IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 8 August 2022, accepted 22 September 2022, date of publication 26 September 2022, date of current version 30 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3209671

RESEARCH ARTICLE

DTS-SNN: Spiking Neural Networks With Dynamic Time-Surfaces

DONGHYUNG YOO AND DOO SEOK JEONG, (Member, IEEE)
Division of Materials Science and Engineering, Hanyang University, Seoul 04763, South Korea

Corresponding author: Doo Seok Jeong (dooseokj@hanyang.ac.kr)

This work was supported by the National Research and Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT under Grant NRF-2021M3F3A2A01037632 and Grant NRF-2019R1C1C1009810.

ABSTRACT Convolution helps spiking neural networks (SNNs) capture the spatio-temporal structures of neuromorphic (event) data as evident in the convolution-based SNNs (C-SNNs) with the state-of-the-art classification-accuracies on various datasets. However, the efficacy aside, the efficiency of C-SNN is questionable. In this regard, we propose SNNs with novel trainable dynamic time-surfaces (DTS-SNNs) as efficient alternatives to convolution. The novel dynamic time-surface proposed in this work features its high responsiveness to moving objects given the use of the zero-sum temporal kernel that is motivated by the simple cells' receptive fields in the early stage visual pathway. We evaluated the performance and computational complexity of our DTS-SNNs on three real-world event-based datasets (DVS128 Gesture, Spiking Heidelberg dataset, N-Cars). The results highlight high classification accuracies and significant improvements in computational efficiency, e.g., merely 1.51% behind of the state-of-the-art result on DVS128 Gesture but a $\times 18$ improvement in efficiency. The code is available online (https://github.com/dooseokjeong/DTS-SNN).

INDEX TERMS Lightweight spiking neural network, spiking neural network, dynamic time-surfaces, event-based data.

I. INTRODUCTION

Convolution-based methods are pervasive in a variety of deep learning application domains given their high efficacy across different domains when implemented in convolutional neural networks (CNNs). The same holds for spiking neural networks (SNNs) in that convolution-based SNNs (C-SNNs) hold the state-of-the-art classification accuracies on a variety of datasets [1], [2], [3]. Convolution is an operation-intensive method that involves a large number of multiply-accumulate operations over 3D feature maps. Therefore, convolution generally results in high computational complexity and high power consumption, which is a daunting challenge, particularly for C-SNNs, because power efficiency is supposed to be one of the key advantages of SNNs over deep neural networks (DNNs).

SNNs are time-dependent hypotheses consisting of spiking units and unidirectional synapses [4]. One of the advantages of SNNs is their operations based on asynchronous spikes, unlike layer-wise sequential operations in DNNs which impose forward locking constraints [5], [6]. To leverage this advantage, it is required to implement SNNs in dedicated hardware, which is referred to as neuromorphic hardware [7], [8], [9], [10], [11]. Generally, a neuromorphic processor consists of multiple cores supporting asynchronous event-based operations across them. The consequent power efficiency is the key feature of neuromorphic hardware.

Time-surface (TS) analyses are effective methods to process asynchronous events (spikes) for various tasks [12], [13], [14]. A TS for a given event is a 2D map of the event timestamps prior to the event in the spatial vicinity of the event. Therefore, the TS can capture the spatio-temporal local structure of the events responding to the object. Nevertheless, the previous TSs are not tailored to SNNs and hardly support end-to-end learning.

In this regard, we attempt to use TSs, in place of convolution, to extract the features of event data in a highly operation-efficient manner to leverage the key advantage,

The associate editor coordinating the review of this manuscript and approving it for publication was Fu-Kwun Wang.

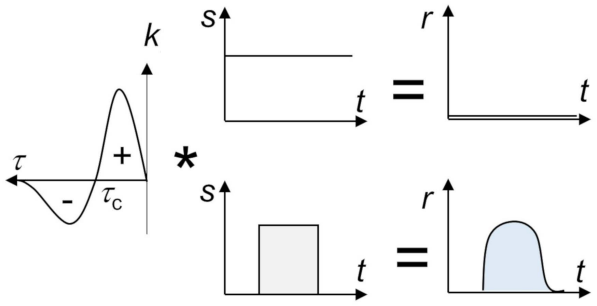

FIGURE 1. Schematic of the temporal kernel (*k*) of a simple cell, rendering the simple cell responsive to time-varying stimuli (*s*). The response is denoted by *r*.

i.e., power efficiency, of SNNs. To the best of our knowledge, this work is the first attempt to integrate TSs into SNNs to process event-based data efficiently. Moreover, we significantly modified the conventional TSs to better capture the dynamic features of event-data by using a zero-sum temporal kernel motivated by the temporal kernels of simple cells in the early stage visual pathways [15]. Additionally, our TSs are dynamic insomuch as they are calculated for every timestep to encode event dynamics unlike the previous TSs, which are referred to as dynamic time-surfaces (DTSs). The primary contributions of our work include the following:

- We propose DTS-SNN to replace C-SNN, which is remarkably lightweight but exhibits high classification accuracy on event-based data.
- We propose trainable DTSs that are susceptible to moving objects and fully support end-to-end learning.
- We evaluate the classification accuracy and computational efficiency of DTS-SNNs on various event-based datasets, including DVS128 Gesture [16], Spiking Heidelberg dataset (SHD) [17], and N-Cars [13].

II. RELATED WORK

In the early stage of visual processing, simple cells (linear neurons) exhibit linear responses to visual inputs in their receptive fields of particular spatial and temporal structures [15]. The temporal structure (kernel) of receptive fields features alternating positive and negative contributions of input to the simple cell's response in time such that the input at a particular point (t_0) causes a positive contribution within a time window (t_c in width, i.e., if $t - t_0 \leq t_c$), which turns into a negative contribution when the time exceeds the window. A schematic of the temporal kernel is illustrated in Figure 1. This type of temporal kernel supports the responsiveness of simple cells to moving objects over their receptive fields. A comparison between simple cell's responses to static and moving objects is depicted in Figure 1.

As the early research of TS analysis, HOTS considers only the last timestamps of the pixels in a given TS. Although HOTS successfully introduced the concept of TS to process event data, the TS was prone to noises that are events irrelevant to the objects under consideration [12].

As a workaround, Sironi *et al.* proposed HATS that is based on averaged TSs [13]. For a compact representation, they partitioned the input field into grid-cells. For each grid-cell, the TSs for several recent events in the grid-cell are averaged over the entire timesteps and grid-cell to acquire a single smooth TS that is representative for the grid-cell. Unlike HOTS, in HATS, the TS of each event considers several recent timestamps by convolving an event stream with an exponentially decaying temporal kernel.

HOTS uses a set of TSs as a dictionary to compare it with the input TS and to consequently build a feature map (matching frequency). This comparison is repeated through multiple stages. The feature histogram from the last stage is used to categorize input data, which is based on the histogram-similarity between the input and instances in each category. HATS uses grid-cell-wise averaged TSs as a dictionary. Similar to HOTS, the feature map is built based on matching frequency but uses a support vector machine as a classifier. Notably, both HOTS and HATS use time-invariant (static) TSs as inputs to their classifiers.

