Temporal Interaction in Spiking Transformers with Multi-Delay Mixer

Supplementary Material

1. Definitions and Properties of TIC

This section introduces the formal definition of the Temporal Interaction Coefficient (TIC) and establishes its theoretical properties.

Definition 1.1. *Let $\mathbf{X}_p \in \mathbb{R}^{N\times D}$ denote the attention input at timestep p and $\mathbf{Z}_q \in \mathbb{R}^{N\times D}$ denote the attention output at timestep q. We define the temporal reliance vector at timestep q as*

$$\mathbf{R}_q = (R_{q\to 1}, R_{q\to 2}, \ldots, R_{q\to q}) \in \mathbb{R}^q, \tag{1}$$

where

$$R_{q\to p} = \left\| \frac{\partial \mathbf{Z}_q}{\partial \mathbf{X}_p} \right\|_1 \tag{2}$$

for $p \le q$, and $\|\cdot\|_1$ denotes the sum of absolute values of all entries in the Jacobian tensor. This vector captures the complete temporal dependency structure of the output on all historical inputs.

Definition 1.2. *The Temporal Interaction Coefficient at timestep q is defined as*

$$\mathrm{TIC}_q = -\sum_{p=1}^{q} \tilde{R}_{q\to p} \log \tilde{R}_{q\to p}, \tag{3}$$

where

$$\tilde{R}_{q\to p} = \frac{R_{q\to p}}{\sum_{i=1}^{q} R_{q\to i}}$$

is the normalized reliance coefficient. We further denote $\tilde{\mathbf{R}}_q = (\tilde{R}_{q\to 1}, \ldots, \tilde{R}_{q\to q}) \in \mathbb{R}^q$ as the normalized temporal reliance vector. This coefficient measures the diversity of temporal dependencies.

Theorem 1.1. *The Temporal Interaction Coefficient $\mathrm{TIC}_q = H(\tilde{\mathbf{R}}_q)$ satisfies the following properties:*

(i) $0 \le \mathrm{TIC}_q \le \log q$.

(ii) $\mathrm{TIC}_q = 0$ *if and only if* $\tilde{R}_{q\to p} = 1$ *for exactly one* $p \in \{1, \ldots, q\}$.

(iii) $\mathrm{TIC}_q = \log q$ *if and only if* $\tilde{R}_{q\to p} = 1/q$ *for all* $p = 1, \ldots, q$.

(iv) The effective number $N_{\text{eff}} = \exp(\mathrm{TIC}_q)$ *satisfies* $1 \le N_{\text{eff}} \le q$.

Proof. (i) Since $\tilde{\mathbf{R}}_q$ is a probability distribution, the non-negativity of entropy gives $H(\tilde{\mathbf{R}}_q) \ge 0$. For the upper bound, by the concavity of entropy, $H(\tilde{\mathbf{R}}_q)$ is maximized when $\tilde{\mathbf{R}}_q$ is uniform, yielding

$$H(\tilde{\mathbf{R}}_q) \le -\sum_{p=1}^{q} \frac{1}{q} \log \frac{1}{q} = \log q.$$

(ii) The entropy $H(\tilde{\mathbf{R}}_q) = 0$ if and only if the distribution has zero uncertainty, which occurs precisely when one component equals 1 and all others equal 0.

(iii) As noted in (i), the uniform distribution uniquely maximizes entropy among all distributions on q outcomes.

(iv) Since exp is strictly increasing and $0 \le \mathrm{TIC}_q \le \log q$, we have

$$1 = \exp(0) \le N_{\text{eff}} = \exp(\mathrm{TIC}_q) \le \exp(\log q) = q. \quad \square$$

2. Training Methods for SNNs

In this section, we introduce the gradient descent training process for SNNs and the parameter update method based on spatio-temporal backpropagation (STBP) [1, 2]. Once the derivative of the spike generation function is established, SNN parameters can be optimized using gradient descent algorithms following principles similar to that in ANNs. The gradients of the loss function $\mathcal{L}$ with respect to the synaptic weights $\mathbf{W}$ at layer ℓ are computed as:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \mathbf{W}^\ell} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial \mathbf{S}_t^{\ell+1}} \frac{\partial \mathbf{S}_t^{\ell+1}}{\partial \mathbf{U}_t^{\ell+1}} \Bigg(\frac{\partial \mathbf{U}_t^{\ell+1}}{\partial \mathbf{W}^\ell} \\ + \sum_{\tau<t} \prod_{i=t-1}^{\tau} \Bigg(\frac{\partial \mathbf{U}_{i+1}^{\ell+1}}{\partial \mathbf{U}_i^{\ell+1}} \\ + \frac{\partial \mathbf{U}_{i+1}^{\ell+1}}{\partial \mathbf{S}_i^{\ell+1}} \frac{\partial \mathbf{S}_i^{\ell+1}}{\partial \mathbf{U}_i^{\ell+1}} \Bigg) \frac{\partial \mathbf{U}_\tau^{\ell+1}}{\partial \mathbf{W}^\ell} \Bigg),\end{aligned} \tag{4}$$

