Activity Pruning for Efficient Spiking Neural Networks

Tong Bu
Institution for Artificial Intelligence
School of Computer Science
Peking University
putong30@pku.edu.cn

Xinyu Shi
Institution for Artificial Intelligence
School of Computer Science
Peking University
xyshi@pku.edu.cn

Zhaofei Yu *
Institution for Artificial Intelligence
School of Computer Science
Peking University
yuzf12@pku.edu.cn

Abstract

While sparse coding plays an important role in promoting the efficiency of biological neural systems, it has not been fully utilized by artificial models as the activation sparsity is not well suited to the current structure of deep networks. Spiking Neural Networks (SNNs), with their event-driven characteristics, offer a more natural platform for leveraging activation sparsity. In this work, we specifically target the reduction of neuronal activity, which directly leads to lower computational cost and facilitates efficient SNN deployment on Neuromorphic hardware. We begin by analyzing the limitations of existing activity regularization methods and identifying critical challenges in training sparse SNNs. To address these issues, we propose a modified neuron model, AT-LIF, coupled with a threshold adaptation technique that stabilizes training and effectively suppresses spike activity. Through extensive experiments on multiple datasets, we demonstrate that our approach achieves significant reductions in average firing rates and synaptic operations without sacrificing much accuracy. Furthermore, we show that our method complements weight-based pruning techniques and successfully trains an SNN with only 0.06 average firing rate and 2.22M parameters on ImageNet, highlighting its potential for building highly efficient and scalable SNN models. Code is available at https://github.com/putshua/Activity-Pruning-SNN.

1 Introduction

The human brain is remarkably energy-efficient, yet capable of performing a wide range of complex tasks, such as reasoning and planning. One of the key factors behind this efficiency lies in the behavior of the neurons. As a general strategy of the neural representation in the neural system [Yoshida and Ohki, 2020], the flexibility of sparse coding enables neurons to operate in a highly energy-efficient manner. For example, despite the large population of neurons in specific areas of the brain tasked with particular functions, only a subset of the neuron populations will be engaged simultaneously. Experimental evidence shows that sparse representation of visual sensory information in the primary visual cortex enhances the selectivity and sparseness of individual neurons, thereby improving the computational efficiency of the visual system [Vinje and Gallant, 2000].

*Corresponding author

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

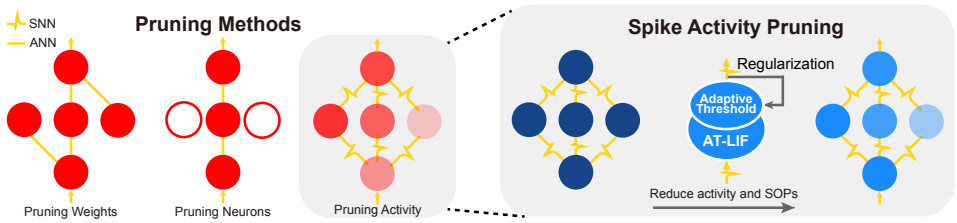

Figure 1: The left figure illustrates three pruning techniques for deep models. While all methods generally reduce model complexity, activity-based pruning is particularly effective for SNNs. The Right part shows the basic idea of our method, where the adaptive neurons (AT-LIF) are applied to regularize the threshold, so as to reduce the firing rate while stabilizing the training process.

A similar principle can be applied to artificial computational models through activation-based pruning, which seeks to moderate neuron activations to mirror the efficiency in sparse neural coding and gain efficiency in terms of time and energy consumption. However, activation pruning has not emerged as a mainstream approach for reducing network complexity compared to more impactful pruning methods that target network weights or structure [Kurtz et al., 2020], since the modest unstructured sparsity induced in activation maps seldom translates into meaningful efficiency gains with artificial neural networks on modern hardware. In contemporary architectures, the primary computational workload is carried out via matrix multiplications, where lowering a few activation values does not significantly cut down on computation time or resource usage.

Unlike traditional architectures, activity-based pruning offers a promising approach for brain-inspired models [Li et al., 2024], such as spiking neural networks (SNNs). SNNs are a well-established type of neural network that emulate the behavior of biological neurons and are increasingly recognized as a more efficient alternative to conventional computational models [Maass, 1997]. One important characteristic of the spiking neural networks is that it encodes information through discrete binary spikes. Therefore, the reduction of the neural activity in SNNs will directly lead to the sparsification of the output spike vector and the reduction of the firing rate, leading to potential computational efficiency. It is possible to create a highly sparse network in which only a few neurons will be engaged in one round of the computation with spike activity pruning, thus directly reducing the number of operations of the model. Thanks to the rapid recent advancements of neuromorphic hardware [Pei et al., 2019, DeBole et al., 2019, Davies et al., 2018, Nieves and Goodman, 2021, Fang et al., 2020, Zenke and Neftci, 2021, Yao et al., 2024], such sparse property can be effectively leveraged by specific neuromorphic hardware to work in an efficient and energy-saving manner, making these networks ideal for deployment in resource-constrained environments. Even on traditional architectures such as GPUs and deep learning accelerators, it is possible that such models can still benefit from activation sparsity to achieve memory savings and inference speedup. This approach enables the creation of models that are capable of performing complex tasks with minimal energy consumption, leading to more brain-like AI systems in terms of efficiency.

In this paper, we investigate an activity pruning technique aimed at suppressing the spike output. Our approach explores the capabilities of a novel neuron model in solving the learning dilemma in sparse models and stabilize the training process. We also introduce a threshold adaptation mechanism to effectively reduce neural activity. Our contributions can be summarized as:

- We specifically target the reduction of neuronal spike activity in SNNs, which directly lowers computational cost and serves as a unique advantage for efficient computation. To achieve this goal, we analyze the limitations of existing activity regularization methods and identify critical challenges associated with training sparse SNN models.
- We propose a novel neuron model, AT-LIF, designed to stabilize the training of sparse spiking networks while effectively suppressing spike activity. It incorporates a current-based output and an adaptive threshold mechanism, which help alleviate the gradient vanishing problem and resolve optimization conflicts.
- Extensive experiments demonstrate that our method significantly reduces spike activity and synaptic operations while maintaining performance comparable to the baseline methods. We further combine our proposed method with weight pruning strategies to illustrate its potential for achieving highly sparse SNNs.

