ELASTIC PREDICTION FROM THE FIRST TIME-STEP: SPIKING NEURAL NETWORKS WITH TEMPORAL MEMBRANE BIASING

Anonymous authors
Paper under double-blind review



(a) Baseline (b) Ours

Figure 1: **Comparison between ours and ANN-to-SNN conversion baseline.** Our method detects and segments objects at earlier time-steps than the baseline with a higher confidence score.

ABSTRACT

Spiking neural networks (SNNs) naturally support elastic inference, where salient inputs trigger early predictions, and additional time-steps progressively refine the output, enabling fast responses and improved accuracy without repeated inference or model retraining. However, existing SNN methods are not well-suited for elastic inference: ANN-to-SNN conversion methods suffer from poor early-time-step accuracy, whereas directly trained SNNs perform well at specific time-steps but often degrade when inference continues, limiting the accuracy refinement.

In this work, we identify the root cause of the early-stage performance degradation as a *membrane-threshold mismatch* between membrane potentials and firing thresholds. To mitigate this mismatch, we propose a temporal membrane biasing (TMB) method that applies sustained calibration across time-steps while preserving Quantized Artificial Neural Network (QANN) equivalence. In addition, we develop a bias-only fine-tuning strategy that enables efficient optimization with minimal training cost. Across multiple tasks, our work markedly improves low time-step accuracy (*e.g.*, 70.15% top-1 accuracy improvement at the first time-step on ImageNet). Combining elastic inference and adaptive methods, TMB achieves 24.3% operation and 24.8% time-step reduction with $< 0.1\%$ accuracy degradation compared to the same SNN without TMB.

1 INTRODUCTION

SNNs (Tal and Schwartz, 1997) attract widespread attention due to their potential to build extremely efficient intelligent systems, along with other merits observed in human brains (Roy et al., 2019). In contrast to traditional artificial neural networks (ANNs), SNN outputs emerge progressively and form in a temporally evolving fashion. Taking object detection as an example, as shown in Figure 1 (right), centrally located and nearby vehicles are recognized earlier at the first time-step, whereas distant ones require additional time-steps, with the segmentation gradually becoming more precise.

Consequently, SNNs possess a unique inference paradigm, named *elastic inference*, which mirrors the early decision-making in biological neural systems (Thorpe et al., 1996), where *salient stimuli trigger faster neural responses*. Elastic Inference has two significant advantages: 1) With adaptive inference methods such as SEENN (Li et al., 2023) and OEDSNN (Wu et al., 2025), SNNs can early exits at intermediate time-steps, improving the accuracy–operation trade-off by avoiding redundant temporal

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

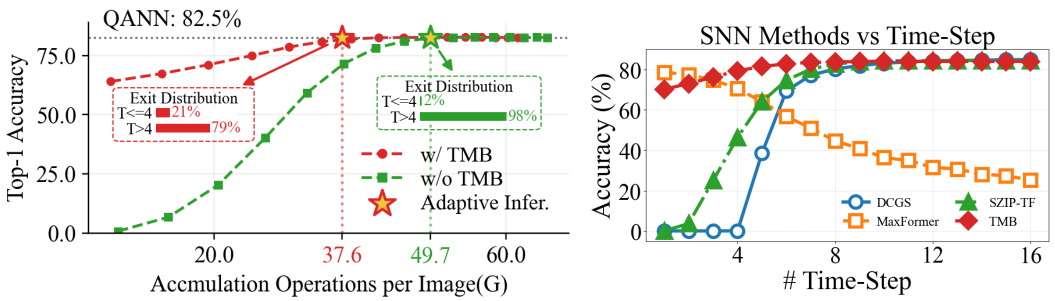


Figure 2: **ImageNet-1K Accuracy–efficiency trade-offs.** Left: Spike-ViT-B accuracy-sop curve. “Star" denotes the trade-off sweet point with adaptive inference SEENN (Li et al., 2023). Right: Comparison between TMB (Spike-ViT-L) and other SNNs.

computation while maintaining prediction accuracy. 2) Elastic inference enables backend systems to make decisions at early time-steps based on front-end perception outputs, which is beneficial for latency-critical applications such as autonomous driving (Qian et al., 2022).

Therefore, **we desire an elastic-inference-enabled SNN that can produce higher-quality outputs in the early or even the first time-step, while not hampering the terminal outputs at late time-step**. The requirement facilitates a better accuracy-operation trade-off and earlier exiting for salient sample. As shown in Figure 2(left), combining TMB with adaptive inference SEENN increases the proportion of samples exiting before the 4^{th} time-step from 2% to 20%, reducing the number of accumulation operations by 24.3% (from 49.7G to 37.6G) compared with SNNs without TMB, while maintaining accuracy with less than 0.1% degradation.

However, existing SNN methods do not adequately satisfy this requirement. As shown in Figure 2(right), prior ANN-to-SNN (A2S) conversion approaches (DCGS (Huang et al., 2025) and SpikeZIP-TF (You et al., 2024)), exhibit poor accuracy at the first time-step, thereby increasing inference latency. In contrast, direct training methods MaxFormer (Fang et al., 2026), achieve the highest accuracy at the target time-step, but their performance degrades sharply at subsequent time-steps.

To exploit elastic inference in SNNs, we start from ANN-to-SNN (A2S) conversion and observe a pronounced representation degradation between the ANN and the converted SNN at early time-steps, which leads to significant accuracy loss. We identify the root of the phenomenon as the “membrane-threshold mismatch" between membrane potential and firing threshold. To address this, we propose TMB, which introduces a learnable bias to calibrate membrane potentials. Unlike prior membrane adjustment methods (Bu et al., 2022; Li et al., 2021a) that rely on manually setting the initial membrane state, TMB performs sustained calibration across time-steps via back-propagation-through-time (BPTT). Our contributions are as follows:

- For the first time, we define and systematically analyze the “*membrane-threshold mismatch*" in SNN inference systems, which hampers the usefulness of elastic inference.
- To address this mismatch, we propose temporal membrane biasing, which boosts ultra-early-step accuracy while preserving equivalence to QANN. On ImageNet, TMB improves the 1^{st}-step accuracy by 70.15% over the SOTA A2S method DCGS (Huang et al., 2025) on ViT-Large.
- To optimize the bias, a bias-only fine-tuning training algorithm is proposed with minimal training cost. In ViT-Small, the training method achieves 73.7× learnable parameter reduction and 3.0× speed-up compared to full-parameter training.

2 Background

2.1 Basics of Spiking Neural Networks

SNN contains spiking neurons and synaptic weights, and processes spikes with an RNN-style procedure. As shown in Figure 3, the input image is encoded into binary spike trains and fed into the network at each time-step ($T = 1, 2, 3$). Similar to Recurrent Neural Networks (RNNs), the SNN is unfolded over time and maintains internal neural states (*e.g.*, membrane potential) across time-steps. Finally, output spikes are accumulated for the final output prediction (car).

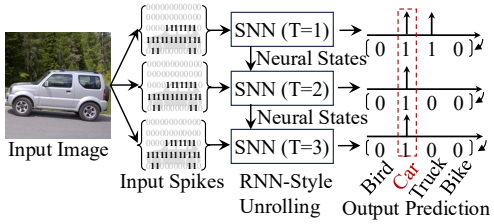


Figure 3: **Inference process in SNNs.**

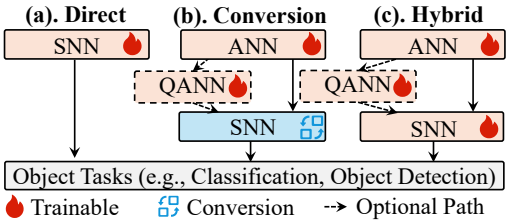


Figure 4: **Synaptic weight learning methods.**

Spiking Neuron is a unit whose inputs and outputs are encoded as a spike train. To overcome the low accuracy issue of SNN, a new spiking neuron called *bipolar integrate&fire with spike tracer* (*aka.* ST-BIF) is recently proposed in (You et al., 2024; 2025). For the i-th neuron at l-th layer, its input and output at time-step t is $s_{i,t}^{\text{in},l}, s_{i,t}^{\text{out},l} \in \{-1, 0, 1\}$, and its pre-synaptic weight is w_i. Each neuron contains a memory unit called *membrane*, modeled as a capacitor with a capacitance of C.

