Received: 16 November 2021 | Revised: 9 February 2022 | Accepted: 8 March 2022

CAAI Transactions on Intelligence Technology

DOI: 10.1049/cit2.12082

ORIGINAL RESEARCH

SpikeGoogle: Spiking Neural Networks with GoogLeNet-like inception module

Xuan Wang[1] | Minghong Zhong[1] | Hoiyuen Cheng[1] | Junjie Xie[1] | Yingchu Zhou[2] | Jun Ren[3] | Mengyuan Liu[1]

[1]School of Intelligent Systems Engineering, Sun Yat-sen University, Shenzhen, China, Guangdong Provincial Key Laboratory of Fire Science and Intelligent Emergency Technology, Guangzhou, China

[2]Shenzhen Academy of Metrology and Quality Inspection, Shenzhen, China

[3]Infocare Systems Limited, New Zealand

Correspondence

Mengyuan Liu, School of Intelligent Systems Engineering, Sun Yat-sen University, Shenzhen, China.
Email: nkliuyifang@gmail.com

Funding information

Key-Area Research and Development Program of Guangdong Province, Grant/Award Number: 2020B0404020005

Abstract

Spiking Neural Network is known as the third-generation artificial neural network whose development has great potential. With the help of Spike Layer Error Reassignment in Time for error back-propagation, this work presents a new network called SpikeGoogle, which is implemented with GoogLeNet-like inception module. In this inception module, different convolution kernels and max-pooling layer are included to capture deep features across diverse scales. Experiment results on small NMNIST dataset verify the results of the authors' proposed SpikeGoogle, which outperforms the previous Spiking Convolutional Neural Network method by a large margin.

KEYWORDS

GoogLeNet, inception, Spiking Neural Networks

1 | INTRODUCTION

In recent years, Deep Neural Networks (DNNs) have developed rapidly in the fields of Computer Vision (CV), audio recognition and Natural Language Processing (NLP), with various systems achieving many important breakthroughs and displaying impressive performance. However, the simulation of biological neurons by DNNs is relatively simple and does not have brain-like interpretability. In contrast, Spiking Neural Networks (SNNs) fully simulate the actual situation of the biological nervous system and have low computing consumption [1]. It uses action potential coding methods to convert input stimulus signal into spike train with a dimension of time and also uses the change states of membrane potential to characterise spike information, which is biologically interpretable and rich in expressing features of neural computing. In addition, SNNs have low dependence on high-performance devices for its sparse event-driven information, simple arithmetic unit and unique calculation module. Once a spike (0 or 1) is received, it will be multiplied with the synapse weight, that is, the weight will be directly passed to the adder without specific multiplier. And the neuron will only be activated when the accumulated input spikes reach the threshold. In particular, learning with a small number of training samples is a potential development area for SNNs, where it is possible to obtain better results than deep learning [2]. In addition, the asynchronous spiking mechanism of SNNs makes it advantageous in event-based scenarios like flow estimation, spike pattern recognition and Simultaneous Localisation and Mapping (SLAM) [3–6].

Although the development of SNNs has great potential, there are still quite a lot of difficulties and uncharted territories in current research. Due to the complex dynamic characteristics of biological neurons and the non-differentiable problem of spike function in the process of error back-propagation, there is no

Xuan Wang and Minghong Zhong have equally contributed this work.

scalable training method in this field. At present, converting Artificial Neural Network or Convolutional Neural Network to Spiking Neural Network (ANN2SNN, CNN2SNN) are widely used SNN training methods, which are mainly based on the target SNN result, applying insights into Artificial Neural Network (ANN) or Convolutional Neural Network (CNN) for parameter training, and then reusing the training weight results into the SNN [7]. This method has achieved good results on some tasks, but it is clear that this kind of learning ignores important information contained between adjacent spikes in the time dimension. Therefore, our work intends to use the Spike Layer Error Reassignment in Time (SLAYER) training method [8], a back-propagation algorithm that combines spatial information with temporal domain information to update synaptic weights and axonal delays. Furthermore, when Spiking Convolutional Neural Network (SCNN) structure processes, a small amount of dataset, especially the point cloud form, is very prone to over-fitting. Based on this, we introduce a model structure, SpikeGoogle, which is implemented with the GoogLeNet-like inception module.

The Inception structure [9] is a module for small sample learning and a multi-scale processing rule, which realises a sparse network structure unit. The structure gathers highly correlated neurons, and to a certain extent, conforms to the neuron connection mode of the biological brain. In Hebbian's theory, it pointed out that the strength of the synaptic connection will change with the state changes of the two neurons, and it is only related to the state of the two [10, 39]. This theory is the important foundation of relative development of spiking neural networks, and it is very consistent with the implementation principles of SNN. Therefore, inspired by the application of the sparseness of biological neuron connections and the plasticity of synaptic expression in Inception structure, we propose to transfer this model architecture to SNN to highlight the advantage of a more complete biological nervous system simulation. The main content of our work lies in the following three parts. First, convert the input stimulus signal into spike train to be suitable for the data mode processed by SNNs. Second, construct the SpikeGoogle network to realise the dynamic complex information processing in the spatial–temporal domain. Finally, use the simulated spike function to solve the non-differentiable problem of spikes.

The innovations of this paper can be summarised in the following two points. First of all, a new network, SpikeGoogle, which is suitable for SNN learning is proposed. It can obtain a small amount of NMNIST [36] data and optimise network performance on the basis of shallow network. Second, the advantage of SNN few-shot learning [11], with the usage of small amount of labelled training data, is highlighted in the paper.

2 | RELATED WORK

In this part, we mainly introduce the related work of SNN model optimisation. There are two different goals in this part, which are to understanding biological systems and to pursue high computational performance [12]. We also give a brief introduction to the SNN training algorithm, which can be classified into four categories, mainly to solve the training problems caused by the non-differentiability of discrete spikes. We will analyse and summarise the advantages and disadvantages of the relevant work and then point out the advantages of our work.

In the process of biological learning, the topological structure of SNNs in the brain changes dynamically [13]. Research shows that the main sensory motor and visual areas have a relatively fixed core, which changes little over time, but they have flexible and frequently changing peripheral areas [15]. Based on this, some scholars use dynamic topology in the SNN model to improve the biological rationality [13, 14]. Wei Fang et al. [16] find different effects on neuronal dynamics caused by adjustments of the synaptic weight and the membrane time constants.

In the pursuit of computing performance, some researchers try to convert the network structure of CNN to SNN, while others try to change the topology of SNN. Hu Yangfan et al. [17] try to convert the residual network into deep SNN. By saturating the firing frequency of neurons to accelerate the response of deep neurons, so as to compensate the propagation error of large-scale network conversion, a spinning ResNet is constructed. In addition, the sparse convolutional network proposed by Loïc Cordone et al. [18] and Spiking YOLO proposed by Seijoon Kim et al. [19] are also inspired by CNN network structure and then converted to SNN. So far, the related work of converting the complex network structure to SNN losslessly, and processing time-domain information on the basis of solving the problem of non-differentiability of spike is not much, which has a lot of research space.

