IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 15, 2022, accepted May 16, 2022, date of publication May 25, 2022, date of current version May 31, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3177744

Fast Classification and Action Recognition With Event-Based Imaging

CHANG LIU, XIAOJUAN QI, (Member, IEEE), EDMUND Y. LAM, (Fellow, IEEE), AND NGAI WONG, (Senior Member, IEEE)

Department of Electrical and Electronic Engineering, The University of Hong Kong, Pokfulam, Hong Kong

Corresponding author: Chang Liu (lcon7@connect.hku.hk)

This work was supported in part by the Research Grants Council of Hong Kong [General Research Fund (GRF)] under Project 17201620 and Project 17209721.

ABSTRACT The neuromorphic event cameras, which capture the optical changes of a scene, have drawn increasing attention due to their high speed and low power consumption. However, the event data are noisy, sparse, and nonuniform in the spatial-temporal domain with an extremely high temporal resolution, making it challenging to process for traditional deep learning algorithms. To enable convolutional neural network models for event vision tasks, most methods encode events into point-cloud or voxel representations, but their performance still has much room for improvement. Additionally, as event cameras can only detect changes in the scene, relative movements can lead to misalignment, i.e., the same pixel may refer to different real-world points at different times. To this end, this work proposes the aligned compressed event tensor (ACE) as a novel event data representation, and a framework called branched event net (BET) for event-based vision under both static and dynamic scenes. We apply them for various object classification and action recognition tasks, and show that they surpass state-of-the-art methods by significant margins. Specifically, our method achieves 98.88% accuracy for the DVS128 action recognition task, and outperforms the second-best method by large margins of 4.85%, 9.56% and 2.33% on N-Caltech101, DVSAction and NeuroIV datasets, respectively. Furthermore, the proposed ACE-BET is efficient, and achieves the fastest inference speed among various methods being tested.

INDEX TERMS Action recognition, classification, event camera, event data representation, neuromorphic imaging, machine learning algorithms.

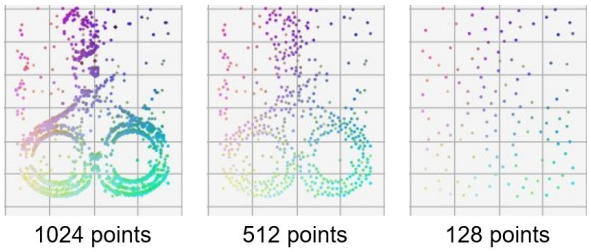
I. INTRODUCTION

Neuromorphic event cameras represent a paradigm change in sensing and imaging [1]. Unlike traditional cameras, which scan the whole frame periodically, event cameras work asynchronously and capture the per-pixel optical changes instead of the absolute brightness. They enjoy various advantages such as low latency (in microsecond level), high dynamic range, low power consumption, and low bandwidth cost [2], [3]. Because of the appealing characteristics of event cameras, event-based vision is gaining more attention in many research areas and applications, such as image reconstruction [4], action recognition [5], auto-driving [6], real-time detection [7], motion estimation [8], [9] and tracking [10].

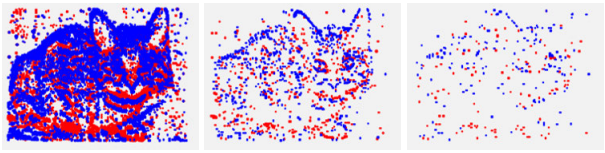
The outputs of event cameras are flows of events defined by (x, y, t, p), for pixel coordinates, timestamps, and polarity. The event flows denote instantaneous increases or decreases of the light intensity in the logarithmic scale. With a very different data structure from spatial images, event data are incompatible with typical convolutional neural networks (CNNs). They are also challenging to process because they do not contain information about the absolute intensity, and are usually rather noisy [11]. Therefore, new algorithms that can efficiently process event data are highly desired [1].

To make use of event cameras for computer vision tasks, some existing approaches take the event data as spatial-temporal point clouds [12], [13], and directly apply point-cloud-based models. However, these methods [13], [14] usually handle small clouds with, for example, 1024 points, and are not efficient for large clouds such as event data with hundreds of thousands of events per second. Moreover, the downsampling algorithms, such as the widely-used farthest point sampling [14], [15], are prone to noise, as shown in Fig. 1a. Consequently, point-cloud-based methods have limitations in accuracy and efficiency for event-based vision.

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



(a) Downsampling event data with the Furthest Point Sampling algorithm. The outlier noises are kept during downsampling and the shape collapses.



(b) Events accumulated for 1/50, 1/100 and 1/500 total time. Proper duration of accumulation is needed for a clear image.

FIGURE 1. Visualization of event data samples.

Additionally, in Fig. 1b, we can observe that a proper duration for accumulating the events is essential for good visual results, while point-cloud-based methods cannot make good use of this property, treating the whole recording as one cloud.

