

Supplementary Materials

Anonymous Author(s)

Affiliation

Address

email

1 S1 Proofs

2 S1.1 Proof for the gradient vanishing problem

3 In the first part, we will first derive the gradient value of the weights with respect to the total loss
 4 during backpropagation. We consider two adjacent neurons i and neuron j with weight (bias)
 5 connection $w_{ij}; b_i$ from j to i . Therefore, we can recursively derive the gradient $\frac{\partial \mathcal{L}}{\partial w_{ij}}$ and $\frac{\partial \mathcal{L}}{\partial s_i[t]}$. For
 6 clarity, we will first write down the neuron function as follows.

$$x_i[t] = w_{ij}s_j[t] + b_i, \quad (\text{S1})$$

$$m_i[t] = \tau u_i[t-1] + x_i[t], \quad (\text{S2})$$

$$s_i[t] = H(m_i[t] - \theta_i), \quad (\text{S3})$$

$$u_i[t] = (1 - H(m_i[t] - \theta_i)) \cdot m_i[t]. \quad (\text{S4})$$

7 According to the neuron functions, we can derive the gradient for each component based on the chain
 8 rule. We use $\frac{\partial \mathcal{L}}{\partial m_i[t]}$ as an intermediate value to derive the gradient at each time-step. For simplicity,
 9 we use $\Theta(m_i[t]) = \frac{\partial s_i[t]}{\partial m_i[t]}$ to represent the surrogate function. In practice, we consider the common
 10 situation that the threshold value is set to one $\theta_i = 1$, and $\Theta(m_i[t]) = |1 - m_i[t]|$.

$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = \sum_{t=1}^T \frac{\partial \mathcal{L}}{\partial m_i[t]} \frac{\partial m_i[t]}{\partial x_i[t]} \frac{\partial x_i[t]}{\partial w_{ij}} = \sum_{t=1}^T \frac{\partial \mathcal{L}}{\partial m_i[t]} s_j[t] \quad (\text{S5})$$

$$\frac{\partial \mathcal{L}}{\partial s_j[t]} = \frac{\partial \mathcal{L}}{\partial m_i[t]} \frac{\partial m_i[t]}{\partial x_i[t]} \frac{\partial x_i[t]}{\partial s_j[t]} = \frac{\partial \mathcal{L}}{\partial m_i[t]} w_{ij} \quad (\text{S6})$$

$$\frac{\partial \mathcal{L}}{\partial m_i[t]} = \frac{\partial \mathcal{L}}{\partial s_i[t]} \frac{\partial s_i[t]}{\partial m_i[t]} + \frac{\partial \mathcal{L}}{\partial m_i(t+1)} \frac{\partial m_i(t+1)}{\partial u_i[t]} \frac{\partial u_i[t]}{\partial m_i[t]} \quad (\text{S7})$$

$$= \frac{\partial \mathcal{L}}{\partial s_i[t]} \Theta(m_i[t]) + \frac{\partial \mathcal{L}}{\partial m_i(t+1)} \tau (1 - s_i[t] - m_i[t] \Theta(m_i[t])) \quad (\text{S8})$$

11 We can then recursively derive the value of $\frac{\partial \mathcal{L}}{\partial m_i[t]}$ at any time t , while T is the total time-step.

$$\frac{\partial \mathcal{L}}{\partial m_i[t]} = \frac{\partial \mathcal{L}}{\partial s_i[t]} \Theta(m_i[t]) + \frac{\partial \mathcal{L}}{\partial m_i(t+1)} \tau (1 - s_i[t] - m_i[t] \Theta(m_i[t])) \quad (\text{S9})$$

$$= \sum_{p=t}^T \left(\tau^{p-t} \frac{\partial \mathcal{L}}{\partial s_i(p)} \Theta(m_i(p)) \prod_{q=t}^{p-1} (1 - s_i(q) - m_i(q) \Theta(m_i(q))) \right) \quad (\text{S10})$$

12 The Eq. (S10) is considered as the intermediate value for recursively calculate all gradient values for
 13 each layer and each time-steps. Due to the special firing nature of the spiking neuron, we can further
 14 derive that the gradient values for weight parameters are finite. We will first prove lemma below.

