STEP: A Unified Spiking Transformer Evaluation Platform for Fair and Reproducible Benchmarking

Sicheng Shen[1,2,4,★] **Dongcheng Zhao**[1,3,★] **Linghao Feng** [1]
Zeyang Yue[1,5] **Jindong Li**[1] **Tenglong Li**[1]
Guobin Shen[1] **Yi Zeng**[1,3,†]
[1] BrainCog Lab, CASIA [2] School of Future Tech., UCAS [3] Long-term AI
[4] Zhongguancun Academy [5] Beihang University
★ Equal contribution † Corresponding author
shensicheng2024@ia.ac.cn yi.zeng@ia.ac.cn

Abstract

Spiking Transformers have recently emerged as promising architectures for combining the efficiency of spiking neural networks with the representational power of self-attention. However, the lack of standardized implementations, evaluation pipelines, and consistent design choices has hindered fair comparison and principled analysis. In this paper, we introduce **STEP**, a unified benchmark framework for Spiking Transformers that supports a wide range of tasks, including classification, segmentation, and detection across static, event-based, and sequential datasets. STEP provides modular support for diverse components such as spiking neurons, input encodings, surrogate gradients, and multiple backends (e.g., SpikingJelly, BrainCog). Using STEP, we reproduce and evaluate several representative models, and conduct systematic ablation studies on attention design, neuron types, encoding schemes, and temporal modeling capabilities. We also propose a unified analytical model for energy estimation, accounting for spike sparsity, bitwidth, and memory access, and show that quantized ANNs may offer comparable or better energy efficiency. Our results suggest that current Spiking Transformers rely heavily on convolutional frontends and lack strong temporal modeling, underscoring the need for spike-native architectural innovations. **The full code is available at:** https://github.com/Fancyssc/STEP.

1 Introduction

Spiking Neural Networks (SNNs) are a biologically inspired paradigm that simulate neural information processing via discrete spikes. These networks excel not only at static image tasks but also in modeling dynamic and temporally structured data [1]. Their event-driven nature contributes to high energy efficiency and strong biological plausibility. However, applying SNNs to deep learning architectures—particularly Transformers—remains challenging due to their non-differentiability, limited scalability, and training instability.

In parallel, Artificial Neural Networks (ANNs) have seen tremendous advances through architectural innovations. ResNet [2] introduced residual learning to ease optimization in deep networks, while Recurrent Neural Networks (RNNs) captured sequential dependencies. The Transformer architecture [3] unified these advances by leveraging self-attention, enabling efficient parallel modeling of long-range dependencies. Vision Transformer (ViT) [4] further demonstrated the potential of attention mechanisms in visual tasks. Drawing inspiration from these architectures, the SNN community has proposed Spiking ResNet [5], SEW-ResNet [6], and spiking RNN variants [7, 8]. Recently, attention-based spiking models such as Spikformer [9], QKFormer [10], and SpikingResformer [11] have emerged.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Track on Datasets and Benchmarks.

The Spike-Driven Transformer series [12, 13, 14] improves both efficiency and scalability, enabling applications in image segmentation and object detection.

Despite advancements, several key challenges persist in Spiking Transformers (STs). First, the performance gap between STs and traditional ANNs remains unclear, especially regarding their unique advantages on temporal or relatively complicated data. A systematic evaluation across diverse datasets—static (e.g., ImageNet), event-driven (e.g., DVS-CIFAR10), and sequential (e.g., SCIFAR10)—is essential for assessing their potential. Second, STs consist of multiple interacting components, including spike encoders, neuron models, surrogate gradients, attention modules, and MLP heads, yet the contribution of each module is underexplored. Module-wise ablation is critical for understanding trade-offs and guiding optimization. Third, while SNNs inherently offer energy benefits through sparse, binary spike-based computation, direct comparisons to quantized Transformers are scarce. Quantifying the energy-performance trade-off is necessary to assess the practical utility of spiking models. Moreover, inconsistencies across development frameworks, such as SpikingJelly [15], BrainCog [16], and BrainPy [17], further hinder progress by complicating reproducibility, hyperparameter tuning, and fair model comparison. Currently, no unified platform exists for evaluating Spiking Transformers across tasks like classification, segmentation, and detection.

To address these challenges, we introduce the **Spiking Transformer Evaluation Platform (STEP)**, a unified benchmarking framework for building, evaluating, and comparing Spiking Transformers. STEP integrates representative implementations, supports modular component replacement, and enables consistent evaluation across visual tasks. It provides both training-from-scratch and pretraining–finetuning pipelines, and supports integration with backends such as SpikingJelly, BrainCog, and BrainPy. Moreover, leveraging MMSegmentation [18] and MMDetection [19], STEP extends support to dense prediction tasks. Our main contributions are as follows:

- We propose a unified benchmarking framework (STEP) for Spiking Transformers, integrating existing implementations to ensure consistency and reproducibility in evaluation.
- We design module-wise ablation experiments to evaluate the contribution of core components, providing guidance for architectural optimization.
- We investigate energy–performance trade-offs between Spiking and quantized Transformers, highlighting the unique advantages of spike-based computation.

2 Preliminary

Spiking Transformers (STs) integrate the sparse, event-driven processing of Spiking Neural Networks (SNNs) with the scalable representation power of Transformer architectures (Fig. 1). This hybrid design enables efficient handling of static and dynamic data, benefiting from both energy efficiency and long-range contextual modeling. Key components of STs include spike-based input encoding, spiking neurons, patch-wise tokenization, position embeddings, spiking self-attention (SSA), and task-specific prediction heads.

SNN Input Encoding To enable spike-based processing, input signals are transformed into temporal spike trains via encoding schemes such as direct, rate, time-to-first-spike (TTFS), and phase encoding [20, 21, 22]. A detailed overview of encoding methods is provided in Appendix A.1.

Spiking Neurons Spiking neurons transmit information via discrete spikes triggered by membrane potential dynamics. The Leaky Integrate-and-Fire (LIF) model [23] is widely used due to its simplicity and biological plausibility:

$$V[t] = V[t-1] + \frac{1}{\tau}(X[t] - V[t-1]), \quad \text{if } V[t] \geq V_{th}, \text{ emit spike and reset.} \tag{1}$$

Variants like PLIF [24] and GLIF [25] enhance adaptability with learnable decay or gated mechanisms. Further details are provided in Appendix A.2.

Spiking Self-Attention SSA adapts the attention mechanism to the spike domain, enabling long-range dependencies without softmax. Given input X, SSA computes spiking queries, keys, and values:

$$Q = SN_Q(W_Q^\top X), \quad K = SN_K(W_K^\top X), \quad V = SN_V(W_V^\top X), \quad SSA = SN(QK^\top V) \cdot \text{scale} \tag{2}$$

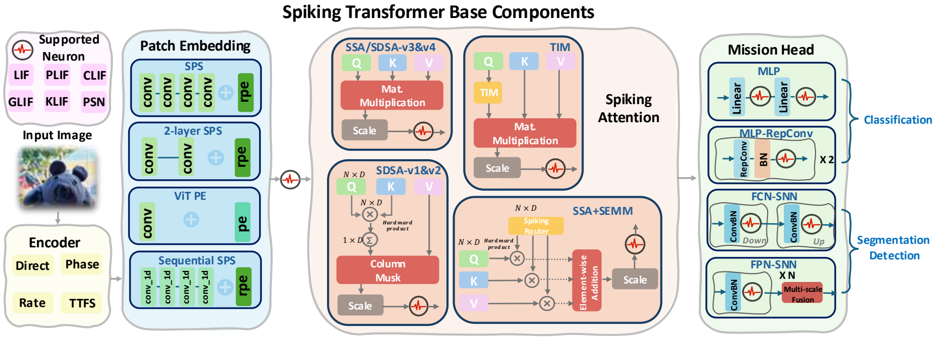

Figure 1: Unified Spiking Transformer Framework with Flexible Encoding, Attention Modules, and Application-specific Heads

Here, $SN(\cdot)$ denotes selected spiking neuron. This mechanism preserves temporal sparsity while capturing global context. See Appendix A.4 for SSA variants.

Other Modules Patch-based tokenization (Spiking Patch Splitting) enables scalable input decomposition, while Position Embeddings inject spatial/temporal order into spike sequences. Final predictions are made via MLP heads adapted for classification, detection, or segmentation.

Recent Advancements Recent ST models propose lightweight attention [12, 13], hierarchical designs (e.g., QKFormer [10]), and multi-task heads (e.g., FCN [26], FPN [27]) to enhance performance across modalities. These improvements drive STs toward practical deployment while retaining neuromorphic efficiency.