2 Background

2.1 Related works

Network pruning is a well-established technique for artificial neural networks, with various approaches designed to reduce model complexity and improve efficiency. These pruning methods can typically be categorized into three types: unstructured weight pruning [Han et al., 2015, Zhu and Gupta, 2018, Kusupati et al., 2020], structured weight pruning [Wen et al., 2016, Mao et al., 2017, Anwar et al., 2017], and structured neuron pruning [Yu et al., 2018, Zhuang et al., 2020]. Similar exploration has been conducted with SNNs, where specific supervised learning algorithms are applied to prune weights or neurons [Wu et al., 2019, Yin et al., 2021]. Most distinguishing SNN pruning methods are inspired by biological principles, aiming to replicate the brain's efficient learning scheme to obtain sparse connected deep SNNs [Bellec et al., 2018, Chen et al., 2021, 2022, Shen et al., 2023]. The others draw inspiration from existing deep learning techniques and focus on jointly sparsifying the connections and neurons, introducing a complete pruning framework on deep SNNs [Deng et al., 2021, Shi et al., 2024]. Notably, while various pruning methods have been proposed for spiking neural networks Yin et al. [2021], Wu et al. [2019], Zhou et al. [2024], most focus on weight or neuron pruning rather than developing activity-based pruning approaches in depth. The majority of the existing pruning methods rely on adding regularization terms to constrain the firing rate of individual neurons [Deng et al., 2021, Yan et al., 2022], or regularizing the activation value in artificial neural networks before converting it into SNN [Neil et al., 2016]. In fact, direct regularization of spike activity actually conflicts with the optimization goal, leading to a trade-off in performance and sparsity, as these methods are either limited to shallow networks or fail to effectively train highly sparse models [Narduzzi et al., 2022, Sakemi et al., 2023].

While pruning techniques are essential for training efficient SNNs, they cannot be separated from the underlying learning process. Specialized optimization techniques are required to ensure both efficient learning of the target function and eliminating as much redundant structure as possible. Most researches that focus on the pruning of SNNs adopts the supervise-learning-based approaches [Bohte et al., 2000], or more specifically, the RNN-like backpropagation-through-time (BPTT) algorithm [Wu et al., 2018]. In this method, the surrogate gradient approximation is proposed to smooth the non-differentiable firing function in the neuron [Lee et al., 2016, Shrestha and Orchard, 2018, Fang et al., 2021a, Neftci et al., 2019, Zenke and Vogels, 2021, Stewart and Neftci, 2022]. Some researches also adopted the event-driven backpropagation that is able to maintain the sparsity of the gradient [Fang et al., 2021b, Kim and Panda, 2020, Zheng et al., 2021, Lee et al., 2020, Deng et al., 2022, Mostafa, 2017, Bohte et al., 2000, Zhu et al., 2022, Zhang et al., 2021]. Another commonly used SNN training methods are ANN-SNN conversion approaches [Cao et al., 2015], which map the weights of a pre-trained ANN into an rate-coding SNN [Diehl et al., 2015, Sengupta et al., 2019, Han et al., 2020, Deng and Gu, 2021, Ding et al., 2021, Bu et al., 2022, Meng et al., 2022]. ANN-SNN conversion is the most practical training method to train SNNs on large-scale datasets since converted SNNs always have outstanding performance [Kim et al., 2020, Hao et al., 2023b,a].

2.2 Neuron model

In this paper, we use the widely adopted Leaky-Integrate-and-Fire model for SNNs [Izhikevich, 2004, Gerstner et al., 2014]. The discretized dynamics of membrane potential can be summarized by

$$x_i[t] = w_{ij} s_j[t] + b_i, \tag{1}$$
$$m_i[t] = \tau u_i[t-1] + x_i[t], \tag{2}$$
$$s_i[t] = H(m_i[t] - \theta_i), \tag{3}$$
$$u_i[t] = (1 - H(m_i[t] - \theta_i)) \cdot m_i[t], \tag{4}$$

where j and i indicate the two adjacent neurons and w_{ij} and b_i represent the weight and bias of the connection from neuron j to neuron i. At arbitrary time-step t, the current input of neuron i is $x_i[t]$ (Eq. 1). $m_i[t]$ and $u_i[t]$ represent the value of the membrane potential before and after neuronal firing. $H(\cdot)$ is the Heaviside step function. The neuron will emit a spike, denoting by the 0-1 scaler $s_i[t]$ when the membrane potential $m_i[t]$ reaches the firing threshold θ_i. τ is the leaky parameter and if we explicitly set τ to 1, the neuron will degenerate to the non-leaky IF model. We demonstrate the hard-reset function in Eq. 4 that reduce the neuron membrane potential to the resting potential 0.

2.3 Spike activity pruning in SNNs

A general method for learning sparse activated models with activation regularization involves modifying the loss function with a penalty term that promotes sparsity in the network's activations. There are various techniques to implement such regularization over neuron activation, including applying L1/L2 regularization over the average spike rate per neuron [Neil et al., 2016, Yan et al., 2022, Deng et al., 2021], or using other regularizers [Narduzzi et al., 2022]. With the constraint of minimizing the spike activity, the general loss function of such activity-based pruning methods consists of both learning objectives for the specific task and the regularization term over the spike activity, which is

$$\mathcal{L}_{total} = \frac{1}{M}\sum_{k=1}^{M}\left(\mathcal{L}_{task}\left(\boldsymbol{f}(\boldsymbol{x}_k), \boldsymbol{y}_k\right) + \lambda_s \sum_{i=1}^{N} \left\|\boldsymbol{s}_i(\boldsymbol{x}_k)\right\|_p\right). \tag{5}$$

We use $\mathcal{L}_{task}$ (term in the left) and $\mathcal{L}_{reg}$ (term in the right) denotes the task-specific loss and activation regularization loss, respectively. M represents the number of training samples and N are total neuron numbers. $\boldsymbol{x}_k, \boldsymbol{y}_k$ denotes the k-th data-label sample from the training dataset. $\boldsymbol{f}(\cdot)$ is the function that represents the input-output mapping of the model. The regularization term is $\lambda_s \sum_{i=1}^{N} \left\|\boldsymbol{s}_i(\boldsymbol{x}_k)\right\|_p$ is the summation of the p-norm of the current output spike vector with the penalty coefficient parameter λ_s, where vector $\boldsymbol{s}_i(\boldsymbol{x}_k)$ represents the output spike pattern of neuron i across all time-steps when the model receives $\boldsymbol{x}_k$ as input. Many existing activity-based pruning methods for spiking neural networks can be considered as specific instances of this general framework, as they apply similar principles to add regularization based on their activity levels. However, using only the activation regularization technique cannot achieve a highly sparse model. While activation regularization can directly promote sparsity by adding constraints on parameters, this approach often leads to a trade-off between efficiency and accuracy. As the penalty term increases or the model becomes sparser, performance degradation becomes more severe, limiting its practical effectiveness.

