

Efficient Structure Slimming for Spiking Neural Networks

Li, Yaxin; Fang, Xuanye; Gao, Yuyuan; Zhou, Dongdong; Shen, Jiangrong; Liu, Jian K.; Pan, Gang; Xu, Qi

DOI:

[10.1109/tai.2024.3352533](https://doi.org/10.1109/tai.2024.3352533)

License:

Other (please specify with Rights Statement)

Document Version

Peer reviewed version

Citation for published version (Harvard):

Li, Y, Fang, X, Gao, Y, Zhou, D, Shen, J, Liu, JK, Pan, G & Xu, Q 2024, 'Efficient Structure Slimming for Spiking Neural Networks', *IEEE Transactions on Artificial Intelligence*. <https://doi.org/10.1109/tai.2024.3352533>

[Link to publication on Research at Birmingham portal](#)

Publisher Rights Statement:

This is the accepted author manuscript of an article published in IEEE Transactions on Artificial Intelligence; final published version available at <https://doi.org/10.1109/TAI.2024.3352533>

© 2024 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

General rights

Unless a licence is specified above, all rights (including copyright and moral rights) in this document are retained by the authors and/or the copyright holders. The express permission of the copyright holder must be obtained for any use of this material other than for purposes permitted by law.

- Users may freely distribute the URL that is used to identify this publication.
- Users may download and/or print one copy of the publication from the University of Birmingham research portal for the purpose of private study or non-commercial research.
- User may use extracts from the document in line with the concept of 'fair dealing' under the Copyright, Designs and Patents Act 1988 (?)
- Users may not further distribute the material nor use it for the purposes of commercial gain.

Where a licence is displayed above, please note the terms and conditions of the licence govern your use of this document.

When citing, please reference the published version.

Take down policy

While the University of Birmingham exercises care and attention in making items available there are rare occasions when an item has been uploaded in error or has been deemed to be commercially or otherwise sensitive.

If you believe that this is the case for this document, please contact UBIRA@lists.bham.ac.uk providing details and we will remove access to the work immediately and investigate.

Efficient Structure Slimming for Spiking Neural Networks

Yaxin Li, Xuanye Fang, Yuyuan Gao, Dongdong Zhou, Jiangrong Shen,
Jian K. Liu, Gang Pan, Qi Xu*

Abstract—Spiking neural networks (SNNs) are deeply inspired by biological neural information systems. Compared to convolutional neural networks (CNNs), SNNs are low power consumption because of their spike based information processing mechanism. However, most of the current structures of SNNs are fully-connected or converted from deep CNNs which poses redundancy connections. While the structure and topology in human brain systems are sparse and efficient. This paper aims at taking full advantage of sparse structure and low power consumption which lie in human brain and proposed efficient structure slimming methods. Inspired by the development of biological neural network structures, this paper designed types of structure slimming methods including neuron pruning and channel pruning. In addition to pruning, this paper also considers the growth and development of the nervous system. Through iterative application of the proposed neural pruning and rewiring algorithms, experimental evaluations on CIFAR-10, CIFAR-100, and DVS-Gesture datasets demonstrate the effectiveness of the structure slimming methods. When the parameter count is reduced to only about 10% of the original, the performance decreases by less than 1%.

Impact Statement—Spiking neural networks, with spiking neuron models as the fundamental units, exhibit excellent qualities such as biological interpretability and low power consumption. Drawing inspiration from the plasticity of the network structure in the biological neural system, we propose a method for adaptive learning in spiking neural networks. This method explores the network structure at two levels: neuron granularity and feature map granularity. During the process of network training, inactive units are identified and pruned to achieve a sparse network structure. This process resembles the sparse connectivity and ultra-low power consumption employed by the human brain in processing various perceptual information. Our method maintains high accuracy even when compressing the network structure to a larger extent. By combining the plasticity of the biological neural system and learning methods of artificial neural networks, we explore a novel framework for adaptive sparse learning in neural networks.

Index Terms—Channel pruning, network slimming, spiking neural networks, structure learning, weight pruning

This work is supported by National Natural Science Foundation of China (NSFC No.62206037, 62306274), The Open Project of Anhui Provincial Key Laboratory of Multimodal Cognitive Computation, Anhui University, No. MMC202104, Open Research Fund from Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), under Grant No. GML-KF-22-11 and the Fundamental Research Funds for the Central Universities (DUT21RC(3)091)

Yaxin Li, Xuanye Fang, Yuyuan Gao, Dongdong Zhou and Qi Xu are with the School of School of Computer Science and Technology, Dalian University of Technology, Dalian, 116024, China. (Corresponding author: Qi Xu xuqi@dlut.edu.cn)

Jiangrong Shen and Gang Pan are with the College of Computer Science and Technology, Zhejiang University, Zhejiang, 310027, China.

Jian K. Liu is with the College of Engineering and Physical Sciences, University of Birmingham, Birmingham, B152TT, United Kingdom.

I. INTRODUCTION

The human brain has billions of neurons and synapse structures, yet very little energy is required. Spiking neural networks (SNNs) are constructed to mimic the working principles and mechanisms of the human brain, utilizing spiking sequences to transmit information. Most of the model structures of SNNs are derived from existing structures in artificial neural networks (ANNs), and such structures are fixed and unchanged during the training process. However, the neurons and connection structure of the human brain are constantly changing during development, including the growth, development, and death of neurons [1], [2].

Considering the event-driven and biological plausible nature, SNNs are supposed to be a more energy efficient model compared to traditional deep neural networks (DNNs). Although SNNs are deeply inspired by biological neural systems, most of the current structures of typical SNNs are not sparse as biological neural systems which results in more energy consumption as DNNs did.

Because of the fixed fully-connected structure, there exists parameter redundancy for most of the SNNs which makes them lack rich structural plasticity and could not mimic highly efficient neural computation of biological neural systems. More importantly, the redundant connections and parameters would get the whole network model into an energy-draining situation whenever during training or inference [3] [4] [5].

