SAEN-BGS: Energy-efficient spiking autoencoder network for background subtraction

Zhixuan Zhang [a], Xiao Peng Li [b], Qi Liu [a],[1],*

[a] School of Future Technology, South China University of Technology, Guangzhou, 511442, Guangdong, China
[b] State Key Laboratory of Radio Frequency Heterogeneous Integration, Shenzhen University, Shenzhen, 518060, Guangdong, China

ARTICLE INFO

ABSTRACT

Background subtraction (BGS) is utilized to detect moving objects in a video and is commonly employed at the onset of object tracking and human recognition processes. Nevertheless, existing BGS techniques utilizing deep learning still encounter challenges with various background noises in videos, including variations in lighting, shifts in camera angles, and disturbances like air turbulence or swaying trees. To address this problem, we design a spiking autoencoder network, termed SAEN-BGS, based on noise resilience and time-sequence sensitivity of spiking neural networks (SNNs) to enhance the separation of foreground and background. To eliminate unnecessary background noise and preserve the important foreground elements, we begin by creating the continuous spiking conv-and-dconv block, which serves as the fundamental building block for the decoder in SAEN-BGS. Moreover, in striving for enhanced energy efficiency, we introduce a novel self-distillation spiking supervised learning method grounded in ANN-to-SNN frameworks, resulting in decreased power consumption. In extensive experiments conducted on CDnet-2014 and DAVIS-2016 datasets, our approach demonstrates superior segmentation performance relative to other baseline methods, even when challenged by complex scenarios with dynamic backgrounds.

1. Introduction

Background subtraction (BGS) is used to differentiate between moving objects in the foreground and the background in videos captured from real-life settings [1–6]. It has a wide variety of uses, such as urban traffic detection [7], long-term video monitoring [8], optical motion capture [9], and social signal processing [10].

Traditional BGS methods concentrate on improving performance in two particular domains. One potential approach is to develop feature representations that possess increased resilience, such as texture features [11], color features [12] and edge features [13]. The second option aims to generate background models with greater accuracy, such as Kernel density estimate (KDE) [14], mixture of Gaussians (MoG) [15], subspace learning models [16], and principal component analysis (PCA) [16]. Nevertheless, they, referred to as pixel-level BGS approaches, rely exclusively on manually created features to classify individual pixels as either foreground or background, and are highly sensitive to changes in the environment, such as differences in lighting and weather conditions.

Several BGS methods that rely on data-driven deep learning techniques demonstrate improved performance by incorporating convolutional neural networks (CNNs). In general, they can be classified as unsupervised BGS and supervised BGS. While many unsupervised BGS methods, such as SemanticBGS [17], enhance their performance by integrating semantic segmentation models with traditional BGS algorithms. However, they are erroneously perceived as the foremost subjects in dim lighting or shadows, regardless of alterations in the backdrop. In supervised BGS methods like those outlined in [18,19], they typically entail training on the first sequence of frames in test videos, resulting in acceptable outcomes yet unresolved of the constraints of unsupervised BGS methods. To address that, various methods utilizing an autoencoder (AE) for nonlinear dimensionality reduction, such as those mentioned in [20–24], have been investigated for reconstructing backgrounds to enhance the segmentation of foregrounds. During this process, they primarily depend on extracting a low-dimensional feature with minimal noise in the encoder, which is subsequently utilized to accomplish the downstream task of background reconstruction through supervised or unsupervised methods in the decoder. Although they can achieve superior segmentation on static backgrounds, there is still progress to be made for dynamic backgrounds. Intuitively, these AE-based BGS methods may inadvertently overlook the necessity of being robust against background noise closely related to the temporal

---

* Corresponding author.
E-mail addresses: zhangzhixuan77@gmail.com (Z. Zhang), x.p.li@szu.edu.cn (X.P. Li), drliuqi@scut.edu.cn (Q. Liu).
[1] Senior Member, IEEE.

information between frames. Conversely, the decrease in power usage by the trained autoencoder during the inference phase will become increasingly important as the size and complexity of the autoencoder used in background subtraction grow. Practically, an advanced BGS technology utilizing neural networks is expected to provide exceptional performance and energy efficiency during its inference stage.

In this paper, we introduce a spiking autoencoder network for background subtraction (SAEN-BGS) to address the issues mentioned above, inspired by the natural noise resistance and sensitivity to temporal sequences of SNNs. To enhance the prominence of the foreground elements and reduce background interference effectively, we develop a continuous spiking conv-and-dconv block as a fundamental building block for constructing the decoder in SAEN-BGS. Afterward, to implement an energy-efficient inference, we further present a self-distillation spiking supervised learning algorithm based on ANN-to-SNN frameworks. Finally, extensive experiments on CDnet-2014 and DAVIS-2016 datasets demonstrate the superiority of the proposed method over baselines.

Our main contributions are summarized as follows:

- To address the background noise stemming from inadequate temporal information extraction, a spiking autoencoder network (SAEN-BGS) is developed using the noise resilience and time-sequence sensitivity of spiking neural networks.
- This is the first instance of solving background subtraction from a spike-based perspective, where a continuous spiking convolutional and deconvolutional (conv-and-dconv) block is employed to enhance foreground features and diminish background noise within the decoder.
- To achieve energy efficiency, a novel self-distillation spiking supervised learning method is proposed within the ANN-to-SNN framework.
- The empirical evaluations on CDnet-2014 and DAVIS-2016 demonstrate the superiority of the proposed method. Remarkably, even in challenging dynamic backgrounds, our method still outperforms other baselines.

The remainder of the paper is organized as follows. We review some related works in Section 2. In Section 3, the proposed neural networks are introduced. In Section 4, experimental results on real-world datasets demonstrate that the suggested algorithms outperform the SOTA methods. Finally, conclusions are drawn in Section 5.

2. Related work

2.1. Background subtraction method

BGS approaches, which aim to distinguish between foreground and background in videos, can be broadly classified as unsupervised or supervised techniques. Unsupervised BGS methods focus on creating a background model to differentiate between foreground and background. For instance, Stauffer et al. [25] propose an online Gaussian Mixture Model (GMM) to update parameters pixel-wise and use multiple Gaussian distributions to represent pixel color distributions. Elgammal et al. [26] improve upon this by utilizing Kernel Density Estimation (KDE), a non-parametric method, to describe pixel intensity distributions. On the other hand, Barnich et al. [27] present a model that utilizes distances to classify pixels through the capture of local spatial features to establish the background model. St-Charles et al. [28] create PAWCS, a word-based approach that combines color and texture characteristics by treating pixels as background words and continually adjusting their trustworthiness according to persistence. Then Braham et al. [17] design a post-processing procedure for the background subtraction algorithm using semantic segmentation prediction. SuBSENSE [29] improves detection accuracy through the inclusion of color features and pixel-level feedback. More recently, Isik

et al. [30] introduce SWCD, a pixel-wise sliding-window technique that updates the background model with a dynamic control system, while Lee et al. [31] devise a multi-step algorithm to decrease false positives in dynamic backgrounds. What is more, An et al. [32] develop an unsupervised background subtraction approach which constructs open-vocabulary instance-level background models through zero-shot object detection, and then identifies foreground by comparing new frame detections with these models.

Conversely, supervised BGS algorithms adjust neural network parameters via reducing the differences between labels and training frames using loss function optimization. Braham et al. [18] pioneer the use of CNNs in BGS with ConvNets, achieving impressive results on the CDnet-2014 dataset, despite employing a costly patch-based training approach. Wang et al. [19] propose a multiscale CNN with a cascading structure, which analyzes individual frames without background frames but requires 200 test video frames for training. Similarly, Babaee et al. [33] develop a video-group-optimized CNN for processing all videos. Sakkos et al. [34] introduce a 3D CNN designed to capture both temporal and color information. However, these techniques rely on utilizing partial frames from test videos during training, which could limit their effectiveness with previously unseen videos.

2.2. Autoencoder-based BGS method

Recent research has integrated nonlinear dimensionality reduction techniques using AE to improve the differentiation between foreground and background components, enabling the extraction of enhanced feature data through an iterative optimization approach. Lim et al. [20] propose an AE for background subtraction, where the encoder is responsible for extracting feature vectors, and the decoder converts these vectors into segmentation maps by utilizing the current frame, a previous frame, and a background model as input. In [21], Mandal et al. introduce 3DFR, which involves three feature reduction networks and utilizes 50 past frames to aid in the current frame's learning process. In addition, Lim et al. design two robust AEs, known as FgSegNet_M and FgSegNet_S [22], which produce multi-scale features in the encoder section. Note that these techniques necessitate a labeled frame from the test video for the purpose of training. Then Tezcan et al. propose a CNN-based on UNET architecture called BSUV-Net [23], which takes the current frame and two background frames as input. At the same time, BSUV-Net 2.0 [24] and Fast BSUV-Net 2.0 [24] are introduced and achieve similar levels of effectiveness.