Alternatively, voxel-based approaches compress the event data into voxels, bridging event cameras with conventional CNN techniques [16]–[19]. They achieve state-of-the-art performance on several event-based datasets [16], [19], partially benefiting from the successful CNN models. They also enjoy higher efficiency than point-cloud-based methods, due to lower complexity and good regularity for parallel computing on GPUs. Nonetheless, the voxel-based methods have to compress the temporal resolution to meet the memory and computational requirements, and hence suffer from information loss. According to the observation in our early-stage experiments, many voxel-based methods suffer from problems such as motion blur, when the temporal resolution is compressed significantly (e.g., 9 frames in Reference [16]). The number of frames is treated as a hyper-parameter [16], [19]. However, searching for the optimal setting can be tedious from one dataset to another. Moreover, the backbone CNN networks used in the previous studies are originally designed for 2D image data, having an obvious gap with the event data.

Another reason causing motion blur for voxel-based methods is the *misalignment problem*. For event camera recordings, there are always relative motions between the object and the camera. Therefore, events on the same pixel from different timestamps can refer to different real-world points. Previous works [16], [19] only apply pixel-wise operation (i.e., MLP and LSTM), hence cannot compensate the motions and reduce blur. Reference [4] proposes a global motion compensation with time-varying motion parameters, but only for frame reconstruction tasks.

To this end, we propose an efficient representation, called the aligned compressed event tensor (ACE), for event data. While inheriting the advantage of voxel-based methods in computational efficiency, ACE can also alleviate the above problems of low temporal resolution and blurring by accumulating event history and compressing in local spatial-temporal regions with stride 3D convolution. In ACE, the event data are quantized and voxelized into a fine-grid structure with a high temporal resolution (e.g., 100 frames per second). The voxels are then compressed into a few key frames, in order to reduce computational cost.

This work mainly focuses on object classification and action recognition tasks with event cameras. For static tasks such as object classification, every instant can provide enough information, while for dynamic tasks such as action recognition, the prediction must be made by observing a time period. Noticing the different nature of the tasks, we propose a two-branch neural network design, namely the branched event net (BET), where decisions are made on two levels: frame-level and video-level.

According to the ablation study results in Section V-E, both ACE and BET can significantly increase the prediction accuracy for event-based vision. They are also time efficient, and enjoy the fastest inference speed among competing schemes. In summary, our main contributions are:

- We present the ACE representation for event data, which is lightweight and proved to boost accuracy efficiently.
- We propose BET, which, to the best of our knowledge, is the first attempt to handle both static and dynamic tasks under a single architecture.
- We evaluate our method on 7 classification and action recognition tasks. Our method significantly improves the accuracy while also having the fastest inference.

II. RELATED WORKS

In this section, previous works on event-based vision are introduced. We categorize most of the existing ANN (artificial neural network) methods into three types: reconstruction-based, point-cloud-based, and voxel-based methods. Then, we briefly introduce spiking neural networks (SNNs).

A. RECONSTRUCTION-BASED METHODS

Several studies such as E2VID [20], EventSR [21] and HDN [22] reconstruct events back to normal images, and then the generated images can be processed similar to conventional vision tasks. The reconstruction methods usually utilize recursive modules such as ConvLSTM [23] with encoder-decoder structures such as UNet [24] to build images based on the events [20], [25]. However, the massive pre-training workload is costly, and the cumbersome two-stage framework is not suitable for real-time applications. It is also challenging to obtain simultaneous ground-truth frames in the real world for training the reconstruction. Simulators such as ESIM [26] are used to generate events from normal videos [20], while Reference [11] uses GAN to generate reconstructed frames, but still with some gaps with real data. While reconstructive methods have the drawbacks mentioned

above, end-to-end models follow a more straightforward workflow and are more widely studied for downstream tasks such as classification [16], [19]. In this research, we will focus on end-to-end designs.

B. POINT-CLOUD-BASED METHODS

Among the end-to-end methods, some researchers tend to consider event data as 3D point clouds in the spatial-temporal domain, and directly apply point-cloud networks. Recalling that the event data is denoted by (x, y, t, p), they treat the time stamp as another coordinate, so the events become a point cloud with (x, y, t) as coordinates and p as point features. Reference [27] adopts the PointNet [13] and PointNet++ [15] to extract local and global features from event point clouds, with a rolling buffer framework to decrease the input size. On the other hand, Reference [12] designs a group shuffle attention module to encode the event clouds, while using the gumbel sampling method to downsample the point clouds during processing. They apply their networks for event-based gesture recognition on the DVS128 dataset [28], demonstrating state-of-the-art accuracy at that time.

However, most of the point-cloud-based methods downsample point clouds with farthest point sampling, K-nearest neighbors and ball query algorithms, which are prone to noise and not efficient for large-size event clouds due to the $\mathcal{O}(n^2)$ complexity, where n is the cloud size. Therefore, they are less commonly used for event-based vision compared with voxel-based methods.

C. VOXEL-BASED METHODS

Voxel-based methods transform the event data into fine-grid representations, which are compatible with conventional CNN techniques. The representations are either end-to-end trainable or manually designed based on heuristics. Two pioneering works [17], [18] present the time-surface concept to convert the sparse events into a 2D representation. The event spike tensor (EST) [16] assigns each event some weights, generated by a multilayer perceptron (MLP) kernel, and projects the events to different anchor frames.

Later on, Matrix-LSTM [19] uses an encoding process with parallel long short-term memory (LSTM) [29] cells. The history of events that have happened over each pixel is encoded into a vector by the LSTM, hence generating a multi-channel image. It improves the prediction accuracy but has a slower speed than EST. Similarly, Event-LSTM [30] also adopts LSTM cells for encoding local history.

