Stable Spike: Dual Consistency Optimization via Bitwise AND Operations for Spiking Neural Networks

Yongqi Ding, Kunshan Yang, Linze Li, Yiyang Zhang, Mengmeng Jing, Lin Zuo*
School of Information and Software Engineering
University of Electronic Science and Technology of China

Description of the Supplementary Material

This supplementary material contains four sections:

Section A interprets the effectiveness of the AND operation from the perspective of mutual information.

Section B details the experimental setup, including the dataset and training strategy.

Section C presents additional experimental results, including further comparative experiments and extensions of our method.

Section D provides an overhead analysis showing that our method requires only negligible training overhead and leaves inference completely unaffected.

Section E provides supplementary visualizations demonstrating that the AND operation consistently extracts stable pulse skeletons.

A. Interpreting the AND Operation from the Perspective of Mutual Information

The AND operation extracts the mutual information between adjacent timesteps. The $(1, 1)$ pair represents the consensus feature between two consecutive timesteps, which eliminates transient noise. According to the information bottleneck principle, the convergence of spike maps toward stable spikes maximizes mutual information in order to identify the minimal sufficient representation. In contrast, the XOR and XOR operations are inferior to the AND operation and may even exhibit degradation due to interference from (0,0) and (0,1).

B. Experimental Details

B.1. Neuromorphic Datasets.

We conducted experiments on three neuromorphic object recognition benchmark datasets: CIFAR10-DVS [11], DVS-Gesture [1], and N-Caltech101 [14]. The CIFAR10-DVS [11] dataset contains a total of 10,000 samples from ten categories. The DVS-Gesture [1] dataset contains neuromorphic data for 11 hand gestures with 1176 training samples and 288 test samples. The spatial resolution of each sample in CIFAR10-DVS and DVS-Gesture is 128×128, which we downsampled to 48×48 and input to SNN, the standard input size for the community [5, 21]. The N-Caltech101 [14] dataset contains 8,109 samples, categorized into 101 categories, with an original resolution of 180×240. We also downsampled the samples in N-Caltech101 to 48×48.

We conducted our experiments on an Ubuntu 20.04.5 operating system with an NVIDIA 4090 GPU. To validate the versatility of the proposed method, we conducted experiments using various architectures, including VGG-9, ResNet-18, VGGSNN, and QKFormer. For VGG-9 and ResNet-18, we use the same training strategy as in [5]. The model was trained for 100 epochs using a stochastic gradient descent optimizer. The initial learning rate was set to 0.1, and it was scaled down tenfold every 30 epochs. The

*Corresponding author (linzuo@uestc.edu.cn).

Table 1. Structures of VGG-9, VGGSNN, ResNet-18, and ResNet-19, where AP, GAP, and FC denote average pooling, global average pooling, and fully connected layers, respectively.

Stage	VGG-9	VGGSNN	ResNet-18	ResNet-19
1	-	-	Conv(3×3@64)	Conv(3×3@128)
1	Conv(3×3@64) Conv(3×3@128)	Conv(3×3@64) Conv(3×3@128)	$\begin{pmatrix}\text{Conv}(3\times3@64)\\ \text{Conv}(3\times3@64)\end{pmatrix}\times2$	$\begin{pmatrix}\text{Conv}(3\times3@128)\\ \text{Conv}(3\times3@128)\end{pmatrix}\times3$
	AP(stride=2)	AP(stride=2)	-	-
2	Conv(3×3@256) Conv(3×3@256)	Conv(3×3@256) Conv(3×3@256)	$\begin{pmatrix}\text{Conv}(3\times3@128)\\ \text{Conv}(3\times3@128)\end{pmatrix}\times2$	$\begin{pmatrix}\text{Conv}(3\times3@256)\\ \text{Conv}(3\times3@256)\end{pmatrix}\times3$
	AP(stride=2)	AP(stride=2)	-	-
3	Conv(3×3@512) Conv(3×3@512)	Conv(3×3@512) Conv(3×3@512)	$\begin{pmatrix}\text{Conv}(3\times3@256)\\ \text{Conv}(3\times3@256)\end{pmatrix}\times2$	$\begin{pmatrix}\text{Conv}(3\times3@512)\\ \text{Conv}(3\times3@512)\end{pmatrix}\times2$
	AP(stride=2)	AP(stride=2)	-	
4	Conv(3×3@512) Conv(3×3@512)	Conv(3×3@512) Conv(3×3@512)	$\begin{pmatrix}\text{Conv}(3\times3@512)\\ \text{Conv}(3\times3@512)\end{pmatrix}\times2$	
	GAP,FC	AP,FC	GAP,FC	GAP,FC$\times$2