3 Spiking Transformer Benchmark

Building on the core components of Spiking Transformers, we present the **Spiking Transformer Evaluation Platform (STEP)**—a unified, extensible benchmark designed to standardize evaluation and accelerate research in this emerging field. STEP supports a wide range of tasks, including classification, segmentation, and object detection, and enables fair, reproducible comparisons across different models and datasets.

STEP is built around four key principles (Fig. 2): (1) *modularity*, allowing flexible integration of neuron models, encodings, and attention mechanisms; (2) *dataset compatibility*, supporting static, event-based, and sequential inputs; (3) *multi-task adaptation*, with pipelines for vision tasks beyond classification; and (4) *backend interoperability*, enabling seamless deployment across major SNN frameworks such as SpikingJelly, BrainCog, and BrainPy.

Together, these design goals make STEP a robust foundation for developing, benchmarking, and extending Spiking Transformers. It not only reduces implementation overhead but also helps identify architectural bottlenecks and promotes best practices, fostering progress toward more generalizable and practical neuromorphic models. For detailed usage instructions, please refer to the Appendix C.

3.1 Flexible and Modular Architecture

The Spiking Transformer Benchmark is designed with a modular and extensible architecture that supports seamless integration across various backend frameworks. It accommodates diverse neuron models, encoding schemes, and surrogate gradients, allowing researchers to tailor the benchmark to specific design requirements or research goals. A unified training pipeline ensures consistent evaluation protocols, while the low-coupling structure enables independent modification of core components such as patch embedding, attention mechanisms, and MLP heads.

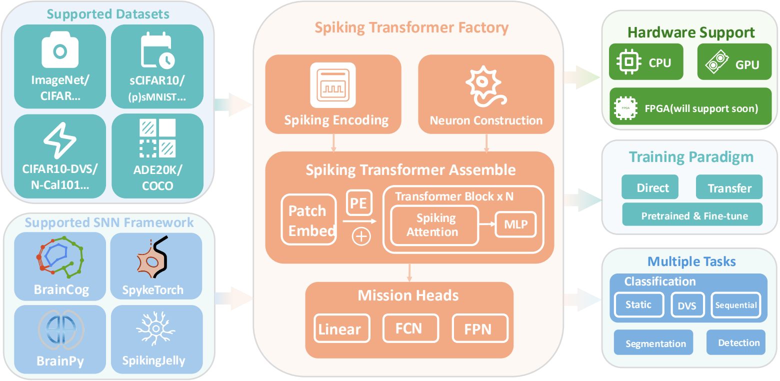

Figure 2: System Architecture of STEP as a Unified Benchmark for Spiking Transformer Development and Evaluation

3.2 Broad Dataset Compatibility

Our benchmark supports a wide spectrum of datasets, encompassing static (e.g., ImageNet [28]), event-based (e.g., DVS-CIFAR10 [29]), and sequential inputs. It also integrates sequential classification tasks to assess temporal modeling capabilities. For dense prediction tasks, we provide plug-and-play support for SNN-adapted segmentation and detection models, such as FCN [26] and FPN [27], based on MMSeg and MMDet toolchains. 3D point cloud and DVS video detection(PKU-DAVIS-SOD [30]) are also supported in the latest version. However, since these datasets are not compatible with most models, we do not conduct unified evaluations on them. Some experimental results can be found in the Appendix B.5.

3.3 Multi-task Adaptation

While early Spiking Transformers (e.g., Spikformer [9], TIM [31]) were largely limited to classification, recent efforts such as Spike-Driven Transformer V2 have expanded their scope to include dense vision tasks. Our benchmark extends this trajectory by enabling flexible configuration across classification, segmentation, and detection pipelines within a unified framework.

3.4 Backend-Agnostic Integration

To enhance accessibility and reusability, our benchmark supports multiple backends such as SpikingJelly [15], BrainCog [16], and BrainPy. This backend-agnostic design ensures broad compatibility and enables cross-framework reproducibility. Overall, the framework is robust, extensible, and task-agnostic, offering a solid foundation for developing and evaluating Spiking Transformer architectures.

4 Experiment

To ensure fair and reliable evaluation, we reproduce several representative Spiking Transformer models under a unified training setup. This section details the experimental protocol and presents the reproduced results on benchmark datasets.

4.1 Reproduction

Tab. 1 presents our reproduced results on CIFAR-10 and CIFAR-100 [36]. All models are trained using the same optimizer, learning rate, batch size (unless otherwise constrained), training epochs, and random seed. Experiments are conducted on NVIDIA A100 GPUs with 40GB memory.

Overall, our reproduced results are consistent with the original papers. Some models, such as QKFormer, even outperform their reported results, suggesting strong reproducibility. Discrepancies stem mainly from (i) implementation differences, e.g., SpikingResformer originally uses transfer learning, while our setup employs end-to-end training; and (ii) memory limitations, e.g., SGLFormer

Table 1: Reproduced top-1 accuracy (%) of Spiking Transformer models on CIFAR-10 and CIFAR-100. *: SGLFormer uses a reduced batch size (16) due to high memory demand. **: SpikingResformer was originally trained with transfer learning; we instead use end-to-end training.

Model	Batch-Size	Step	Epoch	CIFAR10 (Acc@1)	CIFAR100 (Acc@1)
Spikformer [9]	128	4	400	95.12 (95.51)	77.37 (78.21)
SDT [12]	128	4	400	95.77 (95.60)	78.29 (78.40)
QKFormer [10]	128	4	400	96.24 (96.18)	79.72 (81.15)
Spikingformer [32]	128	4	400	95.53 (95.81)	79.12 (79.21)
Spikformer + SEMM [33]	128	4	400	94.98 (95.78)	77.59 (79.04)
Spiking Wavelet [34]	128	4	400	95.31 (96.10)	76.99 (79.30)
SGLFormer [35]*	16	4	400	95.88 (96.76)	80.61 (82.26)
SpikingResformer [32]**	128	4	400	95.69 (97.40)	79.45 (85.98)

requires a smaller batch size. To ensure fairness, we avoid dataset- or model-specific tuning and apply a uniform experimental protocol across all baselines. The metrics commonly used to evaluate the framework and reproduction robustness can be found in the Appendix B.4.

4.2 Experiments on More Complex Tasks

To further evaluate the scalability and task generalization of Spiking Transformer models, we test their performance on ImageNet-1K for large-scale classification, ADE20K for semantic segmentation and COCO for object detection, all of which are significantly more complex than CIFAR-level datasets.

4.2.1 Classification: ImageNet-1K

For ImageNet-1K [28] we evaluate only Spikformer [33] and QKFormer [10]: the former is the seminal Spiking-Transformer baseline, while the latter introduces a hierarchical pyramid and currently delivers SOTA accuracy among SNN-based Transformers. Concentrating our limited GPU budget on these two "end-points" lets us cover the full architectural spectrum without incurring the prohibitive cost of training several similar intermediate models. Because ImageNet-1K is orders of magnitude larger and more complex than CIFAR-10/100—and because Spikformer and QKFormer differ greatly in parameter count and memory footprint—forcing a single batch size and epoch schedule would either overflow A100 GPU memory or demand untenable compute. We therefore keep each model's published regime (QKFormer: 200 epochs × 32/GPU; Spikformer: 300 epochs × 24/GPU), while unifying every other hyper-parameter under a single script; the differing batch sizes and epoch counts are thus an intentional, resource-aware decision rather than an oversight.

The shortfall in our QKFormer accuracy comes from two choices: we evaluated the compact variant and trained every model with one unified script that omits architecture-specific optimisations. This inevitably costs QKFormer a few points, yet the results still validate our reproduction, and we will extend the same benchmark to the remaining Spiking Transformers on ImageNet.

4.2.2 Segmentation: ADE20K

Table 2: Reproduced performance on ImageNet-1K and ADE20K without pretraining.

Model	ImageNet-1K (Classification)				ADE20K (Segmentation)		
	Batch Size	Step	Epochs	Acc@1	aAcc	mIoU	mAcc
QKFormer [10]	256	4	200	73.88	-	-	-
Spikformer [32]	192	4	300	73.69	69.80	23.51	31.43
Spikformer + SEMM [33]	-	-	-	-	63.41	13.13	19.76
SDT [12]	384	4	300	71.96	63.45	12.08	17.17

For semantic segmentation on ADE20K [37], only SDT [12] had been previously evaluated. We conduct a fair comparison by retraining Spikformer [33]and SEMM [38] **without pretraining**. Interestingly, both Spikformer variants outperform SDT under identical settings, despite SDT's original paper reporting strong performance with pretraining. This implies that with appropriate

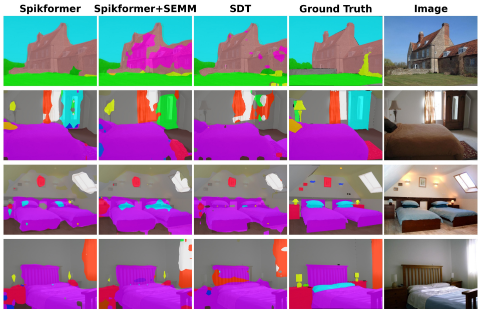

Figure 3: Segmentation predictions on ADE20K for three Spiking Transformer variants.