However, for dynamic scenarios, such as human activity recognition, the above approaches encounter motion blur problems due to the drastic compression in the compromise of the memory cost (e.g., 30k times compression to 9 frames). The accuracy also drops because of information loss during the compression from spatial-temporal 3D to 2D. In addition, since most events are triggered by moving objects in the real world [31], the pixel-wise operation in EST and Matrix-LSTM may mistakenly compress events from different objects (or parts) into one element, causing a misalignment problem. As pointed out by Reference [5], under certain dynamic scenes, hand-crafted features can result in higher accuracy than deep-learning-based methods.

D. SPIKING NEURAL NETWORKS

SNNs mimic human neurons and focus on the spike timing of neural signals. SNN is considered to have advantages in efficiency and better biological plausibility. Reference [10] employs a voxel-based encoder and an SNN controller to enable robot arm grasping. Nonetheless, these SNNs have simpler architectures than CNNs, and their performance is still limited. Reference [32] proposes a 5-layer deep SNN architecture for action recognition using event cameras. Different from conventional SNNs, PLIF [33] proposed a learnable membrane time parameter to replace the original handcrafted configuration. That improves the original leaky integrate-and-fire strategy for training SNNs. Moreover, She *et al.* [34] proposes the STDP strategy and analytically shows that heterogeneity and skip-layer connection can also improve the performance of SNNs in spatiotemporal learning. Noticing the sparse and non-uniform properties of event data, TA-SNN [35] introduces a temporal-wise attention module to assign different importance scores for event-converted frames. On the DVS128 [28] dataset, it achieves state-of-the-art performance. Nevertheless, for object classification tasks such as N-Caltech101, the SNN approaches show lower accuracy than CNN-based methods.

III. PROBLEM STATEMENT

In this work, we focus on object classification and action recognition tasks with event cameras. Given an event camera recording, the network outputs a single prediction indicating the object class or action type of the sample data. Each input is a set of events:

$$\mathcal{E} = \{(x_1, y_1, t_1, p_1), (x_2, y_2, t_2, p_2), \ldots, (x_n, y_n, t_n, p_n)\},$$

where n denotes the number of events. Each event consists of the (x, y) coordinates on the image plane, a timestamp t and a polarity $p \in \{1, -1\}$, indicating a brightness increase or decrease.

We categorize tasks such as object classification as *static*, and tasks such as action recognition as *dynamic*. In the former, objects are stationary and the event camera is slightly shaken to generate the event flows, where predictions can then be made from every instant. For the latter, the objects are moving and the event camera is nearly stationary. Decisions can only be made by observing a time period that is long enough.

IV. METHODOLOGY

In this section we introduce the ACE, an end-to-end trainable voxel-based representation for event camera, followed by the BET, which is a versatile network for both static and dynamic tasks.

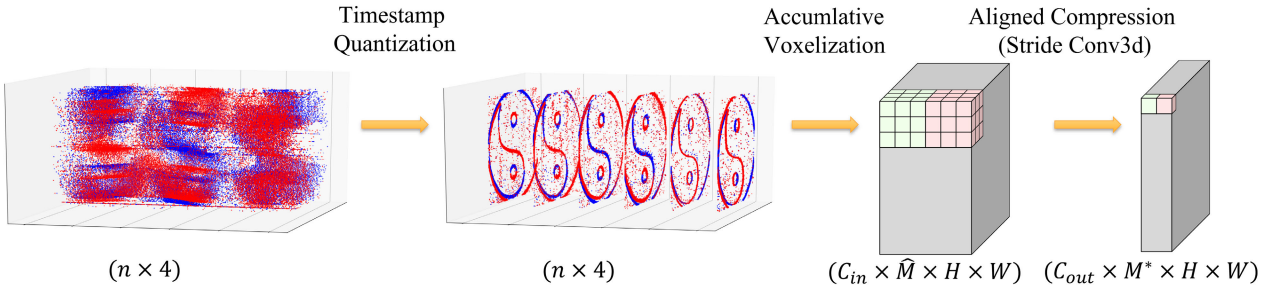


FIGURE 2. ACE workflow. For a better visual effect, $\hat{M} = 6$ and one Conv layer with $G = 3$ is shown in this example. The events are quantized along the time axis, and then accumulatively voxelized (each voxel's value is the sum of all historical events). After that, the frames are compressed by strided 3D convolutional layers along the temporal axis.

A. ALIGNED COMPRESSED EVENT TENSOR

ACE comprises three parts, namely, *timestamp quantization*, *accumulative voxelization*, and *aligned compression*, as shown in Fig. 2.

1) TIMESTAMP QUANTIZATION

Event data are spatial-temporal points with an extremely high temporal resolution [36]. Although theoretically the timestamps are continuous, due to the finite clock rate of hardware and digital storage, in reality they are discrete but with an extremely high resolution. Therefore, event data can also be equivalently represented as a tensor of shape $M \times H \times W$, where M is the number of possible timestamps in one recording, usually in millions depending on the recording length. Due to the large size and high sparsity, it is memory inefficient to store the whole $M \times H \times W$ tensor, and also costly for the following algorithm to process. Existing methods use learnable kernels to reduce the temporal resolution [16], [19], by projecting the events into several anchor frames.

