

## RESEARCH ARTICLE

# Spike-Event Object Detection for Neuromorphic Vision

YUAN-KAI WANG<sup>1,2</sup>, (Senior Member, IEEE), SHAO-EN WANG<sup>1</sup>, AND PING-HSIEN WU<sup>1</sup><sup>1</sup>Department of Electrical Engineering, Fu Jen Catholic University, New Taipei City 24205, Taiwan<sup>2</sup>Graduate Institute of Applied Science and Engineering, Fu Jen Catholic University, New Taipei City 24205, Taiwan

Corresponding author: Yuan-Kai Wang (ykwang@fju.edu.tw)

This work was supported by the Ministry of Science and Technology, Taiwan, under Contract MOST 109-2221-E-030-018-MY3.

**ABSTRACT** Neuromorphic vision is one of the novel research fields that study neuromorphic cameras and spiking neural networks (SNNs) for computer vision. Instead of computing on frame-based images, spike events are streamed from neuromorphic cameras, and novel object detection algorithms have to deal with spike events to achieve detection tasks. In this paper, we propose a solution to the novel object detection method with spike events. Spike events are first encoded to event images according to the computational methodology of neuromorphic theory. The event images can be realized as change-detected images of moving objects with a high frame rate. A redesigned deep learning framework is proposed for object detection to process the event images. We propose a deep SNN method that is achieved by the conversion of successful convolution neural networks but trained by event images. The networks with multiscale representation are discussed and designed in our method. We also design a semi-automatic data labeling method to build event-image datasets by object tracking algorithms. The proposed solution, therefore, includes spike event encoding, a redesigned deep SNN, and an event-image data augmentation algorithm. Experiments are conducted not only on the MNIST-DVS dataset, which is a benchmark dataset for the study of neuromorphic vision but also on our event pedestrian detection dataset. The experimental results show that the performance of the deep SNN trained with our augmented data is close to the model trained on manually labeled data. A performance comparison based on the PAFBenchmark dataset shows that our proposed method has higher accuracy than existing SNN methods, and better energy efficiency and lower energy consumption than existing CNN methods. It demonstrates that our deep SNN method is a feasible solution for the study of neuromorphic vision. The intuition that deep SNN trained with more learning data can achieve better accuracy is also confirmed for this brand-new research field.


**INDEX TERMS** Deep spiking neural network, semi-automatic data labeling, event camera, multiscale representation, YOLO.

## I. INTRODUCTION

Over the past decade, Convolutional Neural Networks (CNNs) have become a mainstream machine learning approach for computer vision with a lot of successful applications such as image classification, object detection, and semantic scene labeling. Going beyond CNN, Spiking Neural Networks (SNNs) [1], [2] are termed the third generation of neural networks because of their potential to supersede deep

learning methods in the fields of computational neuroscience and biologically plausible machine learning. Neuromorphic computing [3], [4], [5] has emerged for several decades as a complementary architecture to von Neumann systems. More recently, the term has come to encompass implementations that are based on biologically-inspired or artificial neural networks in or using non-von Neumann architectures [6], which comes to the SNNs.

SNNs are networks that employ spiking neurons as computational units, taking into account the precise firing times of neurons for information coding. SNNs are one of the most

The associate editor coordinating the review of this manuscript and approving it for publication was Kumaradevan Punithakumar .

popular neural models able to directly handle events. It is thought to be more practical for temporal data processing tasks since the spiking neurons naturally integrate their inputs over time. Moreover, their binary (spiking or no spiking) operation lends itself well to fast and energy efficient simulation on hardware devices. Despite their advantages in terms of speed and power consumption, training deep SNNs models on complex tasks is usually very difficult.

Neuromorphic cameras, such as the Dynamic Vision Sensor (DVS) [7] and Asynchronous Time-based Image Sensor (ATIS) [8], are bio-inspired vision sensors that, in contrast to traditional cameras, do not acquire full images at a fixed frame rate but rather have independent pixels that output only intensity changes (called “events”) asynchronously at the time they occur. They offer significant advantages over standard cameras: high temporal resolution (in the order of  $\mu\text{s}$ ), very high dynamic range (140dB vs. 60dB), low power consumption, and high pixel bandwidth (on the order of kHz) resulting in reduced motion blur. Hence, event cameras have a large potential for robotics and computer vision in challenging scenarios for traditional cameras, such as low latency, high speed, and high dynamic range. Due to that event cameras work in a fundamentally different way from standard cameras, measuring per-pixel brightness changes (called “events”) asynchronously rather than measuring “absolute” brightness at a constant rate, novel methods are required to process their output and unlock their potential.

Object detection has been well-studied in computer vision for decades. It locates multiple objects in a given image or video by drawing bounding boxes and classifying their classes. Hence, the object detection model consists of not only a classifier that classifies multiple objects, but also a regressor that predicts the precise coordinate (row and column), and size (width and height), of the bounding boxes. Compared with image classification, object detection is considered a much more challenging task.

However, there are three challenges to developing an object detection framework for spike events obtained from neuromorphic cameras: learning algorithm, event encoding, and semi-automated data labeling.

This paper aims to bridge the gap between the time-continuous domain of events and frame-based computer vision algorithms. We propose a novel SNN object detection method by transferring successful deep CNNs, and a new training method to train the CNN with the spike images transduced by the spike events from neuromorphic cameras. We cast the event encoding problem as a frame integration problem, and investigate three different methods for event encoding. Our contributions can be summarized as follows:

- A visual way of imaging event data stream by neuromorphic encoding. It is studied through three algorithms: Frequency encoding [9], SAE (Surface of Active Events) [10], and LIF (Leaky Integrate and Fire) [11]. A novel event image method is developed to facilitate the training of the deep SNN model.

- The first 52-layer deep SNN for object detection. We present the SpikingYOLOv4 as an example that enables fast and accurate information transmission in the deep SNN. We developed a fine-grained normalization technique for the deep SNN called channel-wise normalization to replace batch normalization. RELU activation function and pooling layer are also revised for the transferring.

- A semi-automatic data labeling algorithm for training set augmentation of event images. By tracking objects in unlabeled event images, we can generate a labeled dataset for the training of the deep SNN. Event data is augmented for training and fine-tuning the deep SNN model.

The rest of the paper is organized as follows. Section III presents our proposed event-based object detection framework. Encoding and filtering of event stream data are explained in Section III-A. Section III-B elaborates on our proposed deep SNN method. Section III-C presents a semi-automatic approach to event image labeling to generate a labeled dataset for the training of the deep SNN. Experimental results are shown in Section IV. The conclusion is given in Section V.

## II. RELATED WORKS

Typical deep learning frameworks for object detection are divided into two categories [12]: two-stage and one-stage detection frameworks. In the two-stage detection frameworks, an additional component first obtains region proposals where the objects are likely to exist. Then, for each region proposal, the classifier determines which object is present, and the coordinate of the bounding box is produced by the regressor. R-CNN [13], fast R-CNN [14], faster R-CNN [15], and Mask RCNN [16] were proposed to perform the two-stage detection algorithm. The two-stage detection framework suffers from slow inference speed due to that processing through all possible region proposals is computationally expensive. The one-stage detection framework presents one unified network to extract region proposals and the bounding box information. You look only once (YOLO) [17] is known to be the state-of-the-art in one stage detection framework.

YOLO and its variants are a successful single-stage detection method. Likewise, YOLOs extract features at different scales with feature pyramid blocks and makes use of anchor boxes in the bounding box prediction. Different from YOLOv2 and YOLOv3, YOLOv4 [18] has a compound network structure composed of a backbone, neck, and head. Its backbone adopts CSPDarknet53 [18], its neck SPP [18] and PAN [18], and its head uses YOLOv3. YOLOv4 utilizes a Mish activation function [19] which has soft and non-monotonic properties. Besides, YOLOv4 adopts a Bag-of-X strategy to improve learning algorithms, including mosaic data augmentation, dropblock regularization, cosine annealing scheduler, class label smoothing, CIoU-loss, random training shapes, etc.

Object detection tasks have to locate objects under wide variability of scales [20]. Modern CNN object detection

methods divide the input image into regions and predict bounding box coordinates and class probabilities for each region. The most critical improvement of object-detection CNNs over traditional CNNs is multiscale feature representation. Hence, the modeling of contextual multi-scale information is important. Feature pyramid blocks are used in object detection methods to extract deep features at multiple image scales. Hence object detection models can perform inference over scale volumes and leverage contextual cues across different scales.

As event-based vision is rather new [21], only a marginal amount of work tackles object detection. Challenges of object detection on event data, feature representation, and learning mechanisms, are reviewed in this section.

### A. FEATURE REPRESENTATION

An event stands for an intensity change of a pixel, and is sparse and contains temporal information. Event is the basic information element and each event is represented as a quintuple (spacetime coordinates, polarity, and descriptor). Extracting high-level information from groups of events, also called event streams, is a critical issue for object detection and recognition.

LIDAR and radar are sparse signals that are similar to spiking events in neuromorphic computing. Zhou et al. [22] showed a spiking convolutional network for real-time 3D object detection by using LIDAR data. [23] present an SNN to effectively replace the constant false-alarm rate algorithms used for radar signals. These SNN systems are well-suited to recognize the temporal patterns in radar and LIDAR data, however, spatial-temporal data obtained from event cameras is more challenging than temporal signals.

Mapping the event stream to a dense representation [24], [25], [26], [27], [28], [29] is an important approach. Even if these methods lose some of the event's temporal resolution, they gain in terms of accuracy and scalability. A histogram method called "event cube" [24] has been proposed to combine the simplicity of histograms with the temporal information of time surfaces. Voxel grids [25], [26], voxel cube [27], graph [28], and DART (distribution aware retinal transform) [29] are successful examples of dense descriptors, statistical or structural, for event-based data. They preserve partially temporal information and asynchronous characteristics. They are bag-of-words local descriptors along temporal and/or spatial dimensions. However, these hand-crafted features need custom feature-matching algorithms to detect objects, and can not be used as input to standard approaches such as deep neural networks.

Converting groups of events into image-like representations, which are called event images in this paper, is another promising approach. Event images are dense and synchronous grid structures. They enable conventional object detection methods for image data, e.g. CNNs, to be applied to event data. CNNs can then be trained efficiently using back-propagation techniques for event images. Due to the higher signal-to-noise ratio of such representations with respect to

raw event data and the high capacity of deep neural networks, event image methods achieve promising results on the object detection task [30], [31], [32], [33].