With the development of training algorithms for SNNs, there have been attempts to process event data by exploiting the spatio-temporal processing ability of SNNs [1], [2], [18], [19], [20]. Yet, to achieve high classification accuracies, most of them used large C-SNNs with multiple hidden layers, which cause significant computational complexity.

This lets us revisit the initial motivation of SNNs, energy-efficiency, and consequently rethink of efficient methods to extract the spatio-temporal features of event data using TSs as alternatives to convolution. To this end, the prerequisites include (i) the modification of the conventional time-invariant TSs to time-dependent (dynamic) forms with a noise-robust temporal kernel and (ii) development of a DTS builder supporting end-to-end batch learning.

III. SPIKING NEURAL NETWORKS WITH DYNAMIC TIME-SURFACES

DTS-SNNs consists of a DTS builder and SNN classifier. The builder constructs DTSs for the events at every timestep, which are subsequently fed into the SNN as inputs. To validate the feature extraction ability of the proposed DTS builder and the importance of well-defined features for SNNs, we used a simple dense SNN with a single hidden layer, which was trained using a surrogate gradient-based backpropagation algorithm [21]. This section elucidates the DTS in comparison with the previous TSs and a method to build DTSs in parallel for the samples in a single batch.

A. DYNAMIC TIME-SURFACES WITH ZERO-SUM TEMPORAL KERNELS

For an event stream from an event camera, the ith event (e_i) is encoded as $e_i = (p_i, t_i, \mathbf{X}_i)$, where p_i, t_i, and $\mathbf{X}_i$ denote its polarity $p_i \in \{-1, 1\}$, timestamp, and location on a 2D pixel array $\mathbf{X}_i = (x_i, y_i)$, respectively. The DTS for the ith event $\mathcal{T}_{e_i}$ only considers the previous or simultaneous events $e_j (j \leq i)$ of the same polarity ($p_j = p_i$), which are

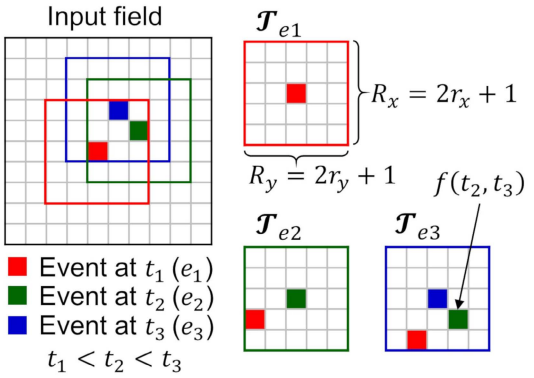

FIGURE 2. TSs for three consecutive events of the same polarity at t_1, t_2, and t_3. We set r_x and r_y to 2 so that the spatial domain for each TS $\mathcal{D}_{e_i}$ is a 5 × 5 grid. The function f denotes a timestamp-encoding function.

located at $\mathbf{X}_j = (x_j, y_j) \in \mathcal{D}_{e_i}$. The spatial domain $\mathcal{D}_{e_i}$ is defined as $\mathcal{D}_{e_i} = \{(x_j, y_j) \,|\, |x_j - x_i| \le r_x, |y_j - y_i| \le r_y\}$. Thus, $\mathcal{T}_{e_i}(p_i, \Delta x_{ij}, \Delta y_{ij}) \in \mathbb{R}^{2\times R_x \times R_y}$, where $\Delta x_{ij} = x_j - x_i$, $\Delta y_{ij} = y_j - y_i$, $R_x = 2r_x + 1$, and $R_y = 2r_y + 1$. An example of building DTSs for three consecutive events is illustrated in Figure 2.

The DTS at timestep t_i is encoded with the novel zero-sum temporal kernel k_{tzs} as

$$\mathcal{T}_{e_i}(p_i, \Delta x_{ij}, \Delta y_{ij}) = (k_{tzs} * \rho)(t_i), \tag{1}$$

The event stream ρ for each location $\mathbf{X}_j$ is described by

$$\rho(t; \mathbf{X}_j) = \sum_{t_k \in \mathbf{t}_k} \delta(t - t_k), \tag{2}$$

where δ is the Dirac delta function, and $\mathbf{t}_k$ is a set of all previous timestamps $\mathbf{t}_k = \{t_k | k \le j, \mathbf{X}_k = \mathbf{X}_j\}$. The zero-sum temporal kernel k_{tzs} is given by

$$k_{tzs} = e^{-\tau/\tau_1} - \frac{\tau_1}{\tau_2} e^{-\tau/\tau_2}, \text{ where } \tau_1 < \tau_2, \tag{3}$$

which is distinguished from a single exponential temporal kernel k_t used in HOTS and HATS.

$$k_t = e^{-\tau/\tau_0}. \tag{4}$$

Eq. (3) is termed zero-sum temporal kernel because the convolution with this kernel over an event stream constant firing rate yields zero due to the balance between the positive and negative sub-kernels in Eq. (3).

Lemma 1: Consider the convolution of a train of Poisson spikes ρ at a constant firing rate r using the zero-sum temporal kernel k_{tzs}, $y(t) = (k_{tzs} * \rho)(t)$. The result converges to zero as time t increases, i.e., $y(\infty) = 0$.

The derivation of **Lemma 1** is shown in Appendix A. According to **Lemma 1**, the kernel yields high responsiveness to spike trains at time-dependent firing rates by filtering out the spikes at constant firing rates. Consequently, the zero-sum temporal kernel endows the timestamp encoding with high responsiveness to moving objects compared with the single-exponential kernel.

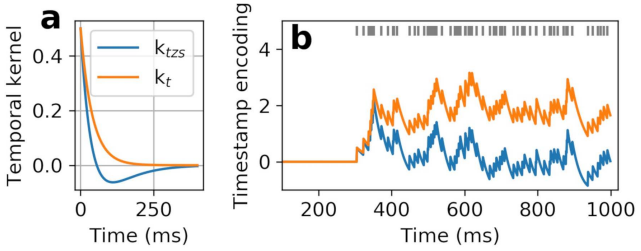

FIGURE 3. (*a*) Zero-sum temporal kernel k_{tzs} with $\tau_1 = 40$ ms and $\tau_2 = 80$ ms compared with the single-exponential kernel k_t with $\tau_0 = 40$ ms. (*b*) Encoding the timestamps (gray bars) using k_{tzs} and k_t. The events were generated at a constant rate of 100 Hz.

Figure 3 shows an example of timestamps encoded using the zero-sum temporal kernel k_{tzs} compared with encoding using a single-exponential kernel k_t (Figure 3(a)). Figure 3(b) highlights the high responsiveness of the encoding function to events varying their rate such that it outputs high responses in the initial encoding phase while the following responses merely fluctuate around zero due to the constant event rate. This can clearly be differentiated from the timestamp encoding using the single-exponential kernel k_t as compared in Figure 3(b).