3 Method

3.1 Learning dilemma of better performance and sparser spikes

The trade-off between the model performance and activation sparsity in artificial neural networks has been widely recognized as a challenge [Li et al., 2023, Georgiadis, 2019, Kurtz et al., 2020]. In SNNs, this issue is even more severe due to the sparser activations and inaccurate gradient estimation when using a surrogate gradient. Here we summarize two key challenges that significantly hinder the learning process in the joint optimization of task-specific loss and regularization loss. These two issues impede the effective optimization, limiting both the efficiency and performance of the model.

Conflicts on optimization target The first issue that hinders the learning process is the dilemma between learning more sparse spike activations and learning non-zero representations from the input data, as the optimization objectives of $\mathcal{L}_{task}$ and $\mathcal{L}_{reg}$ are inherently contradictory. Let us assume that in the i-th neuron, when the model receives the k-th data sample $\boldsymbol{x}_k$ from the training dataset, $\hat{\boldsymbol{y}}_i$ is the optimal output feature representation for neuron i that we attempt to learn. The regularization over the spike patterns are L2 norm. Under such assumption, the learning objective can be rewritten as

$$\mathcal{L}_{total} = \frac{1}{M}\sum_{k=1}^{M}\left(\sum_{i=1}^{N}\|\boldsymbol{s}_i(\boldsymbol{x}_k) - \hat{\boldsymbol{y}}_i\|_2^2 + \lambda_s \sum_{i=1}^{N}\|\boldsymbol{s}_i(\boldsymbol{x}_k)\|_2^2\right) \tag{6}$$

$$= \frac{1}{M}\sum_{k=1}^{M}\sum_{i=1}^{N}\left(\|\boldsymbol{s}_i(\boldsymbol{x}_k) - \hat{\boldsymbol{y}}_i\|_2^2 + \lambda_s\|\boldsymbol{s}_i(\boldsymbol{x}_k) - \boldsymbol{0}\|_2^2\right). \tag{7}$$

The Eq. 7 illustrates a fundamental conflict between the target loss and the regularization term in the learning process. One the one hand, minimizing the target loss typically encourages the model to capture intricate patterns in the data and learning non-zero feature representations. The spike-activity regularization, on the other hand, encourages sparser spike patterns and less spike output. Therefore, there is an obvious trade-off between effectively learning an inner representation from input data and reducing spike activity at each layer. The reduction of the spike disrupts the representation learning at each layer since the regularization term inhibits spike firing. The trade-off between the

two objectives makes it challenging to simultaneously optimize for both effective representation learning and minimal spike activity.

Gradient vanishing problem with sparse activity The other issue that impedes the learning process of sparse SNNs comes from the learning process when using gradient-based learning. It has been observed that the gradient vanishing problem may significantly degrade the performance of the SNN when using Backpropagation-Through-Time (BPTT) [Fang et al., 2021a]. Furthermore, this issue becomes even more severe and prevalent when activity regularization is applied, as the sparsity of spike activity will further induce gradient vanishing and prevent the model from convergence. To illustrate this issue, we first provide a theoretical analysis showing that excessive regularization of the average firing rate can cause the training process to be trapped at a saddle point.

Theorem 1. *We define* $\boldsymbol{S} = \{\boldsymbol{w} | \forall l, t \;\; \boldsymbol{s}^l[t] = \boldsymbol{0}\}$ *as the set of all parameter sets such that the output spike at arbitrary layers and time-steps is all zero, then the* $\boldsymbol{S}$ *is a set of saddle points for the optimization of the total loss* $\mathcal{L}_{total}$*, which is*

$$\forall i \;\; \sum_t \boldsymbol{s}_i[t] \to 0 \;\; \Rightarrow \;\; |\frac{\partial \mathcal{L}_{total}}{\partial w_{ij}}| \to 0, \tag{8}$$

$$\frac{\partial \mathcal{L}_{total}}{\partial w_{ij}} = 0, \;\; \forall w_{ij} \in \boldsymbol{S} \tag{9}$$

$\boldsymbol{S}$ denotes the set of all parameter sets such that the output spike at an arbitrary layer and time is all zero. As described in the Eq. 9, we can prove that, in practice, the $\boldsymbol{S}$ is a set of critical points for the optimization of the total loss $\mathcal{L}_{total}$, since the gradient of the weight vectors with respect to the total loss $\nabla_{\boldsymbol{w}} \mathcal{L}_{total}$ is all zero and normally it is not the global minima of the total loss. Therefore, any $\boldsymbol{w} \in \boldsymbol{S}$ is a saddle point of the optimization problem, and the gradient will converge to zero when the spike output is zero. Detailed proof will be provided in the supplementary material.

This theorem identifies a set of saddle points in the loss function, highlighting a critical issue in the optimization process. When regularizing spike activity, the model might easily fall into the local minima with extremely sparse activity. In this case, the learning process will stop since the gradient comes to zero, and the optimization process will be trapped at the saddle point. This phenomenon emphasizes the need for carefully designed regularization strategies to avoid gradient vanishing and maintain effective learning dynamics.

3.2 Adaptive threshold LIF neuron

In the previous section, we identify two key issues that hinder the training of the sparse activated spiking neural networks from two perspectives: the conflict in optimization target and the potential gradient vanishing problem during optimization. Therefore, in this section, to alleviate the impact of the learning dilemma, we provide a feasible solution by introducing the Adaptive Threshold Neuron (AT-LIF), which includes: 1) introducing the adaptation mechanism to the neuron threshold; 2) using current-based output with 0-θ value instead of 0-1 spike output.