As far as the topology of SNN itself is concerned, SNN has three kinds of topology: feedforward type, recursive type and hybrid type [13]. Narayan Srinivasa and Youngkwan Cho's [20] self-organising spiking neural model for learning fault tolerant spatial motor transformations is a hybrid network, which can learn the spatial motion transformation of 2-DOF robot. At present, there is no great breakthrough in the dynamic topology of SNN. In many experiments, the recursive network which keeps high similarity with the DNN structure is still used.

The biggest difficulty of SNN is the non-differentiable problem of discrete spike. At present, there are four kinds of methods. One is to make SNN approximately differentiable, so as to use the gradient descent method. Backpropagation for networks of spiking neurons (Spikeprop) [21] and related derivatives proposed by Bohte et al. use a linear-similar method to solve the threshold triggering of non-differentiable problems. However, these methods suffer from the 'dead neuron' problem, which means there is no learning when no neuron spikes [22]. The gradient descent method is also considered. Chankyu Lee et al. [23] try to employ an approximate derivative for Leaky Integrated and Fire (LIF) [28] neuronal function, developing a spike-based supervised gradient descent back-propagation (BP) algorithm. The second is to learn by probabilistic reasoning. Y. Huang et al. [24] use the average spike count of neurons to express the time-varying posterior probability distribution for a hidden Markov model, which proves that SNN can be used for Bayesian reasoning [17]. The third is

the biological neuron-like training method based on synaptic plasticity rules. The fourth is ANN2SNN, that is to say, after indirectly training the conventional ANN, its parameters are directly applied to SNN. Because ANN has no time dimension, the transition from ANN to event-driven SNN usually costs information loss and performance loss. Zenke et al. [25] proposed a specific function as a differential substitution term. Lee et al. [26] used spike signal estimation at peak time etc., which was proposed and applied to multi-layer SNN network structure. But these methods ignore the dependence between spikes and do not consider the previous spike signal input.

In this work, we try to build network model with inception layer to ensure that deep SNN structure can avoid over-fitting problem and get high accuracy rate under the limitation of scant data. It is proved in our experiment that SpikeGoogle realises the conversion of inception from CNN to SNN and achieves good results on the NMNIST dataset [36]. When training the network, we choose to use the SLAYER frame and back-propagation algorithm, which can realise the SNN error back-propagation and allocate the error reliability in time [21].

3 | METHOD

In this section, we will mainly introduce the relative methods used in our work. Part A briefly introduces the basic process of SNN learning. Part B further gives the introduction of neuron model we used, Spike Response Model (SRM) [27]. Then, we demonstrate the error back-propagation algorithm in part C, which integrates spatial–temporal information and solves the non-differentiable problem of spike.

3.1 | SNN learning process

Spiking Neural Networks supervised learning algorithm is mainly to achieve the training of spike train and the update of parameters in complicated spatial–temporal domain. For each learning process, it can be divided into the following three stages.

First, convert the sample data into a spike train by using a frequency coding method such as Poisson encoder, or time coding method such as Latency encoder. Take Poisson encoder for example, it encodes the input data into a spike train with a distribution of firing times that conforms to a Poisson process. In the entire spike stream, the number of spikes appearing in mutually disjoint time steps is independent of each other, which has nothing to do with the starting point of the interval but is related to the time step. And the probability of sending a pulse is related to the pixel value of the input two-dimensional image. The Poisson distribution is suitable for describing the number of random events occurring per unit time. Latency encoding is an encoder that delays spiking based on input data. When the stimulus intensity is greater, the firing time is earlier, and there is a maximum spike firing time. Therefore, for each input data, a spike train with a time step of the maximum spike release time can be obtained and has only one spike. Second, the spike sequence

$$S = t^f : f = 1, 2, \ldots F \tag{1}$$

is input into the neural network which is run by applying the strategy of biological simulation mechanism to get the actual output spike sequence. Then, based on the target spike sequence, use the given loss function to calculate the network error, which is used to update synaptic weight:

$$w_{\text{new}} \leftarrow w_{\text{old}} + \Delta w \tag{2}$$

and axonal delay:

$$d_{\text{new}} \leftarrow d_{\text{old}} + \Delta d \tag{3}$$

Finally, judge whether the result reaches the set minimum error value, or the training epoch reaches the specified number. If not, carry out the next learning process. Figure 1 shows the SNN learning algorithm framework that combines the three main stages mentioned above.

3.2 | Neuron model

We use the Spike Response Model (SRM) [27], which fires spikes with a fixed threshold in our work. The model is an extension of the Integrated and Fire model (IF) [28], which is expressed by the explicit mathematical model. In SRM, neurons receive a bank of spike signals from all axon terminals connected to it, and each spike generates a postsynaptic potential. Under the influence of synaptic weights, the sum of these postsynaptic potentials constitutes a part of the membrane potential. Once the membrane potential exceeds the threshold, neuron fire spikes and enters the refractory period [38]. Figure 2 shows the detailed working process of an SRM model.

We use an ordered sequence to represent the time set of spikes:

$$\begin{aligned} \Gamma &= \{t^f : 1 <= f <= O\} \\ &= \{t | V(t) = V_{thresh} V'(t) > 0\} \quad [27] \end{aligned} \tag{4}$$

Then, the mathematical model of SRM can be described by the following formulas:

$$\begin{aligned} V(t) = &\sum_{f=1}^{O} \rho(t - t^f) + \sum_{i=1}^{N} \sum_{g=1}^{G_i} \omega_i \epsilon(t - t_i{}^g) \\ &+ \int_0^{\infty} {}^{+}\kappa(r) I(t - r) \mathrm{d}r \quad [27] \end{aligned} \tag{5}$$

$$\rho(s) = -V_{\text{thresh}} \exp\left(\frac{-s}{\tau_r}\right) H(s) \ [27] \tag{6}$$

$$\epsilon(s) = \left[\exp\left(\frac{-s}{\tau_m}\right) - \exp\left(\frac{-s}{\tau_s}\right)\right] H(s) \ [27] \tag{7}$$