Table 2. Comparative results (%) with adaptive Gaussian noise.

Method	CIFAR10-DVS	DVS-Gesture
Adaptive Gaussian noise	73.9	88.54
Ours (Amplitude-aware spike noise)	**77.1**	**94.44**

batch size is 64, and the weight decay is 1e-3. In particular, for VGG-9 and ResNet-18, we do not use any data augmentation during training. The firing threshold of the spiking neuron was set to 1.0, and the membrane potential time constant was set to 2.0. In the training of the SNN, the rectangular surrogate gradient function is employed for backpropagation, consistent with [5]. Table 1 shows the VGG-9 and ResNet-18 architectures.

When using VGGSNN for recognizing CIFAR10-DVS, we used random cropping and horizontal flipping as data augmentation. When using the VGGSNN architecture on the N-Caltech101 dataset, we incorporate the knowledge transfer strategy [8] and employ the same training strategy as in [8]. Similarly, when using the QKFormer architecture, we follow the training strategy outlined in the original paper [21] to ensure the fairness of the experiments. We set the balancing coefficients β and γ to 1.0 by default, and the experiments in Section 4.1 demonstrate the influence of these coefficients on performance. However, when we use VGGSNN to recognize the N-Caltech101 dataset and incorporate knowledge transfer, the magnitude of our loss does not match the original transfer loss. At this point, we set the balancing coefficients to 1e-3.

B.2. Static Datasets.

For static images, we conducted experiments on the CIFAR10 and CIFAR100 datasets. Both the CIFAR-10 and CIFAR-100 datasets contain 60,000 32×32 images, 50,000 of which are in the training set and 10,000 of which are in the test set. We normalized the CIFAR10 and CIFAR100 samples to a zero mean and unit variance. Then, we applied the standard data augmentation strategies, AutoAugment [3] and Cutout [4].

For CIFAR10 and CIFAR100, we applied the ResNet-18 and ResNet-19 architectures. We used a stochastic gradient descent optimizer with a momentum of 0.9 to train for 300 epochs. The initial learning rate was set to 0.1, and we employed a cosine annealing learning rate strategy. The batch size is 128, and weight deacy is 5e-4. Additionally, we use the QKFormer architecture to recognize static targets with the same training strategy as the original [21].

On the ImageNet dataset, we train 300 epochs using ResNet-18 with a batch size of 256. The initial learning rate value is 0.1 and the learning rate is adjusted using a cosine annealing strategy. The default input size is 224×224, which is consistent with other methods.

Table 1 illustrates the architectures of the VGG-9, VGGSNN, ResNet-18, and ResNet-19 models that were used in the experiment. We use the same architecture and experimental setup as in our method when we reproduce CLIF [9], TAB [10], and SLT [2], but we use the officially released core code implementation to ensure performance.

Table 3. Comparative results (%) under extended training epochs.

Dataset	Method	100 Epoch	200 Epoch	300 Epoch
CIFAR10-DVS	Baseline	72.9	74.1	74.2
	Ours	**77.1**	**77.2**	**77.5**
DVS-Gesture	Baseline	87.15	89.58	89.58
	Ours	**94.44**	**95.14**	**95.49**

Table 4. Further comparative results under identical settings as MPS [5].