With the adaptation mechanism, the threshold value θ_i evolves throughout training and adapt to the spike frequency at each iteration. The threshold thus becomes a learnable parameter during the training process and plays a role of both controlling the firing frequency of the neurons and the efficacy of transmitting outputs to the next layer:

$$o_i[t] = \theta_i \cdot H(m_i[t] - \theta_i) = \theta_i s_i[t], \tag{10}$$

$$x_i[t] = w_{ij} o_j[t] + b_i. \tag{11}$$

We use $o_i[t] \in \{0, \theta_i\}$ to represent the output of the AT-LIF neuron instead of the $s_i[t]$ in the common LIF neuron (Eq. 1, 3), which scales up the neuron output while the average activity is low. Note that this mechanism does not imply graded output spikes, as the scaling by the threshold value does not alter the binary nature of the spike outputs. During inference, when the thresholds are fixed, the scaling can be absorbed into the subsequent layer's weights, ensuring full compatibility with all types of neuromorphic hardware. Intuitively, increasing the threshold reduces the firing frequency while preserving the average output value. Before introducing the detailed adaptive algorithm, we first demonstrate how threshold adaptation gradually reduces the neuron activity (Fig. 2). In this toy example, we continually add constant input to the AT-LIF neuron and vanilla LIF neuron and compare

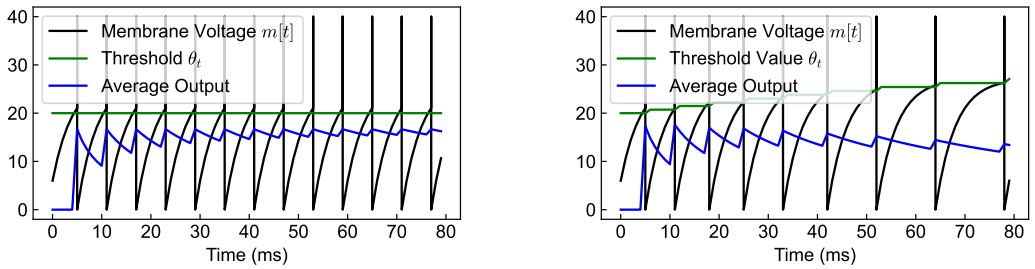

Figure 2: Left: Behavior of the vanilla LIF neuron; Right: Behavior of the AT-LIF neuron. In both figures, the black curve represents the change of the membrane potential, while the green curve represents the threshold value and the blue curve represents the average neuronal output over time.

both the output and average firing rate of these two different types of neurons. In the AT-LIF model, the threshold will increase slightly after each output spike, while the threshold of the vanilla LIF will remain constant. Such adaptation of the threshold helps reduce the firing rate while preventing a drop of the average output of the neuron. These results demonstrate the potential of the AT-LIF model to suppress output spiking activity while preserving stable output dynamics and faithful neuronal representation of input information.

3.3 Threshold adaptation for sparse activity

In practice, we incorporate the threshold adaptation mechanism into the Backpropagation-Through-Time (BPTT) algorithm [Wu et al., 2018] by treating the threshold at each layer as a learnable parameter during training. In order to constrain the spike activity, we regularize the threshold value to minimize the neuron firing rate. To constrain spike activity, we regularize the threshold values by minimizing the firing rate at each layer. Specifically, the thresholds are optimized to minimize both the task-specific loss and the activity regularization loss ($\arg\min_{\boldsymbol{\theta}} \mathcal{L}_{total}$), while the synaptic weights are updated solely to minimize the task-specific target loss. Based on this formulation, we derive the threshold update rule at each update iteration in Eq. 12.

$$\Delta\theta_i = \eta \left(\frac{\partial \mathcal{L}_{task}}{\partial \theta_i} + \lambda_\theta \sum_{t=1}^{T} s_i[t] \frac{\partial o_i[t]}{\partial \theta_i} \right), \tag{12}$$

where T is the inference time-steps for the SNN model and η is the learning rate during the training process. $\frac{\partial s_i[t]}{\partial \theta_i}$ can be estimated by the surrogate function. To distinguish from the regularization-based method over average spike activity, we use λ_θ to represent the coefficient parameter of the threshold adaptation. In practice, we consider each θ_i as a learnable parameter that is involved in the training process, while also manually regularize its value at each iteration according to Eq. 12. We also demonstrate the forward/backward information flow for AT-LIF neuron in right part of the Fig 3. The full derivation and pseudo-code of the overall algorithm are provided in the supplementary material.

As for the surrogate function, we consider the triangle-shaped function as surrogate of the Heaviside function [Esser et al., 2016] to estimate the derivative of the firing function of the AT-LIF neuron (Eq. 13). To prevent gradient explosion, the upper bound of the surrogate function is set to 1 while the design principle of the triangle-shaped surrogate function is strictly followed. Leveraging this surrogate, along with the proposed learning framework and adaptation rules for AT-LIF, we are able to train high-performance spiking neural networks with sparser activation.

$$\frac{\partial o_i[t]}{\partial m_i[t]} = -\frac{\partial o_i[t]}{\partial \theta_i} = \max\left(1 - \frac{|m_i[t] - \theta_i|}{\theta_i}, 0\right). \tag{13}$$

3.4 Stabilizing training with AT-LIF neuron

In this section, we demonstrate how the AT-LIF neuron addresses two key challenges that limit the performance of sparse SNN models. Specifically, the AT-LIF neuron resolves the inherent conflict

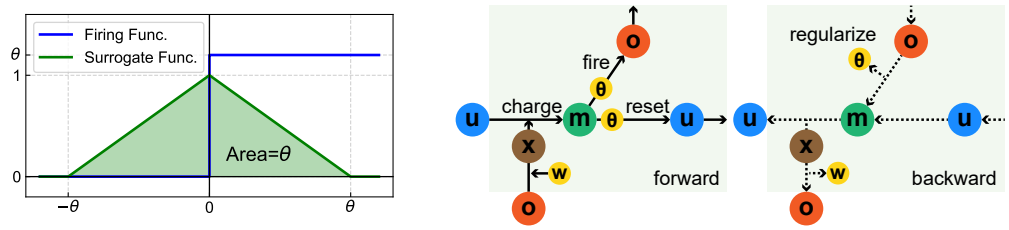

Figure 3: Left: The surrogate function for AT-LIF neuron. The blue curve represents the firing function while the green curve represents the surrogate derivative. Right: Illustration of the forward pass and backward pass of the AT-LIF neuron using the Back-Propagation-Through-Time algorithm.

between competing learning objectives and mitigates the gradient vanishing problem encountered when training highly sparse networks.

Solving the conflict of optimization target

As discussed in the above section, training sparse spiking neural networks typically involves jointly optimizing the task objective and the sparsity constraint as $\boldsymbol{\theta}, \boldsymbol{w} = \arg\min_{\boldsymbol{\theta},\boldsymbol{w}} \mathcal{L}_{total}$, which leads to substantial conflict on optimization target. In our method, however, we introduce the AT-LIF neuron, which outputs a scaled binary signal rather than a standard binary spike.

This scaling mechanism helps maintain a steady average neuron output while simultaneously reducing the firing rate (as illustrated in Fig 2). This property allows us to decouple the learning process while also maintaining stable learning process. Specifically, instead of jointly optimize the $\mathcal{L}_{total}$, we separate the optimization by learning the weight parameters based on the task loss as $\boldsymbol{w} = \arg\min_{\boldsymbol{w}} \mathcal{L}_{task}$ while simultaneously regularize the threshold to reduce spike activity as $\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_{total}$.