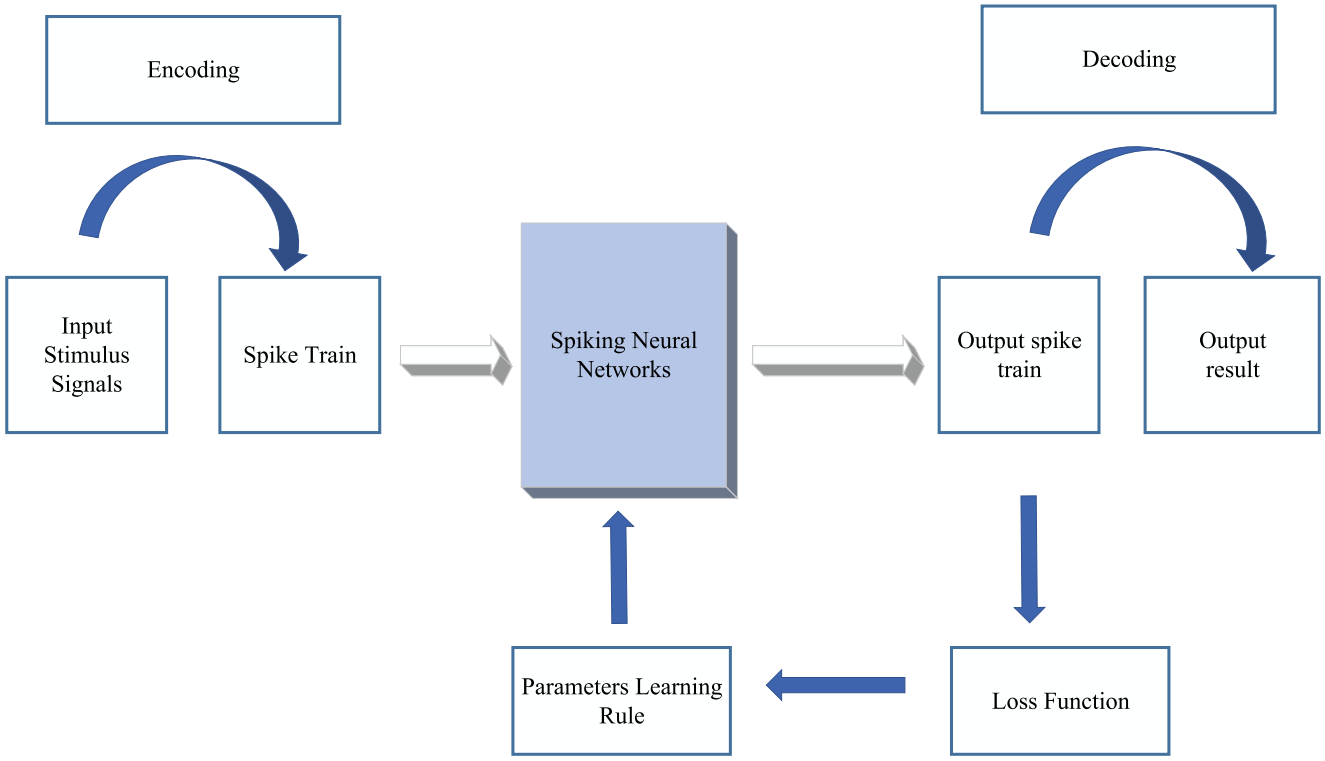

FIGURE 1 The Spiking Neural Network (SNN) training process includes the encoding process to transfer incoming signals into spike sequence, and forward propagation as well as error back-propagation process to realise the parameters of weight and delay update

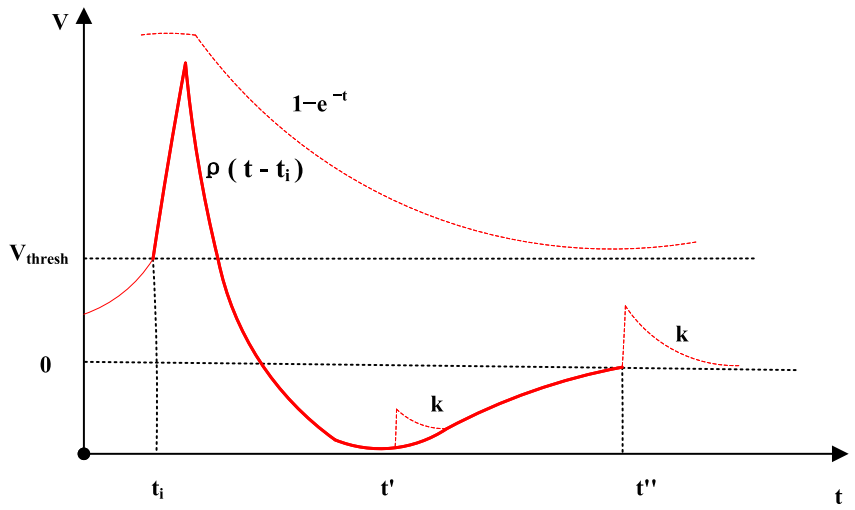

FIGURE 2 The working process of Spike Response Model (SRM) model is shown in the above figure, which is a schematic diagram obtained from Ref. [41]. During $0 - t_i$, the neuron received spike sequences from the end of the axon connected to it and the membrane potential accumulated. At t_i, the membrane potential exceeds the threshold, the neurons send out a spike and then enter the refractory period. From the peak of spike to t' is the absolute refractory period, and $t' - t''$ is the relative refractory period. $\rho(\bullet)$ is a refractory period function, which is used to simulate the recovery process of action potential in the biological neuron model. After a period of time, the neurons return to resting potential

Among them, $\underline{V}(t)$ is the membrane potential of neurons at time t, which is the sum of all postsynaptic potentials, refractory period responses and external inputs. $\rho(\bullet)$ is a refractory period function, which is used to simulate the recovery process of action potential in the biological neuron model. $\epsilon(\bullet)$ is the spike response function; 'N' is the total number of input synapses. G_i represents the total number of input spikes at the ith synapse. t_i^g indicates the generation time of the gth for neurons. $H(s)$ is the heavyside step function; tau_r, tau_m and, tau_s are time constants.

Considering the influence of axon delay on spike generation, the model should be extended to include axon delay by introducing axon delay into spike response function:

$$\epsilon_d(s) = \epsilon(s - d) \quad [27] \tag{8}$$

3.3 | Back-propagation

Based on the research studies on SNN back-propagation algorithms, in this section, we applied a better mechanism, SLAYER [8], in our work. SLAYER not only overcomes the non-differentiable problem of spike function, but also uses the temporal reliability allocation strategy to propagate the error back to previous layers and update parameters—synaptic weight and axonal delay. In the network training process, we learn the synaptic weight and axonal delay. In the process of back-propagation, SLAYER mainly achieved the following two advantages.

- The spike response nucleus introduces timing dependence, which allocates the influence of the spike to the future time value in the forward process, that is, the current membrane potential signal depends on the current and past input values.
- probability density function of the spike neuron state is used to approximate the derivative of the spike function, and stochastic neurons are used to approximate the error.

4 | NETWORK STRUCTURE

In this part, we give a specific explanation of the proposed new network structure SpikeGoogle, which is suitable for spike signal processing. By adding the innovation process of inception structure, we hope the readers can have a better understanding on the characteristics of SpikeGoogle.