2.3. Low-power computational framework

Recently, as the parameters and operators of neural networks have grown, the research community has become increasingly interested in the power consumption of neural networks. As shown in Fig. 1, SNNs offer a natural solution because they have a similar way of transmitting information as the brain, using spike neurons like the leaky integrate-and-fire (LIF) neurons instead of traditional artificial neurons, such as ReLU-activated neurons. Brain-like LIF model corresponds to its physical circuit. The soma constantly receives integrated input from dendrites. When the sum of total input exceeds a certain threshold ϑ, an output spike is generated and then delivered to other neurons by the axon. After firing a spike, the membrane potential is reset to V_{rest}. SNNs have the advantage of lower power consumption during the inference stage. Besides, SNNs have successfully permeated into a myriad of application domains such as image and speech recognition, object detection, autonomous driving and other intelligence-related real-world applications [35–37]. Currently, there are three types of spiking learning frameworks in terms of implementation: (1) due to the discreteness of spikes and the intrinsic non-differentiability of spike firing function impeding the direct applicability of the traditional back-propagation (BP), surrogate-based BP algorithm thus can be derived by replacing the spiking function with differential functions or adding
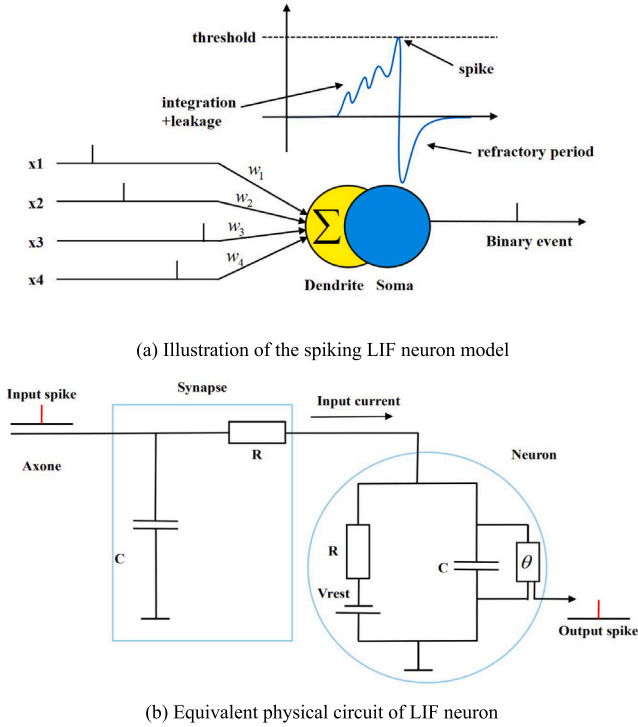
(a) Illustration of the spiking LIF neuron model

(b) Equivalent physical circuit of LIF neuron

Fig. 1. Brain-like leaky integrate-and-fire (LIF) model corresponds to its physical circuit. The soma constantly receives integrated input from dendrites. When the sum of total input exceeds a certain threshold ϑ, an output spike is generated and then delivered to other neurons by the axon. After firing a spike, the membrane potential is reset to V_{rest}.

some limitation or clipping to the BP process (e.g., [38]). (2) Based on spike-timing-dependent-plasticity (STDP) algorithm, SNN models learn each module separately (e.g., [39]). (3) Different from the above SNN algorithms, the ANN-to-SNN algorithms assume that SNNs have equal efficiency as the counterpart traditional neural networks such as (e.g., [40]).

3. Methodology

3.1. Problem formulation

BGS serves as a fundamental research topic in video processing and has been widely investigated nowadays. Given a sequence X of consecutive frames $X_1, \ldots, X_N$ with a scene cluttered by various moving objects, such as cars or pedestrians, and the expected output is a sequence $\hat{X}$ of frames $\hat{X}_1, \ldots, \hat{X}_N$ showing the backgrounds of each scene without those objects. Typically, supervised BGS methods require the assistance of the given label/mask M. Despite the current BGS methods have made much progress on separation performance, it is inevitable that they still get stuck in background noises, such as lighting variation, camera moving and weather changes. Thus, they are erroneously perceived as the foremost subjects in dim lighting or shadows, regardless of alterations in the backdrop. This is verified by the experimental result in Fig. 2, where the continuous frames from $t = 902$ to $t = 905$ are randomly chosen. As compared with ZBS [32] and BSUV-Net 2.0 [24], snow and the car in the background are erroneously perceived as the foremost subjects, respectively. In the subsequent section, we introduce a novel SNN model aimed at enhancing separation performance. This model incorporates a continuous spiking conv-and-dconv block along with a unique self-distillation spiking learning algorithm.

Given a SNN S used for BGS, its internal process is spike-event driven. First, the input frame X_i needs to be encoded into spike trains
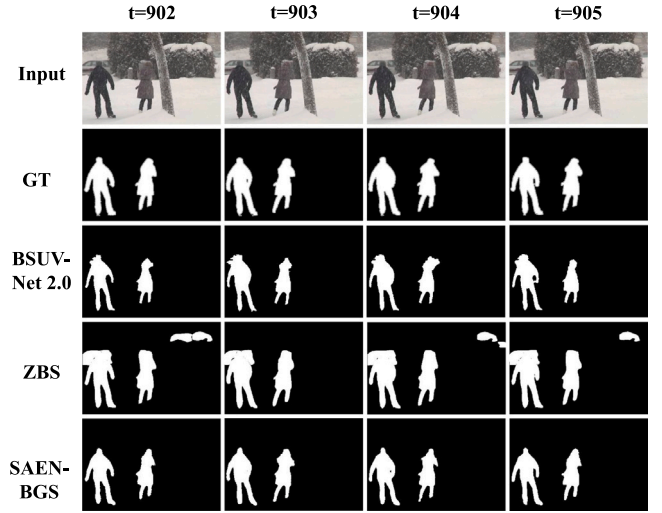

Fig. 2. Separation performance of our method and two state-of-the-art baselines over five continuous frames in the badweather video with high background noise.

$s_i = s_i^1, \ldots, s_i^T$ by rate coding [38], where T indicates the length of the encoding time window. Then the encoded spike trains s_i pass over S, and the outputs of intermediate layers are composed of spike trains and spike counts. Finally, the expected output mask can be described as $\hat{M} = S(X_i)$. Besides, S is inherently sensitive to temporal information, beneficial to alleviating the issues of current BGS methods.

3.2. SAEN-BGS

As mentioned above, we engage to solve the existing problem that current BGS methods tend to be influenced by background noise, resulting from the insufficient extraction of temporal information by our above analysis. Additionally, the increasing number of parameters and operations in neural networks poses a potential challenge. Fortunately, SNNs have inherent sensitivity to temporal information and are friendly to low-energy computation attributed to the spike-based mechanism. Therefore, to address these issues, we apply the spike-based computation to devise a SAEN-BGS model for BGS. In detail, from the perspectives of model structure and learning strategy, we design a continuous spiking conv-and-dconv block and propose a self-distillation spiking learning algorithm, respectively. As shown in Fig. 3, the proposed model is composed of an encoding layer, a spiking encoder, a spiking decoder and a decoding layer. CNN-based conv module and CNN-based dconv module, respectively, represent a convolutional layer and a deconvolutional layer to transmit real values rather than spikes. Details of the proposed SAEN-BGS model are described below.

3.2.1. Neuron model

Our model comprises two types of neurons: ReLU-activated neurons and integrate-and-fire (IF) neurons. ReLU-activated neurons, commonly used in traditional neural networks, clip activations below zero. IF neurons, inspired by biological spike-based information transmission, are utilized in SNNs, which replace LIF neurons for simplicity. Synaptic transmission between presynaptic and postsynaptic neurons is driven by the positive correlation between presynaptic spike timing and membrane potentials. In a simulation time window N_t, the input spikes to neuron j in layer l are integrated into subthreshold membrane potential U_j^l at each time step t. Thus, the membrane potential of neuron j in layer l is modeled as [40]:

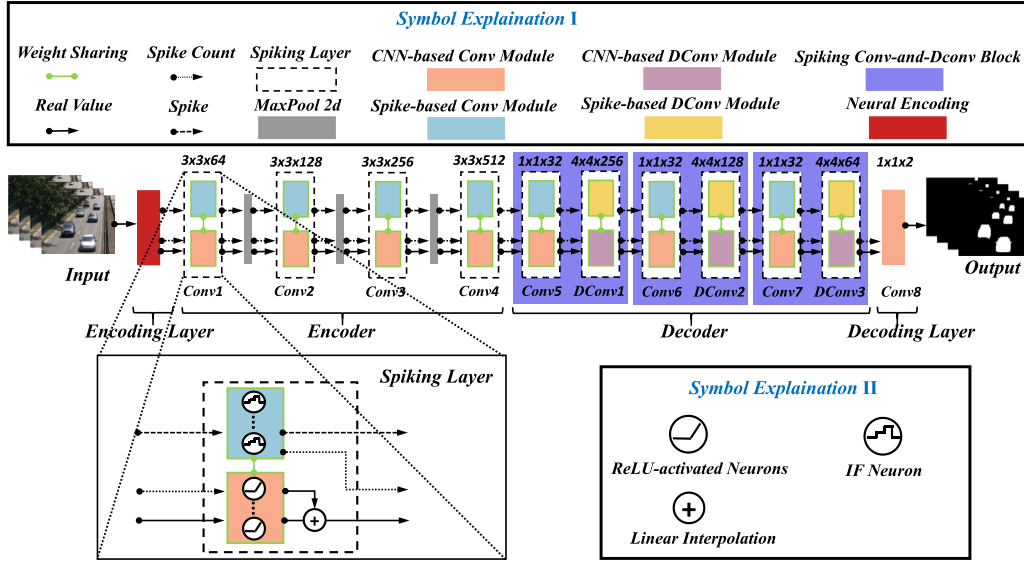$$U_j^l[t] = U_j^l[t-1] + RI_j^l[t] - \vartheta S_j^l[t-1], \tag{1}$$

Fig. 3. Architecture of the proposed SAEN-BGS. Each spiking layer integrates a CNN-based module and a spike-based module with shared weights. The CNN-based module facilitates the spike-based module training but is inactive during inference. The spike-based module processes each frame in 10 time steps.

in which $I_j^l[t] = \sum_i w_{ji}^{l-1} S_i^{l-1}[t] + b_j^l$ and $S_j^l[t] = \delta(U_j^l[t] - \vartheta)$ with the indicator function

$$\delta(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The synaptic weight w_{ji}^{l-1} affects connection from presynaptic neuron i in layer $l-1$ and b_j^l is a constant injecting current. Moreover, $S_j^l[t-1]$ signifies the occurrence of a spike emitted by the neuron j at the previous time step $t-1$, and $I_j^l[t]$ represents the synaptic current generated by the integration of incoming spike trains t. At time t, the membrane potential $U_j^l(t)$ exceeds the predefined threshold ϑ (commonly set to $\vartheta = 1$), leading to the initiation of an action potential, also known as a spike. That is:

$$U_j^l(t) \geq \vartheta \ , \ \frac{dU_j^l(t)}{dt} > 0. \tag{3}$$

Following the generation of a spike, the membrane potential $U_j^l(t)$ is reset to the resting potential U_{rest} and remains in a refractory state for a specific duration. Synaptic weight, or synapse conductance, changes according to presynaptic and postsynaptic neuronal activities. This variability, driven by synaptic plasticity, enables neuronal learning. Encoding methods are categorized into spike-rate-based (counting spikes over time) and spike-time-based (focusing on precise spike timing). Our encoding method is detailed in the next subsection.

