



## OPEN ACCESS

## EDITED BY

Huajin Tang,  
Zhejiang University, China

## REVIEWED BY

Anirban Das,  
Intel, United States  
Alberto Marchisio,  
New York University Abu Dhabi,  
United Arab Emirates  
Yihan Lin,  
Tsinghua University, China

## \*CORRESPONDENCE

Zhang Guo  
✉ guozhang@xidian.edu.cn

RECEIVED 15 December 2024

ACCEPTED 03 March 2025

PUBLISHED 25 March 2025

## CITATION

Gou S, Fu J, Sha Y, Cao Z, Guo Z,  
Eshraghian JK, Li R and Jiao L (2025) Dynamic  
spatio-temporal pruning for efficient spiking  
neural networks. *Front. Neurosci.* 19:1545583.  
doi: 10.3389/fnins.2025.1545583

## COPYRIGHT

© 2025 Gou, Fu, Sha, Cao, Guo, Eshraghian, Li  
and Jiao. This is an open-access article  
distributed under the terms of the [Creative  
Commons Attribution License \(CC BY\)](#). The  
use, distribution or reproduction in other  
forums is permitted, provided the original  
author(s) and the copyright owner(s) are  
credited and that the original publication in  
this journal is cited, in accordance with  
accepted academic practice. No use,  
distribution or reproduction is permitted  
which does not comply with these terms.

# Dynamic spatio-temporal pruning for efficient spiking neural networks

Shuiping Gou<sup>1</sup>, Jiahui Fu<sup>1</sup>, Yu Sha<sup>1</sup>, Zhen Cao<sup>1</sup>, Zhang Guo<sup>1\*</sup>,  
Jason K. Eshraghian<sup>2</sup>, Ruimin Li<sup>1</sup> and Licheng Jiao<sup>1</sup>

<sup>1</sup>Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, School of Artificial Intelligence, Xidian University, Xi'an, China, <sup>2</sup>Department of Electrical and Computer Engineering, University of California, Santa Cruz, Santa Cruz, CA, United States

Spiking neural networks (SNNs), which draw from biological neuron models, have the potential to improve the computational efficiency of artificial neural networks (ANNs) due to their event-driven nature and sparse data flow. SNNs rely on dynamical sparsity, in that neurons are trained to activate sparsely to minimize data communication. This is critical when accounting for hardware given the bandwidth limitations between memory and processor. Given that neurons are sparsely activated, weights are less frequently accessed, and potentially can be pruned to less performance degradation in a SNN compared to an equivalent ANN counterpart. Reducing the number of synaptic connections between neurons also relaxes memory demands for neuromorphic processors. In this paper, we propose a spatio-temporal pruning algorithm that dynamically adapts to reduce the temporal redundancy that often exists in SNNs when processing Dynamic Vision Sensor (DVS) datasets. Spatial pruning is executed based on both global parameter statistics and inter-layer parameter count and is shown to reduce model degradation under extreme sparsity. We provide an ablation study that isolates the various components of spatio-temporal pruning, and find that our approach achieves excellent performance across all datasets, with especially high performance on datasets with time-varying features. We achieved a 0.69% improvement on the DVS128 Gesture dataset, despite the common expectation that pruning typically degrades performance. Notably, this enhancement comes with an impressive 98.18% reduction in parameter space and a 50% reduction in time redundancy.

## KEYWORDS

spiking neural networks, spatio-temporal pruning, dynamic vision sensor, sparse connectivity, adaptive temporal dynamics

## 1 Introduction

Spiking Neural Networks (SNNs) are considered as the third generation of neural network models, and many recent studies aim to integrate spike-based processing from biological neurons with deep learning models with the aim of improving energy efficiency (Hassibi and Stork, 1992; Bohte et al., 2002; Roy et al., 2019). The human visual and auditory scenes can transduced into spike signals, a process that SNNs can better simulate in terms of reception and processing of this information (Eshraghian et al., 2021; Guo et al., 2023). When processed on neuromorphic hardware, neurons can be suppressed to prevent adding to data communication and computation demands, and SNNs are thus considered “event-driven” (Modaresi et al., 2023; Merolla et al., 2014; Davies et al., 2018). When compared to Artificial Neural Networks (ANNs), SNNs can leverage

asynchronous, event-driven hardware, and with the right workloads, can drastically reduce the energy cost while retaining the ability to learn from temporal features (Xing et al., 2024). This advantage is currently widely applied in the research of neuromorphic computing algorithms and event data processing (Liu et al., 2024; Zhou and Zhang, 2021; Perez-Peña et al., 2013). However, SNNs most commonly target lightweight, edge-based applications while the demands of neural networks far exceed what many neuromorphic accelerators are capable of processing. For example, Tianjic can process models with approximately 40,000 neurons and 10 million synapses (Pei et al., 2019; Deng et al., 2020), whereas a VGG-16 architecture requires 14 million synapses and 280,000 neurons to run the CIFAR-10 dataset. While there is ongoing research to build scalable, server-based neuromorphic systems (Gonzalez et al., 2024; Vogginger et al., 2024; Orchard et al., 2021), developing compression methods is crucial to reduce the burden of inter-chip data communication: one of the challenges SNNs are aiming to solve.

Pruning is a fundamental method for model compression (Janowsky, 1989; Chauvin, 1988). In neuromorphic computing, pruning is particularly beneficial as it reduces computational load and memory access, addressing hardware constraints such as bandwidth and energy efficiency. Previous pruning practices in ANNs can be categorized based on the scale of pruning, which includes Filter-level, Group-level, and Connection-level pruning (Gao et al., 2021). Pruning at the filter level is structured but tends to have a considerable impact on model performance. For example, the scaling factors of Batch Normalization layers have been linked with synaptic connection strengths, enabling the quantification and subsequent pruning of filters (Lin et al., 2018; Meng et al., 2023; Li et al., 2024). Connection-level pruning often results in less accuracy degradation and high sparsity, however, the sparsity is unstructured. For instance, pruning connections based on the magnitude of their L1-norm weights, where connections with magnitudes below a certain threshold are pruned (Chauvin, 1988), has been effectively utilized in later works (Gale et al., 2019; Han et al., 2015; Renda et al., 2020), including implementations in the SNN domain (Kim et al., 2022; Shi et al., 2019). However, unstructured sparsity makes it difficult to leverage performance gains on GPUs. Specialized hardware has been designed specifically to deploy networks that have undergone unstructured pruning (Han et al., 2015). In SNNs, because event-driven hardware triggers computations only when both incoming spikes and weights are non-zero, SNNs can better leverage the unstructured sparsity brought about by their event-driven characteristics (Merolla et al., 2014; Chen et al., 2022). Unstructured pruning for SNNs remains an important area of research, with many studies conducted in this regard (Kim et al., 2022; Yin et al., 2023; Chen Y. et al., 2023; Han et al., 2022).

In the context of network pruning, the Lottery Ticket Hypothesis (LTH) (Frankle and Carbin, 2018) suggests that within densely and randomly initialized feed-forward neural networks, there exist subnetworks that can achieve performance comparable to the original network within a similar number of iterations when trained independently. These subnetworks are referred to as “winning tickets”. In the domain of ANN pruning, a series of related

work has achieved notable results. Supermasks that encode strong inductive biases are far superior to random masks (Zhou et al., 2019). The significance of the initial state of winning tickets has prompted the use of a “rewinding” mechanism to ensure stability in the training process (Frankle et al., 2019). The discovery that neural networks transition from learning low-frequency to high-frequency components during optimization has led to the concept of “early-bird” lottery tickets in network pruning (Tanaka et al., 2020). In the SNN domain, the LTH has also been analyzed and applied (Kim et al., 2022). However, several challenges remain in the SNN domain, such as how to evaluate network connections in SNNs to identify optimal winning tickets, how to design pruning strategies for redundant frequency-encoded components in temporal datasets, and how to achieve higher sparsity while minimizing accuracy loss for hardware deployment. Our paper will propose a pruning algorithm from these perspectives.

In this paper, we propose a spatio-temporal pruning algorithm tailored for SNNs. Our approach optimizes network efficiency by accounting for sequential data with temporal features along with the spatial structure of the network. We address the challenge of imbalanced pruning across different layers in SNNs which can introduce bottlenecks at high-traffic layers when processing data asynchronously. To address these challenges, we propose the following: (1) Applying the Layer-Adaptive Magnitude-based Pruning Score (LAMPS) (Lee et al., 2020) technique to SNNs to adjust the pruning scale across layers. This approach helps reduce model distortion caused by weight magnitude biases. LAMPS calculates a layer-specific pruning score based on the connection density and weight magnitudes, promoting a balanced reduction in network complexity with minimal performance degradation. (2) For datasets that include temporal features, we address potential redundancy across time by adjusting the SNN’s adaptive temporal dynamics. Our contributions can be summarized as follows:

- 1 We propose a spatio-temporal pruning algorithm for SNNs, which dynamically reduces both spatial and temporal redundancy. The method integrates adaptive temporal pruning and LAMPS-based layer-wise balanced spatial pruning to achieve high sparsity with minimal performance loss.
- 2 We analyze the relationship between KL-Divergence of neuron outputs and model parameter changes, demonstrating the effectiveness of adaptive temporal dynamics in optimizing pruning decisions.
- 3 Experimental results show that a 98% reduction in parameter space across all four datasets. An improvement in accuracy is obtained on the two DVS datasets with time-varying features, upon conducting a 50% and 20% reduction in time redundancy, respectively.

## 2 Related works

In this section, we present the progress and background of pruning algorithms in the field of SNNs. We provide a detailed introduction to the relevant concepts utilized, such as the

LAMPS and how event-camera datasets are integrated into the pruning process.

## 2.1 Spiking neural network pruning

Spiking Neural Networks (SNNs) leverage dynamical sparsity, while pruning enhances static sparsity, leading to extensive exploration in combining these complementary features. Methods derived from Artificial Neural Networks (ANNs) have been adapted to SNNs, such as the work by [Deng et al. \(2021\)](#), which employs the Alternating Direction Method of Multipliers (ADMM) optimization with sparsity regularization to compress SNNs. [Chen et al. \(2022\)](#) proposed a pruning strategy that regulates pruning speed by modifying the threshold function of state transitions, enabling more effective sparsification. [Chowdhury et al. \(2021, 2022\)](#) introduced a method for pruning the temporal dimension of SNNs by analyzing the principal components of the accumulated membrane potential layer by layer. Recent trends in the SNN community favor extreme, unstructured sparsity over structured pruning, as unstructured sparsity is more compatible with neuromorphic hardware ([Meng et al., 2023](#)). [Kim et al. \(2022\)](#) explored the lottery ticket hypothesis in SNNs and implemented the Iterative Magnitude Pruning (IMP) method, demonstrating that high sparsity can be achieved while maintaining accuracy. Some methods also integrate neuron states to prune both neurons and synaptic weights ([Han et al., 2024](#); [Shi et al., 2024a](#); [Han et al., 2025](#)). However, despite advancements in combining dynamical and static sparsity, the full potential of spatio-temporal pruning remains largely untapped. Our approach is inspired by biological synaptic pruning, a key mechanism in human cognitive development, where redundant synapses are selectively eliminated to enhance brain efficiency ([Fleming and McDermott, 2024](#)). This process is crucial for attention ([Das et al., 2024](#)), cognitive control ([Millán et al., 2019](#)), and memory optimization ([Faust et al., 2021](#)), all of which align with the objectives of artificial neural network pruning. Furthermore, spatio-temporal optimization techniques have been extensively studied in deep reinforcement learning (DRL) ([Rühling Cachay et al., 2023](#); [Cao et al., 2020](#)) and biological network modeling ([Lin et al., 2021](#); [Delasalles et al., 2019](#)). By incorporating similar principles, our method seeks to enhance computational efficiency while preserving task performance.