where $\mathbf{U}_t^\ell$ and $\mathbf{S}_t^\ell$ denote the membrane potential and binary spike output of neurons in layer ℓ at time step t, respectively.

However, it should be noted that $\frac{\partial \mathbf{S}_t^\ell}{\partial \mathbf{U}_t^\ell}$ is non-differentiable due to the binary nature of spike generation. To address this issue, [1] proposed a surrogate gradient approach that assigns nonzero gradients only to neurons whose membrane potentials close to the firing threshold during backpropagation. In this work, we employ the sigmoid surrogate function, which has demonstrated effectiveness in gradient descent optimization and is defined as:

$$\frac{\partial \mathbf{S}_t^\ell}{\partial \mathbf{U}_t^\ell} = \frac{1}{a} \frac{e^{\frac{V_{th}-\mathbf{U}_t^\ell}{a}}}{\left(1 + e^{\frac{V_{th}-\mathbf{U}_t^\ell}{a}}\right)^2}, \tag{5}$$

where a is a scaling parameter that controls the width of the gradient approximation around the threshold V_{th}.

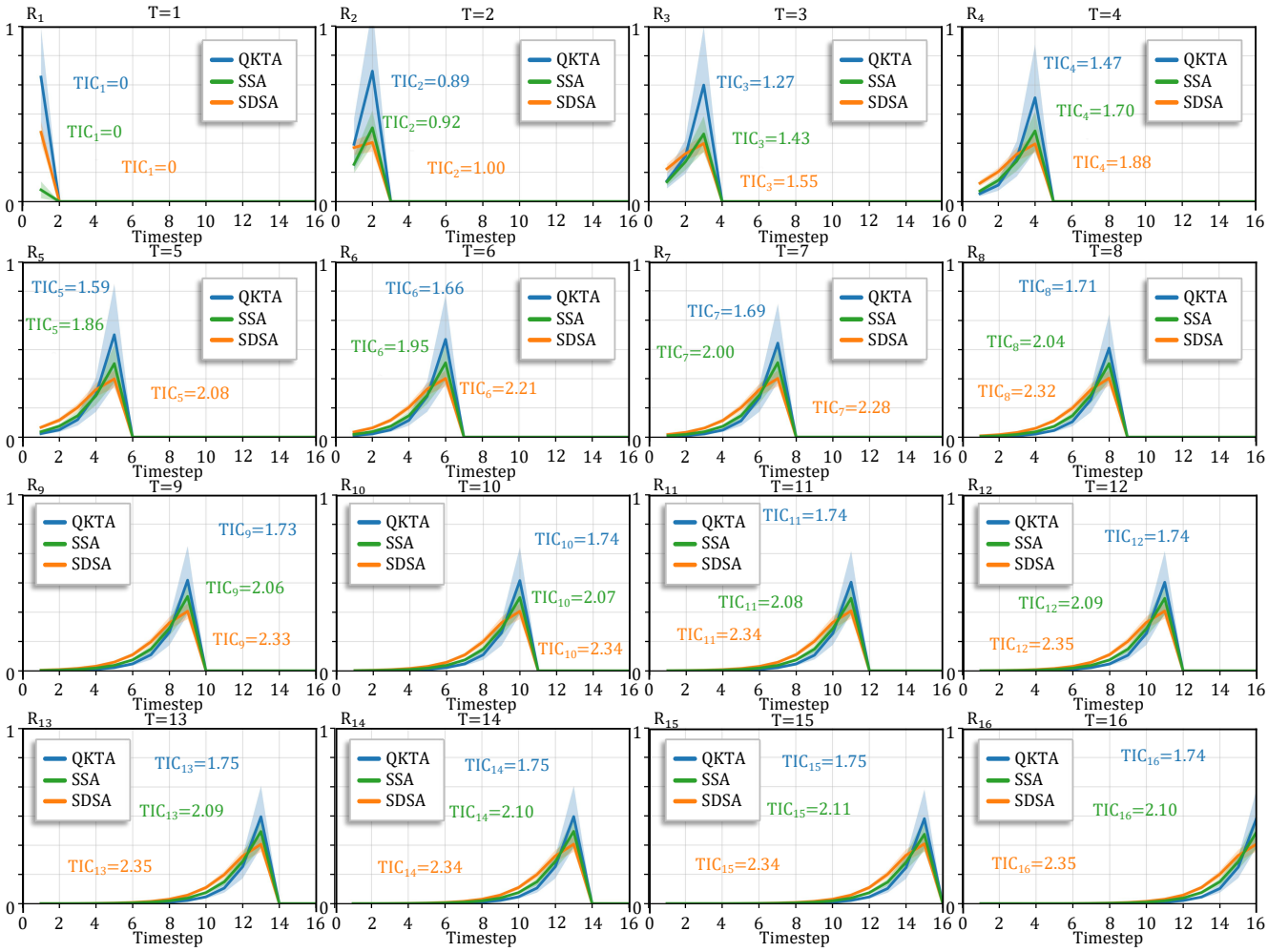

Figure 1. Visualization of temporal reliance distributions for SSA, QKTA, and SDSA attention mechanisms across all timesteps, based on the CIFAR10-DVS dataset.

3. Detailed Analysis of Delay Optimization

In this section, we provide a comprehensive analysis of the delay optimization mechanism in MD-Mixer, including gradient derivation, temperature annealing schedule, and convergence properties.