Dataset	Method	Architecture	$T\downarrow$	Accuracy (%) $\uparrow$
CIFAR10-DVS	MPS$^{ICLR'25}$ [5]	VGG-9	5	76.77
	Ours	VGG-9	5	**77.20**
	MPS$^{ICLR'25}$ [5]	VGGSNN	4	83.20
	Ours	VGGSNN	4	**83.70**
DVS-Gesture	MPS$^{ICLR'25}$ [5]	VGG-9	5	93.23
	Ours	VGG-9	5	**95.14**
N-Caltech101	MPS$^{ICLR'25}$ [5]	VGG-9	5	82.71
	Ours	VGG-9	5	**84.03**
	MPS$^{ICLR'25}$ [5]	VGGSNN	10	93.68
	Ours	VGGSNN	10	**94.25**

Table 5. Computing consistency loss by applying the AND operation across additional layers can further enhance performance.

Dataset	Method	Architecture	$T\downarrow$	Accuracy (%) $\uparrow$
CIFAR10-DVS	Baseline	VGG-9	4	72.9
	Ours	VGG-9	4	**77.1**
	Ours Dense	VGG-9	4	**78.8**
DVS-Gesture	Baseline	VGG-9	4	87.15
	Ours	VGG-9	4	**94.44**
	Ours Dense	VGG-9	4	**95.49**

C. Additional Experimental Results

C.1. Comparison with Adaptive Gaussian Noise

To further demonstrate that discrete spike noise outperforms continuous Gaussian noise in SNNs, we construct adaptive Gaussian noise using the spike firing rate as its standard deviation. As shown in Table 2, our amplitude-aware spike noise still exhibits significant performance advantages over adaptive Gaussian noise.

C.2. Further Comparison with the Baseline

The VGG-9 architecture was trained for 100 epochs by default. To show that our method consistently outperforms the baseline model rather than due to underfitting, we extended the training cycle to 300 epochs. The comparative results are presented in Table 3. These results demonstrate that our method consistently yields significant performance gains.

C.3. Further Comparison with MPS

Due to MPS's strong performance on neuromorphic datasets, Table 4 provides an additional comparison between our method and MPS [5]. This comparison ensures that all training strategies, including model architecture and timestep settings, are identical. Our method demonstrated superior performance across three datasets, consistently outperforming MPS.

C.4. Further Extension of the Proposed Method

By default, we only apply the AND operation to the output of the penultimate layer (before the fully connected layer) to extract the stable pulse skeleton and compute the consistency loss. Naturally, we can extend this implementation to more layers. For now, we'll call it Stable Spike Dense. To accomplish this, we treat every two convolutions in VGG-9 as a stage and compute the consistency loss by performing an AND operation on the output features of each stage. As more layers have been optimized for consistency, the results

Table 6. Comparative results on static CIFAR10 and CIFAR100 datasets.

Dataset	Method	Architecture	$T \downarrow$	Accuracy (%) $\uparrow$
CIFAR10	QKFormer$^{NeurIPS'24}$ [21]	QKFormer	4	96.18
	SNN-ViT$^{ICLR'25}$ [15]	SNN-ViT	4	96.10
	RateBP$^{NeurIPS'24}$ [18]	ResNet-19	4	96.26
	DeepTAGE$^{ICLR'25}$ [13]	ResNet-18	4	95.86
	SLT$^{AAAI'24}$ [2]	ResNet-19	4	95.18
	LSG$^{IJCAI'23}$ [12]	ResNet-19	4	95.17
	RateBP$^{NeurIPS'24}$ [18]	ResNet-18	4	95.61
	KDSNN$^{CVPR'23}$ [16]	ResNet-18	4	93.41
	Weak2Strong$^{NeurIPS'25}$ [6]	ResNet-19	4	96.66
	Ours	ResNet-18	4	**95.70**
		ResNet-19	4	**96.73**
CIFAR100	QKFormer$^{NeurIPS'24}$ [21]	QKFormer	4	81.15
	SNN-ViT$^{ICLR'25}$ [15]	SNN-ViT	4	80.10
	RateBP$^{NeurIPS'24}$ [18]	ResNet-18	4	78.26
	STAA-SNN$^{CVPR'25}$ [20]	ResNet-19	4	82.05
	DeepTAGE$^{ICLR'25}$ [13]	ResNet-19	4	81.39
	RateBP$^{NeurIPS'24}$ [18]	ResNet-19	4	80.71
	TS-SNN$^{ICML'25}$ [19]	ResNet-19	2	80.28
	ReverB-SNN$^{ICML'25}$ [7]	ResNet-19	2	78.46
	TMC$^{ICML'25}$ [17]	ResNet-19	4	77.52
	SLT$^{AAAI'24}$ [2]	ResNet-19	4	75.01
	Weak2Strong$^{NeurIPS'25}$ [6]	ResNet-19	4	82.02
	Ours	ResNet-18	4	**78.83**
		ResNet-19	4	**82.29**