At each time-step t, neuron recursively conducts three successive steps: ❶ *spikes integration*, ❷ *neuron firing* and ❸ *membrane update*. In step-❶, on arrival of pre-synaptic spikes, the membrane integrates them up and updates the membrane potential V_t, like a capacitor charging process:

$$\hat{V}_t = V_{t-1} + (\sum\nolimits_i w_i \cdot s_{i,t}^{\text{in},l})/C \tag{1}$$

where $\hat{V}_t$ is the membrane potential after integration but before the following membrane update in step-❸. In step-❷, the neuron uses a decision function Θ to fire ternary spikes, which is:

$$\Theta(\hat{V}_t, V_{\text{thr}}, M_{t-1}) = \begin{cases} 1; & \hat{V}_t \geq V_{\text{thr}} \ \& \ M_{t-1} < M_{\max} \\ 0; & \text{other} \\ -1; & \hat{V}_t < 0 \ \& \ M_{t-1} > M_{\min} \end{cases}, \ s_t^{\text{out},l} = \Theta(\hat{V}_t, V_{\text{thr}}, M_{t-1}) \times q_{\text{thr}} \tag{2}$$

where M_t is an additional memory unit in ST-BIF neuron (*aka.*, spike tracer), and $M_{\max}, M_{\min}$ are the maximum and minimum values of the spike tracer respectively. By default, we set $V_{\text{thr}} = q_{\text{thr}}/C, M_0 = 0, V_0 = (0.5q_{\text{thr}} + b)/C$, where b is the bias value in the convolution/linear layer. The spike tracer records the firing history of the neuron, which transfers the process in step-❸ into:

$$V_t = \hat{V}_t - s_t^{\text{out}}/C; \quad M_t = M_{t-1} + \Theta(\hat{V}_t, V_{\text{thr}}, M_{t-1}) \tag{3}$$

Synaptic Weight Learning Method. Existing methods to learn the synaptic weights can be categorized into threefold, as shown in Figure 4, *i.e.*, direct training (DT), ANN-to-SNN conversion (A2S), and hybrid training (HT). Figure 4 summarizes the training procedure of these methods. DT methods (Yao et al., 2024; Zhou et al., 2024) direct train an SNN from scratch with the back-propagation-through-time (*aka.* BPTT) algorithm at a small time-step (*e.g.*, 1 or 4 time-steps). A2S methods (You et al., 2024; Bu et al., 2023) transfer the parameters of the pre-trained ANN into its SNN counterpart. HT methods apply A2S first, then fine-tune the converted SNN via DT, to achieve high accuracy with fewer #time-steps (Rathi and Roy, 2020; Xia et al., 2026).

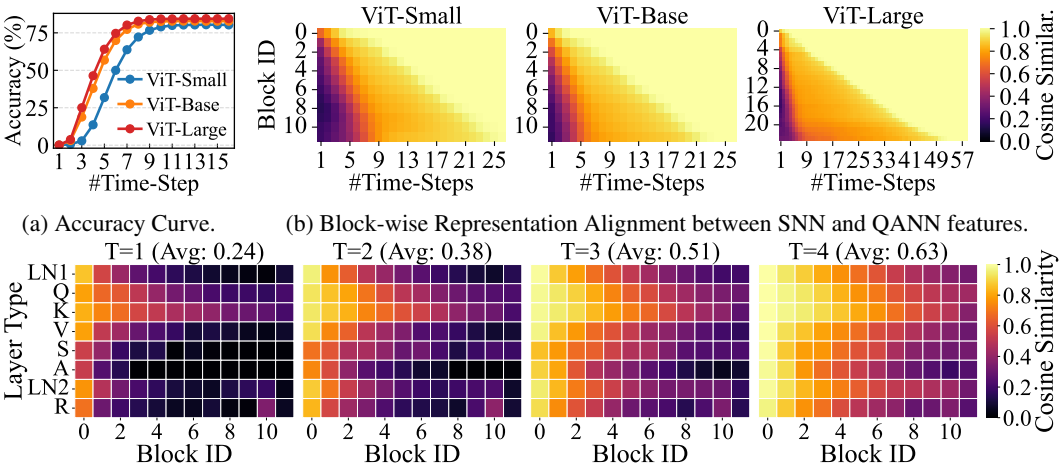
2.2 QANN EQUIVALENCE IN SNN

With ST-BIF neuron (Section 2.1), SNN is proven to be equivalent to the QANN (Hu et al., 2023). This equivalence has been formally proven in (Yan et al., 2026) as follows:

Lemma 2.1 (QANN equivalence for an SNN). *For an L_0-layer feed-forward ST-BIF neural network with weights $\{\mathbf{W}_\theta{}^l\}_{l=0}^{L_0-1}$ and biases $\{\mathbf{b}_\theta{}^l\}_{l=0}^{L_0-1}$, the temporal accumulation of the output spikes at equilibrium (after T_{eq}), denoted by $\sum_{t=0}^{T_{\text{eq}}} \boldsymbol{s}_t^{\text{out},L_0} = \sum_{t=0}^{T_{\text{eq}}} [s_{i,t}^{\text{out},L_0}]_{i=1}^{N_1}$, are:*

$$\sum\nolimits_{t=0}^{T_{\text{eq}}} \boldsymbol{s}_t^{\text{out},L_0} = \mathbf{F}_{\text{Q}}(\mathbf{Q}_{\text{total}}; \{\mathbf{W}_\theta{}^l\}_{l=0}^{L_0-1}; \{\mathbf{b}_\theta{}^l\}_{l=0}^{L_0-1}) \tag{4}$$

where the total input charges are $\mathbf{Q}_{\text{total}} = [\sum_{t=0}^{T_{\text{eq}}} s_{j,t}^{\text{in},1}]_{j=1}^{N_2}$, N_1, N_2 is number of output and input neuron, $\mathbf{F}_{\text{Q}}$ is the corresponding ANN with quantized activation functions, and T_{eq} is the time-step when SNN enters "equilibrium status" with no further activity of neuron firing and membrane update.



(a) Accuracy Curve. (b) Block-wise Representation Alignment between SNN and QANN features.

(c) Layer-wise Representation Alignment $\mathcal{I}$ in ViT-Base at the first 4 time-steps.

Figure 5: **Observations of representation degradation in SNNs.** SNN is obtained by (You et al., 2024). (a) Accuracy versus time-steps in SNNs; (b) Alignment decreases rapidly with increasing depth at early time-steps. (c) Alignment degradation is pervasive across all layers. "LN", "S", "R", and "A" are outputs of LayerNorm, Softmax, ReLU, and Attention.

Theorem 2.1 indicates that for an SNN comprising ST-BIF neurons, the temporal accumulation of output spikes $\sum_{t=0}^{T_{\text{eq}}} \boldsymbol{s}_t^{\text{out}}$ is equal to the QANN output. *Importantly, the equivalence is independent of the input order of spike trains if the total input charges ($\mathbf{Q}_{\text{total}}$) and the input bias ($\mathbf{b}_\theta$) are constants.*

3 Membrane-Threshold Mismatch

As shown in Figure 5a, ViT-Small, ViT-Base, and ViT-Large that learned by A2S method (You et al., 2024) achieve negligible accuracy of 0.13%, 0.11%, 0.09% at the first time-step. The low accuracy is caused by a *membrane-threshold mismatch* phenomenon in the early stage of SNN inference.

3.1 Definition of Representation Alignment Measured by Cosine Similarity

To quantify representation degradation from membrane-threshold mismatch, we measure the alignment $\mathcal{I}$ between the instantaneous output at the τ-th time-step ($\sum_{t=1}^{\tau} \boldsymbol{S}_t^l$) and the converged output at T_{eq}-th time-step ($\sum_{t=1}^{T_{\text{eq}}} \boldsymbol{S}_t^l$) that equals the QANN output (Theorem 2.1) at the l^{th} layer:

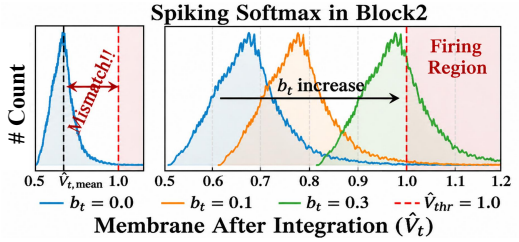
$$\mathcal{I}(l,\tau) = \rho\left(\sum_{t=1}^{T_{\text{eq}}} \boldsymbol{S}_t^l, \sum_{t=1}^{\tau} \boldsymbol{S}^l(\tau)\right) \tag{5}$$

where ρ is an alignment metric that quantifies the representational consistency, which is required to: 1) be invariant to isotropic scaling, *i.e.*, $s(X,Y) = s(\alpha X, \beta Y)$ for any $\alpha, \beta \in R^+$, since $\sum_{\tau=1}^{T_{\text{eq}}} \boldsymbol{S}^l(\tau)$ and $\sum_{\tau=1}^{T} \boldsymbol{S}^l(\tau)$ naturally differ in magnitude due to $T \leq T_{\text{eq}}$; 2) has a strict element-wise correspondence, *e.g.*, $s(X,Y) \neq s(X^T,Y)$, since there exists a strict one-to-one mapping between $\sum_{\tau=1}^{T_{\text{eq}}} \boldsymbol{S}^l(\tau)$ and $\sum_{\tau=1}^{T} \boldsymbol{S}^l(\tau)$. **In this work, we employ cosine similarity (Xia et al., 2015) as the metric ρ, as it naturally satisfies the aforementioned properties.**

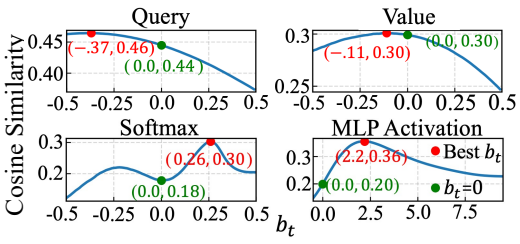
3.2 Observations: Representation Degradation

Observation1: $\mathcal{I}(l,t)$ decreases rapidly with increasing depth at early time-steps. As shown in Figure 5b, the block-wise representation alignment $\mathcal{I}(l,t)$ drops in the bottom-left region, corresponding to large block depth and small inference time-step.