The conflict does not necessarily arise for learning of weight parameters as the model can still effectively learn from the output representation while we simultaneously maximize the threshold to promote sparser firing events. Consequently, the inherent conflict between the optimization objectives can, in principle, be resolved when adaptive threshold neurons are employed. As a result, the network can effective learning representations while reducing spike counts during the optimization process.

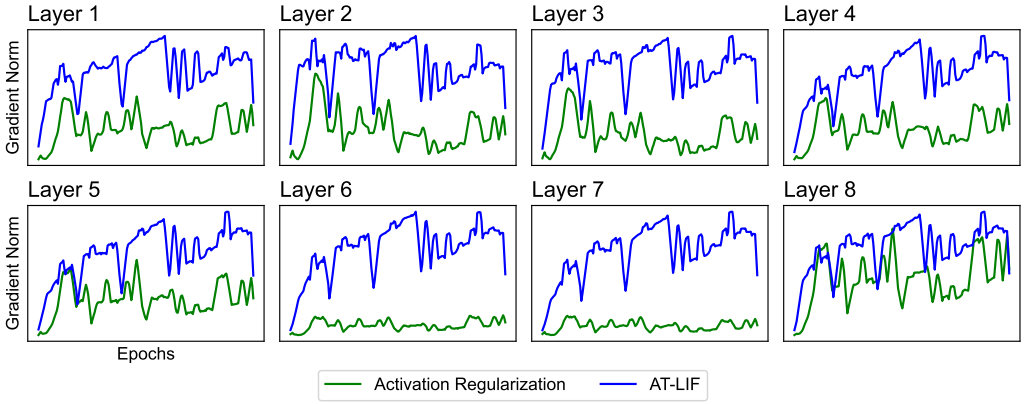

Figure 4: The 2-norm of the gradient value in the first 8 Convolutional layer.

Alleviating the gradient vanishing problem

On the other hand, the gradient vanishing problem will be alleviated since the gradient will no longer necessarily approach zero as the threshold value increases. One possible theoretical explanation is that the saddle point assumption (Theorem 1) no longer holds true as we continually increase the threshold rather than directly constrain the spike activity. We further empirically demonstrate this phenomenon by comparing the change of the gradient value with respect to the training iterations in different activity pruning methods before and after the adoption of the AT-LIF neuron. We trained multiple SNN models using AT-LIF neuron with threshold adaptation and vanilla LIF neuron with activation regularization on CIFAR-10 dataset with ResNet-20 structure and selected two models

with similar average firing rates after training for fair comparison. Here the model using AT-LIF achieved 91.06% accuracy and 0.043 average firing rate, while the model using vanilla LIF only achieved 88.31% accuracy and a larger average firing rate of 0.044. We demonstrate the comparison of the gradient values of the two models at each training iteration (Fig. 4). The green curve represents the normalized gradient 2-norm value of the weight vector in the corresponding convolution layer in the model with vanilla LIF and activation regularization, while the blue curve represents that of the model adopting AT-LIF neuron. Here the blue curve is always above the green curve and in most layers, the gradient value of the model using activation regularization is much closer to zero while the gradient value in the model with AT-LIF neuron still maintains consistency across layers. This indicates that using AT-LIF can effectively mitigate the gradient vanishing problem. Moreover, this is also confirmed by the fact that the final training results of the AT-LIF model achieve better performance while achieving similar sparsity.

4 Experiments

4.1 General experimental setting

We provide the general hyper-parameters and the neuron parameter setting in here and the supplementary material for better reproducibility. Following the previous study [Chen et al., 2022, Shi et al., 2024], we select a similar architecture for evaluation, including 6 Conv, 2FC (CIFAR-10), SEW-ResNet18(ImageNet, ImageNet-100), and VGGSNN(DVS-CIFAR10). We also utilize the standard ResNet-20 for the CIFAR dataset [He et al., 2016].

To demonstrate the effectiveness of the pruning method, we will use the following metrics in the below table, including classification accuracy (Acc.), average firing rate (Avg. FR), total parameters (Params) and total synaptic operation numbers (SOPs). The definition for the average firing rate $\boldsymbol{r} = \sum_{k=1}^{M} \sum_{i=1}^{N} \sum_{t=1}^{T} s_i(\boldsymbol{x}_k)[t]$, where x_k are data samples from the test set and $s_i(\boldsymbol{x}_k)[t]$ is the spike value in each neuron at each time-step with given input $\boldsymbol{x}_k$. The synaptic operation (SOP) quantifies the total number of computational events in spiking neural networks, corresponding to the number of times a spike is transmitted across a synapse [Hu et al., 2018, Fang et al., 2021a]. Additionally, we report the number of parameters left after applying weight-based pruning.

4.2 Compare with existing methods

Dataset	Arch.	Method	T	λ_s	λ_θ	Acc.(%)	Avg FR.	SOPs(M)
CIFAR-10	ResNet-20	Vanilla	8	0	0	92.89	0.236	256.69
	ResNet-20	AR	8	0.01	0	91.31	0.060	57.26
	ResNet-20	AR	8	0.02	0	88.31	0.044	41.86
	ResNet-20	AT-LIF	8	0	1e-3	92.05	0.053	61.70
	ResNet-20	AT-LIF	8	0	2e-3	91.06	0.043	48.85
	ResNet-20	AT-LIF	8	0	5e-3	88.31	0.033	39.33
	ResNet-20	AT-LIF	8	0.01	1e-3	90.70	0.025	29.23
ImageNet-100	SEW-ResNet18	Vanilla	4	0	0	83.44	0.138	1276.42
	SEW-ResNet18	AR	4	0.01	0	82.94	0.083	840.90
	SEW-ResNet18	AR	4	0.05	0	82.26	0.043	469.00
	SEW-ResNet18	AT-LIF	4	0.01	1e-4	82.40	0.038	463.65
	SEW-ResNet18	AT-LIF	4	0.02	2e-4	81.12	0.025	328.34
DVS-CIFAR10	VGGSNN	Vanilla	10	0	0	82.70	0.080	1066.40
	VGGSNN	AR	10	0.005	0	81.30	0.055	673.06
	VGGSNN	AT-LIF	10	0	5e-4	82.20	0.043	626.57
	VGGSNN	AT-LIF	10	0.001	5e-4	82.10	0.039	587.68

Table 1: Comparison of AT-LIF and direct activation-regularization-based methods on CIFAR-10, ImageNet-100, and DVS-CIFAR10 datasets.