Additionally, reducing temporal resolution can be important under some circumstances [37]. We demonstrate the observation in Fig. 1b, where we accumulate events with different durations. It can be observed that when the duration is too short, there are usually too few events to form a recognizable shape, while if it is too long, there will be blurring. We aim to strike a balance with time stamp quantization.

To reduce the time resolution M, we quantize the timestamp into $\hat{M}$ bins, where $\hat{M}$ is a hyper-parameter. Although the optimal value for $\hat{M}$ could be sample-dependent, to facilitate batch training, a constant value is adopted for simplicity. We empirically set $\hat{M}$ to be 100, and ablation results in Section V-E show that the model is tolerant to different values. All the timestamps $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$ in a sample are quantized into $\hat{M}$ bins. Outputting bin indices from 1 to $\hat{M}$, the quantization function Q is defined as

$$Q(\mathcal{T}^{(i)}) = \max\left(\left\lceil \frac{\hat{M} \times \left(\mathcal{T}^{(i)} - \min(\mathcal{T})\right)}{\max(\mathcal{T}) - \min(\mathcal{T})} \right\rceil, 1\right), \quad (1)$$

where $\lceil \cdot \rceil$ is the ceiling function, and the superscript i means the ith element. The outermost maximum operation forces the bin index to start from 1.

By timestamp quantization, events generated within each duration of $M/\hat{M}$ are grouped together, and the temporal resolution is reduced. However, the resolution $\hat{M}$ is deliberately set much higher than previous works (e.g., 9 in Reference [16] and 16 in Reference [19]), to alleviate information loss and motion blur in this step.

2) ACCUMULATIVE VOXELIZATION

After quantization, the events can be scattered into an $\hat{M} \times H \times W$ tensor, mapped by their coordinates and quantized timestamps. Note that several events may be mapped to the same voxel due to the temporal quantization. Due to the internal intrinsic mechanism of an event camera as shown in Fig. 3, we design the accumulative voxelization step to determine the value of each voxel. Event cameras track the optical intensity in a logarithmic scale for every pixel. When the intensity enters a higher/lower level, a positive/negative event is generated. Therefore, by summing the past positive and negative events, we can determine the intensity level of each pixel with errors smaller than the threshold $\pm C$. Specifically, for voxel at the mth bin, yth row and xth column, we sum the polarities for all past events on each pixel, formulated as:

$$\mathcal{V}(m, y, x) = \sum_{j=1}^{n} \mathbb{I}(x_j = x, y_j = y, \tau_j \leq m) \cdot p_j, \quad (2)$$

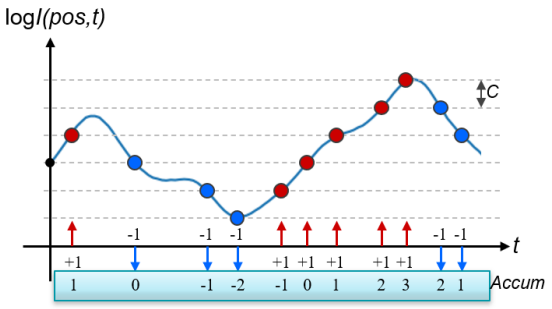


FIGURE 3. Optical intensity curve, with the triggered events marked in red (positive) and blue (negative). The accumulated values are shown in the bottom blue box.

where $\mathcal{V}$ is the $\hat{M} \times H \times W$ voxel representation, $\tau_j = Q(\mathcal{T}^{(j)})$ is the quantized timestamp defined in Eq. 1, $\mathbb{I}$ is the indicator function, and $p_j \in \{-1, +1\}$ is the polarity of the jth event. The tensor form representation $\mathcal{V} \in \mathbb{R}^{(1\times)\hat{M}\times H\times W}$ now implies $\hat{M}$ event frames with $C = 1$ channel each.

The $\hat{M}$ instantaneous frames denote the intensity relative to the initial state. This method encodes the past history of the absolute intensity changes into every frame. Previous methods such as EST [16] and Matrix-LSTM [19] also encode past (and future) histories with trainable kernels. However, they use MLP and LSTM, which are more demanding on computation and memory, and therefore are limited to a small number of frames. Our method, which is simple and attains $\hat{M} = 100$, improves the overall accuracy, as shown in the ablation study in Section V-E.

3) ALIGNED COMPRESSION

During the above two steps, the temporal resolution $\hat{M}$ is kept relatively high (e.g., 100) to avoid information loss. However, it would be burdensome for the neural network to process all $\hat{M}$ frames. Moreover, the contents in adjacent frames could be roughly the same, yet complementary in some details. Therefore, we compress the voxels along the temporal axis by merging nearby frames.

Previous methods such as EST and Matrix-LSTM also compress the event data, but their methods suffer from the misalignment problem described earlier. Instead, we choose strided 3D convolution to aggregate voxels within the local spatial-temporal regions. The trainable convolutional layers can potentially project the voxel at $(t, x + \Delta x, y + \Delta y)$ to (t', x, y) on the target anchor frame, with a spatial offset. Hence, the proposed method performs in spatial-temporal regions rather than just temporal intervals in the pixel-wise methods, which can potentially compensate the movements and project events corresponding to the same real-world object onto the same pixel. While we do not explicitly introduce new loss functions for the alignment, the overall accuracy is shown to improve because of this flexible design.