3.2.2. Encoding and decoding layers

The encoding layer and the decoding layer serve as the input and output modules, respectively, in our proposed model. Time-varying input currents, treated as real-valued inputs, are directly incorporated into Eq. (1) at each time step. This neural encoding method eliminates the sampling errors associated with rate coding, enabling precise and efficient inference [38]. It can be denoted as $s = \phi(x)$, where s and x indicate the produced spike trains and the real-value input, respectively. As illustrated in Fig. 3, the spiking layer processes both spike trains and spike counts as inputs.

To facilitate pattern classification, the output spike trains from the SNNs must be decoded into distinct pattern classes. This decoding can be accomplished at the SNNs' output layer using either discrete spike counts or continuous aggregated membrane potentials (without spiking). Utilizing aggregated membrane potentials provides a smoother learning process, as it allows for the computation of continuous error gradients at the output layer [40]. Therefore, a convolutional layer as the decoding layer is employed.

3.2.3. Spiking conv-and-dconv block

The spiking conv-and-dconv block is designed for information compression and deconvolution for feature reconstruction, which has proven effective in various generative tasks [41]. Specifically, we propose that applying deconvolution (dconv) after convolution (conv) filtering can help reduce background noise. The spiking conv-and-dconv block primarily consists of a spiking conv layer and a spiking dconv layer. In SAEN-BGS, the decoder is constructed using three such blocks. Each spiking layer integrates a CNN-based module for parameter training and a spike-based module for inference (testing or validation), with shared parameters, e.g., convolution kernels. In this block, the kernel size of the spiking conv layer is set to $1 \times 1 \times C_{out}$, where C_{out} denotes the number of output channels. Since the output channel number from the preceding spiking layer exceeds that of the current spiking conv layer, the block effectively compresses channel information, which may aid in suppressing background noise. As illustrated in Fig. 4, the results indicate that the background noise in the outputs of the proposed conv-and-dconv spiking blocks is significantly reduced compared to that in the front encoder, thus confirming the effectiveness of our designed block.

3.3. Self-distillation spiking learning algorithm

Our proposed algorithm aims to provide energy-efficient learning for our SAEN-BGS, inspired by the ANN-to-SNN framework in [40], where the CNN-based module serves as the teacher overseeing the SNN-based counterpart in its learning process, assuming that the spike counts from the spike-based module can be accurately estimated by the actual values from the CNN-based module. However, approximation loss exists along with performance degradation, especially in noisy circumstances. In addition, it is not advisable to make sweeping generalizations to new data sourced from the identical distribution. To address that, as shown in Fig. 5, a learning pathway is built to independently pass over every CNN-based module via linear interpolation $\eta x^l + (1 - \eta)a^l$ of real-valued data x^l and approximated spike count a^l, where the hyperparameter η is set 0.5 as default. This is motivated by the multi-task learning [42] and the self-distillation learning mechanism [43]. Concretely, this design aims to further boost the robustness of spike-based modules to noise by using the approximated spike count from the CNN-based module, and reduce the bias of the CNN-based module in the learning process by the new learning pathway over every CNN-based module. Since it can be seen as increasing new auxiliary
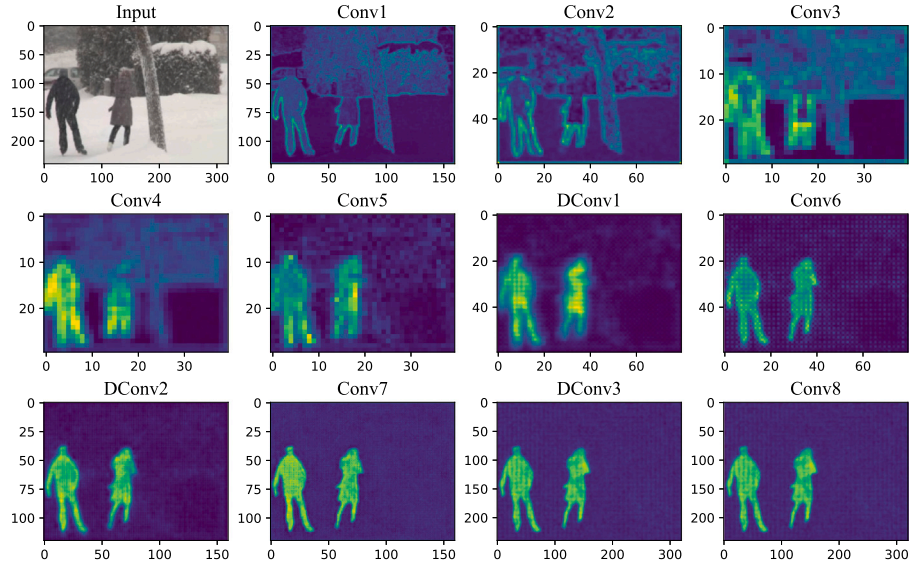
Fig. 4. Outputs of intermediate layers of SAEN-BGS model over a randomly selected frame.
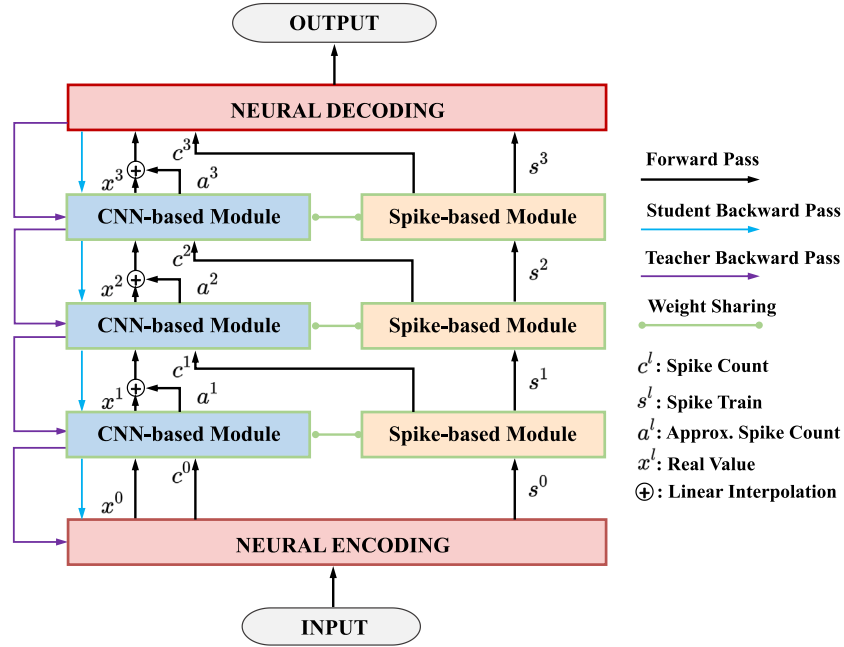

Fig. 5. Frameworks of our proposed self-distillation spiking learning algorithm.

sub-networks within the network itself, the learning process can be termed as self-distillation in multi-task views, so-called self-distillation spiking learning algorithm. The main task is to train the spike-based module by sharing weights from the CNN-based module, while the related task is to train the CNN-based module itself. As shown in Fig. 6, we illustrate the heatmap of activation in all spiking layers, which presents the sparsity of activation on spike neurons directly. More experimental results are presented at Section 4.

For the BGS problem, our goal is to minimize the negative log-likelihood loss between the ground-truth mask $\boldsymbol{M}$ and the estimated outputs $\hat{\boldsymbol{M}}$. Given frames $X_1, \ldots, X_N$ with labels $M_1, \ldots, M_N$, the SAEN-BGS outputs are denoted as $\hat{M}_1, \ldots, \hat{M}_N$. The loss function $\mathcal{L}$ can be formulated as:

$$\mathcal{L}(\hat{\boldsymbol{M}}, \boldsymbol{M}) = -\frac{1}{N} \sum_{i=1}^{N} M_i \log \hat{M}_i. \tag{4}$$

As mentioned above, to achieve the multi-task learning, the overall loss function $\mathcal{L}_{all}$ is applied, where $\hat{\boldsymbol{M}}$ and $\hat{\boldsymbol{M}}^r$, represent the outputs of main task (along the pathway of s^i and c^i) and related task (along the pathway of x^i), respectively. The hyperparameter α is equal to 0.8:

$$\mathcal{L}_{main} = \mathcal{L}(\hat{\boldsymbol{M}}, \boldsymbol{M}), \tag{5}$$

$$\mathcal{L}_{aux} = \mathcal{L}(\hat{\boldsymbol{M}}^r, \boldsymbol{M}), \tag{6}$$

$$\mathcal{L}_{all} = \alpha \mathcal{L}_{main} + (1 - \alpha) \mathcal{L}_{aux}. \tag{7}$$