4.1 | Important innovations in CNNs in recent years

In order to encourage people to improve CNN in the field of image classification, ImageNet Large-Scale Visual Recognition Challenge (ILSVRC competition or ImageNet competition) appeared. We notice that some champion networks made important innovations such as AlexNet, GoogLeNet and ResNet. There are some non-ignorable innovations as follows:

- The depth of the networks was getting deeper and deeper. After 2014, the depth of the network has exceeded 100 layers, and it has completely evolved into deep learning networks eventually: LeNet (7 layers, 1998, it was the first practical CNN in the history) [38], AlexNet (8 layers, 2012) [29], GoogLeNet (22 layers, 2014) [9], and ResNet (152 layers, 2015) [30].
- Inception modules and shortcut connections help networks perform better on the problem of vanishing or exploding gradient and ensure sufficient features for training deeper layers.

Before the appearance of GoogLeNet [9] in 2014, the champion networks in ImageNet competition paid attention on improving the method of LeNet (convolution layer, pooling layer and activation function), but later they proved the advantages of small convolution kernels and depth benefits. After the year of 2014, researchers' attention is shifted to CNN internal structure, which should not stack layers in the traditional way. So, a module that provides sufficient extraction and reduces the amount of computational demanding exponentially was proposed, Inception [29]. After that, a lot of other networks reproposed including ResNet (2015) [31], DenseNet (2016) [32] and SENet (2017) [33], which were inspired by inception module and shortcut connections [34].

The inception module was used for achieving better predicted performance and short training time, which can solve the problem of over-fitting. And our network is suitable for its goal to be designed shallow and energy friendly.

4.2 | Inception structure

The general idea of CNN is to use stacked convolution layers (a convolution kernel followed normalisation and max pooling layer) and fully connected layers to extract features, that is, making the network deeper and broader brings greater fitting potential. But it brings inevitable side results that a large number of increased parameters not only need more computing resources, which is also easy to cause over-fitting. Meanwhile, deep network structure could cause vanishing gradient problem easily.

Inception structure is a popular deep learning structure proposed by Christian in 2014 [9], which reduces the degree of vanishing gradient problem of the network and computing resources demanding. Next, we will mainly introduce the specific structure of this high-performance network architecture. Inspired by the idea of network in Ref. [35], they designed a 2 hidden layer CNN-based structure called the inception module.

- First layer: two convolution kernels (1 × 1) and a max pooling layer (3 × 3) reducing channel dimensions remove computational bottlenecks.
- Second layer: three different convolution kernels (1 × 1, 3 × 3, and 5 × 5) extract distinct features and another dimension reduction convolution (1 × 1) keeps non-linear properties.

To transfer higher abstraction from the two hidden layers to next layer, a spatial concatenation could make it possible [9].

4.3 | SpikeGoogle

Based on the above analysis, the application of inception in spatial–temporal sparse NMNIST dataset should have a certain result, and the amount of network parameters should not be too much. We call the single-inception network after spike conversion as SpikeGoogle. The model structure of SpikeGoogle (x3) is shown in Figure 3 and Figure 4. We show the inception structure, including convolutional kernels of different sizes in the upper right corner of Figure 3. Structurally, the SpikeGoogle (x3) model consists of the following components: three spiking layers, one max pooling layer and one fully connected layer. After spike conversion, three consecutive perceptions, considered as the third part of whole network, can properly extract the features of different receptive fields of NMNIST. Then, it is supposed to output the results through fully connected layer after minimising the parameters. Since the accuracy of SCNN (x3) is higher than that of SCNN (x2) and SCNN (x4), three-layer depth is enough to deal with NMNIST classification.

It is worth noting that we add the postsynaptic potential (PSP) function behind a pooling layer (the last layer in the SpikeGoogle network), fully connected layer (SNN based) and every SpikeGoogle module to process the extracted features by combining the temporal information of NMNIST's sparse data.

Furthermore, in our work, the SRM neuron model is used to simulate the PSP process, as detailed in the neuron model section above.

5 | EXPERIMENTS

In this section, we will first introduce the specific characteristics of the NMNIST dataset. Then, the results of SpikeGoogle will be shown later, with SCNN as its comparison item. This part can be divided into the following three experiments: SCNN model with different layers, mixed model of SpikeGoogle and SCNN, and SpikeGoogle model with different numbers.

5.1 | Dataset and settings

Inspired by the biological visual processing mechanism, neuro morphological vision sensors capture the light intensity changes in the field of vision and generate asynchronous time flow. Representative neuromorphological vision sensors include Dynamic Vision Sensor (DVS) and Dynamic Active Imaging Sensor (Davis) [36].

The NMNIST dataset [36] consists of MNIST images, which contain 10 different hand writing digits, 0–9. These images are converted into spike datasets by using the DVS moving on the Pan/Tilt/Zoom (PTZ). The dataset contains 'on' and 'off' spikes with a duration of about 300 milliseconds. When simulating motion on a computer, moving the sensor instead of the scene or image is a biologically more realistic method to eliminate the timing artefacts introduced by the monitor update. By using the perfect computer vision dataset, we save a lot of time and effort in selecting and collecting topics. Furthermore, the size of each sample event is 34 × 34. This kind of small size helps to reduce the processing time, which is good for rapid testing and algorithm iteration in the prototype design of new ideas. It can be said that the NMNIST dataset is the most complete DVS dataset created so far.

In the training process, the event information is represented as point cloud model with four channels of '$x\ y\ p\ t$' [36]. The parameters of '$x\ y$' include the coordinate information of the point cloud 'p' is the polarity of the event and 't' represents the time domain of event. Figure 5 gives two examples of digit 0 and 4. The dataset used in this paper is a small NMNIST dataset ('small' means that we did not use the full dataset), with 10 ways * 100 shoots. The separation of training and test samples of small data set is the same as that of standard NMNIST 60,000 training samples and 10,000 test samples.

Our network has been tested with CUDA libraries version 9.2 and GCC 7.3.0 on Ubuntu 18.04. The SLAYER frame [8] on which our work is based includes C++ and CUDA code that has to be compiled and installed before it can be used from Python.

We have made changes to the training methods based on the original. When using the Adam optimiser, we sometimes change hyper parameters, such as learning rate, so it is hard to give a definitive guidance to the most effective single way to train the networks. Our code can be found in the link https://github.com/WangXuan2401/SpikeGoogle.

5.2 | Spiking convolutional neural network

Figure 6 shows the SCNN (x3) training results, which tend to be over-fitting. We propose an assumption for the undesired result that the SCNN (x3) structure is too superior to process NMNIST point cloud data, meaning there are a lot of extraneous features being extracted. In order to verify the assumption, we choose to change the number of SCNN structure. Based on the idea of shallow layer, delete or add the layer of SCNN (x3), meaning to train in the network of SCNN (x2) or SCNN (x4).

