Effective Active Learning Method for Spiking Neural Networks

Xiurui Xie, Bei Yu, Guisong Liu, Qiugang Zhan, and Huajin Tang

***Abstract*— A large quantity of labeled data is required to train high-performance deep spiking neural networks (SNNs), but obtaining labeled data is expensive. Active learning is proposed to reduce the quantity of labeled data required by deep learning models. However, conventional active learning methods in SNNs are not as effective as that in conventional artificial neural networks (ANNs) because of the difference in feature representation and information transmission. To address this issue, we propose an effective active learning method for a deep SNN model in this article. Specifically, a loss prediction module ActiveLossNet is proposed to extract features and select valuable samples for deep SNNs. Then, we derive the corresponding active learning algorithm for deep SNN models. Comprehensive experiments are conducted on CIFAR-10, MNIST, Fashion-MNIST, and SVHN on different SNN frameworks, including seven-layer CIFARNet and 20-layer ResNet-18. The comparison results demonstrate that the proposed active learning algorithm outperforms random selection and conventional ANN active learning methods. In addition, our method converges faster than conventional active learning methods.**

***Index Terms*— Active learning method, deep learning, feature representation, spiking neural network (SNN).**

I. Introduction

SPIKING neural networks (SNNs) are known as the "third-generation neural networks," which simulate the information transmission mechanism of biological neurons and add time dimensions into the network [1]. Similar to the neurotransmitters transfer mechanism of biological neuron communication, SNNs send messages in the network by transmitting and receiving discrete spike signals. When the integrated membrane potential of a neuron reaches the threshold, this neuron will emit a spike. This event-driven mechanism allows SNNs to consume less energy [2]. SNNs have a low-power advantage and can fully utilize space–time events, which can better promote the development of artificial intelligence.

SNN models have achieved much success in recent years [3]. For instance, Wu et al. [4] proposed a fast direct training model for large-scale SNNs on AlexNet image classification. Wu et al. [5] proposed an ASF-BP training method to train deep SNNs by accumulated spiking flow and applied it to neuromorphic datasets. Zhou et al. [6] proposed a temporally coded deep SNN with easy training and robust performance. Then, Liu et al. [7] proposed an effective address event representation (AER) object classification method in SNNs using segmented probability-maximization learning. Most of these existing methods focus on improving the accuracy of classification models. Another kind of approach designed credited assignment strategies to address the nondifferentiable problem in SNNs [2], [8]. More related works are in [9], [10], [11], [12], [13], and [14].

Although SNNs have recently achieved much success, training deep SNN models is still expensive because of the high annotation cost compared to conventional deep learning models. Active learning is proposed to address this issue by selecting useful samples to be labeled instead of random sampling [15].

Different active learning approaches have been proposed in artificial neural networks (ANNs) based on different sample selection criteria [16], [17], [18]. For instance, the uncertainty approach defines and measures the quantity of uncertainty to select uncertain data points [19], [20], [21], [22]. The diversity approach selects diverse data points that represent the whole distribution of the unlabeled pool [16], [23], [24]. Expected model change selects data points that will cause the greatest change to the current model parameters or outputs if we know their labels [25], [26], [27]. Learning loss for active learning (LLAL) is a task-agnostic active learning method that attaches a loss prediction module to the target task model proposed in [28]. Batch active learning actively selects a batch of instances to query the corresponding labels for learning an accurate model [15], [19], [29]. More related works are in [16], [30], [31], and [32].

Active learning in conventional ANNs has achieved much success in recent years. However, these methods are not directly applicable to SNNs because of the difference in feature representation and information transmission between SNNs and conventional artificial networks (ANNs).

To address this issue, we explore an effective active learning method for SNNs in this article. Specifically, we propose an active loss prediction model for deep SNNs, which receives the characteristics of the hidden layer of the SNN model and

Manuscript received 13 September 2021; revised 14 March 2022, 26 May 2022, 18 December 2022, and 20 February 2023; accepted 13 March 2023. Date of publication 24 March 2023; date of current version 4 September 2024. This work was supported in part by the Natural Science Foundation of Guangdong Province under Grant 2021A1515011866, in part by the Sichuan Province under Grant 2021YFG0018 and Grant 2022YFG0314, and in part by the Social Foundation of Zhongshan Sci-Tech Institute under Grant 420S36. *(Corresponding author: Guisong Liu.)*

Xiurui Xie, Bei Yu, and Qiugang Zhan are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China.

Guisong Liu is with the School of Computing and Artificial Intelligence, Southwestern University of Finance and Economics, Chengdu 611130, China, and also with the Zhongshan Institute, University of Electronic Science and Technology of China, Zhongshan 528400, China (e-mail: lgs@uestc.edu.cn).

Huajin Tang is with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China, and also with the Zhejiang Laboratory, Hangzhou 311122, China (e-mail: htang@zju.edu.cn).

Digital Object Identifier 10.1109/TNNLS.2023.3257333

2162-237X © 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on November 14,2024 at 11:20:27 UTC from IEEE Xplore. Restrictions apply.

processes discrete spike information. The active loss prediction model takes full account of the information transmission characteristics in SNNs and can select the data with high classification contribution and is more suitable for SNN models. In addition, we build SNN models with different depths, i.e., the CIFARNet of seven layers and ResNet-18 of 20 layers. We apply the active loss prediction model to classification tasks and evaluate it on MNIST, Fashion-MNIST, SVHN, and CIFAR-10 with various network structures. The experimental results demonstrate the superiority of the proposed method compared with conventional state-of-the-art methods in ANNs.

The main contribution of this article is that we propose a new active learning strategy that learns discrete spike features for SNNs. Besides, we derive an active learning algorithm based on surrogate gradient learning for the proposed model. Experimental results on different tasks demonstrate that the proposed method effectively reduces the size of the training set and experimental cost, and our method converges faster than conventional active learning methods.

The rest of this article is arranged as follows. Section II introduces the preliminary approach, including the leaky-integrate-and-fire (LIF) model and the LLAL active learning algorithm. Section III introduces the proposed active learning method, including the active learning strategies and learning algorithm. Then, the experimental results and analysis are presented in Section IV. Finally, Section V summarizes conclusions and future works.

II. Preliminary

In this section, we introduce the LIF spiking neuron model we use in this work and the related LLAL active learning algorithm.

A. LIF Spiking Neuron Model

LIF is one of the most widely used spiking models in recent years because of its simplicity and effectiveness [33]. It abstracts the biological nervous system at a high level and simplifies the precise mechanism of ion activity. The LIF takes the whole neuron as a functional unit and uses a resistor–capacitor (RC) circuit to simulate the relationship between its membrane voltage and the input spikes.

The basic circuit of the LIF model is shown in Fig. 1. We see that inside the RC circuit, the driving current $I(t-t_j)$ consists of two components

$$I(t) = I_R + I_C. \tag{1}$$

The first component is the resistance current I_R through the linear resistor R_m, and $I_R = V(t)/R_m$ is calculated by Ohm's law. The second component I_C is used to charge the capacitor C_m. According to the definition of capacitance, $C = q/u$, where q is the charge and u is the voltage. Then, the capacitor current $I_C = C_m \cdot dV(t)/dt$, i.e.,

$$I(t) = \left(\frac{V(t)}{R_m}\right) + C_m \cdot \left(\frac{dV(t)}{dt}\right). \tag{2}$$

Multiply (2) by R_m and introduce the membrane time constants $\tau_m = R_m C_m$. The equations of the processes of charging, discharging, and resetting of LIF neurons can be obtained as follows:

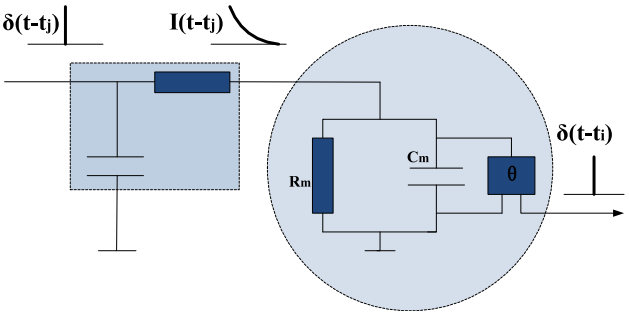


Fig. 1. LIF spiking neuron model. The rectangular dotted line frame represents the low-pass filter, and the circular dotted line frame represents the circuit diagram of the LIF model.

$$U^{l,t} = V^{l,t-1} + \frac{1}{\tau_m}(V_{\text{reset}} - V^{l,t-1} + W^l S^{l-1,t}) \tag{3}$$

$$V^{l,t} = U^{l,t}(1 - S^{l,t}) + V_{\text{reset}} S^{l,t} \tag{4}$$

$$S^{l,t} = g(U^{l,t}, \theta) = \begin{cases} 1, & if\ (U^{l,t} - \theta) \geq 0 \\ 0, & if\ (U^{l,t} - \theta) < 0 \end{cases} \tag{5}$$

where $U^{l,t}$ denotes the membrane potential of the lth layer neuron at the tth time step and $V^{l,t}$ is the leaky voltage from the previous time step. W^l denotes the weight between layers $l-1$ and l. $S^{l-1,t}$ denotes the corresponding spike of the (l-1)th layer neuron at the tth time step, which is binary (0 or 1). When $t = 0$, $U^{l,0}$ and $S^{l,0}$ are initialized to 0. As shown in (4), the leaky voltage $V_{l,t}$ is set as $U_{l,t}$ if the neuron did not fire, i.e., $S^{l,t} = 0$; otherwise, the voltage is reset as V_{reset}, whose process is called hard reset [34]. $g(x)$ is the Heaviside step function, and its output is only 0 or 1, which is used to determine the spike emission of SNNs.

B. Loss Prediction Active Learning Method

LLAL is a task-agnostic framework proposed in [28]. The core part of the active learning algorithm is called the loss prediction module. This module is attached to the target model, and its input is the feature map of the hidden layer of the target model. Each feature map is simplified into a fixed-length feature vector through the global average pooling (GAP) layer and the full connection layer. At the stage of active sample selection, the loss prediction module is carried out on the unlabeled dataset. The top k data with the largest predicted loss value are selected for manual labeling and then added to the training set for the next iteration.

The loss of LLAL consists of target loss L_{target} and loss prediction loss L_{loss}. The target loss is calculated by the output of target model $\hat{y}$ and real label y through cross entropy, denoted as $\mathcal{L} = L_{\text{target}}(\hat{y}, y)$. In the loss prediction module, its output is a predicted loss denoted as $\hat{\mathcal{L}}$, and the target loss is regarded as the real loss. B samples in the training set are divided into $B/2$ sample pairs $x^p = (x_i, x_j)$, the superscript p also represents the pair of $(i; j)$, and the loss prediction module is learned by considering the loss prediction difference of a data pair. The loss function of the loss prediction module, named pair loss, and the final loss function of the whole model

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on November 14,2024 at 11:20:27 UTC from IEEE Xplore. Restrictions apply.

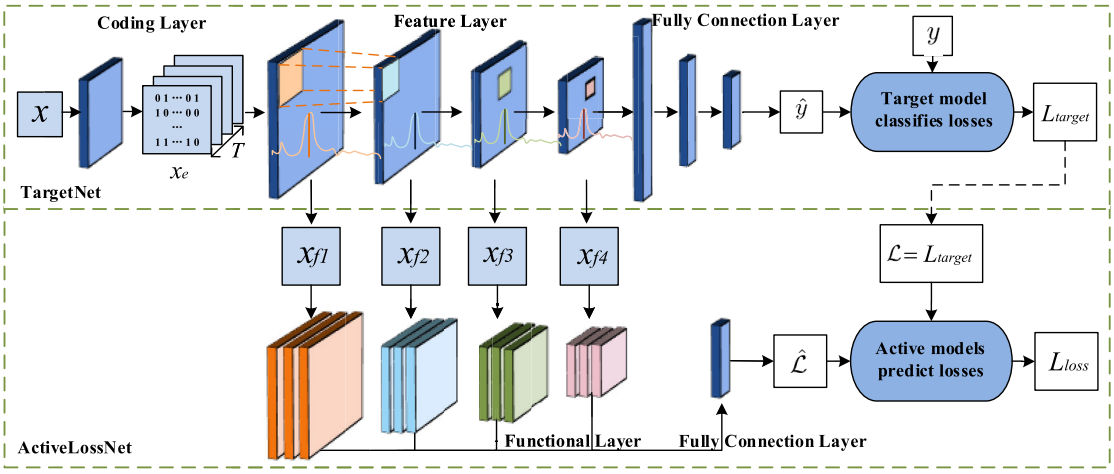


Fig. 2. SAL model structure, which includes the target classification prediction module TargetNet and active loss prediction module ActiveLossNet.

are defined as follows:

$$L_{\text{loss}}(\hat{\mathcal{L}}_p, \mathcal{L}_p) = \max(0, -\mathbb{I}(\mathcal{L}_i, \mathcal{L}_j) \cdot (\hat{\mathcal{L}}_i - \hat{\mathcal{L}}_j) + \varepsilon) \quad (6)$$

$$\mathbb{I}(\mathcal{L}_i, \mathcal{L}_j) = \begin{cases} +1, & \mathcal{L}_i > \mathcal{L}_j \\ -1, & \text{otherwise} \end{cases} \quad (7)$$

$$\text{Loss} = \frac{1}{B} \sum_{(x,y)\in B} L_{\text{target}}(\hat{y}, y) - \frac{2\lambda}{B} \times \sum_{(x^p,y^p)\in B} L_{\text{loss}}(\hat{\mathcal{L}}_p, \mathcal{L}_p) \quad (8)$$

where $\mathcal{L}_i$ and $\mathcal{L}_j$ are the losses of x_i and x_j, respectively, ε is a positive scalar, and λ is a scalar used to control the loss proportion of the loss prediction module. The overall loss Loss of the whole model in (8) is the combination of the conventional classification loss $L_{\text{target}}(\hat{y}, y)$ and the prediction loss $L_{\text{loss}}(\hat{\mathcal{L}}_p, \mathcal{L}_p)$. Different from the LLAL, we propose a novel ActiveNet module for SNNs, which consists of spike-based functional layers and adaptive loss function. Besides, an SNN active learning algorithm is derived in this article.

III. Our Method

In this section, we introduce the proposed active learning method for SNNs, including the model framework, loss function, and learning algorithm.

A. Spiking Active Learning Model

The proposed spiking active learning (SAL) model is composed of two parts: a target network TargetNet and loss prediction model ActiveLossNet, as shown in Fig. 2.

1) TargetNet: It is designed for classification as a conventional method, including a feature extraction layer and a fully connected layer. Each layer is composed of spike neurons that receive and output binary spike sequences. In the feature extraction layer, the input is convolved first, and the neuron membrane potential is calculated and updated by (3)–(5).

Each convolutional feature layer has output feature x_{fi} that is fed into the fully connected layer, and the final output of the fully connected layer is the firing rate of each output neuron

$$\hat{p}_{y_i} = \frac{1}{T} \sum_{t=1}^{T} S_i^t \quad (9)$$

where $\hat{p}_{y_i}$ denotes the probability for class i with label y_i, S_i^t is the output spike of class i at time t, which is calculated by (5), and T is the maximum spike time.