Currently, many existing SNN architectures heavily rely on dense and fixed connections and are often constructed using well-established structures such as ResNet and VGG [6] [7]. However, when these complex architectures are used to build larger and deeper neural networks for training tasks, they demand significant computational power and resources. This poses challenges when it comes to deploying them on resource-constrained embedded devices for practical applications. In the human brain, network connections are typically sparse [8], and the amount of energy required to operate is also very low. Therefore, there is a growing interest in exploring methods to sparsify network structures in the context of SNNs. By doing so, we aim to harness the energy-efficient characteristics of SNNs more effectively. It's crucial to clarify that this sparsification approach is distinct from simply porting traditional ANNs to SNNs. Instead, it involves redesigning and optimizing SNN architectures to better align with the sparse and energy-efficient principles observed in biological neural networks.

Aiming at this, this paper proposed an efficient structure

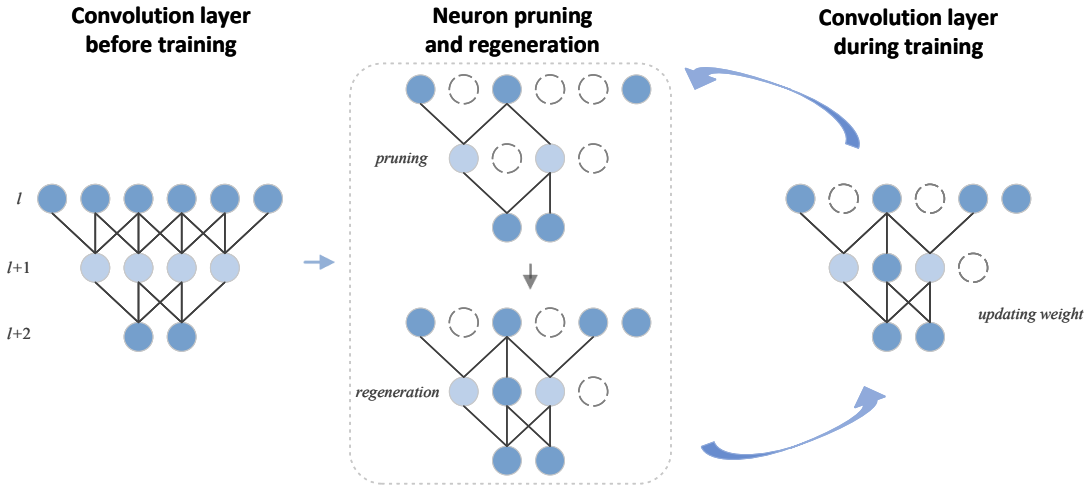


Fig. 1: Slimming based on neuron pruning.

slimming method for SNNs. Inspired by slimming method [9], this paper combining the surrogate gradient method and network pruning method to construct deep but not redundant SNN models.

This framework includes two pruning methods: pruning at the neuron level and pruning at the feature map level. Pruning at the neuron level involves compressing the network structure by removing individual neurons, including removing unimportant neurons during training and allowing important neurons to regenerate. Pruning at the feature map level assesses the importance of channels using a corresponding factor and removes unimportant channels.

Implementing the sparse training of SNNs from scratch by the proposed structure slimming method, any type of SNNs could be optimized with a slimming structure. The proposed spiking based slimming method not only trains the SNNs from scratch with surrogate gradient method, but also makes the structure sparse during training and inference period. The contributions of this paper is summarized as follows:

- This paper proposed a spiking based network slimming method, this kind of pruning method could make the objective SNN model sparse to reduce the redundancy and save energy.
- Combining the surrogate gradient training method, the proposed method analyses the memory and energy quantitatively. From the perspective of channel and neuron, the proposed method could be more targeted to slim the SNNs.
- The experimental evaluations on CIFAR10, CIFAR100, and DVS-Gesture showed that the proposed method could discard the redundant parameters and save memory and energy efficiency while maintaining the performance well and significantly outperforming baseline methods.

II. RELATED WORKS

A. Model training

The learning algorithms of SNNs are mainly divided into: unsupervised learning method, supervised method and ANN-

to-SNN conversion method. The unsupervised learning methods are almost based on the principle of STDP (Spike-Timing-Dependent Plasticity) [10]–[15], but these methods are unable to build deep and complex neural networks. Besides, by adjusting the local connection, global energy consumption cannot be ignored.

Drawing on the idea of backpropagation (BP) in deep learning, some learning methods based on gradient descent rule appear to train SNNs with complex and deep structures. These methods usually back-propagate the error by computing an approximation to the gradient [16]. Wu et al. [17] proposed the STBP (Spatio-Temporal Backpropagation) method utilizes the temporal and spatial domain to update the neuron state. Zheng et al. [18] introduced a tdBN (a threshold-dependent Batch Normalization) layer to the STBP network and this method performs well on ResNet structure. With the exploration of the difference between ANN and SNN at the residual structure, Fang et al. [6] achieved identity mapping with element-wise functions which is applicable to all neuron models and this is the first time that directly training deep SNNs more than 100 layers. The surrogate gradient method has a loss in approximating the gradient. Deng et al. [19] analyze the reason and proposed the TET (Temporal Efficient Training) method to compensate for the loss. Zhang et al. [20] proposed a spike-timing-dependent BP (STDBP) learning algorithm for deep networks which combine temporal coding and BP learning. There is also research that integrates attention mechanisms to design suitable spiking forms for RNNs [21]. The ANN-to-SNN conversion methods transfer the weight values from ANN to SNN which can build complex and deep networks [22]. Alternatively, employing ANN knowledge distillation can also enhance the performance of SNNs [23]. Although these methods normally perform best among all of the training methods, they generally need error compensation to reduce performance degradation due to conversion [24], [25]. For the three methods mentioned above, these deep structures of SNNs usually require high power consumption and have some redundancy in the structure with the restriction