Conventional ANN pruning techniques, including structured and unstructured weight pruning, primarily aim to reduce model size while maintaining accuracy. However, these methods do not account for the event-driven and temporally dynamic characteristics of SNNs ([Eshraghian et al., 2021](#)). Our approach extends beyond static weight sparsity by introducing a temporal pruning component that adaptively removes redundant timesteps based on KL-divergence, a technique not present in traditional ANN pruning. Additionally, while ANN pruning typically applies fixed pruning rates across layers, our method dynamically adjusts pruning based on inter-layer weight distributions using LAMPS, ensuring that critical layers retain sufficient connectivity. By integrating both static and dynamic sparsity mechanisms, our spatio-temporal pruning algorithm effectively

reduces the parameter space and computational cost of SNNs without compromising accuracy, thus bridging the gap between biologically inspired principles and efficient artificial neural network design.

## 2.2 Layer-adaptive magnitude-based pruning score

Following the insights on global and hierarchical pruning scales for SNNs ([Kim et al., 2022](#)), we choose to employ global pruning. However, our approach differs from direct global pruning, in that it first evaluates the significance of connections within layers before introducing inter-layer information. We introduce the LAMPS ([Lee et al., 2020](#)) technique from ANNs into SNNs to score the synaptic weights. For a feed-forward network with depth  $d$ , the synaptic weights can be denoted as  $W^{(1)}, \dots, W^{(i)}, \dots, W^{(d)}$ . We unfold each layer in the  $d$ -layer network into a one-dimensional tensor and sort them in ascending order. For each synaptic weight in these layers, LAMPS is applied as follows:

$$\text{score}(u^{(i)}) := \frac{(W^{(i)}[u^{(i)}])^2}{\sum_{v^{(i)} \geq u^{(i)}} (W^{(i)}[v^{(i)}])^2} \quad (1)$$

where  $u^{(i)}$  and  $v^{(i)}$  represent the weights of two connections after flattening the  $i$ -th layer, with  $v^{(i)} > u^{(i)}$ .  $W^{(i)}[v^{(i)}]$  denotes the  $v^{(i)}$ -th weight in the sorted list of  $W^{(i)}$ . To prune a global  $p\%$  of connections, we can obtain the threshold for each layer and prune the parts where LAMPS score is below this threshold. Overall, this is still a global pruning strategy, but its scoring calculation incorporates intra-layer weight magnitude information. In layers with fewer connections and already strong representational capability (where a smaller number of connections leads to a smaller denominator), the LAMPS score will be higher, preventing further pruning of that layer.

## 2.3 Event camera-based datasets

The DVS128-Gesture ([Li et al., 2017](#)) dataset consists of various hand gestures in different lighting conditions and subjects captured using an event camera. The CIFAR10-DVS ([Amir et al., 2017](#)) dataset is created by capturing the CIFAR-10 dataset with an event camera, through repeated closed-loop smooth (RCLS) motion, generating rich local intensity changes. A common method for processing DVS datasets at present is to integrate events over fixed time intervals, transforming the original event stream into frame-based data ([Fang et al., 2023](#)). Firstly, we denote the Event data as  $E(x, y, t, p_i)$  and divide it evenly into  $T$  segments. A given frame in the integrated frame data is denoted as  $F(j)$ . The value at the position  $(p, x, y)$  is  $F(j, p, x, y)$ . The frame  $F(j)$  is obtained by integrating the events in the event stream with indices ranging between  $j_l$  and  $j_r$ , where

$$j_l = \left\lfloor \frac{N}{T} \right\rfloor \cdot j \quad (2)$$

$$j_r = \begin{cases} \lfloor \frac{N}{T} \rfloor \cdot (j+1), & \text{if } j < T-1 \\ N, & \text{if } j = T-1 \end{cases} \quad (3)$$

$$F(j, p, x, y) = \sum_{i=j_l}^{j_r-1} \mathcal{I}_{p,x,y}(p_i, x_i, y_i) \quad (4)$$

where  $\lfloor \cdot \rfloor$  is downward rounding.  $\mathcal{I}_{p,x,y}(p_i, x_i, y_i)$  is the indicator function, when  $(p, x, y) = (p_i, x_i, y_i)$  takes the value 1.

### 3 Method

In this section, we demonstrate the specific implementation process and theoretical basis of the proposed spatio-temporal pruning scheme for SNNs. Firstly, we introduce the SNNs and the adopted Leaky Integrate-and-Fire (LIF) neuron model. Subsequently, we describe the framework of the proposed spatio-temporal pruning scheme. Following that, we design a unique pruning method for SNNs, with optimization functions and constraints specifically designed for it. Lastly, we present an evaluation design for the pruning algorithm, considering both the output and the variations in network parameters.

#### 3.1 Spiking neural network

SNNs serve as a simulation network for the spatio-temporal dynamic behavior of biological neural circuits, distinguishing themselves from traditional ANNs primarily through neuron designs inspired by biological mechanisms (Hu et al., 2023) and their inherent capacity for temporal information representation. In this study, we focus on the LIF neuron (Hunsberger and Eliasmith, 2015), which is not only the most commonly used model in SNNs but also represents a compromise between biological complexity and computational feasibility, making it suitable for large-scale SNN simulations (Fang et al., 2023). The behavior of the classic LIF model can be modeled as follows:

$$\tau \frac{du}{dt} = -(u - u_{rest}) + R \cdot I(t), u < V_{th} \quad (5)$$

where  $\tau$  and  $R$  are the time constant and resistance,  $I(t)$  are the input current from the pre-synaptic membrane potential.  $V_{th}$  represents the neuron membrane potential firing threshold, and  $u_{rest}$  represents the resting potential membrane potential. If the membrane potential  $u$  exceeds  $V_{th}$ , a spike is fired and  $u$  is reset to  $u_{rest}$ . For numerical simulations of LIF neurons, we need to consider a discrete version of the parameter dynamics. Assuming  $u_{rest}$  is 0 (Fang et al., 2023; Li et al., 2021), the spike firing function and hard reset can be expressed as:

$$\begin{aligned} a(t+1) &= \Theta(u(t+1) - V_{th}), \\ u(t+1) &\leftarrow u(t+1) \cdot (1 - a(t+1)) \end{aligned} \quad (6)$$

where  $\Theta(\cdot)$  denotes the Heaviside step function, which emits a spike when  $u(t+1)$  exceeds the threshold function. The output spike  $a(t+1)$  acts as a messenger for information propagation between network layers and affect the subsequent layer.

#### 3.2 Overall architecture

The overall process of the proposed spatiotemporal pruning framework is illustrated in Figure 1. Given a sequential dataset such as the DVS dataset, it is initially integrated into frame-level representations using a standard event data integration operation. The frame sequence can be represented as  $D \in \mathbb{R}^{T \times 2 \times W \times H}$ , where  $T$  denotes the number of frames. The frame sequence  $D$  is then fed into an SNN for classification, and the output layer of the network is monitored.

To determine the optimal decision timestep, we analyze the model's classification confidence across different time lengths. Specifically, we extract sequences of various lengths  $D_{1 \sim t}$  and compute the accumulated spike responses from the output layer. To quantify the similarity between the model output and the true labels over time, we compute the KL-divergence between the predicted class probabilities and the ground-truth distribution:

$$KL(p(y_{1:t}|x_{1:t}, \theta) || p(y^*)) = \sum_c p(y_{c,1:t}|x_{1:t}, \theta) \log \frac{p(y_{c,1:t}|x_{1:t}, \theta)}{p(y_c^*)} \quad (7)$$

where  $p(y_{1:t}|x_{1:t}, \theta)$  represents the model's predicted probability distribution over classes up to timestep  $t$ , and  $p(y^*)$  denotes the one-hot encoded ground-truth label distribution. The KL divergence measures the discrepancy between these two distributions, providing an indication of the model's decision stability over time.

To facilitate threshold-based pruning, the sequence of computed KL divergences is subjected to min-max normalization:

$$NormKL_t = \frac{KL(p(y_{1:t}|x_{1:t}, \theta) || p(y^*)) - KL_{min}}{KL_{max} - KL_{min}} \quad (8)$$

where  $KL_{min}$  and  $KL_{max}$  are the minimum and maximum KL divergence values observed in the sequence. We then compare this normalized KL divergence with a predefined threshold  $\lambda$  to determine the decision timestep  $t'$ :

$$t' = \min\{t \mid NormKL_t < \lambda\} \quad (9)$$

This means that we select the earliest timestep where the KL divergence stabilizes below the threshold, ensuring that the model has accumulated sufficient information for a reliable classification decision. Applying this decision timestep in the temporal dimension to the sequence data  $D$  yields a new sequence  $D_{1 \sim t'}$ .

After completing the pruning in the temporal dimension, we proceed to sparsify the network weights in the spatial dimension. Since the relative importance of weights varies across layers, we leverage the LAMPS to compute the weight significance of each layer in the SNN. The synaptic weights are ranked globally, and at each pruning iteration, the least important  $p\%$  of the weights are pruned while maintaining a balanced pruning ratio across layers. The iterative pruning process ensures that the network achieves high sparsity while preserving performance. The impact of our pruning strategy compared to the IMP