2) ActiveLossNet: It is designed in our method to predict the classification loss for SNNs. As shown in Fig. 2, ActiveLossNet contains a functional layer and a fully connected layer, and the input of ActiveLossNet is the features in the TargetNet model. Taking x_{fi} as an example, in the functional layer, x_{fi} is first processed through an average pooling layer and then two fully connected layers. This process is represented by the following formula:

$$x_{fi}^l = g\left(W_A^l x_{fi}, \theta\right) \quad (10)$$

where x_{fi}^l represents the output of the fully connected layer and W_A^l denotes the weight matrix in layer l of ActiveLossNet. The g function is calculated as (5). The processed feature spike sequences are fed into the last fully connected layer, and the final output $\hat{\mathcal{L}}$ is calculated by

$$\hat{\mathcal{L}} = W_A^L \sum_{t-1}^{T} S^t \quad (11)$$

where W_A^L is the weight matrix of the final layer in ActiveLossNet.

3) Loss Function: In our network framework, we optimize TargetNet and ActiveLossNet simultaneously. The classification loss of the TargetNet and the predicted loss of the ActiveLossNet are expressed as L_{target} and L_{loss}, respectively. The loss function in our framework is defined as

$$E = L_{\text{target}}(\hat{p_{y_i}}, p_y) + \lambda L_{\text{loss}}(\hat{\mathcal{L}}, \mathcal{L}) \quad (12)$$

where $\mathcal{L} = L_{\text{target}}(\hat{p_{y_i}}, p_y)$ is the classification loss of TargetNet and $\hat{\mathcal{L}}$ is the output of ActiveLossNet. The learning loss of the TargetNet is

$$L_{\text{target}}(\hat{p_{y_i}}, p_y) = \frac{1}{2}\left(\hat{p_{y_i}} - p_y\right)^2 \quad (13)$$

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on November 14,2024 at 11:20:27 UTC from IEEE Xplore. Restrictions apply.

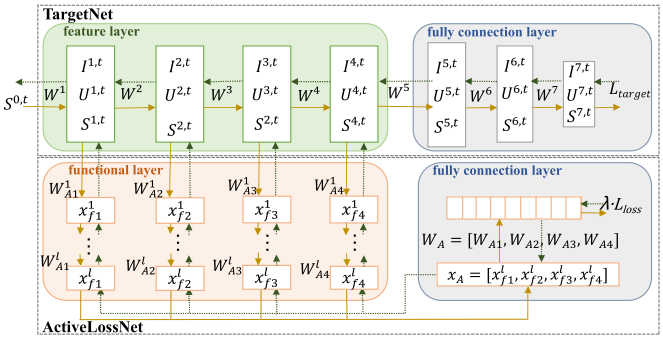


Fig. 3. Forward and backpropagation process of the TargetNet and ActiveLossNet models.

where $\hat{p}_{y_i}$ is the predicted label probability and p_y is the ground-truth one-hot probability.

We also take the mean square error as ActiveLossNet's loss function, as shown in the following equation:

$$L_{\text{loss}}(\hat{\mathcal{L}}, \mathcal{L}) = \frac{1}{2}(\hat{\mathcal{L}} - \mathcal{L})^2. \tag{14}$$

Different from the pair loss in LLAL [28], the mean square error provides a stricter constraint, making it more suitable for the characteristics of SNNs.

B. Active Learning Algorithm

In the active learning process, the error is backpropagated to each layer, and the weight parameters in the TargetNet and ActiveLossNet are updated in both temporal and spatial dimensions. The forward and backpropagation processes of the TargetNet and ActiveLossNet are shown in Fig. 3. It is worth noting that the parameters of feature layers in the TargetNet are affected not only by the error backpropagated from TargetNet but also from ActiveLossNet.

In (12), TargetNet and ActiveLossNet are optimized simultaneously. Considering both the TargetNet loss and ActiveLossNet loss, we have

$$\frac{\partial E}{\partial W^l} = \frac{\partial L_{\text{target}}}{\partial W^l} + \frac{\partial L_{\text{loss}}}{\partial W_A^l}.$$

The gradient of the TargetNet loss is

$$\frac{\partial L_{\text{target}}}{\partial W^l} = \sum_{t=1}^{T} \frac{\partial L_{\text{target}}}{\partial S^{l,t}} \frac{\partial S^{l,t}}{\partial U^{l,t}} \frac{\partial U^{l,t}}{\partial W^l}. \tag{15}$$

The first factor of (15) is different in the output layer and the hidden layer. In the output layer

$$\frac{\partial L_{\text{target}}}{\partial S^{L,t}} = \frac{1}{T}\left(\frac{1}{T}\sum_{t=1}^{T} S^{L,t} - p_y\right)$$

and in the hidden layer

$$\begin{aligned}\frac{\partial L_{\text{target}}}{\partial S^{l,t}} &= \frac{\partial L_{\text{target}}}{\partial S^{l+1,t}} \frac{\partial S^{l+1,t}}{\partial U^{l+1,t}} \frac{\partial U^{l+1,t}}{\partial S^{l,t}}, \quad (l = 1, \ldots, L-1) \\ &= \frac{\partial L_{\text{target}}}{\partial S^{l+1,t}} g'\left(U^{l+1,t}, \theta\right) \frac{W^{l+1}}{\tau_m}.\end{aligned}$$

Because the Heaviside step function $g(x)$ is nondifferential, we use the surrogate gradient as

$$g'(x) = (1 - \text{sigmoid}(x))\text{sigmoid}(x)$$

with

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}.$$

The third factor in (15) is derived as

$$\begin{aligned}\frac{\partial U^{l,t}}{\partial W^l} &= \frac{\partial U^{l,t}}{\partial V^{l,t-1}} \frac{\partial V^{l,t-1}}{\partial W^l} + \frac{1}{\tau_m} S^{l-1,t} \\ &= \left(1 - \frac{1}{\tau_m}\right)\left(\frac{\partial V^{l,t-1}}{\partial U^{l,t-1}} \frac{\partial U^{l,t-1}}{\partial W^l} + \frac{\partial V^{l,t-1}}{\partial S^{l,t-1}} \right. \\ &\qquad \left. \times \frac{\partial S^{l,t-1}}{\partial U^{l,t-1}} \frac{\partial U^{l,t-1}}{\partial W^l}\right) + \frac{S^{l-1,t}}{\tau_m} \\ &= \left(1 - \frac{1}{\tau_m}\right)\left[1 - S^{l,t-1} + g'(U^{l,t-1}, \theta) \right. \\ &\qquad \left. \times \left(V_{\text{reset}} - U^{l,t-1}\right)\right] \frac{\partial U^{l,t-1}}{\partial W^l} \\ &\quad + \frac{S^{l-1,t}}{\tau_m}.\end{aligned} \tag{16}$$

To simplify (16), let

$$\begin{aligned}M(l,t) = \left(1 - \frac{1}{\tau_m}\right)&\left[1 - S^{l,t-1} + g'(U^{l,t-1}, \theta) \right. \\ &\left. \times \left(V_{\text{reset}} - U^{l,t-1}\right)\right].\end{aligned}$$

Then, we continuously derive gradients across the time dimension. When $t = 1$

$$\frac{\partial U^{l,t}}{\partial W^l} = \frac{S^{l-1,1}}{\tau_m}$$

then

$$\begin{aligned}\frac{\partial U^{l,t}}{\partial W^l} &= M(l,t) \frac{\partial U^{l,t-1}}{\partial W^l} + \frac{S^{l-1,t}}{\tau_m} \\ &= \prod_{\tau=1}^{t} M(l,\tau) + \sum_{\tau=1}^{t-1} \prod_{i=\tau}^{t} M(l,i) \frac{S^{l-1,\tau}}{\tau_m} + \frac{S^{l-1,t}}{\tau_m}.\end{aligned}$$

In the ActiveLossNet, the output gradient is

$$\frac{\partial L_{\text{loss}}}{\partial W_A^L} = \frac{\partial L_{\text{loss}}}{\partial \hat{\mathcal{L}}} \frac{\partial \hat{\mathcal{L}}}{\partial W_A^L}. = (\hat{\mathcal{L}} - \mathcal{L}) x_A. \tag{17}$$