As for the non-differentiable nature of spike generation, the error backpropagation (BP) method cannot be directly applicable to the training process. To that end, we apply the ANN-to-SNN rule by spike count approximation. Next, the spike count from neuron i at layer l is
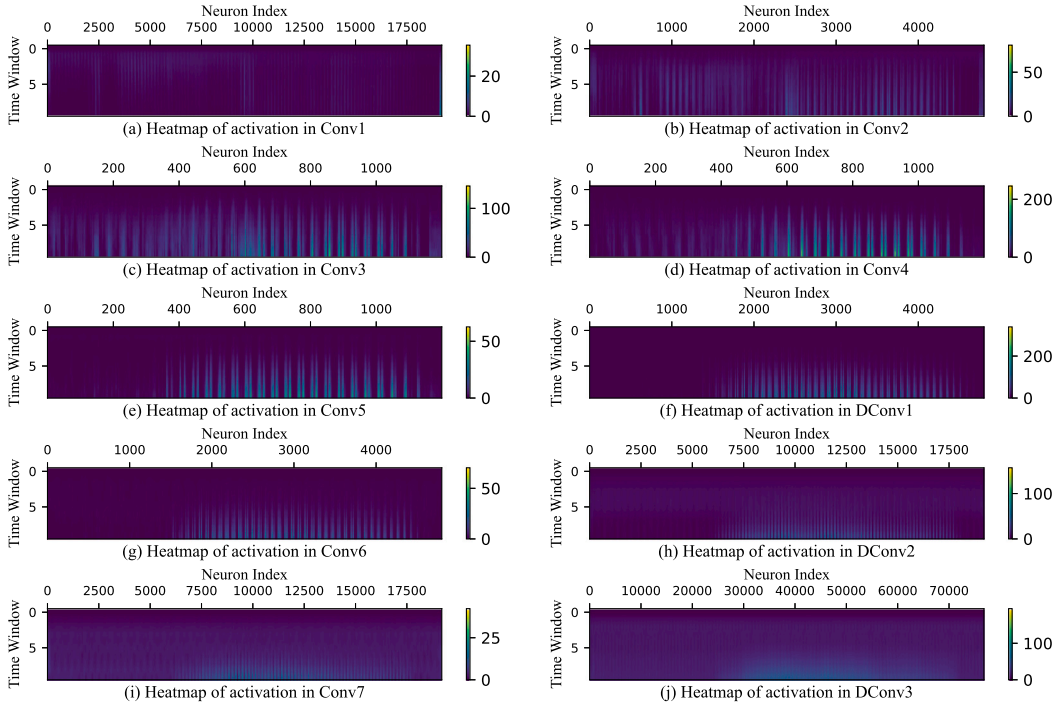
Fig. 6. Illustration of activation heatmap of spiking layer in our SAEN-BGS over an input.

formulated as:

$$c_i^l = \sum_{t=1}^{N_t} S_i^l[t]. \tag{8}$$

The relationship between ANN activation functions and SNN spike counts is established through shared weights, modeled mathematically as:

$$\Delta t = \rho\left(\frac{\vartheta}{U_j^l/N_t}\right), \tag{9}$$

in which the aggregated action potential of neuron j in layer l is written as $U_j^l = \sum_i w_{ji}^{l-1} c_i^{l-1} + b_j^l N_t$. ϑ, ρ and Δt denote the firing threshold, the non-linear transformation of ReLU-activated neurons, and the time step, respectively. Hence, the approximated spike count a_j^l is obtained as:

$$a_j^l = \frac{N_t}{\Delta t} = \frac{1}{\vartheta}\rho\left(\sum_i w_{ji}^{l-1} c_i^{l-1} + b_j^l N_t\right). \tag{10}$$

Thus, a_j^l is determined by the ReLU activation in the CNN-based module, using the spike count c_i^{l-1} as input and $b_j^l N_t$ as the bias term. This simplification enables error gradient estimation at the spike-train level using CNN-based modules, following the tandem learning rule with firing rates instead of spike timings [40].

The overall process of self-distillation spiking learning is shown in Algorithm 1. In a training epoch, training frames X as input to SAEN-BGS F is first encoded into spike trains s^0 and spike counts c^0 at encoder layer of F obeyed by Eq. (1). It notes that the original X is also copied as one of the outputs of the encoding layer, namely x^0, to pass the CNN-based module of the first spiking layer. Then the resulting ten spiking layers all receive the spike trains $s^{\ell-1}$, spike counts $c^{\ell-1}$, real-value output $x^{\ell-1}$ and approximated spike count $a^{\ell-1}$ from the previous blocks and output the new spike trains s^ℓ, spike counts c^ℓ, real-value output x^ℓ and approximated spike count a^ℓ. Besides, the decoding layer processes the spike counts and real-value output from the last spiking layer in the decoder and outputs the mask predictions $\hat{M}$, $\hat{M}^r$. Finally, we construct the loss function using Eq. (7) and update the parameters of SAEN-BGS by SGD.

4. Experiments

4.1. Experimental setup

We present the experimental setup for the CDnet-2014 [44] and DAVIS-2016 [45] datasets. The proposed neural networks are implemented using the PyTorch framework [46], with the negative log-likelihood loss as the objective function and RMSprop as the optimizer. It should be noted that in addition to our SAEN-BGS, we also display the performance of the counterpart CNNs (AEN-BGS) to demonstrate the energy efficiency of our proposed self-distillation spiking learning constantly.

4.1.1. Baselines

To verify the segmentation performance and efficient energy consumption of our proposed SAEN-BGS, we compare it with the counterpart AEN-BGS, and several state-of-the-art methods, including ZBS [32], CwisarDH [47], Spectral-360 [48], FTSG [49], SuBSENSE [50], DeepBS [33], SemanticBGS [17], IUTIS-5 [51], BSUV-Net [23], IUTIS-5+ SemanticBGS [23], BSUV-Net 2.0 [24], WisenetMD [31], RTSS [52], SWCD [30] and PAWCS [28].

4.1.2. Training details

We evaluate our SAEN-BGS and other baselines on CDnet-2014 and DAVIS-2016 datasets. The training details for our SAEN-BGS and its counterpart AEN-BGS, are as follows. It should be noted that except for AEN-BGS, other baselines keep their original setting.

The CDnet-2014 dataset comprises 53 videos across 11 categories, all used to evaluate our proposed SAEN-BGS. In consistency with [24], we also follow the two training schemes employed: a small training set with 200 randomly selected frames from each video and a large training set with randomly selected 70% of the frames from each video. Additionally, since the video frame dimensions vary from 240×240 to 526×720, all frames and ground truth are resized to 240×320 for simplicity. The batch size is set to 8, with training over 100 epochs. As for the learning rate (LR), our SAEN-BGS sets 0.0005 to the night videos and camera jitter categories except for other categories using an initial LR of 0.0002. For its counterpart AEN-BGS, LR is set to 0.001, except for

---

Algorithm 1: Self-Distillation Spiking Learning

Require: Training frames X; Ground truth masks M; Total epoch E; SAEN-BGS $\mathbf{F}$ = {Neural Encoding ϕ, Encoder $\mathbf{F}_{en}$ = (Conv1, Conv2, Conv3, Conv4), Decoder $\mathbf{F}_{de}$ = (⟨Conv5, DConv1⟩, ⟨Conv6, DConv2⟩, ⟨Conv7, DConv3⟩), Decoding Layer Conv8} with parameter $\theta_{\mathbf{F}}$, ⟨·⟩ denotes Spiking Conv-and-DConv Block; Learning rate γ; Hyperparameter α.
Ensure: Model parameters $\theta_{\mathbf{F}}^*$.

   for $e \leftarrow 1$ **to E do**
      for $i \leftarrow 1$ **to** $len(X)$ **do**
1   // Extract training data
2         $x \leftarrow X[i]$, $y \leftarrow M[i]$
3   // In encoding layer
4         $s^0, c^0, x^0, a^0 \leftarrow \phi(x), x, x, 0$ (as in Eq. (1))
5   // In spiking layers
6   **for** $\ell \leftarrow 1$ **to** 10 **do**
7            $x'^{l-1} \leftarrow \eta x^{l-1} + (1-\eta)a^{l-1}$
8            $s^\ell, c^\ell, x^\ell, a^\ell \leftarrow \mathbf{F}[\ell](s^{\ell-1}, c^{\ell-1}, x'^{\ell-1})$
9   **end**
10   // In decoding layer
11        $\hat{y}, \hat{y}^r \leftarrow \text{Conv8}(c^{10}, \eta x^{10} + (1-\eta)a^{10})$
12   // Construct loss functions
13        $\mathcal{L}_{main} \leftarrow$ Eq. (5) with input of $\hat{y}$ and y
14        $\mathcal{L}_{aux} \leftarrow$ Eq. (6) with input of $\hat{y}^r$ and y
15        $\mathcal{L}_{all} \leftarrow \alpha\mathcal{L}_{main} + (1-\alpha)\mathcal{L}_{aux}$ (as in Eq. (7))
16   // Update parameters of SAEN-BGS
17        $\theta_{\mathbf{F}} \leftarrow \theta_{\mathbf{F}} - \gamma\nabla_{\theta_{\mathbf{F}}}\mathcal{L}_{all}$
18        $\theta_{\mathbf{F}}^* \leftarrow \theta_{\mathbf{F}}$
      end
   end

---

the night videos and PTZ categories under the large training set with an initial LR of 0.0002. Regarding LR adapter techniques, our SAEN-BGS uses the MultiStepLR scheduler for the intermittent object motion category and the ReduceLROnPlateau scheduler for night videos in the small training set and the PTZ category, while the StepLR scheduler is used for all other videos. Then the counterpart AEN-BGS utilizes the ReduceLROnPlateau scheduler as an early stopping strategy, except for night videos in the large training set and the PTZ category in the small training set applying the StepLR scheduler. Additionally, our SAEN-BGS only spends a time window of 10 ms with a time step of 1 ms.

Over the DAVIS-2016 dataset, there exist almost no BGS methods evaluating. Thus, we train on the large training set scheme with randomly selected 70% of the frames from each video, aimed to validate the overall performance. All frames and ground truths are resized to 240×320. The batch size is set to 8, over 200 training epochs and LR is set to 0.001 for AEN-BGS and 0.0002 for SAEN-BGS. For LR adapter, AEN-BGS uses the StepLRMulti scheduler, while SAEN-BGS employs the StepLR scheduler. The time window length (10 ms) and time step (1 ms) remain consistent with those used for the CDnet-2014 dataset.

