Integer-Valued Training and Spike-Driven Inference Spiking Neural Network for High-performance and Energy-efficient Object Detection

Xinhao Luo[1*], Man Yao[1*], Yuhong Chou[2], Bo Xu[1], and Guoqi Li[1(✉)]

[1] Institute of Automation, Chinese Academy of Sciences
{luoxinhao2023, man.yao, guoqi.li}@ia.ac.cn
[2] Xi'an Jiaotong University

Abstract. Brain-inspired Spiking Neural Networks (SNNs) have bioplausibility and low-power advantages over Artificial Neural Networks (ANNs). Applications of SNNs are currently limited to simple classification tasks because of their poor performance. In this work, we focus on bridging the performance gap between ANNs and SNNs on object detection. Our design revolves around network architecture and spiking neuron. First, the overly complex module design causes spike degradation when the YOLO series is converted to the corresponding spiking version. We design a SpikeYOLO architecture to solve this problem by simplifying the vanilla YOLO and incorporating meta SNN blocks. Second, object detection is more sensitive to quantization errors in the conversion of membrane potentials into binary spikes by spiking neurons. To address this challenge, we design a new spiking neuron that activates Integer values during training while maintaining spike-driven by extending virtual timesteps during inference. The proposed method is validated on both static and neuromorphic object detection datasets. On the static COCO dataset, we obtain **66.2%** mAP@50 and **48.9%** mAP@50:95, which is **+15.0%** and **+18.7%** higher than the prior state-of-the-art SNN, respectively. On the neuromorphic Gen1 dataset, we achieve 67.2% mAP@50, which is **+2.5%** greater than the ANN with equivalent architecture, and the energy efficiency is improved by **5.7**×. Code: https://github.com/BICLab/SpikeYOLO

Keywords: Spiking neural network · Object detection · Spike-driven · Neuromorphic vision · Neuromorphic computing

1 Introduction

Brain-inspired SNNs are known for their low power consumption [46, 48]. Spiking neurons incorporate spatio-temporal information and emit spikes when the

*Equal contribution.
(✉) Corresponding author.

membrane potentials exceed a threshold [36]. Thus, spiking neurons trigger sparse additions only when they receive a spike and are otherwise idle. This spike-driven enables SNNs to exhibit obvious low-power advantages over ANNs when deployed on neuromorphic chips [7,37,41,61]. However, the negative impact of complex neuronal dynamics and spike-driven nature is that SNNs are difficult to train and have limited task performance and application scenarios [13].

For example, most applications of SNN algorithms in computer vision are limited to simple image classification tasks [9, 14, 20, 25, 27, 31, 33, 42, 54, 60]. Another more commonly used and challenging computer vision task, object detection, is rarely explored in SNNs. In 2020, Spiking-YOLO [29] provided the first object detection model in deep SNNs, exploiting the method of converting ANN to SNN with thousands of timesteps. In 2023, EMS-YOLO [50] became the first work to use direct training SNNs to handle object detection. Recently, the direct training Meta-SpikeFormer [58] can process the object detection in a pre-training and fine-tuning manner for the first time. However, the performance gap between these works and ANNs is significant. In this work, we aim to bridge this gap and demonstrate the low-power of SNNs and their unique advantages in neuromorphic applications. We achieve this goal through two efforts.

First, we design a new architecture, SpikeYOLO, which combines the macro design of YOLO with the micro design of the SNN module. Simply replacing the artificial neurons in the YOLO series [1,43,52] with spiking neurons generally does not work. Existing solutions include establishing the equivalence between ANN activation and spike firing rate [29], or improving the residual design [50]. We argue that another potential reason is that the module design in the YOLO series is too complex, which is effective in ANNs, but not suitable for SNNs. We observe that after the complex YOLO modules are converted into the corresponding spiking versions, there is a phenomenon of spike degradation in which the deep layers almost no longer emit spikes. Therefore, we tend to simplify the design in SpikeYOLO. We only retain the macro architecture in YOLOv8[3] while using the simple meta SNN block in Meta-SpikeFormer [58] as the basic module and performing micro design.

Second, we design a novel spiking neuron, Integer Leaky Integrate-and-Fire (I-LIF), to reduce the quantization error of SNNs. Spike-driven is the key to low-power, while exploiting only binary signals will drop performance, especially in challenging object detection. Numerous studies have attempted to mitigate this problem, such as attention-based membrane optimization [63], ternary spike [18], information max [19], converting quantized ANNs [26], optimal ANN2SNN [2]. However, direct training based on optimization strategies can only alleviate errors; the optimal approximation of ANN2SNN requires large timesteps, and it is difficult to exploit the temporal information. In contrast, the idea of the proposed I-LIF is to use integer-valued activations to drop quantization errors and convert them into spikes during inference by extending the virtual timesteps. The features of I-LIF are: 1) Integer-valued training improves performance and is easy to train; 2) Integer activation during training can be equivalent to spike-driven

[3] https://github.com/ultralytics/ultralytics

in inference that there is only sparse addition; 3) The temporal dynamics of LIF are reserved, capable of processing neuromorphic object detection.

The proposed method is validated on both static COCO [35] and neuromorphic Gen1 [8] object detection datasets. The performance we report on both datasets far exceeds the existing best models in SNNs. In addition, we explore the performance changes after the mutual conversion of ANN and SNN with the same architecture. The main contributions of this work are:

- **SpikeYOLO.** We explore suitable architectures in SNNs for handling object detection tasks and propose SpikeYOLO, which simplifies YOLOv8 and incorporates meta SNN blocks. This inspires us that the complex modules in ANN may not be suitable for SNN architecture design.
- **I-LIF Spiking Neuron.** We propose an I-LIF spiking neuron that combines integer-valued training with spike-driven inference. The former is used to reduce quantization errors in spiking neurons, and the latter is the basis of the low-power nature of SNNs.
- **Performance.** The proposed method achieves outstanding accuracy with low power consumption on object detection datasets, demonstrating the potential of SNNs in complex vision tasks. On the COCO dataset, we obtain **66.2%** mAP@50 and **48.9%** mAP@50:95, which is +**15.0%** and +**18.7%** higher than the prior state-of-the-art SNN, respectively. On the Gen1 dataset, SpikeYOLO is +**2.5%** better than ANN models with **5.7**× energy efficiency.