In the fully connected layer of the grouping processing of the four spike sequences, the gradient is

$$\begin{aligned}\frac{\partial L_{\text{loss}}}{\partial W_A^l} &= \frac{\partial L_{\text{loss}}}{\partial \hat{\mathcal{L}}} \frac{\partial \hat{\mathcal{L}}}{\partial x_{fi}^l} \frac{\partial x_{fi}^l}{\partial W_{Ai}^l}, \quad i = (1, 2, 3, 4, \ldots) \\ &= (\hat{\mathcal{L}} - \mathcal{L}) W_{\text{Ai}}^l g'\left(W_{\text{Ai}}^l x_{\text{fi}}^{l-1}, \theta\right) x_{fi}^{l-1}.\end{aligned}$$

Since the input spike sequence of the ActiveLossNet model comes from the TargetNet middle layer, ActiveLossNet's error will also affect the weight parameters of the first four layers of TargetNet during backpropagation. Thus, the derivation is shown as follows:

$$\frac{\partial E}{\partial W^l} = \frac{\partial E}{\partial x_{\text{fl}}} \frac{\partial x_{\text{fl}}}{\partial W^l} + \sum_{k=l+1}^{m} \frac{\partial L_{\text{loss}}}{\partial x_k} \frac{\partial x_k}{\partial S^{l,t}} \frac{\partial S^{l,t}}{\partial W^l}$$

where m is the number of feature layers from TargetNet. In general, the main difference between the SNN active algorithm and the conventional ANN algorithm is that we use surrogate gradient in our method to address

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on November 14,2024 at 11:20:27 UTC from IEEE Xplore. Restrictions apply.

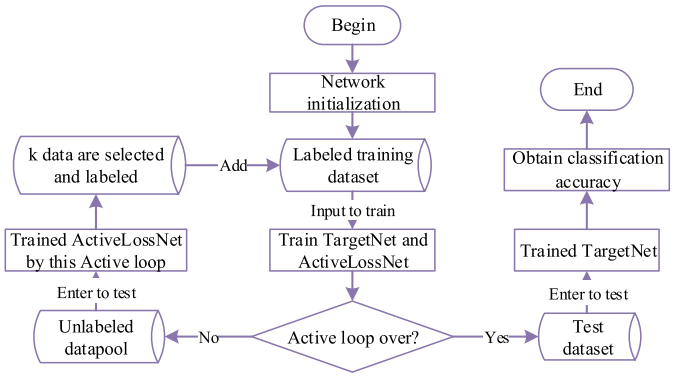


Fig. 4. Active learning workflow. The network models TargetNet and ActiveLossNet are trained simultaneously. In the initialization, k data are randomly selected and labeled as the initial training set. The number of active learning iterations is set as cycles. At the end of each iteration, newly selected k data are labeled and added to the training set. After the completion of all training, the size of the marked training dataset is cycles $\times$ k.

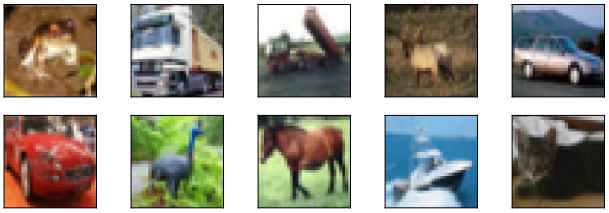


Fig. 5. Sample images in the CIFAR-10 dataset. The database contains the following categories: "airplane," "automobile," "bird," "cat," "deer," "dog," "frog," "horse," "ship," and "truck".

the nondifferentiable problem in the spike generation function.

The active learning workflow is shown in Fig. 4. First, we initialize the model. k data are randomly selected from the unlabeled pool to be labeled as the initial training data. Then, TargetNet and ActiveLossNet are optimized simultaneously. If the epoch does not reach the maximum learning iterations, the active learning model ActiveLossNet is obtained after the ith active iteration. We use ActiveLossNet to predict the classification loss to the unlabeled samples and select the top k data that have the highest error. After labeling, these selected samples are added to the training set as the input of the next iteration.

IV. Experiments

In this section, we first describe the datasets used in the experiment. Then, we introduce two SNN structures, i.e., CIFARNet and ResNet-18. Finally, we conduct comparison experiments on different datasets and SNN models and analyze the comparative experimental results.

A. Datasets

1) CIFAR-10: The CIFAR-10 dataset is composed of 60 000 tiny color images, all of which are $32 \times 32 \times 3$ in size and are divided into ten classes [35]. CIFAR-10 is an unbalanced dataset, and the number of images in each class is not uniform, which requires better performance of our model. One-sixth of this dataset is used as the test set for the model, while the rest are used as the training set. Fig. 5 shows some samples.



Fig. 6. Sample images in the SVHN dataset.



Fig. 7. Sample images in the Fashion-MNIST dataset.

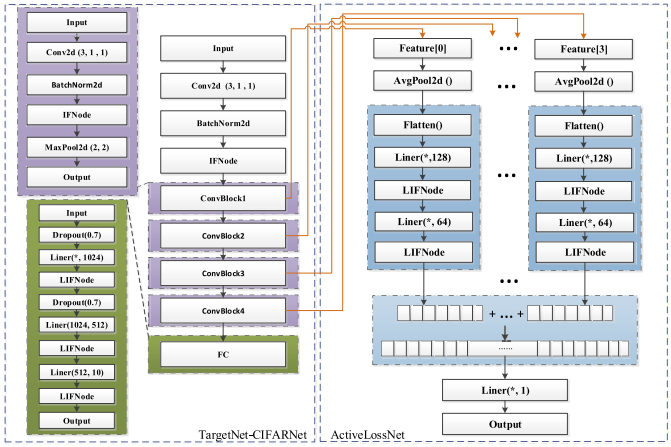


Fig. 8. Network structure of the active SNN-CIFARNet model, including a TargetNet-CIFARNet and a ActiveLossNet. The notation $*$ represents the vector size obtained after the last convolution layer, which is expanded into one dimension. Numbers in brackets of the convolution layer represent kernel, stride, and padding.

2) SVHN: It is also a database of numbers, containing 73 257 pieces of training data and 26 032 pieces of test data [36]. However, unlike MNIST, it collects digital images from real scenes, such as street views and house numbers. As shown in Fig. 6, the quality of data collected from real scenes is often uneven and affected by uncertainties, such as shooting light intensity, differences in letter wear, font style, and background interference.

3) MNIST: The MNIST dataset is a relatively simple dataset [37]. It contains 70 000 binary handwritten digits, and the digits range from 0 to 9 in a total of ten categories. Among them, 60 000 images are used as the training set, and 10 000 images are used as the test set.

4) Fashion-MNIST: The Fashion-MNIST dataset inherits all the attributes of the MNIST dataset, which is still 70 000 grayscale images of 28×28 [38]. However, different from the simple handwritten digital images of MNIST, Fashion-MNIST has collected a variety of Fashion items: clothing, shoes, bags, and so on. The images in these data are more complex, and some samples are shown in Fig. 7.

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on November 14,2024 at 11:20:27 UTC from IEEE Xplore. Restrictions apply.

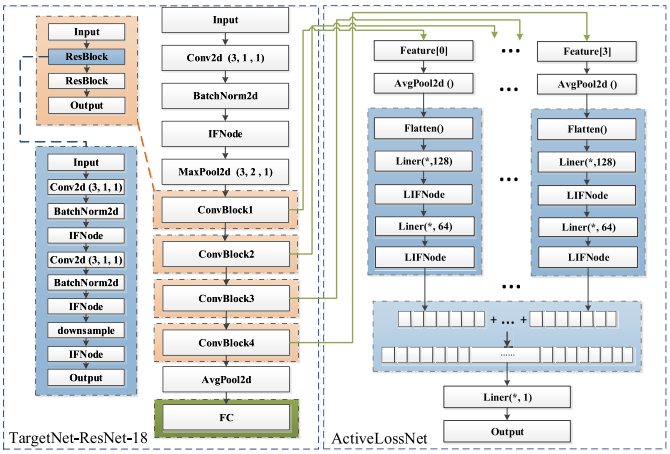


Fig. 9. Network structure of the active SNN-ResNet18, including a TargetNet-ResNet-18 and a ActiveLossNet. The notation $*$ represents the vector size obtained after the last convolutional layer, which is expanded into one dimension. Numbers in brackets of the convolution layer represent kernel, stride, and padding.