Similar to HATS, the input field is partitioned into grid-cells, and a single grid-cell-wise representative DTS is built for each grid-cell. However, unlike HATS, the representative DTS $\overline{\mathcal{T}}_c(t)$ for a given grid-cell c and timestep t is the weighted sum of the DTSs of simultaneous events $\mathcal{T}_{e_i}$.

$$\overline{\mathcal{T}}_c(t) = \sum_{e_i \in \mathbf{e}_{t,c}} a_i \mathcal{T}_{e_i}, \tag{5}$$

where $\mathbf{e}_{t,c} = \{e_i | t_i = t, \mathbf{X}_i \in c\}$. The weight of each element time-surface $\mathcal{T}_{e_i}$ is denoted by a_i which is a trainable parameter. This set of weights is shared among all grid-cells. Note that, in HATS, the representative TS is the simple average of the TSs of simultaneous events as follows.

$$\overline{\mathcal{T}}_c = \frac{1}{|\mathbf{e}_{t,c}|} \sum_{e_i \in \mathbf{e}_{t,c}} \mathcal{T}_{e_i}.$$

B. BUILDING DYNAMIC TIME-SURFACES IN PARALLEL

The key to training SNNs using DTSs on a given dataset is the parallel calculations of DTSs for all samples in a batch. Additionally, the compatibility of parallel calculations with readily available deep learning frameworks enhances efficiency. To this end, we propose pixel-wise timestamp-encoding banks $\mathbf{E}(t)$ that are updated once for every timestep. The timestamp encodings in the bank can readily be retrieved when events at particular pixels occur. This bank is subsequently unfolded to endow each pixel with an element time-surface. At a given timestep, the element time-surfaces for the simultaneous events are retrieved and summed with their weights to calculate the DTS for a given grid-cell.

We consider periodically distributed grid-cells over a $H \times W$ input field for a given polarity; each grid-cell is $h_c \times w_c$ in size so that there exist $H/h_c \times W/w_c$ grid-cells on

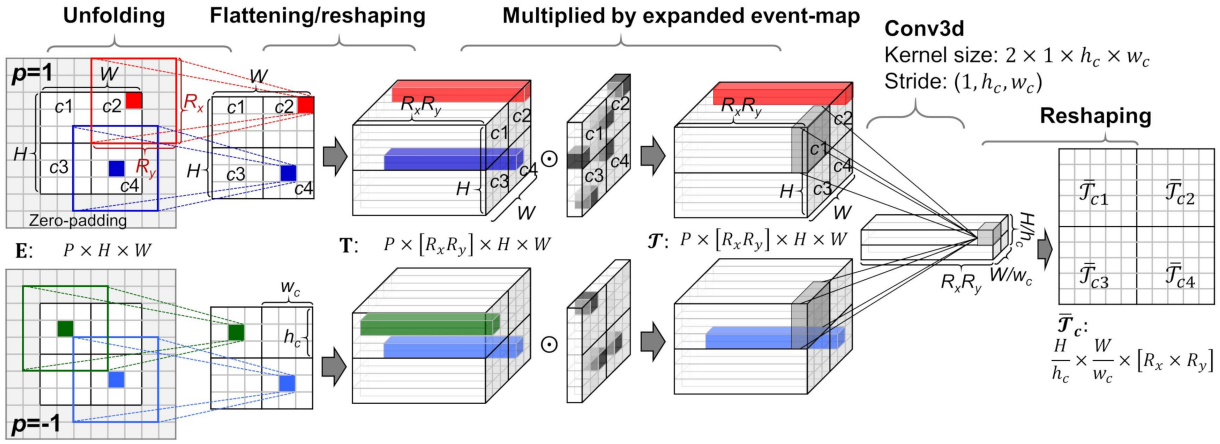

FIGURE 4. Procedure for building DTSs for a given sample at a given timestep t. R_x and R_y define the size of a DTS such that $R_x = 2r_x + 1$ and $R_y = 2r_y + 1$.

the input field. As such, the spatial domain of each DTS $\mathcal{D}_{e_i}$ is $R_x \times R_y$ in size, where $R_x = (2r_x + 1)$ and $R_y = (2r_y + 1)$. The procedure is detailed in the following subsections. The pseudocode is shown in Appendix D.

1) BUILDING TIMESTAMP-ENCODING BANKS

A timestamp-encoding bank $\mathbf{E}(t)$ is a bank of timestamp-encodings for all pixels so that its dimension is identical to the input field. $\mathbf{E}(t) \in \mathbb{R}^{P\times H\times W}$ for a $P \times H \times W$ input field. Each element $E_{pxy}(t)$ is calculated by convolving an event-stream (polarity p) at a location (x, y), $\rho(t; p, x, y)$ with the zero-sum temporal kernel k_{tzs},

$$E_{pxy}(t) = (k_{tzs} * \rho(t; p, x, y))(t). \tag{6}$$

For efficient computation, we transform this convolution into a recursive form as follows:

$$\begin{aligned} \mathbf{E}(t+1) &= \mathbf{E}_1(t+1) - \mathbf{E}_2(t+1), \\ \mathbf{E}_1(t+1) &= \mathbf{E}_1(t)\, e^{-1/\tau_1} + \mathbb{1}_{\mathbf{A}}(p, \mathbf{X}), \\ \mathbf{E}_2(t+1) &= \mathbf{E}_2(t)\, e^{-1/\tau_2} + \tau_1/\tau_2 \mathbb{1}_{\mathbf{A}}(p, \mathbf{X}). \end{aligned} \tag{7}$$

The indicator function $\mathbb{1}_{\mathbf{A}}(p, \mathbf{X})$ is an event-map at timestep t, where $\mathbf{A} = \{(p_i, \mathbf{X}_i) \,|t_i = t\}$.

2) UNFOLDING TIMESTAMP-ENCODING BANKS

The timestamp-encoding bank $\mathbf{E}(t) \in \mathbb{R}^{P\times H\times W}$ is subsequently unfolded to build a preliminary time-surface map $\mathbf{T}(t) \in \mathbb{R}^{P\times H\times W\times (R_x\times R_y)}$ in which each location in the $P \times H \times W$ input field is given a $R_x \times R_y$ preliminary time-surface centered at the location such that

$$T_{pxy}(t) \leftarrow \mathbf{E}_{p,(x-r_x):(x+r_x),(y-r_y):(y+r_y)}(t). \tag{8}$$

This process is indicated by "Unfolding" in Figure 4.

3) RESHAPING UNFOLDED BANKS

Each preliminary time-surface in the map $\mathbf{T}(t)$ is flattened, leading to a $P \times H \times W \times (R_xR_y)$ map. This is subsequently reshaped into a $P \times (R_xR_y) \times H \times W$ map. This reshaping is for the grid-cell-wise calculation of the weighted sum of the DTSs using 3D convolution. This process is depicted in Figure 4, indicated by "Flattening/reshaping".

4) MULTIPLICATION BY EVENT-MAPS

The reshaped preliminary time-surface map $\mathbf{T}(t)$ is read out to acquire the DTSs of the events occurring at timestep t. To this end, we reuse the event-map $\mathbb{1}_{\mathbf{A}}(p, \mathbf{X})$ in Eq. (7). This event-map is expanded by repeating the map along the flattened time-surface axis. This expanded event-map is element-wise multiplied by the map $\mathbf{T}(t)$, resulting in the map $\mathcal{T}(t)$ including nonzero element time-surfaces $\mathcal{T}_{e_i}(t)$ for the events at timestep t only. This process is indicated by "Multiplied by expanded event-map" in Figure 4. The elements in each flattened $\mathcal{T}_{e_i}(t)$ are L2-normalized.