Event stream encoding, also called neural coding [34], can be applied to convert spike events into event images. In general, all encoding techniques can be divided into two main categories [35]: rate and temporal coding. Temporal coding has been shown to offer a higher information capacity compared to rate codes, faster reaction times, and higher transmission speeds. It favors the utilization of local learning rules like STDP (Spike-timing-dependent plasticity) [36]. The strength of rate coding, however, is robust to noises. Rate coding is highly accurate, robust, and efficient on limited inference time steps. It is therefore often used in applications that convert CNNs into SNNs due to their equivalence to activation values. However, spike event recordings are spotted with dense noisy events caused by illumination and backgrounds, resulting in noisy event images which are not discussed in the literature. Filtering of event images according to neural coding is studied in this paper.

### B. DEEP SNN AND EVENT-BASED LEARNING

SNNs are commonly used with event representation for high-level vision tasks. While STDP, the unsupervised learning method proposed by the neuroscience community, is extensively studied as the machine learning algorithm of SNNs on image and object recognition, their limitations introduced the hybrid SNN architecture that can increase the depth of networks and improves accuracy. A parallel way inspired by the recent success of data-driven approaches in frame-based computer vision is to adopt deep neural networks and back-propagation learning into SNN, which motivates the CNN-to-SNN conversion approach. The following gives the reviews of SNN, hybrid SNN, and CNN-converted SNN methods for object detection.

#### 1) SNN

Novel network architecture models, eg. shallow SNNs [37], feedforward SNNs [38], and recurrent SNNs [39], directly tailored to the event data have been proposed. These SNNs perform inference through a dynamical system by processing events as asynchronous spike trains. The biologically realistic spiking neurons communicate using spike trains which do not have obvious derivatives. This makes SNNs unable to directly use derivative-based optimization for training. For the training on these SNNs, several approximation ways of gradient descent are proposed for supervised learning, and STDP is primarily used for unsupervised learning. However, the vast majority of these SNNs have been limited to very simple and shallow network architectures on relatively simple recognition datasets like MNIST and CIFAR-10. SNNs are difficult to train and currently achieve limited accuracy on high-level tasks, such as object detection. Their limited performance has necessitated a rethinking of extending network architectures of SNNs [40].

## 2) HYBRID SNN

Combining SNNs with other network structures can complement each other and increase network depth. Liu et al. [41] facilitate the event-based module by combining a frame-based CNN detector. RNNs (Recurrent Neural Networks) are also applicable to event data because RNNs are capable of remembering information over time. RED (Recurrent Event-camera Detector) [25] is an RNN composed of feed-forward convolutional layers followed by LSTM and SSD (Single Shot Detector) bounding box regression heads. Hybrid-SNN [42] proposes a partial solution by presenting an hybrid neural network composed of a SNN backbone for efficient event-based feature extraction, and an ANN head to solve object detection tasks. [27] offers a SNN composed of a spiking backbone and SSD bounding box regression heads to achieve qualitative results. Unfortunately, these SNNs still need to catch up to CNNs in terms of accuracy. [43] argues that hybrid SNNs lack well-developed training algorithms and appear to be in their infancy in current status.

## 3) CNN-CONVERTED SNN

CNN-to-SNN conversion could be the most promising approach. Tremendous advances in CNNs and deep learning have supplied highly accurate algorithms for computer vision. Deep neural networks are trained most often in a supervised manner using backpropagation. Vast amounts of labeled training examples are required, but the resulting accuracy is impressive, sometimes outperforming humans. Unfortunately, these algorithms and datasets are incompatible with event data and SNNs. Recent works overcome the lack of training algorithms and datasets by converting pre-trained deep convolutional networks to SNNs, achieving promising results even on complex tasks [44]. Conversion algorithms are designed by directly taking trained CNN parameters (e.g., weights and biases) to build up their respective SNN without actual training on the SNN. This is efficient since it avoids the risk of improper SNN training rules by applying CNN learning methods which are relatively mature. The CNN-to-SNN conversion methods achieved comparable performance in a deep SNN (e.g., VGG and ResNet) [45], [46], [47] compared to an original DNN, although they have focused solely on classification tasks. Spiking-YOLO [30] and fcYOLO [31] are CNN-converted SNNs proposed for spike-event object detection. The two works adopt the tiny-YOLO framework and have only 23 layers to extract visual features and classify results. However, these techniques have limited performance because they rely on shallow models and do not learn hierarchical features. The present work proposes a novel conversion method of a 52-layer YOLOv4 [48] trained with event images.

## C. TRAINING DATA

A vital success of deep learning comes from large-scale training data. MNIST [49], ImageNet [50], and MS-COCO [51] are the most representative datasets which are widely used

for image recognition tasks. Tavanaei et al. [43] argue that using deep neural networks for event-based object detection may achieve good performance only if lots of training data are provided. However, due to their sparse and asynchronous nature, traditional deep models cannot be readily applied to event data. Lagorce et al. [52] and Pfeiffer and Pfeil [53] pointed out that the static frame-based benchmarks used for CNNs are inappropriate for training spiking and event-based models. Although event cameras are becoming increasingly popular, only a few datasets for object detection in event streams are available [54]. In this paper, a semi-automatic labeling algorithm is developed to augment the amount of labeled training data from spike events. It is a data augmentation approach that can be applied to improve the accuracy of any event-image deep learning algorithms.

## III. EVENT-BASED OBJECT DETECTION FRAMEWORK

The proposed framework comprises three components: event stream encoding and filtering, deep SNN, and semi-automatic event image labeling. Figure 1 illustrates the process flow of the framework. The event camera first captures the motion of objects. The event camera generates a series of event stream data consisting of timestamps, coordinates, and polarity.

The first component, event stream encoding and filtering, converts temporal events into event images. Three conversion algorithms, Frequency, SAE, and LIF, are discussed. A series of filtering processing is then applied to reduce the noises of events and encoding.

The second component is our proposed object detection method, deep SNN. Our deep SNN model is first constructed as a CNN model. The CNN model is then trained with the CNN learning algorithm and transduced into an SNN model. The original batch normalization is replaced with our proposed fine-grained normalization, called channel-wise normalization. It leads to fast and accurate information transmission in the deep SNN. The added parameters allow us to dynamically adjust the firing rate of the pulse, reduce the weight of unnecessary spike pulses, and increase the importance of the object's essential features, ensuring that the neurons are affected by the correct pulse. On the other hand, a new activation function is adopted and improved to ensure the pulse can be triggered in the neuromorphic model. At the same time, the max pooling layer and the up-sampling layer are replaced with the convolutional layer and the transposed convolutional layer to perform the CNNs-to-SNNs conversion successfully.

The third component is semi-automatic event image labeling. An object tracking algorithm [55] is revised to semi-automatically track the bounding boxes of the objects in event images. The tracking results of event image sequences are adopted as labels for the training of the deep SNN model.

## A. EVENT STREAM ENCODING AND FILTERING

Event data cannot be directly used in traditional computer vision processing algorithms. We have to encode the event data stream and convert it into event image data. The event

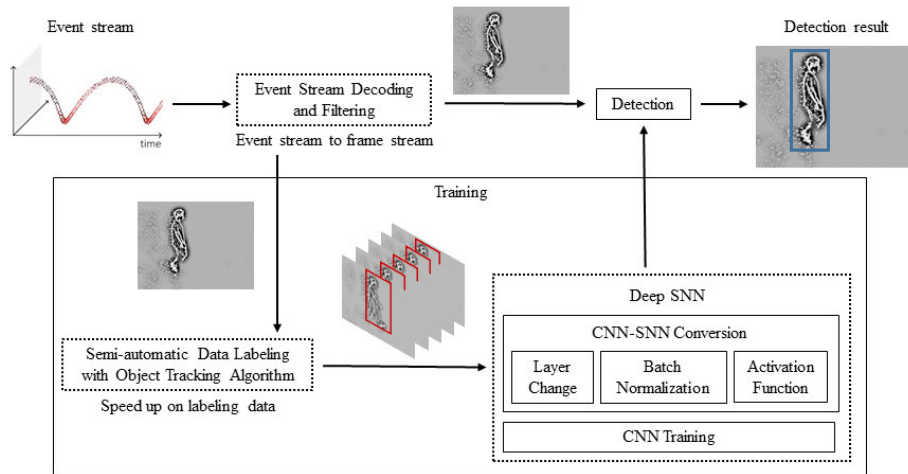


FIGURE 1. The proposed framework for event-based object detection.

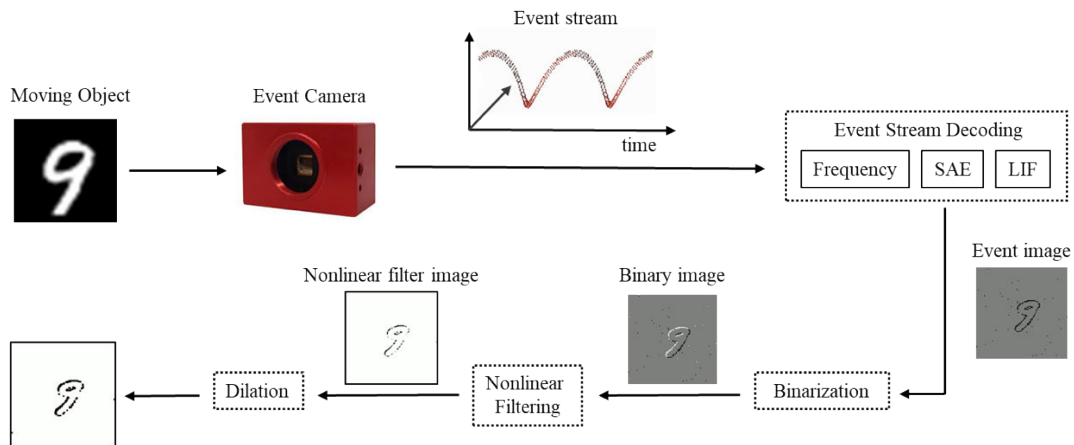


FIGURE 2. Event stream capturing and encoding.

camera records data through the objects trigger sensor in the scene. Because of the change of light when the object moves, it stimulates the sensor’s photosensitive element. Then it records the coordinate data of the point when the object moves or an event occurs, so the background is always stationary. There are three neuromorphic encoding methods, namely frequency, SAE, and LIF, to be studied in this paper.