4.1.3. Basic evaluation metrics

To assess the performance of BGS, seven evaluation metrics are employed: False Positive Rate (FPR), False Negative Rate (FNR), Recall (Rec), Precision (Pre), Specificity (Spe), Percentage of Wrong Classifications (PWC), and F-measure (Fm). These metrics are derived from the definitions of true positive (TP), false positive (FP), false negative (FN), and true negative (TN). Specifically, TP and FP represent pixels correctly and incorrectly classified as foreground, respectively, while TN and FN denote pixels correctly and incorrectly classified as background. The seven metrics are defined as follows:

$$\text{FPR} = \frac{FP}{FP+TN}, \quad \text{FNR} = \frac{FN}{TP+FN},$$

$$\text{Rec} = \frac{TP}{TP+FN}, \quad \text{Pre} = \frac{TP}{TP+FP},$$

$$\text{Spe} = \frac{TN}{TN+FP}, \quad \text{PWC} = \frac{(FN+FP)}{TP+FN+FP+TN},$$

$$\text{Fm} = \frac{2 \times \text{Pre} \times \text{Rec}}{\text{Pre} + \text{Rec}}. \tag{11}$$

Clearly, lower FPR, FNR, and PWC values indicate better performance, while higher Rec, Pre, Spe, and Fm reflect superior results. Except for PWC, the units of all other metrics are %.

4.1.4. Energy evaluation metrics

To verify the energy efficiency of our SAEN-BGS learned by our proposed self-distillation spiking learning algorithm, we compute the energy used in model inference according to [54], which is a common way to compare the energy consumption in SNNs. The computational cost is measured by the total number of Floating-Point Operations (FLOPs), which corresponds to matrix–vector multiplications and scales proportionally. Note that energy calculations focus on the SNN model during the test phase, where CNN-based modules are inactive. For each layer l_i, the FLOPs of the CNN-based module are given by:

$$FLOPs_{ANN\_Module}(l_i) =$$

$$\begin{cases} k^2 \times O^2 \times C_{in} \times C_{out}, & \text{if } l_i \text{ denotes the convolutional layer,} \\ C_{in} \times C_{out}, & \text{if } l_i \text{ denotes the linear layer,} \end{cases} \tag{12}$$

where k and O denote the kernel size and output feature map size, respectively, while C_{in} and C_{out} represent the channel numbers of the input and output. To compute the FLOPs of the spike-based module for each spiking layer l_i, the spiking rate $R_s(l_i)$, as SNN consumes energy only during spike firing, can be defined as:

$$R_s(l_i) = \frac{\text{\#spikes per layer } l_i \text{ over all time steps}}{\text{\#neurons per layer } l_i}, \tag{13}$$

i.e., the average firing rate per neuron. For the whole SNN, the average spike rate (AvR$_s$) is equal to $\frac{1}{11}\sum_{l=1}^{11} R_s(l)$. Therefore, FLOPs for the spike-based module per spiking layer are calculated as:

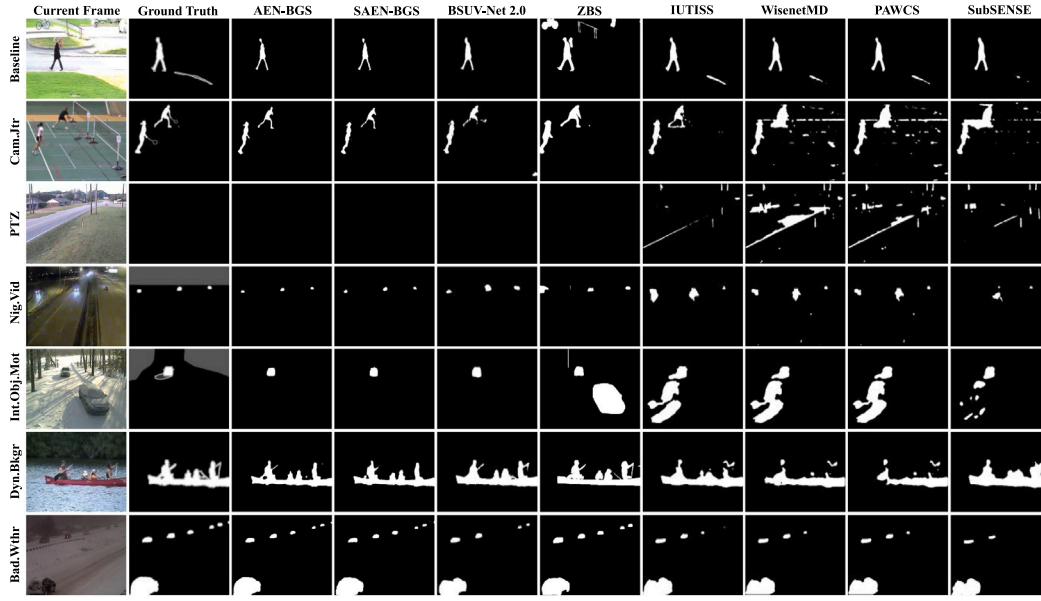$$FLOPs_{Spike\_Module}(l_i) = FLOPs_{ANN\_Module}(l_i) \times R_s(l_i). \tag{14}$$

Fig. 7. Visual comparison of top BGS algorithms on sample frames from different categories of CDnet-2014.

Table 1
Evaluation results on CDnet 2014 dataset under the small training set (randomly selected 200 frames per video) among different compared models using several evaluation metrics. **Blue** indicates the best. **Red** indicates the second.

Model	Rec (↑)	Spe (↑)	FPR (↓)	FNR (↓)	PWC (↓)	Fm (↑)	Pre (↑)
AEN-BGS	91.97	99.84	0.12	8.01	0.296	92.34	92.74
SAEN-BGS	89.59	99.78	0.20	10.38	0.388	90.12	90.71
CwisarDH [47]	66.08	99.48	0.52	33.92	1.527	68.12	77.25
Spectral-360 [48]	73.45	98.61	1.39	26.55	2.272	67.32	70.54
FTSG [49]	76.57	99.22	0.78	23.43	1.376	72.83	76.96
SuBSENSE [50]	80.70	98.84	1.16	19.30	1.842	73.31	74.63
CascadeCNN [53]	95.06	99.68	0.32	4.94	0.405	90.09	89.97
DeepBS [33]	75.45	99.05	0.95	24.55	1.992	74.58	83.32
SemanticBGS [17]	78.90	99.61	0.39	21.10	1.072	83.05	78.92
IUTIS-5 [51]	78.49	99.48	0.52	21.51	1.199	80.87	77.17

Table 2
Evaluation results on CDnet 2014 dataset under the large training set (randomly selected 70% frames per one video) among different compared models using several evaluation metrics. **Blue** indicates the best. **Red** indicates the second.

Model	Rec (↑)	Spe (↑)	FPR (↓)	FNR (↓)	PWC (↓)	Fm (↑)	Pre (↑)
AEN-BGS	94.69	99.91	0.07	5.29	0.162	94.60	94.56
SAEN-BGS	92.92	99.88	0.11	7.05	0.229	92.82	92.74
ZBS [32]	84.03	99.81	0.19	15.97	0.563	85.15	88.02
IUTIS-5 [51]	78.49	99.48	0.52	21.51	1.199	80.87	77.17
BSUV-Net [23]	82.03	99.46	0.54	17.97	1.140	78.68	81.13
IUTIS-5+SemanticBGS [23]	78.90	99.61	0.39	21.10	1.072	78.92	83.05
BSUV-Net+SemanticBGS [23]	81.79	99.44	0.56	18.21	1.133	79.86	83.19
BSUV-Net 2.0 [24]	81.36	99.79	0.21	18.64	0.761	83.87	90.11
Fast BSUV-Net 2.0 [24]	81.81	99.56	0.44	18.19	0.905	80.39	84.25
WisenetMD [31]	81.79	99.04	0.96	18.21	1.614	76.68	75.35

Thus, the total inference energy consumption for the CNN-based module (E_{ANN_Module}) and the spike-based module (E_{Spike_Module}) is calculated as:

$$E_{ANN\_Module} = \sum_{l_i} FLOPs_{ANN\_Module}(l_i) \times E_{MAC}, \qquad (15)$$

$$E_{Spike\_Module} = \sum_{l_i} FLOPs_{Spike\_Module}(l_i) \times E_{AC}, \qquad (16)$$

where E_{AC}, E_{MAC} are derived from a standard 45 nm Complementary Metal–Oxide–Semiconductor (CMOS) process, with $E_{MAC} = 4.6pJ$ and $E_{AC} = 0.9pJ$ for 32 bit FP [55]. Here, we also utilize AvR_s as the other metric to represent power consumption.

4.2. Performance comparison

We begin by assessing the performance on the CDnet-2014 dataset in this section. Table 1 displays measurements for various algorithms when utilizing a limited training dataset. The counterpart AEN-BGS excels over the other nine methods in terms of specificity, FPR, PWC, F-measure, and precision, while also ranking second in recall and FNR. It confirms that our network designs are effective. Moreover, the proposed SAEN-BGS achieves similar performance to AEN-BGS, demonstrating that our model can still perform well with a limited training dataset. The results in Table 2 for the large training set show that our SAEN-BGS approach continues to outperform other current methods. Further examination of the F-measure in 11 categories, as
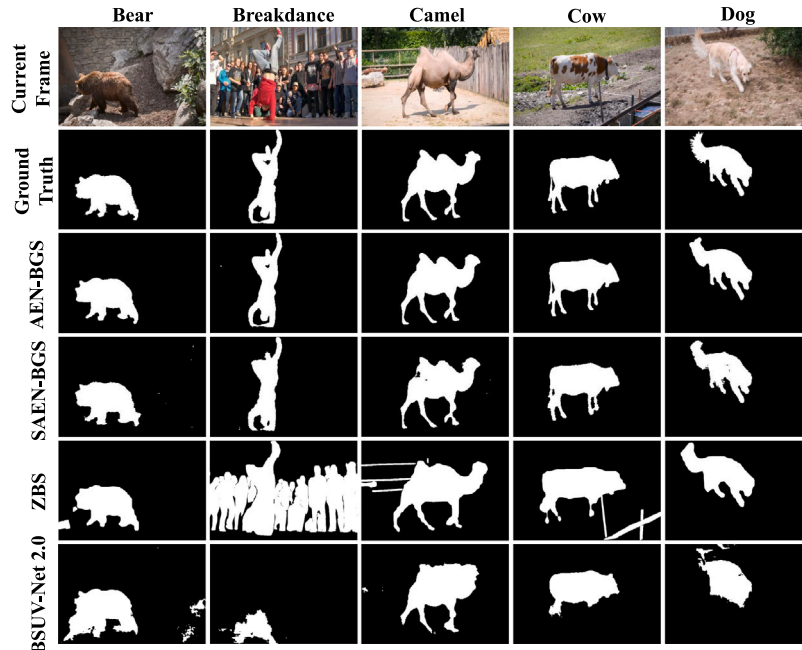
Fig. 8. Visual comparison of top BGS algorithms on sample frames from different categories of DAVIS-2016 dataset.