Observation2: $\mathcal{I}(l,t)$ degradation is pervasive across all layers. Figure 5c shows the layer-wise representation alignment $\mathcal{I}(l,t)$. The heatmap exhibits a consistent darkening trend from left to right, suggesting that the degradation is systemic across the entire network.



(a) **Membrane distribution of spiking softmax** in the 2^{nd} ViT-Base block at the 1^{st} time-step.



(b) **Representation Alignment** $\mathcal{I}(l,t)$ in the 2^{nd} block of ViT-Base across different bias b_t.

Figure 6: **Analysis of Membrane-Threshold Mismatch**. (a). By increasing the injected bias b_t, we can shift the overall distribution of membrane potential $\hat{V}_t$ horizontally. (b). $\mathcal{I}(l,t)$ differs significantly between the default (green dot) and optimal biased (red dot) membrane distributions.

3.3 Analysis: Membrane-Threshold Mismatch

To further investigate the reason for the representation degradation, we plot the distribution of the neural membrane $\hat{V}_t$ after integration but before firing in Figure 6a. There exists a severe mismatch between the membrane distribution (the mean of the membrane distribution $\hat{V}_{t,\text{mean}}$) and firing threshold V_{thr}. Such a mismatch inhibits the firing activity of a majority of neurons, causing a misalignment between the actual output representation and the target representation.

To align $\hat{V}_t$ with the firing threshold, we introduce a bias term b_t and add it to the membrane $\tilde{V}_t = \hat{V}_t + b_t$. By adjusting b_t, we can shift the overall distribution of $\hat{V}_t$ horizontally (from the blue curve to the green curve in the right subplot of Figure 6a). With the distribution shifting, we can adjust the neural firing pattern, making the output representation more consistent with the target.

Figure 6b shows the evolution of representation alignment $\mathcal{I}(l,t)$ with different bias b_t. The default membrane distribution (green point) differs from the optimal distribution (red point) that achieves the highest alignment $\mathcal{I}(l,t)$. For instance, in MLP activation layers, the optimal $b_t = 2.2$, increasing the $\mathcal{I}(l,t)$ from 0.2 to 0.36. This suboptimal membrane distribution is pervasive across SNN layers, motivating the optimization of b_t to mitigate this issue.

4 Methodology

In this work, we adopt two guidelines to design a method that mitigates mismatch: 1) **Prediction Invariance:** The method should ensure that the final prediction of SNN is unchanged. 2) **Minimal Overhead:** The method should incur negligible costs during both training and inference.

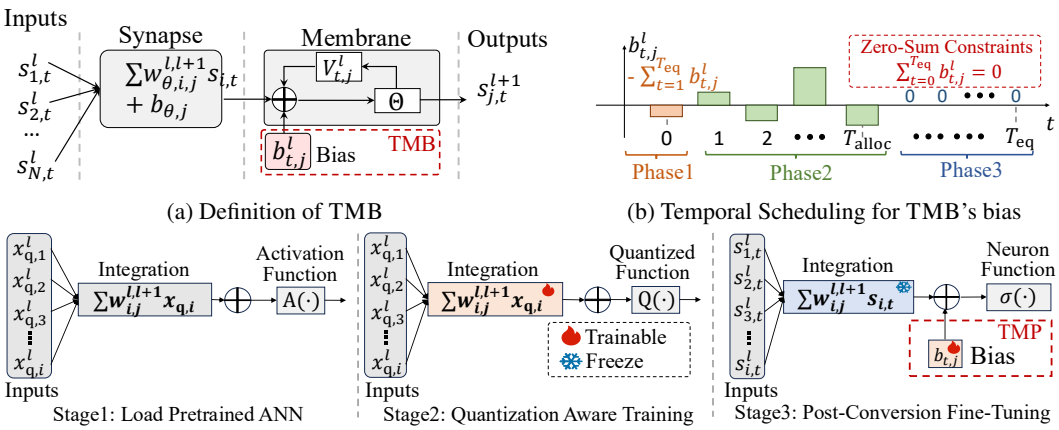
4.1 Temporal Membrane Biasing

To solve the membrane-threshold mismatch problem, we propose a *temporal membrane biasing* (TMB) that adds a sustained bias to the membrane potential at each time-step.

Definition of TMB. As shown in Figure 7a, TMB modulates the membrane potential by adding a bias $b^l_{t,j}$. Formally, considering the j^{th} spiking neuron at the $(l+1)^{\text{th}}$ layer with N pre-synaptic spiking neurons at l^{th} layer, TMB is defined below:

$$s^{\text{in},l+1}_{t,j} = \sigma\left(\underbrace{\sum\nolimits_{i=0}^{N} w^{l,l+1}_{\theta,i,j}\, s^{\text{in},l}_{t,i} + b^l_{\theta,j}}_{\text{integration}} + \underbrace{b^l_{t,j}}_{\text{TMB}}\right) \quad \text{(neuron function)} \tag{6}$$

where $\sigma(\cdot)$ is the ST-BIF neuron function, and $w^{l,l+1}_{\theta,i,j}$ is the synaptic weight connecting from i^{th} neuron in l^{th} layer to j^{th} neuron in $(l+1)^{\text{th}}$ layer. $b^l_{t,j}$ comprises two dimensions: a channel-wise dimension j at layer l and a temporal dimension t, enabling the membrane calibration for each channel of SNN layers at every time-step.



(a) Definition of TMB (b) Temporal Scheduling for TMB's bias

(c) Three-stage A2S Conversion Procedure with TMB.

Figure 7: **TMB method overview.** TMB inserts a spatio-temporal bias into the membrane of a spiking neuron and performs temporal scheduling of the bias to enforce the zero-sum constraint.

Temporal Scheduling for TMB. To preserve the SNN's prediction consistency after inserting TMB's bias, we perform a temporal scheduling for the bias, as shown in Figure 7b. We perform a three-phase scheduling: *Phase 1: Compensation.* A compensatory bias is applied. In the standard setting, this is defined as the negative sum of all biases used in Phase 2, *i.e.*, $-\sum_{t=2}^{T_{\text{alloc}}} b_{t,j}^l$. *Phase 2: Injection.* A learnable bias is injected to calibrate the membrane potential before T_{alloc}. *Phase 3: Resting.* No bias is injected (*i.e.*, zero bias). With temporal scheduling, the TMB's bias satisfies the zero-sum constraints $\sum_{t=0}^{T_{\text{eq}}} b_{t,j}^l = 0$, and the output prediction remains invariant:

Theorem 4.1 (Prediction Invariance under TMB)**.** *For an L_0-layer feed-forward ST-BIF neural network with weights $\{\mathbf{W}_\theta{}^l\}_{l=0}^{L_0-1}$, biases $\{\mathbf{b}_\theta{}^l\}_{l=0}^{L_0-1}$, and TMB bias $\{\mathbf{b}_t^l\}_{l=0}^{L_0-1}$, the temporal accumulation of the output spikes at equilibrium, denoted by $\sum_{t=0}^{T_{\text{eq}}} \hat{\boldsymbol{s}}_t^{\text{out},L_0}$, remains unchanged:*

$$\begin{aligned} &\sum_{t=0}^{T_{\text{eq}}} \hat{\boldsymbol{s}}_t^{\text{out},L_0} = \sum_{t=0}^{T_{\text{eq}}} \boldsymbol{s}_t^{\text{out},L_0} = \mathbf{F}_{\text{Q}}(\mathbf{Q}_{\text{total}}; \{\mathbf{W}_\theta{}^l\}_{l=0}^{L_0-1}; \{\mathbf{b}_\theta{}^l + \sum_{t=0}^{T_{\text{eq}}} \mathbf{b_t}{}^l\}_{l=0}^{L_0-1}) \\ &s.t., \quad \{\sum_{t=0}^{T_{\text{eq}}} \boldsymbol{b}_t^l\}_{l=0}^{L_0-1} = \{\mathbf{0}\}_{l=0}^{L_0-1} \end{aligned} \tag{7}$$

where $\boldsymbol{s}_t^{\text{out}}$ is the accumulated output of SNN sharing the same weights and biases without using TMB bias, l is the l^{th} layer, and t is the t^{th} time-step.

Theorem 4.1 shows that with the zero-sum constraint $\sum_{t=0}^{T_{\text{eq}}} b_{t,j}^l = 0$, the SNN's final output prediction with TMB is identical to that of the original SNN without TMB. Detailed proof is in the Appendix.