We use two stacked 3D Conv Layers to gradually compress the $\hat{M}$ frames. As illustrated in Fig. 2, for each step, the 3D Conv Layers merge G frames into one, and compress the tensor from $C_{in} \times \hat{M} \times H \times W$ into $C_{out} \times M^* \times H \times W$, where $M^* = \hat{M}/G$ and C_{in}/C_{out} is the number of input/output channels. In the beginning, $C_{in} = 1$ from Eq. 2. Also, G is a hyperparameter, and we empirically use $G = 10$.

The kernel size of the convolution layers determines the size of the local spatial region. The spatial kernel size is set to be 5×5, and the temporal kernel size is set as 2 and 5, respectively, in the two convolution layers. The temporal strides are 2 and 5, respectively, resulting in every $G = 10$ adjacent frames being grouped and compressed into one frame of the same size. We also examine different settings in Section V-F, and find that our method performs similarly well.

B. BRANCHED EVENT NET

The BET processes the ACE-encoded event data for object classification and action recognition tasks. It is designed to be versatile for both static and dynamic scenes.

1) CNN ENCODER

ACE encodes the event data into $C \times M^* \times H \times W$ tensors, denoting M^* frames with C channels each. For each individual frame, we use an identical CNN backbone model to encode the geometric information into a vector. Following Reference [16] and [19], we use the ImageNet-pretrained ResNet [38] with the last decision-making layer removed.

2) FRAME BRANCH

For static scenes, the frames are similar, and decisions can be made independently from any instance. For example, static objects can be classified using any single frame. Therefore, the frame branch generates predictions based on each independent frame.

As shown at the bottom of Fig. 4, each frame is assigned with an independent classifier with one fully connected layer, for frame-level prediction. The number of classifiers is equal to M^*. For real-time event-based vision, sliding windows can be used [12] to enable a fixed number of classifiers. We use a grouped 1×1 convolutional layer with $g = M^*$ groups for implementation to enable parallel computing. This operation is mathematically equivalent with M^* fully connected layers.

3) VIDEO BRANCH

For tasks such as moving direction recognition, it is necessary to make decisions based on a series of frames. Therefore, we propose the video branch of BET, as illustrated in the upper part of Fig. 4. The vector representations of frame features are concatenated to form a $C' \times M^*$ sequence, where C' is the length of the vector. Conv1d modules then process the sequence to generate the video-level prediction.

Techniques such as LSTM [29] and Transformer [39] are usually considered more powerful for sequential data processing. However, we find that Conv1d can also achieve similar accuracy with much fewer parameters at a faster speed, possibly because the sequence length M^* is short in ACE. Moreover, the tasks of action recognition and object classification do not involve complicated temporal modeling.

4) RESULT SYNTHESIS

To merge multiple predictions from the video branch and the frame branch, a separate synthesis module shown in Fig. 5 is used during the inference.

Intuitively, since BET is designed to handle both static and dynamic tasks, the reliability of classifiers in the frame branch and the video branch may also fluctuate under different scenes. Therefore, we use a statistical method to merge the predictions. We evaluate the trained model on the validation set while recording the per-class accuracy $\boldsymbol{A}(p, q)$ for each classifier p, which records the accuracy when p makes a

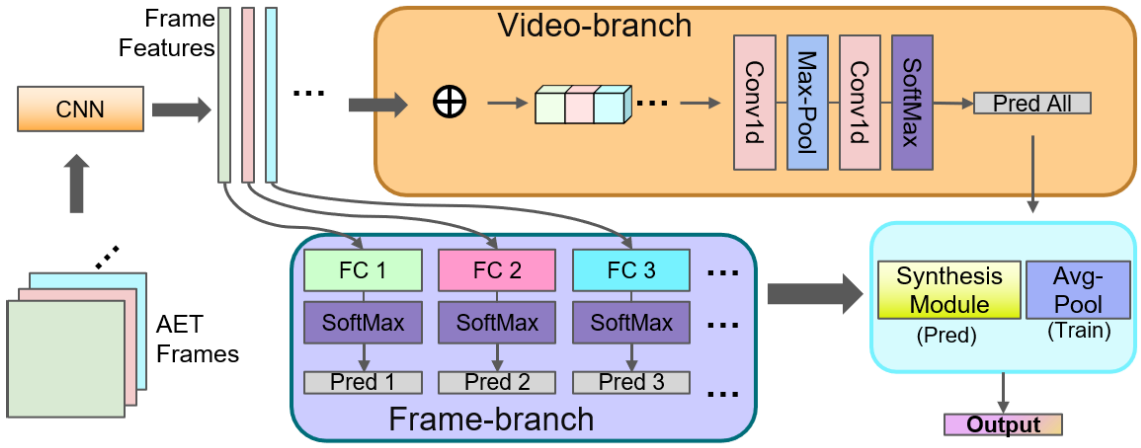


FIGURE 4. The BET workflow. The ACE frames are encoded into vector representations by the backbone CNN, and then processed in two branches. The frame branch makes a prediction for each frame with independent classifiers. The video branch treats the vectors as a sequence and processes it with 1D convolutional layers. During the training, the predictions are averaged as the final output. For inference, a synthesis module described in Fig. 5 and Section IV-B4 is used to merge the results.