In the frequency encoding method, image intensity is obtained by accumulating the number of events occurring in a period of time. It is defined as:

$$\sigma(n) = 255 \times 2 \times \left( \frac{1}{1 + e^{-n}} - 0.5 \right), \quad (1)$$

where  $n$  is the total number of all events that occurred in a particular pixel point in a fixed time interval, including positive and negative triggering events. The resulting  $\sigma(n)$  value is the pixel’s grayscale value in a specified time interval, and it is between 0 and 255. It is converted into a grayscale image composed of 8-bit grayscale values for each pixel.

The SAE encoding method takes advantage of the event camera’s low latency and accurate recording of the event occurrence time. Through the benefits of low latency, we record the occurrence of events in more detail and record a large amount of event data in a short period of time. The advantage of accurately recording the event occurrence time is that it allows us to clearly understand the sequence of events. As a result, combining two advantages of the event camera will enable us to understand the movement direction and speed of the objects in the event data. Each event data contains four data types: timestamp,  $x$ -coordinate,  $y$ -coordinate, and polarity. We define the event time point closest to the current state as  $t_p$ ,

$$t(x, y) \Rightarrow t_p \quad (2)$$

Regardless of the polarity of the event at this time point, only the pixel point time is recorded, and further, we have to calculate the pixel value of this point. The pixel value is also

the 8-bit grayscale value. The formula is defined as:

$$g(x, y) = 255 \times \frac{t_p - t_0}{T}, \quad (3)$$

where  $T$  is the time interval that composes each frame of the image,  $t_0$  is the first timestamp of each time interval  $T$ . According to the definition, it can be understood that the time when the event occurs is closer to the beginning of the time interval, the lower the pixel value will be. It also means that older events will be represented by a color closer to black, and newly generated events will be represented by white. However, this method also converts the noise at the same time. A large number of events will also cause excessive stacking, and the contours of objects will be less precise.

LIF encoding method assumes that each pixel in the image data is regarded as a neuron with the concept of membrane potential. The membrane potential will flow with time, or the input pulse changes. Regardless of the positive or negative trigger, the membrane potential of the pixel at that position will increase, and it will decline with the increase of time at a fixed ratio. When the membrane potential exceeds the threshold potential set in the domain, the pixel will be triggered to output a pulse signal. At the same time, the membrane potential will be reset back to 0 and no longer decay over time. Until the next input event is triggered, the membrane potential will continue to increase and decays over time with the fixed ratio.

When we get the trigger of a pixel, we continue to accumulate it in a fixed period of time. It will be reset to zero if no more triggers happen until the end of the period of time. We define the number of triggers obtained in each period of time as  $n$  to calculate the image gray level value. The highest number of times obtained is defined as  $n_h$ , and the grayscale value of each pixel is determined as:

$$g(x, y) = 255 \times \frac{n}{n_h} \quad (4)$$

From Eq. (4), it shows that the pixels with the most frequent event triggers will be closer to white, and no trigger or lower trigger will be more immediate to black. This method suppresses the generation of noise by adjusting the threshold potential. However, suppressing the noise may also make it difficult for the object to trigger the disappearance of the boundary generated by the event.

The event stream encoding and filtering process are illustrated in Figure 2. Binarization and filtering are the following processing steps for the event image obtained by encoding methods because the original event data has a polarity property. Therefore the pixel values in the event image remains polarity. As the object moves in different directions, the polarity of the object boundary changes. This changing polarity problem induces tracking errors, which can be solved by binarizing the grayscale values of pixels to maintain the consistency of the object boundary. Therefore, the threshold is set to be 200, and the binarized grayscale value of each

pixel is defined as:

$$\begin{cases} \hat{g}(x, y) = 255 & g(x, y) \leq 200 \\ \hat{g}(x, y) = 0 & g(x, y) > 200 \end{cases} \quad (5)$$

Nonlinear filtering, ex. the median filter of kernel size 3, is applied after the binarization to remove unnecessary background noise. While the filtering removes the noises in the event image, the edges of the target object are degraded and the outline of the object is unclear. We then restore the degraded boundary by morphological dilation.

## B. DEEP SNN

The proposed deep SNN comprises staircase activation function, channel-wise batch normalization, and upsampling and downsampling layers of multiscale representation for the conversion. The conversion from CNN to SNN is performed after learning. It means only network structure has to be considered in this paper, but the learning algorithm does not. A layer-to-layer and component-to-component approach achieve network conversion. The activation function, convolutional layer, batch normalization layer, and pooling layer must be converted.

Challenges of those conversions are accuracy loss due to the reduction of numerical precision from number to spike, and low firing rate of spiking neurons of deep networks. ReLU (Rectified Linear Unit) and traditional normalization layers are not compelling enough due to the low firing rate of neurons. Activation function and normalization layers are crucial because they improve spikes' firing rate and prevent conversion loss.

ReLU is the activation function generally considered for the CNN-to-SNN conversion for two reasons: ReLU is a dominant activation function used in CNN, and it bears functional equivalence to the Integrate-and-Fire (IF) spiking neuron without any leak and refractory period [56], [57], [58]. IF spiking neurons accumulate membrane potentials by integrating presynaptic spikes at each time step. Once the membrane potential accumulates beyond the spiking threshold, the neuron fires a spike, and the membrane potential reset to zero. Rueckauer et al. [58] extend the IF with a reset-by-subtraction mechanism that can have a higher firing rate and improve the conversion loss for deeper networks. The drawback of the ReLU and IF mechanism is that once a neuron becomes negative, it outputs a zero value and the information contained will be useless.

Leaky ReLU, such as the Mish function in YOLOv4, still retains positive values but keeps negative values with a leakage term, i.e.,  $f(x) = \alpha x$  when  $x < 0$ , otherwise  $f(x) = x$  [59]. It extends the function bound to the negative region and represents the leak and refractory period, similarly to the LIF spiking neuron. However, leaky ReLU is not biologically plausible because a biological spike is a discrete signal, not a continuous value. [30] adopts threshold methods to produce negative and positive spikes.

Here we propose an activation function formulated with a staircase function form and a reset-by-subtraction rule, which can reserve negative information and increase the firing rate. The staircase activation function is defined as follows:

$$\theta_i^l = \sum_{k=0}^1 H_k \left( V_i^l + t_k \right) - 1, \quad (6)$$

where  $V_i^l$  is the membrane potential of the spiking neuron  $i$  in layer  $l$ ,  $H_k$  is a Heaviside function, and  $t_k$  is the threshold. The function produces positive and negative pulses concerning two thresholds.  $\theta_i^l$  is a positive spike when  $V_i^l$  is greater than a positive threshold, and a negative spike when  $V_i^l$  is less than a negative threshold. No spike is produced if  $V_i^l$  is close to zero. This staircase function reserves more information for negative values and increases firing rate of spikes. In an SNN, it is essential to ensure that a neuron generates spike trains according to the magnitude of the input value of a neuron. Threshold voltage is responsible for sufficient and balanced activation of the neuron. For instance, under-activation occurs with a high threshold because the neuron needs to take a long period of time before  $V_i^l$  reaches the threshold. Vice versa, if the threshold is too small, then the  $V_i^l$  is most likely to exceed the threshold and generate spikes regardless of the input value of the neuron (i.e., over-activation). Both under- and over-activation cause low firing rates and induce lost information and poor performance [57].

After a spike is generated from the neuron, the membrane potential must be reduced. We apply the reset-by-subtraction rule when the membrane potential reaches the threshold voltage, subtracting the membrane potential with a threshold voltage value instead of resetting it to zero. The dynamic equation of the reset-by-subtraction rule is defined as

$$V_i^l(t) = V_i^l(t-1) - z_i^l(t) - \theta_i^l(t) \quad (7)$$

$$z_i^l(t) = \frac{dV_i^l(t)}{dt} = \sum_i w_{ij}^l \theta_i^{l-1}(t) + b_j^l \quad (8)$$

where  $V_i^l(t)$  is the membrane potential at time  $t$ ,  $\theta_i^l(t)$  is the output spike at time  $t$ ,  $z_i^l(t)$  is the sum of input currents from the neuron outputs of the previous layer at time  $t$ .  $w_{ij}^l$  and  $b_j^l$  are synaptic weights and bias. The reset-by-subtraction rule does not reset the membrane potential to zero when a pulse is generated. It only subtracts a threshold voltage value and continues to accumulate the potential. It maintains higher membrane potential, making it easier to generate pulses. Therefore, this rule increases the firing rate for small datasets and deep network models.

The batch normalization proposed by Peter U. Diehl et al. [57] is the most well-known data-based normalization method. It normalizes weights in a specific layer  $l$  by the maximum activation of the corresponding layer as follows:

$$\bar{w}^l = w^l \frac{\lambda^{l-1}}{\lambda^l} \text{ and } \bar{b}^l = \frac{b^l}{\lambda^l}, \quad (9)$$

where  $w^l$ ,  $\lambda^l$ , and  $b^l$  respectively, are weights, maximum activation, and bias of layer  $l$ . This layer-wise normalization adjusts weights and biases uniformly for all channels but

---

### Algorithm 1: CNN-SNN Conversion

---

**Input:** Pre-trained CNN unit  $C$ , weights  $w^l$ , maximum activation  $\lambda^l$ , bias  $b^l$  of the layer  $l$ , the max-pooling layer  $L_{max}$ , and the up-sampling layer  $L_u$  in the pre-trained CNN.  
**Output:** SNN with adopted activation function, channel-wise batch normalization layer, and converted convolutional layer.