Optimization of TMB To find an optimal bias for TMB, we use the BPTT algorithm with a loss function to train the bias. Formally, for an L_0-layer feed-forward ST-BIF neural network F_{SNN} with weights $\{\mathbf{W}_\theta^l\}$ and biases $\{\mathbf{b}_\theta^l\}$, the bias sequence $\{\boldsymbol{b}_t^l\}$ of TMB is optimized as follows:

$$\begin{aligned} \min_{\{\boldsymbol{b}_t^l\}} \quad & \mathbb{E}_{(x_t,y)\sim\mathcal{D}}\big[\mathcal{L}\big(y, \sum_{t=0}^{T_{\text{tar}}} \mathbf{F}_{\text{SNN}}(x_t; \{\mathbf{W}_\theta^l\}; \{\boldsymbol{b}_\theta^l\} + \{\boldsymbol{b}_t^l\})\big)\big] \\ \text{s.t.} \quad & \forall\, l \in \{0, \ldots, L_0 - 1\}, \qquad \sum_{t=0}^{T_{\text{eq}}} \boldsymbol{b}_t^l = \mathbf{0}, \end{aligned} \tag{8}$$

where $\mathcal{L}$ denotes the task loss function (*e.g.*, cross-entropy for classification), T_{tar} is the target training time-step, and $(x_t, y) \sim \mathcal{D}$ represents an input–target pair (x_t, y) at time-step t sampled from the data distribution $\mathcal{D}$. In Equation (8), we only modify the TMB's bias $\{\boldsymbol{b}_t^l\}$ and freeze other network parameters (*i.e.* weights $\{\mathbf{W}_\theta^l\}$ and biases $\{\mathbf{b}_\theta^l\}$) to reduce the training cost.

4.2 Training Method in TMB

TMB obtains an SNN via a three-stage framework shown in Figure 7c. We initialize the model with open-source pretrained ANN weights (Stage-1). Then, we insert quantization functions and employ quantization-aware training (QAT) to derive a high-precision QANN (Stage-2). In this stage, no additional bias is inserted; therefore, TMB *introduces no additional learnable parameters*. Finally,

Table 1: **Comparison between TMB and previous A2S works on the ImageNet dataset.** n is the number of thresholds in a spiking neuron. *: results reproduced with the official repository. "ViT-S-10Level" means the quantized ViT-Small with 10 quantization levels.

Method	Type	Arch.	Param. (M)	(Q)ANN Acc(%)	Feature Type	SNN Accuracy(%) at T Time-Step					
						T	Acc	T	Acc	T	Acc
TS (Deng and Gu, 2021)	CNN-to-SNN	VGG-16	138	72.40	Binary	-	-	-	-	64	70.97
SNM (Wang et al., 2022)	CNN-to-SNN	VGG-16	138	73.18	Binary	-	-	-	-	64	71.50
MMSE 2021b	CNN-to-SNN	ResNet-34	21.8	75.66	Binary	-	-	-	-	64	71.12
		VGG-16	138	75.36	Binary	-	-	-	-	64	70.69
QCFS 2023	CNN-to-SNN	ResNet-34	21.8	74.32	Binary	-	-	-	-	64	72.35
		VGG-16	138	74.29	Binary	-	-	-	-	64	72.85
SRP 2023	CNN-to-SNN	ResNet-34	21.8	74.32	Binary	-	-	4	66.71	64	68.61
		VGG-16	138	74.29	Binary	-	-	4	66.47	64	69.43
MST 2023	Transformer-to-SNN	Swin-T(BN)	28.5	80.51	Binary	-	-	128	77.88	512	78.51
STA 2024	Transformer-to-SNN	ViT-B/32	86	83.60	Binary	-	-	32	78.72	256	82.79
SZIP-TF* 2024	Transformer-to-SNN	ViT-S-10Level	22.1	80.10	Ternary	1	0.13	8	72.22	16	80.25
		ViT-B-10Level	86.6	82.84		1	0.11	8	80.82	16	82.84
		ViT-L-10Level	304.3	84.05		1	0.09	8	82.78	16	84.42
ECMT* 2024	Transformer-to-SNN	ViT-S-5	22.1	81.38	11-Value ($n = 10$)	1	0.10	8	53.95	16	78.30
		ViT-B-5	86.6	84.54		1	0.10	8	19.83	16	75.99
		ViT-L-5	304.3	85.84		1	0.10	8	74.50	16	84.01
DCGS* 2025	Transformer-to-SNN	ViT-S-5/4	22.1	81.38	11-Value ($n = 10$)	1	0.10	8	63.71	16	79.46
		ViT-B-5/4	86.6	84.54		1	0.10	8	54.30	16	83.13
		ViT-L-5/4	304.3	85.84		1	0.10	8	80.40	16	84.93
TMB (Our)	Transformer-to-SNN	ViT-S-10Level	22.4	80.10	Ternary	1	**40.33**	8	**79.21**	16	**80.46**
		ViT-B-10Level	87.1	82.50		1	**63.79**	8	**82.60**	16	82.50
		ViT-B-18Level	87.1	84.49		1	22.91	8	73.85	16	**84.49**
		ViT-L-10Level	305.6	84.05		1	**70.25**	8	**83.76**	16	84.04

we convert the QANN into an SNN and freeze the weights w_θ and bias b_θ (highlighted in blue) and only update the TMB's bias $b_{t,j}$ (highlighted in orange). The definition of surrogate gradient of the non-differentiable spiking dynamic (Θ in Equation (2)) is introduced in Appendix.

5 Experiment

5.1 Experiment Setup

Benchmarks: 1) Classification dataset: ImageNet (Deng et al., 2009). 2) Neuromorphic vision datasets: CIFAR10-DVS (Hongmin et al., 2017) and HAR-DVS (Wang et al., 2024). 3) Detection dataset: COCO datasets (Lin et al., 2014). **Networks:** 1) transformer-based: vision transformers (Dosovitskiy, 2020) and swin transformer (Liu et al., 2021) for image classification and neuromorphic vision tasks. Mask-RCNN architecture proposed by ViTDet (Li et al., 2022) for the detection task. 2) CNN-based: ResNet50 (He et al., 2015). 3) Hybrid-based: CvT-13 (Wu et al., 2021). To make a fair comparison, we create a baseline that shares the architecture of TMB without TMB. **Energy Estimation:** We follow the previous works (Zhou et al., 2023; 2024) that count the #synaptic operation (#SOP) and # multiply accumulate operation (#MAC) to estimate the inference energy. **Training Setting:** In default, we only train the loss $\mathcal{L}$ between target and SNN output at the first time-step $T_{\text{tar}} = 1$ in Equation (8), to minimize the training cost.

5.2 Results Comparison

Comparison on ImageNet of TMB and previous state-of-the-art (SOTA) conversion methods are shown in Table 1. TMB shows a significant accuracy improvement at extremely low time-steps. For instance, TMB surpasses SOTA method DCGS with 10 threshold (5 positive and 5 negative) (Huang et al., 2025) 40.23%, 63.69%, and 70.15% at 1^{st} time-step and 15.5%, 28.30%, and 3.36% at 8^{th} time-step with only ternary activation. The substantial improvement stems from the success of threshold-membrane mismatch alignment in early-stage SNN inference. We also train ViT-B-18level to achieve a comparable accuracy (84.49%) to DCGS (84.54%) for a fair comparison. TMB outperforms DCGS across all time-steps in the ViT-base model, achieving higher accuracy.

Comparison on neuromorphic datasets are reported in Table 2 and Table 3. TMB achieves SOTA accuracy on both the CIFAR10-DVS and HAR-DVS datasets. Specifically, on CIFAR10-DVS, TMB

Table 2: **Comparison on CIFAR10-DVS.**

Methods	Category	Param(M)	Energy (mJ)	T	Acc(%)
ViT-S	ANN	21.70	-	-	90.4
QViT-S-16L	QANN	21.70	-	-	88.4
tdBN	SNN (DT)	-	-	10	67.8
ASpikformer		8.46	-	-	-
Spikformer		9.32	-	16	80.9
SDformer		-	-	16	80.0
Sformer+CML		9.32	-	16	81.4
MST	SNN (A2S)	27.60	-	512	88.1
SZIP-TF		21.70	-	32	87.6
Baseline		21.70	16.7(5.6)	16(4)	88.4(74.5)
TMB		21.96	17.8(5.8)	16(4)	88.4(79.1)

Table 3: **Comparison on HAR-DVS.**

Methods	Category	Param (M)	Energy (mJ)	T	Acc.(%)
SlowFast	ANN	33.6	-	8	46.5
ACTION-Net		27.9	-	-	46.9
TimeSformer		121.2	-	-	50.8
Res-SNN-34	SNN (DT)	21.8	-	8	46.1
ASA-SNN		41.5	-	8	47.1
Meta-SFormer		18.3	8.0	8	47.5
E-SFormer	SNN (DT)	18.7	18.1	8	48.9
		18.7	23.5	8	49.2
Baseline	SNN(A2S)	21.70	9.6(5.4)	8(4)	50.5(42.6)
TMB	SNN(A2S)	21.96	9.9(5.7)	8(4)	50.5(47.7)

Table 4: **Comparison with A2S Works on COCO.**

Methods	Categ.	Param (M)	Energy (mJ)	T	mAP@ 50 (%)	mAP@ 50:95 (%)
PVT	ANN	32.9	520.3	1	59.2	36.7
DETR		41.0	197.8	1	62.4	42.0
YOLOv5		21.2	112.5	1	64.1	45.4
YOLOv10-S		7.2	-	1	63.0	-
Spiking-Yolo	A2S	10.2	-	3500	-	25.7
Bayesian Optim	A2S	10.2	-	500	-	21.1
		10.2	-	5000	-	25.8
Spike Calibration	A2S	17.1	-	128	43.6	-
		17.1	-	512	45.4	-
Baseline	A2S	233.6	160.7(106.4)	16(8)	56.8(25.4)	37.6(15.1)
TMB		234.0	166.7(112.3)	16(8)	56.8(27.5)	37.6(16.5)