5) GRID-CELL-WISE WEIGHTED SUM OF TIME-SURFACES

The input field is a $H/h_c \times W/h_w$ grid, and each grid-cell is $h_c \times w_c$ in size. For a given grid-cell, the DTS $\bar{\mathcal{T}}_c(t)$ in Eq. (5) is calculated by convolving the map $\mathcal{T}(t)$ along the time-surface axis using a kernel $\mathbf{a} \in \mathbb{R}^{P\times 1\times h_c\times w_c}$ with a stride of one. Note that the kernel elements indicate the contributions of the element time-surfaces $\mathcal{T}(t)$ to the grid-cell-wise representative DTS $\bar{\mathcal{T}}_c(t)$ as in Eq. (5). The same process is repeated for the next grid-cell: moving the kernel to the next grid-cell (with a stride of h_c or h_w) and convolving $\mathcal{T}(t)$ with a stride of one. Thus, the calculation of $\bar{\mathcal{T}}_c(t)$ is equivalent to 3D convolution of the element time-surface map $\mathcal{T}(t)$ using the rank-4 kernel $\mathbf{a}$. This allows us to readily use the convolution methods in the deep learning frameworks.

The resulting $(R_xR_y) \times H/h_c \times W/w_c$ map is a map of flattened DTSs for all grid-cells. The map is reshaped into $H/h_c \times W/w_c \times (R_x \times R_y)$. The number of operations involved ($\#\mathrm{OP}_{\mathrm{DTS}}$) is given by

$$\#\mathrm{OP}_{\mathrm{DTS}} = 2PHWR_xR_y. \tag{9}$$

C. SPIKING NEURAL NETWORK CLASSIFIER

The SNN classifier used is a simple fully-connected network (FCN) with a single hidden layer. The $H/h_c \times W/w_c \times R_x \times R_y$ DTS is flattened and fed to the SNN classifier as synaptic currents at a given timestep. Each dense layer consists of spiking neurons conforming to the spike-response model (SRM) but without a refractory mechanism. The sub-threshold membrane potential of the ith neuron in the lth layer is denoted by $u_i^{(l)}(t)$. Hereafter, the subscript and superscript of a variable mean a neuron index and layer index, respectively. The potential $u_i^{(l)}(t)$ is given by

$$u_i^{(l)}(t) = \sum_j w_{ij}^{(l)} \left(\epsilon * \rho_j^{(l-1)}\right)(t),$$
$$\epsilon = \frac{\tau_m}{\tau_m - \tau_s}\left(e^{-\tau/\tau_m} - e^{-\tau/\tau_s}\right)\Theta(\tau), \qquad (10)$$

where $w_{ij}^{(l)}$, τ_m, and τ_s denote the weight from the jth neuron in the $(l-1)$th layer, time-constant for potential decay, and time-constant for synaptic current decay, respectively. A spike train from the jth neuron in the $(l-1)$th layer is denoted by $\rho_j^{(l-1)}$. We use a spike function $S_\vartheta\left(u_i^{(l)}\right)$,

$$S_\vartheta\left(u_i^{(l)}\right) = \begin{cases} 1 & \text{if } u_i^{(l)} \geq \vartheta, \\ 0 & \text{otherwise.} \end{cases} \qquad (11)$$

When the potential in Eq. (10) crosses a threshold for spiking ϑ, a spike is emitted. Subsequently, the potential is reset to zero.

We trained the SNN classifier using the spatio-temporal backpropagation (STBP) algorithm based on surrogate gradient [21]. However, for simplicity, we modified STBP such that the gradient $\partial S_\varepsilon/\partial u_i^{(l)}$ is replaced by a boxcar function B as follows:

$$\frac{\partial S_\vartheta}{\partial u_i^{(l)}} \leftarrow B\left(u_i^{(l)}\right) = \begin{cases} 1 & \text{if } |u_i^{(l)} - \vartheta| < a, \\ 0 & \text{otherwise,} \end{cases} \qquad (12)$$

where a is a positive constant.

IV. EXPERIMENTS

We evaluated the performance of DTS-SNNs on three real-world datasets, DVS128 Gesture, SHD datasets, N-Cars. For all datasets, we reduced the input event sampling rate to reduce the computational complexity, which is equivalent to the reciprocal timestep Δt^{-1}. The hyper-parameters used for each dataset are listed in Appendix E, which were found using manual searches. We used the raw event datasets without any pre-processing.

To identify the effect of the zero-sum temporal kernel k_{tzs} on performance, we compared the classification accuracy for the zero-sum temporal kernel k_{tzs} with that for the single-exponential kernel k_t. We evaluated the computational efficiency of DTS-SNN in terms of the number of OPs per timestep. Note that the number of OPs includes $\#\text{OP}_{\text{DTS}}$ in Eq. (9). All experiments were conducted on a GPU workstation (GPU: RTX 2080 TI). DTS-SNNs were implemented in Python using the Pytorch's Autograd framework [22]. We trained the networks using Adam [23] without weight decay and learning rate scheduling.

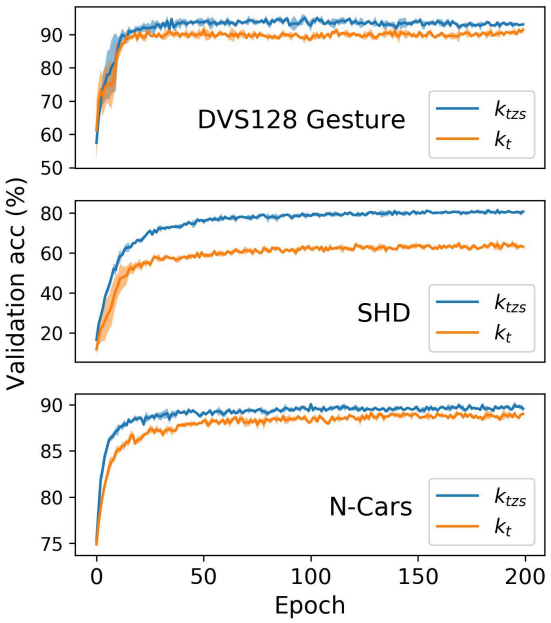

FIGURE 5. Validation accuracy change with training epoch.

A. DVS128 GESTURE

DVS128 Gesture is an event-based hand-gesture dataset, which comprises 1,342 samples labeled as 11 classes. We set the input sampling time Δt and the number of timesteps to 5 ms and 300, respectively. Each original sample ($H = W = 128$) was mapped onto an 8×8 grid, i.e., $h_c = w_c = 16$. The size of each time-surface was set to 7×7, i.e., $R_x = R_y = 7$. The flattened DTS was thus 3136 in length and fed into a 3136-400-11 SNN classifier.