3.1. Gradient Derivation

The learnable delay center d^* and the scaling coefficients $\{\alpha_j^{(k)}\}_{k=1}^K$ are jointly optimized via backpropagation. To understand how gradients flow through the delay distribution, we first examine the partial derivative of the delay weight $\phi(d; d^*)$ with respect to the delay center d^*.

Recall that the delay distribution is defined as:

$$\phi(d; d^*) = \mathcal{N}\left(\sigma\left(1 - \frac{|d - d^*|}{\tau}\right)\right), \tag{6}$$

where $\mathcal{N}(\cdot)$ denotes normalization and $\sigma(\cdot) = \max(0, \cdot)$ is the ReLU activation. Let $z_d = 1 - \frac{|d-d^*|}{\tau}$ denote the argument to the ReLU, and let $a_d = \sigma(z_d)$ be the activated value before normalization. The normalized weight is then:

$$\phi(d; d^*) = \frac{a_d}{\sum_{d'=0}^{T-1} a_{d'}}. \tag{7}$$

The gradient with respect to d^* can be derived using the chain rule:

$$\begin{aligned}\frac{\partial \phi(d; d^*)}{\partial d^*} &= \frac{\partial}{\partial d^*}\left(\frac{a_d}{\sum_{d'} a_{d'}}\right) \\ &= \frac{1}{(\sum_{d'} a_{d'})^2}\left(\frac{\partial a_d}{\partial d^*}\sum_{d'} a_{d'} - a_d \sum_{d'}\frac{\partial a_{d'}}{\partial d^*}\right).\end{aligned} \tag{8}$$

For the ReLU activation, we have:

$$\frac{\partial a_d}{\partial d^*} = \begin{cases} \frac{\operatorname{sgn}(d-d^*)}{\tau}, & \text{if } z_d > 0, \\ 0, & \text{otherwise,} \end{cases} \tag{9}$$