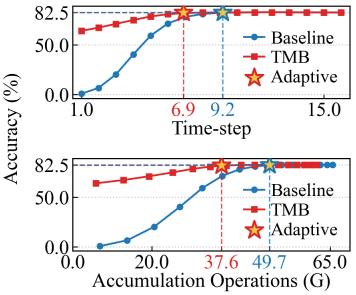


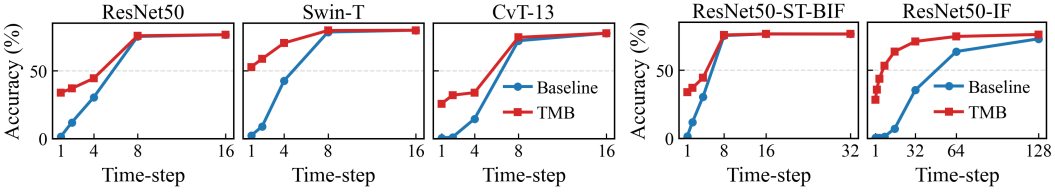
Figure 8: **Accuracy v.s. Time-Step&Energy in ViT-Base**.

outperforms SpikeZIP-TF (You et al., 2024) by 0.8% while reducing the required time-steps by half. On HAR-DVS, it achieves a 1.3% accuracy gain over E-SpikeFormer (Yao et al., 2025) under the same time-step constraint. Note that TMB has the same accuracy as the baseline when time-steps equal to $T_{eq} = 16$, experimentally proving the prediction invariance under TMB (Theorem 4.1).

Detection Comparisons on COCO are shown in Table 4. TMB significantly improves accuracy with fewer inference time-steps compared to existing A2S approaches. It achieves 56.8% mAP@50, an 11.4% improvement over Spike Calibration (45.4%) with a $32\times$ reduction in time-steps.

5.3 Accuracy, Time-step, and Operation Trade-off by Elastic Inference

TMB achieves superior time-step/operation–accuracy trade-offs in elastic inference. As shown in Figure 8, compared with the baseline, TMB's trade-off curve shifts left, showing average 44.8% and 45.2% SOPs and inference time-step reductions, respectively, at matched time-steps. We also adopt an adaptive inference method (SEENN (Li et al., 2023)) with a confidence threshold for Spike-ViT-B with/without TMB. As illustrated by the "star" in Figure 8, under 0.99 and 0.94 confidence, respectively, TMB achieves 24.3% operation and 24.8% time-step reduction with $< 0.1\%$ accuracy reduction (from 82.508% to 82.498%) compared to the same SNN without TMB.

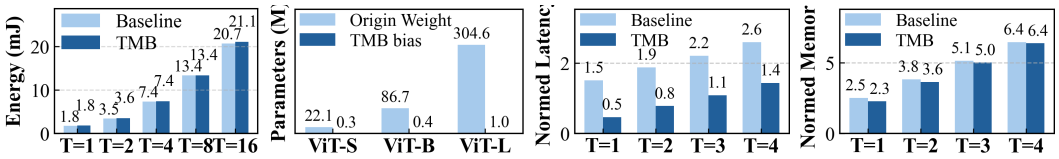


(a) Generalization across model structures. (b) Generalization across neuron types,

Figure 9: **Generalization of TMB** across (a) model architectures and (b) neuron types.

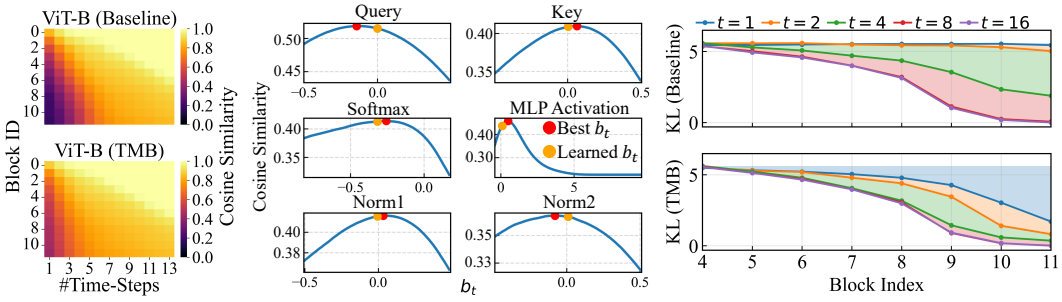
5.4 Generalization of TMB

TMB demonstrates strong generalization across different A2S algorithms, including variations in datasets, model architectures, and neuron types. Specifically, in Tables 1 to 4, TMB shows significant accuracy improvement at a small time-step across various datasets. In Figure 9a, TMB achieves accuracy improvements of 32.44%, 50.56%, and 25.27% at the first time-step for CNN-based,



(a) ViT-Small energy (b) # Parameters (c) GPU wall clock time (d) Memory Usage

Figure 10: **Cost Analysis of TMB,** including (a) energy consumption in inference, (b)# parameters, (c) GPU wall clock, and (d) memory usage in training. The test model is ViT-Small.



(a) Block-wise $\mathcal{I}$ (b) Learned bias b_t after Fine-tuning (c) KL Divergence via LogitLens 2020

Figure 11: **Visualization of TMB Optimization**. (a) block-wise representation alignment $\mathcal{I}$, measured by cosine similarity, w/ and w/o TMB. (b) $\mathcal{I}$ between learned bias and optimal bias at the first time-step. (c) block-wise KL Divergence (nostal., 2020) w/ and w/o TMB at different time-steps.

Transformer-based, and hybrid-based models, respectively. Figure 9b illustrates that TMB achieves 32.44% and 27.96% accuracy gains at the first time-step for IF (QCFS (Bu et al., 2023)) and ST-BIF (You et al., 2024) neurons, significantly accelerating the convergence of the QCFS algorithm.

5.5 Cost Analysis

Inference Cost Analysis. TMB incurs a negligible increase in energy consumption during inference. Added cost is only 1.9% of the total inference energy for ViT-Small and 4.4% for ViT-Large, as bias injection consumes far less energy than matrix multiplication in SNNs. Figure 10a further breaks this down by time-step on ViT-Small, where the per-step overhead stays small throughout. **Training Cost Analysis.** TMB accelerates training through bias-only fine-tuning. As shown in Figure 10b, only 0.32% of the parameters in ViT-Large are updated, leading to faster training. Figure 10c demonstrates up to a *3.0× speedup* when training the first time-step accuracy. However, as shown in Figure 10d, GPU memory usage is only marginally reduced, as memory overhead is primarily driven by spike activations rather than weight gradients in bias-only fine-tuning.

5.6 Visualization for TMB Optimization

We visualize TMB's optimization in Figure 11. As shown in Figure 11a, TMB brightens the dark regions, showing the improvement of block-wise representation alignment between SNN and QANN. Figure 11b shows learned b_t aligns more closely with optimal biases (Figure 6b), reducing membrane-threshold mismatch. In Figure 11c, TMB exhibits lower KL divergence in shallow layers and early time-steps, accelerating convergence. At later time-steps, TMB maintains lower KL divergence than baseline, *demonstrating that optimizing the first time-step benefits subsequent accuracy.*

6 Conclusion

Elastic inference is an inherent temporal property of SNNs, enabling earlier responses to salient inputs. To solve the low accuracy problem at early time-steps and enhance the elastic inference, we define and systematically analyze the "membrane-threshold mismatch" in the SNN inference system. To mitigate the information dissipation, we propose temporal membrane biasing method that adds a bias to the neural membrane for information calibration. TMB improves the SNN accuracy at the first time-step substantially with a small inference cost.

AI USE STATEMENT

Large Language Models (LLMs) were used solely for language polishing, including sentence rephrasing, grammar checking, and improving readability. All research ideas, methodological design, code implementation, experiments, and analyses were independently developed and conducted by the authors. The authors take full responsibility for the content of the manuscript.

REFERENCES

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. 2022. Beit: Bert pre-training of image transformers.

Tong Bu, Jianhao Ding, Zhaofei Yu, and Tiejun Huang. 2022. Optimized potential initialization for low-latency spiking neural networks.

Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang. 2023. Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks. *arXiv preprint arXiv:2303.04347*.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators.

Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. 2019. Randaugment: Practical automated data augmentation with a reduced search space.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Shikuang Deng and Shi Gu. 2021. Optimal conversion of conventional artificial neural networks to spiking neural networks. *arXiv preprint arXiv:2103.00476*.

Alexey Dosovitskiy. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Yuetong Fang, Deming Zhou, Ziqing Wang, Hongwei Ren, ZeCui Zeng, Lusong Li, Shibo Zhou, and Renjing Xu. 2026. Spiking neural networks need high frequency information.

Zecheng Hao, Tong Bu, Jianhao Ding, Tiejun Huang, and Zhaofei Yu. 2023. Reducing ann-snn conversion error through residual membrane potential. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11–21.