Table 1 shows the performance and efficiency of DTS-SNN on DVS128 Gesture in comparison with previous methods using CNN-based SNNs. It highlights (i) high classification accuracy (1.51% lower than the state-of-the-art result though) and (ii) extremely high computational efficiency ($\times 18$ of that of the state-of-the-art result). The high computational efficiency arises from the use of a small FCN instead of a CNN and the high efficiency of the DTS builder. The evolution of test accuracy with training epoch is plotted in Figure 5.

Additionally, we achieved a 3.12% accuracy improvement by using the zero-sum temporal kernel k_{tzs} instead of the single-exponential kernel k_t, which indicates the higher temporal responsiveness of the zero-sum temporal kernel than the conventional single-exponent temporal kernel. We visualize the DTSs $\overline{\mathcal{T}}_c$ of the two temporal kernels at five timesteps (200 – 240) in Figure 6. A detailed comparison between the two time-surfaces is addressed in Appendix B.

TABLE 1. Performance comparisons of DTS-SNN on DVS128 Gesture, SHD, and N-Cars.

Method	Architecture	Accuracy	# Params	# OPs	# Timesteps
		DVS128 Gesture			
SLAYER [24]	8-layer CNN	93.64±0.49	-	-	300
SCRNN [18]	3-layer CNN	92.01	-	-	20
DECOLLE [19]	5-layer CNN	95.54±0.16	1.6M	340M	500/1800
Streaming rollout SNN [2]	DenseNet	95.56±0.14	0.8M	15G	-
STBP [20]	4-layer CNN	93.4	2.3M	230M	60
STBP-tdBN [1]	ResNet-17	96.87	1.5M	258M	40
PLIF [25]	7-layer CNN	97.57	1.7M	1.6G	20
This work (with k_{tzs})	1-layer FCN	**96.06**±0.33	**1.3M**	**5.7M**	300
This work (with k_t)	(3136-400-11)	**92.94**±0.43			
		SHD			
Surrogate gradient BP [17]	1-layer FCN	48.1±1.6	92K	184K	2000
Surrogate gradient BP [17]	2-layer FCN	48.6±0.9	108K	217K	2000
Surrogate gradient BP [17]	3-layer FCN	47.5±2.3	125K	250K	2000
Surrogate gradient BP [17]	1-layer RSNN	71.4±1.9	108K	217K	2000
Surrogate gradient BP [26]	1-layer RSNN	82.2	250K	500K	500
Surrogate gradient BP [27]	1-layer RSNN	82.7±0.8	108K	217K	2000
Surrogate gradient BP [3]	2-layer RSNN	90.4	141K	283K	250
This work (with k_{tzs})	1-layer FCN	**82.17**±0.17	**16K**	**36K**	500
This work (with k_t)	(105-128-20)	**66.21**±0.63			
		N-Cars			
Gabor-SNN [13]	2-layer	78.9	7K	119M	-
HATS [13]	-	90.2	-	115M	-
Streaming rollout SNN [2]	DenseNet	94.07±0.05	0.1M	1.4G	165
CarSNN [28]	4-layer CNN	86.3	0.8M	5.4M	10
This work (with k_{tzs})	1-layer FCN	**90.28**±0.06	**1.2M**	**3.6M**	100
This work (with k_t)	(3000-400-2)	**89.47**±0.19			

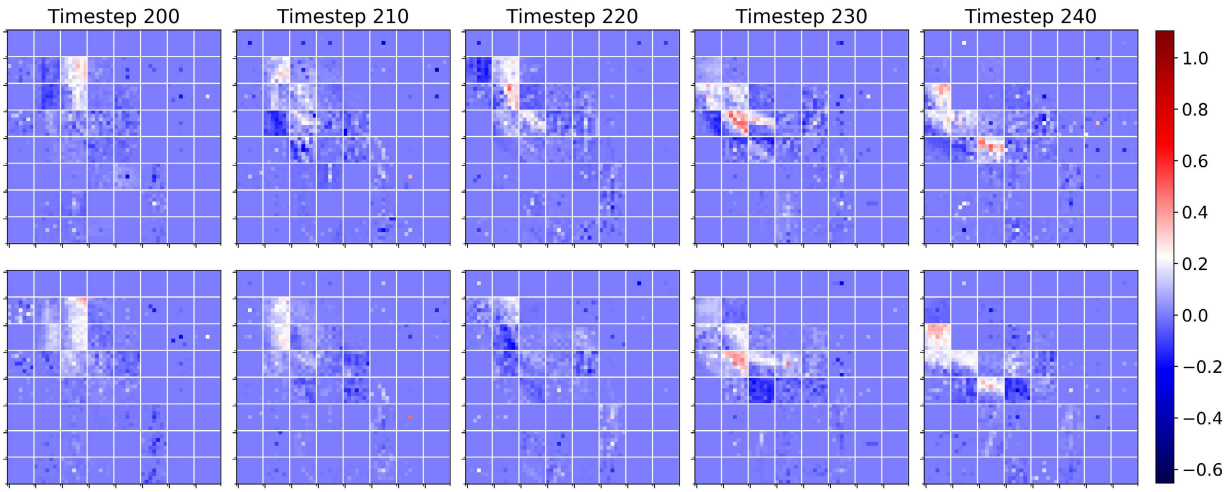

FIGURE 6. DTSs for (upper panel) the zero-sum temporal kernel and (lower panel) the single-exponential kernel. Each grid-cell in an 8 × 8 grid is a 7 × 7 DTS. (Right hand clockwise sample from DVS128 Gesture).

B. SHD

SHD is an audio classification dataset. It consists of 10,420 samples of spoken digits (0 – 9) in English and German, and thus labeled as 20 classes. The recorded samples were analyzed using 700 channels as bases which determine the input data dimension. Each sample varies in length (0.24 – 1.17 s). We set the input sampling time Δt to 1 ms. The other hyper-parameters are listed in Table 4. We considered the original 700-long 1D sample at a given timestep as a 2D sample ($H = 1$, $W = 700$) and mapped it onto a 1×35 grid,