Figure 7 shows the result of modified network SCNN (x2) and SCNN (x4), which are not as good as that of the original SCNN (x3) in terms of convergence speed and accuracy. However, it is worth noting that after 80 epochs, the accuracy of the SCNN (x2) training is higher than that of original SCNN (x3), which is still on the rise. Then, we continue to analyse the training result of SCNN (x4) structure, whose accuracy has been difficult to break through 0.5 though the overall upward trend is similar to SCNN (x3). It is shown that the process of NMNIST data on deep network become subsided, that is, most of sparse data features are discarded.

Based on the above experiments, we draw some conclusion that the SCNN structure can extract the sparse features of NMNIST data well and the training result of 4 layers and deep network is not as good as that of shallow network.

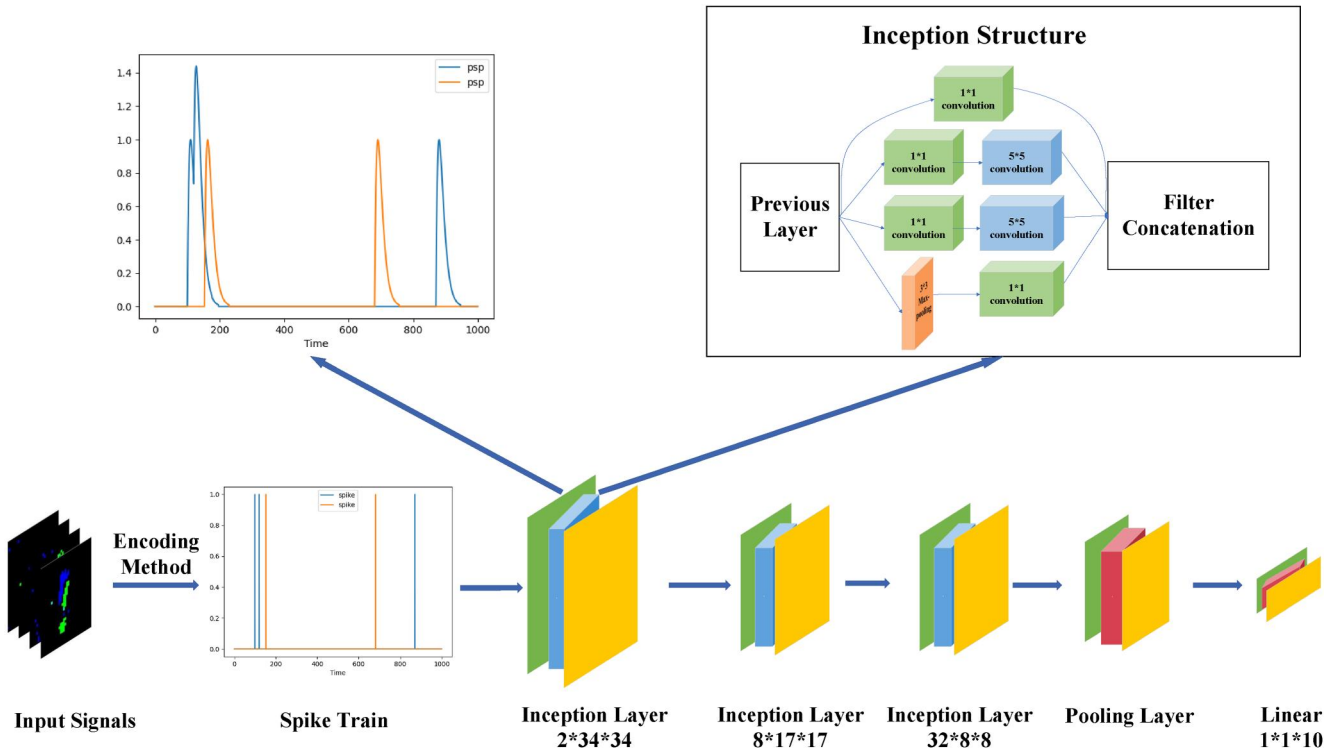

FIGURE 3 As shown in the figure, our SpikeGoogle (x3) model consists of three parts: three inceptions, one layer of max-pooling and a full connected layer. Three inceptions can extract the features of different receptive fields of NMNIST. The PSP function is added behind every module. The input signals are encoded first. Processed by inceptions, they will then be output by fully connected layer after minimising parameters

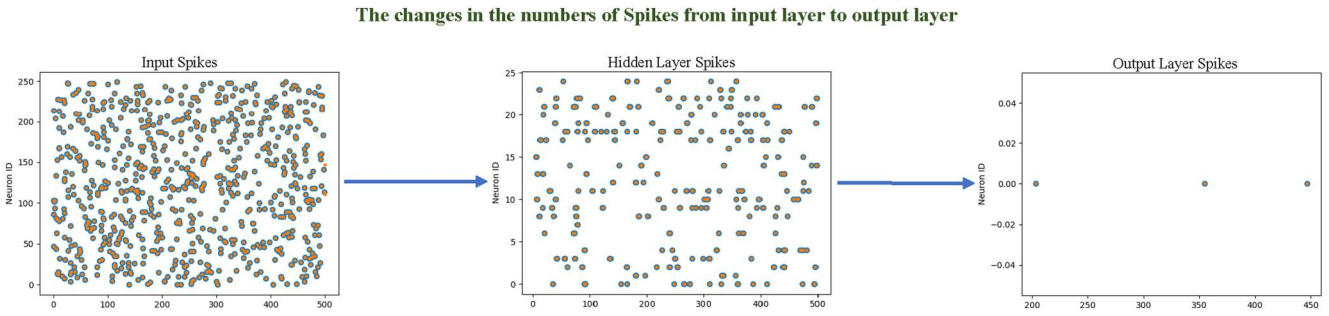

FIGURE 4 The input spikes are described as point cloud chart. After being processed, the spikes become sparse in the hidden layer. At last, what the fully connected layer outputs are linear spikes

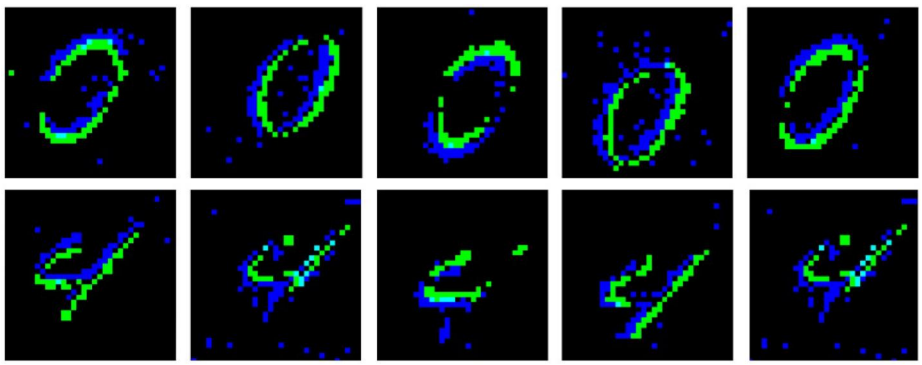

FIGURE 5 This figure shows the result of the dataset we used. A single event contains four channels of $x\ y\ p\ t$ information, $x\ y$ determines the coordinates of the point cloud in the graph, and two different colours represent the positive and negative polarity of the event. For time domain variable t, we capture different images of an event at five time points and arrange them into a column in time order to present the time domain information in the NMNIST dataset [36]. The first row is the number '0', and the second row is the number '4'