B. Deep SNNs' Models

Two popular deep SNN models, i.e., SNN-CIFARNet and SNN-ResNet18, are tested on comparison experiments to evaluate the effectiveness of the proposed method.

1) SNN-CIFARNet: The SNN-CIFARNet model is a typical neural network structure, which is composed of four convolutional layers and three fully connected layers. We take an additional convolution layer as the encoder of the SNN network and then add the batch norm (BN) layer and IFNode layer after all the convolution layers. The intermediate spiking output of these four convolutional layers is the input data of ActiveLossNet. After the remaining three fully connected layers, we also insert the dropout and LIFNode layers. The network structure diagram of CIFARNet is shown in Fig. 8.

2) SNN-ResNet18: Inspired by the ANN ResNet model proposed in 2015, there has been some research that transferred the trained ANN ResNet into SNN [39], [40], [41], [42], [43], [44]. In this article, we utilize the residual structure and design an SNN-ResNet18 that can be trained directly to avoid the gradient disappearance caused by the deep model, as shown in Fig. 9.

The SNN-ResNet18 network is divided into the input encoding, intermediate hidden layer, and output voting. We consider the convolution layer in the input layer as an encoder. The middle hidden layer is composed of four basic blocks. Each block contains two convolution layers, two BN layers, and three IF layers, which are used to accumulate the membrane potential. The output features of the four block layers are input into the loss prediction model of active learning as the input data of the active model. We change the original output layer to three fully connected layers and add dropout and LIF layers.

C. Experimental Setup

1) Hyperparameters: In the experiments, the membrane potential time constant τ_m is set as 2.0, and the neural firing threshold θ and resting potential V_{reset} are set to 1.0 and 0.0, respectively. The SNNs model time window T is set as 12. We train our networks with a maximum of 160 epochs and an initial learning rate of 0.00002. In the training process, we dynamically adjust the learning rate to avoid the loss concussion phenomenon at the later stage of training, which affects the model convergence. When the epoch achieves 110 and 140, the learning rate is multiplied by 0.1. We set a total of ten rounds of active learning iterations, and the active learning data increment k is set to 1000. After each iteration, k data are selected from the unlabeled dataset and added to the training set, which is used as the input data of the model in the next iteration. In each iteration, the SNN model and the active loss prediction model are retrained.

2) Baselines: Because there is no existing SNN active learning method, we compare our method with SNN models trained without active learning and the corresponding conventional ANN active learning method LLAL. The LLAL is chosen because it is also a loss prediction model as our method.

D. Comparison Results

1) Comparison on SNN-CIFARNet: In this section, we conduct comparison experiments on the SNN-CIFARNet architecture on the CIFAR-10, MNIST, and Fashion-MNIST datasets. To demonstrate the effectiveness of the proposed active learning strategy, we compare our method with the random sample selection method and the directly applied LLAL (Orignal LLAL) active learning baseline.

The results on MNIST are shown in the left of Fig. 10. We see that when the training dataset reaches 10 000, our active learning method, random selection method, and LLAL method have similar classification accuracies of 98.77%, 98.2%, and 97.51%, respectively. However, at the initial stage of active learning, the quantity of training data is small, and our method shows better performance. It can first select the data suitable for the SNN model and add it to the training set to provide more information support for the next active learning iteration. With the increase in labeled data volume, the classification accuracy gap between our method and random selection and LLAL gradually decreases.

The results of Fashion-MNIST are shown in the middle of Fig. 10. The average accuracy of our proposed active learning method for SNNs reached 90.92% in the final stage of active learning (the training data reach 10 000), which is 0.44% higher than the result of random selection and 6% higher than the LLAL. In the whole process of active learning, our method has obvious advantages. In addition, our method converges faster than conventional active learning methods.

The comparative experimental results of CIFAR-10 are shown in the right of Fig. 10. The difference between the classification accuracy of the active learning method and the random selection method we designed gradually decreases in several training iterations. After the tenth iteration of active learning, the accuracy rate of the active learning method we designed reaches 81.44%, which is 3.41% higher than that of the random selection method. Compared with LLAL, our method always maintains an advantage, finally reaching 26.74%. It can be seen from the experiment that the active loss prediction model we designed can capture the characteristics of the discrete data output from the hidden layer of the SNN model and effectively select the data that are more valuable to the SNN model.

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on November 14,2024 at 11:20:27 UTC from IEEE Xplore. Restrictions apply.

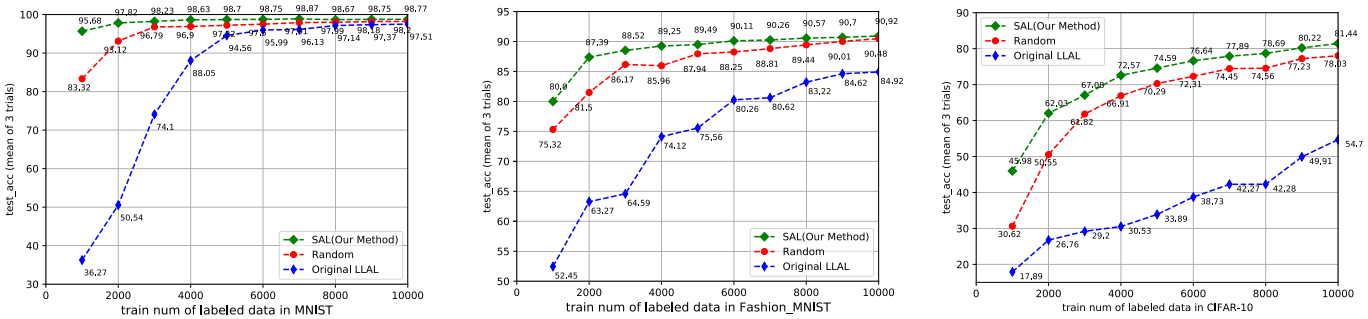


Fig. 10. Comparison results of SNN-CIFARNet on the MNIST, Fashion-MNIST, and CIFAR-10 datasets.

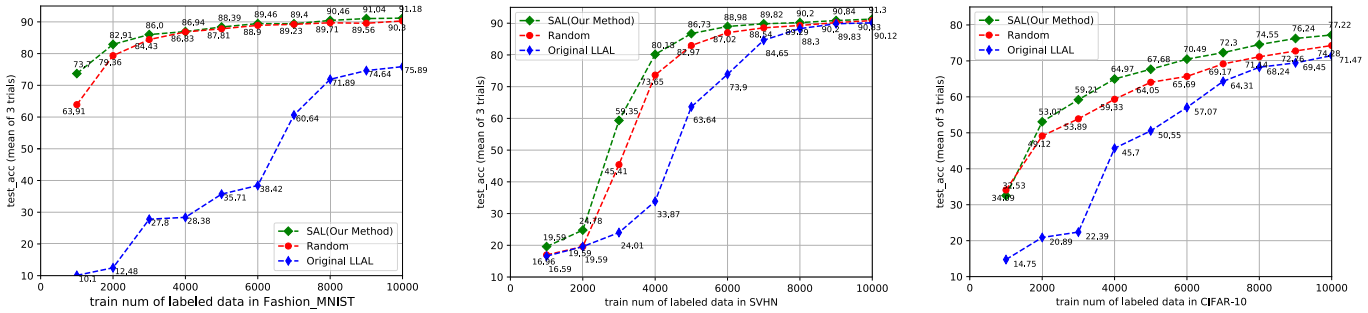


Fig. 11. Comparison results of SNN-ResNet18 on the Fashion-MNIST, SVHN, and CIFAR-10 datasets.