Table 3: Top-1 accuracy (%) of different neuron types on CIFAR-10.

Model	LIF	CLIF	GLIF	KLIF	PLIF
Spikformer [9]	95.12	95.38	95.41	95.85	**96.06**
SDT [12]	95.77	95.49	95.45	95.63	**95.91**
Spikformer + SEMM [33]	94.98	95.44	**95.78**	95.59	95.66

Table 4: SDTv2 detection result on COCO. Step:1; Epoch:10.

Pre-training	bbox mAP@0.5	segm mAP@0.5
No	1.7	1.6
Yes	10.5	10.4

initialization strategies, Spikformer-based models could potentially surpass existing baselines in dense prediction tasks. The segmentation result can be viewed in Fig. 3

These results demonstrate that Spiking Transformers, when carefully trained, are capable of scaling to more complex tasks beyond image classification, including semantic segmentation, and are promising candidates for broader real-world neuromorphic applications.

4.2.3 Detection: COCO

Object detection requires simultaneous localisation and classification across diverse scales, a challenge naturally addressed by multi-resolution features. Among existing Spiking Transformers, only **SDTv2** produces genuine multi-scale outputs, making it the sole candidate for COCO. Training SDTv2 from scratch yields poor box regressors, whereas ImageNet pre-training boosts mAP by an order of magnitude (Tab. 4 & Fig. 4). Unlike segmentation, where models converge without priors, detection proves highly sensitive to object-level cues and foreground–background balance. Thus, effective spiking detectors must combine multi-scale backbones with strong pre-training. These findings inspire future spiking designs with built-in pyramids and large-scale (self-supervised) pre-training to bridge the gap with ANNs and enable energy-efficient event-driven detection. More detailed results are reported in Appendix B.3.

5 Analysis

Transformers' ability to model sequential dependencies has recently been questioned, particularly regarding the actual benefits of sparse attention in SNN contexts. Among existing models, Spikformer first introduced attention mechanisms into SNNs, while SDT significantly reduced their computational complexity. Many subsequent works, including Spikformer+SEMM (which incorporates a Mixture of Experts with minimal modification), are derived from or inspired by these two. We focus our analysis on these three representative models.

5.1 Neuron Model Evaluation

To investigate the impact of different spiking neuron types on model performance, we replace the default LIF neuron with five widely used variants: PLIF [24], CLIF [39], GLIF [25], and KLIF [40]. These models extend the basic LIF neuron [23] by incorporating enhancements such as learnable time constants, gating mechanisms, and surrogate gradient improvements. To quantify the influence of neuron choice, we replace the default LIF cell with four mainstream variants—PLIF [24], CLIF [39], GLIF [25], and KLIF [40]. Each variant augments the canonical LIF formulation [23] with additional biological or optimization benefits, ranging from a learnable membrane constant (PLIF) to gated internal states (GLIF) and surrogate-gradient refinements (CLIF, KLIF). Detailed explaination can be found in Appendix A.2.

Tab. 3 reports consistent accuracy gains across three backbone architectures once these enhanced neurons are introduced. PLIF delivers the largest improvement—surpassing even architectural upgrades on Spikformer—yet it adds only one scalar parameter. We attribute the gain to richer, more biologically plausible membrane dynamics that encourage sparse, spike-driven learning.

These results indicate that Spiking Transformers lean more on intrinsic neuron dynamics than on explicit temporal modules. Progress therefore calls for biologically faithful yet efficient additions—such as dendritic processing or multi-compartment cells—perhaps embedded in hybrid recurrent-spiking frameworks.

5.2 Sequence Modeling

Recent studies [41] question the ability of standard Transformers to model long-range temporal dependencies, prompting alternatives like Spiking SSM [42]. Datasets such as sCIFAR [43] and (p)sMNIST [44] serialize 2D images into 1D sequences, emphasizing temporal structure. Spiking SSM processes inputs at the pixel level (e.g., 784 steps for a full MNIST image), incurring high computational costs. To adapt Spiking Transformers for serialized inputs, we replace 2D convolutions in the SPS module with 1D convolutions.

Table 5: Top-1 accuracy (%) of selected models on sequential image classification datasets. Batch size = 128, epochs = 400, steps = 4. *: Original ViT; **: ViT with 4-layer-conv embedding.

Model	SNN	sMNIST	psMNIST	sCIFAR
FlexTCN [45]	No	99.62	98.63	80.82
SMPConv [46]	No	**99.75**	**99.10**	84.86
LMUformer [47]	No	-	98.55	-
ViT [4] *	No	98.00	97.73	74.95
ViT[4] + SPS **	No	99.19	98.19	**85.62**
SpikingSSM [42]	Yes	99.60	98.40	-
SpikingLMUformer [47]	Yes	-	97.92	-
Spikformer [9]	Yes	98.84	97.97	84.26
SDT [12]	Yes	98.77	97.80	82.31
Spikformer + SEMM [33]	Yes	99.33	98.46	85.61

As shown in Tab. 5, SNN-based Spiking Transformers lag behind ANN counterparts like ViT+SPS and SMPConv, even with MoE enhancements in Spikformer+SEMM. This suggests that spike-based attention mechanisms are more suited for spatial rather than temporal modeling. We hypothesize that performance limitations stem from the restricted number of training steps and sparse neuron activations, which weaken temporal expressiveness. Future work should explore spatiotemporal attention designs and biologically inspired mechanisms like spike-timing-dependent plasticity (STDP) to improve temporal modeling without excessive computational cost.

5.3 Encoding Schemes

RGB inputs can be converted to spikes through four encoding schemes: direct, phase, rate, and TTFS [20, 21, 22]. As shown in Tab. 6, direct encoding—being lossless and repeating the full image at each timestep—aligns with current Spiking Transformers that compute attention independently, yielding the highest Top-1 accuracy. In contrast, the sparser phase, rate, and TTFS encodings reduce spike density and spatial coherence, leading to lower accuracy and emphasizing the need for temporally aware attention or recurrent designs.

5.4 Sparse Attention Analysis

In Sec. 5.3, we observed that Spiking Transformers struggle to model temporal dependencies. Here, we further examine whether their attention mechanisms meaningfully contribute to spatial feature extraction.

Randomized Attention. To ablate the role of attention, we fix the Q and K branches to randomly initialized, frozen weights, while keeping V trainable for gradient propagation:

$$Q = \text{LIF}(W^{Q}_{\text{detach}} X), \quad K = \text{LIF}(W^{K}_{\text{detach}} X), \quad V = \text{LIF}(W^{V} X) \tag{3}$$

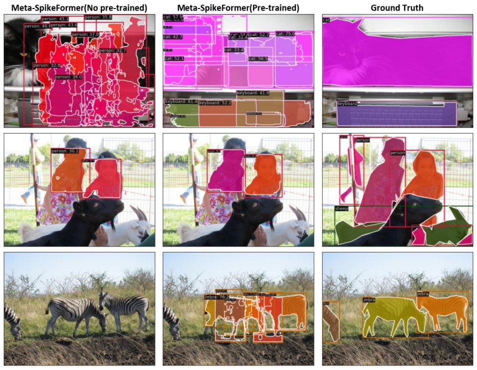

Figure 4: Result of SDTv2 on COCO datasets.

Table 6: Top-1 accuracy (%) of different encoding methods on CIFAR-10. Batch size: 128, Epoch: 400, Step: 4.

Model	Direct	Phase	Rate	TTFS
Spikformer [9]	**95.12**	82.75	82.83	82.10
SDT [12]	**95.77**	85.37	83.77	84.30
Spikformer + SEMM [33]	**94.98**	85.81	83.04	83.37

Table 7: Top-1 accuracy (%) with SDSA-v3 under varying SPS depths.

Model	SPS (4 conv)	SPS-2conv	SPS-1conv
Spikformer [9]	95.57	93.43	89.97
SDT [12]	96.38	94.68	87.33
Spikformer + SEMM [33]	95.83	93.37	84.95

We apply this to three representative models (Spikformer, SDT, and Spikformer+SEMM), and include ViT as an ANN-based baseline. As shown in Fig. 8, Spiking Transformers maintain performance under randomized attention (drop $< 0.35\%$), with Spikformer+SEMM even slightly improving. In contrast, ViT suffers a notable drop, indicating its strong reliance on attention.