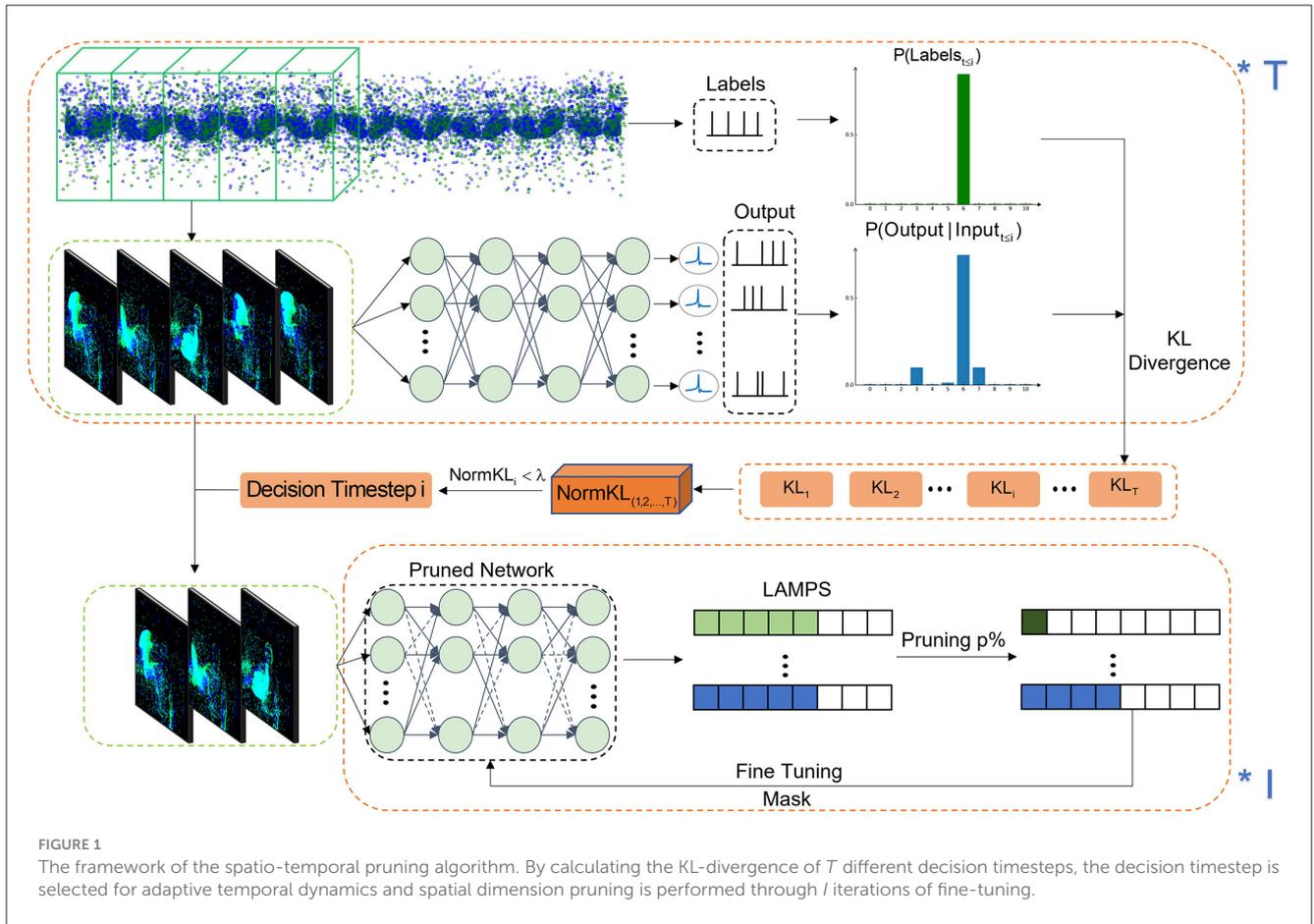


FIGURE 1 The framework of the spatio-temporal pruning algorithm. By calculating the KL-divergence of  $T$  different decision timesteps, the decision timestep is selected for adaptive temporal dynamics and spatial dimension pruning is performed through  $I$  iterations of fine-tuning.

(Kim et al., 2022) method is illustrated in Figure 2. Compared to IMP, which prunes connections based solely on weight magnitudes in an iterative manner, our proposed spatio-temporal pruning algorithm additionally considers inter-layer weight distribution and temporal redundancy, leading to a more balanced and efficient pruning strategy.

The full pruning procedure is detailed in Algorithm 1. The TrainSNN function utilizes the widely adopted Spatio-Temporal Backpropagation (STBP) algorithm for training, incorporating the Atan function as a surrogate gradient to handle the non-differentiability of spikes (Wu et al., 2018). The network architectures and hyperparameters used follow the configurations provided in SpikingJelly for each corresponding dataset (Fang et al., 2023).

### 3.3 Optimization for sparsity

In general, the pruning methods for SNNs are constrained by the conditions of actual deployment, aiming to increase the sparsity of the network while minimizing accuracy degradation. In this section, we analyze the problem from two perspectives: the sparsity of weights in the spatial dimension and the necessity of timesteps in the temporal dimension. The pruning constraint optimization problem for SNNs is formulated

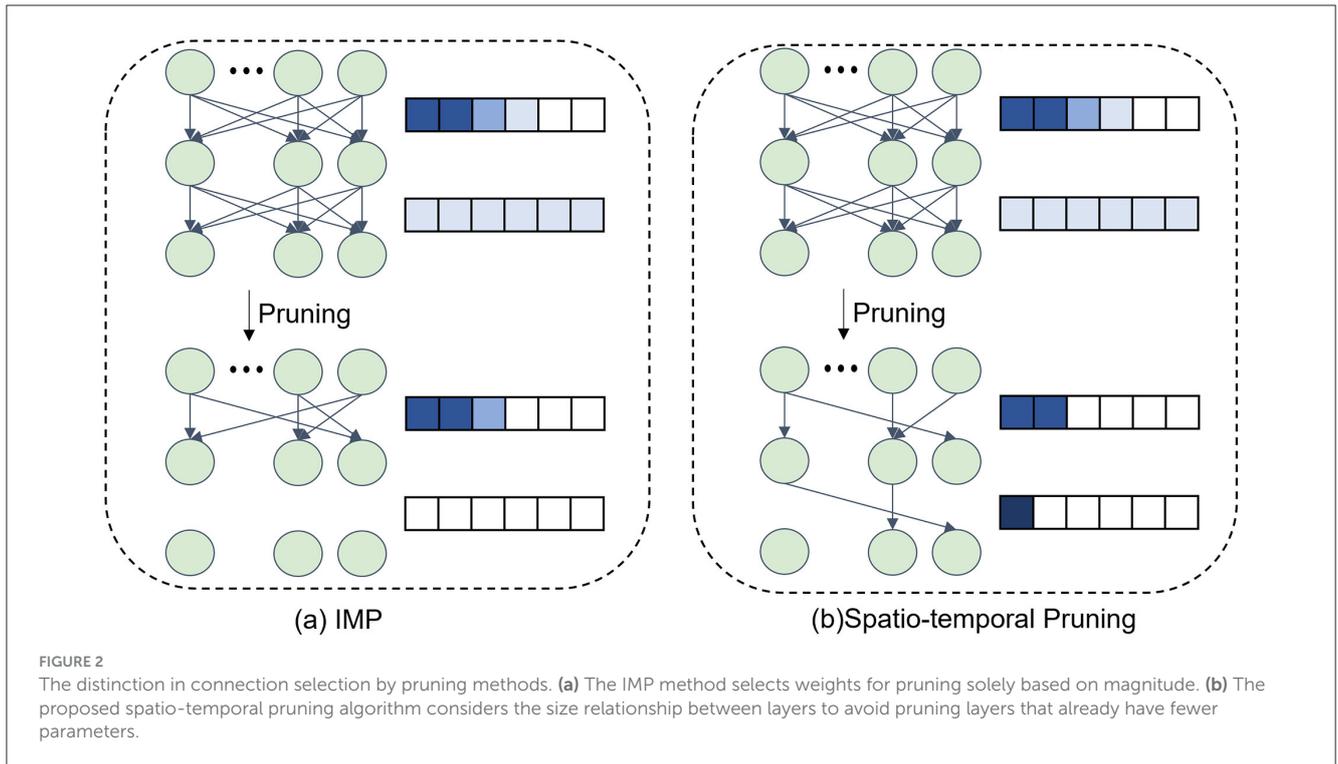
as follows:

$$\begin{aligned} & \arg \min_{\theta, t} \mathcal{L}(\theta, t) \\ & s.t. \quad \ell_0(\theta) \leq \ell_0(\theta^+) \\ & \quad \quad t \leq T_{max} \end{aligned} \tag{10}$$

where  $\mathcal{L}(\cdot)$  represents the loss function of the model.  $\theta$  and  $t$  respectively denote the optimization parameters of SNNs and the timesteps required by the model.  $\ell_0(\theta^+)$  represents the L0-norm of the model parameters that meet the deployment constraints, implying that the actual deployment model parameters  $\theta$  should be more sparse than  $\theta^+$ . The constraint  $\ell_0(\theta) \leq \ell_0(\theta^+)$  ensures that the number of nonzero parameters remains within the permissible range for hardware deployment, considering factors such as limited memory bandwidth, energy efficiency, and real-time inference constraints.  $T_{max}$  represents the maximum allowable timesteps, ensuring that the computation cost remains feasible.

### 3.4 Adaptive temporal dynamics

In Algorithm 1, selecting the decision timestep (line 8-14) is a crucial step in temporal pruning. This section elaborates on how to determine the optimal classification timestep by leveraging KL-divergence. Considering the output of SNNs in specific tasks, the optimal classification should occur as early as



possible while ensuring correctness. The optimization objective function is formulated as:

$$\arg \min_{\theta, t} KL [p(y_{1:t} | x_{1:t}, \theta); p(y_{1:t} | x_{1:t}, \theta^*)] \quad (11)$$

where  $\theta^*$  represents the globally optimized network parameters that yield the best classification performance.  $x_{1:t}$  represents the input data from timestep 1 to  $t$ , and  $y_{1:t}$  represents the corresponding model output. It is worth noting that this can be regarded as two random variables. The probability function  $p(\cdot)$  encodes the model's confidence in different classes.

To determine the optimal decision timestep, we normalize the KL-divergence sequence across different timesteps using min-max normalization:

$$NormKL_t = \frac{KL [p(y_{1:t} | x_{1:t}, \theta); p(y_{1:t} | x_{1:t}, \theta^*)] - KL_{\min}}{KL_{\max} - KL_{\min}} \quad (12)$$

where  $KL_{\max}$  and  $KL_{\min}$  denote the maximum and minimum KL-divergence values observed across all timesteps.

We define a threshold  $\lambda$  for selecting the decision timestep:

$$t' = \min\{t | NormKL_t < \lambda\} \quad (13)$$

This ensures that we select the earliest timestep where the normalized KL-divergence falls below the threshold, indicating that the model has accumulated sufficient information for a reliable classification decision.