2) Comparison on SNN-ResNet18: The comparative experimental results are shown in Fig. 11. We see that as in previous experiments, our method outperforms the random selection method and the LLAL on all three datasets, with accuracy in the final stage of active learning (when the training set reached 10 000) 90.95%, 91.3%, and 76.43% for three tasks. In addition, we see that as the number of training samples increases, the difference between active learning and random selection decreases. This is because when there are enough labeled samples, even the random selection model can obtain good performance. However, when the number of samples is insufficient and the label is high cost, the active learning method helps the model select the most valuable samples to label.

3) Comparison on the Latest SNN Model: To give more comprehensive comparison, we further test our method on the current state-of-the-art SNN model on CIFAR-10 [45]. For a fair comparison, we use the same experimental setup as the reference, in which there are three convolution layers, two pooling layers, and two fully connected layers. The Conv2d has 256 out channels, kernal size $=3$, stride $=1$, and the pooling layer has kernel size $=2$ and stride $=2$.

The comparison results are shown in Fig. 12. As in previous conclusions, our method outperforms the random selection model across a different number of training samples, and this advantage is more significant with less training data.

To give a more comprehensive evaluation, we test the recall, precision, and F1-score of each class and take the average value listed in Table I. The results show that as in previous conclusions, our SAL outperforms random selection on all of these three criteria across a different number of training samples from 1000 to 10 000. This indicates that the SAL model can not only predict the full degree of positive samples correctly but also has high accuracy in positive class recognition. A high F1-score demonstrates a good overall advantage of model precision and recall of the proposed method.

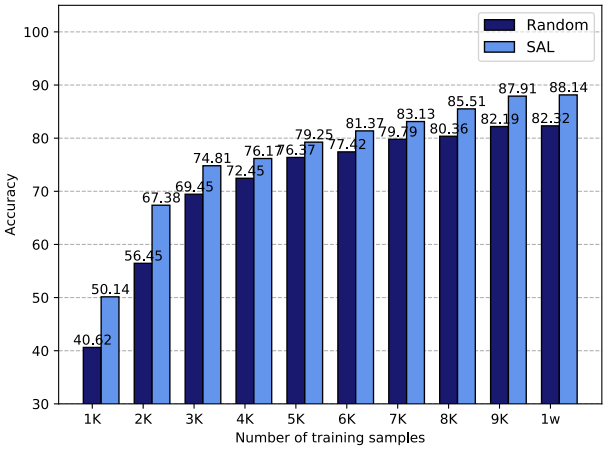


Fig. 12. Comparison results on the SOTA SNN model.

TABLE I
COMPARISON RESULTS ON SOTA MODEL

Train size	*Recall*		*Precision*		*F1-score*	
	Random	**SAL**	Random	**SAL**	Random	**SAL**
1,000	30.80	**43.07**	41.56	**50.23**	36.34	**44.57**
2,000	48.9	**57.43**	56.62	**68.25**	51.10	**63.16**
3,000	60.73	**66.63**	70.05	**75.38**	64.35	**69.09**
4,000	64.18	**67.93**	72.53	**76.36**	68.30	**71.28**
5,000	68.00	**70.04**	77.03	**80.17**	71.64	**73.91**
6,000	68.95	**72.98**	77.58	**82.29**	72.34	**76.95**
7,000	71.78	**75.38**	80.72	**83.39**	75.35	**77.95**
8,000	71.34	**76.29**	80.94	**85.65**	75.64	**80.11**
9,000	73.59	**78.14**	82.33	**88.08**	78.06	**83.77**
10,000	72.65	**81.14**	80.05	**88.34**	78.18	**82.14**

4) Ablation Experiments: Two extension ablation experiments are conducted to further explore the effectiveness of the proposed strategies.

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on November 14,2024 at 11:20:27 UTC from IEEE Xplore. Restrictions apply.

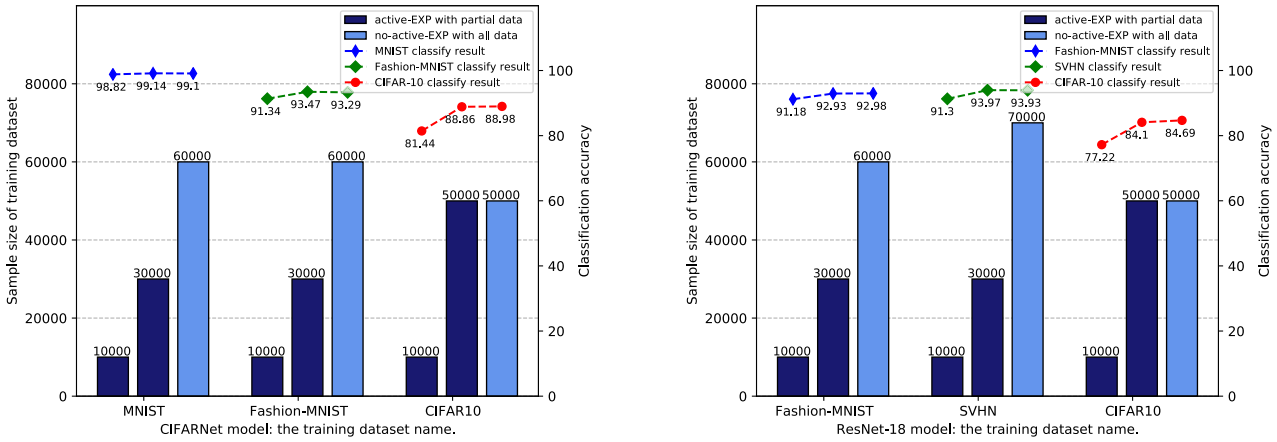


Fig. 13. Comparison results of active learning and nonactive methods on SNN-CIFARNet and SNN-ResNet18 models.

TABLE II
ABLATION EXPERIMENTS ON CIFARNET

Dataset / *Train size*	*MNIST*			*Fashion-MNIST*			*CIFAR-10*		
	LLAL	SNN-LLAL	**our method**	LLAL	SNN-LLAL	**our method**	LLAL	SNN-LLAL	**our method**
1,000	36.27	91.82	**95.68**	52.45	73.22	**80.00**	17.89	28.37	**45.98**
2,000	50.54	97.03	**97.82**	63.27	82.98	**87.39**	26.76	31.50	**62.03**
3,000	74.10	97.53	**98.23**	64.59	87.40	**88.52**	29.2	33.17	**67.08**
4,000	88.05	97.65	**98.63**	74.12	88.49	**89.25**	30.53	51.29	**72.57**
5,000	94.56	97.96	**98.70**	75.56	89.63	**89.49**	33.89	60.33	**74.59**
6,000	95.99	98.30	**98.75**	80.26	89.95	**90.11**	38.73	68.41	**76.64**
7,000	96.13	98.29	**98.87**	80.62	90.07	**90.26**	42.27	72.85	**77.89**
8,000	97.14	98.53	**98.67**	83.22	90.57	**90.57**	42.28	75.77	**78.69**
9,000	97.37	98.35	**98.75**	84.62	90.62	**90.70**	49.91	78.39	**80.22**
10,000	97.51	98.69	**98.77**	84.92	90.93	**90.92**	54.7	78.39	**81.44**

In the first experiment, we compare the proposed method with the inactive method on more labeled data. Different from previous experiments, the inactive model is trained using sufficient labeled training data, which are 50 000 samples or more on different tasks. While our method only uses some of the labeled samples, similar to Sections IV-D1 and IV-D2, different datasets and network structures are evaluated in this experiment.

The comparison results are shown in Fig. 13, with the left subfigure showing the results on SNN-CIFARNet and the right subfigure showing those on SNN-ResNet18. We see that our method achieves comparable performance only with half labeled samples compared with the inactive model, which demonstrates the effectiveness of the sample selection strategy in our active learning method.