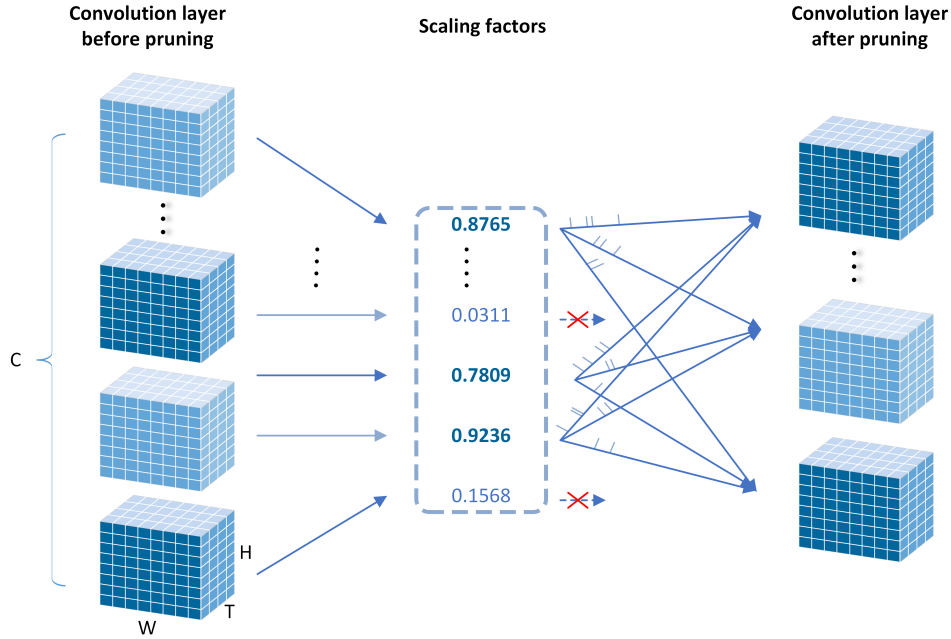


Fig. 2: Slimming based on channel pruning.

of redundant connections.

B. Pruning method

The network architectures used in deep learning always are complex and computationally expensive. Thus, some researchers pruned synaptic connections which lead to a sparse structure to improve the efficiency. The earliest selective removal methods were proposed by Reed [26] as early as the late 1980s for smaller-scale multilayer perceptron models. As the network structures become more and more complex, compressing the number of network parameters becomes an important requirement. In ANNs, there are two main pruning methods which can be categorized as unstructured pruning and structured pruning. Unstructured pruning methods generally remove the weight connections with lower weight [7], [27], [28]. The method proposed by Han et al. [7] prunes the network weights through a two-step iteration of pruning connections and training weights. Deep R can regenerate the pruned connections during training [29]. Unstructured pruning methods are flexible but usually require specific hardware and library support. Structured pruning methods tend to be coarse-grained pruning such as channel pruning, filter pruning, and layer pruning. The filters with the lower sum of the absolute values mean unimportant and can be discarded which is proposed by Li et al. [30]. Liu et al. [9] introduced a scaling factor to judge the importance of feature maps. Structured pruning can more effectively compress the number of network parameters while maintaining performance.

SNNs have bionic properties and use spikes to transmit information. In order to better simulate the sparsity and low power consumption of biological brains, some sparsity learning methods are introduced in SNNs. Qi et al. [31] proposed a method of jointly training the connections and weights

which uses STDP (Spike Timing Dependent Plasticity) and Tempotron algorithm to iterate in turn. This method learned the connections structure by training a gate function according to the timing information of spikes. With the emergence of some deep SNN models, some studies have proposed slimming methods that are suitable for spiking based surrogate gradient training methods. Shen et al. [32] proposed an evolutionary structure learning framework which learning the structure from an initial sparse network. Deng et al. [3] combined ADMM optimization tools with STBP methods for pruning. Inspired by changes in the state of dendritic spines, Chen et al. [5] pruned the SNN models through three rules when the trainable parameters are in different states. For ANN2SNN training method, a sparse learning method called attention-guided compression (AGC) technique uses attention-maps to guide the compression [33]. More than just pruning, some algorithms combine pruning and regeneration together such as Grad R [4]. Different from the existing methods, the proposed considered the impact of time which plays a crucial role in neural dynamics. Thus, the whole SNN could be pruned dynamically and adaptively.

C. Motivation

Due to the binary nature of the spiking sequence, it is difficult to train the SNNs with BP method directly. What's more, because if lacking suitable network structures, SNNs did not play to their advantage in energy consumption. General SNNs with deep structures generally have the disadvantage of redundancy. Inspired by the synaptic connection structure of the human brain, this paper proposed a structure slimming method for SNNs that can better utilize the low power consumption of SNNs. In order to get the sparse but efficient

model, this paper pruned the redundant connections of the pre-trained SNNs at different grading. In this paper, we proposed two structure learning methods which are slimming based on weight pruning and slimming based on channel pruning.

After pruning the pre-trained SNNs models, the amount of parameters is much reduced while the performance did not drop too much compared to the original model. This method is designed to build a model with a more low-power sparse connection which is more biomimetic. The slimming SNN model not only simplifies the network structure but is also more conducive to deployment in resource-constrained embedded devices such as spiking based neuromorphic hardware.

III. METHODS

Aiming at finding an efficient structure of SNN, the proposed method prunes redundant and unimportant neurons during the training phase dynamically. Our method provides two ways for structure slimming including neuron pruning and channel pruning at different grading. It utilizes the surrogate gradient method for training which can apply to deep SNNs with different structures on static and neuromorphic datasets.

A. The slimming framework

Overall framework of the slimming method. The slimming framework is designed to get a sparse connection and efficient SNN model which is inspired by the growth and death of neurons during biological neural systems development [1] [34] [2]. As shown in Fig.1 and Fig.2, the slimming framework prunes redundant neurons at the individual neuron level and at the feature map level. These redundant neurons usually make little contribution to the correct classification or may even cause interference. Sparse training is also used to reduce the computational cost caused by the redundant neurons in the network. Neuron-level sparsity is achieved by removing and growing neurons simultaneously during training to obtain a sparse model. Feature map-level pruning removes redundant channels based on the magnitude of the normalization layer's factors.