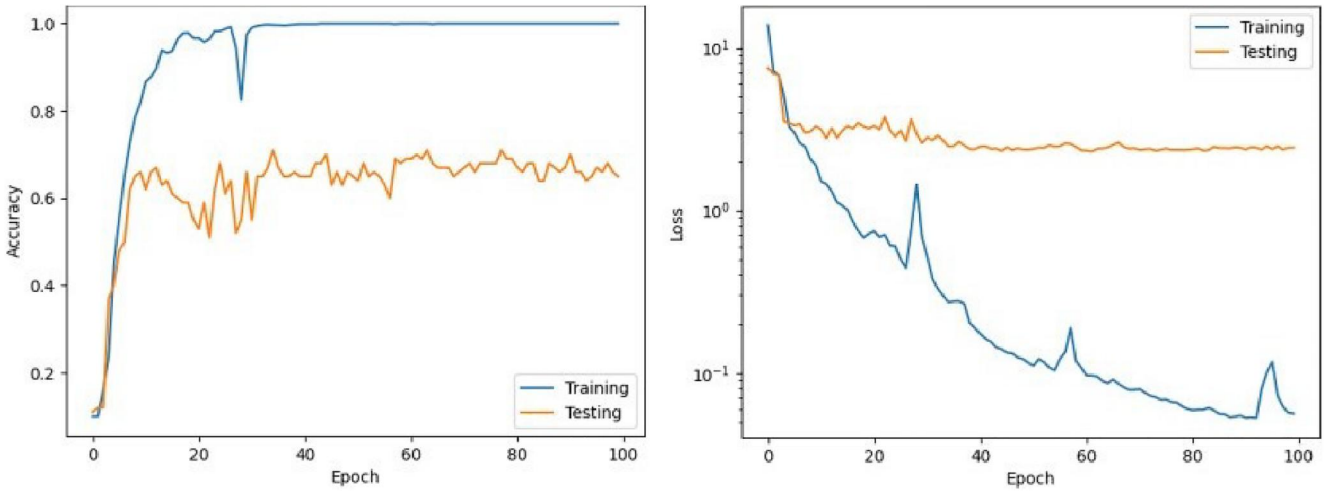

FIGURE 6 Performances of spiking convolutional neural network (SCNN) (x3), where 'x3' denotes three layers

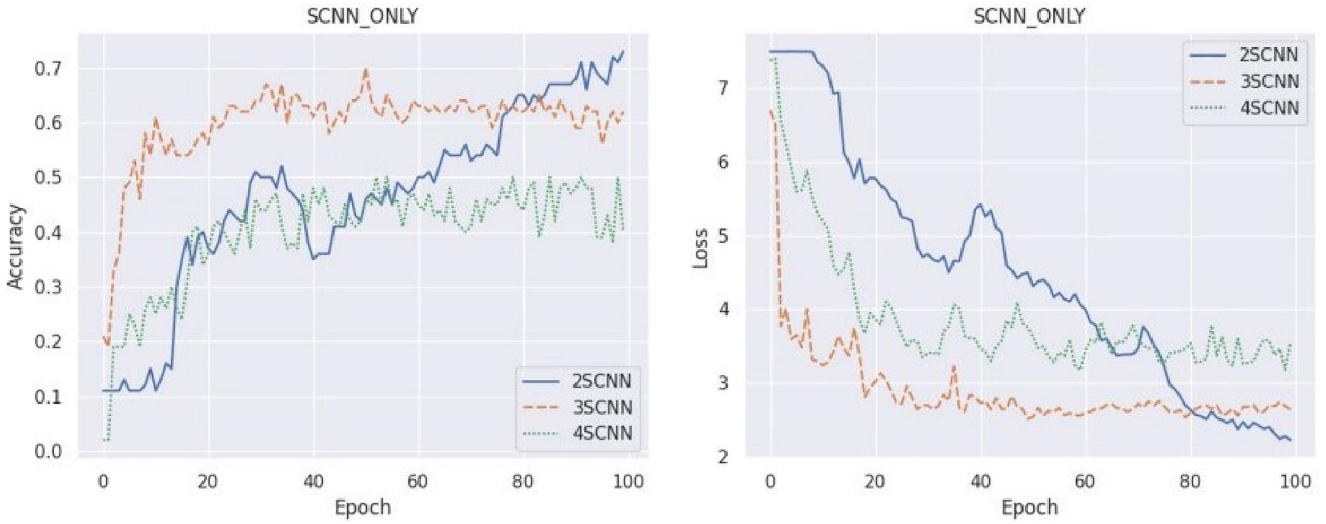

FIGURE 7 Performances of Spiking Convolutional Neural Network (SCNN) with different layers

5.3 | Mixed model of SpikeGoogle and SCNN

Figure 8 shows the result of SCNN (x1)-SpikeGoogle (x1) and SpikeGoogle (x1)-SCNN (x1) structure in test accuracy and loss. We combine SpikeGoogle with SCNN structure by the following two kinds of modes. One is the structure of SpikeGoogle lying in front of SCNN. And the other one is the upside-down mode, meaning SpikeGoogle lies after SCNN. From the figure, we can see that about 40 epochs, the accuracy rate of these two kinds of structures have exceeded that of SCNN (x3). The highest accuracy rate of SCNN (x1) + SpikeGoogle (x1) is 78%, and the convergence speed of it is not slower than that of SCNN (x3). It is worth noting that the training speed of the structure is greatly increased (in the same GPU environment), and the number of parameters has been greatly reduced.

Therefore, we draw a conclusion that SpikeGoogle structure has higher utilisation of NMNIST's sparse features than that of SCNN, which is verified in its improvement of performance in network training speed and accuracy.

5.4 | SpikeGoogle

Figure 9 demonstrates the test accuracy and loss of different numbers of SpikeGoogle structure, including SpikeGoogle (x2), SpikeGoogle (x3), and SpikeGoogle (x4). The compression and extraction of NMNIST data features are both realised in SpikeGoogle.

