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

Sicheng Shen[1,2,3,4,★] **Dongcheng Zhao**[1,3,★] **Linghao Feng**[1,3]
Zeyang Yue[5,♠] **Jindong Li**[1,3] **Tenglong Li**[1,3]
Guobin Shen[1,3] **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 ♠ Intern at CASIA when writing this paper
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.

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:

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

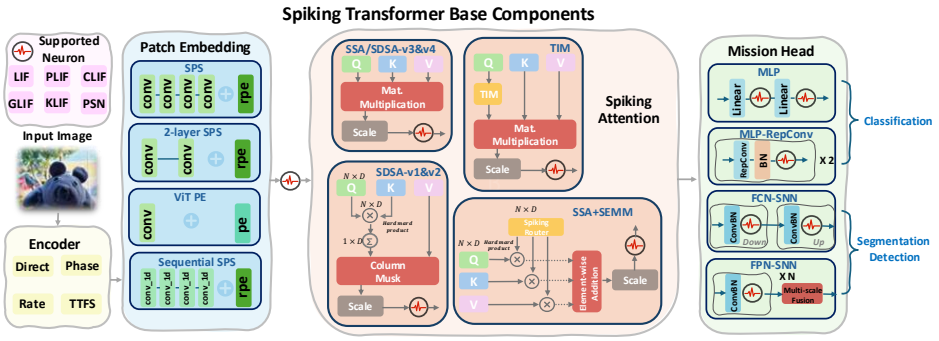

Figure 1: Unified Spiking Transformer Framework with Flexible Encoding, Attention Modules, and Application-specific Heads

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.

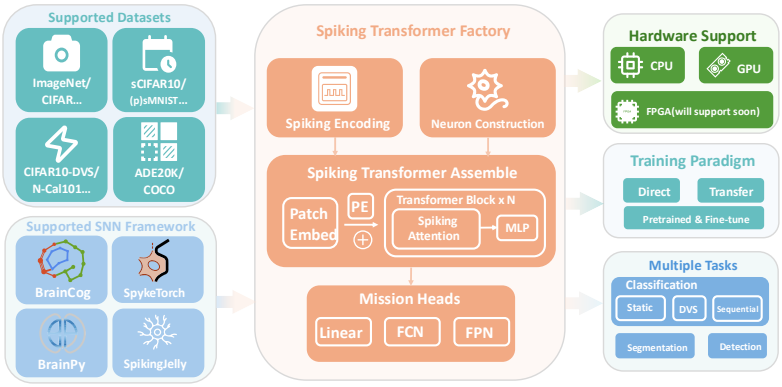

Figure 2: System Architecture of STEP as a Unified Benchmark for Spiking Transformer Development and Evaluation

3.2 Broad Dataset Compatibility

Our benchmark supports a wide spectrum of datasets, encompassing static (e.g., ImageNet [28], CIFAR10/100 [29]), event-based (e.g., DVS-CIFAR10 [30], N-Caltech101 [31]), 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.

3.3 Multi-task Adaptation

While early Spiking Transformers (e.g., Spikformer [9], TIM [32]) 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 maximize accessibility and reuse, the benchmark supports multiple backends including SpikingJelly [15], BrainCog [16], and BrainPy, among others. This backend-agnostic design ensures compatibility with different development environments and facilitates cross-framework reproducibility. Together, these components form a robust, extensible, and task-agnostic benchmarking framework. By supporting flexible model construction, diverse datasets, multi-task adaptation, and backend interoperability, our benchmark provides a solid foundation for future development and systematic evaluation of 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 [29]. 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 (95.41)	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 [33]	128	4	400	95.53 (95.81)	79.12 (79.21)
Spikformer + SEMM [34]	128	4	400	94.98 (95.78)	77.59 (79.04)
Spiking Wavelet [35]	128	4	400	95.31 (96.10)	76.99 (79.30)
SGLFormer [36]*	16	4	400	95.88 (96.76)	80.61 (82.26)
SpikingResformer [33]**	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.