in Table 5 indicate further performance improvements.

C.5. Experimental Results on CIFAR

Table 6 shows the experimental results of our method using the ResNet-18 and ResNet-19 architectures on CIFAR10 and CIFAR100. With four timesteps, we achieved accuracy rates of 96.73% on CIFAR10 and 82.29% on CIFAR100, significantly outperforming other methods.

C.6. Additional Hyperparameter Sensitivity Experiments

To further investigate the sensitivity of the proposed method to hyperparameters, we conducted additional experiments across architectures and timesteps on neuromorphic datasets. Table 7 shows that our method exhibits stable overall performance as long as the hyperparameters remain within reasonable ranges ($\beta = \gamma = 1.0$ and $\alpha = 2.0$). We recommend increasing β and γ appropriately when tasks exhibit significant temporal fluctuations and increasing α appropriately when interclass differences are excessive to smooth out variations in probability distributions.

C.7. Stable spikes can maintain temporal information.

Stable spikes maintain $T-1$ step temporal coherence by eliminating transient noise rather than erasing all temporal information. Legitimate temporal dynamics are essentially the ordered evolution of spatial features over time. This evolution rarely isolates the features from adjacent timesteps, thus enabling their preservation by stable spikes. Additionally, Table 8 shows that optimizing only the first and last two timesteps results in suboptimal performance. This further validates that applying our method to all timesteps does not eliminate useful temporal information, but rather enhances it.

D. Negligible Training Overhead

Our method only requires performing the AND operation on the last layer spike maps during training. Then, forward propagation is performed with noise added. The AND bit operation and the cost of generating random noise are negligible, and forward propagation only involves one fully connected layer. Therefore, we did not observe any significant memory or time overhead during training. During infer-

Table 7. Hyperparameter sensitivity experiments (%) across architectures and timesteps on multiple datasets.

T=4,VGG-9,$\alpha=$	1	2	3	4	$\beta=0.25$	0.5	0.75	1.0	1.25	1.5	$\gamma=0.25$	0.5	0.75	1.0	1.25	1.5
CIFAR10-DVS	76.2	77.1	76.9	77.1	76.0	76.6	**77.3**	77.1	76.5	76.8	76.1	75.8	76.2	77.1	75.6	76.4
N-Caltech101	82.39	**83.92**	82.49	82.82	82.71	82.60	82.60	**83.92**	83.15	82.82	83.15	83.26	82.71	**83.92**	82.71	82.28
DVS-Gesture	93.40	94.44	**95.14**	94.10	93.75	95.04	94.10	94.44	93.75	93.40	94.10	94.10	93.06	94.44	93.75	93.06
T=16,QKFormer,$\alpha=$	1	2	3	4	$\beta=0.25$	0.5	0.75	1.0	1.25	1.5	$\gamma=0.25$	0.5	0.75	1.0	1.25	1.5
DVS-Gesture	98.26	**98.61**	98.26	98.26	98.26	98.26	98.61	98.61	98.26	98.26	98.26	98.26	98.26	98.61	98.26	98.61