2 Related Works

SNN Training Method. Training methods have restricted the development of SNN for a long time. To make SNNs deeper, two training methods have been developed. ANN2SNN substitutes the ReLU function with spiking neurons, aiming to mimic the continuous activation by controlling the firing rate of spiking neurons [3, 10, 49]. It can achieve high performance but often requires long timesteps and is difficult to process sequence tasks that exploit the spatio-temporal dynamic nature of SNNs. In contrast, another directly training an SNN leverage gradient surrogate to circumvent the non-differentiability of binary spikes [39, 55]. Direct training is more flexible and requires fewer timesteps, but its performance usually suffers compared to ANNs of the same architecture. In this work, we focus on utilizing directly trained SNNs to process object detection due to its more flexibility in architectural design.

SNN Architecture Design. The architecture of SNNs can be roughly divided into two categories: CNN-based and Transformer-based SNNs. Spiking ResNet has long dominated the SNN field because residual learning [22] can address the performance degradation of SNNs as they become deeper. Typical spiking ResNet includes vanilla spiking ResNet [68], SEW-ResNet [14], and MS-ResNet [27]. The main difference between them is the location of shortcuts and the ability to achieve identity mapping [23]. Recently, the Transformer [12, 51] architecture has become popular [21, 38, 58, 60, 66, 67, 67, 69] in the SNN field and

has refreshed the performance upper bound. The SpikeYOLO we designed draws on the idea of the meta SNN block in Meta-SpikeFormer [58] and merges it with the YOLOv8 architecture.

Quantization Errors in SNNs. Spiking neurons quantize continuous membrane potentials with complex spatio-temporal dynamics into binary spikes. Obviously, quantization error will limit the performance of SNNs. In ANN2SNN, the reduction of quantization error is relatively simple, which can be achieved by increasing the timestep [2]. In direct training, adjusting the relationship between membrane potential distribution and threshold is the main solution, such as attention mechanism [63] optimizes membrane potential distribution to reduce noise spikes; information max [19] is derived from the information entropy angle to optimize spike firing. However, optimization cannot change the inherently quantized error nature of binary spikes. The solution of this work is to use integer values to reduce quantization errors during training and convert them to binary spikes to ensure spike-driven inference.

Object Detection. Object detection is a challenging yet pivotal task. Existing object detection frameworks can be simply categorized into: two-stage frameworks (RCNN series [16, 17, 45]) and single-stage frameworks (YOLO [1, 43, 52], Detr [4, 70] series). The former generate region proposals before determining precise object locations and classifications, whereas the latter directly ascertain these elements, offering a swifter, more streamlined solution. The object detection task has always been difficult for SNNs, and there are few related works. Current works include ANN2SNN-based YOLO [29, 65] and directly-trained spiking YOLO [50, 58]. Their performance is poor, and it isn't easy to meet the needs of real scenarios.

3 Methods

We exploit SpikeYOLO to process both static and neuromorphic object detection datasets. We first introduce how network inputs are unified. Then, the details of SpikeYOLO architecture and I-LIF spiking neuron are presented, respectively.

3.1 Network Input

The input of SNNs can be denoted as $X \in \mathbb{R}^{T\times C\times H\times W}$, where T is the timestep, C is the channel, $H \times W$ denote the spatial resolution.

Static Image. To leverage the spatio-temporal capabilities of SNNs, it is common practice that static images are repeated and utilized as input for each timestep T. This is called direct input encoding [30, 56], where the first layer of spiking neurons in the network encodes the continuous values of the input into spike signals.

Neuromorphic Event Stream. Neuromorphic data (also known as event-based vision) are generated by a Dynamic Vision Sensor (DVS), which only generates spikes when the logarithmic change in light intensity at a pixel surpasses a predefined threshold. An event-based stream is characterized as (x_n, y_n, t_n, p_n),

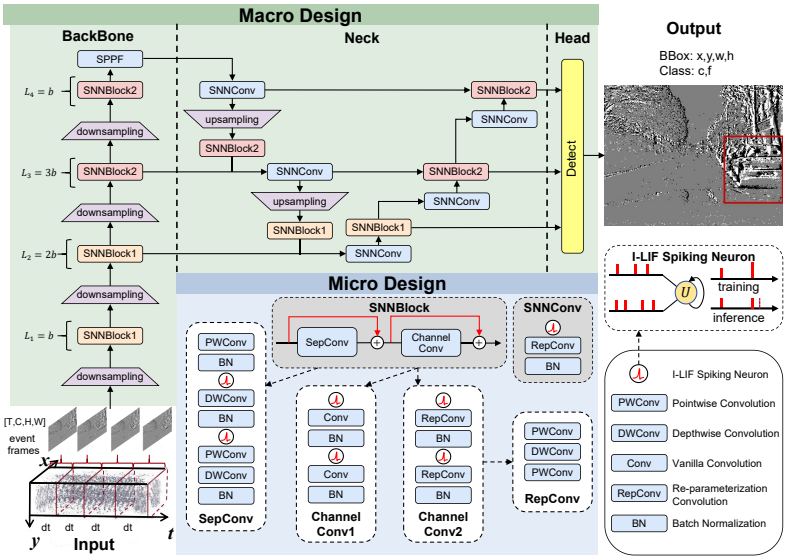

Fig. 1: The overall architecture of SpikeYOLO. We designed two SNN blocks, SNN-Block-1 and SNN-Block-2, and kept other architectures remain as YOLOv8. SNN-Block-1 employs standard convolution within its ChannelConv $(\cdot)$ component, whereas SNN-Block-2 utilizes re-parameterization convolution. That is, the difference between the two is the channel mixer module. In the low and high stages, we use SNN-Block-1 and SNN-Block-2, respectively. The spiking neuron is I-LIF, which activates integer values during training while converting them to binary spikes during inference.

with each event capturing spatial coordinates(x, y), timestamp t and polarity p, where $p \in \{-1, 1\}$ indicates whether light intensity has increased or decreased. Neuromorphic vision offers several advantages [15, 61], such as low resource requirements, high temporal resolution, and high robustness. The spike-driven nature of SNN makes it naturally suitable for processing event streams. The general strategy for neuromorphic pre-processing is to aggregate the event stream within a fixed time window into a frame format [57, 59, 62]. In this work, we follow this operation. Specifically, the total input window length is $T \times dt$, where dt and T are temporal resolution and timestep, respectively.

3.2 SpikeYOLO Architecture

Overview. SpikeYOLO integrates the macro design of the YOLOv8 with the micro design of Meta-SpikeFormer [58]. The motivation is that we observe complex computations within YOLO's modules result in spike degradation [27] upon direct conversion to the SNN version. Consequently, we maintain the overarching design principles of the YOLO architecture while incorporating the inverted residual structure [47] and re-parameterization convolution [11] design of the Meta-SpikeFormer for detailed aspects.