Sadri Hassani and Sadri Hassani. 2009. Dirac delta function. *Mathematical Methods: For Students of Physics and Related Fields*, pages 139–170.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

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

Yangfan Hu, Qian Zheng, Xudong Jiang, and Gang Pan. 2023. Fast-snn: fast spiking neural network by converting quantized ann. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Weinberger. 2016. Deep networks with stochastic depth.

Zihan Huang, Wei Fang, Tong Bu, Peng Xue, Zecheng Hao, Wenxuan Liu, Yuanhong Tang, Zhaofei Yu, and Tiejun Huang. 2025. Differential coding for training-free ann-to-snn conversion. *arXiv preprint arXiv:2503.00301*.

Zihan Huang, Xinyu Shi, Zecheng Hao, Tong Bu, Jianhao Ding, Zhaofei Yu, and Tiejun Huang. 2024. Towards high-performance spiking transformers from ann to snn conversion. In *Proceedings of the 32nd ACM international conference on multimedia*, pages 10688–10697.

Yizhou Jiang, Kunlin Hu, Tianren Zhang, Haichuan Gao, Yuqian Liu, Ying Fang, and Feng Chen. 2024. Spatio-temporal approximation: A training-free snn conversion for transformers. In *The twelfth international conference on learning representations*.

Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. 2022. Exploring plain vision transformer backbones for object detection.

Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. 2021a. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration.

Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. 2021b. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. In *International conference on machine learning*, pages 6316–6325. PMLR.

Yuhang Li, Tamar Geller, Youngeun Kim, and Priyadarshini Panda. 2023. Seenn: Towards temporal spiking early-exit neural networks.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer Vision – ECCV 2014*, pages 740–755, Cham. Springer International Publishing.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022.

nostal. 2020. Interpreting gpt: The logit lens. `https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens`. LessWrong blog post.

Rui Qian, Xin Lai, and Xirong Li. 2022. 3d object detection for autonomous driving: A survey. *Pattern Recognition*, 130:108796.

Nitin Rathi and Kaushik Roy. 2020. Diet-snn: Direct input encoding with leakage and threshold optimization in deep spiking neural networks. *arXiv preprint arXiv:2008.03658*.

Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. 2019. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826.

Doron Tal and Eric L Schwartz. 1997. Computing with the leaky integrate-and-fire neuron: logarithmic computation and multiplication. *Neural computation*, 9(2):305–318.

Simon Thorpe, Denis Fize, and Catherine Marlot. 1996. Speed of processing in the human visual system. *nature*, 381(6582):520–522.

Xiao Wang, Zongzhen Wu, Bo Jiang, Zhimin Bao, Lin Zhu, Guoqi Li, Yaowei Wang, and Yonghong Tian. 2024. Hardvs: Revisiting human activity recognition with dynamic vision sensors. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 5615–5623.

Yuchen Wang, Malu Zhang, Yi Chen, and Hong Qu. 2022. Signed neuron with memory: Towards simple, accurate and high-efficient ann-snn conversion. In *IJCAI*, pages 2501–2508.

Ziqing Wang, Yuetong Fang, Jiahang Cao, Qiang Zhang, Zhongrui Wang, and Renjing Xu. 2023. Masked spiking transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1761–1771.

Eric W Weisstein. 2002. Heaviside step function. *https://mathworld. wolfram. com/*.

Dengyu Wu, Gaojie Jin, Han Yu, Xinping Yi, and Xiaowei Huang. 2025. Optimizing event-driven spiking neural network with regularization and cutoff. *Frontiers in Neuroscience*, 19.

Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. 2021. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 22–31.

Long Xia, XiaoRan Wang, RuGuo Wang, YingYing Yin, HaoNan Xu, and XiangYu Li. 2026. Enhancing ann-snn conversion: Addressing low latency and negative thresholds with modified activation and bptt fine-tuning. *Neurocomputing*, page 132604.

Peipei Xia, Li Zhang, and Fanzhang Li. 2015. Learning similarity with cosine similarity ensemble. *Information sciences*, 307:39–52.

Jing Yan, Kang You, Zhezhi He, and Yaoyu Zhang. 2026. Determinism in the undetermined: Deterministic output in charge-conserving continuous-time neuromorphic systems with temporal stochasticity.

Man Yao, Jiakui Hu, Tianxiang Hu, Yifan Xu, Zhaokun Zhou, Yonghong Tian, Bo Xu, and Guoqi Li. 2024. Spike-driven transformer v2: Meta spiking neural network architecture inspiring the design of next-generation neuromorphic chips. *arXiv preprint arXiv:2404.03663*.

Man Yao, Xuerui Qiu, Tianxiang Hu, Jiakui Hu, Yuhong Chou, Keyu Tian, Jianxing Liao, Luziwei Leng, Bo Xu, and Guoqi Li. 2025. Scaling spike-driven transformer with efficient spike firing approximation training. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(4):2973–2990.

Kang You, Ziling Wei, Jing Yan, Boning Zhang, Qinghai Guo, Yaoyu Zhang, and Zhezhi He. 2025. Vistream: Improving computation efficiency of visual streaming perception via law-of-charge-conservation inspired spiking neural network. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 8796–8805.

Kang You, Zekai Xu, Chen Nie, Zhijie Deng, Qinghai Guo, Xiang Wang, and Zhezhi He. 2024. Spikezip-tf: Conversion is all you need for transformer-based snn. *arXiv preprint arXiv:2406.03470*.

Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Seong Joon Oh, Youngjoon Yoo, and Junsuk Choe. 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6022–6031.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization.

Chenlin Zhou, Liutao Yu, Zhaokun Zhou, Zhengyu Ma, Han Zhang, Huihui Zhou, and Yonghong Tian. 2023. Spikingformer: Spike-driven residual learning for transformer-based spiking neural network. *arXiv preprint arXiv:2304.11954*.

Zhaokun Zhou, Kaiwei Che, Wei Fang, Keyu Tian, Yuesheng Zhu, Shuicheng Yan, Yonghong Tian, and Li Yuan. 2024. Spikformer v2: Join the high accuracy club on imagenet with an snn ticket. *arXiv preprint arXiv:2401.02020*.

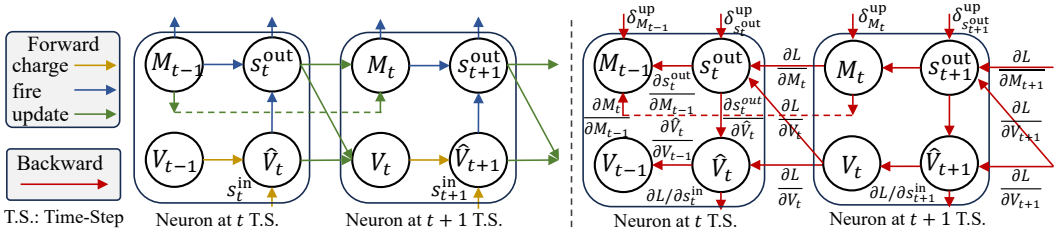


Figure A1: **Computation Graph of ST-BIF Neuron.**

A1 Appendix

A1.1 Limitation

Although TMB substantially improves low-time-step accuracy and enables faster SNN responses, TMB introduces a small amount of additional computation for membrane calibration. This overhead is negligible in our evaluated settings, but may become a concern in extremely energy-constrained scenarios. In addition, TMB mainly improves the early-stage behavior of converted SNNs and does not further improve the terminal accuracy after convergence. Therefore, more advanced conversion algorithms are still needed to obtain higher-accuracy SNNs at late time steps.

A1.2 Proof of Theorem 4.1

Theorem A1.1 (Repeated: Prediction Invariance under TMB)**.** *For an L_0-layer feed-forward ST-BIF neural network with weights $\{\mathbf{W}_\theta{}^l\}_{l=0}^{L_0-1}$, biases $\{\mathbf{b}_\theta{}^l\}_{l=0}^{L_0-1}$, and TMB bias $\{\mathbf{b}_t^l\}_{l=0}^{L_0-1}$, the temporal accumulation of the output spikes at equilibrium, denoted by $\sum_{t=0}^{T_{\text{eq}}} \hat{\boldsymbol{s}}_t^{\text{out}}$, remains unchanged:*

$$\begin{aligned}
&\sum_{t=0}^{T_{\text{eq}}} \hat{\boldsymbol{s}}_t^{\text{out}} = \sum_{t=0}^{T_{\text{eq}}} \boldsymbol{s}_t^{\text{out}} \\
&= \mathbf{F}_{\text{Q}}(\mathbf{Q}_{\text{total}}; \{\mathbf{W}_\theta{}^l\}_{l=0}^{L_0-1}; \{\mathbf{b}_\theta{}^l + \sum_{t=0}^{T_{\text{eq}}} \mathbf{b_t}{}^l\}_{l=0}^{L_0-1}) \\
&s.t., \quad \{\sum_{t=0}^{T_{\text{eq}}} \boldsymbol{b}_t^l\}_{l=0}^{L_0-1} = \{\mathbf{0}\}_{l=0}^{L_0-1}
\end{aligned} \tag{A1}$$

where $\boldsymbol{s}_t^{\text{out}}$ is the accumulated output of SNN sharing the same weights and biases without using TMB bias, l is the l^{th} layer, and t is the t^{th} time-step.