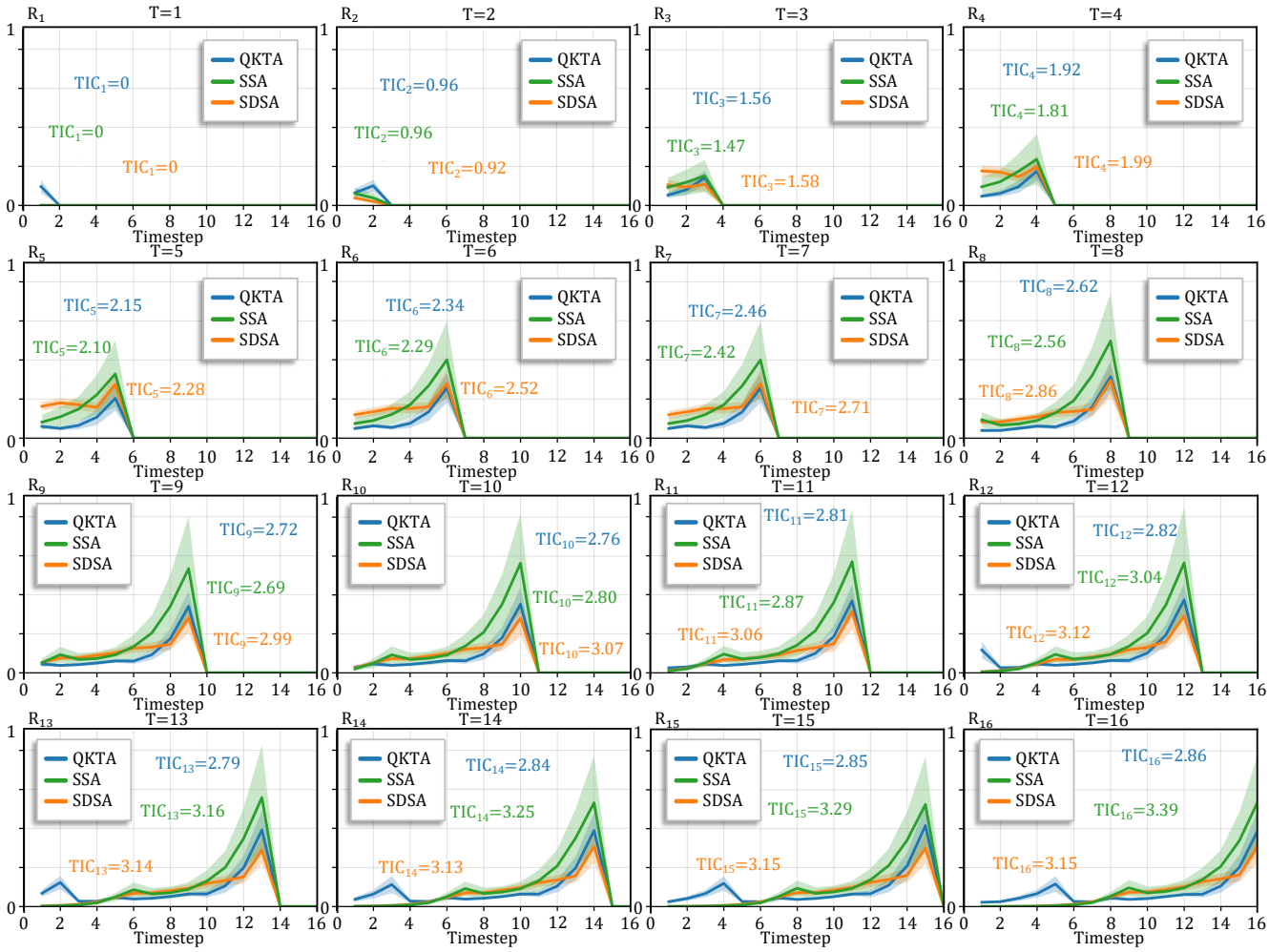

Figure 2. Visualization of temporal reliance distributions for SSA, QKTA, and SDSA attention mechanisms integrated with MD-Mixer across all timesteps, based on the CIFAR10-DVS dataset.

where $\mathrm{sgn}(\cdot)$ is the sign function. This gradient indicates that d^* receives non-zero gradients only from delays d within the support of the triangular distribution, i.e., those satisfying $|d - d^*| < \tau$.

The overall gradient of the loss $\mathcal{L}$ with respect to d^* is obtained by accumulating gradients through the membrane potential dynamics:

$$\frac{\partial \mathcal{L}}{\partial d_j^{*(k)}} = \sum_{t=1}^{T} \sum_{d=0}^{T-1} \frac{\partial \mathcal{L}}{\partial \mathbf{U}_{t,j}} \cdot \frac{\partial \mathbf{U}_{t,j}}{\partial \mathbf{I}_{t,j}} \cdot \alpha_j^{(k)} \cdot \frac{\partial \phi(d)}{\partial d_j^{(k),*}} \cdot \mathbf{X}_{t-d,j}. \tag{10}$$

This formulation reveals that the gradient signal is modulated by three factors: (1) the temporal contribution of each timestep through $\frac{\partial \mathcal{L}}{\partial \mathbf{U}_{t,j}}$, (2) the importance weight $\alpha_j^{(k)}$ associated with the k-th delay path, and (3) the sensitivity of the delay distribution to changes in d^*.