Network Output. In object detection, the network outputs the class and position of each object based on the input image sequence $X = \{X_t\}_{t=1}^{T}$. Suppose the input image sequence has N goals, the output $B = \{B_n\}_{n=1}^{N}$ can be calculated:

$$B = \text{Model}(X), \tag{1}$$

where each $B_n = \{f_n, c_n, x_n, y_n, w_n, h_n\}$ contains information about degree of confidence f_n, class C_n, center coordinates (x_n, y_n) and target size (w_n, h_n). $\text{Model}(\cdot)$ refers to the proposed SpikeYOLO architecture.

Macro Design. Fig. 1 shows the overview of SpikeYOLO, a variation of the YOLO framework that is more suitable for the feature extraction scheme of SNNs. Specifically, YOLOv8 is a classic single-stage detection framework that partitions the image into numerous grids, with each grid responsible for predicting a target independently. Some classic designs, such as the feature pyramid network [34] in YOLOv8, play a crucial role in facilitating efficient feature extraction and fusion. By contrast, its feature extraction module, such as C2F, performs repeated feature extraction from the same set of feature maps. This module can enhance feature extraction in ANNs but does not work well in SNNs. As a compromise, we preserve the classic Backbone/Neck/Head architecture in YOLOv8 while incorporating strategies from the meta SNN block in Meta-SpikeFormer.

Micro Design. Meta-SpikeFormer [58] is the current state-of-the-art architecture in SNNs, which explores the meta design of SNN and consists of CNN-based and Transformer-based SNN blocks. The meta block comprises a token mixer module and a channel mixer module. The difference between CNN-based and Transformer-based SNN blocks lies in the token mixer, which are spike-driven convolution and spike-driven self-attention, respectively. In this work, we mainly redesign the channel mixer module for the object detection task. As shown in Fig. 1, SNN-Block-1 and SNN-Block-2 are designed to extract low-stage and high-stage features, respectively.

The meta SNN block in [58] can be written as:

$$U' = U + \text{SepConv}(U), \tag{2}$$

$$U'' = U' + \text{ChannelConv}(U'), \tag{3}$$

where $U \in \mathbb{R}^{T\times C\times H\times W}$ is the layer input, $\text{SepConv}(\cdot)$ is an inverted separable convolution module [47] with 7×7 kernel size in MobileNetv2 to capture global features, followed by a 3×3 depthwise convolution for further spatial feature fusion. $\text{Sepconv}(\cdot)$ can be expressed as:

$$\text{SepConv}(U) = \text{Conv}_{\text{dw2}}(\text{Conv}_{\text{pw2}}(SN(\text{Conv}_{\text{dw1}}(SN(\text{Conv}_{\text{pw1}}(SN(U))))))), \tag{4}$$

where $\text{Conv}_{\text{dw1}}(\cdot)$ and $\text{Conv}_{\text{dw1}}(\cdot)$ are depthwise convolutions, $\text{Conv}_{\text{pw1}}(\cdot)$ and $\text{Conv}_{\text{pw1}}(\cdot)$ are pointwise convolutions [5]. $SN(\cdot)$ is the spiking neuron layer.

ChannelConv $(\cdot)$ is the channel mixer, which facilitates inter-channel information fusion. We redesign ChannelConv $(\cdot)$ in object detection task.

For SNN-Block-1, it is written as:

$$\text{ChannelConv1}\left(U'\right) = \text{Conv}\left(SN\left(\text{Conv}\left(SN\left(U'\right)\right)\right)\right), \tag{5}$$

where Conv $(\cdot)$ is a standard convolution with expansion ratio $r = 4$. In contrast, addressing high-stage features, SNN-Block-2 employs a re-parameterization convolution to minimize parameter count, which can be described as:

$$\text{ChannelConv2}\left(U'\right) = \text{RepConv}\left(SN\left(\text{RepConv}\left(SN\left(U'\right)\right)\right)\right), \tag{6}$$

$$\text{RepConv}\left(U'\right) = \text{Conv}_{\text{pw2}}\left(\text{Conv}_{\text{dw1}}\left(\text{Conv}_{\text{pw1}}\left(U'\right)\right)\right), \tag{7}$$

where RepConv $(\cdot)$ is the re-parameterization convolution [11] with kernel size 3×3, it can be re-parameterizated to a standard convolution during inference.

3.3 I-LIF Spiking Neuron

Spiking neurons propagate information in both spatial and temporal domains, and they mimic the spiking communication scheme of biological neurons. However, there are inherent quantization errors in converting the membrane potential of spiking neurons into binary spikes, which severely limits the representation of the model. Recently, Fast-SNN [26] achieves high-performance conversion with small timesteps by converting quantized ANNs into SNNs. This inspires us to "why not train directly with integer values", which can significantly reduce the quantization error. We just need to be careful to ensure that the inference is spike-driven. So, we came up with the idea of I-LIF.

LIF. Leaky Integrate-and-Fire (LIF) spiking neuron [36] is the most popular neuron to construct SNNs due to its balance between bio-plausibility and computing complexity. The dynamics of LIF with soft reset is:

$$U\left[t\right] = H\left[t-1\right] + X\left[t\right], \tag{8}$$

$$S\left[t\right] = \Theta\left(U\left[t\right] - V_{th}\right), \tag{9}$$

$$H\left[t\right] = \beta\left(U\left[t\right] - S\left[t\right]\right), \tag{10}$$

where t denotes the timestep, $U\left[t\right]$ is the membrane potential that integrates the temporal information $H\left[t-1\right]$ and spatial information $X\left[t\right]$. $\Theta\left(\cdot\right)$ is the Heaviside step function which equals 1 for $x \geq 0$ and 0 otherwise. If $U\left[t\right]$ exceeds the firing threshold V_{th}, spiking neuron fire a spike $S\left[t\right]$ and $U\left[t\right]$ will subtract it subsequently. Otherwise, $H\left[t\right]$ will remain unchanged. And, $U\left[t\right]$ decays to $H\left[t\right]$ by a factor of β, which denotes the decay constant. For simplicity, we focus on Eq.9 and denote the spiking neuron layer as $SN(\cdot)$, with its input as membrane potential tensor U and its output as spike tensor S.