- 1 Represent  $C$  by a population of spiking neurons according to [58] except the activation function.  
 //Adopt Eq. (6) as our activation function instead of Leaky ReLU.
- 2 **for**  $l$  in layers **do**
- 3     **for**  $i$  in spiking neurons **do**
- 4         **if** ( $V_i^l < t_k$  &&  $t_k < 0$ ) **then**
- 5              $\theta_i^l = -1$      // A negative spike  $\theta_i^l$
- 6         **else if** ( $V_i^l > t_k$  &&  $t_k > 0$ ) **then**
- 7              $\theta_i^l = 1$      // A positive spike  $\theta_i^l$
- 8         **else**
- 9              $\theta_i^l = 0$      // No spike is produced
- 10 Adjust the membrane potential after a spike is generated from the neuron in Eq. (7) and Eq. (8).  
 //Use channel-wise batch normalization to make information transmission faster and more efficient in deep SNNs [57].
- 11 **for**  $l$  in layers **do**
- 12     **for**  $j$  correspond to the channels in layers  $l$  **do**
- 13          $\bar{b}_j^l = b_j^l / \lambda_j^l$
- 14         **for**  $i$  correspond to the channels in layers  $l-1$  **do**
- 15             **if**  $l = \text{first layer}$  **then**
- 16                  $\bar{w}_{i,j}^l = w_{i,j}^l / \lambda_j^l$
- 17             **else**
- 18                  $\bar{w}_{i,j}^l = w_{i,j}^l \lambda_i^{l-1} / \lambda_j^l$
- 19 Replace the max-pooling layer  $L_{max}$  with the convolutional layer.
- 20 Use the transposed convolutional layer to replace the up-sampling layer  $L_u$ .
- 21 Convert convolutional  $L_{max}$  and  $L_u$  into a population of spiking neurons according to [58].

---

not specifically for a channel in the layer. The method over-suppresses neuron outputs, causing the membrane potential of neurons in the next layer to rise slowly. The firing rate of the neuron pulse will be pretty low, and many object features may be lost during the conversion process.

We adjust the normalization in a channel-wise manner. By normalizing the channel parameters individually, the feature characteristics of channels are channel-wisely adjusted, which will effectively increase the membrane potential and firing rate. It is defined as:

$$\bar{w}_{i,j}^l = w_{i,j}^l \frac{\lambda_i^{l-1}}{\lambda_j^l} \text{ and } \bar{b}_j^l = \frac{b_j^l}{\lambda_j^l} \quad (10)$$

where  $i$  and  $j$  correspond to the channels in layers  $l-1$  and  $l$  respectively. The fine-grained normalization compares the activation amounts of the same channels in the  $l-1$  layer and the  $l$  layer. If the activation amount is gradually reduced, the

ratio of the two activation amounts is more significant than one, and then the weight increase. Vice versa, the weight is gradually reduced to suppress the excessive activation amount, thereby suppressing the membrane potential and reducing the pulse trigger rate.

Algorithm 1 outlines the proposed CNN to SNN conversion method. The pseudocode is composed of activation function conversion, channel-wise batch normalization, and the conversion of mac pooling and upsampling layers. Max pooling is usually discarded in shallow SNN because the max-pooling layer in CNNs is applied to spatially down-sample the feature maps of image data. However, the multiscale feature representation of object detection networks needs both down-sampling and up-sampling processes to obtain the features with different scales. We keep the multiscale representation by replacing the max-pooling layer with the convolutional layer. To replace the up-sampling layer, the transposed convolutional layer is used. The softmax layer is not necessarily converted. We predict the output class corresponding to the neuron that generates spikes most during the presentation of the stimulus. By altering the input data of the original softmax layer into membrane potential, a spike caused by the membrane potential of a neuron will be used to determine the final classification result.

### C. SEMI-AUTOMATIC DATA LABELING WITH OBJECT TRACKING

The event image obtained by encoding event stream data can intrinsically achieve the separation of the foreground from the background. As shown in Figure 3, the event image is more accessible for CNN models to learn and predict object detection tasks. However, CNN requires a considerable number of manually labeled datasets for training, and traditional RGB image datasets are not applicable to our framework. We propose a data augmentation approach with an object tracking algorithm that is designed to automatically label the object positions in all event images instead of the first one and then augment labeled training data for CNN.

Several advanced tracking algorithms, such as Discriminative Correlation Filter with Channel and Spatial Reliability (CSR-DCF) [55], Kernelized Correlation Filter (KCF) [60], and Minimum Output Sum of Squared Error [61], can be employed in our framework. DCF and KCF are both based on fast Fourier transform and regularization to update the parameters in tracking. Still, errors of parameter updating are produced and aggregated when tracking objects with irregular shape or too many holes. CSR-DCF adds channel and spatial reliability to solve this problem and improve performance.

We adopt CSR-DCF as the core architecture of our tracking algorithm because of its robustness in tracking non-rectangular objects. An outwardly diffused circular weight space called spatial prior is centered on the manually labeled target object in the first event image. A spatial reliability map is obtained by backprojecting the foreground histogram calculated based on the spatial prior. The estimated spatial reliability map restricts correlation filters to the region of

irregularly shaped objects. Spatially constrained correlation filters are applied to extract filter responses, also called channel features of the target object. All channel features have corresponding channel weights. Both the filters and weights are adaptively updated through a learning optimization process in the following event images. A new target location in a new event image is obtained by weighted averaging filter response.

The revised CSR-DCF algorithm is described in the following using a non-deterministic form. Assume a set of  $N_d$ -dimensional channel features  $f = \{f_d\}_{d=1:N_d}$  and the corresponding correlation filter  $h = \{h_d\}_{d=1:N_d}$  for an event image, where  $f_d \in R^{d_w \times d_h}$  and  $h_d \in R^{d_w \times d_h}$ . The target object position  $x$  can be defined as the probability  $p(x|h)$ :

$$p(x|h) = \sum_{d=1}^{N_d} p(x|f_d) p(f_d) \quad (11)$$

The density  $p(x|f_d) = [f_d * h_d](x)$  is the convolution response of the channel feature at the target position  $x$ .  $p(f_d)$  is the channel reliability that will be abbreviated as  $w$  the weight considered as scaling factors based on the discriminative power of each channel feature.  $h$  has to be learned.

Assume there are previous target position  $x_{t-1}$ , previously learned filter  $h_{t-1}$ , previous channel reliability  $w_{t-1}$  for an event image at time  $t$ ,  $E_t$ . New target position  $x_t$  is estimated as the maximum of  $\sum_{d=1}^{N_d} [f_{d,t-1} * h_{d,t-1}](x_{t-1}) \times w_{d,t-1}$ . Then  $h_t$  and  $w_t$  are updated based on the information around the new location  $x_t$ . It is called online training that the training region of the event image  $E_t$  is centered at  $x_t$ .

The object model  $M$  has to be maintained and updated frame by frame. Here  $M_t = (w_t, y_t)$  represents the object model at time  $t$ , where  $y_t$  is the observation, i.e., the histograms of foreground and background event intensities.  $M_t$  is updated according to the exponential moving average with a predefined learning rate  $\eta$ , that is,  $M_t = (1-\eta)M_{t-1} + \eta M_t$ .

The optimized filter  $h_t$  is obtained through a complex optimization process called constrained correlation filter learning. A spatial reliability map is first constructed using a MRF (Markov random field) on the training patch. The MRF treats the prior and posterior label distributions over pixels as random variables. Figure 4 illustrates the construction of a spatial reliability map. The spatial reliability map is used to constrain the search of  $h_t$  by an alternative minimization process, which can be efficiently implemented by a fast Fourier transform algorithm. Spatial prior is derived from a training patch with a modified Epanechnikov kernel. Foreground and background are obtained by back-projection of the spatial prior. The posterior map is a refined spatial prior after MRF regularization. Finally overlaid patch shows the spatial reliability map.

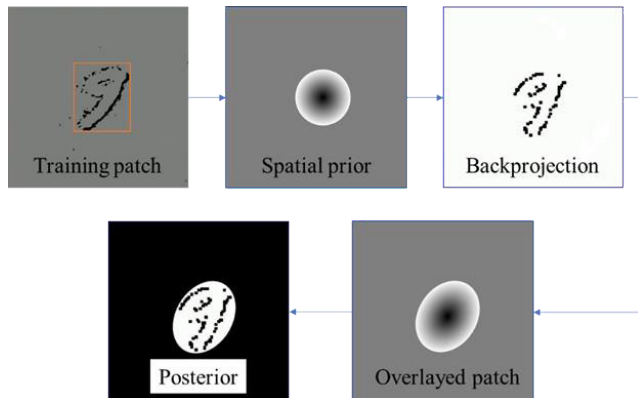
## IV. EXPERIMENTAL RESULTS

Experiments are conducted on benchmark datasets and our event data to validate the performance of the proposed framework. MNIST-DVS [31] and PAFBenchmark [54] are adopted in our experiments for preliminary study and pretraining. An FJU pedestrian detection (FJUPD) event





**FIGURE 3.** Images of the same scene. (a) RGB image, (b) Event image.

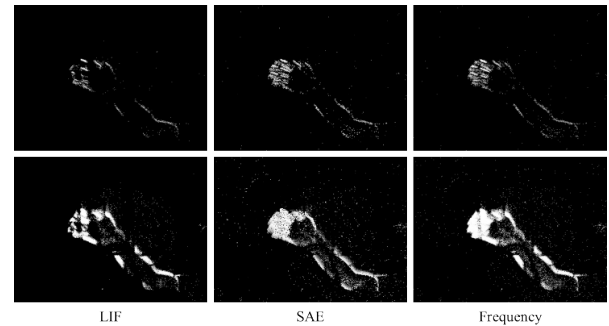


**FIGURE 4.** Construction of spatial reliability map for the filter training of the semi-automatic labeling.

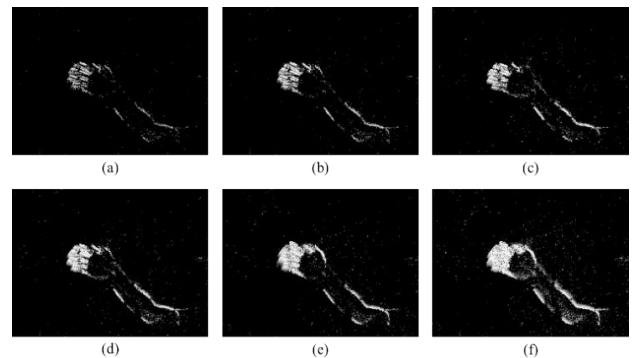
dataset was fabricated to evaluate the effects of semi-automatic labeling and the deep SNN model. The FJU dataset was created with the event camera DAVIS346 [31] and has two different backgrounds and three pedestrian scales. All the three datasets are stored in address event data (AEDAT) format. The AEDAT event data stores the coordinates, polarity, time stamp and other data of the event in a specific length whenever the sensor detects the occurrence of an event.