Spiking neuron network. We use the IF (Integrate-And-Fire Models) spiking neuron model as the basic neuron model of the SNN. The spiking neuron model is a mathematical model of differential equations established to simulate the action potential process of biological neurons. The spiking neuron models can be expressed as three equations Eq.(1), Eq. (2), Eq. (3) which is charge, discharge and reset:

$$H[t] = V[t - 1] + X[t] \quad (1)$$

$$S[t] = \Theta(H[t] - V_{\text{threshold}}) \quad (2)$$

$$V[t] = H[t] * (1 - S[t]) + V_{\text{reset}} * S[t] \quad (3)$$

where $V[t]$ denotes the membrane potential of the spiking neuron model. $X[t]$ is the external input current from the presynaptic neurons at time step t . The IF model receives the external current and the state of membrane potential is stimulated to $H[t]$. If the membrane potential exceeds the threshold $V_{\text{threshold}}$, the neuron will fire a spike and reset to the threshold V_{reset} . In this paper, we use the surrogate gradient

method to train the SNN. The common surrogate gradient function is sigmoid function Eq. (4) which can be expressed as:

$$\sigma(\alpha x) = \frac{1}{1 + \exp(-\alpha x)} \quad (4)$$

where α control the smoothness of the function and the α is larger the function is closer to the Heaviside step function.

B. Slimming based on neuron pruning

Sparsity method. Slimming based on neuron pruning removes redundant or unimportant neurons and regenerates important neurons to obtain an appropriate network structure. Neurons that fire fewer spikes may not contribute to correct classification or may interfere with it, so we can compress the network structure by removing redundant neurons while maintaining accuracy. However, some important neurons may be mistakenly pruned, so we can reactivate these incorrectly pruned neurons through neuron regeneration. This is similar to the growth and degeneration of neurons in the human brain during development.

During pruning, we remove neurons with a lower number of spikes at each iteration. First, we calculate the cumulative number of spikes fired by each neuron in each layer of the model on the training set and remove neurons with a number of spikes below a certain threshold. Then, this method calculates a pruning mask M_{pr}^l where the corresponding positions of the neurons to be removed are set to 0. During training, we multiply this mask with the feature map.

During regeneration, we accumulate the membrane potential of each neuron at each iteration. The neurons that were removed can still accumulate membrane potential during training. If the membrane potential exceeds a certain threshold, the neuron is regenerated. In the regenerate mask M_{re}^l , the corresponding positions of the regenerated neurons are set to 1, indicating that these neurons are activated. After updating the neuron mask, the weight matrix of layer l can be expressed as Eq. (5):

$$W_l = W_l * M_l = W_l * (M_{\text{pr}}^l + M_{\text{re}}^l) \quad (5)$$

where W_l and M_l represent the weight and the pruning mask of the layer l .

Training algorithm. In the training stage, L2 regularization is added to better perform sparse training. During each iteration of training, neuron pruning, and regeneration operations are performed to update the mask. This method learns a sparse network structure while training the parameters. The overall algorithm is shown in Algorithm 1.

C. Slimming based on channel pruning

Sparsity method. Slimming based on channel pruning prunes the redundant and unimportant connections by removing the entire channel which sparse the network more effectively. This method introduced a scaling factor γ [9] corresponding to each channel at the last layer which can be trained with the network weight jointly. We use the values of parameters to determine the importance of the channel and remove the channel with small scaling factors. We know that

Algorithm 1 Training SNN model with neuron pruning and regeneration.

Require: the neuron mask M , the weight matrix W , input dataset samples X , true labels y_{true} .

Ensure: Sparse SNN model

```

1: for each training epoch  $i$  do
2:   Update the neuron mask;
3:   (1)remove the neurons with lower spikes;
4:   (2)regenerate the neurons with high membrane potential;
5:   (3)update the neuron mask  $M$ ;
6:   for each layer  $l$  of neurons do
7:     The weight matrix  $W$  is  $W = W * M$ ;
8:     Compute the forward propagation;
9:     Compute the cumulative spike count and membrane;
       the potential of each neuron;
10:  end for
11:  Compute the backward propagation;
12: end for

```

the batch normalization (BN) layer computes the meaning and standard deviation values of each corresponding channel and then scales the features with the parameters γ and β of the BN layer. If there is a BN layer after the convolution layer, the scaling parameters γ of the BN layer as follows Eq. (6) can be used as the scaling factors.

$$\hat{z} = \frac{z_{\text{in}} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}; \quad z_{\text{out}} = \gamma \hat{z} + \beta \quad (6)$$

where γ and β are the scale and shift parameters of the BN layer. μ_B and σ_B denote the meaning and standard deviation of the features z_{in} of the last layer.

In order to achieve sparsity, we add an L1-norm penalty to the BN layer of the network. Training with sparse makes the scaling factors corresponding to redundant connections converge to 0 and then this method removes smaller weights proportionally. Thus, the loss function used in our method includes two parts: the cross entropy loss and the L1-norm regularization which can be expressed as Eq. (7):

$$L = \sum_{(x,y)} l(f(x, W), y) + \lambda \sum_{\gamma \in \Gamma} g(\gamma) \quad (7)$$

where $f(x, W)$ represents the output of the spiking rate of the output layer and the $g(\gamma)$ is the L1-norm regularization. If the network model is skip-layer connections or pre-activation such as ResNet, a channel selection layer will be added to select the channels that contribute.