Table 3
F-measure comparison of different BGS algorithms according to the per-category on CDNet-2014 dataset. Blue indicates the best. Red indicates the second.

Model	Bad. Wthr	Low. Frm. Rate	Nig. Vid	PTZ	Thrml	Shadow	Int. Obj. Mot	Cam. Jtr	Dyn. Bkgr	Base-line	Turb	Overall
AEN-BGS	96.44	94.56	90.75	87.76	97.53	95.83	94.54	96.75	96.28	97.47	92.75	94.60
SAEN-BGS	95.67	93.16	87.67	85.88	95.51	93.83	90.60	96.22	94.96	96.49	91.34	92.85
ZBS [32]	92.29	74.33	68.00	81.33	86.98	97.65	87.58	95.45	92.90	96.53	63.58	85.15
BSUV-Net [23]	87.13	67.97	69.87	62.82	85.81	92.33	74.99	77.43	79.67	96.93	70.51	78.68
BSUV-Net+SemanticBGS [23]	87.30	67.88	68.15	65.62	84.55	96.64	76.01	77.88	81.76	96.40	76.31	79.86
BSUV-Net 2.0 [24]	88.44	79.02	58.57	70.37	89.32	95.62	82.63	90.04	90.57	96.20	81.74	83.87
Fast BSUV-Net 2.0 [24]	89.09	78.24	65.51	50.14	83.79	88.90	90.16	88.28	73.20	96.94	79.98	80.39
IUTIS-5+SemanticBGS [23]	82.60	78.88	50.14	56.73	82.19	94.78	78.78	83.88	94.89	96.04	69.21	78.92
IUTIS-5 [51]	82.48	77.43	52.90	42.82	83.03	90.84	72.96	83.32	89.02	95.67	78.36	77.17
WisenetMD [31]	86.16	64.04	57.01	33.67	81.52	89.84	72.64	82.28	83.76	94.87	83.04	76.68
RTSS [52]	86.62	67.71	52.95	54.89	85.10	95.51	78.64	83.96	93.25	95.97	76.30	79.17
SWCD [30]	82.33	73.74	58.07	45.45	85.81	87.79	70.92	74.11	86.45	92.14	77.35	75.83
PAWCS [28]	81.52	65.88	41.52	46.15	83.24	89.13	77.64	81.37	89.38	93.97	64.50	74.03

Table 4
Results comparison of different approaches on DAVIS-2016 dataset. Blue indicates the best. Red indicates the second.

Model	Fm (↑)
AEN-BGS	**87.40**
SAEN-BGS	85.20
ZBS [32]	69.04
BSUV-Net 2.0 [24]	63.62

Table 5
Comparison of parameter number and energy between the counterpart AEN-BGS, ZBS and BSUV-Net 2.0. BOLD indicates the best.

Model	Param (↓)	Energy (↓)
AEN-BGS	**1.8M**	**4.10e+10**
ZBS [32]	105.4M	8.23e+10
BSUV-Net 2.0 [24]	30.3M	1.92e+11

shown in Table 3, shows that our SAEN-BGS and the AEN-BGS perform the best overall, with some limitations in the shadow and baseline categories. It reflects that our design model structure can benefit to

improving BGS performance. Alternatively, Fig. 7 provides a visual comparison of different approaches by showcasing one frame from each of seven videos spanning various categories. AEN-BGS, SAEN-BGS, and BSUV-Net 2.0 have shown superior performance compared to other baseline models. Additionally, our SAEN-BGS model shows superior visual performance compared to BSUV-Net 2.0 in categories with dynamic backgrounds such as Cam.Jir, Dyn.Bkgr, and Bad.Wthr, while achieving similar results to AEN-BGS.

In order to assess the effectiveness of our SAEN-BGS, we compare it to two other top-performing methods (ZBS [32] and BSUV-Net 2.0 [24]) as well as the similar AEN-BGS on the DAVIS-2016 dataset. According to Table 4, our SAEN-BGS displays a notable increase of over 15% in F-measure compared to ZBS and BSUV-Net 2.0. Moreover, as shown in Fig. 8, our SAEN-BGS and the competing AEN-BGS both generate visually accurate solutions that closely match the ground truth, unlike the two other top-performing baseline methods. It is worth mentioning that DAVIS-2016 is more challenging than CDnet-2014 because it has more categories and fewer shots in each category. Hence, it is evident that the ZBS and BSUV-Net 2.0 models can easily be deceived by the intricate background objects in DAVIS-2016. Our proposed SAEN-BGS and its counterpart AEN-BGS are able to circumvent this issue.

Table 6
Energy performance comparison between our proposed SAEN-BGS and its counterpart AEN-BGS under the small training set (randomly selected 200 frames per video) on CDnet-2014 dataset. **BOLD** indicates the best.

Scene	Model	Rec (↑)	Spe (↑)	FPR (↓)	FNR (↓)	PWC (↓)	Fm (↑)	Pre (↑)	AvR$_s$ (↓)	Energy (↓)
Baseline	AEN-BGS	**95.05**	**99.81**	**0.18**	4.94	**0.375**	**95.18**	**95.31**	100.0	4.10e+10
	SAEN-BGS	94.07	99.78	0.21	5.92	0.430	94.19	94.30	**12.06**	**3.84e+9**
Bad.Wthr	AEN-BGS	**93.36**	**99.91**	**0.08**	6.63	**0.196**	**94.30**	**95.27**	100.0	4.10e+10
	SAEN-BGS	92.32	99.83	0.16	7.67	0.327	92.37	92.42	**11.50**	**3.09e+9**
Cam.Jtr	AEN-BGS	**94.29**	**99.76**	**0.23**	5.70	**0.429**	**93.89**	**93.48**	100.0	4.10e+10
	SAEN-BGS	90.62	99.67	0.32	9.30	0.653	91.05	91.48	**11.98**	**3.78e+9**
Dyn.Bkgr	AEN-BGS	**93.95**	**99.95**	**0.04**	6.04	**0.101**	**94.26**	**94.58**	100.0	4.10e+10
	SAEN-BGS	91.91	99.92	0.07	8.08	0.174	92.85	93.81	**12.25**	**3.73e+9**
Int.Obj.Mot	AEN-BGS	**90.11**	**99.92**	**0.07**	9.88	**0.171**	**91.40**	**92.73**	100.0	4.10e+10
	SAEN-BGS	86.40	99.91	0.08	13.59	0.240	89.12	92.02	**12.15**	**3.69e+9**
Low.Frm.Rate	AEN-BGS	**93.75**	**99.78**	**0.21**	6.24	**0.336**	**92.03**	**90.38**	100.0	4.10e+10
	SAEN-BGS	88.73	99.76	0.23	11.26	0.475	89.12	89.52	**12.62**	**4.20e+9**
Nig.Vid	AEN-BGS	**85.23**	**99.91**	**0.08**	14.76	**0.212**	**87.78**	**90.50**	100.0	4.10e+10
	SAEN-BGS	80.48	99.90	0.09	19.51	0.265	84.35	88.61	**12.46**	**5.09e+9**
PTZ	AEN-BGS	**88.32**	99.87	0.12	**11.67**	0.212	**86.67**	**85.09**	100.0	4.10e+10
	SAEN-BGS	84.77	**99.89**	**0.10**	15.22	**0.185**	83.14	81.57	**12.80**	**4.77e+9**
Shadow	AEN-BGS	**93.54**	**99.77**	**0.22**	6.45	**0.459**	**93.75**	**93.97**	100.0	4.10e+10
	SAEN-BGS	91.77	99.63	0.36	8.22	0.647	91.17	90.57	**12.38**	**4.47e+9**
Thrml	AEN-BGS	**94.62**	**99.70**	**0.29**	5.37	**0.692**	**95.48**	**96.35**	100.0	4.10e+10
	SAEN-BGS	93.39	99.41	0.58	6.60	1.025	93.02	92.66	**11.97**	**3.84e+9**
Turb	AEN-BGS	89.54	**99.96**	**0.03**	10.45	**0.079**	91.01	**92.54**	100.0	4.10e+10
	SAEN-BGS	**91.12**	99.94	0.05	**8.87**	0.113	**91.01**	90.90	**12.37**	**3.98e+9**

Table 7
Energy performance between our proposed SAEN-BGS and its counterpart AEN-BGS under the large training set (randomly selected 70% frames per video) on CDnet-2014 dataset. **BOLD** indicates the best.