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 [34] 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 [33]	192	4	300	73.69	69.80	23.51	31.43
Spikformer + SEMM [34]	-	-	-	-	63.41	13.13	19.76
SDT [12]	-	-	-	-	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 [34]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 initialization strategies, Spikformer-based models could potentially surpass existing baselines in dense prediction tasks. The segmentation result can be viewed in Fig. 3

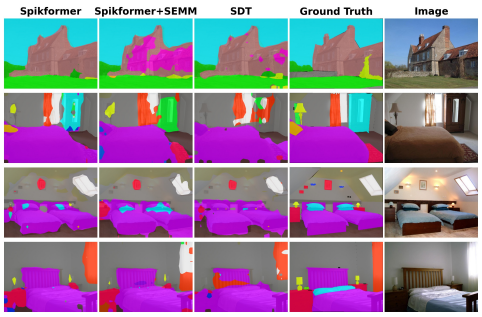

Figure 3: Segmentation predictions on ADE20K for three Spiking Transformer variants.

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 [34]	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

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 [34]	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 using four common encoding schemes: direct, phase, rate, and TTFS [20, 21, 22]. As shown in Tab. 6, direct encoding, a lossless approach ensures that no information is lost—which repeats the entire image at every time step matches current Spiking Transformers that compute attention independently across steps, and therefore yields the highest Top-1 accuracy on all models. In contrast, the sparser phase, rate, and TTFS encodings reduce spike density and weaken spatial coherence, leading to lower accuracy and highlighting the need for future architectures with temporally aware attention or recurrent mechanisms.

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:

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

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

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 [34]	**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 [34]	95.83	93.37	84.95

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:

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

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 advances in Spiking Transformers have primarily focused on improving task performance, our findings suggest that directly transplanting ANN modules—such as attention and convolution—may overlook the unique computational principles of SNNs. Future efforts should shift beyond performance-centric adaptation and draw inspiration from neuroscience. Promising directions include biologically grounded mechanisms such as dendritic computation, spike-timing-dependent plasticity (STDP), and temporal coding. These ideas could lead to the development of spike-native architectures that are not only energy-efficient and robust, but also more interpretable and aligned with neuromorphic hardware.

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.

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

[30] 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.

[31] Garrick Orchard, Ajinkya Jayawant, Gregory K Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in neuroscience*, 9:437, 2015.

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

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[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.

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.

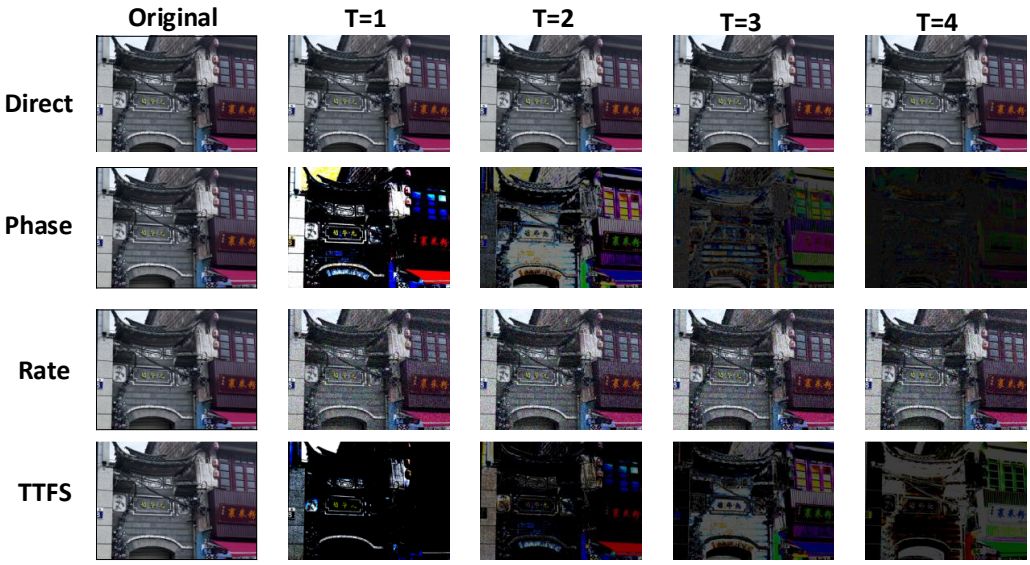

Figure 5: Visualization of different encoding methods.

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

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

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.