Existing methods, tiny YOLOv3, YOLOv4 [48], fcYOLO [31], and SpikingYOLO [30], are compared with our method. Performance metrics including power consumption are used for comparison. The algorithmic study of neuromorphic vision currently lacks well-developed software environment and tools. Therefore, we have to establish our development environment with a plethora of programming languages including Python, C/C++, and JAVA. Experimental environment is developed under Ubuntu with several supplemental library such as CUDA driver, cuDNN, OpenCV, and PyTorch. Neuromorphic tools developed by the University of Zurich such as Dynamic Vision software and jAER(Java tools for Address-Event Representation) are employed in our experiments for event data analysis and inspection. The source code for the event based object detection framework can be downloaded from the Github repository.<sup>1</sup>

<sup>1</sup>Source code: [https://github.com/fjcu-ee-islab/Spiking\\_Converted\\_YOLOv4](https://github.com/fjcu-ee-islab/Spiking_Converted_YOLOv4)



**FIGURE 5.** Visual comparison of three event encoding methods with two different step times, taking hand gestures for example. The time step of the first row is 10000, and the second row is 100000 SAE imaging analysis. The time steps of (a) to (f) are 10000, 20000, 30000, 40000, 50000, and 100000 respectively.



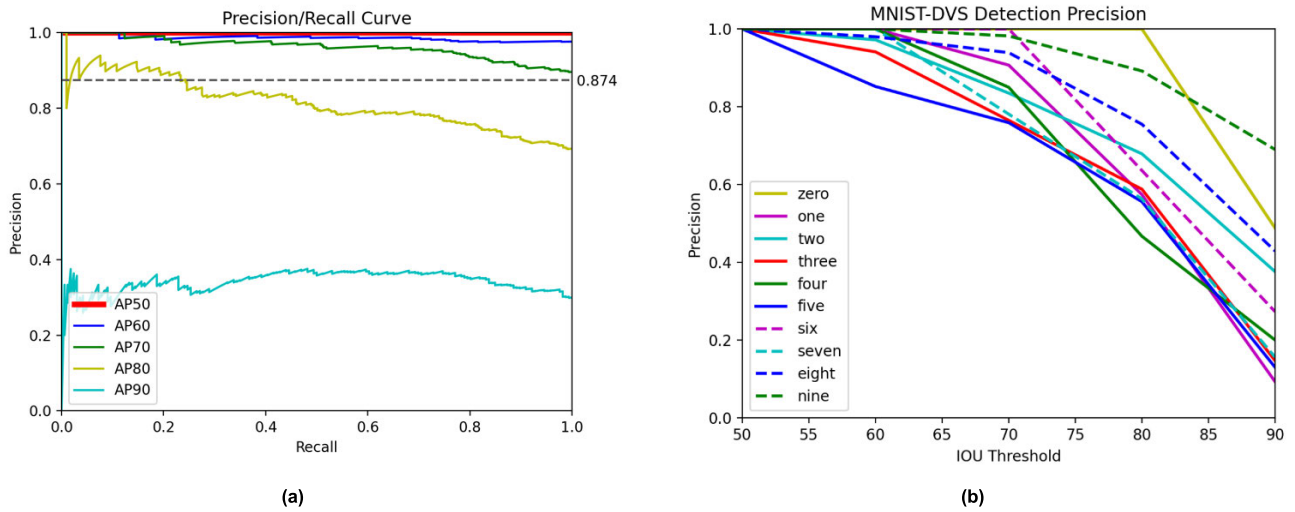
**FIGURE 6.** MNIST-DVS performance of the deep SNN model. (a) Precision/recall curves of all 10 classes. (b) Precision curves of individual classes.

The experiments are performed on NVIDIA GPU GTX 1080Ti. The event camera, DAVIS 346 [31], is a dynamic vision sensor that includes active pixel frame sensor and has  $346 \times 260$  pixels resolution. It offers significant advantages over standard frame-based cameras: only 20  $\mu$ s temporal latency, high dynamic range about to 120dB, only 30mW power consumption, and 12 mega-events per second bandwidth resulting in reduced motion blur and transmitting event data to software platform in time.

#### A. PRELIMINARY ANALYSIS

We first compare the three event encoding methods. The three encoding methods: frequency, SAE, and LIF, are compared with the event streams captured by the DAVIS346. The time unit of events is 20 microseconds. The three methods have to accumulate enough events to encode pixel values, which is called time step. Time steps from 10000 to 100000 events are experimented.

Figure 5 gives the encoding results of the three methods with two time steps. With more time steps all methods can encode more contour information of objects, but at the expense of more noises. Frequency method gives worse object contour for both short and long time steps. While SAE and LIF have better object contours, LIF encounters the issue of broken object contour in complex background.



**FIGURE 7.** SAE imaging analysis. The time steps of (a) to (f) are 10000, 20000, 30000, 40000, 50000, and 100000 respectively.

Figure 6 gives more time step results with respect to SAE. The decision of time steps will reply on object motion. For objects moving slowly, it is better to choose a longer time step to increase object's detail and contour. For fast moving objects, a shorter step-time can avoid object's shadow. A compromise between object contour and noise has to be considered for time steps. The following experiments will adopt SAE because of the compromise. Time step 20000 is used in the following experiments.

The proposed method, the deep SNN model trained with the data augmentation, is first evaluated with MNIST-DVS. MNIST is a small benchmark dataset originally developed for classification task. But it was adapted for event object detection with a re-capturing process from the MNIST by DVS128 event camera. The produced MNIST-DVS dataset was then stored with event stream data format of the size  $128 \times 128$ , and has three scales for each number class. Every scale is composed of 1000 AEDAT files, and each AEDAT file contains about 24 seconds of event stream. A total of 5912 labeled images as the training set is generated with the semi-automatic data labeling algorithm. 590 validation images are generated with the semi-automatic data labeling algorithm, and 590 manually labeled images are used as the test set.

IOU (Intersection over Union) are used to obtain precision and recall. IOUs with different thresholds 0.5, 0.6, 0.7, 0.8, and 0.9 are represented as AP50, AP60, AP70, AP80 and AP90. From Figure 7 (a), it can be seen obviously that the model performance decreases with the increase of the IOU threshold. Compared with the precision 0.874 of *fcYOLO* [31] which did not provide its IOU threshold and precision-recall curve, our precision is higher in all cases of AP50, AP60, AP70. The precision of each class is further illustrated in Figure 7(b). All classes can achieve the precision higher than 80% in AP70. Figure 8 illustrates the detection results with AP70. Most prediction boxes are larger than the label boxes but the prediction results are correct visually.

Since the proposed method with the threshold AP70 can successfully detect the number objects in MNIST-DVS dataset with good accuracy, AP70 is used in our experiments.

A pedestrian detection experiment done with PAFBenchmark is given in Figure 9. PAFBenchmark has 4665 labeled event images. 3,699 images are used for training, 482 images for validation, and 484 images for test. As shown in Figure 9 our method has greatly high accuracy in AP50. The accuracy does not drop significantly even for AP60 and AP70.

## B. PEDESTRIAN DETECTION

This section evaluates the performance of our method on pedestrian detection. In addition to the PAFBenchmark dataset, we create the FJUPD dataset to perform more detailed experiments. Both datasets are captured by DAVIS346 and have event images with the size of  $346 \times 260$ . The event images in this section are encoded by SAE, with time steps ranging from 10,000 to 20,000. The analysis of the data augmentation and pre-training is experimented.

The FJUPD has two background situations and three object scales. Simple and complex backgrounds are differentiated according to light variations and shadow changes. Complex background has rich light changes and moving shadows, which induce challenges to object detection. Three object scales according to the proportion of pedestrians in the scene are established. Each object scale has 12 event sequences ranging from 5 to 7 seconds, so there are a total of 72 event sequences. As shown in Figure 10, the edges of pedestrian are obviously less defective in simple background.

The FJUPD dataset is used for the test of the proposed method. Tests of different training models with respect to data augmentation and pre-training are conducted. The training model could be pre-trained by PAFBenchmark and then trained with or without semi-automatically labeled data. There are four models trained with: manually labeled image data and without pre-training, manually labeled image data

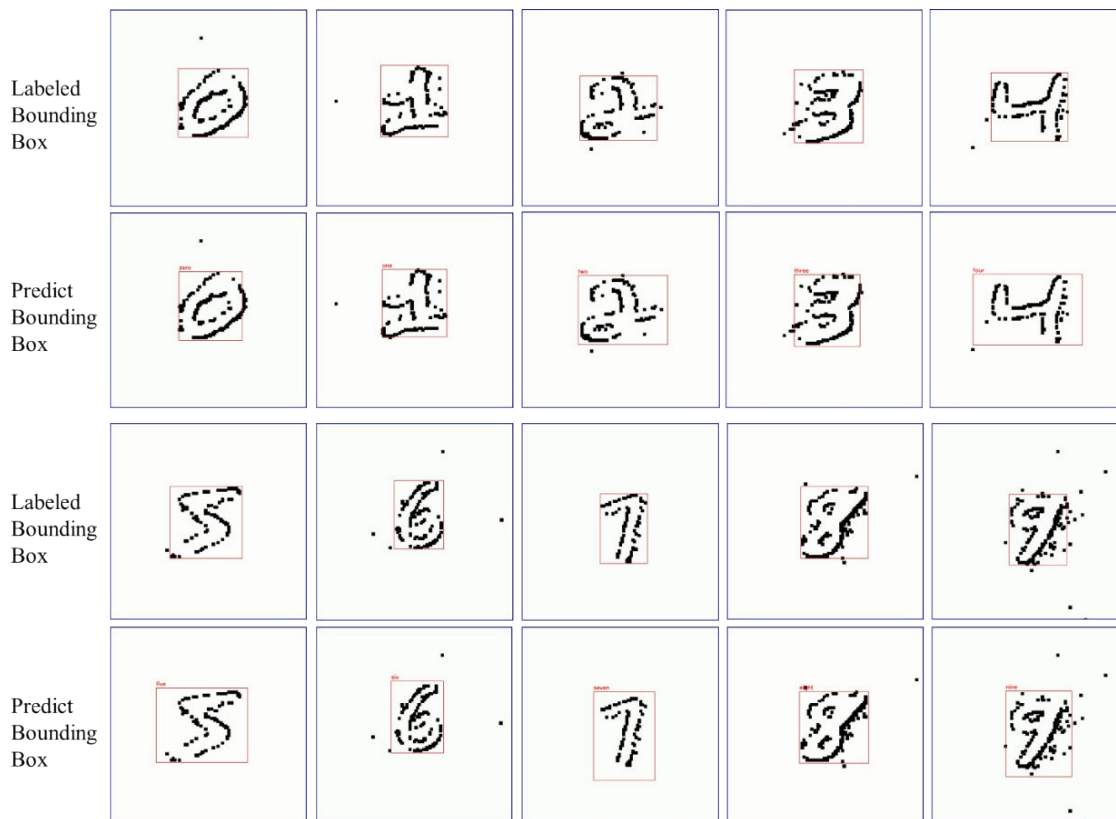


FIGURE 8. Comparison between MNIST-DVS Detection Labeled Bounding Box and Predict Box.