15 **Lemma 1.** The absolute value of the gradient of the reset function $|\frac{\partial u_i[t]}{\partial m_i[t]}|$ is less than 1, which is

$$|\frac{\partial u_i[t]}{\partial m_i[t]}| = |(1 - s_i[t] - m_i[t]\Theta(m_i[t]))| \leq 1. \quad (\text{S11})$$

16 **Proof.** Here we consider three situations. If $m_i[t] \leq 0$, then $|(1 - s_i[t] - m_i[t]\Theta(m_i[t]))| = 1$;
 17 If $0 < m_i[t] < 1$, then $(1 - s_i[t] - m_i[t]\Theta(m_i[t])) = |(1 - m_i^2[t])| < 1$; If $m_i[t] \geq 1$, then
 18 $|(1 - s_i[t] - m_i[t]\Theta(m_i[t]))| = |m_i[t]|2 - m_i[t]| \leq 1$;

19 Based on the above lemma, we further derive the upper bound for $|\frac{\partial \mathcal{L}}{\partial s_j[t]}|$ as

$$|\frac{\partial \mathcal{L}}{\partial s_j[t]}| = |\frac{\partial \mathcal{L}}{\partial m_i[t]} w_{ij}| \quad (\text{S12})$$

$$= \left| w_{ij} \sum_{p=t}^T \left(\tau^{p-t} \frac{\partial \mathcal{L}}{\partial s_i(p)} \Theta(m_i(p)) \prod_{q=t}^{p-1} 1 - s_i(q) - m_i(q)\Theta(m_i(q)) \right) \right| \quad (\text{S13})$$

$$\leq |w_{ij}| \sum_{p=t}^T \left| \left(\tau^{p-t} \frac{\partial \mathcal{L}}{\partial s_i(p)} \Theta(m_i(p)) \prod_{q=t}^{p-1} 1 - s_i(q) - m_i(q)\Theta(m_i(q)) \right) \right| \quad (\text{S14})$$

$$\leq |w_{ij}| \sum_{p=t}^T \left| \frac{\partial \mathcal{L}}{\partial s_i(p)} \right| \quad (\text{S15})$$

20 We assume the target loss is MSE or Cross-entropy loss during training, at arbitrary time t in the
 21 output layer, we can easily drive that the gradient value of the output spike is finite, which is for each
 22 neuron in the last layer, we have $\sum_t s_i[t] < \epsilon$, $\left(\max_t \left| \frac{\partial \mathcal{L}_{total}}{\partial s_i[t]} \right| \right) < +\infty$. Therefore, according to
 23 Eq. (S15), we can recursively derive that values of $|\frac{\partial \mathcal{L}}{\partial s_j[t]}|$ in each layer is finite, which is

$$|\frac{\partial \mathcal{L}}{\partial s_i[t]}| < +\infty \text{ and also } \left(\max_t \sum_{p=t}^T \left| \frac{\partial \mathcal{L}}{\partial s_i(p)} \right| \right) < +\infty. \quad (\text{S16})$$

24 Therefore, similarly, we can derive the upper bound for the absolute value of the gradient for all
 25 weights.

$$|\frac{\partial \mathcal{L}}{\partial w_{ij}}| = \left| \sum_{t=1}^T \frac{\partial \mathcal{L}}{\partial m_i[t]} s_j[t] \right| \leq \sum_{t=1}^T \left| \frac{\partial \mathcal{L}}{\partial m_i[t]} s_j[t] \right| \quad (\text{S17})$$

$$= \sum_{t=1}^T \left| s_j[t] \sum_{p=t}^T \left(\tau^{p-t} \frac{\partial \mathcal{L}}{\partial s_i(p)} \Theta(m_i(p)) \prod_{q=t}^{p-1} 1 - s_i(q) - m_i(q)\Theta(m_i(q)) \right) \right| \quad (\text{S18})$$