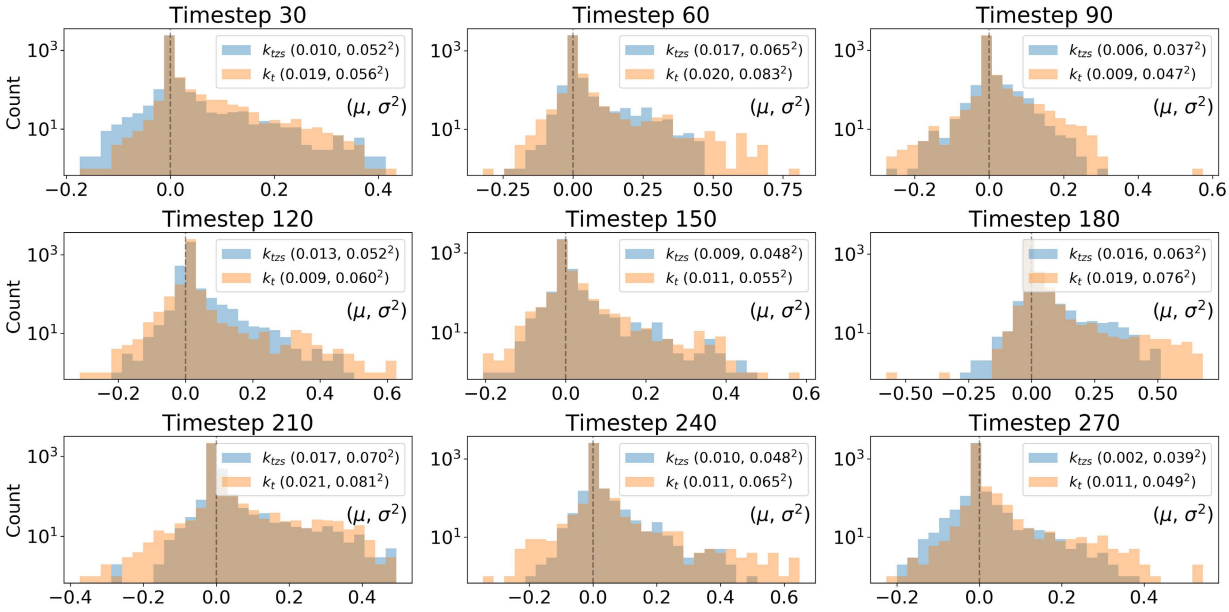

FIGURE 7. Distribution of timestamp encoding values for the zero-sum temporal kernel k_{tzs} and single-exponential kernel k_t (Right hand clockwise sample from DVS128 Gesture).

i.e., $h_c = 1$ and $w_c = 20$. The size of each time-surface was set to 1×3, i.e., $R_x = 1$ and $R_y = 3$. The 105-long flattened DTS was fed into a 105-128-20 SNN classifier.

Table 1 shows the performance and efficiency on SHD compared with previous works. Note that all models except [26] on SHD in Table 1 have 128 neurons in each hidden layer. The use of the zero-sum temporal kernel realized a significant improvement in classification accuracy (by 15.96%) compared with the single-exponential kernel. The learning kinetics for both cases is plotted in Figure 5.

C. N-CARS

N-Cars is an event-based dataset that was directly recorded using an event camera for car detection task. This dataset aims to binary classification task (car or background) with event data of static objects (rather than dynamic objects as for DVS128 Gesture and SHD). Each sample is 100 ms long, and we set the input sampling time Δt to 1 ms so that the number of timesteps was 100.

We mapped each sample at a given timestep ($H = 100$, $W = 120$) onto a 10×12 grid, i.e., $h_c = w_c = 10$. We used the same size of time-surfaces as for DVS128 Gesture and SHD, i.e., $R_x = R_y = 5$. The 3,000-long flattened DTS was fed into a 3000-400-2 SNN classifier. The results are shown in Table 1 and compared with several state-of-the-art methods. The results indicate an accuracy improvement by 0.81% by using the zero-sum temporal kernel instead of the single-exponential kernel. The evolution of test accuracy with training epoch is plotted in Figure 5.

V. DISCUSSION

The zero-sum temporal kernel k_{tzs} avoids large timestamp encoding values caused by persistent events so that it enhances the responsiveness to time-varying events. To show this, we address the distribution of timestamp encoding values on a $H/h_c \times W/w_c \times (R_x \times R_y)$ DTS map $\overline{\mathcal{T}}_c(t)$ at given timesteps. We plot the distribution for the zero-sum temporal kernel k_{tzs} and single-exponential kernel k_t at given timesteps in Figure 7. We used a sample from DVS128 Gesture. The comparison evidently indicates the larger dispersion of encoding values for the single-exponential kernel, and thus the larger standard deviation than the zero-sum temporal kernel. The larger encoding values for the single-exponential kernel are likely attributed to persistent events. The zero-sum temporal kernel filters out such large encoding values and consequently allows the SNN classifier to pay attention to time-varying events.

TABLE 2. Classification (best) accuracy for different time-surface sizes ($R \times R$ for DVS128 Gesture and N-Cars, and R for SHD) and grid-cell sizes ($C \times C$ for DVS128 Gesture and N-Cars, and C for SHD).

Grid-cell size C	Time-surface size R = 3	5	7
DVS128 Gesture			
8	91.32	91.67	91.32
16	93.4	95.49	96.53
32	90.97	93.05	92.36
SHD			
10	58.69	75.98	80.61
20	82.03	54.10	81.25
N-Cars			
10	89.42	90.35	90.06
20	87.55	89.13	90.33

The dimensions of each time-surface and grid-cell are important hyper-parameters, which are given by ($R \times R$ and $C \times C$) for 2D data and (R and C) for 1D data. The dependency of classification accuracy on these hyper-parameters is shown in Table 2. The larger the value R, the further

events are considered to build time-surfaces, capturing the spatio-temporal correlation of likely incoherent long-range events. The larger the value C, the further time-surfaces are considered to build a single representative time-surface per grid-cell. Table 2 highlights the presence of optimal dimensions of time-surfaces and grid-cells which may optimally take into account coherent events by filtering out incoherent long-range events. We chose the optimal values R and C for each dataset with reference to the data in Table 2.

VI. CONCLUSION

We proposed DTS-SNN that merges the time-surface analysis and event-based classification to realize lightweight inference models with high classification capability. The DTS builder with the conventional single-exponential temporal kernel successfully extracted spatio-temporal features of event data, allowing the following simple SNN classifier (1-layer FCN) to classify input data at high precision. To enhance the feature extraction, we introduced the zero-sum temporal kernel, which was motivated by the temporal structure of biological receptive fields. With the zero-sum kernel, we improved the classification accuracy further. The accuracy improvement is deemed to be due to the zero-sum temporal kernel endowing the DTS builder with high responsiveness to dynamic objects. Nevertheless, systematic studies on the role of the zero-sum temporal kernel on accuracy improvements need to be conducted, which are left for future work.

APPENDIX

A. PROOFS OF LEMMAS

Lemma 2: Consider the convolution of a train of Poisson spikes ρ at a firing rate r,

$$y(t) = (k * \rho(t))(t),$$

where k is a single exponential kernel $k = e^{-t/\tau}$. The expected value $\bar{y}$ is approximated to the convolution of the firing rate r using the same kernel,

$$\bar{y}(t) = (k * r(t))(t).$$

Proof: The probability of a particular pattern of probabilistic spikes in a period is calculated using the probability of spiking $P_s(t)$ and not spiking $1-P_s(t)$. Consider a pattern of spikes at timesteps T_s and no spikes at timesteps T_{ns}. The probability of the pattern is given by

$$P = \prod_{t\in T_s} P_s(t) \prod_{t\in T_{ns}} (1 - P_s(t)). \tag{13}$$

The probability of spiking P_s is the product of spiking rate r and timestep size Δt, i.e., $P_s = a\Delta t$. Generally, spiking rate is below 50 Hz, and setting Δt to 1 ms is commonplace. Even for $r = 50$ Hz and $\Delta t = 1$ ms, the spiking probability P_s is 0.05. Therefore, ignoring nonlinear terms in Eq. (13) is a reasonable approximation. Considering this approximation, the nontrivial cases include only one spike in the entire period T, Thus, there are T nontrivial cases whose probabilities are given by

$$\forall k \in T, P_k = P_s(k) \prod_{t\neq k} (1 - P_s(t)) \approx P_s(k).$$

The expected value $\bar{y}$ at timestep t can be calculated considering the nontrivial cases only.

$$\bar{y}(t) = \sum_{k=1}^{t} e^{-(t-k)\Delta t/\tau} P_s(k).$$

Because $P_s(k) = r(k)\Delta t$, we have

$$\bar{y}(t) = \sum_{k=1}^{t} e^{-(t-k)\Delta t/\tau} r(k)\Delta t. \tag{14}$$

Eq. (14) is the discrete form of the convolution:

$$\bar{y}(t) = (k * r(t))(t).$$

□

Lemma 3: Consider the convolution of a train of Poisson spikes ρ at a constant firing rate r using the zero-sum temporal kernel k_{tzs}, $y(t) = (k_{tzs} * \rho)(t)$. The result converges to zero as time t increases, i.e., $y(\infty) = 0$.