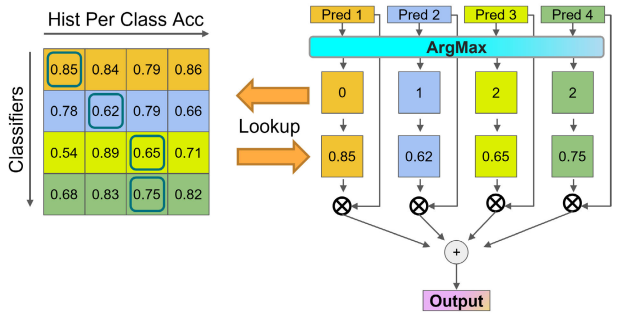


FIGURE 5. The synthesis module. The results are merged by a weighted sum, where the weights are derived from each classifier's performance on the validation set.

prediction being q. Then, the per-class accuracy matrix $\boldsymbol{A}$ is built, as shown on the left of Fig. 5.

Specifically, for the ith classifier, we calculate the predicted class v_i from its prediction vector $\boldsymbol{l}_i$, formulated as

$$v_i = \arg\max_k \boldsymbol{l}_i^{(k)},$$

where $\boldsymbol{l}_i$ is a vector of length s containing the probabilities of each class. The synthesized output is then given by a weighted sum of the predictions from M^* frame classifiers and one video classifier, where the per-class accuracies are taken as weights, given by

$$\boldsymbol{l}_{syn} = \sum_{i=1}^{M^*+1} \boldsymbol{A}(i, v_i) \cdot \boldsymbol{l}_i.$$

5) TRAINING

As BET is a versatile architecture, the frame branch and video branch are always open under both static and dynamic scenes, during training and testing. The whole workflow of ACE + BET is trained end-to-end simultaneously, supervised by ground-truth object class or action type. Cross entropy loss is used as the loss function.

During the training and validation, predictions of frame branch and video branch are directly averaged as the final output. After that, the per-class accuracy matrix is built, and the synthesis module from Section IV-B4 is then used for testing.

V. EXPERIMENTS

We conduct detailed experiments to evaluate our method's overall performance and efficiency. Ablation studies are carried out to quantify the contribution of each module. The results show that our method consistently surpasses all the baselines for both static and dynamic tasks.

A. DATASETS AND PREPROCESSING

We evaluate our approach on different datasets for static and dynamic tasks, whose statistics are shown in Table 1. The N-Caltech101 [40] and N-Cars [18] are benchmark datasets used in many previous studies [16], [18], [19]. They contain static daily objects and cars. Besides, recently Reference [41] proposed a larger static dataset N-Caltech256 with more classes, which is more challenging.

For dynamic scenes, the DVS128 dataset [28] includes human gestures under different illumination environments, which has been adopted in Reference [12]. Another challenging dynamic dataset, the UCF-50, is also newly proposed by Reference [41], which contains various sports actions. Moreover, a large-scale gesture recognition dataset named NeuroIV [43] contains 16 types of human gestures while driving. Another action recognition dataset of DVSAction [42] is also covered in this section. Nonetheless, both DVSAction[1]

[1] https://github.com/CrystalMiaoshu/PAFBenchmark

TABLE 1. Statistics of the datasets used in the experiments.

Dataset	Type	#classes	Total samples	Avg. events (kEv)	Avg. length (s)	Avg kEv/s
N-Caltech101 [40]	Static	101	8709	112.76	0.30	375.24
N-Cars [18]	Static	2	24029	3.09	0.10	30.93
N-Caltech256 [41]	Static	257	30607	104.96	1.00	105.65
DVS128 [28]	Dynamic	11	1464	367.98	6.52	55.62
UCF50 [41]	Dynamic	50	6681	1030.83	6.80	160.25
DVSAction [42]	Dynamic	10	292	801.4	5.18	154.67
NeuroIV [43]	Dynamic	16	4640	920.2	2.62	351.22

and NeuroIV[2] have some samples missing, hence we only use the available portion of the data, as shown in Table 1.

Following common practices, the datasets are split into training, validation, and test sets. The DVS128 dataset is preprocessed by cutting the event flow with a window size of 1500ms, same as in TA-SNN [35]. The N-Caltech101, N-Cars, NeuroIV, DVSAction and DVS128 are recorded in the real world, while the other datasets are converted from existing RGB datasets by simulators.

B. EXPERIMENT SETTINGS

We train our network with the Adam optimizer at a learning rate of 10^{-4}. The learning rate is adjusted by a cosine annealing scheduler [44], with a 10% linear warm-up.

For ACE, we quantize the timestamps with $\hat{M} = 100$. We use two Conv2d layers for a two-step aligned compression. The kernel size is 5, with $G_1 = 5$ and $G_2 = 2$, and $G = G_1 \times G_2 = 10$. Consequently, $M^* = \hat{M}/G = 10$. LeakyReLU is used as the activation function. Output channel C_{out} is 3. As in Section IV-A3, this design is equivalent to two Conv3d layers with kernel sizes of (5, 5, 5) and (2, 5, 5), and strides of (5, 1, 1) and (2, 1, 1), along the temporal and spatial axes.