Reduced Convolutional Depth in SPS. We next evaluate model robustness under reduced convolutional depth in the SPS module. When decreasing SPS from four to two and one layers, performance deteriorates sharply across all models. With only one conv layer, models behave like pure attention-based Spiking Transformers and fail to match baseline accuracy—highlighting the dominant role of convolution in feature extraction.

Table 8: Comparison of Acc@1 for Different Model Configurations on CIFAR-10.

Model	Original	Random_Attn	SPS (1 Conv)	SPS (2 Conv)
Spikformer	95.12	94.96	78.21	91.92
SDT	95.77	95.45	77.34	94.03
Spikformer+SEMM	94.98	95.57	89.24	93.33
ANN_ViT	90.89	88.46	—	—

Replacement with SDSA-v3. To test whether stronger attention can compensate for weaker convolutional backbones, we replace SSA with SDSA-v3 [14, 48], where QKV are generated using depthwise separable convolutions:

$$W = \textbf{SSA+(SEMM)} : \text{Linear}(\cdot);\ \textbf{SDSA} : \text{ConvBN}(\cdot);\ \textbf{SDSA-V3} : \text{BN}(\text{SepConv}(\cdot)) \tag{4}$$

Even with SDSA-v3, performance remains positively correlated with convolutional depth (Tab. 7). While SDSA-v3 reduces the performance gap, it does not eliminate reliance on convolution. These findings suggest that current spike-based attention mechanisms contribute limited spatial modeling capacity, with most representational power still residing in the convolutional frontend.

5.5 Energy Efficiency Modeling

Energy modeling in SNNs traditionally estimates cost based on the number of accumulate (AC) operations, whereas for ANNs, it relies on multiply-accumulate (MAC) operations. However, we argue that current methodologies overlook two critical aspects:

- **Quantized ANNs are underestimated in efficiency.** Bit-serial execution in low-bitwidth ANNs [49, 50] can transform MACs into sequences of ACs, which can exploit bit-level sparsity to skip ineffectual operations—similar to spike sparsity in SNNs. This makes quantized ANNs significantly more efficient than previously assumed.
- **Memory access energy is often ignored.** Previous comparisons often overlook the energy cost associated with on-chip and off-chip memory accesses. In SNNs, high-precision membrane potentials must be maintained and updated throughout multiple time steps, necessitating frequent accesses. In contrast, ANNs only require writing back quantized activations, which has less memory burden. This omission in existing energy models can result in an overestimation of the energy efficiency of SNNs relative to ANNs.

To address these gaps, we propose a analytical framework that models both spiking and quantized neural networks shown in Tab. 9, and Tab. 10 presents an quantitive comparison. While the spiking transformer show a small advantage in compute efficiency over the quantized transformer, its overall energy consumption is unexpectedly higher once memory access is factored in.

Table 9: Energy analysis modeling. F_{Conv} and F_{Mlp} denote FLOPs of Conv and MLP modules in ANNs. B is the quantized bit-width in quantized Transformers; T is the time steps in spiking Transformers. R_s (firing rate) and R_b (bit rate) represent spike sparsity and bit-level sparsity of the quantized activation. $E_{Mac} = 4.6pJ$, $E_{Ac} = 0.9pJ$, and $E_{Mem} = 3.12pJ$ denote energy per MAC, AC, and memory access (per bit energy access from a 16MB cache), respectively [51].

Module	**Op.**	**Type**	**Vanilla Transformer**	**Quantized Transformer**	**Spiking Transformer**
SPS	Conv	Compute	$E_{Mac}F_{Conv}$	$BR_b \cdot E_{Ac}F_{Conv}$	$TR_s \cdot E_{Ac}F_{Conv}$
		Memory	$32 \cdot E_{Mem}C_oHW$	$B \cdot E_{Mem}C_oHW$	$32T \cdot E_{Mem}C_oHW$
Self Attention	Q,K,V	Compute	$E_{Mac}3ND^2$	$BR_b \cdot E_{Ac}3ND^2$	$TR_s \cdot E_{Ac}3ND^2$
		Memory	$32 \cdot E_{Mem}3ND$	$B \cdot E_{Mem}3ND$	$32T \cdot E_{Mem}3ND$
	f(Q,K,V)	Compute	$E_{Mac}2N^2D$	$BR_b \cdot E_{Ac}2N^2D$	$TR_s \cdot E_{Ac}ND$
		Memory	$32 \cdot E_{Mem}2N^2$	$B \cdot E_{Mem}2N^2$	$32T \cdot E_{Mem}ND$
	Linear	Compute	$E_{Mac}F_{Mlp}$	$BR_b \cdot E_{Ac}F_{Mlp}$	$TR_s \cdot E_{Ac}F_{Mlp}$
		Memory	$32 \cdot E_{Mem}C_o$	$B \cdot E_{Mem}C_o$	$32T \cdot E_{Mem}C_o$
MLP	Linear	Compute	$E_{Mac}F_{Mlp}$	$BR_b \cdot E_{Ac}F_{Mlp}$	$TR_s \cdot E_{Ac}F_{Mlp}$
		Memory	$32 \cdot E_{Mem}C_o$	$B \cdot E_{Mem}C_o$	$32T \cdot E_{Mem}C_o$

Table 10: Energy analysis comparison.

Model	**Param**	**Neuron**	**Compute**	**Mem**	**Total**
Transformer-8-512 Float	29.68M	14M	41.77mJ	1.39mJ	43.16mJ
Transformer-8-512 Quant	29.68M	14M	16.34mJ	0.17mJ	16.51mJ
SpikingTransformer-8-512	29.68M	14M	11.57mJ	5.59mJ	17.16mJ

6 Future Work

While recent progress in Spiking Transformers has mainly aimed to boost task performance, our results indicate that directly transplanting ANN modules like attention or convolution overlooks key SNN principles. Future work should move beyond performance-oriented adaptation and draw from neuroscience, exploring mechanisms such as dendritic computation, STDP, and temporal coding to design spike-native architectures that are more efficient, robust, and interpretable.

7 Conclusion

In this work, we present STEP, a unified benchmarking framework for Spiking Transformers, aiming to standardize evaluation across architectures, datasets, and tasks. STEP integrates diverse implementations under a consistent pipeline, supporting classification, segmentation, and detection on both static and event-based datasets. Through extensive experiments, we reproduced and compared multiple representative models, revealing that current Spiking Transformers rely heavily on convolutional preprocessing while benefiting only marginally from attention mechanisms. Our module-wise ablation further demonstrates that the choice of spiking neuron model and input encoding has a non-trivial impact on final performance, highlighting the importance of biologically inspired design. We also revisited energy efficiency comparisons between SNNs and ANNs. By introducing a unified analytical model that incorporates compute sparsity, bitwidth effects, and memory access costs, we showed that quantized ANNs may be more competitive than previously assumed, urging more careful benchmarking. Taken together, our study highlights the need for deeper integration of neuroscience principles and task-aligned architectural innovations. We hope STEP can serve as a foundation for building truly spike-native Transformers that are efficient, robust, and biologically grounded.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (Grant No. 62406325).

References

[1] Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[5] Yangfan Hu, Huajin Tang, and Gang Pan. Spiking deep residual networks. *IEEE Transactions on Neural Networks and Learning Systems*, 34(8):5200–5205, 2021.

[6] Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. *Advances in Neural Information Processing Systems*, 34:21056–21069, 2021.

[7] Yannan Xing, Gaetano Di Caterina, and John Soraghan. A new spiking convolutional recurrent neural network (scrnn) with applications to event-based hand gesture recognition. *Frontiers in neuroscience*, 14:590164, 2020.

[8] Qi Xu, Xuanye Fang, Yaxin Li, Jiangrong Shen, De Ma, Yi Xu, and Gang Pan. Rsnn: Recurrent spiking neural networks for dynamic spatial-temporal information processing. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 10602–10610, 2024.

[9] Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng Yan, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer. *arXiv preprint arXiv:2209.15425*, 2022.

[10] 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. *arXiv preprint arXiv:2403.16552*, 2024.

[11] Xinyu Shi, Zecheng Hao, and Zhaofei Yu. Spikingresformer: bridging resnet and vision transformer in spiking neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5610–5619, 2024.

[12] 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.

[13] 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.