Given these constraints, the final optimization problem is formulated as:

$$\begin{aligned} \arg \min_{\theta, t} KL [p(y_{1:t} | x_{1:t}, \theta); p(y_{1:t} | x_{1:t}, \theta^*)] \\ \text{s.t. } \ell_0(\theta) \leq \ell_0(\theta^+) \\ t \leq T_{\max} \\ NormKL_t < \lambda \end{aligned} \quad (14)$$

### 3.5 Dynamics from the information perspective

The Fisher Information Matrix (FIM) is widely used to quantify the information content of model parameters. In our case, it helps assess how much information changes across timesteps. To approximate the KL-divergence function around the optimal parameters  $\theta^*$ , we apply a second-order Taylor expansion:

$$\begin{aligned} KL [p(y_{1:t} | x_{1:t}, \theta); p(y_{1:t} | x_{1:t}, \theta^*)] \\ \approx KL [p(y_{1:t} | x_{1:t}, \theta); p(y_{1:t} | x_{1:t}, \theta^*)]_{\theta^*=\theta} \\ + \nabla_{\theta^*} KL [p(y_{1:t} | x_{1:t}, \theta); p(y_{1:t} | x_{1:t}, \theta^*)]_{\theta^*=\theta}^T (\theta^* - \theta) \\ + \frac{1}{2} (\theta^* - \theta)^T \nabla_{\theta^*}^2 KL [p(y_{1:t} | x_{1:t}, \theta); p(y_{1:t} | x_{1:t}, \theta^*)]_{\theta^*=\theta} (\theta^* - \theta) \\ = \frac{1}{2} (\theta^* - \theta)^T F_t (\theta^* - \theta) \end{aligned} \quad (15)$$

where the first and second terms vanish, leaving only the Fisher Information Matrix  $F_t$  in the final term. This demonstrates that the KL-divergence is directly related to the FIM, reinforcing the intuition that pruning based on KL-divergence preserves informative parameters.

```

Input: Model SNN; Training data  $D$ ; Original timestep  $T$ ; Data labels  $L$ ; Threshold  $\lambda$  for normalized KL-divergence; Pruning ratio  $p\%$  per round; Number of pruning rounds  $R$ ; Number of training epochs  $E$  per round; Rewind epoch  $E_{rewind}$ 
Output: Pruned SNN model
1 SNN  $\leftarrow$  TrainSNN(SNN,  $D$ ,  $E$ ) // Train for  $E$  epochs
2 KLList  $\leftarrow$  [ ]
3 for  $t \leftarrow 1$  to  $T$  do
4   KLList.append(KLDivergence(SNN( $D_{1:t}$ ),  $L_{1:t}$ )) // Compute the KL-divergence and add it to the list.
5 end
6 NormKLList  $\leftarrow$  Normalize(KLList)
7 DecisionTimestep  $\leftarrow T$  // Default value in case threshold  $\lambda$  is not met
8 // Compute the decision timestep (see Equation 9)
9 for  $t \leftarrow 1$  to  $T$  do
10  if NormKLList[ $t-1$ ]  $<$   $\lambda$  then
11    DecisionTimestep  $\leftarrow t$ 
12    break
13  end
14 end
15 for  $r \leftarrow 1$  to  $R$  do
16  TrainSNN(SNN,  $D_{1:DecisionTimestep}$ ,  $E - E_{rewind}$ )
17  [ $P_1, P_2, \dots, P_{len(SNN)}$ ]  $\leftarrow$  Flattening SNN by layers
18  LAMPS  $\leftarrow$  [ ]
19  // Compute LAMPS for each layer based on parameter ranking and cumulative sum (see Equation 1)
20  for  $j \leftarrow 1$  to  $len(SNN)$  do
21    Ranked $P_j \leftarrow$  RankParameters( $P_j$ )
22    CumRanked $P_j \leftarrow$  CumulativeSum(Ranked $P_j^2$ )
23    LAMPS.append(Ranked $P_j^2$ /CumRanked $P_j$ )
24  end
25  SNN  $\leftarrow$  PruneSNN(SNN,  $p$ , LAMPS) // Prune parameters with lowest  $p\%$  LAMPS
26 end
27 return SNN

```

Algorithm 1. Two-stage pruning framework.

The FIM is essential for understanding the sensitivity of network parameters. However, directly computing the true FIM is computationally intractable due to its high dimensionality and dependence on the full dataset. In practice, the empirical FIM is used to approximate the true FIM (Singh and Alistarh, 2020; Kim et al., 2023; Kunstner et al., 2019):

$$\hat{F}_t = \frac{1}{N} \sum_{n=1}^N \|\nabla_{\theta} \log p(y_{1:t} | x_{1:t}, \theta)\|^2 \quad (16)$$

where  $N$  is the batch size, and  $\nabla_{\theta}$  represents the gradient computed during training. Since the optimization process relies on minimizing the divergence between predicted and true outputs,

the empirical FIM provides a meaningful measure of parameter importance.

This establishes a direct relationship between KL-divergence-based Adaptive Temporal Dynamics and changes in model parameters, allowing for an efficient pruning strategy that dynamically adapts to both spatial and temporal constraints. This theoretical foundation justifies the decision timestep selection process in Algorithm 1 (line 8–14), ensuring that pruning decisions are made based on a principled optimization framework.

## 4 Experiments

The proposed method is evaluated on static image classification benchmarks (CIFAR-10 and CIFAR-100) as well as DVS data benchmarks (DVS128 Gesture and CIFAR10-DVS). The implementation of the pruning is based on the open-source SNN framework SpikingJelly (Fang et al., 2023). Firstly, we explain the detailed settings and experimental environment. Then, we conduct ablation experiments on the temporal and spatial modules of the proposed spatio-temporal pruning algorithm, calculating the performance of the ablated modules and some evaluation metrics. Finally, we perform experimental statistics on the neuron firing rates and the model's synaptic weight connections before and after the use of the spatio-temporal pruning algorithm.

### 4.1 Settings

We use four datasets in our experiments: CIFAR10/100, DVS128-Gesture, and CIFAR10-DVS. The basic information of each dataset and the network architectures used are shown in Table 1. The experiments are conducted on an Ubuntu 20.04 LTS system with two NVIDIA RTX 4090 GPUs. The specific parameters and training pipeline settings are as follows:

#### 4.1.1 CIFAR-10/100

For the CIFAR-10 dataset, we use the ResNet19 network. For the CIFAR-100 dataset, we use both the VGG16 and ResNet19 networks, following previous work (Kim et al., 2022). The first convolutional layer converts static images into spike form. Data augmentation techniques such as RandomCrop, RandomHorizontalFlip, Cutout, and Normalize are applied (Kim et al., 2022). We set the batch size to 128 and the timestep to 5. The models are trained using the SGD optimizer with a learning rate of 0.3, momentum of 0.9, and weight decay of 0.0005. The cosine learning rate scheduling strategy is adopted.

#### 4.1.2 DVS128-Gesture

For the DVS128Gesture dataset, we use the official implementation from SpikingJelly (Fang et al., 2023). The batch size is set to 8 and the timestep to 20. The Adam optimizer is used with a learning rate of 0.001. The surrogate gradient function is the Atan function with a parameter  $\alpha$  of 2.0. The hyperparameter  $\lambda$  is set to 0.01, and the cosine learning rate scheduling strategy is used.

TABLE 1 The basic information of the datasets and the corresponding structures.

Dataset	Image	Training samples	Testing samples	Category	Structure	Surrogate function
CIFAR10	32×32×3	50,000	1,000	10	ResNet19	ATan
CIFAR100	32×32×3	50,000	10,000	100	VGG16 & ResNet19	ATan
DVS128-Gesture	128×128×2	1,176	288	11	5Conv, 2FC	ATan
CIFAR10-DVS	128×128×2	10,000	1,000	10	4Conv, 2FC & VGGsNN	ATan

TABLE 2 Comparison with other algorithms on static datasets.

Dataset	Pruning method	Structure	Base acc(%)	Sparsity(%)	Param. (M)	ΔAccuracy(%)
CIFAR10	SCCD-SNN (Meng et al., 2023)	ResNet20	92.14	70.00	1.30	-2.78
				94.25	0.73	-1.31
	Deep R (Bellec et al., 2017)	ResNet19	93.22	97.56	0.31	-2.10
				99.27	0.26	-3.52
	Grad R (Chen et al., 2021)	6 Conv, 2 FC	92.84	97.65	0.86	-1.47
				99.27	0.26	-3.52
	IMP (Kim et al., 2022)	ResNet19	93.22	97.54	0.31	-0.04
				98.13	<b>0.23</b>	-0.79
	Ours	ResNet19	92.40	96.75	0.41	<b>0.42</b>
				97.54	0.31	<b>0.34</b>
98.13				<b>0.23</b>	<b>-0.18</b>	
CIFAR100	Grad R (Chen et al., 2021)	ResNet19	71.34	94.92	0.64	-3.87
				97.65	0.30	-4.03
	IMP (Kim et al., 2022)	VGG16	69.19	96.75	0.48	-1.84
				97.54	0.36	-2.31
				98.13	0.27	-3.34
	ResNet19	ResNet19	71.34	95.69	0.55	-0.89
				97.54	0.31	-2.29
				98.13	<b>0.24</b>	-3.99
	Ours	VGG16	69.19	96.75	0.48	<b>-1.12</b>
				97.54	0.36	<b>-1.65</b>
98.13				0.27	<b>-2.86</b>	
ResNet19		ResNet19	71.58	95.69	0.55	<b>-0.68</b>
			97.54	0.31	<b>-2.01</b>	
			98.13	<b>0.24</b>	<b>-2.86</b>	

Bold values indicate cases where the parameter count is relatively low or the accuracy degradation is minimal, demonstrating efficient pruning results.

### 4.1.3 CIFAR10-DVS

We adopt the official implementation from SpikingJelly (Fang et al., 2023) with a 4Conv, 2FC network. Under this network architecture, the batch size is set to 16. Additionally, we implement the VGGsNN network (Shi et al., 2024a), where we follow the same TET loss (Deng et al., 2022) and data augmentation (Li et al., 2022; Shi et al., 2024a) strategies as in Shi et al. (2024a), with the batch size set to 64. In both network architectures, the timestep is set to 10. The model is trained using the Adam optimizer with a learning rate of 0.001. Similar to the DVS128Gesture dataset, the surrogate gradient function is ATan with  $\alpha = 2.0$ , and the hyperparameter  $\lambda$  is set to 0.01. The cosine learning rate scheduling strategy is applied.

## 4.2 Performance

Table 2 demonstrates the performance variations of our spatial pruning algorithm on static images at different levels of sparsity. It is evident that our method can maintain good performance while achieving higher sparsity. On the CIFAR10 dataset, when sparsity reaches 98.13%, it can still maintain a high accuracy rate (with a reduction of about 0.18%). On the larger CIFAR100 dataset, we compared two network architectures. Our method outperforms the comparison methods at higher levels of sparsity. On ResNet19, when the sparsity rate reaches 98.13%, our method exhibits very little loss in accuracy, which is a significant improvement (greater than 1%) compared to the comparison algorithm (Kim et al., 2022).

TABLE 3 Comparison with other algorithms on the DVS datasets.