In this section, we compare our proposed model with other existing SNN pruning approaches to evaluate its performance and highlight its advantages. Here we will present two tables for fair

comparison. The first table compares our pruning method with existing activity regularization approaches, all of which focus only on reducing spike activity or average firing rate. The second table reports results under the setting of unconstrained structural pruning, where activity-based pruning is combined with connection-level or neuron-level pruning methods. Specifically, we integrate our method with the weight pruning approach STDS [Chen et al., 2022] to demonstrate its potential to learn highly efficient SNNs.

Spike activity pruning

Since existing activity regularization methods primarily focus on penalizing spike activity, we adopt L1 spike regularization as a representative regularization-based approach for comparison. Tab. 1 presents a comprehensive evaluation of our proposed AT-LIF model (AT-LIF) against two baselines: the vanilla model (Vanilla) and vanilla model trained with spike activity regularization (AR), across three benchmark datasets: CIFAR-10, ImageNet-100, and DVS-CIFAR10. The columns "λ_s" and "λ_θ" are the regularization coefficients for the spike activity term (Eq. 7) and the coefficient for the threshold adaptation mechanism, respectively. The column T refers to the simulation timestep for SNN.

Across all tasks, the AT-LIF neuron demonstrates a strong capability to reduce spike activity and computational cost while maintaining high classification accuracy compared with regularization-based methods. On CIFAR-10, AT-LIF achieves a similar accuracy to the unpruned Vanilla model, while reducing the average firing rate by approximately 89.4% and lowering SOPs by nearly 88.6%. Compared to activation regularization, AT-LIF achieves lower SOPs and firing rates while maintaining higher accuracy. Similar trends are observed on the ImageNet-100 dataset, where AT-LIF maintains competitive accuracy with 1% accuracy loss while significantly reducing SOPs and average firing rate. If sacrificing 2.3% more accuracy is acceptable, the reduction of average firing rate on the AT-LIF model will be more than 5.5 times. On the event-based DVS-CIFAR10 dataset, AT-LIF again achieves efficient spike suppression with substantial reductions in SOPs (up to 45% fewer than Vanilla) and firing rates, while preserving accuracy. These results collectively highlight the effectiveness of AT-LIF in achieving sparse, energy-efficient SNNs without sacrificing much performance, outperforming conventional activity regularization techniques.

Dataset	Arch.	Pruning	Method	Acc.(%)	Avg FR.	Param(M)	SOPs(M)
CIFAR-10	6 Conv, 2 FC	Conn	STDS	90.71	0.072	0.12	45.48
				89.40	0.059	0.07	28.38
	6 Conv, 2 FC	Neu;Conn	Unstru.	91.74	0.051	9.48	15.15
				90.76	0.049	6.93	8.49
	6 Conv, 2 FC	Act;Conn	AT-LIF	90.72	0.024	0.19	30.11
				90.20	0.020	0.09	16.88
ImageNet	SEW-ResNet18	Conn	STDS	58.90	0.189	1.84	461.26
	SEW-ResNet18	Neu;Conn	Unstru.	59.23	0.106	4.38	271.98
	SEW-ResNet18	Act;Conn	AT-LIF	59.61	0.063	2.22	310.53

Table 2: Comparison of the composite AT-LIF method with existing SNN pruning frameworks.

Pruning both activity and weights

We also integrate our method with the STDS [Chen et al., 2022] to achieve the pruning of both weights and neuron activity. STDS operates by reparameterizing the weights and progressively eliminating those with small absolute values during training, leading to a sparse connectivity pattern. Our integration is straightforward: we replace the standard LIF neurons with AT-LIF neurons throughout the network, and apply threshold adaptation as described in our method. Meanwhile, the STDS pruning algorithm is used to train the synaptic weights and induce weight sparsity. This combination allows us to simultaneously achieve both activity sparsity and weight sparsity. We compare the composite approach with reproduced implementations of existing SNN pruning frameworks, including unstructured weight pruning (STDS) and unstructured weight and neuron pruning (Unstru.) [Shi et al., 2024]. We compare the above methods in accuracy, average firing rate, parameter numbers and total SOPs. The final results are demonstrated in Tab. 2.

On CIFAR-10, the composite AT-LIF method achieves comparable accuracy to both STDS and Unstru., while significantly reducing the average firing rate. Notably, AT-LIF achieves the lowest average firing rate at 0.020 and reduces SOPs to 16.88M, representing a substantial gain in computational

efficiency. On ImageNet, AT-LIF again achieves the lowest firing rate (0.063) and a favorable balance between accuracy and efficiency, with SOPs reduced compared to STDS (461.26M) and close to those achieved by the unstructured method (271.98M). Overall, the results highlight the strength of AT-LIF in achieving efficient sparse SNNs with low energy cost, while maintaining performance across datasets and architectures. Compared with the Unstru. that aims to reduce SOPs and STDS that aims to reduce parameters, although our method is not optimal in every individual metric, it strikes an effective balance between spike sparsity and weight sparsity, achieving the best performance across these two key indicators. This highlights the flexibility and effectiveness of the proposed AT-LIF model in enabling both activity and weight sparsity when combined with weight pruning techniques.

4.3 Ablation study

An ablation study was conducted to evaluate the impact of specific components of the AT-LIF neuron model on the overall performance of the SNNs. We systematically remove or alter the two components as mentioned in Sec. 3.2 and therefore evaluate their individual contributions. The Tab. 3 demonstrates

CurO.	Adap.	λ_θ	Acc.(%)	Avg. FR	SOPs(M)
×	×	0.0	92.89	0.236	256.69
✓	×	0.0	93.32	0.237	258.95
×	✓	1e-3	50.99	0.110	126.13
✓	✓	1e-3	92.05	0.054	61.70

Table 3: Ablation study for AT-LIF neuron on CIFAR-10 dataset.

the model performance and efficiency under different neuron configurations. Here column "CurO." represents whether the neuron has 0-1 output or 0-θ output while column "Adap." indicates whether the threshold adaptation mechanism is enabled. The model without the current-based output feature fails to converge when combined with the adaptation mechanism, indicating that the resulting sparse spike output hinders effective model learning. Models that applied the adaptive threshold exhibit a significant efficiency advantage over the one with the vanilla LIF neuron, with a notable reduction in overall computational cost. The model with AT-LIF neuron achieves the most outstanding results across all experiments as it is four times more efficient compared with the baseline model in terms of spike sparsity and energy cost. Results show that both current-based output and threshold adaptation are crucial for maintaining an optimal trade-off between minimizing excessive spike activity and preserving the model's ability to learn effectively.