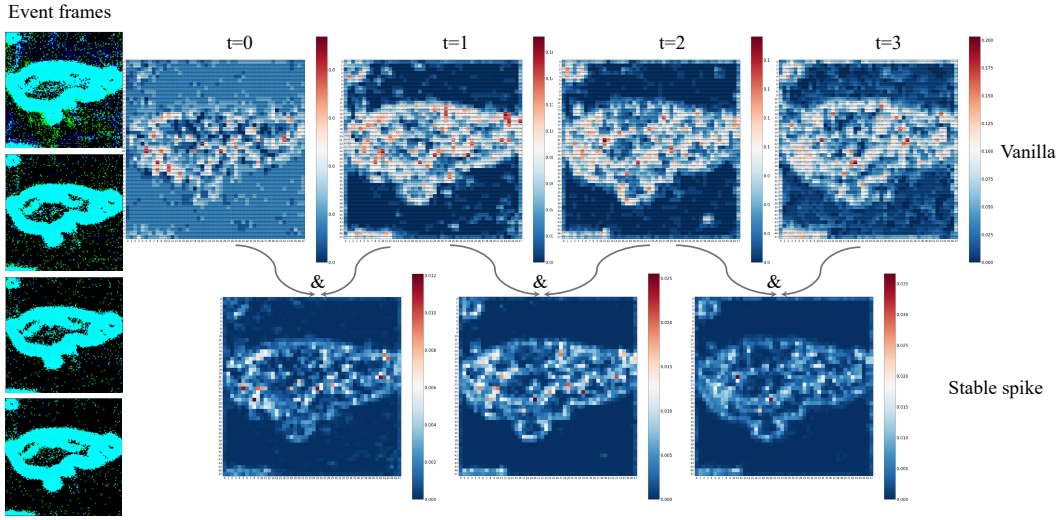

Figure 1. Visualization of the second layer spike maps of VGG-9 on CIFAR10-DVS. The stable spike maps decoupled by the minimal & operation precisely depict the feature skeleton, echoing the results of Fig. 1 in the main paper. To ensure the visualization effect, the average of the spike maps of all channels is displayed.

Table 8. Optimizing consistency across all timesteps does not eliminate useful temporal information.

Consistency	CIFAR10-DVS	DVS-Gesture
All timestep	**77.1**	**94.44**
First two timesteps	75.2	91.67
Last two timesteps	75.9	93.06

ence, our method is the same as the vanilla SNN, so it does not affect inference efficiency at all. Overall, our method significantly improves the inference performance of SNNs with negligible training overhead.

E. Additional Visualizations

In this section, we present additional spike map visualizations to demonstrate the consistency of "the vanilla spike map differences across timesteps and the ability of the stable spike to accurately represent the feature skeleton" across multiple datasets and layers. Fig. 1 in the main paper shows the spike map of the first layer of the VGG-9 architecture on CIFAR10-DVS. Fig. 1 shows the spike map of the second layer. Fig. 2 and Fig. 3 show the spike maps of the first two layers on DVS-Gesture, respectively.

References

[1] Arnon Amir et al. A low power, fully event-based gesture recognition system. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7388–7397, 2017. 1

[2] Srinivas Anumasa, Bhaskar Mukhoty, Velibor Bojkovic, Giulia De Masi, Huan Xiong, and Bin Gu. Enhancing training of spiking neural network with stochastic latency. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 10900–10908, 2024. 2, 4