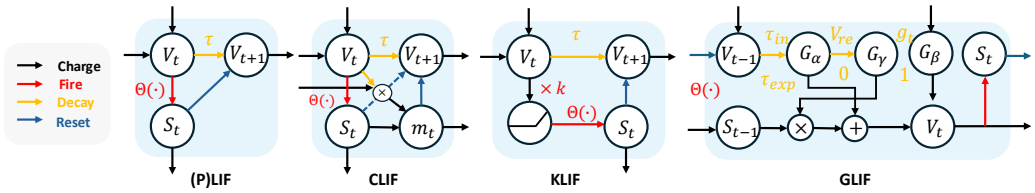

Figure 6: Visualization of different spiking neurons used in this work.

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: $\mathrm{SSA{+}SEMM} = \sum_{i=1}^{m} r_i A_i$.

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

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.

Model \ Datasset	CIFAR10	CIFAR100	ImageNet-1K	CIFAR10-DVS	N-Cal101
Spikformer [9]	95.41	78.21	74.81	78.9	-
Spikformer v2 [34]	-	-	80.38 *(8-512)*	-	-
QKFormer [10]	96.18	81.15	85.65 *(10-768)*	84.0*(T=16)*	-
Spikingformer [33]	95.81	79.21	75.85	79.9	-
SGLFormer [36]	96.76	82.26	83.73	82.9	-
Spiking Wavelet Transformer [35]	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 [32]	-	-	-	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 [34] 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.

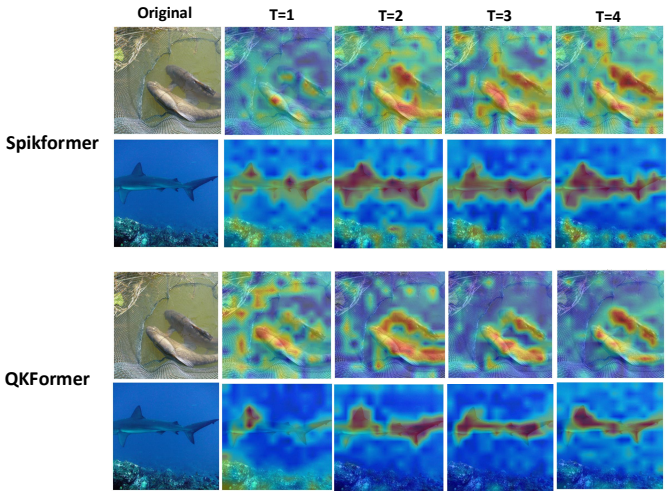

Figure 7: Visualization the importance weight using GradCam++ on QKFormer and Spikformer ImageNet-1k

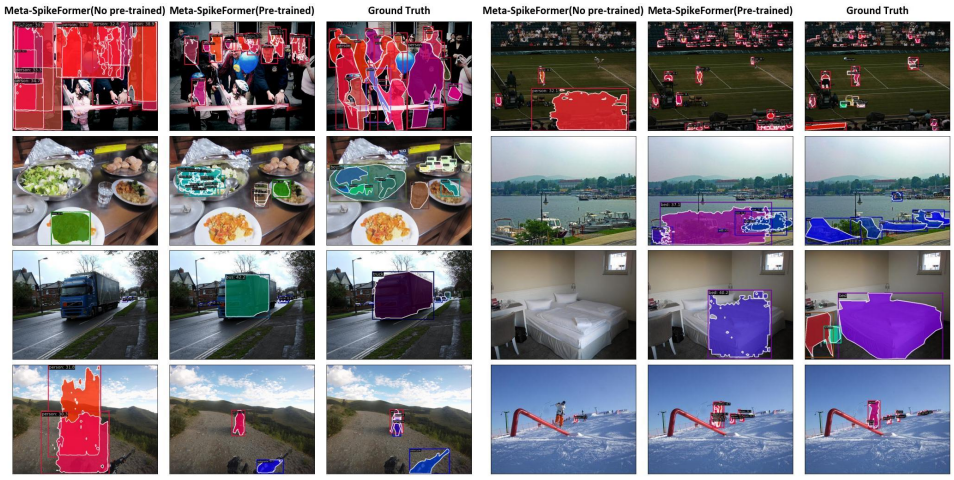

Figure 8: Detection predictions on COCO for SDTv2.

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.

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
```