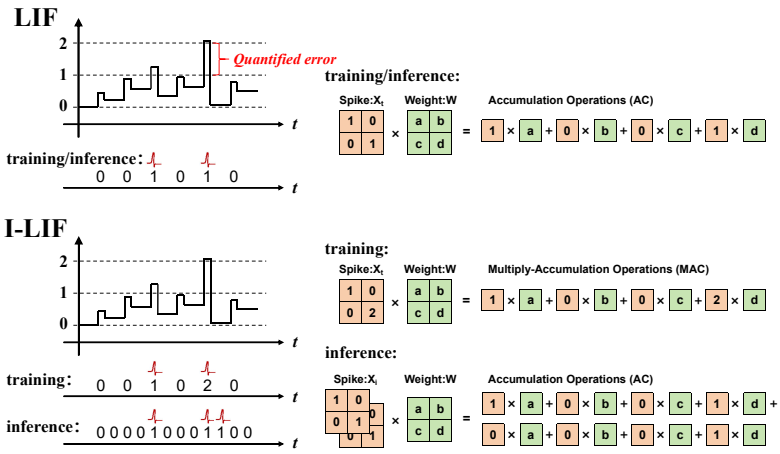

Fig. 2: Comparison of I-LIF and LIF. Binary spikes are emitted by LIF during both training and inference processes, which results in quantization errors. I-LIF emits integer values during the training process to reduce quantization errors, and converts them into binary spikes during inference to make the network only perform sparse addition.

I-LIF. We propose the Integer Leaky Integrate-and-Fire (I-LIF) neuron to reduce the quantization error. As shown in Fig. 2, I-LIF emits integer values while training, and converts them into $0/1$ spikes when inference. Specifically, in I-LIF, Eq.9 is rewritten as:

$$S\left[t\right] = Clip\left(round(U[t]), 0, D\right), \tag{11}$$

where $round(\cdot)$ is a round symbol, $Clip\left(x, min, max\right)$ denotes that clipping x to $[min, max]$, D is a hyperparameter indicating the maximum emitted integer value by I-LIF.

Training Stage. Eq.11 is not a continuous function, making its derivative a step function, potentially causing training instability. Previous studies have introduced several surrogate gradient functions, which primarily address binary spike outputs. We consistently utilize rectangular windows as the surrogate function. For simplicity, We retain gradients solely for neurons activated in the $[0, D]$ range, nullifying all others.

Inference Stage. Introducing integer value necessitates additional MACs (Multiply-Accumulation operations), potentially diminishing the energy efficiency of SNNs. Thus, converting integer values to binary spikes is essential. Fig. 3 shows an example of how integer values convert to binary spikes by extending virtual timesteps during inference. Specifically, the input to the spiking neuron at $l+1$ layer can be described as $X^{l+1}\left[t\right] = W^l S^l\left[t\right]$. We extend the T time step to $T \times D$, and convert the integer value $S^l\left[t\right]$ to a spike sequence $\{S^l\left[t,d\right]\}_{d=1}^{D}$, which satisfied:

$$\sum_{d=1}^{D} S^l\left[t,d\right] = S^l\left[t\right]. \tag{12}$$

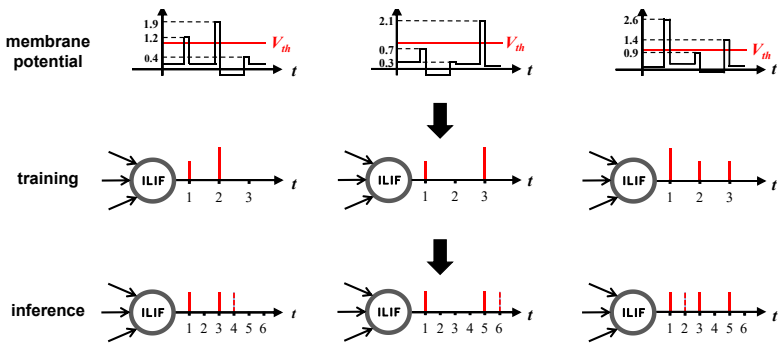

Fig. 3: An example of how the proposed I-LIF works. We assume $T = 3$,$D = 2$, and show the corresponding binary spike sequences of integer value during inference. The membrane potential in $[0.5, 1.5)$ are quantized to 1, while those in $[1.5, 2.5)$ are quantized to 2. membrane potential that > 2.5 are also quantized to 2 due to the maximum integer value $D = 2$. Subsequently, the membrane potential will be subtracted from the integer value. The training spike with a value of 2 will be converted into two binary spikes by extending virtual timesteps during inference.

Thus, the neuron's input at $l + 1$ layer is reformulated as:

$$X^l\left[t\right] = W^l \sum_{d=1}^{D} S^l\left[t, d\right]. \tag{13}$$

Given that matrix multiplication functions as linear operators, we establish:

$$W^l \sum_{d=1}^{D} S^l\left[t, d\right] = \sum_{d=1}^{D} (W^l S^l\left[t, d\right]). \tag{14}$$

Therefore, the input of the neuron at l+1 layer can be computed by:

$$X^l\left[t\right] = \sum_{d=1}^{D} (W^l S^l\left[t, d\right]). \tag{15}$$

The spike sequence $S^l\left[t, d\right]$ only contains $0/1$, so all MACs can be converted into sparse ACs(Accumulation operations), which can ensure spike-driven when inference.

4 Experiments

We evaluate the proposed method on COCO 2017 val [35] and neuromorphic Gen1 [8] datasets. The mean Average Precision(mAP) at IOU=0.5(mAP@50), the average mAP between 0.5 and 0.95(mAP@50:95), and energy cost are reported for each model. Specifically, the power of ANNs and SNNs can be calculated as:

$$E_{ANN} = O^2 \times C_{in} \times C_{out} \times k^2 \times E_{MAC}, \tag{16}$$

Table 1: Results on COCO 2017 val [35]. $T \times D$ means that we set up T timesteps, and each timestep is expanded D times. In prior SNNs and ANNs, D defaults to 1.