In the second ablation experiment, we compare our method with different active learning baselines, including LLAL and SNN-LLAL, across different numbers of samples and datasets. In our comparison, the LLAL method applies the active learning strategy directly to SNN models without any adaptation modification, and the SNN-LLAL replaces the loss prediction module in LLAL with our own designed ActiveLossNet model adapted to SNNs. Compared with SNN-LLAL, our method takes the mean square error as the prediction loss instead of the pair loss.

Table II shows the comparison results on various datasets across different numbers of training samples. We see that SNN-LLAL achieves higher accuracy than conventional LLAL, which proves the effectiveness of the proposed ActiveLssNet in SNN models. In addition, our method outperforms both LLAL and SNN-LLAL, which demonstrates the advantage of the proposed loss function in SNN model training.

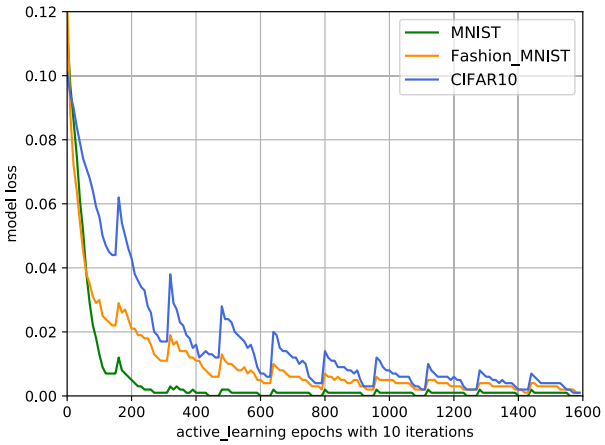


Fig. 14. SNN-CIFARNet training loss curve.

E. Convergence and Feature Analysis

Fig. 14 shows the training loss curve of the SNN-CIFARNet model. In all ten iterations of active learning, 160 epochs are set in each round, and there are a total of 1600 epochs. We see that the loss increases sharply at the beginning of each round of active learning training and then gradually declines because more data are added to the training set. We see that the training loss has a large decreasing trend in the early stage of active learning training and changes slowly in the later stage

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on November 14,2024 at 11:20:27 UTC from IEEE Xplore. Restrictions apply.

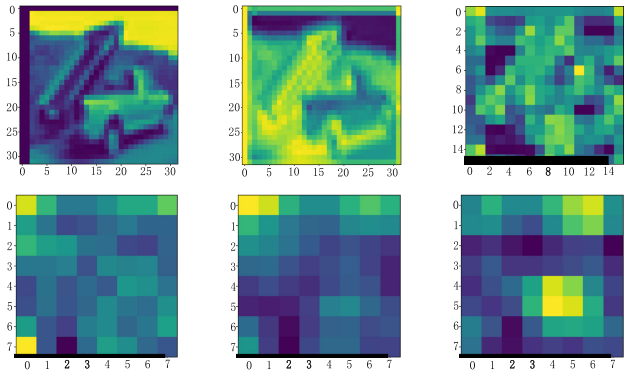


Fig. 15. Feature representation of SNN-CIFARNet. The first two subfigures show the feature before and after the coding layer. The last four subfigures show the feature representation after each convolution layer.

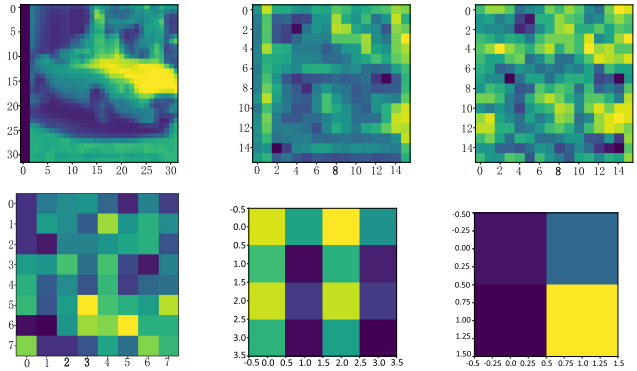


Fig. 16. Feature representation of SNN-ResNet18. The first two subfigures show the feature before and after the coding layer. The last four subfigures show the feature representation after each block of SNN-ResNet18.

of active iteration. This shows that the data selected by the active learning algorithm are very effective and useful during the early training stage.

Figs. 15 and 16 show the feature representation on SNN-CIFARNet and SNN-ResNet18, respectively. The first two subfigures represent the feature representations before and after the coding layer, and the last four subfigures show the feature representations after each block of SNN models. We see that as in conventional ANN models, SNN features become increasingly abstract with the increase in network layers. In addition, we see that the SNN-ResNet18 model learns rougher features than those in the same layer of SNN-CIFARNet because the residual cross-layer connections can compensate for these features.

V. Conclusion

Training a deep SNN model usually requires a large number of labeled samples. However, labels in real applications are usually very expensive. To address this issue, we proposed an effective active learning model for SNNs. Specifically, we proposed a loss prediction module ActiveLossNet and derived the corresponding algorithm for deep SNN models. To provide a comprehensive evaluation, we constructed two deep SNN models, SNN-CIFARNet and SNN-ResNet18, in the experiments and tested our method on CIFAR10, SVHN, MINST, and Fashion-MNIST. Comparison results demonstrate the effectiveness of the proposed method both in accuracy and convergence speed. The proposed method is effective in sample selection, and more low-power strategies will be studied in future works to take advantage of neuromorphic chips.

References

[1] S. Ghosh-Dastidar and H. Adeli, "Spiking neural networks," *Int. J. Neural Syst.*, vol. 19, no. 4, pp. 295–308, Aug. 2009.

[2] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, Nov. 2019.

[3] A. Taherkhani, A. Belatreche, Y. Li, G. Cosma, L. P. Maguire, and T. M. McGinnity, "A review of learning in biologically plausible spiking neural networks," *Neural Netw.*, vol. 122, pp. 253–272, Feb. 2020.

[4] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, and L. Shi, "Direct training for spiking neural networks: Faster, larger, better," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 1311–1318.

[5] H. Wu et al., "Training spiking neural networks with accumulated spiking flow," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 12, 2021, pp. 10320–10328.

[6] S. Zhou, X. Li, Y. Chen, S. T. Chandrasekaran, and A. Sanyal, "Temporal-coded deep spiking neural network with easy training and robust performance," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 12, May 2021, pp. 11143–11151.

[7] Q. Liu, H. Ruan, D. Xing, H. Tang, and G. Pan, "Effective AER object classification using segmented probability-maximization learning in spiking neural networks," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 2, 2020, pp. 1308–1315.

[8] S. B. Shrestha and G. Orchard, "Slayer: Spike layer error reassignment in time," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1412–1421.

[9] G. Liu, W. Deng, X. Xie, L. Huang, and H. Tang, "Human-level control through directly-trained deep spiking Q-Networks," 2021, *arXiv:2201.07211*.

[10] S. B. Shrestha and Q. Song, "Adaptive learning rate of spikeprop based on weight convergence analysis," *Neural Netw.*, vol. 63, pp. 185–198, Mar. 2015.

[11] S. B. Shrestha and Q. Song, "Robust learning in SpikeProp," *Neural Netw.*, vol. 86, pp. 54–68, Nov. 2016.

[12] W. Tan, D. Patel, and R. Kozma, "Strategy and benchmark for converting deep Q-networks to event-driven spiking neural networks," 2020, *arXiv:2009.14456*.

[13] J. Kim, K. Kim, and J.-J. Kim, "Unifying activation- and timing-based learning rules for spiking neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 19534–19544.

[14] Q. Zhan, G. Liu, X. Xie, G. Sun, and H. Tang, "Effective transfer learning algorithm in spiking neural networks," *IEEE Trans. Cybern.*, vol. 52, no. 12, pp. 13323–13335, Dec. 2022.

[15] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *J. Artif. Intell. Res.*, vol. 4, pp. 129–145, Mar. 1996.