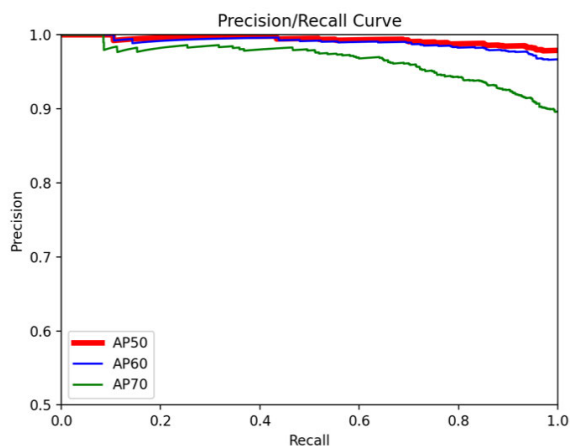


FIGURE 9. Precision and recall results of PAFBenchmark experiment.

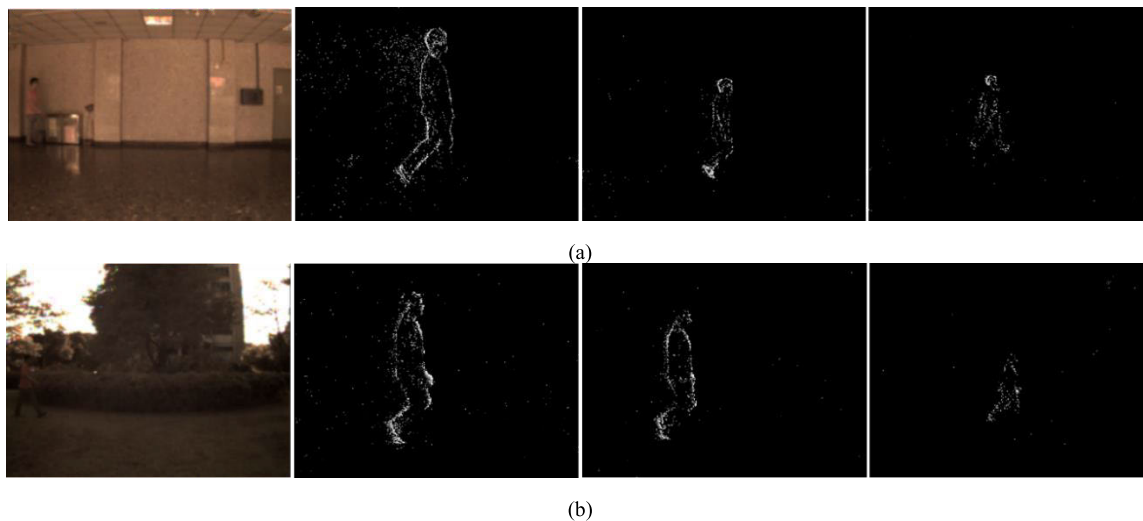
and with pre-training, semi-automatically labeled image and without pre-training, and semi-automatically labeled image data and with pre-training.

The pre-training adopts all data of PAFBenchmark for training. Test data comes from FJUPD. The models with pre-training are called PAFBPretrain. The data augmentation adopts semi-automatically labeled FJUPD data to train. 1157 event images are labeled manually, where 684 manually-labeled data are used as a common test

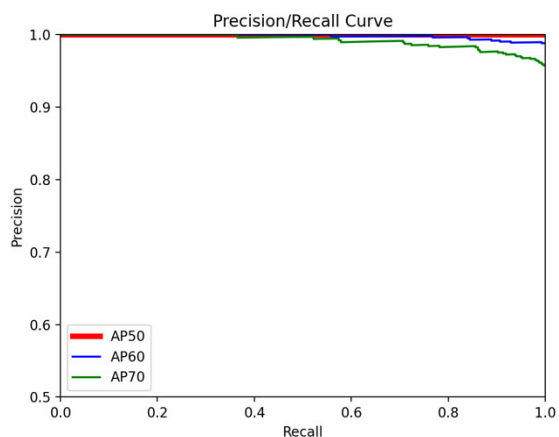
set for all the four models, 377 images for training, and 96 for validation. On the other hand, there are 7,526 semi-automatically labeled image data as training sets and 684 semi-automatically labeled image data for validation.

The comparisons are shown in Figure 11. Figures 11(a)(b) are results with manual labels, and Figures 11(c)(d) are results with semi-automatic labels. Comparing Figures 11(a)(b) illustrate the effects of pre-training. It can be found that when the PAFBPretrain model is used, there is a slight decrease of the precision result in the case of AP50 and AP60. But there is a relatively high increase in the cases of AP70. The increase of up to 10% clearly reflects the effectiveness of the PAFBPretrain model. Similar results of performance improvement by PAFBPretrain can be observed from Figures 11(c)(d). Comparing Figures 11(a)(c) and Figures 11(b)(d) show the effects of semi-automatic labeling. While the performance drops when semi- automatic labeling is adopted, the combination of semi-automatic labeling with PAFBPretrain can achieve comparable performance, as shown in Figure 11(d).

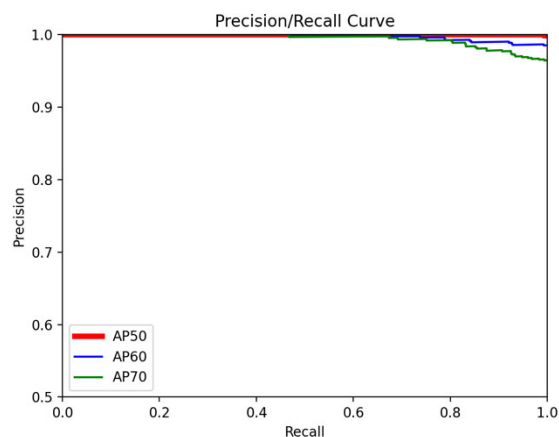
Figure 12 shows F-measure of the four models especially for AP50 and AP70. It can be found that the use of the pre-trained model can effectively improve F-measure. The effect of the training data generated by semi-automatic labeling can be very close to that of the manually labeled training data. The performance difference is not significant and can be neglected, as proved in Figure 13. The detection results of



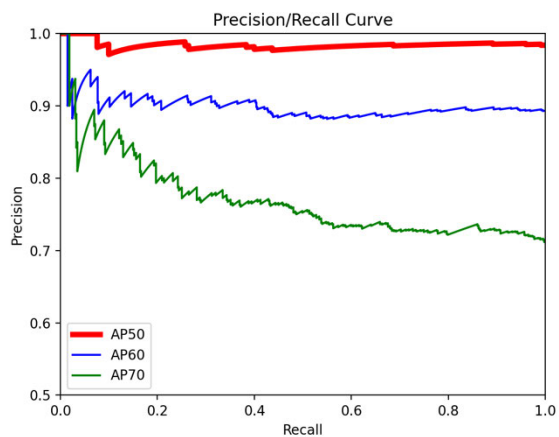
**FIGURE 10.** Multi-scale event images in (a) simple background and (b) complex background. Images in the first column are the RGB frame images introducing the scene, and images in the second to the fourth column are respectively large, medium, and small-scale object event images.



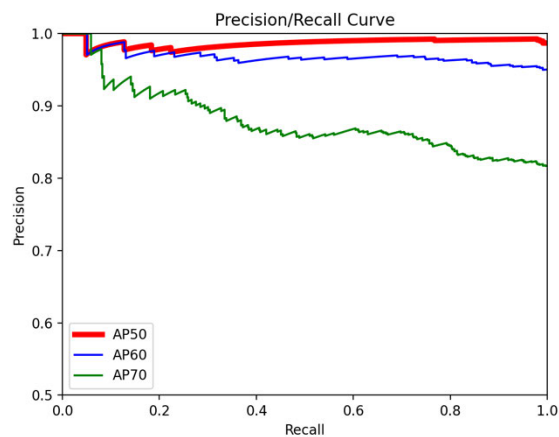
(a)



(b)

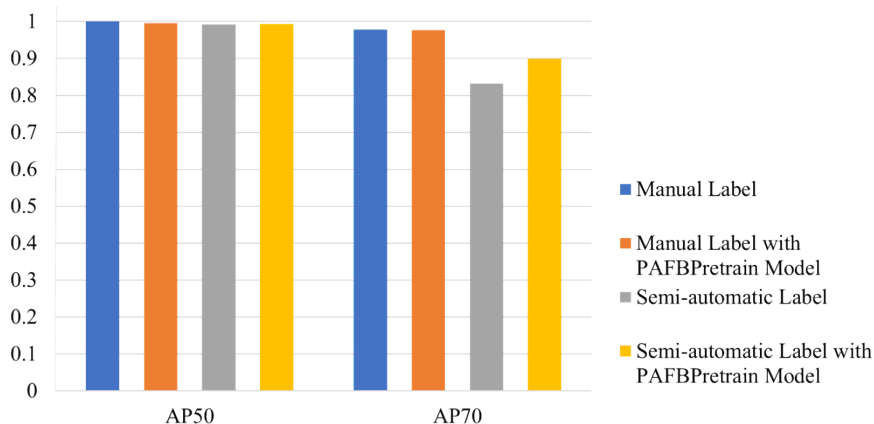


(c)