5 Conclusion and limitation

In this paper, we propose a novel technique to significantly reduce spike activity in SNNs while preserving their performance. By effectively suppressing spike events, our method substantially promotes the sparsity of spike outputs and decreases required synaptic operations. As the proposed method focuses on reducing spike activity, it is also compatible with weight pruning or other model compression techniques. The proposed method potentially enables efficient mapping onto neuromorphic hardware or even on modern GPU architectures.
Limitation While this work introduces the concept of exploiting the sparse spike activity of SNNs, it still remains at the algorithmic level. The next step may involve exploring practical hardware implementations and evaluating the effectiveness of the approach in real-world applications.

6 Acknowledgement

This work is funded by National Natural Science Foundation of China (62422601, U24B20140, and 62088102), Beijing Municipal Science and Technology Program (Z241100004224004) and Beijing Nova Program (20230484362, 20240484703), and National Key Laboratory for Multimedia Information Processing.

References

Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems*, 2017.

Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. Deep rewiring: Training very sparse deep networks. In *International Conference on Learning Representations*, 2018.

Sander M Bohte, Joost N Kok, and Johannes A La Poutré. Spikeprop: backpropagation for networks of spiking neurons. In *European Symposium on Artificial Neural Networks*, 2000.

Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ANN-SNN conversion for high-accuracy and ultra-low-latency spiking neural networks. In *International Conference on Learning Representations*, 2022.

Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 2015.

Yanqi Chen, Zhaofei Yu, Wei Fang, Tiejun Huang, and Yonghong Tian. Pruning of deep spiking neural networks through gradient rewiring. In *International Joint Conferences on Artificial Intelligence*, 2021.

Yanqi Chen, Zhaofei Yu, Wei Fang, Zhengyu Ma, Tiejun Huang, and Yonghong Tian. State transition of dendritic spines improves learning of sparse spiking neural networks. In *International Conference on Machine Learning*, 2022.

Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 2018.

Michael V DeBole, Brian Taba, Arnon Amir, Filipp Akopyan, Alexander Andreopoulos, William P Risk, Jeff Kusnitz, Carlos Ortega Otero, Tapan K Nayak, Rathinakumar Appuswamy, et al. TrueNorth: Accelerating from zero to 64 million neurons in 10 years. *Computer*, 2019.

Lei Deng, Yujie Wu, Yifan Hu, Ling Liang, Guoqi Li, Xing Hu, Yufei Ding, Peng Li, and Yuan Xie. Comprehensive snn compression using admm optimization and activity regularization. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. In *International Conference on Learning Representations*, 2021.

Shikuang Deng, Yuhang Li, Shanghang Zhang, and Shi Gu. Temporal efficient training of spiking neural network via gradient re-weighting. In *International Conference on Learning Representations*, 2022.

Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *International Joint Conference on Neural Networks*, 2015.

Jianhao Ding, Zhaofei Yu, Yonghong Tian, and Tiejun Huang. Optimal ANN-SNN conversion for fast and accurate inference in deep spiking neural networks. In *International Joint Conference on Artificial Intelligence*, 2021.

Steven K. Esser, Paul A. Merolla, John V. Arthur, Andrew S. Cassidy, Rathinakumar Appuswamy, Alexander Andreopoulos, David J. Berg, Jeffrey L. McKinstry, Timothy Melano, Davis R. Barch, Carmelo di Nolfo, Pallab Datta, Arnon Amir, Brian Taba, Myron D. Flickner, and Dharmendra S. Modha. Convolutional networks for fast, energy-efficient neuromorphic computing. *National Academy of Sciences*, 2016.

Biao Fang, Yuhao Zhang, Rui Yan, and Huajin Tang. Spike trains encoding optimization for spiking neural networks implementation in fpga. In *International Conference on Advanced Computational Intelligence*, 2020.

Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. *Advances in Neural Information Processing Systems*, 2021a.

Wei Fang, Zhaofei Yu, Yanqi Chen, Timothée Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *International Conference on Computer Vision*, 2021b.

Georgios Georgiadis. Accelerating convolutional neural networks via activation map compression. In *Computer Vision and Pattern Recognition*, 2019.

Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

Bing Han, Gopalakrishnan Srinivasan, and Kaushik Roy. RMP-SNN: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In *Computer Vision and Pattern Recognition*, 2020.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, 2015.

Zecheng Hao, Tong Bu, Jianhao Ding, Tiejun Huang, and Zhaofei Yu. Reducing ann-snn conversion error through residual membrane potential. *arXiv preprint arXiv:2302.02091*, 2023a.

Zecheng Hao, Jianhao Ding, Tong Bu, Tiejun Huang, and Zhaofei Yu. Bridging the gap between anns and snns by calibrating offset spikes. In *International Conference on Learning Representations*, 2023b.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition*, 2016.

Yangfan Hu, Huajin Tang, and Gang Pan. Spiking deep residual networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2018.

Eugene M Izhikevich. Which model to use for cortical spiking neurons? *IEEE Transactions on Neural Networks*, 2004.

Seijoon Kim, Seongsik Park, Byunggook Na, and Sungroh Yoon. Spiking-YOLO: Spiking neural network for energy-efficient object detection. In *AAAI Conference on Artificial Intelligence*, 2020.

Youngeun Kim and Priyadarshini Panda. Revisiting batch normalization for training low-latency deep spiking neural networks from scratch. *Frontiers in Neuroscience*, 2020.

Mark Kurtz, Justin Kopinsky, Rati Gelashvili, Alexander Matveev, John Carr, Michael Goin, William Leiserson, Sage Moore, Nir Shavit, and Dan Alistarh. Inducing and exploiting activation sparsity for fast inference on deep neural networks. In *International Conference on Machine Learning*, 2020.

Aditya Kusupati, Vivek Ramanujan, Raghav Somani, Mitchell Wortsman, Prateek Jain, Sham Kakade, and Ali Farhadi. Soft threshold weight reparameterization for learnable sparsity. In *International Conference on Machine Learning*, 2020.

Chankyu Lee, Syed Shakib Sarwar, Priyadarshini Panda, Gopalakrishnan Srinivasan, and Kaushik Roy. Enabling spike-based backpropagation for training deep neural network architectures. *Frontiers in Neuroscience*, 2020.

Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, 2016.

Guoqi Li, Lei Deng, Huajin Tang, Gang Pan, Yonghong Tian, Kaushik Roy, and Wolfgang Maass. Brain-inspired computing: A systematic survey and future trends. *Proceedings of the IEEE*, 2024.