Batchnorm in spatio-temporal dimension. In SNNs, there is an extra dimension of time which is different from ANN. Therefore, we use the tdbN (threshold-dependent batch normalization) as the BN layer which normalizes the features in both spatio-temporal dimensions. The tdbN computes the meaning $E[x_k]$ and standard deviation $\text{Var}[x_k]$ of the pre-synaptic inputs along the time axis T and spatial axis (H, W) . The features of the k -th channel $x_k = (x_k^1, x_k^2, \dots, x_k^T)$ can be normalized as Eq. (8):

$$\hat{x}_k = \frac{\alpha V_{th} (x_k - E[x_k])}{\sqrt{\text{Var}[x_k] + \epsilon}}; \quad y_k = \lambda_k \hat{x}_k + \beta_k \quad (8)$$

where the αV_{th} denotes the threshold of the spiking neuron model. The normalization method of tdbN is more suitable for SNNs which utilize spikes to transmit information. It normalizes the features to $N(0, (\alpha V_{th})^2)$ which makes the features keep the form of spikes in forward inference.

Training algorithm. In the first stage, we pre-train the SNN model with a sparsity penalty so that we can select the important channels better through the values of the scaling factors. In the second stage, we prune a certain percentage of channels with the lower absolute values of scaling factors to get the slimming model. Then the slimming model is fine-tuned without a sparsity penalty. The convolution kernel simulates the receptive field of the biological visual system so pruning on a coarse-grained makes it more compatible with the sparsity and low-power nature of the human brain. Besides, the proposed structure pruning method does not require specific hardware and library support. The overall algorithm is shown in Algorithm 2.

Algorithm 2 Training SNN model with channel pruning.

Require: input dataset samples X , true labels y_{true} .

Ensure: Sparse SNN model

```

1: #During training
2: for each training epoch do
3:   train the SNN model based on the surrogate gradient
       with sparsity penalty;
4: end for
5: #During pruning
6: sort all the scaling parameters  $\gamma$  in tdbN layer;
7: compute the threshold  $thre$ 
8: for each tdbN layer do
9:   if  $\gamma < thre$  then
10:    remove the tdbN channel;
11:    remove the corresponding Conv2d channel;
12:   end if
13: end for
14: #During fine-tuning
15: for each training epoch do
16:   retrain the pruned SNN model;
17: end for

```

IV. EXPERIMENTS

In this section, we analyze the experiments of the proposed slimming framework on static datasets and neuromorphic datasets including MNIST, CIFAR10, CIFAR100, and DVS-Gesture. In order to prove the robustness, we evaluate this method on CIFAR10 with different levels of Gaussian noise. Also, we compare this method with other SNN pruning methods which demonstrate the superiority of the proposed algorithm.

TABLE I: Test accuracies of the SNN slimming framework based on neuron pruning.

Dataset	Arch.	Reserved nerons	ACC.(%)	ACC loss(%)	Synops	Energy
CIFAR10	Resnet18	100%	92.92	-	334K	8.68E-06J
		89.36%	92.91	-0.01	324K	8.42E-06J
		81.55%	92.6	-0.32	313K	8.13E-06J
		71.21%	91.73	-1.19	295K	7.67E-06J
		64.95%	91.34	-1.58	280K	7.28E-06J
CIFAR100	WRN16-4	100%	69.14	-	322K	8.37E-06J
		88.22%	69.40	0.26	291K	7.56E-06J
		76.46%	69.25	-0.11	269K	6.99E-06J
		63.63%	68.15	-0.99	250K	6.50E-06J
		57.20%	67.46	-1.68	227K	5.90E-06J
CIFAR100	Resnet18	100%	70.41	-	472K	1.22E-05J
		97.13%	70.69	0.28	456K	1.18E-05J
		92.74%	70.39	-0.02	456K	1.18E-05J
		87.95%	69.79	-0.62	441K	1.14E-05J
		78.23%	69.04	-1.37	406K	1.05E-05J

TABLE II: Test accuracies of the SNN slimming framework based on channel pruning.

Dataset	Arch.	Pruned	Pruned ACC.(%)	Parameters	Fine-tune ACC.(%)
CIFAR10	VGG16	baseline	91.71	14.72M	-
		0	91.22	14.72M	-
		0.1	91.24	11.36M	-
		0.2	91.22	8.52M	-
		0.3	91.22	6.14M	-
		0.4	91.22	4.30M	-
		0.5	91.22	2.92M	91.24
		0.6	90.90	2.18M	91.16
CIFAR10	Resnet18	baseline	92.92	11.18M	-
		0	92.71	11.18M	-
		0.1	92.70	9.01M	-
		0.2	92.70	7.02M	-
		0.3	92.71	5.39M	-
		0.4	92.71	3.91M	-
		0.5	88.36	3.45M	92.31
DVS-Gesture	VGG13	baseline	95.40	9.41M	-
		0	94.80	9.41M	-
		0.1	93.42	6.86M	94.79
		0.2	66.31	5.35M	94.79
		0.3	24.03	4.30M	94.44
		0.4	10.86	3.46M	94.44
		0.5	8.31	2.57M	92.36

A. Experimental Settings

Datasets. The static datasets are MNIST, CIFAR10, and CIFAR100. This static dataset is encoded as spikes sequence when input into the network and transmitted as the form of spikes. For the neuromorphic dataset, we need to pre-process the dataset which integrates the event data to frame form. The dataset DVS-Gesture is in the shape of 128*128*2 and has 11 categories. What's more, we build noise datasets that add Gaussian noise on CIFAR10 with variances of 0.01, 0.03, and 0.05.

Model training. The experiments are evaluated on the server with 8 NVidia GeForce RTX 2080 Ti GPUs. The SNN architectures we used are VGG and Resnet structures which are usually used in deep networks. We construct the SNN

TABLE III: Test accuracies of the SNN slimming framework based on channel on noise CIFAR10 datasets.