By analysing the data in Table 1, we find that the training result of SpikeGoogle (x4) is greatly reduced, which is similar to the training result of SCNN (x4). At this point, our previous analysis of the reasons for the performance degradation

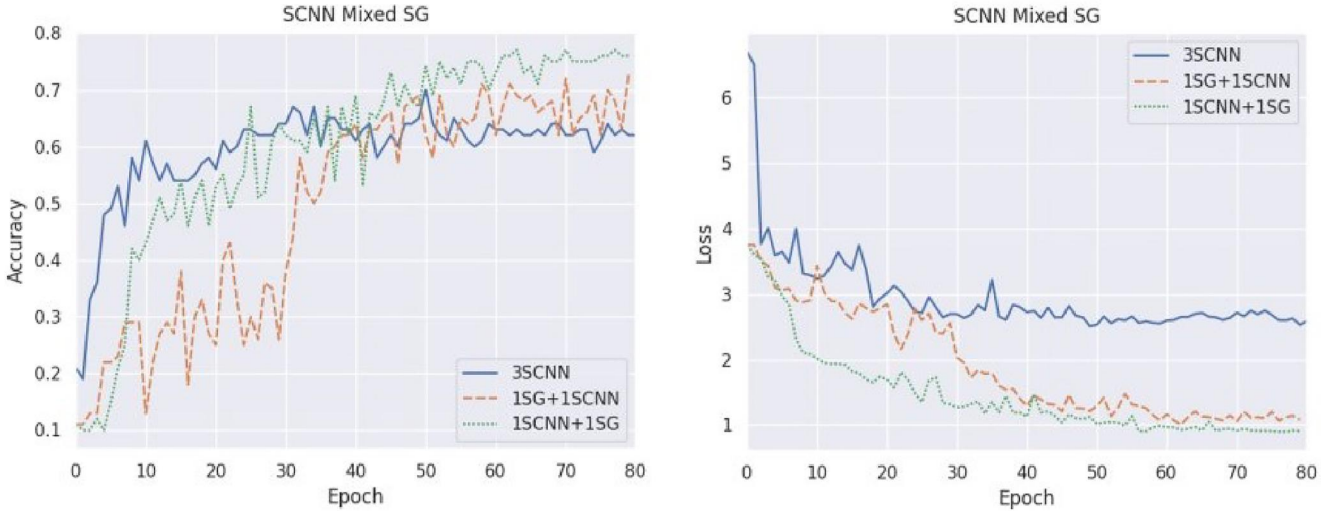

FIGURE 8 Performances of mixed model of Spiking Convolutional Neural Network (SCNN) and SpikeGoogle

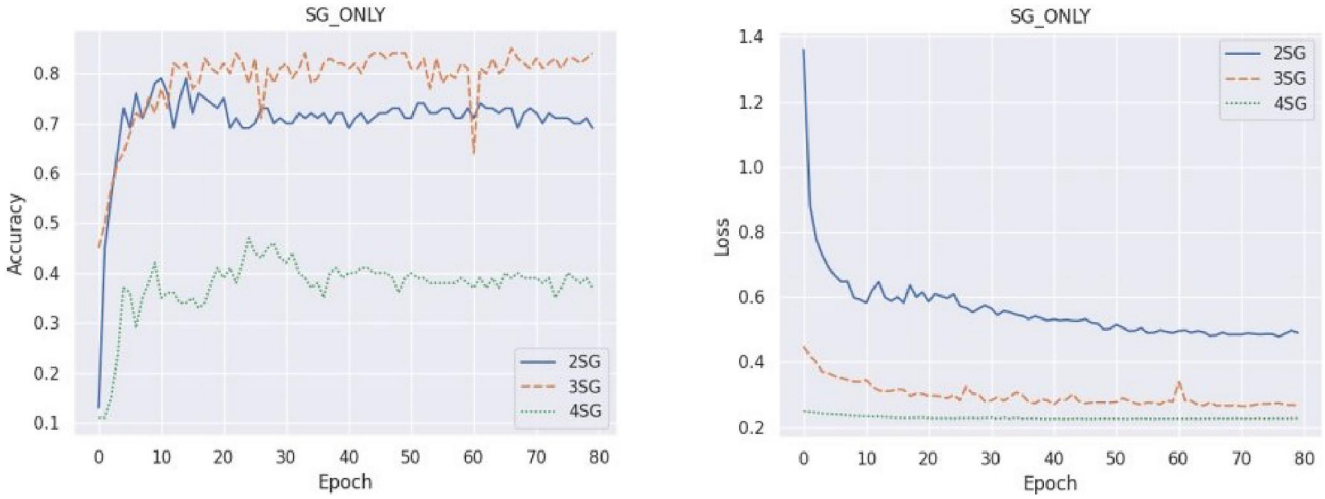

FIGURE 9 Performances of SpikeGoogle with different layers

of deep network is further confirmed that deep network process NMNIST data has subsided. And also, it still has other problems, which are not found in research studies to date.

In addition, we can find that the convergence speed and accuracy of SpikeGoogle, including SpikeGoogle (x2) and SpikeGoogle (x3), have been improved to a certain extent on the basis of shallow layer. In particular, the performance of SpikeGoogle (x3) has been improved significantly, which realises 85%.

And test accuracy is almost maintained at more than 80% in epoch. So, this further verifies our idea that the SpikeGoogle network can not only perform better with the use of NMNIST's sparse features but also be competent for the classification task of NMNIST.

Daniel Neil and Shih-Chii Liu [37] used to do the similar things. They converted the framed-based networks to spiking neural networks [37] as described in Ref. [40]. It is described that the experiment is based on spiking inputs, which are 1k events of N-MINST. Frame-based networks are trained by three different data processing methods before conversion.

TABLE 1 Comparison between Spiking Convolutional Neural Network (SCNN) and SpikeGoogle

Network	Accuracy (%)	Time (s/epoch)
SCNN (x2)	0.73	38.5
SCNN (x3)	0.7	48.3
SCNN (x4)	0.5	35.6
SCNN (x1) + SpikeGoogle (x1)	0.77	29.5
SpikeGoogle (x1) + SCNN (x1)	0.73	49.2
SpikeGoogle (x2)	0.79	18.8
SpikeGoogle (x3)	**0.85**	**15.2**
SpikeGoogle (x4)	0.47	17.3

TABLE 2 Comparison with other work

Network	Dateset	Accuracy (%)
Intensity	N-MNIST (uncentred, 1k)	0.2288
	N-MNIST (centred, 1k)	0.4274
Canny	N-MNIST (uncentred, 1k)	0.2317
	N-MNIST (centred, 1k)	0.4351
N-MNIST	N-MNIST (uncentred, 1k)	0.4750
	N-MNIST (centred, 1k)	0.6450
SpikeGoogle (x3) (ours)	Small N-MNIST (1k)	**0.85**

The names of networks in the first column in Table 2 are datasets used when training the unconverted networks. Intensity refers to the dataset obtained by extracting the spikes from the image with a probability proportional to the intensity of the pixel. Canny refers to NMNIST where the Canny edge filter is applied to the static image before training. NMNIST refers to the original NMNIST dataset. Table 2 shows the performance of different networks on centred and uncentred NMNIST data. It can be seen that SpikeGoogle (x3) achieve better performance.