[3] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation

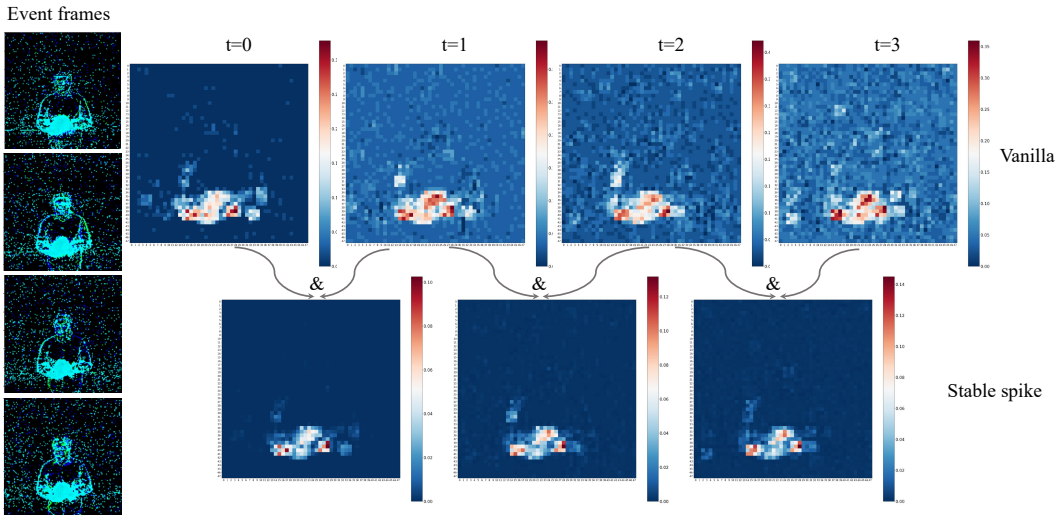

Figure 2. Visualization of the first layer spike maps of VGG-9 on DVS-Gesture. The stable spike maps decoupled by the minimal & operation precisely depict the feature skeleton, echoing the results of Fig.1 in the main paper. To ensure the visualization effect, the average of the spike maps of all channels is displayed.

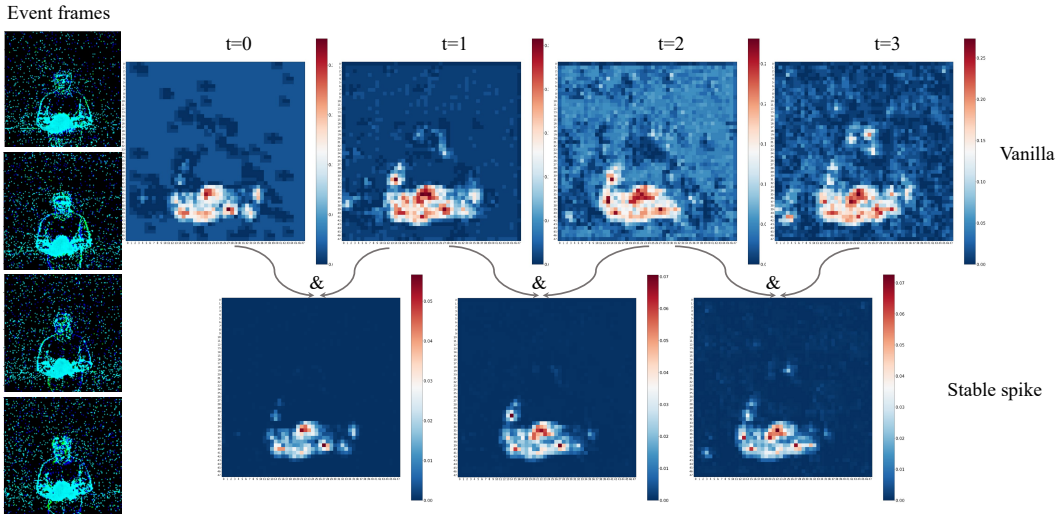

Figure 3. Visualization of the second layer spike maps of VGG-9 on DVS-Gesture. The stable spike maps decoupled by the minimal & operation precisely depict the feature skeleton, echoing the results of Fig.1 in the main paper. To ensure the visualization effect, the average of the spike maps of all channels is displayed.

strategies from data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 113–123, 2019. 2

[4] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 2

[5] Yongqi Ding, Lin Zuo, Mengmeng Jing, Pei He, and Hanpu Deng. Rethinking spiking neural networks from an ensemble learning perspective. In *The Thirteenth International Conference on Learning Representations*, 2025. 1, 2, 3