Gaussian noise	Arch.	Pruned	Pruned ACC.(%)	Parameters	Fine-tune ACC.(%)
0.01	VGG16	baseline	83.01	14.72M	-
		0	82.81	14.72M	-
		0.1	82.84	11.31M	-
		0.2	82.82	8.46M	-
		0.3	82.84	6.09M	-
		0.4	82.83	4.25M	-
		0.5	82.81	2.90M	-
		0.6	81.77	2.41M	82.05
0.03	VGG16	baseline	78.11	14.72M	-
		0	77.45	14.72M	-
		0.1	77.47	11.30M	-
		0.2	77.46	8.42M	-
		0.3	77.46	6.07M	-
		0.4	77.45	4.24M	-
		0.5	77.45	2.98M	-
		0.6	51.79	2.60M	76.00
0.05	VGG16	baseline	74.41	14.72M	-
		0	73.58	14.72M	-
		0.1	73.57	11.34M	-
		0.2	73.58	8.46M	-
		0.3	73.56	6.07M	-
		0.4	73.59	4.20M	-
		0.5	73.64	3.06M	-
		0.6	21.04	2.64M	55.93

based on the Spikingjelly [35] which trains the SNN with the surrogate gradient method. When pruning we gradually increase the percentage of the pruned connections and then fine-tune the whole network.

B. Evaluation on the slimming structures at different grading

To show the effectiveness of the slimming framework at different grading, we evaluate the experiment at different compression ratios.

Evaluation of slimming based on neuron pruning. As shown in Table I, we use Resnet18 and WRN16-4 for the slimming based on neuron pruning. For the CIFAR10 dataset, Removing some neurons leads to a slight decrease in accuracy, but reduces the number of synaptic spikes required for computation. For the CIFAR100 dataset, after removing some neurons the accuracy, there is a slight improvement in accuracy. As the pruning percentage increases, the accuracy still does not drop much. This result indicates that some neurons with fewer spikes do not play a significant role in the classification. We know that neurons that fire fewer or no spikes have little impact on the spike firing of the neurons in the layers that follow, which means that only a small fraction of the connections in the network structure actually contribute to the classification.

Evaluation of slimming based on channel pruning. Channel pruning is a coarse-grained pruning method, as shown in Table II, we evaluate this method with VGG16 and Resnet18 structures on static and neuromorphic datasets. The baseline means the result without sparse training in which there is no $L1$ regularization to the BN layer. After sparse training, the accuracy of the model is slightly reduced. As the pruning

TABLE IV: Parameters and synaptic operations evaluation of the slimming framework on CIFAR10.

	Baseline	Pruned 0	Pruned 0.1	Pruned 0.2	Pruned 0.3	Pruned 0.4	Pruned 0.5	Pruned 0.6	0.7	0.8
ACC.(%)	91.71	91.22	91.24	91.22	91.22	91.22	91.22	90.90	6.32	9.97
Fine-tune ACC.(%)	-	-	-	-	-	-	91.24	91.16	91.11	89.01
Param.	14.72M	14.72M	11.36M	8.52M	6.14M	4.30	2.92M	2.18M	1.64M	0.71M
Synops	32.46K	28.96K	28.28K	28.29K	28.35K	27.96K	28.33K	28.33K	28.79K	27.79K
Energy	8.43E-07J	7.52E-07J	7.35E-07J	7.35E-07J	7.37E-07J	7.26E-07J	7.36E-07J	7.36E-07J	7.49E-07J	7.22E-07J

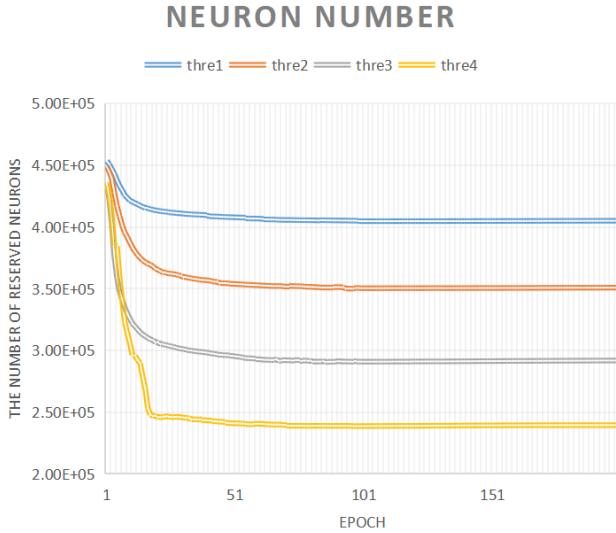


Fig. 3: The neuron number during training.

percentage increases, the accuracy loss is all about 0.5. The accuracy also can be made to go back up by fine-tuning even when the pruning percentage is too large. It can be seen that the accuracy of the VGG16 model on the CIFAR10 dataset only decreases 0.55 when the number of parameters drops to 14.8% of the original. For the DVS-Gesture dataset, the accuracy of the pruned network drops slightly but not much. The reason for this result is that some feature maps in the network are meaningless and their information in the spike sequence does not contribute to the classification even generates noise.

C. Evaluation on Noisy Influence

Noise resistance is an advantage of SNNs. To verify the validity of this slimming method on noisy datasets, we added Gaussian noise to the CIFAR10 images.

The variance of Gaussian noise we add is 0.01, 0.03, and 0.05, respectively. Here, we choose the channel pruning method to verify the noise resistance. As shown in Table III, the accuracy loss after pruning is around 0.5 for different intensities of noise. Especially, when the pruned channel reaches 50%, the accuracy is almost the same as the sparse training accuracy. Thus, it indicates that this method has good robustness which maintains the original performance after pruning. Under noisy conditions, this method can still distinguish important feature maps from redundant ones effectively.

D. Parameter Analysis

Here we analyze the parameters and synaptic operations of this slimming framework. Though pruning, this method takes better advantage of SNN's low power consumption.

As shown in Table I, We can see that reducing the number of neurons can decrease the overall number of synaptic operations in the network, which is very beneficial for reducing computational power consumption. When deployed on the TrueNorth neurosynaptic chip, each synaptic operation incurs an energy cost of 26 pJ [36]. It can be observed from the table that the energy consumption per operation decreases as the pruning ratio increases. Fig.3 shows the variation of the number of neurons during the training process. It can be observed that the number of neurons decreases rapidly and then stabilizes. *thre* represents different pruning thresholds of 0.05, 0.1, 0.15, and 0.2, which indicate the average number of pulse firings of each neuron over the entire dataset. The larger the pruning threshold, the fewer neurons will be converged to. As shown in Table IV, we use the slimming method based on channel pruning and analyze the consumption. As the percentage of pruning increases, the number of parameters decreased substantially, while the synaptic operations remain essentially the same. It indicates that the number of spike firing in the network does not change much after pruning which means only a few neurons transmit important features.