$$\leq \sum_{t=1}^T s_j[t] \sum_{p=t}^T \left| \left(\tau^{p-t} \frac{\partial \mathcal{L}}{\partial s_i(p)} \Theta(m_i(p)) \prod_{q=t}^{p-1} 1 - s_i(q) - m_i(q)\Theta(m_i(q)) \right) \right| \quad (\text{S19})$$

$$\leq \sum_{t=1}^T s_j[t] \sum_{p=t}^T \left| \frac{\partial \mathcal{L}}{\partial s_i(p)} \right| \leq \left(\sum_{t=1}^T s_j[t] \right) \left(\max_t \sum_{p=t}^T \left| \frac{\partial \mathcal{L}}{\partial s_i(p)} \right| \right) \quad (\text{S20})$$

26 Since we already proved that $\left(\max_t \sum_{p=t}^T \left| \frac{\partial \mathcal{L}}{\partial s_i(p)} \right| \right) < +\infty$, we have

$$\begin{aligned} \forall i \quad \sum_t s_i[t] \rightarrow 0 &\Rightarrow \left| \frac{\partial \mathcal{L}}{\partial w_{ij}} \right| \rightarrow 0, \\ \forall i \quad \sum_t s_i[t] = 0 &\Rightarrow \left| \frac{\partial \mathcal{L}}{\partial w_{ij}} \right| = 0. \end{aligned} \quad (\text{S21})$$

27 Therefore, we prove that \mathbf{S} are one sets of the saddle points during optimization.

28 S1.2 Derivation for the threshold update equation

29 As we discuss in the main text, we regularize the threshold to balance the sparser activation and
30 overall performance, which is to minize the total loss as

$$\arg \min_{\theta} \mathcal{L}_{total} \quad (\text{S22})$$

31 Since we use the BPTT algorithm for training, we calculate the first-order derivative of the threshold
32 with respect to the total loss as update value.

$$\frac{\partial \mathcal{L}_{total}}{\partial \theta_i} = \frac{\partial \mathcal{L}_{task}}{\partial \theta_i} + \frac{\partial \mathcal{L}_{reg}}{\partial \theta_i} \quad (\text{S23})$$

$$= \frac{\partial \mathcal{L}_{task}}{\partial \theta_i} + \frac{\partial \lambda_s \sum_{i=1}^N \|s_i\|_2^2}{\partial \theta_i} \quad (\text{S24})$$

$$= \frac{\partial \mathcal{L}_{task}}{\partial \theta_i} + \frac{\partial \lambda_s \sum_{i=1}^N \sum_{t=1}^T s_i^2[t]}{\partial \theta_i}. \quad (\text{S25})$$

33 As discussed in Section 3.2 of main text, adjusting the threshold of the AT-LIF neuron reduces the
34 firing rate while exerting minimal influence on the neuron’s average output. Therefore, one can
35 consider each threshold as independent variable that only affect the internal dynamic of the neuron.
36 Consequently, we can simplify the optimization process by detaching the threshold gradient between
37 neurons. Based on this formulation, we further derive

$$\frac{\partial \mathcal{L}_{total}}{\partial \theta_i} = \frac{\partial \mathcal{L}_{task}}{\partial \theta_i} + \frac{\partial \lambda_s \sum_{t=1}^T s_i^2[t]}{\partial \theta_i} \quad (\text{S26})$$

$$= \frac{\partial \mathcal{L}_{task}}{\partial \theta_i} + 2\lambda_s \sum_{t=1}^T s_i[t] \frac{\partial s_i[t]}{\partial \theta_i} \quad (\text{S27})$$

$$= \frac{\partial \mathcal{L}_{task}}{\partial \theta_i} + \lambda_t \sum_{t=1}^T s_i[t] \frac{\partial s_i[t]}{\partial \theta_i}. \quad (\text{S28})$$

38 To distinguish between the regularization-based methods, we use λ_t instead of λ_s to stand for the
39 coefficient parameter of the threshold adaptation.