[16] O. Sener and S. Savarese, "Active learning for convolutional neural networks: A core-set approach," 2017, *arXiv:1708.00489*.

[17] C. Lee, X. Wang, J. Wu, and X. Yue, "Failure-averse active learning for physics-constrained systems," *IEEE Trans. Autom. Sci. Eng.*, early access, Oct. 19, 2022, doi: 10.1109/TASE.2022.3213827.

[18] C. Lee, K. Wang, J. Wu, W. Cai, and X. Yue, "Partitioned active learning for heterogeneous systems," *J. Comput. Inf. Sci. Eng.*, vol. 23, no. 4, Aug. 2023, Art. no. 041009.

[19] D. D. Lewis and W. A. Gale, "A sequential algorithm for training text classifiers," in *Proc. SIGIR*. Cham, Switzerland: Springer, 1994, pp. 3–12.

[20] A. J. Joshi, F. Porikli, and N. Papanikolopoulos, "Multi-class active learning for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 2372–2379.

[21] K. Wang, D. Zhang, Y. Li, R. Zhang, and L. Lin, "Cost-effective active learning for deep image classification," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 12, pp. 2591–2600, Dec. 2017.

[22] W. H. Beluch, T. Genewein, A. Nurnberger, and J. M. Kohler, "The power of ensembles for active learning in image classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9368–9377.

[23] Y. Guo, "Active instance sampling via matrix partition," in *Proc. 23rd Int. Conf. Neural Inf. Process. Syst.*, vol. 1, 2010, pp. 802–810.

[24] M. Bilgic and L. Getoor, "Link-based active learning," in *Proc. NIPS Workshop Analyzing Netw. Learn. Graphs*, vol. 4, 2009, p. 9.

[25] N. Roy and A. McCallum, "Toward optimal active learning through Monte Carlo estimation of error reduction," in *Proc. ICML*, vol. 2, Williamstown, NJ, USA, 2001, pp. 441–448.

[26] B. Settles, M. Craven, and S. Ray, "Multiple-instance active learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 20, 2007, pp. 1289–1296.

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on November 14,2024 at 11:20:27 UTC from IEEE Xplore. Restrictions apply.

[27] A. Freytag, E. Rodner, and J. Denzler, "Selecting influential examples: Active learning with expected model output changes," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2014, pp. 562–577.
[28] D. Yoo and I. S. Kweon, "Learning loss for active learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 93–102.
[29] S. C. H. Hoi, R. Jin, J. Zhu, and M. R. Lyu, "Batch mode active learning and its application to medical image classification," in *Proc. 23rd Int. Conf. Mach. Learn. (ICML)*, 2006, pp. 417–424.
[30] T. Tran, T.-T. Do, I. Reid, and G. Carneiro, "Bayesian generative active deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6295–6304.
[31] D. Gissin and S. Shalev-Shwartz, "Discriminative active learning," 2019, *arXiv:1907.06347*.
[32] K. Konyushkova, R. Sznitman, and P. Fua, "Discovering general-purpose active learning strategies," 2018, *arXiv:1810.04114*.
[33] A. N. Burkitt, "A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input," *Biol. Cybern.*, vol. 95, no. 1, pp. 1–19, Jul. 2006.
[34] E. Ledinauskas, J. Ruseckas, A. Juršėnas, and G. Buračas, "Training deep spiking neural networks," 2020, *arXiv:2006.04436*.
[35] A. Krizhevsky et al., "Learning multiple layers of features from tiny images," M.S. thesis, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 2009.
[36] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS Workshop Deep Learn. Unsupervised Feature Learn.*, Granada, Spain, 2011.
[37] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[38] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.
[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[40] Y. Hu, H. Tang, and G. Pan, "Spiking deep residual networks," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Nov. 1, 2021, doi: 10.1109/TNNLS.2021.3119238.
[41] J. Wu, Y. Chua, M. Zhang, G. Li, H. Li, and K. C. Tan, "A tandem learning rule for effective training and rapid inference of deep spiking neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 1, pp. 446–460, Jan. 2023.
[42] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: VGG and residual architectures," *Frontiers Neurosci.*, vol. 13, p. 95, Mar. 2019.
[43] P. Panda, S. A. Aketi, and K. Roy, "Toward scalable, efficient, and accurate deep spiking neural networks with backward residual connections, stochastic softmax, and hybridization," *Frontiers Neurosci.*, vol. 14, p. 653, Jun. 2020.
[44] W. Fang, Z. Yu, Y. Chen, T. Huang, T. Masquelier, and Y. Tian, "Deep residual learning in spiking neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 21056–21069.
[45] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, and Y. Tian, "Incorporating learnable membrane time constant to enhance learning of spiking neural networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 2661–2671.



Xiurui Xie received the Ph.D. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2016.

She worked as a Research Fellow at Nanyang Technological University, Singapore, from 2017 to 2018, and a Research Scientist at the Agency for Science, Technology and Research (A*STAR), Singapore, from 2018 to 2020. She has authored over ten technical papers in prominent journals and conferences. Her primary research interests are neural networks, neuromorphic chips, transfer learning, and pattern recognition.



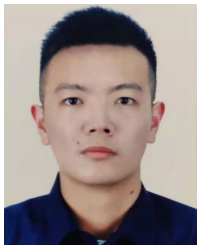
Bei Yu received the bachelor's degree from Jilin University, Changchun, China, in 2018. She is currently pursuing the M.S. degree with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China.

Her current research interests include spiking neural networks and active learning algorithms.



Guisong Liu received the B.S. degree in mechanics from Xi'an Jiaotong University, Xi'an, China, in 1995, and the M.S. degree in automatics and the Ph.D. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2000 and 2007, respectively.

He was a Visiting Scholar with Humbolt University, Berlin, Germany, in 2015. Before 2021, he was a Professor with the School of Computer Science and Engineering, University of Electronic Science and Technology of China. He is currently a Professor and the Dean of the School of Computing and Artificial Intelligence, Southwestern University of Finance and Economics, Chengdu. He has filed over 20 patents and published over 70 scientific conference and journal papers. His research interests include pattern recognition, neural networks, and machine learning.



Qiugang Zhan received the bachelor's degree from Yantai University, Yantai, China, in 2017. He is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China.

His current research interests include spiking neural networks and transfer learning algorithms.



Huajin Tang received the B.Eng. degree from Zhejiang University, Hangzhou, China, in 1998, the M.Eng. degree from Shanghai Jiao Tong University, Shanghai, China, in 2001, and the Ph.D. degree from the National University of Singapore, Singapore, in 2005.

He was a System Engineer with STMicroelectronics, Singapore, from 2004 to 2006. From 2006 to 2008, he was a Post-Doctoral Fellow with the Queensland Brain Institute, The University of Queensland, Brisbane, QLD, Australia. Since 2008, he was the Head of the Robotic Cognition Laboratory, Institute for Infocomm Research, A*STAR, Singapore. Since 2014, he has been a Professor with the College of Computer Science, Sichuan University, Chengdu, China. He is currently a Professor with the College of Computer Science and Technology, Zhejiang University. His current research interests include neuromorphic computing, neuromorphic hardware and cognitive systems, and robotic cognition.

Dr. Tang received the 2016 IEEE Outstanding TNNLS Paper Award and the 2019 IEEE Computational Intelligence Magazine Outstanding Paper Award. He is a Board of Governor Member of the International Neural Networks Society. He was the Program Chair of IEEE CIS-RAM in 2015 and 2017 and ISNN in 2019, and the Co-Chair of the IEEE Symposium on Neuromorphic Cognitive Computing from 2016 to 2019. He has served as an Associate Editor for IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON COGNITIVE AND DEVELOPMENTAL SYSTEMS, *Frontiers in Neuromorphic Engineering*, and *Neural Networks*.

Authorized licensed use limited to: JILIN UNIVERSITY. Downloaded on November 14,2024 at 11:20:27 UTC from IEEE Xplore. Restrictions apply.