Dataset	Pruning method	Structure	Base acc (%)	Sparsity (%)	Param. (M)	Accuracy (%)	$\Delta$ Accuracy (%)	
DVS128-Gesture	STDS (Chen et al., 2022)	5Conv, 2FC	95.83	94.50	0.094	94.44	-1.39	
				96.90	0.053	92.36	-3.47	
				97.30	0.046	83.68	-12.15	
	IMP (Kim et al., 2022)	5Conv, 2FC	95.83	95.83	96.72	0.056	91.32	-4.51
					97.51	0.042	86.11	-9.72
					98.11	0.032	78.47	-17.36
	SCCD-SNN (Meng et al., 2023)	8Conv, 1FC	94.44	94.44	20.00	0.136	94.44	0.00
					50.00	0.085	91.67	-2.77
					90.00	<b>0.017</b>	75.35	-19.09
	Our(+S)	5Conv, 2FC	95.83	95.83	96.72	0.056	<b>95.83</b>	<b>0.00</b>
					97.51	0.042	<b>96.18</b>	<b>0.35</b>
					98.11	0.032	95.13	-0.70
	Our(+S+T 10/20)	5Conv, 2FC	95.83	95.83	96.72	0.056	<b>95.83</b>	<b>0.00</b>
					97.51	0.042	<b>96.18</b>	<b>0.35</b>
					98.11	0.032	<b>96.52</b>	<b>0.69</b>
CIFAR10-DVS	STDS (Chen et al., 2022)	4Conv, 2FC	71.70	84.12	0.745	71.37	-0.33	
				95.29	0.221	69.05	-2.65	
				96.20	0.178	64.92	-6.78	
	IMP (Kim et al., 2022)	4Conv, 2FC	71.70	71.70	96.80	0.150	70.70	-1.00
					97.59	0.113	68.00	-3.70
					98.18	<b>0.085</b>	63.30	-8.40
	SCCD-SNN (Meng et al., 2023)	8Conv, 1FC	72.60	72.60	20.00	0.952	73.60	1.00
					50.00	0.595	71.04	-1.56
					90.00	0.119	66.64	-5.96
	TEE-SNN (Shi et al., 2024a)	VGGsNN	82.50	82.50	93.20	0.658	81.90	-0.5
					95.54	0.432	81.00	-1.40
					98.73	<b>0.123</b>	79.00	-3.40
	Our(+S)	4Conv, 2FC	71.70	71.70	96.8	0.150	72.48	0.78
					97.59	0.113	<b>72.78</b>	<b>1.08</b>
					98.18	<b>0.085</b>	<b>72.28</b>	<b>0.58</b>
	Our(+S+T 8/10)	4Conv, 2FC	71.70	71.70	96.80	0.150	<b>72.88</b>	<b>1.18</b>
					97.59	0.113	72.58	0.88
					98.18	<b>0.085</b>	71.98	0.28
	Our(+S+T 8/10)	VGGsNN	82.50	82.50	94.31	<b>0.551</b>	<b>83.7</b>	<b>1.20</b>
					95.72	<b>0.414</b>	<b>83.20</b>	<b>0.70</b>
					98.61	0.135	81.50	-1.00

Bold values indicate cases where the parameter count is relatively low or the accuracy degradation is minimal, demonstrating efficient pruning results.

The performance of our model under different levels of sparsity on the DVS dataset and the numerical values of accuracy degradation are shown in Table 3. The “+S” indicates the results after applying spatial pruning strategies, and “+T” represents the outcomes after incorporating the Adaptive Temporal Dynamics strategy. “\*/\*” represents the decision timestep and total timestep

of Adaptive Temporal Dynamics. For example, “10/20” indicates that the model originally needs to calculate 20 timesteps, but the decision timestep is set to 10. After performing spatial pruning on the DVS128-Gesture dataset, our model was able to achieve good performance at higher sparsity levels. The performance started to decline when the Pruning rate exceeded 98%. Surprisingly, after

TABLE 4 Ablation experiments on different datasets using temporal and spatial pruning algorithms.

Dataset	Spatial	Temporal	Accuracy (%)	Sparsity (%)	Decision timestep	Param. (M)	SOPs (M)
DVS128-Gesture			95.83	0	20	1.70	601.52
	✓		95.83	98.11	20	0.03	30.62
		✓	96.18	0	10	1.70	348.82
	✓	✓	96.52	98.11	10	0.03	15.41
CIFAR10-DVS			71.70	0	10	4.69	579.08
	✓		72.28	98.18	10	0.08	62.57
		✓	71.88	0	8	4.69	479.54
	✓	✓	71.98	98.18	8	0.08	53.93

adding Adaptive Temporal Dynamics, by training and inferring in the temporal dimension with a reduced decision timestep from 20 to 10 (which directly halves the synaptic operations), the model's performance improved. On the CIFAR10-DVS dataset, our model still maintained good performance at high sparsity levels (greater than 98%) after spatial pruning. With the introduction of Adaptive Temporal Dynamics, the decision timestep used for training and inference in the temporal dimension was reduced from 10 to 8, and the model still achieved excellent performance.

In comparison with other methods, our spatial strategy exhibited lower performance degradation under high sparsity conditions in both static datasets, as shown in Table 2. Our spatiotemporal strategy was successful in the DVS dataset, achieving outstanding results in the CIFAR10-DVS dataset derived from static data (Li et al., 2017). In the DVS128-Gesture dataset, which contains richer temporal information captured by DVS cameras (Shi et al., 2024b; Anumasa et al., 2024), our strategy achieved remarkable performance improvements compared to other methods, as illustrated in Table 3.

### 4.3 Ablation study

We conduct ablation experiments on spatio-temporal pruning algorithms on two DVS datasets as shown in Table 4. The network architectures used were still 5Conv, 2FC and 4Conv, 2FC, respectively. The variables were the presence or absence of spatial and temporal strategies. We compared the model on five metrics: number of parameters, sparsity, synaptic operations (SOPs), accuracy, and decision timestep. SOPs is the number of spike-based AC operations (Chen G. et al., 2023; Zhou et al., 2023).  $SOPs = \sum_i s_i c_i$  denotes the total number of synaptic operations. For each presynaptic neuron  $i$  in an SNN,  $s_i$  denotes the number of spikes fired by this neuron, and  $c_i$  denotes the number of synaptic connections from this presynaptic neuron.

The ablation study in Table 4 revealed that our spatio-temporal pruning scheme could reduce synaptic operations when using either the temporal or spatial strategies alone. For the spatial strategy, higher sparsity levels often directly dictate the change in the number of parameters, thereby reducing model computations. For the temporal strategy, the computed decision timestep can decrease the number of steps required for the model to make a decision, thus reducing the model's synaptic operations. The

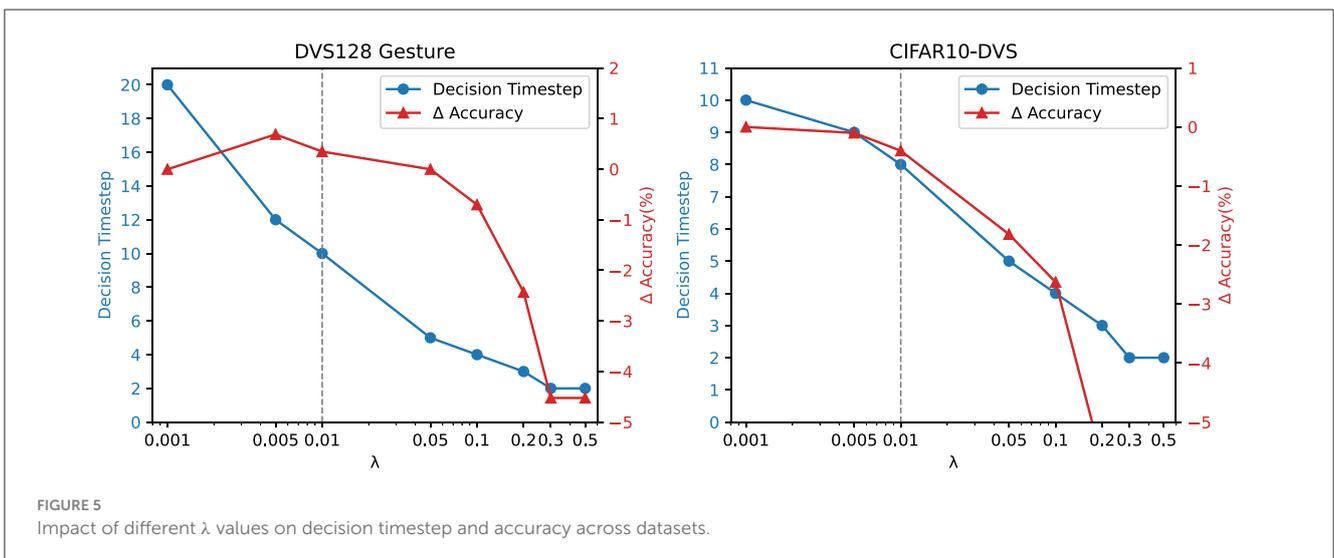
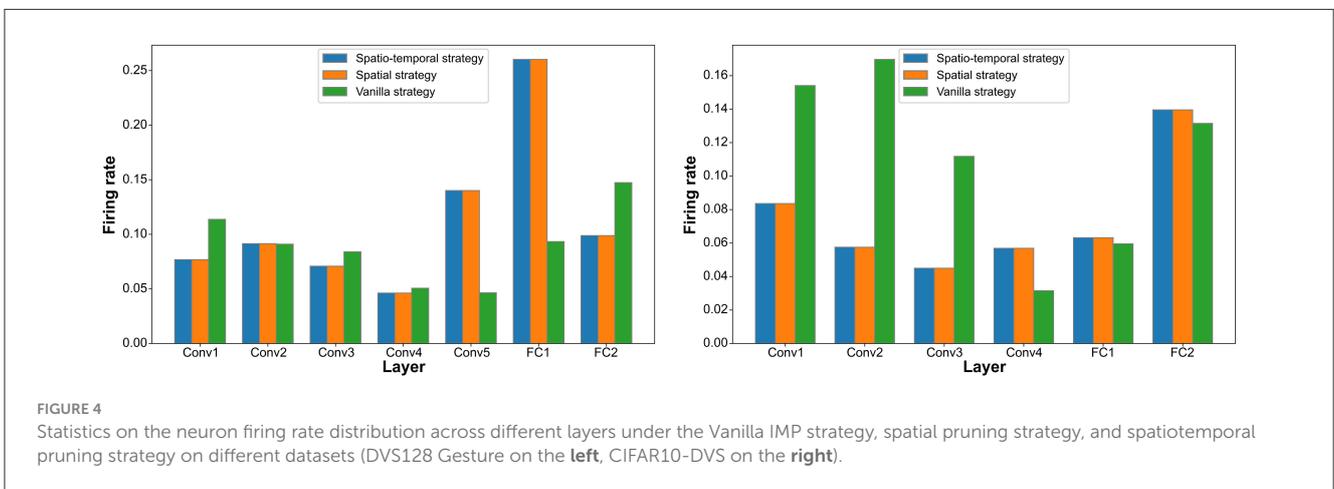
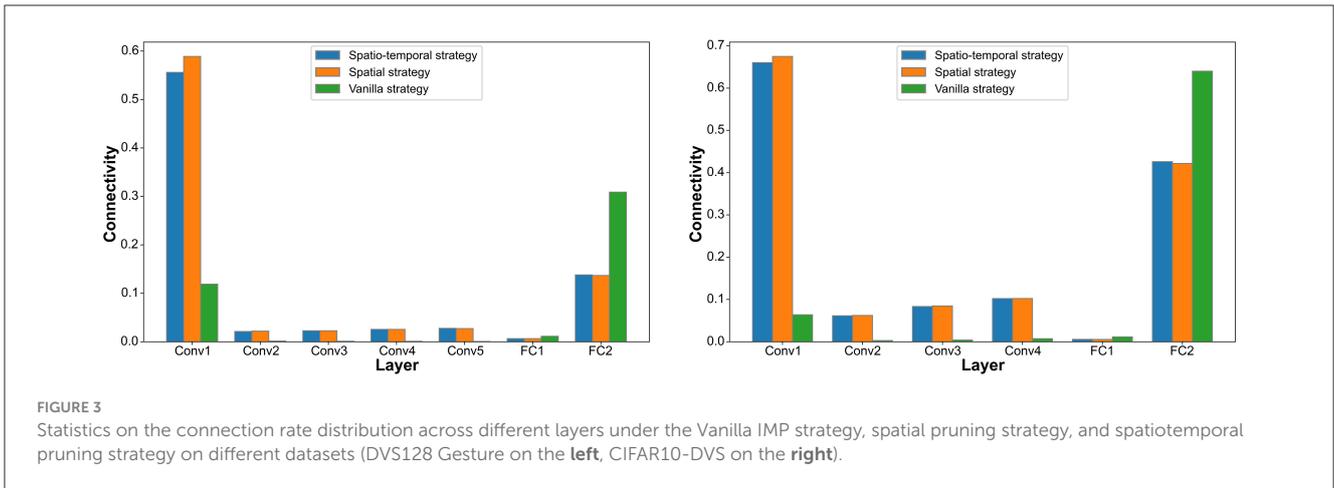
changes in accuracy indicate that both temporal and spatial strategies achieved good performance and low synaptic operations separately. Combining the temporal and spatial strategies often yields a model with even lower synaptic operations while maintaining good performance.