[14] Man Yao, Xuerui Qiu, Tianxiang Hu, Jiakui Hu, Yuhong Chou, Keyu Tian, Jianxing Liao, Luziwei Leng, Bo Xu, and Guoqi Li. Scaling spike-driven transformer with efficient spike firing approximation training. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[15] Wei Fang, Yanqi Chen, Jianhao Ding, Zhaofei Yu, Timothée Masquelier, Ding Chen, Liwei Huang, Huihui Zhou, Guoqi Li, and Yonghong Tian. Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, 9(40):eadi1480, 2023.

[16] Yi Zeng, Dongcheng Zhao, Feifei Zhao, Guobin Shen, Yiting Dong, Enmeng Lu, Qian Zhang, Yinqian Sun, Qian Liang, Yuxuan Zhao, et al. Braincog: A spiking neural network based, brain-inspired cognitive intelligence engine for brain-inspired ai and brain simulation. *Patterns*, 4(8), 2023.

[17] Chaoming Wang, Tianqiu Zhang, Xiaoyu Chen, Sichao He, Shangyang Li, and Si Wu. Brainpy, a flexible, integrative, efficient, and extensible framework for general-purpose brain dynamics programming. *elife*, 12:e86365, 2023.

[18] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020.

[19] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[20] Edgar D Adrian and Yngve Zotterman. The impulses produced by sensory nerve endings: Part 3. impulses set up by touch and pressure. *The Journal of physiology*, 61(4):465, 1926.

[21] Seongsik Park, Seijoon Kim, Byunggook Na, and Sungroh Yoon. T2fsnn: deep spiking neural networks with time-to-first-spike coding. In *2020 57th ACM/IEEE design automation conference (DAC)*, pages 1–6. IEEE, 2020.

[22] Jaehyun Kim, Heesu Kim, Subin Huh, Jinho Lee, and Kiyoung Choi. Deep neural networks with weighted spikes. *Neurocomputing*, 311:373–386, 2018.

[23] Eric Hunsberger and Chris Eliasmith. Spiking deep networks with lif neurons. *arXiv preprint arXiv:1510.08829*, 2015.

[24] Wei Fang, Zhaofei Yu, Yanqi Chen, Timothée Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2661–2671, 2021.

[25] Xingting Yao, Fanrong Li, Zitao Mo, and Jian Cheng. Glif: A unified gated leaky integrate-and-fire neuron for spiking neural networks. *Advances in Neural Information Processing Systems*, 35:32160–32171, 2022.

[26] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[27] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

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

[29] Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience*, 11:309, 2017.

[30] Dianze Li, Yonghong Tian, and Jianing Li. Sodformer: Streaming object detection with transformer using events and frames. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(11):14020–14037, 2023.

[31] Sicheng Shen, Dongcheng Zhao, Guobin Shen, and Yi Zeng. Tim: an efficient temporal interaction module for spiking transformer. *arXiv preprint arXiv:2401.11687*, 2024.

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

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

[34] Yuetong Fang, Ziqing Wang, Lingfeng Zhang, Jiahang Cao, Honglei Chen, and Renjing Xu. Spiking wavelet transformer. In *European Conference on Computer Vision*, pages 19–37. Springer, 2024.

[35] Han Zhang, Chenlin Zhou, Liutao Yu, Liwei Huang, Zhengyu Ma, Xiaopeng Fan, Huihui Zhou, and Yonghong Tian. Sglformer: spiking global-local-fusion transformer with high performance. *Frontiers in Neuroscience*, 18:1371290, 2024.

[36] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images.(2009), 2009.

[37] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017.

[38] Zhaokun Zhou, Yijie Lu, Yanhao Jia, Kaiwei Che, Jun Niu, Liwei Huang, Xinyu Shi, Yuesheng Zhu, Guoqi Li, Zhaofei Yu, et al. Spiking transformer with experts mixture. *Advances in Neural Information Processing Systems*, 37:10036–10059, 2024.

[39] 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. *arXiv preprint arXiv:2402.04663*, 2024.

[40] Chunming Jiang and Yilei Zhang. Klif: An optimized spiking neuron unit for tuning surrogate gradient slope and membrane potential. *arXiv preprint arXiv:2302.09238*, 2023.

[41] Matei-Ioan Stan and Oliver Rhodes. Learning long sequences in spiking neural networks. *Scientific Reports*, 14(1):21957, 2024.

[42] Shuaijie Shen, Chao Wang, Renzhuo Huang, Yan Zhong, Qinghai Guo, Zhichao Lu, Jianguo Zhang, and Luziwei Leng. Spikingssms: Learning long sequences with sparse and parallel spiking state space models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 20380–20388, 2025.

[43] Shiyu Chang, Yang Zhang, Wei Han, Mo Yu, Xiaoxiao Guo, Wei Tan, Xiaodong Cui, Michael Witbrock, Mark Hasegawa-Johnson, and Thomas S. Huang. Dilated recurrent neural networks, 2017.

[44] Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. A simple way to initialize recurrent networks of rectified linear units, 2015.

[45] David W Romero, Robert-Jan Bruintjes, Jakub M Tomczak, Erik J Bekkers, Mark Hoogendoorn, and Jan C van Gemert. Flexconv: Continuous kernel convolutions with differentiable kernel sizes. *arXiv preprint arXiv:2110.08059*, 2021.

[46] Sanghyeon Kim and Eunbyung Park. Smpconv: Self-moving point representations for continuous convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10289–10299, 2023.

[47] Zeyu Liu, Gourav Datta, Anni Li, and Peter Anthony Beerel. Lmuformer: low complexity yet powerful spiking model with legendre memory units. *arXiv preprint arXiv:2402.04882*, 2024.

[48] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[49] Charles Eckert, Xiaowei Wang, Jingcheng Wang, Arun Subramaniyan, Ravi Iyer, Dennis Sylvester, David Blaaauw, and Reetuparna Das. Neural cache: Bit-serial in-cache acceleration of deep neural networks. In *2018 ACM/IEEE 45Th annual international symposium on computer architecture (ISCA)*, pages 383–396. IEEE, 2018.

[50] Xiandong Zhao, Ying Wang, Cheng Liu, Cong Shi, Kaijie Tu, and Lei Zhang. Bitpruner: Network pruning for bit-serial accelerators. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[51] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC)*, pages 10–14. IEEE, 2014.

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

[53] Xuerui Qiu, Malu Zhang, Jieyuan Zhang, Wenjie Wei, Honglin Cao, Junsheng Guo, Rui-Jie Zhu, Yimeng Shan, Yang Yang, and Haizhou Li. Quantized spike-driven transformer. *arXiv preprint arXiv:2501.13492*, 2025.

[54] Yufei Guo, Xiaode Liu, Yuanpei Chen, Weihang Peng, Yuhan Zhang, and Zhe Ma. Spiking transformer: Introducing accurate addition-only spiking self-attention for transformer. *arXiv preprint arXiv:2503.00226*, 2025.

[55] 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. *arXiv preprint arXiv:2502.12677*, 2025.

[56] Zhaokun Zhou, Jun Niu, Yang Zhang, Li Yuan, and Yuesheng Zhu. Spiking transformer with spatial-temporal spiking self-attention. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2025.

[57] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 839–847. IEEE, 2018.

[58] Peixi Wu, Bosong Chai, Hebei Li, Menghua Zheng, Yansong Peng, Zeyu Wang, Xuan Nie, Yueyi Zhang, and Xiaoyan Sun. Spiking point transformer for point cloud classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 21563–21571, 2025.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Our abstract and introduction are centered around two key contributions: (1) We propose a unified framework for Spiking Transformers, designed to facilitate the development and evaluation of models across multiple backbones and tasks; (2) Built upon this framework, we conduct a systematic assessment of representative Spiking Transformer models, revealing several objective limitations. Our findings offer constructive insights and practical guidelines for future advancements in Spiking Transformer research.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Owing to limitations in time and computational resources, we refrain from exhaustively evaluating every existing Spiking Transformer and instead concentrate on a subset that best represents the current architectural landscape. Certain models are intrinsically ill-suited to specific downstream tasks—for instance, Spikformer lacks the structural components required for object detection—so we cannot reproduce their performance on those benchmarks. Moreover, as a benchmark framework, fairness dictates that every model be trained and tested under an identical experimental setup; thus, task-specific tricks reported in the original papers are intentionally omitted. These methodological choices, while essential for consistency, inevitably lead to modest discrepancies between our reproduced results and the figures originally published.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [NA]

Justification: This work primarily presents a comprehensive benchmark. It is grounded in extensive empirical experiments and observations, with minimal reliance on theoretical derivations.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: This work is primarily built upon the BrainCog platform, where we develop a unified Spiking Transformer framework that supports a wide range of tasks, including classification, detection, and segmentation. Detailed experimental results are provided in both the main text and the appendix. As the Dataset & Benchmark Track follows a single-blind review policy, all code and experiments have been made publicly available; the link can be found in the abstract. Key experimental hyperparameters are reported in the paper. For full details, the complete set of configurations is available in the codebase's configuration files.