40 S2 General Experimental Settings

41 S2.1 Overall training algorithm

42 In this section, we demonstrate the overall training algorithm of the model with AT-LIF neuron while
43 using naive SGD as the optimizer. We provide the pseudo-code of the training process with naive
44 SGD optimizer. As shown in Algorithm 1, we treat the threshold at each layer as a trainable variable.
45 During each training iteration, standard backpropagation (BPTT) is first applied to update the network
46 weights, including all thresholds. Then, an additional update is computed and manually applied for
47 each threshold based on the spike activity. The decoupled update ensures that threshold regularization
48 is not influenced by the choice of optimizer, enabling more faithful optimization of spike activity
49 through threshold adaptation. For detailed PyTorch implementation, please refer to the provided code
50 examples.

51 S2.2 Detailed experimental setting

52 **Environment** The experiments are conducted with Nvidia RTX 4090 GPU on Ubuntu 22.04. For
53 all experiments other than ImageNet, we use single GPU for training and evaluating. While for
54 ImageNet, a 4-GPU parallel training is applied.

55 **Architecture** Following previous researches [Chen et al., 2022, Shi et al., 2024], we use ResNet-20,
56 6 Conv 2 FC for CIFAR-10 dataset, SEW-ResNet18 for ImageNet, ImageNet-100 and VGGSNN
57 for DVS-CIFAR10 dataset. In Tab. 3 of the main text, for fair comparison, we use exactly the same
58 architecture as those previous works, While in Tab. 2, we utilize the standard ResNet-20 for CIFAR
59 datasets [He et al., 2016].

Algorithm 1 Training Algorithm for model with AT-LIF (With Naive SGD)

```
1: input SNN  $f_{SNN}(\cdot; \theta, w)$  with weights and thresholds as parameters.
2: Random Initialize  $w, \theta \leftarrow 1$ 
3: for  $k = 1$  to  $M$  do
4:   Random sample  $(x_k, y_k)$  in  $\mathcal{D}$ .
5:   Model forward  $\hat{y}_k = f_{SNN}(x_k; \theta, w)$ .
6:   Calculate Loss  $\mathcal{L} = \mathcal{L}_{reg}(\hat{y}_k, y_k)$ .
7:   Model backward and get  $\frac{\partial \mathcal{L}}{\partial w}$  and  $\frac{\partial \mathcal{L}}{\partial \theta}$ .
8:   Model update  $w \leftarrow w - \eta \frac{\partial \mathcal{L}}{\partial w}, \theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}}{\partial \theta}$ .
9:   for  $i = 1$  to  $N$  do
10:    Threshold Regularization  $\theta_i \leftarrow \theta_i - \eta \lambda_t \sum_{t=1}^T s_i[t] \frac{\partial s_i[t]}{\partial \theta_i}$ .
11:   end for
12: end for
13: return  $f_{SNN}(\cdot; \theta, w)$ .
```

60 **Hyper-parameter Settings** In Tab. S1, we list all neuron settings and hyper-parameter settings. For
61 fair comparison, we use fixed hyper-parameter setting for each dataset (Column 1) across different
62 methods (Column 2).

Dataset	Method	T	τ	LR	Weight Decay	Batchsize	Epochs	Optimizer	Loss Func
CIFAR-10	AT-LIF	8	0.9	0.1	5e-4	128	200	SGD	CE
CIFAR-10	STDS	8	0.9	0.1	0	32	500	SGD	MSE
ImageNet-100	AT-LIF	4	1.0	0.1	5e-4	512	200	SGD	CE
ImageNet-100	STDS	4	1.0	0.1	0	512	200	SGD	CE
ImageNet	AT-LIF	4	1.0	0.1	5e-4	512	200	SGD	CE
ImageNet	STDS	4	1.0	0.1	0	512	200	SGD	CE
DVS-CIFAR	AT-LIF	10	0.9	0.1	0.001	64	300	SGD	CE

Table S1: Hyper-parameter and neuron settings.