### 4.4 Firing rate and connectivity

In the DVS128 Gesture and CIFAR10-DVS datasets, we analyze the connectivity and firing rates of different layers under the spatiotemporal pruning strategy, spatial pruning strategy, and Vanilla IMP strategy. As shown in Figures 3, 4, our proposed spatiotemporal pruning method tends to maintain a higher connectivity rate in shallower layers while reducing the number of connections in deeper layers. This facilitates feature extraction, as the connectivity in shallow layers often plays a more decisive role in network performance, and the feature extraction capability of shallow layers typically determines that of deeper layers. Additionally, since fully connected layers account for a significant proportion of the network parameters, our method tends to apply a more aggressive pruning strategy to them. Specifically, as shown in Figure 3, the spatiotemporal pruning strategy applies more targeted pruning to the FC1 layer, which has a higher proportion of total connections, while retaining more connections in the Conv1 layer, which has fewer total connections but plays a more crucial role. This benefit stems from the pruning strategy's ability to better estimate the remaining connection ratio for each layer. As shown in Figure 4, it can also be observed that fewer connections generally correspond to a higher neuronal firing rate, compensating for the representational capacity loss caused by weight pruning.

## 5 Analysis and discussion

In this section, we analyze the proposed spatio-temporal pruning algorithm. Firstly, we conduct a parameter analysis on the hyperparameters proposed by the model. Secondly, we analyze the experimental results brought about by Adaptive Temporal Dynamics. Then, we examine the impact of the model's total timesteps on the pruning algorithm. Finally, we conduct a statistical analysis of the magnitude of the model parameters and analyze the limitations of the algorithm.



### 5.1 Parameter analysis

We conducted an experimental analysis on the hyperparameter  $\lambda$  used in this paper, as shown in Figure 5. The hyperparameter  $\lambda$  serves as a threshold for the normalized KL-divergence,

controlling the decision timestep in Adaptive Temporal Dynamics. A higher  $\lambda$  leads to a shorter inference time, as the decision timestep is reduced, allowing the model to make earlier classification decisions. Conversely, a lower  $\lambda$  results in a longer inference time, as the model waits for more timesteps to

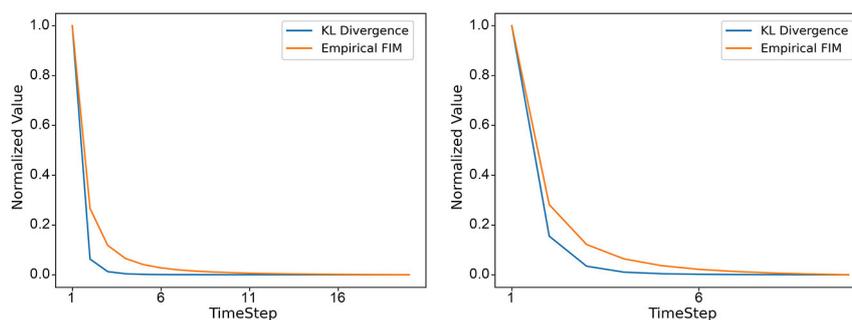


FIGURE 6 On different datasets (DVS128 Gesture on the left, CIFAR10-DVS on the right), the changes in Norm-KL and empirical FIM with the selection of different decision timesteps.

TABLE 5 Experimental results on the impact of different timesteps on the pruning algorithm.

TimeStep	Temporal	Spatial	Decision timestep	Sparsity(%)	Accuracy(%)
10			10	0	94.09
	✓		8	0	94.44
	✓	✓	8	98.11	92.01
15			15	0	95.48
	✓		9	0	95.83
	✓	✓	9	98.11	95.48
20			20	0	95.83
	✓		10	0	96.18
	✓	✓	10	98.11	96.52
25			25	0	95.48
	✓		10	0	95.83
	✓	✓	10	98.11	95.48

accumulate information before making a final decision, which may improve accuracy.

To explore this trade-off, we evaluated different values of  $\lambda$  on the DVS128 Gesture and CIFAR10-DVS datasets, analyzing their impact on both decision timestep and accuracy. As illustrated in Figure 5, increasing  $\lambda$  generally decreases the decision timestep, reducing computational cost. However, beyond a certain threshold (e.g.,  $\lambda = 0.01$ ), the accuracy gain becomes marginal, indicating diminishing returns from allowing additional timesteps.

Based on these observations, we select  $\lambda = 0.01$  as the default threshold, as it provides a balance between maintaining high accuracy and reducing inference time. This choice ensures that the model reaches a stable classification decision while avoiding unnecessary computations. By tuning  $\lambda$ , different trade-offs between accuracy and efficiency can be achieved depending on application requirements.

## 5.2 Analysis of the results of adaptive temporal dynamics

From Table 3, it is evident that by applying Adaptive Temporal Dynamics, the obtained decision timestep on the DVS128 Gesture

and CIFAR10-DVS datasets is 10 and 8, respectively. This result enables us to achieve decision timesteps that are 50% and 80% of the original ones, effectively saving about 50% and 20% of computation on the temporal dimension. After spatial pruning, the resulting sparse network performance on both datasets is comparable to that of the unpruned network. This indicates that our spatiotemporal pruning scheme can achieve higher sparsity and maintain accuracy while reducing synaptic operations.

After training the model until convergence without any pruning, we employed the KL-divergence between the output layer’s response at different timesteps after feeding in the training set and the true labels as an intuitive metric. Subsequently, we computed the normalized KL-divergence for different timesteps through min-max normalization. To demonstrate the efficacy of this normalized KL-divergence in representing the information content across the temporal dimension, we calculated the information encapsulated by the model at different decision timesteps, which corresponds to the values of the empirical Fisher Information Matrix (FIM). We compared the trends of these two metrics, as shown in Figure 6, and found that their variations are almost identical. This indicates that the trend of changes in the model’s outputs aligns with the trend of changes in the model parameters during the learning process. This further suggests that

Adaptive Temporal Dynamics is meaningful for the selection of decision timestep.

### 5.3 Analysis of the total timestep

We also analyzed the time step used for the employed SNNs, setting the total timesteps to 10, 15, 20, and 25 respectively for parameter experiments on the DVS128 Gesture dataset. The performance of the baseline model, as well as the models employing Adaptive Temporal Dynamics and spatiotemporal pruning at different timesteps, is compared and presented in Table 5. As can be observed, our pruning method maintains stability under various time step conditions and often achieves an earlier decision timestep, thereby saving a relatively larger amount of time when more steps are used. In terms of temporal consumption, our method directly saved 20%, 40%, 50%, and 60% of the time when the timesteps were set to 10, 15, 20, and 25, respectively. This further illustrates that for the classification of DVS datasets, it is not necessary to complete the computation of all time dimensions. Correct classification results can be obtained through a portion of the input, which is also due to the information partition problem caused by the integration mechanism of DVS.

### 5.4 Statistical analysis of model parameter magnitudes

We further conducted a statistical analysis of the amplitude of model parameters. This analysis was performed on models using the DVS128 Gesture and CIFAR10-DVS datasets, focusing on both the unpruned baseline models and the sparse models after spatiotemporal pruning. The results, as illustrated in Figures 7, 8, indicate that our spatiotemporal pruning algorithm tends to remove weights that have a minimal impact on the output. Consequently, the overall amplitude of the weights in the sparse model is relatively larger compared to those in the unpruned baseline model.

### 5.5 Discussion of limitations

When developing pruning algorithms for SNNs, it is crucial to consider strategies that address temporal information more thoroughly than for ANNs. Our spatio-temporal pruning algorithm is particularly effective for datasets with rich temporal information, especially those captured by raw DVS cameras. The temporal information in SNNs depends on the accumulation of membrane potential and spike generation, which is less pronounced in static datasets compared to temporal datasets. Therefore, our strategy is more suited for temporal data, emphasizing an important characteristic where SNNs focus more on neuron dynamics compared to ANNs. Future work can explore hardware-aware pruning techniques to further minimize overhead. Additionally, while our method is designed for classification tasks, extending it to detection or generative tasks is worth exploring. These tasks involve

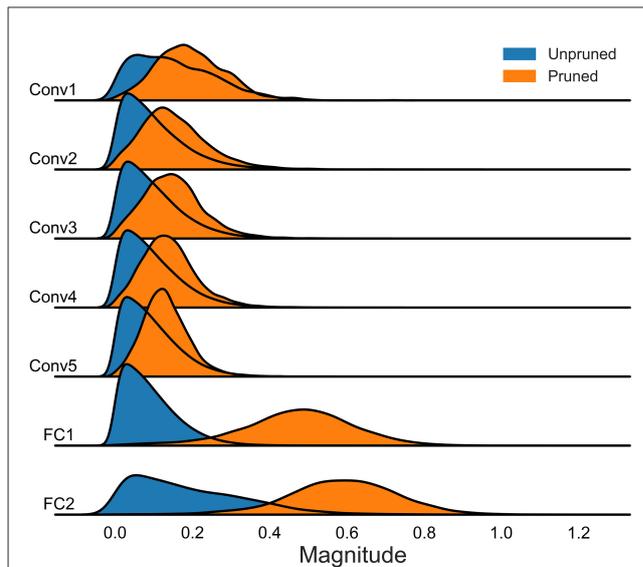


FIGURE 7  
The distribution of model parameter magnitudes without pruning and after spatiotemporal pruning on DVS128 gesture.

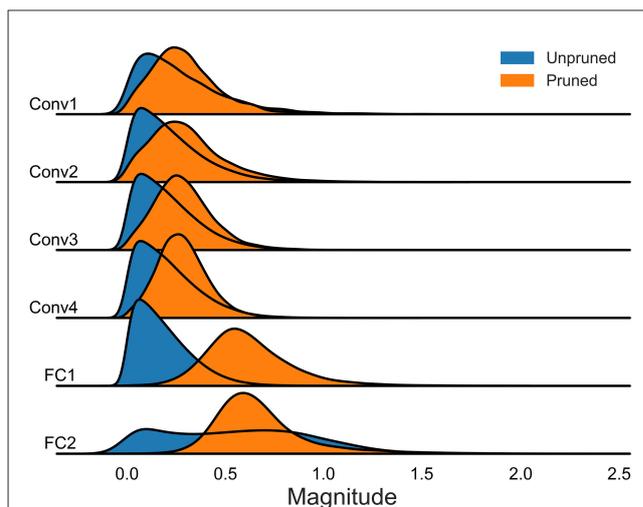


FIGURE 8  
The distribution of model parameter magnitudes without pruning and after spatiotemporal pruning on CIFAR10-DVS.

more complex temporal dependencies, which may require adapting our pruning strategy.