E. Performance Comparison with Other Methods

In order to better demonstrate the proposed slimming framework, we compare this work with other pruning methods of SNNs.

As shown in Table V, our method can obtain sparse models with better performance but need fewer time steps only 4. The number of connections we calculate refers to the quantity of parameters, which are the weights present in the network. For the neuron pruning on the CIFAR100 dataset, the accuracy loss has a slight improvement in accuracy when pruning a few neurons.

For the channel pruning on the MNIST dataset, when connectivity is only 26.21%, the accuracy loss is small only 0.10. Compared to other models such as Deep R, Grad R and ADMM-based, this method has less accuracy loss at the same pruning percentage. For the channel pruning on the CIFAR10 dataset, the accuracy loss is all around 0.5 when the pruning ratio is small. When the parameter count is reduced to 4.82 times the original, the accuracy decreases by 2.70%. Although the sparsity is not better than Deep R and State Transition, the accuracy loss is less than theirs. This work only needs 4 time steps which is much smaller than the time steps of

TABLE V: Test accuracies comparison of the SNN slimming framework with other methods.

Dataset	Pruning method	Training method	Arch.	Acc.(%)	ACC.(%) loss	Step	Connectivity(%)
MNIST	Deep R [29]	Surrogate Gradient	2FC	98.92	-0.36	8	37.14
					-0.56		13.30
					-2.47		4.18
					-9.67		1.10
	Grad R [4]	Surrogate Gradient	2FC	98.92	-0.33	8	25.71
					-0.43		17.94
					-2.02		5.63
					-3.55		3.06
	ADMM-based [3]	Surrogate Gradient	LeNet-5	99.19	-0.01	10	75
					0.01		60
					0.03		50
					-0.43		40
					-2.23		25
	Proposed (channel)	Surrogate Gradient	VGG11	98.61	-0.02	4	75.94
				98.61	-0.02		54.93
				98.62	-0.01		36.72
				98.61	-0.02		34.23
				98.53	-0.10		26.21
CIFAR10	Deep R [29]	Surrogate Gradient	6Conv 2FC	92.84	-1.98	8	5.24
					-2.56		1.95
					-3.53		1.04
	Grad R [4]	Surrogate Gradient	6Conv 2FC	92.84	-0.30	8	28.41
					-0.34		12.04
					-0.81		5.08
					-1.47		2.35
	ADMM-based [3]	Surrogate Gradient	LeNet-5	89.8	-3.52	8	0.73
					0.27		75
					0.18		60
					-0.38		50
					-1.18		40
	State Transition [5]	Surrogate Gradient	6 Conv, 2FC	92.84	-2.15	4	25
					-0.35		2.23
	Spike-Thrift [33]	ANN-to-SNN conversion	VGG16	91.29	-2.63	2500	0.75
					-1.63		40
					-2.56		5
	Proposed (neuron)	Surrogate Gradient	Resnet18	90.15	-3.53	4	2.99
					-0.01		89.36
					-0.32		81.55
					-1.19		71.21
	Proposed (channel)	Surrogate Gradient	VGG16	91.34	-1.58	4	64.95
					-0.47		77.17
					-0.49		57.88
					-0.49		41.71
					-0.49		28.21
					-0.47		19.83
					-0.55		14.80
					-0.60		11.13
CIFAR100	ADMM-based [3]	Surrogate Gradient	11-layer CNN	57.83	-2.70	8	4.82
					-0.52		50
	Spike-Thrift [33]	ANN-to-SNN conversion	VGG16	63.02	-1.21	2500	10
					0.26		88.22
	Proposed (neuron)	Surrogate Gradient	WRN16-4	69.40	0.11	4	81.55
					-0.99		63.63
					-1.68		57.20
					-0.27		76.03
	Proposed (channel)	Surrogate Gradient	VGG16	66.42	-0.29	4	56.53
					-0.33		40.89
					-1.48		28.90
					-3.25		20.24
					-10.06		13.06

the spike-thrift method. This method can also be used on complex datasets CIFAR100. The accuracy of this method only decreases by 0.33 when the connectivity is 40.89%, which is better performance than ADMM-based method. Compared with the Spike-Thrift method, the time steps of our work are much smaller although the sparsity is not better than it.

V. CONCLUSION

The deep structures improve the effectiveness of the SNNs, however, the practical application effect of the model is hindered by the redundancy of the connections. This paper fully considers the characteristics of the human brain, neural circuits could be both pruned and rewired during the development

of the biological brain. We proposed efficient slimming and rewiring methods for deep but sparse structures to SNNs, combining the surrogate gradient method the proposed method could facilitate the time based neural dynamics of spiking model. Inspired by biological plausibility, the proposed methods could not only perform better than some dense models but also have enormous potential for low power consumption, especially on resource-constrained neuromorphic hardware platforms.