For the BET video branch, two Conv1d layers with kernel size of 3 are used, without padding. We also use a max-pooling layer of window size 2 in between. The original sequence length is reduced to 1 with a final average-pooling layer. ResNet-18 [38] is used as the CNN backbone of BET.

After every training epoch, the model is validated on the validation set. The best checkpoint with the highest validation accuracy is selected for the test. During inference, the synthesis module is used to combine the predictions. The seed is fixed to reduce the effect of randomness. Also, we repeat the experiments with different seeds and take the average scores for Tables 2 and 3.

C. RESULTS ON STATIC DATASETS

As shown in Table 2, our proposed method surpasses existing methods by a significant margin, including HOTS [17], HATS [18], Two-Channel Image [45], Voxel Grid [47], Matrix-LSTM [19], EST [16] and Gao *et al.* [46]. Recently, some models based on GCN (graph convolutional network) and SNN have been proposed [34], [50], but still with inferior performance compared with state-of-the-art CNN-based

[2]https://github.com/ispc-lab/NeuroIV

TABLE 2. Accuracy (%) on static datasets. 'Type' denotes the backbone model type and 'E2E' indicates end-to-end method. Our method performs strongly even with the light Res18 backbone. MatrixLSTM+E2VID is reconstruction-based and EvDistill uses additional images as inputs, hence they are not in the scope of this work which focuses on end-to-end models using only event data.

Model	Type	E2E	NCars	NCal101	NCal256
HOTS [17]	-	✓	62.4	21.0	-
HATS [18]	Res34	✓	90.9	69.1	-
Two-Channel [45]	Res34	✓	86.1	71.3	-
Gao *et al.* [46]	-	✓	93.9	72.2	-
Voxel Grid [47]	Res34	✓	86.5	78.5	-
Event-LSTM [30]	Res18	✓	95.9	80.2	-
EST [16]	Res34	✓	92.5	83.7	63.11
Matrix-LSTM [19]	Res18	✓	94.37	84.31	-
N-ImageNet [48]	Res34	✓	94.73	86.81	-
MVF-Net [49]	Res18+34	✓	*96.8*	*87.1*	-
She *et al.* [34]	SNN	✓	-	71.2	-
SlideGCN [50]	GCN	✓	93.1	76.1	-
MLSTM+E2VID [19]	Res34	✗	(95.65)	(85.72)	-
MLSTM+E2VID [19]	Res18	✗	(95.80)	(84.12)	-
EvDistill [51] **(w/ Img)**	Res34	✓	-	90.2	-
ACE-BET (Ours)	Res18	✓	**96.99**	**89.25**	**67.86**
ACE-BET (Ours)	Res34	✓	**97.06**	**89.95**	**68.51**

TABLE 3. Accuracy (%) on dynamic datasets. ACE-BET surpasses existing state-of-the-art methods on all tasks. We reimplement the EST as a baseline for voxel-based methods. Res18 backbone is used for both EST and ACE-BET.

Model	Type	DVS128	UCF50	DVSAction	NeuroIV
PointNet [13]	PointCloud	88.8	-	-	-
Amir *et al.* [28]	PointCloud	94.4	-	-	-
Wang *et al.* [27]	PointCloud	95.32	-	-	-
PointNet++ [15]	PointCloud	95.2	-	-	-
PAT [12]	PointCloud	96.0	-	-	-
ST-EVNet [52]	PointCloud	97.27	-	*88.75*	-
Liu *et al.* [32]	SNN	92.7	-	78.1	-
PLIF [33]	SNN	97.57	-	-	-
SEW [53]	SNN	97.97	-	-	-
She *et al.* [34]	SNN	98.0	-	-	-
TA-SNN [35]	SNN	*98.61*	-	-	-
Salah *et al.* [5]	HandCrafted	-	*68.96*	61.94	-
Chen *et al.* [43]	Voxel	-	-	-	*89.02*
EST [16]	Voxel	96.91	67.99	-	-
ACE-BET (Ours)	Voxel	**98.88**	**84.03**	**98.31**	**91.35**

methods such as MVF-Net [49]. Our approach achieves the best performance for all datasets. Compared with the second-best solution, ACE-BET increases the precision by 2.85% on N-Caltech101, resulting in 89.95% accuracy. Also, for the N-Cars dataset, our method achieves the highest 97.06% accuracy. Although the EvDistill [51] yields 90.2% accuracy on the N-Caltech101 dataset, we stress that it utilizes both image and event modalities, which takes advantage of the extra information. Despite this, in Section V-E3 we show that by using a more efficient CNN backbone, our method can yield 91.95% accuracy at best, which is 1.75% higher than EvDistill, while only using the event modality of data.