Architecture	Model	Param (M)	Power (mJ)	$T \times D$	mAP@ 50(%)	mAP@ 50:95(%)
ANN	PVT [53]	32.9	520.3	1	59.2	36.7
	DETR [4]	41.0	197.8	1	62.4	42.0
	YOLOv5 [4]	21.2	112.5	1	64.1	45.4
ANN2SNN	Spiking-Yolo [29]	10.2	-	3500	-	25.7
	Bayesian Optim [28]	10.2	-	5000	-	25.9
	Spike Calib [32]	17.1	-	512	45.4	-
Directly-trained SNN	EMS-YOLO [50]	26.9	29.0	4	50.1	30.1
	Meta-SpikeFormer	34.9	49.5	1	44.0	-
	(MaskRCNN) [58]	75.0	140.8	1	51.2	-
	Meta-SpikeFormer	16.8	34.8	1	45.0	-
	(YOLO) [58]	16.8	70.7	4	50.3	-
	SpikeYOLO(Ours)	13.2	23.1	1×4	59.2	42.5
		23.1	18.4	1×1	52.7	36.1
		23.1	34.6	1×4	62.3	45.5
		23.1	67.6	4×1	55.7	38.7
		23.1	134.7	4×4	63.3	46.3
		48.1	68.5	1×4	64.6	47.4
		68.8	84.2	1×4	**66.2**	**48.9**

$$E_{SNN} = (T \times D) \times fr \times O^2 \times C_{in} \times C_{out} \times k^2 \times E_{AC}, \tag{17}$$

where O is the feature output size, C_{in} and C_{out} denotes the number of input channel and output channel, k is the kernel size, fr denotes the average spike firing rate, T is the timestep, D is the upper limit of integer activation during training. We follow the most commonly used energy consumption evaluation method in the SNN field [40, 63, 64]. All operations assume a 32-bit floating-point implementation on 45nm technology, where $E_{MAC} = 4.6pJ$ and $E_{AC} = 0.9pJ$ [24]. As can be seen from Eq. 16 and 17, SNN's low power comes from its sparse addition operation. The fewer spikes, the sparser the computation.

4.1 COCO 2017 val Dataset

Experimental Setup. As a predominant static dataset for object detection, COCO 2017 val [35] comprises 80 classes split into 118K training and 5K validating images. In all experiments, we set decay factor $\beta = 0.25$, learning rate to 0.01, and adopt SGD optimizer. The models are trained for 300 epochs with a batch size of 40 on 4 NVIDIA V100 GPUs. Mosaic data augmentation [1] technique is employed. The network structure is given in the supplementary material. Note,

[4] https://github.com/ultralytics/yolov5

Table 2: Ablation studies of architectural design. We first convert YOLOv8 directly into the corresponding spiking version. Then we convert the SpikeYOLO designed in this work into the corresponding ANN version.

Architecture	Model	Param (M)	Power (mJ)	$T \times D$	mAP@ 50(%)	mAP@ 50:95(%)
ANN → SNN	YOLOv8	25.8	183.5	1	67.2	50.2
	Spiking YOLOv8	25.8	7.2	1×1	46.8	31.3
SNN → ANN	SpikeYOLO (**Ours**)	23.1	18.4	1×1	52.7	36.1
	YOLO	23.1	314.1	1	65.0	48.1

in our method, inference timestep is reported as $T \times D$, e.g., 1×4 denotes $T = 1, D = 4$.

Main Results are shown in Table 1. The proposed SpikeYOLO significantly improves the performance upper bound of the COCO dataset in SNNs. We obtain **66.2%** mAP@50 and **48.9%** mAP@50:95, which is **+15.0%** and **+18.7%** higher than the prior state-of-the-art SNN [58], respectively. SpikeYOLO also has significant advantages over existing SNNs in terms of parameters and power: **SpikeYOLO** vs. EMS-YOLO [50]: Param, **23.1M** vs. 26.9M; mAP@50, **62.3**% vs. 50.1%; mAP@50:95, **45.5**% vs. 30.1%; Power, **33.2mJ** vs. 29.0mJ. Moreover, the performance gap between SNNs and ANNs is significantly narrowed. For example, under similar parameters, the performance of SpikeYOLO and YOLO v5 are comparable, and the energy efficiency is 3.3×.

Ablation Studies of Architectural Design. We simplify YOLOv8 for SNN and incorporate meta SNN blocks. As shown in Table 1, this architectural improvement enables the accuracy of spikeYOLO at $T = 1$ and $D = 1$ to reach 52.7%, better than the prior state-of-the-art SNN. We are also interested in the question "*whether the architectures in SNNs and ANNs can be used directly interchangeably?*". We conduct the experiments in Table 2. We observe that directly converting the ANN architecture into the corresponding SNN brings significant performance degradation. The special architectural design of SNNs can improve its representation.

Ablation Studies of Quantization Error. Integer-valued training is designed to reduce quantization error in SNNs. The larger D is, the smaller the quantization error is. In Table 1, we fixed the parameters to 23.1M. When $T = 1$ and $T = 4$, we expand $D = 1$ to $D = 4$, respectively, and the accuracies of mAP@50 are increased by +9.6% and +7.6%. In contrast, if we fix $D = 1$ and increase $T = 1$ to $T = 4$, the performance improvement of mAP@50 is only 3%. These results show that quantization error has a greater impact on performance than the setting of timesteps. And, in terms of power, increasing D is more cost-effective than increasing T. For instance, when 1×1 changes to 1×4, power increases by 88%; while 1×1 changes to 4×1, power improves by 267%.

4.2 Gen1 Automotive Detection Dataset

Fig. 4: The object detection results on the COCO dataset. The first two columns compare the effect of maximum integer value D on performance for the same structure. The second and third columns compare the effect of the size of the model on performance.

Experimental Setup. As a large neuromorphic object detection dataset, Gen1 [8] encompasses 39 hours of open road and various driving scenarios, captured using an ATIS sensor with a resolution of 304×240 pixels. The dataset is organized into training, validation, and testing subsets. The bounding box annotations of pedestrians and cars(over 255,000) were manually labeled. For each annotation, we process the event-based stream 2.5 seconds before its occurrence, dividing it into T slices for model input. We train the model for 50 epochs and maintain other hyperparameters same as the COCO 2017 dataset.

Main Results on Gen1 dataset are shown in Table 3. The proposed SpikeYOLO notably elevates the performance benchmark for the Gen1 dataset in SNNs. We achieve **67.2%** mAP@50 with 23.1M parameters, which outperforms the prior state-of-the-art SNN model by +**8.2%**. For example, when $T = 5$, **SpikeYOLO** vs. EMS-YOLO [50]: Param, **13.2M** vs. 14.4M; mAP@50, **66.0**% vs. 59.0%; mAP@50:95, **38.5**% vs. 31.0%. In contrast to the COCO dataset, Gen1 contains temporal information that is more suitable for SNN processing. We conduct experiments on the performance of SNN and ANN with the same architecture. We observe that SpikeYOLO's mAP@50 accuracy is +**2.5%** higher than the corresponding ANN, and shows a **5.7**× energy efficiency. This indicates that SNN has attractive potential in processing neuromorphic data.