Guidelines:

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

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: Given that the Dataset & Benchmark Track adopts a single-blind review process, all related code has been publicly released on GitHub. The accompanying repository includes detailed documentation and usage guidelines to help users quickly get started with our framework.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Key parameters such as the number of epochs, batch size, and training steps are explicitly reported in the paper. Additional training details can be found in the configuration files provided in the code repository.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

Justification: The paper includes extensive experimental results. However, as a benchmark-focused study, it does not require statistical significance testing in the traditional sense. Instead, we report absolute errors relative to the original results and provide detailed analysis and discussion of these discrepancies.

Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The paper includes a comprehensive list of the computational resources and specific parameters used to conduct our experiments.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: he research strictly follows the NeurIPS Code of Ethics. It involves only benchmark evaluations on publicly available datasets (e.g., CIFAR-10/100) without using any personally identifiable information or human-related data. No ethical concerns are identified in this work.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [No]

    Justification: This study aims to explore the performance and limitations of Spiking Transformers, with the goal of providing insights and recommendations for the future development of Spiking Transformers and related models. The work does not involve any societal or ethical impact.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [No]

    Justification: The work does not involve the release of any models or datasets that carry a high risk of misuse. It purely benchmarks existing open-source models on public datasets.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: We properly cite the original papers for all datasets (e.g., CIFAR-10/100) and models used in this study, and respect their corresponding licenses and terms of use.

Guidelines:

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

13. **New assets**

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

Answer: [NA]

Justification: The paper does not introduce or release any new datasets, models, or code assets.

Guidelines:

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

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

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

Answer: [NA]

Justification: The paper does not involve any crowdsourcing experiments or research with human participants.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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

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

    Answer: [NA]

    Justification: The paper does not involve any research with human participants and thus does not require IRB approval.

    Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [NA]

    Justification: The paper does not involve the use of LLMs as any important, original, or non-standard component in the development of the core methodology.

    Guidelines:

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

A Spiking Transformer Achitectures

A.1 Spiking Encoding

Fig.5 illustrates the four spike-based input encoding schemes used in our study: Direct, Phase[22], Rate [20], and Time-to-First-Spike (TTFS) [21]. For each static frame, we visualize its transformation over four discrete simulation steps (T=1. . . 4), showing how pixel intensities are mapped into temporally distributed spikes through different strategies. Direct encoding preserves raw intensity at every step, Phase encoding modulates spike timing periodically, Rate encoding converts intensity to firing frequency, and TTFS uses the latency of the first spike to encode information. These complementary methods introduce diverse temporal input dynamics for our Spiking Transformer benchmark, enabling fair model evaluation under varied temporal signal structures. The specific formulations for these encodings can be found in Eq. 5- 8.

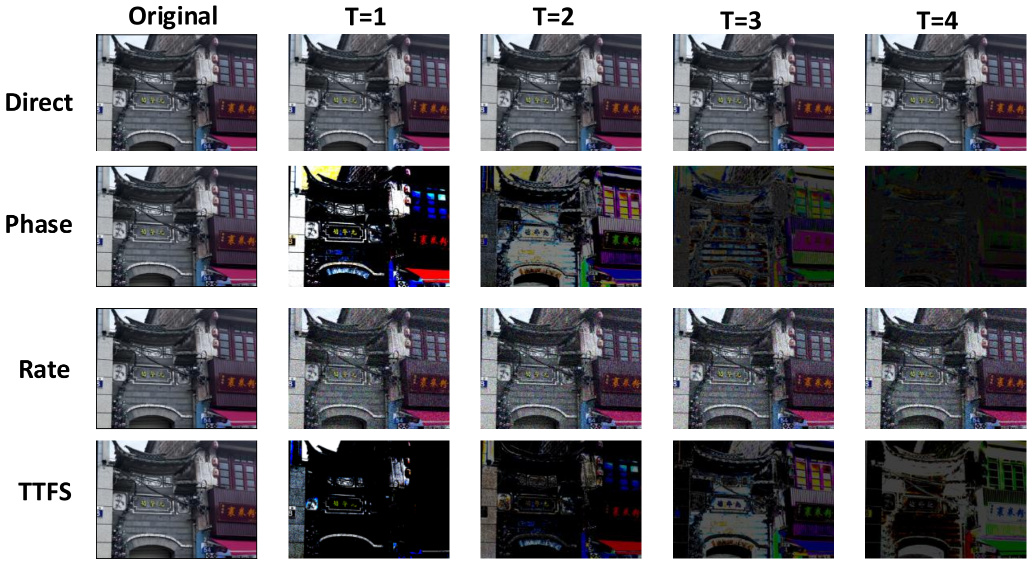

Figure 5: Visualization of different encoding methods.

Direct Encoding

$$S_t(\mathbf{p}) \;=\; x(\mathbf{p}), \qquad t = 1, \ldots, T, \tag{5}$$

where $x(\mathbf{p}) \in [0, 1]$ denotes the normalized pixel (or feature) intensity at spatial coordinate $\mathbf{p}$. The same constant input current is injected at every time step, i.e. the spike train is temporally uniform.

Phase Encoding

$$S_t(\mathbf{p}) \;=\; \begin{cases} 2^{-(b+1)}, & \text{if } v_{7-b}(\mathbf{p}) = 1,\; b \equiv (t-1) \pmod 8, \\ 0, & \text{otherwise}, \end{cases} \qquad v(\mathbf{p}) = \lfloor 256\, x(\mathbf{p}) \rfloor, \tag{6}$$

where v_k is the k-th bit of the 8-bit integer $v(\mathbf{p})$ (most significant bit $k = 7$). The encoder cycles through the eight bit-planes, assigning a weight that halves with each less-significant bit.

Rate Encoding

$$S_t(\mathbf{p}) \;\sim\; \text{Bernoulli}\big(x(\mathbf{p})\big), \qquad \mathbb{E}\big[S_t(\mathbf{p})\big] \;=\; x(\mathbf{p}), \qquad t = 1, \ldots, T. \tag{7}$$

A spike is emitted at time t with probability proportional to the input magnitude, so that the average firing rate reflects $x(\mathbf{p})$.

Time-to-First-Spike (TTFS) Encoding

$$t^{\star}(\mathbf{p}) = 1 + \left\lfloor \big(1 - x(\mathbf{p})\big)\, T \right\rfloor, \tag{8}$$

$$S_t(\mathbf{p}) = \begin{cases} \dfrac{1}{t^{\star}(\mathbf{p})}, & t = t^{\star}(\mathbf{p}), \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

Each neuron fires exactly once; higher input values trigger earlier spikes. We scale the spike amplitude by $1/t^{\star}$ to preserve energy across different latencies, but a binary value 1 can be used instead if desired.

A.2 Spiking Neuron

In this appendix, we describe the five discrete-time spiking neuron models integrated into our benchmark Spiking Transformer and summarize their defining characteristics. The vanilla LIF model implements the classical leak–fire–reset cycle, accumulating synaptic input and decaying with a fixed time constant τ before emitting a spike via a hard threshold [23]. PLIF extends this formulation by introducing a learnable membrane time constant for each neuron, allowing decay dynamics to adapt during training [24]. Building on PLIF, CLIF incorporates a complementary trace variable that smooths the surrogate-gradient around threshold crossings, thereby improving gradient flow in deep SNNs [39]. GLIF further enriches membrane dynamics with multiple gated internal states, capturing adaptation and refractory processes to more closely mimic biological neurons [25]. Finally, KLIF replaces the standard exponential leak with a learnable kernel constant, optimizing decay behavior for both biological realism and computational efficiency [40]. Section 4 reports the classification accuracy of each variant, enabling a comparative analysis of how these neuron-level enhancements influence Spiking Transformer performance and hardware requirements. The specific structure of neuron can be viewed in Fig. 6.

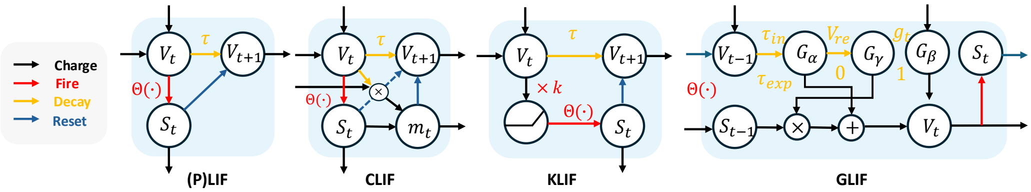

Figure 6: Visualization of different spiking neurons used in this work.