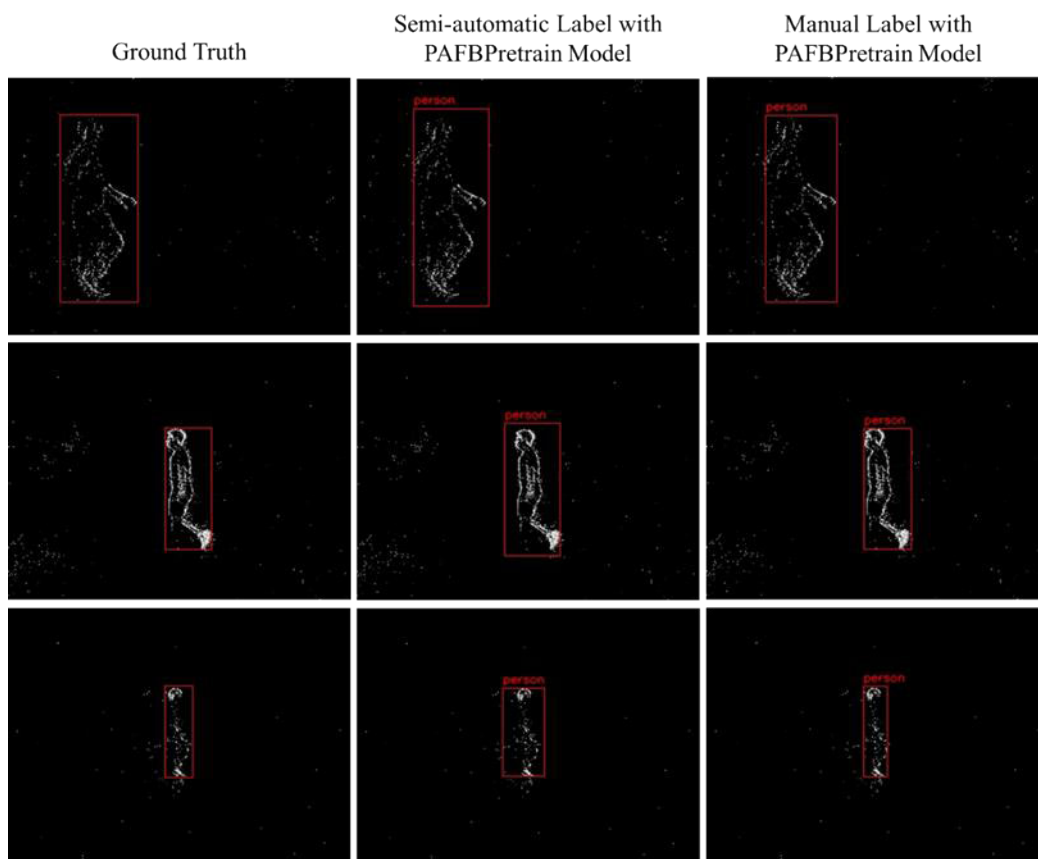


(d)

**FIGURE 11.** The F-measure scores of AP50 and AP70.



**FIGURE 12.** Detection results of three scales and two models. The object scales from above to below respectively are large, normal, and small. The middle column are the results of semi-automatically labelling with PAFBPretrain model, and the right column the results of manually labelling with PAFBPretrain model.



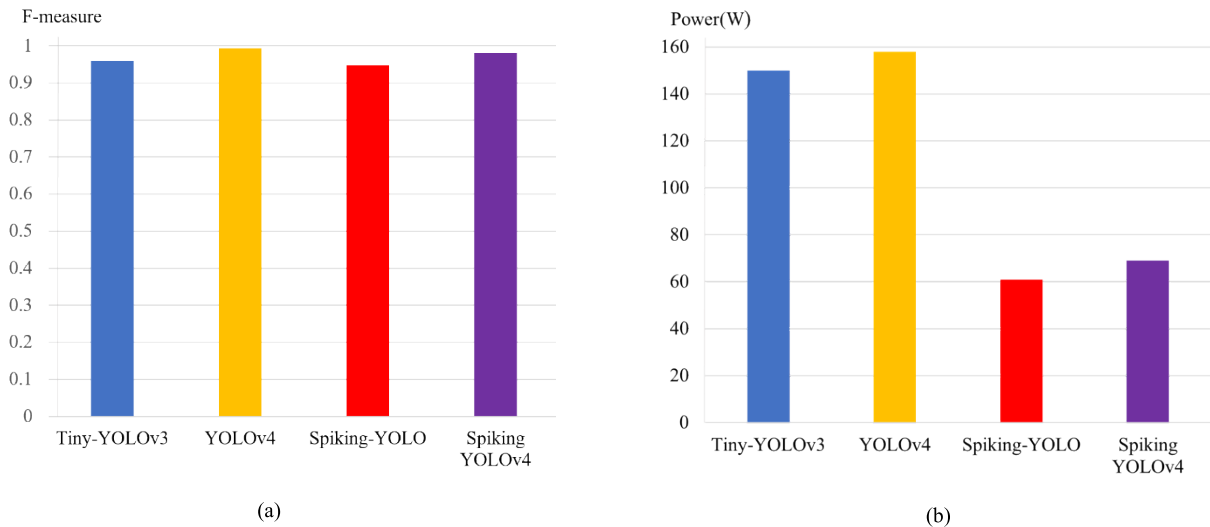
**FIGURE 13.** The comparison of each model on (a) F-measure score and (b) power consumption.

semi-automatically labelling are larger than the ground truth, which decrease the IOU and F-measure. But the results are still correct and can be claimed to be accurate.

**C. STATE-OF-THE-ART COMPARISON**

Performance comparisons with existing SNNs and object detection models are evaluated. Tiny YOLOv3, fcYOLE [31],

SpikingYOLO [30], and YOLOv4 [48] are compared using the MNIST-DVS dataset. In this experiment our deep SNN method adopts YOLOv4 as a basic CNN model for conversion, and uses the FJUPD dataset and semi-automatic labeling algorithm for data augmentation. Our method is called SpikingYOLOv4 in the following discussions. Manually labeled test set is used to obtain the performance metrics, where



**FIGURE 14.** Pedestrian detection test results of the FJUPD dataset. Test results are obtained with respect to four different training models trained with (a) manually labeled image data, (b) manually labeled image data and with PAFBPretrain, (c) semi-automatically labeled image, and (d) semi-automatically labeled image data with PAFBPretrain.

especially power consumption is also compared. We use the semi-automatically labeling algorithm to generate a total of 5,912 labeled images for training. Their size is 438\*438. The labeling algorithm generates 590 labeled images for validation, and 590 manually labeled image data are used as the test set.

The performance of SpikingYOLOv4 on FJU event pedestrian detection is shown in Figure 14. Both accuracy and power consumption are evaluated. Figure 14(a) shows that our SpikingYOLOv4 gets better accuracy than Tiny-YOLOv3 and Spiking-YOLO. Comparing SpikingYOLOv4 with YOLOv4, while both have comparative accuracy, our power consumption is significantly reduced as shown in Figure 14(b), which proves that SNNs have more power advantages than CNNs. The four programs are executed on NVIDIA GPU, and the power consumptions are evaluated by NVIDIA System Management Interface (NVSMI) [62] which reads the power consumption metric of GPU sensor. NVSMI is a platform-independent command line utility based on NVIDIA Management Library (NVML) [63].

## V. DISCUSSIONS AND CONCLUSION

Three important issues of high-level event-based vision, namely feature representation, deep SNNs, and training data, are used to be addressed separately in literatures. This paper tackles the three issues together by proposing a framework with three algorithms. The proposed framework uses SNNs to detect objects, but trains CNNs with event images. Asynchronous sparse events are neural-encoded into event images, a noise-robust grid representation that can be connected to conventional deep neural networks of object detection. However, the training of event images for deep CNN has the challenge of feeding large-scale dataset, which is solved by our semi-automatic labeling algorithm. The trained

deep CNN is converted to SNN with the critical designs: staircase activation function, channel-wise batch normalization, and upsampling and downsampling layers of multi-scale representation. The first 52-layer deep SNN for object detection, Spiking-YOLOv4, is presented as an example that enables fast and accurate information transmission in the deep SNN. While the success of the proposed framework comes at the cost of synchronization and densification, a high accuracy SNN model is achieved. Thresholding, firing rate and various parameters may have influences on the accuracy, however, detailed empirical study can be further explored.

The proposed framework of event object detection includes event stream encoding, multiscale deep SNN, and event-image data augmentation. The spike events captured from the event camera are first decoded into event images for the learning of deep CNN models. The conversion of multiscale network representation is discussed and designed in our deep SNN method. A semi-automatic data augmentation algorithm is devised by using a visual tracking algorithm to label the objects in event images.

Experiments of three different encoding schemes are evaluated with discussions of their advantages and disadvantages. The results on the MNIST-DVS benchmark dataset show that the proposed algorithmic flow successfully converts an existing CNN into a spike-event domain application and still has high accuracy. A detailed performance analysis conducted on the FJUPD dataset demonstrates the multiscale representation of the SpikingYOLOv4 network. The final PAFBenchmark experiment shows that our proposed method has higher accuracy than existing SNN methods, better energy efficiency, and lower energy consumption than current CNN methods. It demonstrates that our deep SNN method is a feasible solution for studying neuromorphic vision.

The intuition that deep SNN trained with more data can achieve better accuracy is also confirmed.

Future works are planned to conduct experiments on more complex datasets, such as hand gesture detection, pedestrian detection, and image classification. The softmax layer can be further improved. Extending the deep SNN into a more complex network structure will be an exciting challenge.