Ablation Studies of Quantization Error. Both T and D significantly influence outcomes when processing neuromorphic datasets. Table 4 gives a comprehensive ablation study on SpikeYOLO with 23.1M parameters that evaluate the effects of varying T and D. We observe some interesting experimental results. *First*, boosting the timestep T will bring improvement in accuracy and power. For instance, with the set of $D = 1$, extending $T = 1$ to $T = 4$ yields a +**6.7%** increase in mAP@50, and the power will increase by 3.7×. But further extending $T = 4$ to $T = 8$ results in a marginal increase of only +0.7% and significantly

Table 3: Results on the Gen1 dataset [8]. * We convert 23.1M of SpikeYOLO into ANN with the same architecture.

Architecture	Model	Param (M)	Power (mJ)	$T \times D$	mAP@ 50(%)	mAP@ 50:95(%)
ANN	YOLOv3-tiny [44]	10.2	5.1	1	44.5	-
	SpikeYOLO*	23.1	78.5	1	64.7	39.7
SNN	EMS-YOLO [50]	6.2	1.2	5	54.7	26.7
		9.3	2.0	5	56.5	28.6
		14.4	3.4	5	59.0	31.0
	VGG-11+SDD	12.6	11.1	1	-	17.4
	MobileNet-64+SSD	24.3	5.7	1	-	14.7
	DenseNet121-24+SSD [6]	8.2	3.9	1	-	18.9
	Spiking-Yolo [29]	7.9	102.3	500	44.2	-
	Tr-Spiking-Yolo [65]	7.9	0.9	5	45.3	-
	SpikeYOLO(Ours)	13.2	11.0	5×1	66.0	38.5
		23.1	19.7	5×1	66.4	38.9
		23.1	12.9	4×2	**67.2**	**40.4**

Table 4: The influence of T and D on Gen1. We set SpikeYOLO (23.1M) as the baseline and vary T and D for each study.

Method	$T \times D$	Power(mJ)	mAP@50(%)	mAP@50:95(%)
SpikeYOLO	1×1	4.0	59.3	33.1
	1×4	3.9 (-0.1)	65.1 (+5.8)	38.9 (+5.8)
	2×1	8.1	63.6	36.5
	2×2	7.8 (-0.3)	66.1 (+2.5)	39.0 (+2.5)
	2×4	7.1 (-1.0)	67.0 (+3.4)	40.1 (+3.6)
	4×1	14.8	66.0	38.4
	4×2	12.9 (-1.9)	67.2 (+1.2)	40.4 (+2.0)
	8×1	27.0	66.7	39.3

increases energy cost. *Second*, We were surprised to see that by expanding D at a fixed T, the performance will be improved, while the power will be dropped. For example, 2×1 vs. 2×2 vs. 2×4: mAP@50, 63.6% vs. 66.1% vs. 67.0%; Power, 8.1mJ vs. 7.8mJ vs. 7.1mJ. **This trend is completely different from SpikeYOLO's performance in COCO,** where the increase of D will bring more energy cost. We argue that this phenomenon is because SNN exhibits various spike firing for dense/sparse data.

4.3 Architecture Ablation Experiments

Re-parameterization Design. As shown in Table 5, if we remove re-parameterization by adding neurons into inverted separable convolutions, the mAP@50 and mAP@50:95 will decrease 1.7% and 1.8% respectively.

Table 5: Ablation studies of architecture design. We set $T \times D = 1 \times 4$ and modify just one point of baseline to test how the parameters, power and performance vary.

Method	Param(M)	mAP@50(%)	mAP@50:95(%)
SpikeYOLO(Baseline)	23.1	62.3	45.5
Remove re-parameterization	23.1	60.6	43.7
SNN-Block-1 → SNN-Block-2	19.9	61.2	44.7
SNN-Block-2 → Transformer Block	24.5	61.0	44.2
Anchor-free head → Anchor-based head	21.2	59.5	39.7

SNN Block Design. Including a 3×3 standard convolution within the initial stages of convolution blocks is crucial. As shown in Table 5, substituting SNN-Block-1 for SNN-Block-2 leads to a reduction in performance of around 1%. Moreover, we try to replace high-stage SNN-Block-2 with meta Transformer-based SNN block, just like Meta-SpikeFormer [58]. We find that there is little to no performance gain by doing this and that the parameters increase. Therefore, only spiking CNN blocks are exploited in our SpikeYOLO.

Detection Head. The detection mechanisms within YOLO are categorized into anchor-based heads(*e.g.*, YOLOv5) and anchor-free heads(*e.g.*, YOLOv8). The former directly predicts each bounding box's dimensions, whereas the latter estimates the probability distribution of each bounding box. Previous EMS-YOLO [50] and Meta-SpikeFormer [58] employ anchor-based heads. SpikeYOLO exploits the anchor-free head because of its higher accuracy (see Table 5).

5 Conclusion

This work significantly narrows the performance gap between SNNs and ANNs on object detection tasks. We achieve this through network architecture and spiking neuron design. The proposed SpikeYOLO architecture abandons the complex module design in the vanilla YOLO series and exploits simple meta spike blocks to build the model. Then, the I-LIF spiking neuron capable of integer-valued training and spike-driven inference is proposed to drop quantization errors. We improve the upper bound of the SNN domain's performance on the COCO dataset by +15.0% (mAP@50) and +18.7% (mAP@50:95), respectively. On the neuromorphic Gen1 dataset, SpikeYOLO achieves better performance and lower power than ANN of the same architecture. Furthermore, we investigate the performance of equivalent architecture ANNs and SNNs in different datasets, and the results show that the redesigned SNN architecture performed better. This work enables SNNs to handle complex object detection and can inspire the application of SNNs in more visual scenarios.

Acknowledgements

This work was partially supported by National Distinguished Young Scholars (62325603), and National Natural Science Foundation of China (62236009, U22A20103,62441606), Beijing Natural Science Foundation for Distinguished Young Scholars (JQ21015), China Postdoctoral Science Foundation (GZB20240824, 2024M753497), and CAAI-MindSpore Open Fund, developed on OpenI Community.