Proof: We first divide the zero-sum temporal kernel k_{tzs} into two sub-kernels $k_{tzs}^{(1)}$ and $k_{tzs}^{(2)}$.

$$\begin{aligned} k_{tzs} &= k_{tzs}^{(1)} - \frac{\tau_1}{\tau_2} k_{tzs}^{(2)}, \\ k_{tzs}^{(1)} &= e^{-\tau/\tau_1}, \\ k_{tzs}^{(2)} &= e^{-\tau/\tau_2}. \end{aligned}$$

Using **Lemma 2**, we have

$$\begin{aligned} \bar{y}(t) &= \bar{y}^{(1)}(t) - \frac{\tau_1}{\tau_2}\bar{y}^{(2)}(t), \\ \bar{y}^{(i)}(t) &= \left(k_{tzs}^{(i)} * r(t)\right)(t), i \in \{1, 2\}. \end{aligned} \tag{15}$$

We consider a Poisson-spike train whose firing rate r is given by a boxcar function with constant nonzero firing rate r_0 in the range $t_0 < t < t_1$.

$$r(t) = r_0 (H(t - t_0) - H(t - t_1)),$$

where H is the Heaviside step function. Consequently, we have the result of the convolution in Eq. (15) as follows.

$$\begin{aligned} \bar{y}(t) &= 0 \quad \text{if } 0 \le t < t_0, \\ \bar{y}(t) &= r_0\tau_1 \left[e^{-(t-t_0)/\tau_2} - e^{-(t-t_0)/\tau_1}\right] \quad \text{if } t_0 \le t < t_1, \\ \bar{y}(t) &= r_0\tau_1 \left[e^{-(t-t_0)/\tau_2} - e^{-(t-t_0)/\tau_1}\right] \\ &\quad -r_0\tau_1 \left[e^{-(t-t_1)/\tau_2} - e^{-(t-t_1)/\tau_1}\right] \quad \text{if } t \ge t_1. \end{aligned} \tag{16}$$

Eq. (16) indicates that $\bar{y}(t)$ converges to zero if $t_0 \ll t < t_1$. □

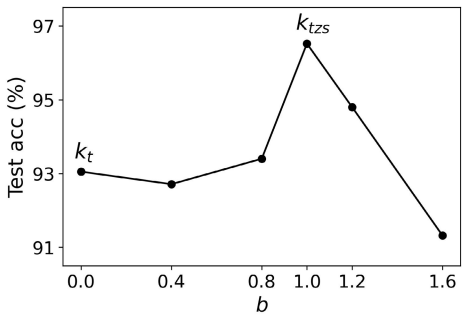

FIGURE 8. Effect of nonzo-sum temporal kernel on classification accuracy on DVS128 Gesture. The time-constants τ_1 and τ_2 were set to 50 ms and 100 ms, respectively.

B. NONZERO-SUM TEMPORAL KERNEL

The use of the zero-sum temporal kernel instead of the single-exponential kernel improves classification accuracy on all three datasets considered in this study. We evaluate the effect of nonzero-sum temporal kernel on classification accuracy in depth by introducing a temporal kernel k_{t0},

$$k_{t0} = e^{-\tau/\tau_1} - b\frac{\tau_1}{\tau_2}e^{-\tau/\tau_2}, \tag{17}$$

where b is a non-negative constant that determines the contribution of a single event to timestamp encoding such that

$$k_{t0} = \begin{cases} k_t & \text{if } b = 0, \\ \text{positive-sum temporal kernel} & \text{if } 0 < b < 1, \\ k_{tzs} & \text{if } b = 1, \\ \text{negative-sum temporal kernel} & \text{if } b > 1. \end{cases} \tag{18}$$

We evaluated the classification accuracy on DVS128 Gesture with respect to b (Figure 8). The results indicate the highest accuracy achieved at $b = 1$, i.e., $k_{t0} = k_{tzs}$.

C. ZERO-SUM TEMPORAL KERNELS WITH DIFFERENT TIME-CONSTANTS

We manually explored time-constant space in search of the optimal set of time-constants τ_1 and τ_2 for the zero-sum temporal kernel ($\tau_2 = 2\tau_1$). Here, we report the classification accuracy on DVS128 Gesture, SHD, and N-Cars for three sets of time-constants (τ_1/τ_2 = 10/20, 20/40, and 50/100) in Table 3.

TABLE 3. Classification accuracy for three different sets of time-constants τ_1 and τ_2. This is the best validation accuracy from a single trial for each case.

Time-constants	Accuracy		
$\tau_1/\tau_2(ms)$	DVS128 Gesture	SHD	N-Cars
10/20	95.14	76.37	90.35
20/40	95.49	82.47	90.04
50/100	96.53	53.47	89.78

D. PSEUDOCODE

The pseudocode for constructing dynamic time-surfaces in parallel is shown in Algorithm 1.

Algorithm 1: Building Dynamic Time-Surfaces. Functions encTstamp and Unfold Are Given by Eqs. 7 and 8. Function Mul Element-Wise Multiplies $\mathbf{T}$ by the Event-Map Built Using $\mathbf{e} = \{e_i | t_i = t\}$. Function conv3d Executes the 3D Convolution of $\mathcal{T}$ Using the Rank-4 Kernel $\mathbf{a}$

Input: Set of events $\mathbf{e} = \{e_i\}_{i=1}^{N}$ for a sample
Output: Dynamic time-surfaces $\overline{\mathcal{T}}_c(t)$ at every timestep t for all grid-cells $\mathbf{C} = \{c_i\}_{i=1}^{M}$
Initialization: $\mathbf{E} \leftarrow \mathbf{0}$
for $t = 1$ **to** T **do**
  /* Update of timestamp-encoding bank $\mathbf{E}$ */
  $\mathbf{E} \leftarrow \text{encTstamp}(\mathbf{E}_T, \mathbf{e} = \{e_i | t_i = t\})$
  /* Building preliminary time-surface map $\mathbf{T}$ */
  $\mathbf{T} \leftarrow \text{unfold}(\mathbf{E})$
  /* Flattening/reshaping $\mathbf{T}$ to $P \times [R_x R_y] \times H \times W$ */
  $\mathbf{T} \leftarrow \text{flatten/reshape}(\mathbf{T})$
  /* Building $\mathcal{T}$ using the event-map */
  $\mathcal{T} \leftarrow \text{Mul}(\mathbf{T}, \mathbf{e} = \{e_i | t_i = t\})$
  $\mathcal{T} \leftarrow \text{Normalize}(\mathcal{T})$
  /* 3D convolution using rank-4 kernel $\mathbf{a}$ */
  $\overline{\mathcal{T}}_c \leftarrow \text{conv3d}(\mathcal{T}, \mathbf{a})$
  /* Reshaping $\overline{\mathcal{T}}_c$ to $H/h_c \times W/w_c \times (R_x \times R_y)$ */
  $\overline{\mathcal{T}}_c \leftarrow \text{Reshape}\left(\overline{\mathcal{T}}_c\right)$
  Output $\overline{\mathcal{T}}_c$
end

TABLE 4. Hyper-parameters used for evaluation.