Scene	Model	Rec (↑)	Spe (↑)	FPR (↓)	FNR (↓)	PWC (↓)	Fm (↑)	Pre (↑)	AvR$_s$ (↓)	Energy (↓)
Baseline	AEN-BGS	**97.66**	**99.87**	**0.12**	2.33	**0.221**	**97.47**	**97.27**	100.0	4.10e+10
	SAEN-BGS	97.04	99.81	0.18	2.95	0.308	96.49	95.94	**12.42**	**5.52e+9**
Bad.Wthr	AEN-BGS	**96.05**	**99.96**	**0.03**	3.94	**0.089**	**96.44**	**96.82**	100.0	4.10e+10
	SAEN-BGS	95.42	99.94	0.05	4.57	0.108	95.67	95.91	**12.27**	**5.93e+9**
Cam.Jtr	AEN-BGS	**96.54**	**99.85**	**0.14**	3.45	**0.290**	**96.75**	**96.96**	100.0	4.10e+10
	SAEN-BGS	96.22	99.82	0.17	3.77	0.337	96.22	96.23	**12.41**	**5.43e+9**
Dyn.Bkgr	AEN-BGS	**96.20**	**99.97**	**0.02**	3.79	**0.059**	**96.28**	**96.36**	100.0	4.10e+10
	SAEN-BGS	94.51	99.95	0.04	5.48	0.084	94.69	94.88	**12.77**	**7.49e+9**
Int.Obj.Mot	AEN-BGS	**94.04**	**99.95**	**0.04**	5.95	**0.091**	**94.54**	**95.05**	100.0	4.10e+10
	SAEN-BGS	90.48	99.92	0.07	9.51	0.158	90.60	90.72	**12.69**	**6.30e+9**
Low.Frm.Rate	AEN-BGS	**94.64**	**99.91**	**0.08**	5.35	**0.166**	**94.56**	**94.48**	100.0	4.10e+10
	SAEN-BGS	93.31	99.89	0.10	6.68	0.209	93.16	93.01	**12.91**	**6.05e+9**
Nig.Vid	AEN-BGS	**88.62**	**99.94**	**0.05**	11.37	**0.136**	**90.75**	**92.98**	100.0	4.10e+10
	SAEN-BGS	86.50	99.91	0.08	13.49	0.183	87.67	88.86	**12.69**	**1.00e+10**
PTZ	AEN-BGS	**92.58**	99.95	**0.04**	7.41	**0.069**	**87.76**	**83.41**	100.0	4.10e+10
	SAEN-BGS	89.13	**99.95**	**0.04**	10.86	0.078	85.88	82.87	**12.56**	**5.44e+9**
Shadow	AEN-BGS	**95.96**	**99.82**	**0.17**	4.03	**0.325**	**95.83**	**95.70**	100.0	4.10e+10
	SAEN-BGS	94.24	99.73	0.26	5.75	0.478	93.89	93.54	**13.18**	**9.61e+9**
Thrml	AEN-BGS	**97.25**	**99.86**	**0.13**	2.74	**0.282**	**97.53**	**97.81**	100.0	4.10e+10
	SAEN-BGS	94.42	99.80	0.19	5.57	0.508	95.51	96.63	**13.05**	**1.05e+10**
Turb	AEN-BGS	**92.15**	**99.97**	**0.02**	7.84	**0.053**	**92.75**	**93.36**	100.0	4.10e+10
	SAEN-BGS	91.05	99.96	0.03	8.94	0.064	91.34	91.63	**12.42**	**7.29e+9**

Overall, these experiments effectively showcase our suggested techniques for improving BGS performance, as our SAEN-BGS has proven to surpass the current benchmarks. When dealing with practical situations, our main focus is on achieving efficient energy consumption, which involves striking a balance between effectiveness and energy efficiency. Ultimately, efficient energy consumption decreases the amount of active neurons, but this comes at the expense of effectiveness. We

will validate the energy performance of AEN-BGS and SAEN-BGS in the upcoming experiments.

4.3. Energy comparison

We not only improve segmentation performance but also verify our power consumption. According to Eqs. (12) and (14), it is evident that

Table 8
Energy performance between our proposed SAEN-BGS and its counterpart AEN-BGS on DAVIS-2016 dataset. **BOLD** indicates the best.

Model	Fm (↑)	AvR_s (↓)	Energy (↓)
AEN-BGS	**87.40**	100.0	4.10e+10
SAEN-BGS	85.20	**13.97**	**1.17e+10**

the calculation of flops for ANNs is independent of the particular task, unlike SNNs. In order to do this, we initially calculate the parameter count and energy of AEN-BGS, BSUV-Net 2.0, and ZBS, all of which rely on ANNs and demonstrate superior performance compared to other baseline models. According to the data in Table 5, AEN-BGS has significantly fewer parameters and uses less energy than both ZBS and BSUV-Net 2.0. We will then compare our SAEN-BGS with its counterpart CNN (AEN-BGS) on CDnet-2014 and DAVIS-2016 individually to demonstrate the energy efficiency of our SAEN-BGS.

In CDnet-2014, we evaluate not just the seven fundamental metrics but also two additional energy metrics (Energy and AvR_s) across 11 categories, following the two previously mentioned training schemes. Our SAEN-BGS demonstrates similar performance to AEN-BGS across the seven fundamental metrics, as shown in Tables 6 and 7. Our method achieves significantly lower AvR_s and energy consumption with an average neuron activation of only 12% and energy savings ranging from 74% to 92%. Furthermore, we also perform energy comparisons on DAVIS-2016 using a large training set scheme. As shown in Table 8, our SAEN-BGS achieves a comparable F-measure with significantly lower energy usage, with just 13.97% neuron activation and almost 71% less energy consumed.

In summary, SAEN-BGS is suggested for situations with limited energy, whereas AEN-BGS is the choice for less crucial energy limitations.

5. Conclusion

This paper proposes a spiking autoencoder network (termed SAEN-BGS) for background subtraction to address sensitivity to temporal background noise. The continuous spiking conv-dconv block is designed for temporal feature refinement, which achieves an average 7% higher F-measure on CDnet-2014 and 16% higher F-measure on average over DAVIS-2016 compared with other baselines. These results demonstrate our superior capability to disentangle foreground temporal patterns from noisy backgrounds. Then, we present the self-auxiliary spiking learning under the ANN-to-SNN framework, which facilitates our SAEN-BGS to reduce energy by more than 50% compared to state-of-the-art baselines while maintaining at least 70% energy saving compared to its counterpart. These findings suggest that spike-event-driven SNNs inherently suppress background noise propagation.

This approach has produced significant outcomes on commonly utilized benchmarks. Nevertheless, it does possess specific constraints. Compared to conventional neural networks, there are fewer established libraries and frameworks for working with SNNs, which can hinder development and experimentation. Furthermore, as the size of the network increases, the tradeoff between efficiency and effectiveness can become problematic, particularly with standard training algorithms using backpropagation.

CRediT authorship contribution statement

Zhixuan Zhang: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology. **Xiao Peng Li:** Writing – review & editing, Writing – original draft, Supervision. **Qi Liu:** Writing – review & editing, Validation, Supervision, Software, Project administration, Methodology, Investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data and code availability

Our key codes are released at: https://github.com/ZhixuanZhang77/SAEN-BGS..

References

[1] Q. Liu, X.P. Li, H. Cao, Y. Wu, From simulated to visual data: A robust low-rank tensor completion approach using lp-regression for outlier resistance, IEEE Trans. Circuits Syst. Video Technol. (2021) http://dx.doi.org/10.1109/TCSVT.2021.3114208, 1–1.

[2] Q. Liu, X.P. Li, J. Yang, Optimum codesign for image denoising between type-2 fuzzy identifier and matrix completion denoiser, IEEE Trans. Fuzzy Syst. 30 (1) (2022) 287–292, http://dx.doi.org/10.1109/TFUZZ.2020.3030498.

[3] M. Ashraphijuo, X. Wang, Structured alternating minimization for union of nested low-rank subspaces data completion, IEEE J. Sel. AREA. Inf. Theory 1 (3) (2020) 632–644, http://dx.doi.org/10.1109/JSAIT.2020.3039170.

[4] J. Jang, Y. Wang, C. Kim, Fcgnet: Foreground and class guided network for human parsing, Pattern Recognit. 157 (2025) 110879.

[5] M. Ling, T. Pan, Y. Ren, K. Wang, X. Geng, Motional foreground attention-based video crowd counting, Pattern Recognit. 144 (2023) 109891.

[6] F. Zhong, Y. Wu, H. Yu, G. Wang, Z. Lu, A benchmark dataset and semantics-guided detection network for spatial–temporal human actions in urban driving scenes, Pattern Recognit. 158 (2025) 111035.

[7] S.-C. Cheung, K. Chandrika, Robust background subtraction with foreground validation for urban traffic video, EURASIP J. Adv. Signal Process. 14 (2005) 2330–2340, http://dx.doi.org/10.1155/ASP.2005.2330.

[8] J.-W. Seo, S.D. Kim, Recursive on-line 2D PCA and its application to long-term background subtraction, IEEE Trans. Multimed. 16 (8) (2014) 2333–2344.

[9] Z. Chen, T. Ellis, A self-adaptive Gaussian mixture model, Comput. Vis. Image Understand. 122 (2014) 35–46, http://dx.doi.org/10.1016/j.cviu.2014.01.004.

[10] A. Vinciarelli, M. Pantic, H. Bourlard, Social signal processing: Survey of an emerging domain, Image Vis. Comput. 27 (12) (2009) 1743–1759, http://dx.doi.org/10.1016/j.imavis.2008.11.007.

[11] P. Chiranjeevi, S. Sengupta, New fuzzy texture features for robust detection of moving objects, IEEE Signal Process. Lett. 19 (10) (2012) 603–606.

[12] H. Zhang, D. Xu, Fusing color and texture features for background model, in: International Conference on Fuzzy Systems and Knowledge Discovery, Springer, 2006, pp. 887–893.

[13] S. Jabri, Z. Duric, H. Wechsler, A. Rosenfeld, Detection and location of people in video images using adaptive fusion of color and edge information, in: Proceedings 15th International Conference on Pattern Recognition, ICPR-2000, vol. 4, (4) 2000, pp. 627–630.

[14] A. Elgammal, D. Harwood, L. Davis, Non-parametric model for background subtraction, in: Proc. European Conf. Computer Vision, Berlin, Heidelberg, 2000, pp. 751–767.