Proof. For an SNN layer without using TMB, by using Theorem 2.1, we have:

$$\sum_{t=0}^{T_{\text{eq}}} \boldsymbol{s}_t^{\text{out}} = \mathbf{F}_{\text{Q}}(\mathbf{Q}_{\text{total}}; \{\mathbf{W}_\theta{}^l\}_{l=0}^{L_0-1}; \{\mathbf{b}_\theta{}^l\}_{l=0}^{L_0-1})$$

For an SNN layer with TMB, by using Theorem 2.1, we have:

$$\sum_{t=0}^{T_{\text{eq}}} \hat{\boldsymbol{s}}_t^{\text{out}} = \mathbf{F}_{\text{Q}}(\mathbf{Q}_{\text{total}}; \{\mathbf{W}_\theta{}^l\}_{l=0}^{L_0-1}; \{\mathbf{b}_\theta{}^l + \sum_{t=0}^{T_{\text{eq}}} \mathbf{b_t}{}^l\}_{l=0}^{L_0-1})$$

Since $\{\sum_{t=0}^{T_{\text{eq}}} \boldsymbol{b}_t^l\}_{l=0}^{L_0-1} = \{\mathbf{0}\}_{l=0}^{L_0-1}$, we have:

$$\sum_{t=0}^{T_{\text{eq}}} \hat{\boldsymbol{s}}_t^{\text{out}} = \mathbf{F}_{\text{Q}}(\mathbf{Q}_{\text{total}}; \{\mathbf{W}_\theta{}^l\}_{l=0}^{L_0-1}; \{\mathbf{b}_\theta{}^l\}) = \sum_{t=0}^{T_{\text{eq}}} \boldsymbol{s}_t^{\text{out}}$$

□

A1.3 BPTT for SNN with ST-BIF Neuron

The forward procedure of ST-BIF neuron is different from LIF neuron. Consequently, deriving the backward gradient of ST-BIF neuron is essential for implementing the BPTT algorithm on the SNN comprised of ST-BIF neurons. For the convenience of derivation, we summarize the meaning of notations appearing in Table A1.

Table A1: The Notations in Formula Derivation

Symbol	Description
$\mathcal{L}$	Loss function
w_i	synaptic weights connected to i-th neuron
s_t^{in}	Input spike at t time-step
s_t^{out}	Output spikes at t time-step
V_t	Membrane potential at t time-step
V_{thr}	Membrane threshold for neuron firing
M_t	Spike tracer at t time-step
$\hat{V}_t$	Membrane potential after charging before firing at t time-step
$M_{\min}$	The minimum value of spike tracer
$M_{\max}$	The maximum value of spike tracer
Θ	Fire decision function of ST-BIF Neuron You et al. (2024)
θ	Step function Weisstein (2002)
ε	A small value

A1.4 Feed-forward of ST-BIF Neuron

Before the derivation of backward gradient, we display the forward path of ST-BIF neuron in Figure A1(left). ST-BIF neuron charges the membrane potential V_{t-1} with the input s_t^{in} to get $\hat{V}_t$ (yellow arrows in Figure A1), fires spikes according to the firing decision function Θ (blue arrows in Figure A1) and update the $M_{t-1}, \hat{V}_t$ to the states at the next time-step (green arrows in Figure A1). The feed-forward definition of ST-BIF neuron is below:

$$\hat{V}_t = V_{t-1} + (\sum\nolimits_i w_i \cdot S_{i,t}^{\text{in}}) \tag{A2}$$

$$\begin{aligned} S_t^{\text{out}} &= \Theta(\hat{V}_t, V_{\text{thr}}, M_{t-1}) \\ \Theta(\hat{V}_t, V_{\text{thr}}, M_{t-1}) &= \theta(\hat{V}_t - V_{\text{thr}}) \cdot \theta(M_{\max} - M_{t-1} - \varepsilon) - \theta(-\hat{V}_t - \varepsilon) \cdot \theta(M_{t-1} - M_{\min} - \varepsilon) \end{aligned} \tag{A3}$$

$$V_t = \hat{V}_t - V_{\text{thr}} \times s_t^{\text{out}}; \quad M_t = M_{t-1} + \Theta(\hat{V}_t, V_{\text{thr}}, M_{t-1}) \tag{A4}$$

where Equation (A2),Equation (A3),Equation (A4) are the charging, firing and updating procedures in ST-BIF neuron, respectively.

A1.5 Backward of ST-BIF Neuron

After forwarding, we calculate the gradients through the opposite direction of the forward computing graph. As displayed in the backward pass in Figure A1 (right), ST-BIF neuron at t time-step receives partial derivatives of the loss function $\mathcal{L}$ to membrane potential V_t and spike tracer M_t from the same neuron at $t+1$ time-step, *i.e.*, $\partial\mathcal{L}/\partial V_t, \partial\mathcal{L}/\partial M_t$, as well as the partial derivatives of $\partial\mathcal{L}/\partial Y_t$ from neurons in next layer (red arrows outside the neuron at t time-step in Figure A1).

After receiving these partial derivatives, the neuron calculates the partial derivatives of the loss function $\mathcal{L}$ to membrane potential V_{t-1}, spike tracer M_{t-1} and input X_t, *i.e.*, $\partial\mathcal{L}/\partial V_{t-1}, \partial\mathcal{L}/\partial M_{t-1}$ and $\partial\mathcal{L}/\partial X_t$ for the $t-1$ time-step. Then, the neuron transmits $\partial\mathcal{L}/\partial V_{t-1}$ and $\partial\mathcal{L}/\partial M_{t-1}$ to the same neuron at $t-1$ time-step, and sends $\partial\mathcal{L}/\partial X_t$ to the neurons in the last layer. In summary, we need to derive three partial derivatives: $\partial\mathcal{L}/\partial V_{t-1}, \partial\mathcal{L}/\partial M_{t-1}$ and $\partial\mathcal{L}/\partial X_t$.

According to the computation graph in Figure A1 and chain rule, we write the calculation formulation for the three partial derivatives:

$$\begin{aligned} \frac{\partial\mathcal{L}}{\partial V_{t-1}} &= \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}s_t^{\text{out}}}\frac{\partial s_t^{\text{out}}}{\partial \hat{V}_t}\frac{\partial \hat{V}_t}{\partial V_{t-1}} + \frac{\partial\mathcal{L}}{\partial V_t}\frac{\partial \hat{V}_t}{\partial V_{t-1}}, \quad \frac{\partial\mathcal{L}}{\partial M_{t-1}} = \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}s_t^{\text{out}}}\frac{\partial s_t^{\text{out}}}{\partial M_{t-1}} + \frac{\partial\mathcal{L}}{\partial M_t}\frac{\partial M_t}{\partial M_{t-1}} \\ \frac{\partial\mathcal{L}}{\partial s_t^{\text{in}}} &= \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}s_t^{\text{out}}}\frac{\partial s_t^{\text{out}}}{\partial \hat{V}_t}\frac{\partial \hat{V}_t}{\partial s_t^{\text{in}}} + \frac{\partial\mathcal{L}}{\partial V_t}\frac{\partial \hat{V}_t}{\partial s_t^{\text{in}}}, \quad \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}s_t^{\text{out}}} = \frac{\partial\mathcal{L}}{\partial s_t^{\text{out}}} + \frac{\partial\mathcal{L}}{\partial M_t}\frac{\partial M_t}{\partial s_t^{\text{out}}} + \frac{\partial\mathcal{L}}{\partial V_t}\frac{\partial V_t}{\partial s_t^{\text{out}}} \end{aligned} \tag{A5}$$

where $\mathrm{d}\mathcal{L}/\mathrm{d}s_t^{\text{out}}$ contains all the partial derivatives transmitted to Y_t, including $\partial\mathcal{L}/\partial s_t^{\text{out}}$, $\partial\mathcal{L}/\partial M_t$ and $\partial\mathcal{L}/\partial V_t$ (the red arrows to s_t^{out} in Figure A1). Then, combining with the firing decision function

of ST-BIF neuron defined in Equation (A3), we calculate the partial derivatives in Equation (A5) and have:

$$\frac{\partial \hat{V}_t}{\partial V_{t-1}} = \frac{\partial M_t}{\partial s_t^{\text{out}}} = \frac{\partial \hat{V}_t}{\partial s_t^{\text{in}}} = 1, \quad \frac{\partial V_t}{\partial s_t^{\text{out}}} = -V_{\text{thr}} \tag{A6}$$

$$\frac{\partial M_t}{\partial M_{t-1}} = 1 - \frac{\partial s_t^{\text{out}}}{\partial M_{t-1}} \tag{A7}$$

$$\frac{\partial s_t^{\text{out}}}{\partial \hat{V}_t} = \frac{\partial \Theta(\hat{V}_t, V_{\text{thr}}, M_t)}{\partial \hat{V}_t}, \quad \frac{\partial s_t^{\text{out}}}{\partial M_{t-1}} = \frac{\partial \Theta(\hat{V}_t, V_{\text{thr}}, M_t)}{\partial M_{t-1}} \tag{A8}$$

$$\frac{\partial \Theta(\hat{V}_t, V_{\text{thr}}, M_t)}{\partial \hat{V}_t} = \theta'(\hat{V}_t - V_{\text{thr}})\,\theta(M_{\max} - M_{t-1} - \varepsilon) + \theta'(-\hat{V}_t - \varepsilon)\,\theta(M_{t-1} - M_{\min} - \varepsilon) \tag{A9}$$