## REFERENCES

- W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Netw.*, vol. 10, no. 9, pp. 1659–1671, Dec. 1997.
- J. D. Nunes, M. Carvalho, D. Carneiro, and J. S. Cardoso, "Spiking neural networks: A survey," *IEEE Access*, vol. 10, pp. 60738–60764, 2022.
- H. Mostafa, B. U. Pedroni, S. Sheik, and G. Cauwenberghs, "Fast classification using sparsely active spiking networks," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2017, pp. 1–4.
- S. Al-Obaidi, H. Al-Khafaji, and C. Abhayaratne, "Making sense of neuromorphic event data for human action recognition," *IEEE Access*, vol. 9, pp. 82686–82700, 2021.
- N. Aparajita, P. K. Sa, S. K. Choudhury, S. Bakshi, and B. Majhi, "A neuromorphic person re-identification framework for video surveillance," *IEEE Access*, vol. 5, pp. 6471–6482, 2017.
- M. J. Dominguez-Morales, A. Jimenez-Fernandez, G. Jimenez-Moreno, C. Conde, E. Cabello, and A. Linares-Barranco, "Bio-inspired stereo vision calibration for dynamic vision sensors," *IEEE Access*, vol. 7, pp. 138415–138425, 2019.
- P. Lichtsteiner, C. Posch, and T. Delbruck, "A  $128 \times 128$  120 db 30 mw asynchronous vision sensor that responds to relative intensity change," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2006, pp. 2060–2069.
- C. Posch, D. Matolin, and R. Wohlgenannt, "A QVGA 143 dB dynamic range asynchronous address-event PWM dynamic image sensor with lossless pixel-level video compression," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2010, pp. 400–401.
- N. F. Y. Chen, "Pseudo-labels for supervised learning on dynamic vision sensor data, applied to object detection under ego-motion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 644–653.
- E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza, "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM," *Int. J. Robot. Res.*, vol. 36, no. 2, pp. 142–149, Feb. 2017.
- A. N. Burkitt, "A review of the integrate-and-fire neuron model," *Biol. Cybern.*, vol. 95, pp. 1–19, Jan. 2006.
- L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," in *Proc. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 1–15.
- R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.
- R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.
- S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Comput. Vis. Pattern Recognit. (CVPR)*, 2015, pp. 1–13.
- K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2961–2969.
- J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.
- A. Bochkovskiy, C.-Y. Wang, and H.-Y. Mark Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.
- D. Misra, "Mish: A self regularized non-monotonic activation function," 2019, *arXiv:1908.08681*.
- Y.-K. Wang, J. Guo, and T.-M. Pan, "Multidomain joint learning of pedestrian detection for application to quadrotors," *Drones*, vol. 6, no. 12, p. 430, Dec. 2022.
- G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, "Event-based vision: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 1, pp. 154–180, Jul. 2022.
- J. López-Randulfe, T. Duswald, Z. Bing, and A. Knoll, "Spiking neural network for Fourier transform and object detection for automotive radar," *Frontiers Neurobot.*, vol. 15, Jun. 2021, Art. no. 688344.
- A. Shalumov, R. Halaly, and E. E. Tsur, "LiDAR-driven spiking neural network for collision avoidance in autonomous driving," *Bioinspiration Biomimetics*, vol. 16, no. 6, Nov. 2021, Art. no. 066016.
- Prophesee. *Event Preprocessing Tutorial—Event Cube*. Accessed: Dec. 14, 2022. [Online]. Available: [https://docs.prophesee.ai/stable/metavision\\_sdk/modules/ml/data\\_processing/event\\_preprocessing.html#event-cube](https://docs.prophesee.ai/stable/metavision_sdk/modules/ml/data_processing/event_preprocessing.html#event-cube)
- E. Perot, P. D. Tournemire, D. Nitti, J. Masci, and A. Sironi, "Learning to detect objects with a 1 megapixel event camera," in *Proc. Neural Inf. Process. Syst. (NeurIPS)*, 2020, pp. 16639–16652.
- M. Cannici, M. Ciccone, A. Romanoni, and M. Matteucci, "A differentiable recurrent surface for asynchronous event-based data," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 136–152.
- L. Cordone, B. Miramond, and P. Thierion, "Object detection with spiking neural networks on automotive event data," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2022, pp. 1–8.
- Y. Bi, A. Chadha, A. Abbas, E. Bourtsoulatzé, and Y. Andreopoulos, "Graph-based spatio-temporal feature learning for neuromorphic vision sensing," *IEEE Trans. Image Process.*, vol. 29, pp. 9084–9098, 2020.
- B. Ramesh, H. Yang, G. Orchard, N. A. Le Thi, S. Zhang, and C. Xiang, "DART: Distribution aware retinal transform for event-based cameras," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 11, pp. 2767–2780, May 2019.
- S. Kim, S. Park, B. Na, and S. Yoon, "Spiking-YOLO: Spiking neural network for real-time object detection," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 11270–11277.
- M. Cannici, M. Ciccone, A. Romanoni, and M. Matteucci, "Asynchronous convolutional networks for object detection in neuromorphic cameras," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2019, pp. 1–10.
- B. Ramesh, A. C. U. Caycedo, L. D. Vedova, H. Yang, and G. Orchard, "PCA-RECT: An energy-efficient object detection approach for event cameras," in *Proc. Asian Conf. Comput. Vis.*, 2018, pp. 434–449.
- O. Johansson, "Training of object detection spiking neural networks for event-based vision," Ph.D. dissertation, Dept. Comput. Sci., Elect. Space Eng., Luleå Univ. Technol., Luleå, Sweden, 2021.
- W. Guo, M. E. Fouda, A. M. Eltawil, and K. N. Salama, "Neural coding in spiking neural networks: A comparative study for robust neuromorphic systems," *Frontiers Neurosci.*, vol. 15, Mar. 2021, Art. no. 638474.
- D. Auge, J. Hille, E. Mueller, and A. Knoll, "A survey of encoding techniques for signal processing in spiking neural networks," *Neural Process. Lett.*, vol. 53, no. 6, pp. 4693–4710, Dec. 2021.
- P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers Comput. Neurosci.*, vol. 9, p. 99, Aug. 2015.
- S. R. Kheradpisheha, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, "STDP-based spiking deep convolutional neural networks for object recognition," *Neural Netw.*, vol. 99, pp. 56–67, Mar. 2018.
- S. B. Shrestha and G. Orchard, "SLAYER: Spike layer error reassignment in time," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1412–1421.
- W. Zhang and P. Li, "Spike-train level backpropagation for training deep recurrent spiking neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 7802–7813.
- B. Illing, W. Gerstner, and J. Brea, "Biologically plausible deep learning—But how far can we go with shallow networks?" *Neural Netw.*, vol. 118, pp. 90–101, Oct. 2019.
- H. Liu, D. P. Moeyes, G. Das, D. Neil, S.-C. Liu, and T. Delbruck, "Combined frame- and event-based detection and tracking," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2016, pp. 2511–2514.
- A. Kugele, T. Pfeil, M. Pfeiffer, and E. Chicca, "Hybrid SNN-ANN: Energy-efficient classification and object detection for event-based vision," in *Proc. DAGM German Conf. Pattern Recognit.*, 2021, pp. 297–312.
- A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, "Deep learning in spiking neural networks," *Neural Netw.*, vol. 111, pp. 47–63, Mar. 2019.
- B. Rückauer, "Event-based vision processing in deep neural networks," Ph.D. dissertation, Inst. Neuroinform., Univ. Zürich, Zürich, Switzerland, 2020.

- [45] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: VGG and residual architectures," *Frontiers Neurosci.*, vol. 13, pp. 1–10, Mar. 2019.
- [46] Q. Yu, C. Ma, S. Song, G. Zhang, J. Dang, and K. C. Tan, "Constructing accurate and efficient deep spiking neural networks with double-threshold and augmented schemes," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 4, pp. 1714–1726, Apr. 2022.
- [47] Y. Hu, H. Tang, and G. Pan, "Spiking deep residual network," 2018, *arXiv:1805.01352*.
- [48] A. Bochkovskiy, C. Wang, and H. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," in *Proc. Comput. Vis. Pattern Recognit. (CVPR)*, 2020, pp. 1–17.
- [49] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
- [50] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
- [51] T. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. Lawrence Zitnick, and P. Dollár, "Microsoft COCO: Common objects in context," in *Proc. Comput. Vis. Pattern Recognit. (CVPR)*, 2014, pp. 740–755.
- [52] X. Lagorce, G. Orchard, F. Gallupi, B. E. Shi, and R. Benosman, "HOTS: A hierarchy of event-based time-surfaces for pattern recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 7, pp. 1346–1359, Jan. 2017.
- [53] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: Opportunities and challenges," *Frontiers Neurosci.*, vol. 12, no. 774, pp. 1–18, 2018.
- [54] S. Miao, G. Chen, X. Ning, Y. Zi, K. Ren, Z. Bing, and A. Knoll, "Neuromorphic vision datasets for pedestrian detection, action recognition, and fall detection," *Frontiers Neurobot.*, vol. 13, p. 38, Jun. 2019.
- [55] A. Lukezic, T. Vojir, L. C. Zajc, J. Matas, and M. Kristan, "Discriminative correlation filter with channel and spatial reliability," *Int. J. Comput. Vis.*, vol. 126, no. 7, pp. 671–688, Jan. 2018.
- [56] Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," *Int. J. Comput. Vis.*, vol. 113, no. 1, pp. 54–66, 2015.
- [57] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2015, pp. 1–8.
- [58] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers Neurosci.*, vol. 11, p. 699, Dec. 2017.
- [59] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1–12.
- [60] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 583–596, Mar. 2015.
- [61] D. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 2544–2550.
- [62] NVIDIA. *NVIDIA System Management Interface*. Accessed: Apr. 1, 2021. [Online]. Available: <https://developer.nvidia.com/nvidia-system-managementinterface>
- [63] NVIDIA. *NVIDIA Management Library (NVML)*. Accessed: Apr. 1, 2021. [Online]. Available: <https://developer.nvidia.com/nvidiamanagement-library-nvml>



**YUAN-KAI WANG** (Senior Member, IEEE) received the B.S. degree in electrical engineering and the Ph.D. degree in computer science and information engineering from National Central University, in 1990 and 1995, respectively.

He was a Postdoctoral Fellow with the Institute of Information Science of Academia Sinica, from 1995 to 1999. Since 1999, he has been joining the Department of Electrical Engineering, Fu Jen Catholic University, as an Associate Professor, and as a Professor, since 2017. His research interests include computer vision, pattern recognition, neural networks, genetic algorithms, machine learning, and artificial intelligence, with special focus on video surveillance, face recognition, biometrics, robotic vision, embedded computer vision, and medical image analysis.

Dr. Wang has been the chair, the co-chair, and a program committee member of many international conferences, and a reviewer of many journals and IEEE TRANSACTIONS. He served the Chinese Image Processing and Pattern Recognition Society for the Board of Directors and Supervisors, from 2004 to 2022. He served the Information Service Association of Chinese Colleges for the Board of Directors, from 2005 to 2007. He was invited as a Panel Speaker for the International Conference of Pattern Recognition, in 2014.



**SHAO-EN WANG** received the B.S. and M.S. degrees in electrical engineering from Fu Jen Catholic University, New Taipei City, Taiwan, in 2018 and 2021, respectively. From 2018 to 2021, he was a Research Assistant at the Intelligent System Laboratory, Fu Jen Catholic University.



**PING-HSIEN WU** received the B.S. degree in electrical engineering from Fu Jen Catholic University, New Taipei City, Taiwan, in 2019, where he is currently pursuing the M.S. degree in electrical engineering. Since 2019, he has been a Research Assistant at the Intelligent System Laboratory, Fu Jen Catholic University.

• • •