6 | CONCLUSION

In this paper, we have presented SpikeGoogle based on the GoogLeNet-like inception module by using the SLAYER framework and stacking multiple inception structures. Compared with the traditional SCNN, which only involves common convolution operation, SpikeGoogle is able to achieve the highest accuracy of 85% on a small NMNIST dataset and can also improve the training speed. However, as the number of network layers increases, the training results will deteriorate, which is due to the sparsity of the data that is easy to diverge in the training process. Therefore, this reminds us that we need to balance the gap between training speed and accuracy while improving the network.

ACKNOWLEDGEMENT

This project is sponsored by Key-Area Research and Development Program of Guangdong Province, No. 2020B0404020005.

CONFLICT OF INTEREST

The authors declared that they have no conflicts of interest to this work.

DATA AVAILABILITY STATEMENT

The dataset NMINST that this work used are openly available in Dropbox at https://www.dropbox.com/sh/tg2ljlbmtzygrag/AABrCc6FewNZSNsoObWJqY74a?dl=0.

ORCID

Xuan Wang https://orcid.org/0000-0002-1144-1587

REFERENCES

1. Perez-Carrasco, J.A., et al.: Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing—application to feedforward ConvNets, IEEE Trans. Pattern Anal. Mach. Intell. 35, 2706–2719 (2013)
2. Roy, K., Jaiswal, A., Panda, P.: Towards spike-based machine intelligence with neuromorphic computing. Nature 575, 607–617 (2019)
3. Deng, L, et al.: Rethinking the performance comparison between SNNS and ANNS, Neural Network 121, 294–307 (2020)
4. Haessig, G., et al.: Spiking optical flow for event-based sensors using IBM's TrueNorth neurosynaptic system. arXiv:1710.09820v1 (2017)
5. Lam, M.W.Y., et al.: Gaussian process Lstm recurrent neural network language models for speech recognition. In: International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE (2019)
6. Vidal, A.R., et al.: Ultimate SLAM? Combining events, images, and IMU for robust visual SLAM in HDR and high-speed scenarios. IEEE Robot. Autom. Lett. 3(2), 994–1001 (2018)
7. Cao, Y., Chen, Y., Khosla, D.: Spiking deep convolutional neural networks for energy-efficient object recognition. Int. J. Comput. Vis. 113, 54–66 (2015)
8. Shrestha, S.B., Orchard, G.: Slayer: Spike layer error reassignment in time. arXiv:1810.08646 (2018)
9. Szegedy, C., et al.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9 (2015)
10. Paugam-Moisy, H., Bohte, S.M.: Computing with Spiking Neuron Networks. Springer Berlin Heidelberg (2012)
11. Wang, Y., et al.: Generalizing from a few examples: a survey on few-shot learning. ACM Comput. Surv. 53(3) (2020)
12. Tie-Lin, Z., Bo, X.: Research advances and perspectives on spiking neural networks. Chin. J. Comput. 0–21 (2020)
13. Taherkhani, A., et al.: A review of learning in biologically plausible spiking neural networks. Neural Network. 122, 253–272 (2020)
14. Belatreche, A., Paul, R.: Dynamic cluster formation using populations of spiking neurons. In: The 2012 International Joint Conference on the Neural Networks, pp. 1–6 (2012)
15. Wysoski, S.G., Benuskova, L., Kasabov, N.: Evolving spiking neural networks for audiovisual information processing. Neural Network. 23(Sept), 819–835 (2010)
16. Fang, W., et al.: Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In: International Conference on Computer Vision (ICCV), pp. 2661–2671 (2021)
17. Yangfan, H., Huajin, T., Gang, P.: Spiking deep residual network. arXiv:1805.01352v2 (2020)
18. Cordone, L., Miramond, B., Ferrante, S.: Learning from event cameras with sparse spiking convolutional neural networks. arXiv:2104.12579v1 (2021)
19. Kim, S., et al.: Spiking-YOLO: Spiking neural network for energy-efficient object detection. arXiv:1903.06530v2 (2019)
20. Srinivasa, N., Cho, Y.: Self-organizing spiking neural model for learning fault-tolerant spatio-motor transformations. IEEE Transact. Neural Networks Learn. Syst. 23, 1526–1538 (2012)
21. Xianghong, L., Zuzheng, G.: Review of modeling methods for single chamber pulsed neurons. Comput. Eng. Appl. J. 41–44 (2011)
22. Xiao, M., et al.: Training feedback spiking neural networks by implicit differentiation on the equilibrium state. arXiv:2109.14247 (2021)
23. Lee, C., et al.: Enabling spike-based backpropagation for training deep neural network architectures. Front. Neurosci. 14, 119 (2020)
24. Huang, Y., Rao, R.P.: Bayesian inference and online learning in Poisson neuronal networks. Neural Comput. 1503–1526 (2016)
25. Zenke, F., Ganguli, S.: Superspike: Supervised learning in multilayer spiking neural networks. arXiv preprint arXiv:1705.11146 (2017)
26. Lee, J.H., Delbruck, T., Pfeiffer, M.: Training deep spiking neural networks using backpropagation. Front. Neurosci. 10, 508 (2016)
27. Javed, A., et al.: Predicting networks-on-chip traffic congestion with spiking neural networks. J. Parallel Distr. Comput. 154, 6 (2021)

28. Ganguly, C., et al.: First level approximation of measured neuronal response using a leaky integrate and fire model. In: IEEE 16th India Council International Conference (INDICON). IEEE (2020)
29. Krizhevsky, A., Ilya, S., Geoffrey, E.: Imagenet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. 1097–1105 (2012)
30. He, K., et al.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and pattern Recognition, pp. 770–778 (2016)
31. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7132–7141 (2018)
32. Huang, G., et al.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4700–4708 (2017)
33. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning, pp. 448–456 (2015)
34. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)
35. Min, L., Qiang, C., Shuicheng, Y.: Network in network. arXiv preprint arXiv:1312.4400 (2013)
36. Orchard, G., et al.: Converting static image datasets to spiking neuromorphic datasets using saccades. Front. Neurosci. 9(Oct), 437 (2015)
37. Neil, D., Liu, S.: Effective sensor fusion with event-based sensors and deep network architectures. In: International Symposium on Circuits and Systems (ISCAS). IEEE (2016)
38. Wulfram, G.: Time structure of the activity in neural network models. Phys. Rev. 738–758 (1995)
39. Brown, R.E., Bligh, T.W., Garden, J.F.: The Hebb synapse before Hebb: Theories of synaptic function in learning and memory before, with a discussion of the long-lost synaptic theory of William McDougall. Front Behav. Neurosci. (2021)
40. Diehl, P.U., et al.: Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In: International Joint Conference on Neural Networks (IJCNN) (2015)
41. Gerstner, W.: Spike-response model. Scholarpedia 3(12), 1343 (2008)

How to cite this article: Wang, X., et al.: SpikeGoogle: Spiking Neural Networks with GoogLeNet-like inception module. CAAI Trans. Intell. Technol. 7(3), 492–502 (2022). https://doi.org/10.1049/cit2.12082