63 S3 Experiments

64 S3.1 Average spike activity at each layer

65 To provide a more detailed view of spike activity suppression across network layers, we present
66 layer-wise mean firing rate comparisons between our proposed AT-LIF model and the baseline
67 activity regularization (AR) method on CIFAR-10, DVS-CIFAR10, and ImageNet-100. In all three
68 datasets, Most AT-LIF results (blue bars) exhibits lower firing rates across most layers compared to
69 AR (green bars), particularly in the deeper layers. On CIFAR-10, AT-LIF maintains significantly
70 lower firing activity in the final layer, contributing to an significant reduction in average firing rate
71 while preserving a higher accuracy. A similar trend is observed on DVS-CIFAR10, where AT-LIF
72 suppresses firing more effectively in both early and late layers while achieving slightly better accuracy.
73 On the more challenging ImageNet-100 task, AT-LIF achieves comparable accuracy to AR but with
74 markedly lower spike activity across most layers. These results highlight AT-LIF’s ability to uniformly
75 regulate neuronal activity across the network, enabling efficient spike sparsity without compromising
76 performance.

77 S3.2 Trade-off between Efficiency and Accuracy

78 To more accurately evaluate the effectiveness of spike activity pruning, we plot the trade-off between
79 total spike count and classification accuracy, as shown in the figure. The spike count is computed as
80 the average number of total spike events required to classify a single image across the CIFAR-10
81 test set. Each point on the curve represents a model trained under different regularization strengths.
82 The AT-LIF model (blue line) consistently achieves significantly lower spike counts compared to
83 the AR baseline (green line) at comparable or even higher accuracy levels. Notably, while both
84 methods exhibit a trade-off between spike sparsity and performance, AT-LIF maintains a much flatter