Beyond algorithmic considerations, our pruning approach could offer potential benefits for neuromorphic processors such as Speck (Richter et al., 2023) and Loihi (Davies et al., 2018). The Speck Development Kit, which utilizes an Address Event Representation protocol, naturally benefits from the unstructured sparsity generated by our method, reducing memory access without requiring additional hardware modifications. Additionally, the adaptive temporal pruning lowers the number of SOPs, making it highly efficient for low-power, always-on edge AI applications (Yao et al., 2024). Similarly, Loihi's event-driven architecture can

leverage the reduction in redundant spikes and synaptic activations, further optimizing computational efficiency.

## 6 Conclusion

Given the inherent spatio-temporal information processing capabilities of SNNs and the resource constraints of neuromorphic hardware, efficient pruning across both the spatial and temporal dimensions of SNNs is a critical issue. In this paper, we present a spatio-temporal pruning algorithm framework tailored for the information dimensions of SNNs. For the spatial dimension, we perform a global assessment of connections by combining inter-layer weight information of SNNs, ensuring that pruning of a particular layer is not excessive. In the temporal dimension, we choose the decision timestep by comparing the model's output with the labels. We propose a pruning scheme for the temporal dimension of SNNs and link it with the Fisher Information Matrix, allowing us to evaluate the temporal pruning scheme from both the perspectives of the network's output and the network parameters' information. Our method has achieved commendable performance on several datasets tested. Especially, we have demonstrated the potential for compression in the temporal characteristics of SNNs.

However, scalability to more complex datasets and larger SNN architectures remains an open challenge. A key direction for future work is to refine the quantification of both weight importance and timestep significance to ensure balanced sparsity across different network structures. Additionally, better integration of spatial and temporal pruning is necessary to prevent undesired trade-offs, such as excessive weight pruning leading to unstable temporal dynamics. Moreover, while our approach adopts a frame-based strategy, fully asynchronous event-triggered processing remains a challenge due to current hardware constraints and is an important direction for future research.

## Data availability statement

The employed datasets are publicly available online. Code is available at [github.com/gzxd/SNN\\_Pruning](https://github.com/gzxd/SNN_Pruning).

## Author contributions

SG: Writing – review & editing, Supervision. JF: Conceptualization, Investigation, Methodology, Project

administration, Validation, Writing – original draft, Writing – review & editing. YS: Conceptualization, Methodology, Writing – original draft. ZC: Funding acquisition, Project administration, Writing – review & editing. ZG: Conceptualization, Funding acquisition, Investigation, Project administration, Supervision, Writing – original draft, Writing – review & editing. JE: Writing – review & editing. RL: Funding acquisition, Project administration, Writing – original draft. LJ: Methodology, Supervision, Writing – review & editing.

## Funding

The author(s) declare that financial support was received for the research and/or publication of this article. This research was funded by the National Natural Science Foundation of China (Nos. 62301395, 62104176, and 52077160), the Shaanxi Province Postdoctoral Science Foundation (No. 2023BSHEDZZ177), the Xidian University Specially Funded Project for Interdisciplinary Exploration (No. TZJH2024026), and the Fundamental Research Funds for the Central Universities (No. XJSJ24071).

## Conflict of interest

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

## Generative AI statement

The author(s) declare that no Gen AI was used in the creation of this manuscript.

## Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

## References

- Amir, A., Taba, B., Berg, D., Melano, T., McKinstry, J., Di Nolfo, C., et al. (2017). "A low power, fully event-based gesture recognition system," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Honolulu, HI: IEEE), 7243–7252.
- Anumasa, S., Mukhoty, B., Bojkovic, V., De Masi, G., Xiong, H., and Gu, B. (2024). Enhancing training of spiking neural network with stochastic latency. *Proc. AAAI Conf. Artif. Intell.* 38, 10900–10908. doi: 10.1609/aaai.v38i10.28964
- Bellec, G., Kappel, D., Maass, W., and Legenstein, R. (2017). Deep rewiring: training very sparse deep networks. *arXiv [preprint]* arXiv:1711.05136. doi: 10.48550/arXiv.1711.05136
- Bohte, S. M., Kok, J. N., and La Poutre, H. (2002). Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing* 48, 17–37. doi: 10.1016/S0925-2312(01)00658-0

- Cao, D., Zeng, Y., Liu, M., He, X., Wang, M., and Qin, Z. (2020). "Strong: Spatio-temporal reinforcement learning for cross-modal video moment localization," in *Proceedings of the 28th ACM International Conference on Multimedia*, 4162–4170.
- Chauvin, Y. (1988). "A back-propagation algorithm with optimal use of hidden units," in *Advances in Neural Information Processing Systems*, 1.
- Chen, G., Peng, P., Li, G., and Tian, Y. (2023). Training full spike neural networks via auxiliary accumulation pathway. *arXiv [preprint]* arXiv:2301.11929. doi: 10.48550/arXiv.2301.11929
- Chen, Y., Ma, Z., Fang, W., Zheng, X., Yu, Z., and Tian, Y. (2023). A unified framework for soft threshold pruning. *arXiv [preprint]* arXiv:2302.13019. doi: 10.48550/arXiv.2302.13019
- Chen, Y., Yu, Z., Fang, W., Huang, T., and Tian, Y. (2021). Pruning of deep spiking neural networks through gradient rewiring. *arXiv [preprint]* arXiv:2105.04916. doi: 10.24963/ijcai.2021/236
- Chen, Y., Yu, Z., Fang, W., Ma, Z., Huang, T., and Tian, Y. (2022). "State transition of dendritic spines improves learning of sparse spiking neural networks," in *International Conference on Machine Learning* (New York: PMLR), 3701–3715.
- Chowdhury, S. S., Garg, I., and Roy, K. (2021). "Spatio-temporal pruning and quantization for low-latency spiking neural networks," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 1–9.
- Chowdhury, S. S., Rathi, N., and Roy, K. (2022). "Towards ultra low latency spiking neural networks for vision and sequential tasks using temporal pruning," in *European Conference on Computer Vision* (Cham: Springer), 709–726.
- Das, A., Sheffield, A. G., Nandy, A. S., and Jadhav, M. P. (2024). Brain-state mediated modulation of inter-laminar dependencies in visual cortex. *Nat. Commun.* 15:5105. doi: 10.1038/s41467-024-49144-w
- Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., et al. (2018). Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro* 38, 82–99. doi: 10.1109/MM.2018.112130359
- Delasalles, E., Ziat, A., Denoyer, L., and Gallinari, P. (2019). Spatio-temporal neural networks for space-time data modeling and relation discovery. *Knowl. Inf. Syst.* 61, 1241–1267. doi: 10.1007/s10115-018-1291-x
- Deng, L., Wang, G., Li, G., Li, S., Liang, L., Zhu, M., et al. (2020). Tianjic: a unified and scalable chip bridging spike-based and continuous neural computation. *IEEE J. Solid-State Circuits* 55, 2228–2246. doi: 10.1109/JSSC.2020.2970709
- Deng, L., Wu, Y., Hu, Y., Liang, L., Li, G., Hu, X., et al. (2021). Comprehensive snn compression using admm optimization and activity regularization. *IEEE Trans. Neural Netw. Learn. Syst.* 34, 2791–2805. doi: 10.1109/TNNLS.2021.3109064
- Deng, S., Li, Y., Zhang, S., and Gu, S. (2022). Temporal efficient training of spiking neural network via gradient re-weighting. *arXiv [preprint]* arXiv:2202.11946. doi: 10.48550/arXiv.2202.11946
- Eshraghian, J. K., Ward, M., Neftci, E., Wang, X., Lenz, G., Dwivedi, G., et al. (2021). Training spiking neural networks using lessons from deep learning. *arXiv [preprint]* arXiv:2109.12894.
- Fang, W., Chen, Y., Ding, J., Yu, Z., Masquelier, T., Chen, D., et al. (2023). Spikingjelly: an open-source machine learning infrastructure platform for spike-based intelligence. *Sci. Adv.* 9:eadi1480. doi: 10.1126/sciadv.adi1480
- Faust, T. E., Gunner, G., and Schafer, D. P. (2021). Mechanisms governing activity-dependent synaptic pruning in the developing mammalian CNS. *Nat. Rev. Neurosci.* 22, 657–673. doi: 10.1038/s41583-021-00507-y
- Fleming, L. L., and McDermott, T. J. (2024). Cognitive control and neural activity during human development: evidence for synaptic pruning. *J. Neurosci.* 44:26. doi: 10.1523/JNEUROSCI.0373-24.2024
- Frankle, J., and Carbin, M. (2018). The lottery ticket hypothesis: finding sparse, trainable neural networks. *arXiv [preprint]* arXiv:1803.03635. doi: 10.48550/arXiv.1803.03635
- Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. (2019). Stabilizing the lottery ticket hypothesis. *arXiv [preprint]* arXiv:1903.01611. doi: 10.48550/arXiv.1903.01611
- Gale, T., Elsen, E., and Hooker, S. (2019). The state of sparsity in deep neural networks. *arXiv [preprint]* arXiv:1902.09574. doi: 10.48550/arXiv.1902.09574
- Gao, H., Tian, Y., Xu, F., and Zhong, S. (2021). Survey of deep learning model compression and acceleration. *J. Softw.* 2021:32.
- Gonzalez, H. A., Huang, J., Kelber, F., Nazeer, K. K., Langer, T., Liu, C., et al. (2024). Spinnaker2: A large-scale neuromorphic system for event-based and asynchronous machine learning. *arXiv [preprint]* arXiv:2401.04491. doi: 10.48550/arXiv.2401.04491
- Guo, M., Zhao, K., Sun, J., Wen, S., and Dou, G. (2023). Implementing bionic associate memory based on spiking signal. *Inf. Sci.* 649:119613. doi: 10.1016/j.ins.2023.119613
- Han, B., Zhao, F., Pan, W., and Zeng, Y. (2025). Adaptive sparse structure development with pruning and regeneration for spiking neural networks. *Inf. Sci.* 689:121481. doi: 10.1016/j.ins.2024.121481
- Han, B., Zhao, F., Zeng, Y., and Shen, G. (2022). Developmental plasticity-inspired adaptive pruning for deep spiking and artificial neural networks. *arXiv [preprint]* arXiv:2211.12714. doi: 10.48550/arXiv.2211.12714
- Han, B., Zhao, F., Zeng, Y., and Shen, G. (2024). Developmental plasticity-inspired adaptive pruning for deep spiking and artificial neural networks. *IEEE Trans. Pattern Anal. Mach. Intellig.* 47, 240–251. doi: 10.1109/TPAMI.2024.3467268
- Han, S., Pool, J., Tran, J., and Dally, W. (2015). "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems*, 28.
- Hassibi, B., and Stork, D. (1992). "Second order derivatives for network pruning: optimal brain surgeon," in *Advances in Neural Information Processing Systems*, 5.
- Hu, Y., Zheng, Q., Jiang, X., and Pan, G. (2023). Fast-snn: Fast spiking neural network by converting quantized ann. *IEEE Trans. Pattern Anal. Mach. Intell.* 45, 14546–14562. doi: 10.1109/TPAMI.2023.3275769
- Hunsberger, E., and Eliasmith, C. (2015). Spiking deep networks with lif neurons. *arXiv [preprint]* arXiv:1510.08829. doi: 10.48550/arXiv.1510.08829
- Janowsky, S. A. (1989). Pruning versus clipping in neural networks. *Phys. Rev. A* 39:6600. doi: 10.1103/PhysRevA.39.6600
- Kim, Y., Li, Y., Park, H., Venkatesha, Y., Hambitzer, A., and Panda, P. (2023). Exploring temporal information dynamics in spiking neural networks. *Proc. AAAI Conf. Artif. Intellig.* 37, 8308–8316. doi: 10.1609/aaai.v37i7.26002
- Kim, Y., Li, Y., Park, H., Venkatesha, Y., Yin, R., and Panda, P. (2022). "Exploring lottery ticket hypothesis in spiking neural networks," in *European Conference on Computer Vision* (Cham: Springer), 102–120.
- Kunstner, F., Hennig, P., and Balles, L. (2019). "Limitations of the empirical fisher approximation for natural gradient descent," in *Advances in Neural Information Processing Systems*, 32.
- Lee, J., Park, S., Mo, S., Ahn, S., and Shin, J. (2020). Layer-adaptive sparsity for the magnitude-based pruning. *arXiv [preprint]* arXiv:2010.07611.
- Li, H., Liu, H., Ji, X., Li, G., and Shi, L. (2017). Cifar10-dvs: an event-stream dataset for object classification. *Front. Neurosci.* 11:244131. doi: 10.3389/fnins.2017.00309
- Li, Y., Guo, Y., Zhang, S., Deng, S., Hai, Y., and Gu, S. (2021). "Differentiable spike: rethinking gradient-descent for training spiking neural networks," in *Differentiable Spike: Rethinking Gradient-Descent for Training Spiking Neural Networks*, Vol. 34, eds. M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan (Curran Associates, Inc.), 23426–23439. Available online at: [https://proceedings.neurips.cc/paper\\_files/paper/2021/file/c4ca4238a0b923820d5c509a6f75849b-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2021/file/c4ca4238a0b923820d5c509a6f75849b-Paper.pdf)
- Li, Y., Kim, Y., Park, H., Geller, T., and Panda, P. (2022). "Neuromorphic data augmentation for training spiking neural networks," in *European Conference on Computer Vision* (Cham: Springer), 631–649.
- Li, Y., Xu, Q., Shen, J., Xu, H., Chen, L., and Pan, G. (2024). Towards efficient deep spiking neural networks construction with spiking activity based pruning. *arXiv [preprint]* arXiv:2406.01072. doi: 10.48550/arXiv.2406.01072
- Lin, C., Zhong, Z., Wei, W., and Yan, J. (2018). "Synaptic strength for convolutional neural network," in *Advances in Neural Information Processing Systems*, Vol. 31, eds. S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Curran Associates, Inc.). Available online at: [https://proceedings.neurips.cc/paper\\_files/paper/2018/file/4d19b37a2c399deace9082d464930022-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2018/file/4d19b37a2c399deace9082d464930022-Paper.pdf)
- Lin, X., Zou, X., Ji, Z., Huang, T., Wu, S., and Mi, Y. (2021). A brain-inspired computational model for spatio-temporal information processing. *Neural Netw.* 143, 74–87. doi: 10.1016/j.neunet.2021.05.015
- Liu, Y., Xu, Y., Ye, W., Cui, Y., Zhang, B., and Lin, W. (2024). High-speed, low-power, and configurable on-chip training acceleration platform for spiking neural networks. *Appl. Intellig.* 2024, 1–16. doi: 10.1007/s10489-024-05689-3
- Meng, L., Qiao, G., Zhang, X., Bai, J., Zuo, Y., Zhou, P., et al. (2023). An efficient pruning and fine-tuning method for deep spiking neural network. *Appl. Intellig.* 53, 28910–28923. doi: 10.1007/s10489-023-05056-8
- Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., et al. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* 345, 668–673. doi: 10.1126/science.1254642
- Millán, A. P., Torres, J. J., and Marro, J. (2019). How memory conforms to brain development. *Front. Comput. Neurosci.* 13:22. doi: 10.3389/fncom.2019.00022
- Modaresi, F., Guthaus, M., and Eshraghian, J. K. (2023). "Openspike: an openram snn accelerator," in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)* (IEEE), 1–5.
- Orchard, G., Frady, E. P., Rubin, D. B. D., Sanborn, S., Shrestha, S. B., Sommer, F. T., et al. (2021). "Efficient neuromorphic signal processing with Loihi 2," in *2021 IEEE Workshop on Signal Processing Systems (SIPS)* (Coimbra: IEEE), 254–259.
- Pei, J., Deng, L., Song, S., Zhao, M., Zhang, Y., Wu, S., et al. (2019). Towards artificial intelligence with hybrid tianjic chip architecture. *Nature* 572, 106–111. doi: 10.1038/s41586-019-1424-8