REFERENCES

- [1] M. Biebl, C. M. Cooper, J. Winkler, and H. G. Kuhn, "Analysis of neurogenesis and programmed cell death reveals a self-renewing capacity in the adult rat brain," *Neuroscience letters*, vol. 291, no. 1, pp. 17–20, 2000.
- [2] K. L. Spalding, O. Bergmann, K. Alkass, S. Bernard, M. Salehpour, H. B. Huttner, E. Boström, I. Westerlund, C. Vial, B. A. Buchholz *et al.*, "Dynamics of hippocampal neurogenesis in adult humans," *Cell*, vol. 153, no. 6, pp. 1219–1227, 2013.
- [3] L. Deng, Y. Wu, Y. Hu, L. Liang, G. Li, X. Hu, Y. Ding, P. Li, and Y. Xie, "Comprehensive snn compression using admm optimization and activity regularization," *IEEE transactions on neural networks and learning systems*, 2021.
- [4] Y. Chen, Z. Yu, W. Fang, T. Huang, and Y. Tian, "Pruning of deep spiking neural networks through gradient rewiring," *arXiv preprint arXiv:2105.04916*, 2021.
- [5] Y. Chen, Z. Yu, W. Fang, Z. Ma, T. Huang, and Y. Tian, "State transition of dendritic spines improves learning of sparse spiking neural networks," in *International Conference on Machine Learning*. PMLR, 2022, pp. 3701–3715.
- [6] W. Fang, Z. Yu, Y. Chen, T. Huang, T. Masquelier, and Y. Tian, "Deep residual learning in spiking neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 21 056–21 069, 2021.
- [7] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *Advances in neural information processing systems*, vol. 28, 2015.
- [8] P. Lennie, "The cost of cortical computation," *Current biology*, vol. 13, no. 6, pp. 493–497, 2003.
- [9] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2736–2744.
- [10] S. Song, K. D. Miller, and L. F. Abbott, "Competitive hebbian learning through spike-timing-dependent synaptic plasticity," *Nature neuroscience*, vol. 3, no. 9, pp. 919–926, 2000.
- [11] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in computational neuroscience*, vol. 9, p. 99, 2015.
- [12] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, "Stdp-based spiking deep convolutional neural networks for object recognition," *Neural Networks*, vol. 99, pp. 56–67, 2018.
- [13] G. Srinivasan, C. Lee, A. Sengupta, P. Panda, S. S. Sarwar, and K. Roy, "Training deep spiking neural networks for energy-efficient neuromorphic computing," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 8549–8553.
- [14] Q. Xu, Y. Li, J. Shen, P. Zhang, J. K. Liu, H. Tang, and G. Pan, "Hierarchical spiking-based model for efficient image classification with enhanced feature extraction and encoding," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.
- [15] J. Wu, M. Zhang, H. Li, and Y. Chua, "Competitive stdp-based feature representation learning for sound event classification," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.
- [16] Y. Li, Y. Guo, S. Zhang, S. Deng, Y. Hai, and S. Gu, "Differentiable spike: Rethinking gradient-descent for training spiking neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 23 426–23 439, 2021.
- [17] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *Frontiers in neuroscience*, vol. 12, p. 331, 2018.
- [18] H. Zheng, Y. Wu, L. Deng, Y. Hu, and G. Li, "Going deeper with directly-trained larger spiking neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 11 062–11 070.
- [19] S. Deng, Y. Li, S. Zhang, and S. Gu, "Temporal efficient training of spiking neural network via gradient re-weighting," *arXiv preprint arXiv:2202.11946*, 2022.
- [20] M. Zhang, J. Wang, J. Wu, A. Belatreche, B. Amornpaisannon, Z. Zhang, V. P. K. Miriyala, H. Qu, Y. Chua, T. E. Carlson *et al.*, "Rectified linear postsynaptic potential function for backpropagation in deep spiking neural networks," *IEEE transactions on neural networks and learning systems*, vol. 33, no. 5, pp. 1947–1958, 2021.
- [21] Q. Xu, Y. Gao, J. Shen, Y. Li, X. Ran, H. Tang, and G. Pan, "Enhancing adaptive history reserving by spiking convolutional block attention module in recurrent neural networks," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
- [22] Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," *International Journal of Computer Vision*, vol. 113, no. 1, pp. 54–66, 2015.
- [23] Q. Xu, Y. Li, J. Shen, J. K. Liu, H. Tang, and G. Pan, "Constructing deep spiking neural networks from artificial neural networks with knowledge distillation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 7886–7895.
- [24] Y. Hu, H. Tang, and G. Pan, "Spiking deep residual networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2018.
- [25] Y. Li, S. Deng, X. Dong, R. Gong, and S. Gu, "A free lunch from ann: Towards efficient, accurate spiking neural networks calibration," in *International Conference on Machine Learning*. PMLR, 2021, pp. 6316–6325.
- [26] R. Reed, "Pruning algorithms-a survey," *IEEE transactions on Neural Networks*, vol. 4, no. 5, pp. 740–747, 1993.
- [27] T. Gale, E. Elsen, and S. Hooker, "The state of sparsity in deep neural networks," *arXiv preprint arXiv:1902.09574*, 2019.
- [28] A. Renda, J. Frankle, and M. Carbin, "Comparing rewinding and fine-tuning in neural network pruning," *arXiv preprint arXiv:2003.02389*, 2020.
- [29] G. Bellec, D. Kappel, W. Maass, and R. Legenstein, "Deep rewiring: Training very sparse deep networks," *arXiv preprint arXiv:1711.05136*, 2017.
- [30] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.
- [31] Y. Qi, J. Shen, Y. Wang, H. Tang, H. Yu, Z. Wu, G. Pan *et al.*, "Jointly learning network connections and link weights in spiking neural networks," in *IJCAI*, 2018, pp. 1597–1603.
- [32] J. Shen, Q. Xu, J. K. Liu, Y. Wang, G. Pan, and H. Tang, "Esl-snns: An evolutionary structure learning strategy for spiking neural networks," *arXiv preprint arXiv:2306.03693*, 2023.
- [33] S. Kundu, G. Datta, M. Pedram, and P. A. Beerel, "Spike-thrift: Towards energy-efficient deep spiking neural networks by limiting spiking activity via attention-guided compression," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 3953–3962.
- [34] G. Kempermann, S. Jessberger, B. Steiner, and G. Kronenberg, "Milestones of neuronal development in the adult hippocampus," *Trends in neurosciences*, vol. 27, no. 8, pp. 447–452, 2004.
- [35] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, and Y. Tian, "Incorporating learnable membrane time constant to enhance learning of spiking neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 2661–2671.
- [36] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.