Zonglin Li, Chong You, Srinadh Bhojanapalli, Daliang Li, Ankit Singh Rawat, Sashank J Reddi, Ke Ye, Felix Chern, Felix Yu, Ruiqi Guo, et al. The lazy neuron phenomenon: On emergence of activation sparsity in transformers. In *International Conference on Learning Representations*, 2023.

Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural Networks*, 1997.

Huizi Mao, Song Han, Jeff Pool, Wenshuo Li, Xingyu Liu, Yu Wang, and William J Dally. Exploring the granularity of sparsity in convolutional neural networks. In *Computer Vision and Pattern Recognition Workshops*, 2017.

Qingyan Meng, Shen Yan, Mingqing Xiao, Yisen Wang, Zhouchen Lin, and Zhi-Quan Luo. Training much deeper spiking neural networks with a small number of time-steps. *Neural Networks*, 2022.

Hesham Mostafa. Supervised learning based on temporal coding in spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2017.

Simon Narduzzi, Siavash A Bigdeli, Shih-Chii Liu, and L Andrea Dunbar. Optimizing the consumption of spiking neural networks with activity regularization. In *International Conference on Acoustics, Speech and Signal Processing*, 2022.

Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 2019.

Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Learning to be efficient: Algorithms for training low-latency, low-compute deep spiking neural networks. In *ACM Symposium on Applied Computing*, 2016.

Nicolas Perez Nieves and Dan F. M. Goodman. Sparse spiking gradient descent. In *Advances in Neural Information Processing Systems*, 2021.

Jing Pei, Lei Deng, Sen Song, Mingguo Zhao, Youhui Zhang, Shuang Wu, Guanrui Wang, Zhe Zou, Zhenzhi Wu, Wei He, et al. Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature*, 2019.

Yusuke Sakemi, Kakei Yamamoto, Takeo Hosomi, and Kazuyuki Aihara. Sparse-firing regularization methods for spiking neural networks with time-to-first-spike coding. *Scientific Reports*, 2023.

Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: VGG and residual architectures. *Frontiers in Neuroscience*, 2019.

Jiangrong Shen, Qi Xu, Jian K. Liu, Yueming Wang, Gang Pan, and Huajin Tang. ESL-SNNs: An evolutionary structure learning strategy for spiking neural networks. In *AAAI Conference on Artificial Intelligence*, 2023.

Xinyu Shi, Jianhao Ding, Zecheng Hao, and Zhaofei Yu. Towards energy efficient spiking neural networks: An unstructured pruning framework. In *International Conference on Learning Representations*, 2024.

Sumit B Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. *Advances in Neural Information Processing Systems*, 2018.

Kenneth Michael Stewart and Emre Neftci. Meta-learning spiking neural networks with surrogate gradient descent. *Neuromorphic Computing and Engineering*, 2022.

William E Vinje and Jack L Gallant. Sparse coding and decorrelation in primary visual cortex during natural vision. *Science*, 287(5456):1273–1276, 2000.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. *Advances in Neural Information Processing Systems*, 2016.

Doudou Wu, Xianghong Lin, and Pangao Du. An adaptive structure learning algorithm for multi-layer spiking neural networks. In *International Conference on Computational Intelligence and Security*. IEEE, 2019.

Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience*, 2018.

Yulong Yan, Haoming Chu, Yi Jin, Yuxiang Huan, Zhuo Zou, and Lirong Zheng. Backpropagation with sparsity regularization for spiking neural network learning. *Frontiers in Neuroscience*, 2022.

Man Yao, Ole Richter, Guangshe Zhao, Ning Qiao, Yannan Xing, Dingheng Wang, Tianxiang Hu, Wei Fang, Tugba Demirci, Michele De Marchi, et al. Spike-based dynamic computing with asynchronous sensing-computing neuromorphic chip. *Nature Communications*, 2024.

Hang Yin, John Boaz Lee, Xiangnan Kong, Thomas Hartvigsen, and Sihong Xie. Energy-efficient models for high-dimensional spike train classification using sparse spiking neural networks. In *ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021.

Takashi Yoshida and Kenichi Ohki. Natural images are reliably represented by sparse and variable populations of neurons in visual cortex. *Nature communications*, 2020.

Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *Computer Vision and Pattern Recognition*, 2018.

Friedemann Zenke and Emre O Neftci. Brain-inspired learning on neuromorphic substrates. *Proceedings of the IEEE*, 2021.

Friedemann Zenke and Tim P Vogels. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *Neural Computation*, 2021.

Malu Zhang, Jiadong Wang, Jibin Wu, Ammar Belatreche, Burin Amornpaisannon, Zhixuan Zhang, Venkata Pavan Kumar Miriyala, Hong Qu, Yansong Chua, Trevor E Carlson, et al. Rectified linear postsynaptic potential function for backpropagation in deep spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. In *AAAI Conference on Artificial Intelligence*, 2021.

Zhaokun Zhou, Kaiwei Che, Jun Niu, Man Yao, Guoqi Li, Li Yuan, Guibo Luo, and Yuesheng Zhu. Spatial-temporal spiking feature pruning in spiking transformer. *IEEE Transactions on Cognitive and Developmental Systems*, 2024.

Michael H Zhu and Suyog Gupta. To prune, or not to prune: Exploring the efficacy of pruning for model compression. In *Advances in Neural Information Processing Systems*, 2018.

Yaoyu Zhu, Zhaofei Yu, Wei Fang, Xiaodong Xie, Tiejun Huang, and Timothée Masquelier. Training spiking neural networks with event-driven backpropagation. In *Advances in Neural Information Processing Systems*, 2022.

Tao Zhuang, Zhixuan Zhang, Yuheng Huang, Xiaoyi Zeng, Kai Shuang, and Xiang Li. Neuron-level structured pruning using polarization regularizer. In *Advances in Neural Information Processing Systems*, 2020.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We faithfully list the contribution of the paper in the introduction and also in the abstract.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitation of the proposed work in the discussion section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: For each theoretical results, the proof can be either found in the main text or supplementary material.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.

- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We will make sure that the paper contain all the information to reproduce the results and we will also open-source the code.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We will provide open access to the code while the data we use are all publicly available.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide all the detail in the training and evaluating process in the supplementary material.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Due to limited computational resources, we report results from a single run. However, all results are reproducible.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The information of computer resources for each experiments can be found in the supplementary material.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This research adheres fully to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss both potential positive societal impacts and negative societal impacts of the research.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The creators, original owners of assets are properly credited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: New assets introduced in the paper are well documented.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: the core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.