3.2. Temperature Annealing Schedule

The temperature parameter τ controls the sharpness of the delay distribution and undergoes progressive annealing during training to facilitate the soft-to-hard transition. We adopt a squared cosine annealing schedule defined as:

$$\tau(e) = \tau_{\min} + \frac{\tau_{\max} - \tau_{\min}}{2} \left(1 + \cos\left(\frac{\pi e}{E}\right)\right)^2, \tag{11}$$

where e denotes the current epoch, E is the total number of training epochs, $\tau_{\max}$ is the initial temperature, and $\tau_{\min}$ is the final temperature. The squared cosine schedule provides a smooth decay that starts slowly, accelerates in the middle of training, and stabilizes toward the end, allowing the model to first explore diverse temporal dependencies before committing to specific delay values.

In our experiments, we set $\tau_{\max} = T/2$ (half of the total timesteps) and $\tau_{\min} = 0.5$. The choice of $\tau_{\max} = T/2$ ensures that the initial distribution covers a sufficiently broad

range of delays, enabling effective exploration of multi-scale temporal patterns. The final temperature $\tau_{\min} = 0.5$ is chosen to produce a sharp distribution while maintaining numerical stability.

4. Visualizations

4.1. Temporal Reliance Vectors

We visualize the temporal reliance vectors of three representative spiking self-attention mechanisms: SSA [6], QKTA [5], and SDSA [3], on the CIFAR10-DVS dataset, which contains 10,000 samples with 16 timesteps each. For each mechanism, the temporal reliance vectors $\{\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_{16}\}$ are computed by averaging over 1,000 test samples. Figure 1 presents the resulting temporal reliance patterns across all timesteps. For visualization, each temporal reliance vector $\mathbf{R}_q$ is rescaled to $[0, 1]$, and the corresponding TIC value is annotated for each mechanism, revealing clear differences in temporal integration behavior.

Across all three baseline mechanisms, the reliance distributions exhibit sharp peaks at the current timestep, with QKTA showing the most severe concentration. At q=16, QKTA exhibits overwhelming reliance on the current timestep with TIC_{16}=1.74 bits, substantially below the theoretical maximum of $\log_2 16$=4 bits. SSA demonstrates similarly narrow temporal integration, achieving TIC_{16}=2.10 bits, where contributions from earlier timesteps (e.g., $p \leq 8$) remain nearly negligible. SDSA shows marginally broader distributions with TIC_{16}=2.35 bits, yet still falls far short of optimal temporal integration. These results show that existing mechanisms rely predominantly on recent inputs and fail to effectively exploit distant temporal context, leading to weak long-range dependency modeling and potentially inferior performance on temporal classification tasks.

To alleviate these limitations, we incorporate the proposed MD-Mixer into the baseline mechanisms. As shown in Fig. 2, all three variants exhibit broader temporal reliance distributions, accompanied by consistently higher TIC values. In particular, MD-Mixer substantially improves TIC_{16} across all mechanisms: QKTA increases from 1.74 to 2.86 bits, SSA from 2.10 to 3.39 bits, and SDSA from 2.35 to 3.15 bits, indicating richer temporal interactions. More importantly, the resulting reliance distributions assign non-negligible reliance to distant timesteps, suggesting improved long-range temporal integration. These quantitative improvements are consistent with the performance gains observed on classification tasks. The consistent enhancement across different baseline architectures demonstrates the generality of MD-Mixer and highlights the importance of explicit temporal memory mechanisms for spiking self-attention.

4.2. Delay Optimization

We visualize the distribution of learned delay and adaptive weights of MD-Mixer for SDT V1 on the N-Caltech101 dataset in Fig. 3. The model exhibits a clear preference for short delays, with the largest percentage concentrated on adjacent temporal steps ($d \leq 3$). This suggests that local temporal dependencies play a dominant role in the learned representation. When multiple layers are stacked, these local temporal interactions can be composed hierarchically to yield a larger effective temporal receptive field. The lower histogram further shows that MD-Mixer does not simply sum the delay branches, but instead dynamically reweights different channel-delay combinations according to their relevance. In particular, a non-trivial portion of the weights lies close to zero, implying that some channel-delay responses make only marginal contributions to the final representation. This observation further implies that pruning strategies could be incorporated in future work to improve computational efficiency.