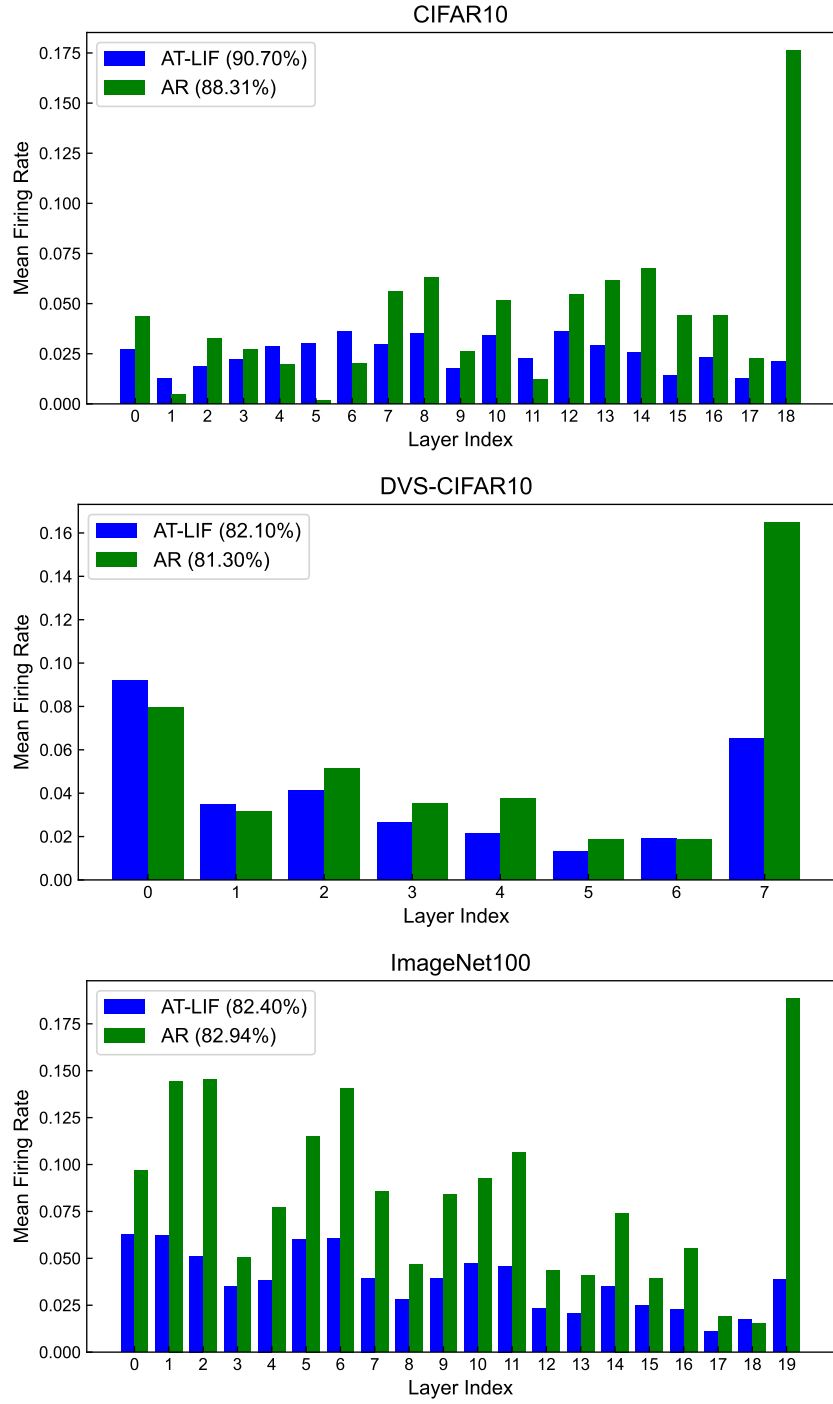


Figure S1: Comparison of average firing rate at each layer using activation regularization and AT-LIF.

85 curve, indicating its superior ability to suppress spike activity without compromising accuracy. This
 86 demonstrates the advantage of the AT-LIF neuron in enabling energy-efficient SNNs with minimal
 87 performance degradation.

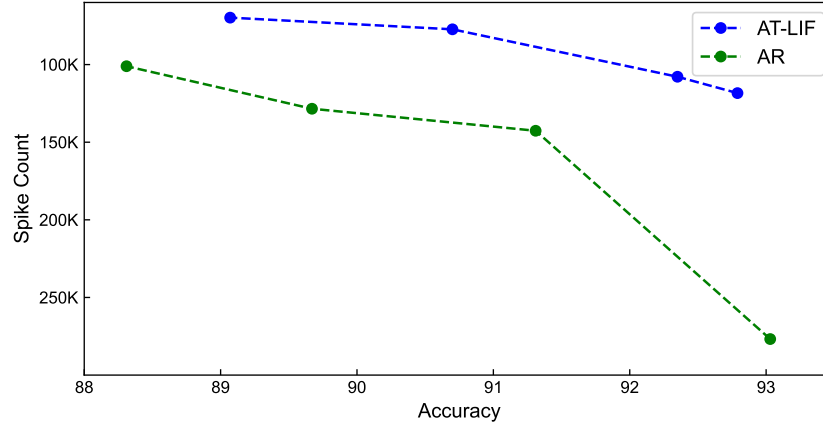


Figure S2: Comparison of the trade-off between spike count and accuracy using activation regularization and AT-LIF in CIFAR-10.

88 References

- 89 Yanqi Chen, Zhaofei Yu, Wei Fang, Zhengyu Ma, Tiejun Huang, and Yonghong Tian. State tran-
 90 sition of dendritic spines improves learning of sparse spiking neural networks. In *International*
 91 *Conference on Machine Learning*, 2022.
- 92 Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image
 93 recognition. In *Computer Vision and Pattern Recognition*, 2016.
- 94 Xinyu Shi, Jianhao Ding, Zecheng Hao, and Zhaofei Yu. Towards energy efficient spiking neu-
 95 ral networks: An unstructured pruning framework. In *International Conference on Learning*
 96 *Representations*, 2024.