$$\frac{\partial \Theta(\hat{V}_t, V_{\text{thr}}, M_t)}{\partial M_{t-1}} = -\theta(\hat{V}_t - V_{\text{thr}})\,\theta'(M_{\max} - M_{t-1} - \varepsilon) - \theta(-\hat{V}_t - \varepsilon)\,\theta'(M_{t-1} - M_{\min} - \varepsilon) \tag{A10}$$

where θ' is Dirac delta function Hassani and Hassani (2009), which is the derivative of θ. Since Dirac delta function is hard to implement in hardware, we use the normal distribution function as the surrogate function of θ'. Then, we substitute Equation (A6)-Equation (A10) to Equation (A5) and finally get:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial V_{t-1}} &= \Big(\frac{\partial \mathcal{L}}{\partial s_t^{\text{out}}} + \frac{\partial \mathcal{L}}{\partial M_t} - V_{\text{thr}}\frac{\partial \mathcal{L}}{\partial V_t}\Big)\frac{\partial s_t^{\text{out}}}{\partial \hat{V}_t} + \frac{\partial \mathcal{L}}{\partial V_t}, \\
\frac{\partial \mathcal{L}}{\partial M_{t-1}} &= \Big(\frac{\partial \mathcal{L}}{\partial s_t^{\text{out}}} + \frac{\partial \mathcal{L}}{\partial M_t} - V_{\text{thr}}\frac{\partial \mathcal{L}}{\partial V_t}\Big)\frac{\partial s_t^{\text{out}}}{\partial M_{t-1}} + \frac{\partial \mathcal{L}}{\partial M_t}\Big(1 - \frac{\partial s_t^{\text{out}}}{\partial M_{t-1}}\Big), \\
\frac{\partial \mathcal{L}}{\partial s_t^{\text{in}}} &= \Big(\frac{\partial \mathcal{L}}{\partial s_t^{\text{out}}} + \frac{\partial \mathcal{L}}{\partial M_t} - V_{\text{thr}}\frac{\partial \mathcal{L}}{\partial V_t}\Big)\frac{\partial s_t^{\text{out}}}{\partial \hat{V}_t} + \frac{\partial \mathcal{L}}{\partial V_t}.
\end{aligned} \tag{A11}$$

where we do not substitute $\partial s_t^{\text{out}}/\partial \hat{V}_t$ and $\partial s_t^{\text{out}}/\partial M_{t-1}$ for the convenient of reading. After simply transforming, we finally get:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial V_{t-1}} &= \Big(\frac{\partial \mathcal{L}}{\partial s_t^{\text{out}}} + \frac{\partial \mathcal{L}}{\partial M_t} - V_{\text{thr}}\frac{\partial \mathcal{L}}{\partial V_t}\Big)\frac{\partial \Theta(\hat{V}_t, V_{\text{thr}}, M_t)}{\partial \hat{V}_t} + \frac{\partial \mathcal{L}}{\partial V_t}, \\
\frac{\partial \mathcal{L}}{\partial M_{t-1}} &= \Big(\frac{\partial \mathcal{L}}{\partial s_t^{\text{out}}} - V_{\text{thr}}\frac{\partial \mathcal{L}}{\partial V_t}\Big)\frac{\partial \Theta(\hat{V}_t, V_{\text{thr}}, M_t)}{\partial M_{t-1}} + \frac{\partial \mathcal{L}}{\partial M_t}, \\
\frac{\partial \mathcal{L}}{\partial s_t^{\text{in}}} &= \Big(\frac{\partial \mathcal{L}}{\partial s_t^{\text{out}}} + \frac{\partial \mathcal{L}}{\partial M_t} - V_{\text{thr}}\frac{\partial \mathcal{L}}{\partial V_t}\Big)\frac{\partial \Theta(\hat{V}_t, V_{\text{thr}}, M_t)}{\partial \hat{V}_t} + \frac{\partial \mathcal{L}}{\partial V_t}.
\end{aligned} \tag{A12}$$

With Equation (A12), we correctly estimate the gradient for ST-BIF neuron and make the SNN in TMB support the BPTT training algorithm.

A1.6 Detailed Training Setup

For the reproduction of our experiment results, we show detailed training settings in Tables A2 to A6, including training hyperparameters, data augmentations, and other training technologies.

Table A2: **Quantization-Aware Training and Post-Conversion Fine-tuning Setting with ViT-small on ImageNet.**

config	Quantization-Aware Training	Post-Conversion Fine-tuning
optimizer	AdamW	AdamW
base learning rate	2.5e-5	3e-4
weight decay	0.0001	0.05
optimizer momentum	β_1, β_2=0.9, 0.999	β_1, β_2=0.9, 0.999
layer-wise lr decay Clark et al. (2020); Bao et al. (2022)	None	None
batch size	1024	512
learning rate schedule	cosine decay	cosine decay
quantization level	10	10
training epochs	100	1
augmentation	RandAug (9, 0.5) Cubuk et al. (2019)	RandAug (9, 0.5) Cubuk et al. (2019)
label smoothing Szegedy et al. (2016)	0.1	0.1
mixup Zhang et al. (2018)	0.2	None
cutmix Yun et al. (2019)	0.2	None
drop path Huang et al. (2016)	0.1	0.1

Table A3: **Quantization-Aware Training and Post-Conversion Fine-tuning Setting with ViT-base on ImageNet.**

config	Quantization-Aware Training	Post-Conversion Fine-tuning
optimizer	AdamW	AdamW
base learning rate	5e-6	1e-4
weight decay	0.0001	0.05
optimizer momentum	β_1, β_2=0.9, 0.999	β_1, β_2=0.9, 0.999
layer-wise lr decay Clark et al. (2020); Bao et al. (2022)	None	None
batch size	512	256
learning rate schedule	cosine decay	cosine decay
quantization level	10	10
training epochs	50	1
augmentation	RandAug (9, 0.5) Cubuk et al. (2019)	RandAug (9, 0.5) Cubuk et al. (2019)
label smoothing Szegedy et al. (2016)	0.1	0.1
mixup Zhang et al. (2018)	0.2	None
cutmix Yun et al. (2019)	0.2	None
drop path Huang et al. (2016)	0.1	0.1

Table A4: **Quantization-Aware Training and Post-Conversion Fine-tuning Setting with ViT-Large on ImageNet.**

config	Quantization-Aware Training	Post-Conversion Fine-tuning
optimizer	AdamW	AdamW
base learning rate	8e-5	2e-4
weight decay	0.0001	0.05
optimizer momentum	β_1, β_2=0.9, 0.999	β_1, β_2=0.9, 0.999
layer-wise lr decay Clark et al. (2020); Bao et al. (2022)	None	None
batch size	256	256
learning rate schedule	cosine decay	cosine decay
quantization level	10	10
training epochs	20	1
augmentation	RandAug (9, 0.5) Cubuk et al. (2019)	RandAug (9, 0.5) Cubuk et al. (2019)
label smoothing Szegedy et al. (2016)	0.1	0.1
mixup Zhang et al. (2018)	0.2	None
cutmix Yun et al. (2019)	0.2	None
drop path Huang et al. (2016)	0.2	0.1

Table A5: **Quantization-Aware Training and SNN Training Setting on CIFAR10-DVS and HAR-DVS.**

config	ANN	QANN	SNN
optimizer	AdamW	AdamW	AdamW
learning rate	4e-4	1e-4	3e-4
weight decay	0.05	0.05	0.05
optimizer momentum	β_1, β_2=0.9, 0.999	β_1, β_2=0.9, 0.999	β_1, β_2=0.9, 0.999
layer-wise lr decay Clark et al. (2020); Bao et al. (2022)	None	None	None
batch size	512	512	256
learning rate schedule	cosine decay	cosine decay	cosine decay
warmup epochs	10	5	None
training epochs	600	100	10
augmentation	RandAug (9, 0.5) Cubuk et al. (2019)	RandAug (9, 0.5) Cubuk et al. (2019)	RandAug (9, 0.5) Cubuk et al. (2019)
label smoothing Szegedy et al. (2016)	0.1	0.1	0.1
mixup Zhang et al. (2018)	0.8	0.2	None
cutmix Yun et al. (2019)	1.0	0.2	None
drop path Huang et al. (2016)	0.1	0.1	0.1

Table A6: **Quantization-Aware Training and SNN Training Setting on COCO.**

config	QANN	SNN
optimizer	AdamW	AdamW
learning rate	1e-4	1e-4
weight decay	0.1	0.1
optimizer momentum	β_1, β_2=0.9, 0.999	β_1, β_2=0.9, 0.999
layer-wise lr decay Clark et al. (2020); Bao et al. (2022)	None	None
batch size	32	8
learning rate schedule	step decay	None
warmup iters	500	None
training epochs	80	1
augmentation	RandomCrop + RandomFlip + FilterAnnotations	RandomCrop+ RandomFlip + FilterAnnotations
label smoothing Szegedy et al. (2016)	0.1	0.1
mixup Zhang et al. (2018)	None	None
cutmix Yun et al. (2019)	None	None
drop path Huang et al. (2016)	None	None