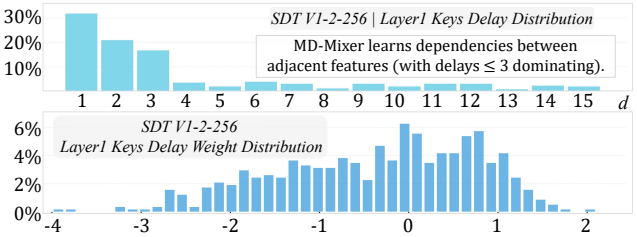

Figure 3. Learned delay and weight distribution (N-Caltech101).

5. Experimental Configurations

We conduct comprehensive experiments on four representative Spiking Transformer architectures, namely Spikformer [6], QKFormer [5], Spike-Driven Transformer V1 [3], and Spike-Driven Transformer V2 [4], to evaluate their performance on both static image datasets and neuromorphic event-based datasets. To ensure a rigorous and fair comparison, all experiments follow the original training protocols and hyperparameter settings reported in their respective papers, with modifications made only when necessary to accommodate dataset-specific characteristics or computational constraints. Model configurations are denoted using the *depth-channels* format, where *depth* denotes the number of stacked Transformer blocks and *channels* specifies the feature dimension.

Table 1. Training hyperparameters for Spikformer across different datasets.

Hyperparameters	**CIFAR-10**	**CIFAR-100**	**CIFAR10-DVS**	**N-Caltech101**
Timestep	32	32	16	16
Input size	32×32	32×32	64×64	64×64
Batch size	256	256	16	16
Training epochs	500	500	500	500
Optimizer	AdamW	AdamW	AdamW	AdamW
Initial learning rate	5e-3	5e-3	5e-3	5e-3
LR schedule	Cosine	Cosine	Cosine	Cosine
Architecture	4-256	4-256	2-256	2-256

Table 2. Training hyperparameters for QKFormer across different datasets.

Hyperparameters	**CIFAR-10**	**CIFAR-100**	**CIFAR10-DVS**	**N-Caltech101**
Timestep	32	32	16	16
Input size	32×32	32×32	64×64	64×64
Batch size	256	256	16	16
Training epochs	400	400	105	200
Optimizer	AdamW	AdamW	AdamW	AdamW
Initial learning rate	5e-4	5e-4	5e-3	1e-3
LR schedule	Cosine	Cosine	Cosine	Cosine
Architecture	4-384	4-384	2-256	2-256

Table 3. Training hyperparameters for Spike-Driven Transformer V1 across different datasets.

Hyperparameters	**CIFAR-10**	**CIFAR-100**	**CIFAR10-DVS**	**N-Caltech101**	**ImageNet**
Timestep	32	32	16	16	4
Input size	32×32	32×32	64×64	64×64	224×224
Batch size	32	32	32	32	32
Training epochs	300	300	200	200	300
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW
Initial learning rate	3e-4	3e-4	7.5e-3	7.5e-3	1.5e-3/1e-3
LR schedule	Cosine	Cosine	Cosine	Cosine	Cosine
Architecture	4-256	4-256	2-256	2-256	8-512/8-768

Table 4. Training hyperparameters for Spike-Driven Transformer V2 across different datasets.

Hyperparameters	**UCF101-DVS**	**HMDB51-DVS**	**ImageNet**
Timestep	16	16	4
Input size	64×64	64×64	224×224
Batch size	16	16	32
Training epochs	200	200	200
Fine-tuning epochs	–	–	20
Optimizer	LAMB	LAMB	LAMB
Training learning rate	1e-4	1e-4	6e-4
Fine-tuning learning rate	–	–	2e-5
LR schedule	Cosine	Cosine	Cosine
Architecture	4-256	4-256	8-512/8-768

References

[1] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:331, 2018. 1

[2] Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, Di He, and Zhouchen Lin. Online training through time for spiking neural networks. *Advances in neural information processing systems*, 35:20717–20730, 2022. 1

[3] Man Yao, Jiakui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer. *Advances in neural information processing systems*, 36:64043–64058, 2023. 4

[4] Man Yao, JiaKui Hu, Tianxiang Hu, Yifan Xu, Zhaokun Zhou, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer v2: Meta spiking neural network architecture inspiring the design of next-generation neuromorphic chips. *arXiv preprint arXiv:2404.03663*, 2024. 4

[5] 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 qk attention. *Advances in Neural Information Processing Systems*, 37:13074–13098, 2024. 4

[6] Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng YAN, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer. In *The Eleventh International Conference on Learning Representations*, 2023. 4