A.3 Model Basic Achitecture

With only a few specialised variants as exceptions, the architecture of the Spiking Transformer can be succinctly formalised by the following equations:

$$\begin{gathered} X = \mathrm{SPS}(\mathrm{Input}),\ \mathrm{PE} = \mathrm{SN}(\mathrm{BN}(\mathrm{Conv2d}(X))), \\ X_0 = x + \mathrm{PE}, \\ X_l' = \text{Spiking Attn}(X_{l-1}) + X_{l-1},\ X_l = \mathrm{MLP}(X_l') + X_l', \\ Y = \mathrm{Heads}(\mathrm{AP}(X_L)) \end{gathered} \tag{10}$$

Eq. 10 factorises the pipeline into four stages. **(1) SPS.** A four-stage Sequential Patch Splitting module—each stage stacks `Conv`–`BN`–`Pool`–Spiking-Neuron—upsamples the raw input into token feature maps X. **(2) Positional Encoding.** A shallow `Conv` followed by `BN` and a spiking activation produces PE, which is added to X to obtain the embedded sequence X_0. **(3) Transformer Block.** L residual blocks alternate *Spiking Self-Attention* and *MLP* layers, yielding hidden states $\{X_l\}_{l=1}^{L}$. **(4) Head.** Global average pooling $\mathrm{AP}(\cdot)$ followed by a task-specific head maps the final representation to the output Y.

Together, SPS and positional encoding realise the input-to-embedding conversion, while the stacked spiking blocks capture spatiotemporal dependencies with neuromorphic efficiency.

A.4 Spiking Attention

We select Spikformer, Spike-driven Transformer, and Spikformer+SEMM as three representative models. Detailed descriptions of the Attention mechanisms used in the latter two are provided below.

SDSA In Eq. 11, $Q, K, V \in \mathbb{R}^{B\times N\times C}$ denote the query, key, and value tensors for a batch of size B with N tokens and C channels. The operator $\otimes$ is an element-wise outer product between Q and K; $\mathrm{SUM}_c(\cdot)$ sums this product across the channel dimension C. $\mathrm{SN}(\cdot)$ is a spiking-neuron activation that returns a binary spike map. The final SDSA output is obtained by the element-wise product of this map with the value tensor V.

$$SDSA(Q,K,V) = SN(SUM_c(Q \otimes K)) \otimes V \tag{11}$$

Spikformer+SEMM In Eq. 12, m is the number of experts. For each expert $i \in \{1,\dots,m\}$, Q_m is its private query tensor and $A_m = \mathrm{SSA}_m(Q_m, K, V)$ is the corresponding sub-attention result. The input feature tensor is X, and $W_R^\top$ is the router's weight matrix. $\mathrm{BN}(\cdot)$ applies batch normalisation, while $\mathrm{SN}(\cdot)$ converts the routed signal into a set of spiking coefficients $\{r_1, r_2, \dots, r_m\}$. These coefficients weight the expert outputs to form the final mixture: $\mathrm{SSA{+}SEMM} = \sum_{i=1}^{m} r_i A_i$.

$$\begin{gathered} A_m = SSA_m(Q_m, K, V), \quad Router = SN\left(BN(W_R^T X)\right) = \{r_1, r_2, \dots, r_m\} \\ SSA + SEMM = \sum_{i=1}^{m} \mathbf{r}_i * \mathbf{A}_i, \end{gathered} \tag{12}$$

B Spiking Transformer Experiments

B.1 Selected Spiking Transformer Performance

We collect the performance of mainstream Spiking Transformers across a variety of static and dynamic datasets. We transcribe the original experimental results into Tab. 11 for direct comparison. For key reproduced models, we explicitly highlight their results in the tables, with detailed experimental setups and discussions available in Sec. 4 and Sec. 5.

Table 11: Selected Spiking Transformers *A2S*: ANN-SNN Conversion Model; *Transfer*: Transfer Learning Model; *T*: Time Step.

Dataset **Model**	**CIFAR10**	**CIFAR100**	**ImageNet-1K**	**CIFAR10-DVS**	**N-Cal101**
Spikformer [9]	95.41	78.21	74.81	78.9	-
Spikformer v2 [33]	-	-	80.38 *(8-512)*	-	-
QKFormer [10]	96.18	81.15	85.65 *(10-768)*	84.0*(T=16)*	-
Spikingformer [32]	95.81	79.21	75.85	79.9	-
SGLFormer [35]	96.76	82.26	83.73	82.9	-
Spiking Wavelet Transformer [34]	96.1	79.3	75.34 *(8-512)*	82.9	88.45
Spike-driven Transformer [12]	95.6	78.4 (2-512)	77.07	80.0 *(T=16)*	-
Meta-SpikeFormer(SDT v2) [13]	-	-	80.00	-	-
E-SpikeFormer(SDT v3) [14]	-	-	86.20 *(T=8)*	-	-
MST [52]	97.27 *(A2S)*	86.91 *(A2S)*	78.51 *(A2S)*	88.12 *(A2S)*	91.38 *(A2S)*
QSD [53]	98.4 *(Transfer)*	87.6 *(Transfer)*	80.3	89.8 *(Transfer)*	-
Spiking Transformer [54]	96.32	79.69	78.66 *(10-512)*	-	-
SNN-ViT [55]	96.1	80.1	80.23	82.3	-
STSSA [56]	-	-	-	83.8	81.65
Spikformer + SEMM [38]	95.78	79.04	75.93 *(8-512)*	82.32	-
SpikingResformer [11]	97.40 *(Transfer)*	85.98 *(Transfer)*	79.40	84.8 *(Transfer)*	-
TIM [31]	-	-	-	81.6	79.00

B.2 Visualization on ImageNet-1k

To further interpret the temporal dynamics of Spiking Transformers, we employ Grad-CAM++ [57] to visualize the attention maps across four simulation steps (T=1 to T=4) for both Spikformer [33] and QKFormer [10] on ImageNet-1k samples (Fig. 7). These visualizations offer insight into how each model accumulates temporal evidence and localizes discriminative features over time.

QKFormer demonstrates consistent and focused attention on the object regions across all time steps, especially for challenging examples such as the shark in a low-contrast underwater scene. This indicates a stable spatial grounding and effective temporal integration, likely attributable to its hierarchical pyramid architecture that supports multiscale representation.

In contrast, Spikformer exhibits broader and more diffuse activation patterns in early time steps, gradually converging toward the object. However, its attention maps remain noisier and less confined, particularly in scenes with complex backgrounds. This suggests that while Spikformer may respond quickly to salient regions, its spatial precision is relatively limited compared to QKFormer.

Overall, these results underscore the importance of temporal consistency and multiscale design in spiking vision transformers. QKFormer's clear and persistent localization highlights the benefit of incorporating hierarchical cues, aligning well with its superior top-1 performance.

B.3 Result on COCO

To assess the impact of pretraining on detection performance, we adopt SDTv2 [13] as the backbone for object detection, integrating it into the Mask R-CNN framework. SDTv2 replaces the standard CNN backbone with a custom Spiking Vision Transformer, featuring embedding dimensions of [128, 256, 512, 640], 8 heads, and 8 layers per stage. Its spike-driven self-attention (SDSA) enables efficient feature extraction with reduced computational cost.

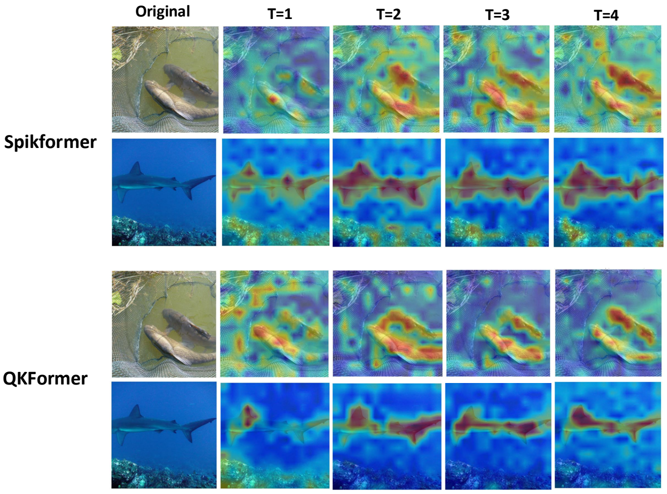

Figure 7: Visualization the importance weight using GradCam++ on QKFormer and Spikformer ImageNet-1k

