23

Han, B., Srinivasan, G., and Roy, K. (2020). "RMP-SNN: residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13558–13567. doi: 10.1109/CVPR42600.2020.01357

Hao, Y., Huang, X., Dong, M., and Xu, B. (2020). A biologically plausible supervised learning method for spiking neural networks using the symmetric stdp rule. *Neural Netw.* 121, 387–395. doi: 10.1016/j.neunet.2019.09.007

He, K., Zhang, X., Ren, S., and Sun, J. (2016). "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778. doi: 10.1109/CVPR.2016.90

He, X., and Cheng, J. (2018). "Learning compression from limited unlabeled data," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 752–769. doi: 10.1007/978-3-030-01246-5_46

Hu, Y., Tang, H., Wang, Y., and Pan, G. (2018). Spiking deep residual network. *arXiv preprint arXiv:1805.01352*.

Illing, B., Gerstner, W., and Brea, J. (2019). Biologically plausible deep learning–but how far can we go with shallow networks? *Neural Netw.* 118, 90–101. doi: 10.1016/j.neunet.2019.06.001

Ito, M. (1989). Long-term depression. *Annu. Rev. Neurosci.* 12, 85–102. doi: 10.1146/annurev.ne.12.030189.000505

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Trans. Neural Netw.* 14, 1569–1572. doi: 10.1109/TNN.2003.820440

Jang, H., Simeone, O., Gardner, B., and Gruning, A. (2019). An introduction to probabilistic spiking neural networks: probabilistic models, learning rules, and applications. *IEEE Signal Process. Mag.* 36, 64–77. doi: 10.1109/MSP.2019.2935234

Kemker, R., McClure, M., Abitino, A., Hayes, T., and Kanan, C. (2018). "Measuring catastrophic forgetting in neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*. doi: 10.1609/aaai.v32i1.11651

Kim, J., Kim, H., Huh, S., Lee, J., and Choi, K. (2018). Deep neural networks with weighted spikes. *Neurocomputing* 311, 373–386. doi: 10.1016/j.neucom.2018.05.087

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science* 350, 1332–1338. doi: 10.1126/science.aab3050

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 2278–2324. doi: 10.1109/5.726791

Liang, Q., and Zeng, Y. (2021). Stylistic composition of melodies based on a brain-inspired spiking neural network. *Front. Syst. Neurosci.* 15, 21. doi: 10.3389/fnsys.2021.639484

Liu, F., Zhao, W., Chen, Y., Wang, Z., and Jiang, L. (2022). "SpikeConverter: an efficient conversion framework zipping the gap between artificial neural networks and spiking neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*. doi: 10.1609/aaai.v36i2.20061

Liu, F., Zhao, W., Chen, Y., Wang, Z., Yang, T., and Jiang, L. (2021). SSTDP: supervised spike timing dependent plasticity for efficient spiking neural network training. *Front. Neurosci.* 15, 756876. doi: 10.3389/fnins.2021.756876

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., et al. (2016). "SSD: single shot multibox detector," in *European Conference on Computer Vision* (Springer), 21–37. doi: 10.1007/978-3-319-46448-0_2

Lobo, J. L., Del Ser, J., Bifet, A., and Kasabov, N. (2020). Spiking neural networks and online learning: an overview and perspectives. *Neural Netw.* 121, 88–100. doi: 10.1016/j.neunet.2019.09.004

Maass, W. (1997). Networks of spiking neurons: the third generation of neural network models. *Neural Netw.* 10, 1659–1671. doi: 10.1016/S0893-6080(97)00011-7

Malenka, R. C. (2003). The long-term potential of LTP. *Nat. Rev. Neurosci.* 4, 923–926. doi: 10.1038/nrn1258

Marder, E., Abbott, L., Turrigiano, G. G., Liu, Z., and Golowasch, J. (1996). Memory from the dynamics of intrinsic membrane currents. *Proc. Natl. Acad. Sci. U.S.A.* 93, 13481–13486. doi: 10.1073/pnas.93.24.13481

Marian, I., Reilly, R., and Mackey, D. (2002). Efficient event-driven simulation of spiking neural networks.

Meng, Q., Xiao, M., Yan, S., Wang, Y., Lin, Z., and Luo, Z.-Q. (2022). "Training high-performance low-latency spiking neural networks by differentiation on spike representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12444–12453.