[15] C. Stauffer, W.E.L. Grimson, Adaptive background mixture models for real-time tracking, in: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2, (2) 1999, pp. 246–252, http://dx.doi.org/10.1109/CVPR.1999.784637.

[16] X. Liu, G. Zhao, J. Yao, C. Qi, Background subtraction based on low-rank and structured sparse decomposition, IEEE Trans. Image Process. 24 (8) (2015) 2502–2514, http://dx.doi.org/10.1109/TIP.2015.2419084.

[17] M. Braham, S. Pierard, M. Van Droogenbroeck, Semantic background subtraction, in: Proc. Int. Conf. Image Process., 2017, pp. 4552–4556, http://dx.doi.org/10.1109/ICIP.2017.8297144.

[18] M. Braham, M.V. Droogenbroeck, Deep background subtraction with scene-specific convolutional neural networks, in: Proc. Int. Conf. Syst. Signals Image Process., 2016, pp. 1–4, http://dx.doi.org/10.1109/IWSSIP.2016.7502717.

[19] Y. Wang, Z. Luo, P.M. Jodoin, Interactive deep learning method for segmenting moving objects, Pattern Recognit. 96 (2017) 66–75, http://dx.doi.org/10.1016/j.patrec.2016.09.014.

[20] K. Lim, W.D. Jang, C.S. Kim, Background subtraction using encoder–decoder structured convolutional neural network, in: Proc. IEEE Int. Conf. Adv. Video Signal Based-Surveill, 2017, http://dx.doi.org/10.1109/avss.2017.8078547.

[21] M. Mandal, V. Dhar, A. Mishra, S.K. Vipparthi, 3Dfr: A swift 3d feature reductionist framework for scene independent change detection, IEEE Signal Process. Lett. 26 (12) (2019) 1882–1886.

[22] L.A. Lim, H.Y. Keles, Foreground segmentation using convolutional neural networks for multiscale feature encoding, Pattern Recognit. 112 (2018) 256–262, http://dx.doi.org/10.1016/j.patrec.2018.08.002.

[23] M.O. Tezcan, P. Ishwar, J. Konrad, Bsuv-net: A fully-convolutional neural network for background subtraction of unseen videos, in: Proc. IEEE Winter Conf. Applicat. Comput. Vis, 2020, pp. 2763–2772, http://dx.doi.org/10.1109/WACV45572.2020.9093464.

[24] M.O. Tezcan, P. Ishwar, J. Konrad, Bsuv-net2.0: Spatio-temporal data augmentations for video-agnostic supervised background subtraction, IEEE Access. 9 (2021) 53849–53860, http://dx.doi.org/10.1109/ACCESS.2021.3071163.

[25] C. Stauffer, W. Grimson, Adaptive background mixture models for real-time tracking, in: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2, IEEE Computer Society, Los Alamitos, CA, USA, 1999, p. 2246, http://dx.doi.org/10.1109/CVPR.1999.784637.

[26] A. Elgammal, R. Duraiswami, D. Harwood, L.S. Davis, Background and foreground modeling using nonparametric kernel density estimation for visual surveillance, Proc. IEEE 90 (2002) 1151–1163, http://dx.doi.org/10.1109/JPROC.2002.801448.

[27] O. Barnich, M.V. Droogenbroeck, Vibe: A universal background subtraction algorithm for video sequences, IEEE Trans. Image Process. 20 (6) (2010) 1709–1724, http://dx.doi.org/10.1109/TIP.2010.2101613.

[28] P.L. St-Charles, G.A. Bilodeau, R. Bergevin, A self-adjusting approach to change detection based on background word consensus, in: Proc. IEEE Winter Conf. Applicat. Comput. Vis, 2015, pp. 990–997, http://dx.doi.org/10.1109/WACV.2015.137.

[29] P. St-Charles, G. Bilodeau, R. Bergevin, Subsense: A universal change detection method with local adaptive sensitivity, IEEE Trans. Image Process. 24 (1) (2015) 359–373, http://dx.doi.org/10.1109/TIP.2014.2378053.

[30] S. Isik, K. Ozkan, S. Gunal, O.N. Gerek, Swcd: a sliding window and self-regulated learning-based background updating method for change detection in videos, J. Electron. Imaging. 27 (2) (2018) 1, http://dx.doi.org/10.1117/1.JEI.27.2.023002.

[31] S.H. Lee, S.C. Kwon, J.W. Shim, J. Yoo, Wisenetmd: Motion detection using dynamic background region analysis, 2018, arXiv arXiv:1805.09277.

[32] Y. An, X. Zhao, T. Yu, H. Guo, C. Zhao, M. Tang, J. Wang, Zbs: Zero-shot background subtraction via instance-level background modeling and foreground selection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 6355–6364.

[33] M. Babaee, D.T. Dinh, G. Rigoll, A deep convolutional neural network for video sequence background subtraction, Pattern Recognit. 76 (2018) 635–649, http://dx.doi.org/10.1016/j.patcog.2017.09.040.

[34] D. Sakkos, H. Liu, J. Han, L. Shao, End-to-end video background subtraction with 3d convolutional neural networks, Multimedia Tools Appl. (2017) 1–19, http://dx.doi.org/10.1007/s11042-017-5460-9.

[35] G. Li, L. Deng, H. Tang, G. Pan, Y. Tian, K. Roy, W. Maass, Brain-inspired computing: A systematic survey and future trends, Proc. IEEE 112 (6) (2024) 544–584, http://dx.doi.org/10.1109/JPROC.2024.3429360.

[36] G. Zhang, L. Feng, F. Zhou, Z. Yang, Q. Zhang, A. Saleh, P.K. Donta, C.K. Dehury, Spiking neural networks in intelligent edge computing, IEEE Consum. Electron. Mag. (2024) 1–9, http://dx.doi.org/10.1109/MCE.2024.3506502.

[37] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A.J. Davison, J. Conradt, K. Daniilidis, D. Scaramuzza, Event-based vision: A survey, IEEE Trans. Pattern Anal. Mach. Intell. 44 (1) (2022) 154–180, http://dx.doi.org/10.1109/TPAMI.2020.3008413.

[38] R. Xiao, Q. Yu, R. Yan, H. Tang, Fast and accurate classification with a multi-spike learning algorithm for spiking neurons, in: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, vol. 7, 2019, pp. 1445–1451.

[39] M. Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, S.J. Thorpe, T. Masquelier, Bio-inspired digit recognition using reward-modulated spike-timing-dependent plasticity in deep convolutional networks, Pattern Recognit. 94 (2019) 87–95.

[40] J. Wu, Y. Chua, M. Zhang, G. Li, H. Li, K.C. Tan, A tandem learning rule for effective training and rapid inference of deep spiking neural networks, IEEE Trans. Neural Netw. Learn. Syst. 34 (1) (2021) 446–460.

[41] M. Arjovsky, L. Bottou, Towards principled methods for training generative adversarial networks, 2017, arXiv preprint arXiv:1701.04862.

[42] R. Ronan, J. Weston, A unified architecture for natural language processing: deep neural networks with multitask learning, in: Machine Learning, Proceedings of the Twenty-Fifth International Conference, ICML 2008, Helsinki, Finland, 2008, pp. 160–167, http://dx.doi.org/10.1145/1390156.1390177.

[43] D. Sun, A. Yao, A. Zhou, H. Zhao, Deeply-supervised knowledge synergy, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 6997–7006.

[44] Y. Wang, P. Jodoin, F. Porikli, J. Konrad, Y. Benezeth, P. Ishwar, CDnet 2014: An expanded change detection benchmark dataset, in: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., 2014, pp. 393–400, http://dx.doi.org/10.1109/CVPRW.2014.126.

[45] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. VanGool, M. Gross, A. Sorkine-Hornung, A benchmark dataset and evaluation methodology for video object segmentation, in: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., 2016, pp. 724–732, http://dx.doi.org/10.1109/CVPR.2016.85.

[46] A. Paszke, S. Gross, F. Massa, A. Lerer, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems, vol. 33, 2019.

[47] M. De Gregorio, M. Giordano, Change detection with weightless neural networks, in: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., 2014, pp. 409–413, http://dx.doi.org/10.1109/CVPRW.2014.66.

[48] M. Sedky, M. Moniri, C.C. Chibelushi, Spectral-360: A physics-based technique for change detection, in: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., 2014, pp. 405–408, http://dx.doi.org/10.1109/CVPRW.2014.65.

[49] R. Wang, F. Bunyak, G. Seetharaman, K. Palaniappan, Static and moving object detection using flux tensor with split Gaussian models, in: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., 2014, pp. 420–424, http://dx.doi.org/10.1109/CVPRW.2014.68.

[50] P.-L. St-Charles, G.-A. Bilodeau, R. Bergevin, Flexible background subtraction with self-balanced local sensitivity, in: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., 2014, pp. 414–419, http://dx.doi.org/10.1109/CVPRW.2014.67.

[51] S. Bianco, G. Ciocca, R. Schettini, Combination of video change detection algorithms by genetic programming, IEEE Trans. Evol. Comput. 21 (6) (2017) 914–928, http://dx.doi.org/10.1109/TEVC.2017.2694160.

[52] D. Zeng, X. Chen, M. Zhu, M. Goesele, A. Kuijper, Background subtraction with real-time semantic segmentation, IEEE Access 7 (2019) 153869–153884, http://dx.doi.org/10.1109/ACCESS.2019.2899348.

[53] Y. Wang, Z. Luo, P.-M. Jodoin, Interactive deep learning method for segmenting moving objects, Pattern Recognit. Lett. 96 (2017) 66–75, http://dx.doi.org/10.1016/j.patrec.2016.09.014.

[54] Y. Kim, P. Panda, Revisiting batch normalization for training low-latency deep spiking neural networks from scratch, 2020, arXiv arXiv:2010.01729.

[55] M. Horowitz, 1.1 computing's energy problem (and what we can do about it), in: 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers, ISSCC, 2014, pp. 10–14.