Parameter	DVS128 Gesture	SHD	N-Cars
Learner			
u_{th}/a in Eq. (12)	0.05/0.05		
Learning rate	0.001		
Zero-sum temporal kernel in Eq. (3)			
τ_1/τ_2 (ms)	50/100	20/40	10/20
Single-exponential kernel in Eq. (4)			
τ_0 (ms)	50	20	10
SRM in Eq. (10)			
τ_m/τ_s (ms)	60/20	40/20	12/4
General setting			
Batch size	16	256	64
Δt (ms)	5	1	1

E. HYPER-PARAMETERS

The hyper-parmeters used for evaluation are shown in Table 4.

REFERENCES

[1] H. Zheng, Y. Wu, L. Deng, Y. Hu, and G. Li, "Going deeper with directly-trained larger spiking neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 11062–11070.
[2] A. Kugele, T. Pfeil, M. Pfeiffer, and E. Chicca, "Efficient processing of spatio-temporal data streams with spiking neural networks," *Frontiers Neurosci.*, vol. 14, p. 439, May 2020.
[3] B. Yin, F. Corradi, and S. M. Bohté, "Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks," *Nature Mach. Intell.*, vol. 3, no. 10, pp. 905–913, Oct. 2021.
[4] D. S. Jeong, "Tutorial: Neuromorphic spiking neural networks for temporal learning," *J. Appl. Phys.*, vol. 124, no. 15, Oct. 2018, Art. no. 152002.
[5] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: Opportunities and challenges," *Frontiers Neurosci.*, vol. 12, p. 774, Oct. 2018.
[6] M. Jaderberg, W. M. Czarnecki, S. Osindero, O. Vinyals, A. Graves, D. Silver, and K. Kavukcuoglu, "Decoupled neural interfaces using synthetic gradients," in *Proc. Int. Conf. Mach. Learn.*, 2017, p. 1627–1635.
[7] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, Aug. 2014.
[8] M. Davies, N. Srinivasa, T. H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, and Y. Liao, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan. 2018.
[9] A. Neckar, S. Fok, B. V. Benjamin, T. C. Stewart, N. N. Oza, A. R. Voelker, C. Eliasmith, R. Manohar, and K. Boahen, "Braindrop: A mixed-signal neuromorphic architecture with a dynamical systems-based programming model," *Proc. IEEE*, vol. 107, no. 1, pp. 144–164, Jan. 2019.
[10] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri, "A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs)," *IEEE Trans. Biomed. Circuits Syst.*, vol. 12, no. 1, pp. 106–122, Feb. 2018.
[11] V. Kornijcuk and D. S. Jeong, "Recent progress in real-time adaptable digital neuromorphic hardware," *Adv. Intell. Syst.*, vol. 1, no. 6, Oct. 2019, Art. no. 1900030.
[12] 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.
[13] A. Sironi, M. Brambilla, N. Bourdis, X. Lagorce, and R. Benosman, "HATS: Histograms of averaged time surfaces for robust event-based object classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1731–1740.
[14] J. Manderscheid, A. Sironi, N. Bourdis, D. Migliore, and V. Lepetit, "Speed invariant time surface for learning to detect corner points with event-based cameras," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 10237–10246.
[15] P. Dayan and L. Abbott, *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. Cambridge, MA, USA: MIT Press, 2005.
[16] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza, J. Kusnitz, M. Debole, S. Esser, T. Delbruck, M. Flickner, and D. Modha, "A low power, fully event-based gesture recognition system," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7243–7252.
[17] B. Cramer, Y. Stradmann, J. Schemmel, and F. Zenke, "The Heidelberg spiking data sets for the systematic evaluation of spiking neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 7, pp. 2744–2757, Jul. 2022.
[18] Y. Xing, G. D. Caterina, and J. Soraghan, "A new spiking convolutional recurrent neural network (SCRNN) with applications to event-based hand gesture recognition," *Frontiers Neurosci.*, vol. 14, p. 1143, Nov. 2020.
[19] J. Kaiser, H. Mostafa, and E. Neftci, "Synaptic plasticity dynamics for deep continuous local learning (DECOLLE)," *Frontiers Neurosci.*, vol. 14, p. 424, May 2020.
[20] W. He, Y. Wu, L. Deng, G. Li, H. Wang, Y. Tian, W. Ding, W. Wang, and Y. Xie, "Comparing SNNs and RNNs on neuromorphic vision datasets: Similarities and differences," *Neural Netw.*, vol. 132, pp. 108–120, Dec. 2020.
[21] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *Frontiers Neurosci.*, vol. 12, p. 331, May 2018.
[22] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, and A. Desmaison, "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8026–8037.
[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[24] S. B. Shrestha and G. Orchard, "Slayer: Spike layer error reassignment in time," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1412–1421.
[25] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, and Y. Tian, "Incorporating learnable membrane time constant to enhance learning of spiking neural networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 2661–2671.
[26] F. Zenke and T. P. Vogels, "The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks," *Neural Comput.*, vol. 33, no. 4, pp. 899–925, 2021.
[27] N. Perez-Nieves, V. C. H. Leung, P. L. Dragotti, and D. F. M. Goodman, "Neural heterogeneity promotes robust learning," *Nature Commun.*, vol. 12, no. 1, pp. 1–9, Dec. 2021.
[28] A. Viale, A. Marchisio, M. Martina, G. Masera, and M. Shafique, "CarSNN: An efficient spiking neural network for event-based autonomous cars on the loihi neuromorphic research processor," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2021, pp. 1–10.

DONGHYUNG YOO received the B.S. degree in materials science and engineering from Hanyang University, Seoul, South Korea, in 2017, where he is currently pursuing the Ph.D. degree in materials science and engineering. His current research interests include learning algorithms for spiking neural networks and neuromorphic vision sensor.

DOO SEOK JEONG (Member, IEEE) received the B.E. and M.E. degrees in materials science from Seoul National University, in 2002 and 2005, respectively, and the Ph.D. degree in materials science from RWTH Aachen, Germany, in 2008. He was with the Korea Institute of Science and Technology, from 2008 to 2018. He is an Associate Professor with Hanyang University, South Korea. His research interests include spiking neural networks for sequence learning, future prediction, learning algorithms, spiking neural network design, and digital neuromorphic processor design.