Nguyen, A., Yosinski, J., and Clune, J. (2015). "Deep neural networks are easily fooled: high confidence predictions for unrecognizable images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 427–436. doi: 10.1109/CVPR.2015.7298640

Noble, W. S. (2006). What is a support vector machine? *Nat. Biotechnol.* 24, 1565–1567. doi: 10.1038/nbt1206-1565

Park, D. S., Zhang, Y., Jia, Y., Han, W., Chiu, C.-C., Li, B., et al. (2020). Improved noisy student training for automatic speech recognition. *arXiv preprint arXiv:2005.09629*. doi: 10.21437/Interspeech.2020-1470

Park, S., Kim, S., Choe, H., and Yoon, S. (2019). "Fast and efficient information transmission with burst spikes in deep spiking neural networks," in *2019 56th ACM/IEEE Design Automation Conference (DAC)* (IEEE), 1–6. doi: 10.1145/3316781.3317822

Pfeiffer, M., and Pfeil, T. (2018). Deep learning with spiking neurons: opportunities and challenges. *Front. Neurosci.* 12, 774. doi: 10.3389/fnins.2018.00774

Rathi, N., Srinivasan, G., Panda, P., and Roy, K. (2020). Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. *arXiv preprint arXiv:2005.01807*.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 779–788. doi: 10.1109/CVPR.2016.91

Roy, K., Jaiswal, A., and Panda, P. (2019). Towards spike-based machine intelligence with neuromorphic computing. *Nature* 575, 607–617. doi: 10.1038/s41586-019-1677-2

Rueckauer, B., and Liu, S.-C. (2018). "Conversion of analog to spiking neural networks using sparse temporal coding," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)* (IEEE), 1–5. doi: 10.1109/ISCAS.2018.8351295

Rueckauer, B., Lungu, I.-A., Hu, Y., Pfeiffer, M., and Liu, S.-C. (2017). Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Front. Neurosci.* 11, 682. doi: 10.3389/fnins.2017.00682

Sengupta, A., Ye, Y., Wang, R., Liu, C., and Roy, K. (2019). Going deeper in spiking neural networks: VGG and residual architectures. *Front. Neurosci.* 13, 95. doi: 10.3389/fnins.2019.00095

Song, S., Balaji, A., Das, A., Kandasamy, N., and Shackleford, J. (2020). "Compiling spiking neural networks to neuromorphic hardware," in *The 21st ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems*, 38–50. doi: 10.1145/3372799.3394364

Strubell, E., Ganesh, A., and McCallum, A. (2019). Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv*:1906.02243. doi: 10.18653/v1/P19-1355

Tavanaei, A., Ghodrati, M., Kheradpisheh, S. R., Masquelier, T., and Maida, A. (2019). Deep learning in spiking neural networks. *Neural Netw.* 111, 47–63. doi: 10.1016/j.neunet.2018.12.002

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Wang, X., Lin, X., and Dang, X. (2020). Supervised learning in spiking neural networks: a review of algorithms and evaluations. *Neural Netw.* 125, 258–280. doi: 10.1016/j.neunet.2020.02.011

Wu, H., Zhang, Y., Weng, W., Zhang, Y., Xiong, Z., Zha, Z.-J., et al. (2021). "Training spiking neural networks with accumulated spiking flow," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 10320–10328. doi: 10.1609/aaai.v35i12.17236

Wu, J., Leng, C., Wang, Y., Hu, Q., and Cheng, J. (2016). "Quantized convolutional neural networks for mobile devices," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4820–4828. doi: 10.1109/CVPR.2016.521

Xing, F., Yuan, Y., Huo, H., and Fang, T. (2019). "Homeostasis-based CNN-to-SNN conversion of inception and residual architectures," in *International Conference on Neural Information Processing* (Springer), 173–184. doi: 10.1007/978-3-030-36718-3_15

Yan, M., Chan, C. A., Gygax, A. F., Yan, J., Campbell, L., Nirmalathas, A., et al. (2019). Modeling the total energy consumption of mobile network services and applications. *Energies* 12, 184. doi: 10.1007/978-3-030-28468-8

Yan, Z., Zhou, J., and Wong, W. -F. (2021). "Near lossless transfer learning for spiking neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35* (AAAI Press), 10577–10584.

Yang, X., Zhang, Z., Zhu, W., Yu, S., Liu, L., and Wu, N. (2020). Deterministic conversion rule for CNNs to efficient spiking convolutional neural networks. *Sci. China Inform. Sci.* 63, 122402. doi: 10.1007/s11432-019-1468-0

Zeng, Y., Zhang, T., and Xu, B. (2017). Improving multi-layer spiking neural networks by incorporating brain-inspired rules. *Sci. China Inform. Sci.* 60, 052201. doi: 10.1007/s11432-016-0439-4

Zhang, L., Zhou, S., Zhi, T., Du, Z., and Chen, Y. (2019). "TDSNN: from deep neural networks to deep spike neural networks with temporal-coding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 1319–1326. doi: 10.1609/aaai.v33i01.33011319

Zhang, T., Zeng, Y., Zhao, D., and Shi, M. (2018). "A plasticity-centric approach to train the non-differential spiking neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*. doi: 10.1609/aaai.v32i1.11317

Zhao, B., Yu, Q., Ding, R., Chen, S., and Tang, H. (2014). "Event-driven simulation of the tempotron spiking neuron," in *2014 IEEE Biomedical Circuits and Systems Conference (BioCAS) Proceedings* (IEEE), 667–670. doi: 10.1109/BioCAS.2014.6981814