References

1. Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934 (2020)
2. Bu, T., Fang, W., Ding, J., DAI, P., Yu, Z., Huang, T.: Optimal ANN-SNN conversion for high-accuracy and ultra-low-latency spiking neural networks. In: International Conference on Learning Representations (2022), https://openreview.net/forum?id=7B3IJMM1k_M
3. Cao, Y., Chen, Y., Khosla, D.: Spiking deep convolutional neural networks for energy-efficient object recognition. International Journal of Computer Vision **113**, 54–66 (2015)
4. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European Conference on Computer Vision. pp. 213–229. Springer (2020)
5. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1800–1807 (2017)
6. Cordone, L., Miramond, B., Thierion, P.: Object detection with spiking neural networks on automotive event data. In: 2022 International Joint Conference on Neural Networks. pp. 1–8. IEEE (2022)
7. Davies, M., Srinivasa, N., Lin, T.H., Chinya, G., Cao, Y., Choday, S.H., Dimou, G., Joshi, P., Imam, N., Jain, S., et al.: Loihi: A neuromorphic manycore processor with on-chip learning. IEEE Micro **38**(1), 82–99 (2018)
8. De Tournemire, P., Nitti, D., Perot, E., Migliore, D., Sironi, A.: A large scale event-based detection dataset for automotive. arXiv preprint arXiv:2001.08499 (2020)
9. Deng, S., Li, Y., Zhang, S., Gu, S.: Temporal efficient training of spiking neural network via gradient re-weighting. In: International Conference on Learning Representations (2022), https://openreview.net/forum?id=_XNtisL32jv
10. Diehl, P.U., Neil, D., Binas, J., Cook, M., Liu, S.C., Pfeiffer, M.: Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In: 2015 International Joint Conference on Neural Networks. pp. 1–8. IEEE (2015)
11. Ding, X., Zhang, X., Ma, N., Han, J., Ding, G., Sun, J.: Repvgg: Making vgg-style convnets great again. In: Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition. pp. 13733–13742 (2021)
12. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2021), https://openreview.net/forum?id=YicbFdNTTy

13. Eshraghian, J.K., Ward, M., Neftci, E.O., Wang, X., Lenz, G., Dwivedi, G., Bennamoun, M., Jeong, D.S., Lu, W.D.: Training spiking neural networks using lessons from deep learning. Proceedings of the IEEE **111**(9), 1016–1054 (2023)
14. Fang, W., Yu, Z., Chen, Y., Huang, T., Masquelier, T., Tian, Y.: Deep residual learning in spiking neural networks. Advances in Neural Information Processing Systems **34**, 21056–21069 (2021)
15. Gallego, G., Delbrück, T., Orchard, G., Bartolozzi, C., Taba, B., Censi, A., Leutenegger, S., Davison, A.J., Conradt, J., Daniilidis, K., et al.: Event-based vision: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**(1), 154–180 (2020)
16. Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1440–1448 (2015)
17. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition. pp. 580–587 (2014)
18. Guo, Y., Chen, Y., Liu, X., Peng, W., Zhang, Y., Huang, X., Ma, Z.: Ternary spike: Learning ternary spikes for spiking neural networks. arXiv preprint arXiv:2312.06372 (2023)
19. Guo, Y., Chen, Y., Zhang, L., Liu, X., Wang, Y., Huang, X., Ma, Z.: Im-loss: information maximization loss for spiking neural networks. Advances in Neural Information Processing Systems **35**, 156–166 (2022)
20. Guo, Y., Liu, X., Chen, Y., Zhang, L., Peng, W., Zhang, Y., Huang, X., Ma, Z.: Rmp-loss: Regularizing membrane potential distribution for spiking neural networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 17391–17401 (2023)
21. Han, M., Wang, Q., Zhang, T., Wang, Y., Zhang, D., Xu, B.: Complex dynamic neurons improved spiking transformer network for efficient automatic speech recognition. Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI 2023) (2023)
22. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)
23. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) Computer Vision – ECCV 2016. pp. 630–645. Springer International Publishing, Cham (2016)
24. Horowitz, M.: 1.1 computing's energy problem (and what we can do about it). In: 2014 IEEE International Solid-State Sircuits Conference digest of technical papers. pp. 10–14. IEEE (2014)
25. Hu, J., Yao, M., Qiu, X., Chou, Y., Cai, Y., Qiao, N., Tian, Y., XU, B., Li, G.: High-performance temporal reversible spiking neural networks with $\mathcal{O}(l)$ training memory and $\mathcal{O}(1)$ inference cost. In: Forty-first International Conference on Machine Learning (2024)
26. Hu, Y., Zheng, Q., Jiang, X., Pan, G.: Fast-snn: Fast spiking neural network by converting quantized ann. IEEE Transactions on Pattern Analysis and Machine Intelligence **45**(12), 14546–14562 (2023)
27. Hu, Y., Deng, L., Wu, Y., Yao, M., Li, G.: Advancing spiking neural networks toward deep residual learning. IEEE Transactions on Neural Networks and Learning Systems pp. 1–15 (2024)
28. Kim, S., Park, S., Na, B., Kim, J., Yoon, S.: Towards fast and accurate object detection in bio-inspired spiking neural networks through bayesian optimization. IEEE Access **9**, 2633–2643 (2020)