[6] Yongqi Ding, Lin Zuo, Mengmeng Jing, Kunshan Yang, Pei He, and Tonglan Xie. Synergy between the strong and the weak: Spiking neural networks are inherently self-distillers. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. 4

[7] Yufei Guo, Yuhan Zhang, Zhou Jie, Xiaode Liu, Xin Tong, Yuanpei Chen, Weihang Peng, and Zhe Ma. Reverb-SNN: Reversing bit of the weight and activation for spiking neural networks. In *Forty-second International Conference on Machine Learning*, 2025. 4

[8] Xiang He, Dongcheng Zhao, Yang Li, Guobin Shen, Qingqun Kong, and Yi Zeng. An efficient knowledge trans-

fer strategy for spiking neural networks from static to event domain. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 512–520, 2024. 2

[9] Yulong Huang, Xiaopeng LIN, Hongwei Ren, Haotian FU, Yue Zhou, Zunchang LIU, biao pan, and Bojun Cheng. CLIF: Complementary leaky integrate-and-fire neuron for spiking neural networks. In *Forty-first International Conference on Machine Learning*, 2024. 2

[10] Haiyan Jiang, Vincent Zoonekynd, Giulia De Masi, Bin Gu, and Huan Xiong. TAB: Temporal accumulated batch normalization in spiking neural networks. In *The Twelfth International Conference on Learning Representations*, 2024. 2

[11] Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: An event-stream dataset for object classification. *Frontiers in Neuroscience*, 11, 2017. 1

[12] Shuang Lian, Jiangrong Shen, Qianhui Liu, Ziming Wang, Rui Yan, and Huajin Tang. Learnable surrogate gradient for direct training spiking neural networks. In *IJCAI*, pages 3002–3010, 2023. 4

[13] Wei Liu, Li Yang, Mingxuan Zhao, Shuxun Wang, Jin Gao, Wenjuan Li, Bing Li, and Weiming Hu. DeepTAGE: Deep temporal-aligned gradient enhancement for optimizing spiking neural networks. In *The Thirteenth International Conference on Learning Representations*, 2025. 4

[14] Garrick Orchard, Ajinkya Jayawant, Gregory K. Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in Neuroscience*, 9, 2015. 1

[15] Shuai Wang, Malu Zhang, Dehao Zhang, Ammar Belatreche, Yichen Xiao, Yu Liang, Yimeng Shan, Qian Sun, Enqi Zhang, and Yang Yang. Spiking vision transformer with saccadic attention. In *The Thirteenth International Conference on Learning Representations*, 2025. 4

[16] Qi Xu, Yaxin Li, Jiangrong Shen, Jian K. Liu, Huajin Tang, and Gang Pan. Constructing deep spiking neural networks from artificial neural networks with knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7886–7895, 2023. 4

[17] Jiaqi Yan, Changping Wang, De Ma, Huajin Tang, Qian Zheng, and Gang Pan. Training high performance spiking neural network by temporal model calibration. In *Forty-second International Conference on Machine Learning*, 2025. 4

[18] Chengting Yu, Lei Liu, Gaoang Wang, Erping Li, and Aili Wang. Advancing training efficiency of deep spiking neural networks through rate-based backpropagation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 4

[19] Kairong Yu, Tianqing Zhang, Qi Xu, Gang Pan, and Hongwei Wang. TS-SNN: Temporal shift module for spiking neural networks. In *Forty-second International Conference on Machine Learning*, 2025. 4

[20] Tianqing Zhang, Kairong Yu, Xian Zhong, Hongwei Wang, Qi Xu, and Qiang Zhang. Staa-snn: Spatial-temporal attention aggregator for spiking neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13959–13969, 2025. 4

[21] Chenlin Zhou, Han Zhang, Zhaokun Zhou, Liutao Yu, Liwei Huang, Xiaopeng Fan, Li Yuan, Zhengyu Ma, Huihui Zhou, and Yonghong Tian. QKFormer: Hierarchical spiking transformer using q-k attention. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 1, 2, 4