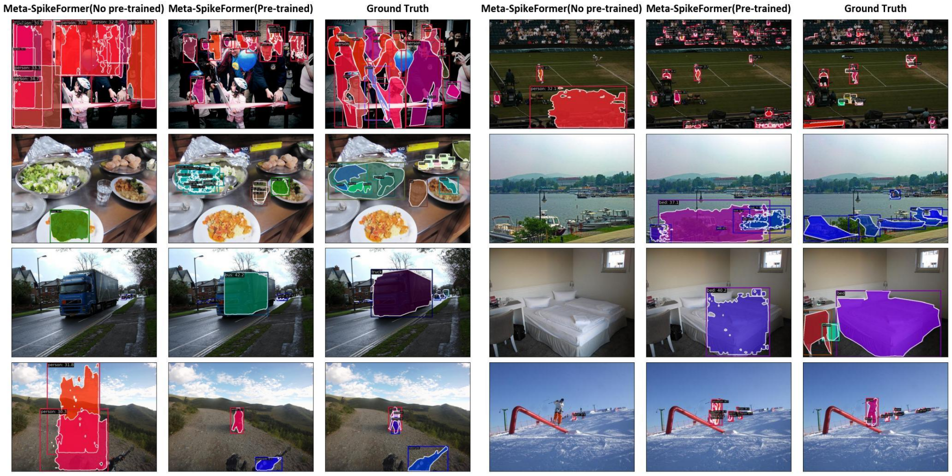
Meta-SpikeFormer(No pre-trained)
Meta-SpikeFormer(Pre-trained)
Ground Truth
Meta-SpikeFormer(No pre-trained)
Meta-SpikeFormer(Pre-trained)
Ground Truth

Figure 8: Detection predictions on COCO for SDTv2.

Multi-scale features extracted by SDTv2 are fused via a SpikeFPN neck. The RPN and ROI heads are adapted to the spiking domain using SpikeRPNHead and SpikeStandardRoIHead, preserving spike-driven computation throughout the pipeline.

Training follows a standard augmentation regime (random flipping, resizing) and a linear warm-up/decay schedule. We use AdamW with a learning rate of 2.5e-5, betas (0.9, 0.999), and weight decay of 0.05.

On COCO, SDTv2 achieves competitive performance with significantly lower power consumption. As shown in Fig. 8, visual comparisons further illustrate the benefit of pretraining, highlighting the suitability of SDTv2 for efficient, neuromorphic detection systems.

B.4 Framework Robustness

To demonstrate the robustness of the framework, we subsequently tested the main models on the primary datasets. The results in Tab. 12 show that the models reproduced based on our framework maintained both standard deviation and confidence intervals within a reasonable range across multiple datasets, including both static and neuromorphic ones.

Table 12: Classification accuracy and confidence intervals on CIFAR10, CIFAR100, and CIFAR10-DVS datasets.

Model	**Dataset**	**Acc@1 (%)**	**Std (%)**	**t-95% Confidence Interval**
Spikformer	CIFAR10	95.16	0.09	[95.04%, 95.27%]
SDT	CIFAR10	95.77	0.04	[95.72%, 95.82%]
QKFormer	CIFAR10	96.21	0.03	[96.18%, 96.25%]
Spikformer+SEMM	CIFAR10	95.55	0.33	[95.14%, 95.96%]
Spikingformer	CIFAR10	95.47	0.11	[95.32%, 95.61%]
Spikformer	CIFAR100	77.76	0.31	[77.38%, 78.14%]
SDT	CIFAR100	78.37	0.14	[78.19%, 78.54%]
QKFormer	CIFAR100	79.95	0.18	[79.72%, 80.17%]
Spikformer+SEMM	CIFAR100	78.41	0.52	[77.76%, 79.06%]
Spikingformer	CIFAR100	79.33	0.21	[79.06%, 79.59%]
Spikformer	CIFAR10-DVS	83.40	1.76	[81.12%, 85.49%]
SDT	CIFAR10-DVS	80.33	0.60	[79.58%, 81.08%]
QKFormer	CIFAR10-DVS	79.77	0.58	[79.05%, 80.49%]

B.5 Complicated Datasets

In addition to traditional tasks such as classification, segmentation, and detection, the capabilities of the Spiking Transformer have also been generalized to other datasets. These datasets include 3D point cloud classification tasks such as ModelNet10/40, as well as event-based video detection using DVS data. However, processing these datasets requires model-specific optimization, meaning that some basic baselines cannot be quickly adapted to support these tasks. Nevertheless, our framework integrates them to facilitate development and research for users. To demonstrate the successful integration of these tasks within our framework, we obtained preliminary results using the corresponding models on their respective tasks:

model	**mAP@[0.50:0.95]**	**mAP@0.5**	**AR@[0.50:0.95] (all)**	**AR@[0.50:0.95] (large)**
SDT [12]	0.000	0.001	0.008	0.028
SODFormer [30]	0.000	0.001	0.023	0.034

Table 13: SpikeDriven Transformer & SODFormer(baseline) on PKU_DAVIS_SOD with 3 epochs

Model	**Dataset**	**Step**	**Acc@1(%)**	**Epoch**
Spiking Point Transformer [58]	ModelNet10	4	90.5	200

Table 14: Performance of Spiking Point Transformer on the ModelNet10 dataset

According to Tab. 13, SDT exhibits a significant performance gap compared to the baseline on tasks without specific adaptation. In addition, it should be noted that the results of SODFormer and Spiking Point Transformer are presented merely to demonstrate the framework's compatibility with different datasets and tasks; their parameters are not fully aligned with those in the original papers and therefore do not reflect the actual performance of the models.

C STEP Quick Start

C.1 STEP Structure Overview

STEP is a modular benchmark framework designed for multi-task evaluation. It features a well-structured architecture while maintaining strong accessibility for users. The core structure of the STEP codebase is organized as follows:

STEP Repo Structure

```
STEP/
+- cls/     # Classification submodule
| +- README.md
| +- configs/
| +- datasets/
| +- ...
+- seg/     # Segmentation submodule
| +- README.md
| +- configs/
| +- mmseg/
| +- ...
+- det/     # Object detection submodule
| +- README.md
| +- configs/
| +- mmdet/
| +- ...
```

C.2 Classification Demo

In STEP, once components such as attention modules, neuron models, or encoding schemes are implemented, a complete model is assembled via a configuration file (.yml file per model setting), which then initiates the training pipeline.

For the classification task, the model can be configured using a configuration file as shown below. Here, we take the example of Spikformer evaluated on the CIFAR-10 dataset:

Spikformer CIFAR-10 Config

```
# dataset
data_dir:  '/data/datasets/CIFAR10'
dataset:  torch/cifar10
num_classes:  10
img_size:  32

# data augmentation
mean:
  - 0.4914
  - 0.4822
  - 0.4465
std:
  - 0.2470
  - 0.2435
  - 0.2616
crop_pct:  1.0
mixup:  0.5
cutmix:  0.0
reprob:  0.25
remode:  const
...
```

Spikformer CIFAR-10 Config (Continuous)

```
# model structure
model:  "spikformer_cifar"
step:  4
patch_size:  4
in_channels:  3
embed_dim:  384
num_heads:  12
mlp_ratio:  4
depths:  4

# meta transformer layer
embed_layer:  ’SPS’
attn_layer:  ’SSA’

# node
tau:  2.0
threshold:  1.0
act_function:  SigmoidGrad
node_type:  LIFNode
alpha:  4.0

# train hyperparam
amp:  True
batch_size:  128
val_batch_size:  128
lr:  5e-4
min_lr:  1e-5
sched:  cosine
...
# log dir
output:  ".output/cls/Spikformer"
# device
device:  0
```

After assembly, the training script can be launched directly from the terminal. In our configuration, multiple scripts can be defined for each model to facilitate controlled, multi-round comparative experiments.

Example Bash Command

```
conda activate [your_env]
python train.py config configs/spikformer/cifar10.yml
```

The above command launches the training process for a single model. For ImageNet, our models support multi-GPU training, which can be enabled by modifying the corresponding settings in the configuration file.

C.3 Segmentation & Detection Demo

hese two tasks are implemented based on the MMSegmentation and MMDetection frameworks. For models already constructed in the classification module, code can be directly migrated to the corresponding task directory with minimal modification. Given the computational demands of segmentation and detection, multi-GPU training is enabled by default. The configuration structure for these tasks is largely similar to that of classification and is therefore omitted here for brevity. The corresponding task can be launched using the following command:

Example Bash Command

```
conda activate [your_env]
cd tools
CUDA_VISIBLE_DEVICES=0,1 ./dist_train.sh ../configs/spikformer_8-512.py
2
```

C.4 Visualization

In addition, we provide model visualization support base on GradCam++. You may load pretrained weights at any time and insert hooks at the appropriate locations to visualize internal representations or dynamic behaviors of the model:

Example Bash Command

```
conda activate [your_env]
python -m cls.vis.gradcam_vis
```