- Perez-Peña, F., Morgado-Estevez, A., Linares-Barranco, A., Jimenez-Fernandez, A., Gomez-Rodriguez, F., Jimenez-Moreno, G., et al. (2013). Neuro-inspired spike-based motion: From dynamic vision sensor to robot motor open-loop control through spike-vite. *Sensors* 13, 15805–15832. doi: 10.3390/s131115805
- Renda, A., Frankle, J., and Carbin, M. (2020). Comparing rewinding and fine-tuning in neural network pruning. *arXiv [preprint]* arXiv:2003.02389. doi: 10.48550/arXiv.2003.02389
- Richter, O., Xing, Y., De Marchi, M., Nielsen, C., Katsimpris, M., Cattaneo, R., et al. (2023). Speck: a smart event-based vision sensor with a low latency 327k neuron convolutional neuronal network processing pipeline. *arXiv [preprint]* arXiv:2304.06793. doi: 10.48550/arXiv.2304.06793
- Roy, K., Jaiswal, A., and Panda, P. (2019). Towards spike-based machine intelligence with neuromorphic computing. *Nature* 575, 607–617. doi: 10.1038/s41586-019-1677-2
- Rühling Cachay, S., Zhao, B., Joren, H., and Yu, R. (2023). “Diffusion: A dynamics-informed diffusion model for spatiotemporal forecasting,” in *Advances in Neural Information Processing Systems, Vol. 36*, eds. A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Curran Associates, Inc.), 45259–45287. Available online at: [https://proceedings.neurips.cc/paper\\_files/paper/2023/file/8df90a1440ce782d1f5607b7a38f2531-Paper-Conference.pdf](https://proceedings.neurips.cc/paper_files/paper/2023/file/8df90a1440ce782d1f5607b7a38f2531-Paper-Conference.pdf)
- Shi, X., Ding, J., Hao, Z., and Yu, Z. (2024a). “Towards energy efficient spiking neural networks: An unstructured pruning framework,” in *The Twelfth International Conference on Learning Representations*. Available online at: <https://openreview.net/forum?id=eoSeaK4QJo>
- Shi, X., Hao, Z., and Yu, Z. (2024b). “Spikingformer: bridging resnet and vision transformer in spiking neural networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (Seattle, WA: IEEE), 5610–5619.
- Shi, Y., Nguyen, L., Oh, S., Liu, X., and Kuzum, D. (2019). A soft-pruning method applied during training of spiking neural networks for in-memory computing applications. *Front. Neurosci.* 13:405. doi: 10.3389/fnins.2019.00405
- Singh, S. P., and Alistarh, D. (2020). “Woodfisher: Efficient second-order approximation for neural network compression,” in *Advances in Neural Information Processing Systems, Vol. 33*, eds. H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Curran Associates, Inc.), 18098–18109. Available online at: [https://proceedings.neurips.cc/paper\\_files/paper/2020/file/d1ff1ec86b62cd5f3903ff19c3a326b2-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/d1ff1ec86b62cd5f3903ff19c3a326b2-Paper.pdf)
- Tanaka, H., Kunin, D., Yamins, D. L., and Ganguli, S. (2020). “Pruning neural networks without any data by iteratively conserving synaptic flow,” in *Advances in Neural Information Processing Systems*, eds. H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Curran Associates, Inc.), 6377–6389. Available online at: [https://proceedings.neurips.cc/paper\\_files/paper/2020/file/46a4378f835dc8040c8057beb6a2da52-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/46a4378f835dc8040c8057beb6a2da52-Paper.pdf)
- Vogginger, B., Rostami, A., Jain, V., Arfa, S., Hantsch, A., Kappel, D., et al. (2024). Neuromorphic hardware for sustainable ai data centers. *arXiv [preprint]* arXiv:2402.02521. doi: 10.48550/arXiv.2402.02521
- Wu, Y., Deng, L., Li, G., Zhu, J., and Shi, L. (2018). Spatio-temporal backpropagation for training high-performance spiking neural networks. *Front. Neurosci.* 12:331. doi: 10.3389/fnins.2018.00331
- Xing, X., Gao, B., Zhang, Z., Clifton, D. A., Xiao, S., Du, L., et al. (2024). SpikeLLM: Scaling up spiking neural network to large language models via saliency-based spiking. *arXiv [preprint]* arXiv:2407.04752. doi: 10.48550/arXiv.2407.04752
- Yao, M., Richter, O., Zhao, G., Qiao, N., Xing, Y., Wang, D., et al. (2024). Spike-based dynamic computing with asynchronous sensing-computing neuromorphic chip. *Nat. Commun.* 15:4464. doi: 10.1038/s41467-024-47811-6
- Yin, R., Kim, Y., Li, Y., Moitra, A., Satpute, N., Hambitzer, A., et al. (2023). Workload-balanced pruning for sparse spiking neural networks. *arXiv [preprint]* arXiv:2302.06746. doi: 10.1109/TETCI.2024.3393367
- Zhou, C., Yu, L., Zhou, Z., Zhang, H., Ma, Z., Zhou, H., et al. (2023). Spikingformer: Spike-driven residual learning for transformer-based spiking neural network. *arXiv [preprint]* arXiv:2304.11954. doi: 10.48550/arXiv.2304.11954
- Zhou, H., Lan, J., Liu, R., and Yosinski, J. (2019). “Deconstructing lottery tickets: zeros, signs, and the supermask,” in *Advances in Neural Information Processing Systems, Vol. 32*, eds. H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc.). Available online at: [https://proceedings.neurips.cc/paper\\_files/paper/2019/file/1113d7a76ffceca1bb350bfe145467c6-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2019/file/1113d7a76ffceca1bb350bfe145467c6-Paper.pdf)
- Zhou, Y., and Zhang, A. (2021). Improved integrate-and-fire neuron models for inference acceleration of spiking neural networks. *Appl. Intellig.* 51, 2393–2405. doi: 10.1007/s10489-020-02017-3