29. Kim, S., Park, S., Na, B., Yoon, S.: Spiking-yolo: spiking neural network for energy-efficient object detection. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 11270–11277 (2020)
30. Kim, Y., Park, H., Moitra, A., Bhattacharjee, A., Venkatesha, Y., Panda, P.: Rate coding or direct coding: Which one is better for accurate, robust, and energy-efficient spiking neural networks? In: ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 71–75. IEEE (2022)
31. Li, C., Jones, E.G., Furber, S.: Unleashing the potential of spiking neural networks with dynamic confidence. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 13304–13314 (2023)
32. Li, Y., He, X., Dong, Y., Kong, Q., Zeng, Y.: Spike calibration: Fast and accurate conversion of spiking neural network for object detection and segmentation. arXiv preprint arXiv:2207.02702 (2022)
33. Li, Y., Geller, T., Kim, Y., Panda, P.: Seenn: Towards temporal spiking early exit neural networks. Advances in Neural Information Processing Systems **36** (2024)
34. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 2117–2125 (2017)
35. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13. pp. 740–755. Springer (2014)
36. Maass, W.: Networks of spiking neurons: the third generation of neural network models. Neural Networks **10**(9), 1659–1671 (1997)
37. Merolla, P.A., Arthur, J.V., Alvarez-Icaza, R., Cassidy, A.S., Sawada, J., Akopyan, F., Jackson, B.L., Imam, N., Guo, C., Nakamura, Y., et al.: A million spiking-neuron integrated circuit with a scalable communication network and interface. Science **345**(6197), 668–673 (2014)
38. Mueller, E., Studenyak, V., Auge, D., Knoll, A.: Spiking transformer networks: A rate coded approach for processing sequential data. In: 2021 7th International Conference on Systems and Informatics (ICSAI). pp. 1–5. IEEE (2021)
39. Neftci, E.O., Mostafa, H., Zenke, F.: Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. IEEE Signal Processing Magazine **36**(6), 51–63 (2019)
40. Panda, P., Aketi, S.A., Roy, K.: Toward scalable, efficient, and accurate deep spiking neural networks with backward residual connections, stochastic softmax, and hybridization. Frontiers in Neuroscience **14**, 653 (2020)
41. Pei, J., Deng, L., Song, S., Zhao, M., Zhang, Y., Wu, S., Wang, G., Zou, Z., Wu, Z., He, W., et al.: Towards artificial general intelligence with hybrid tianjic chip architecture. Nature **572**(7767), 106–111 (2019)
42. Qiu, X., Zhu, R.J., Chou, Y., Wang, Z., Deng, L.j., Li, G.: Gated attention coding for training high-performance and efficient spiking neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38, pp. 601–610 (2024)
43. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition. pp. 779–788 (2016)
44. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767 (2018)
45. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in Neural Information Processing Systems **28** (2015)

46. Roy, K., Jaiswal, A., Panda, P.: Towards spike-based machine intelligence with neuromorphic computing. Nature **575**(7784), 607–617 (2019)
47. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition. pp. 4510–4520 (2018)
48. Schuman, C.D., Kulkarni, S.R., Parsa, M., Mitchell, J.P., Kay, B., et al.: Opportunities for neuromorphic computing algorithms and applications. Nature Computational Science **2**(1), 10–19 (2022)
49. Sengupta, A., Ye, Y., Wang, R., Liu, C., Roy, K.: Going deeper in spiking neural networks: Vgg and residual architectures. Frontiers in Neuroscience **13**, 95 (2019)
50. Su, Q., Chou, Y., Hu, Y., Li, J., Mei, S., Zhang, Z., Li, G.: Deep directly-trained spiking neural networks for object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6555–6565 (2023)
51. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems. pp. 5998–6008 (2017)
52. Wang, C.Y., Bochkovskiy, A., Liao, H.Y.M.: Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7464–7475 (2023)
53. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 568–578 (2021)
54. Wang, Z., Fang, Y., Cao, J., Zhang, Q., Wang, Z., Xu, R.: Masked spiking transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1761–1771 (2023)
55. Wu, Y., Deng, L., Li, G., Zhu, J., Shi, L.: Spatio-temporal backpropagation for training high-performance spiking neural networks. Frontiers in Neuroscience **12**, 331 (2018)
56. Wu, Y., Deng, L., Li, G., Zhu, J., Xie, Y., Shi, L.: Direct training for spiking neural networks: Faster, larger, better. In: Proceedings of the AAAI conference on artificial intelligence. vol. 33, pp. 1311–1318 (2019)
57. Yao, M., Gao, H., Zhao, G., Wang, D., Lin, Y., Yang, Z., Li, G.: Temporal-wise attention spiking neural networks for event streams classification. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10221–10230 (2021)
58. Yao, M., Hu, J., Hu, T., Xu, Y., Zhou, Z., Tian, Y., XU, B., Li, G.: Spike-driven transformer v2: Meta spiking neural network architecture inspiring the design of next-generation neuromorphic chips. In: The Twelfth International Conference on Learning Representations (2024), https://openreview.net/forum?id=1SIBN5Xyw7
59. Yao, M., Hu, J., Zhao, G., Wang, Y., Zhang, Z., Xu, B., Li, G.: Inherent redundancy in spiking neural networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 16924–16934 (2023)
60. Yao, M., Hu, J., Zhou, Z., Yuan, L., Tian, Y., Xu, B., Li, G.: Spike-driven transformer. Advances in neural information processing systems **36** (2024)
61. Yao, M., Richter, O., Zhao, G., Qiao, N., Xing, Y., Wang, D., Hu, T., Fang, W., Demirci, T., De Marchi, M., Deng, L., Yan, T., Nielsen, C., Sheik, S., Wu, C., Tian, Y., Xu, B., Li, G.: Spike-based dynamic computing with asynchronous sensing-computing neuromorphic chip. Nature Communications **15**(1), 4464 (May 2024)

62. Yao, M., Zhang, H., Zhao, G., Zhang, X., Wang, D., Cao, G., Li, G.: Sparser spiking activity can be better: Feature refine-and-mask spiking neural network for event-based visual recognition. Neural Networks **166**, 410–423 (2023)
63. Yao, M., Zhao, G., Zhang, H., Hu, Y., Deng, L., Tian, Y., Xu, B., Li, G.: Attention spiking neural networks. IEEE Transactions on Pattern Analysis and Machine Intelligence **45**(8), 9393–9410 (2023)
64. Yin, B., Corradi, F., Bohté, S.M.: Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks. Nature Machine Intelligence **3**(10), 905–913 (2021)
65. Yuan, M., Zhang, C., Wang, Z., Liu, H., Pan, G., Tang, H.: Trainable spiking-yolo for low-latency and high-performance object detection. Neural Networks **172**, 106092 (2024)
66. Zhang, J., Dong, B., Zhang, H., Ding, J., Heide, F., Yin, B., Yang, X.: Spiking transformers for event-based single object tracking. In: Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition. pp. 8801–8810 (2022)
67. Zhang, J., Tang, L., Yu, Z., Lu, J., Huang, T.: Spike transformer: Monocular depth estimation for spiking camera. In: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VII. pp. 34–52. Springer (2022)
68. Zheng, H., Wu, Y., Deng, L., Hu, Y., Li, G.: Going deeper with directly-trained larger spiking neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 11062–11070 (2021)
69. Zhou, Z., Zhu, Y., He, C., Wang, Y., YAN, S., Tian, Y., Yuan, L.: Spikformer: When spiking neural network meets transformer. In: The Eleventh International Conference on Learning Representations (2023), https://openreview.net/forum?id=frE4fUwz_h
70. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable {detr}: Deformable transformers for end-to-end object detection. In: International Conference on Learning Representations (2021), https://openreview.net/forum?id=gZ9hCDWe6ke