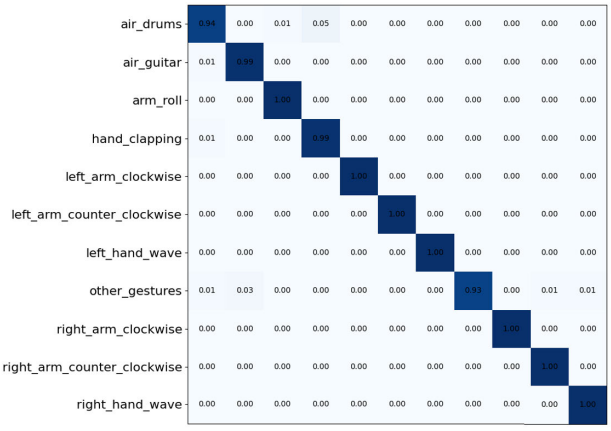
The N-Caltech256 is a new dataset that has not been used in previous works. We re-implemented EST [16] on N-Caltech256 as the baseline, because EST is one of the state-of-the-art methods on the previous two datasets and is open-sourced. With the new dataset, ACE-BET can still improve the accuracy by 5.4% over EST. Considering its large scale, the improvements on the N-Caltech256 may solidly support the superiority of our method.

D. RESULTS ON DYNAMIC DATASETS

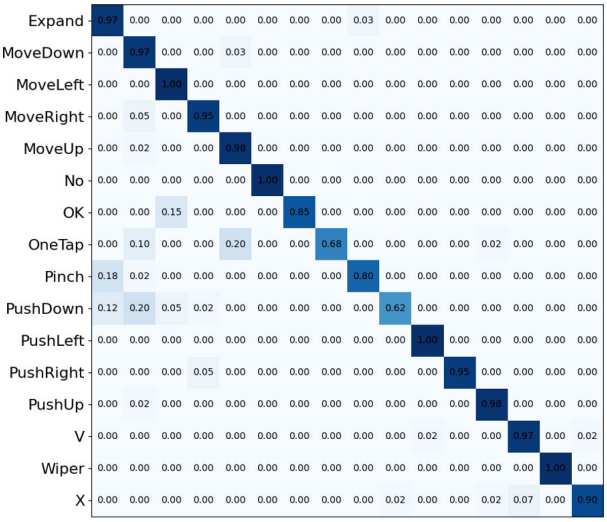
Table 3 shows the results on the dynamic datasets, where our method outperforms all other baselines, yielding 98.88% accuracy for the DVS128 and 84.03% for the UCF-50, surpassing all other point-cloud-based methods, such as Amir *et al.* [28], PAT [12], PointNet [13], PointNet++ [15] and TA-SNN [35]. While the baselines in Table 2 only consider static scenes and have not been applied to dynamic datasets, we re-implemented the EST method [16] as the representative of the voxel-based methods. The EST method achieves an accuracy of 96.91%, higher than state-of-the-art point cloud methods on the DVS128. On the other hand, some SNN-based methods perform strongly on the DVS128 gesture recognition task. However, the accuracy drops significantly for static scenes such as N-Caltech101. For example, Reference [34] that achieves 92.7% accuracy on DVS128 has only 71.2% accuracy on the N-Caltech101 dataset in Table 2.

We also evaluate ACE-BET on two more action recognition datasets of DVSAction and NeuroIV. For DVSAction, which contains 10 different daily actions, ACE-BET achieves 98.31% accuracy, 9.56% higher than the second best. However, the dataset is small and contains only 292 recordings, making the result relatively less authoritative. On the contrary, the NeuroIV dataset is one of the largest event camera gesture datasets, including 4,640 recordings of 16 gestures. To the best of our knowledge, no other studies have been tested on the dataset except the original paper [43]. On the large-scale NeuroIV, our method still yields state-of-the-art accuracy of 91.35%, performing robustly under different tasks. We show confusion matrices for DVS128 and NeuroIV in Fig. 6, including the accuracy for each class.

The UCF-50 is also a new dataset on which few existing works has been tested. Different from the DVS128, the dataset contains much longer recordings and complex actions, hence less reasonable for clipping. The



(a) DVS128 confusion matrix. Overall accuracy: 98.88%.



(b) NeuroIV confusion matrix. Overall accuracy: 91.35%.

FIGURE 6. Confusion matrices on DVS128 and NeuroIV datasets. Rows denote ground truths, and columns denote predictions. The darker area mainly concentrates on the diagonal, indicating high accuracy for all classes.

point-cloud-based methods are not applicable due to the cloud size (1030k events per sample on average). Therefore, we re-implemented the EST, which is one of the state-of-the-art methods and is open-sourced, on UCF-50. For this challenging dataset, EST drops to 67.99%, slightly lower than Reference [5]'s 68.96%, whereas our method achieves 84.03%, namely, a substantial improvement over 15% absolute accuracy.

E. ABLATION STUDIES

We conduct ablation studies to analyze the contributions of different modules in the following.

1) ABLATION ON ACE

To quantify the contributions of the ACE module and its three sub-modules, we conduct ablation studies in Table 4. Compared with the original ACE+BET design (89.25%),

TABLE 4. Ablation study on ACE. N-Caltech101 dataset is used. 'Quant' means the time stamp quantization step, 'Voxel' means the accumulative voxelization and 'Align' means the aligned compression. Removing any of the three modules results in an accuracy drop, demonstrating their contributions.

Variants	Quant	Voxel	Align	#bins	Acc (%)	Time (ms)
ACE (Ours)	✓	✓	✓	10	**89.25**	3.28
(Avg Comp)	✓	✓	✗	10	87.19	2.20
(Spike)	✓	✗	✓	10	88.57	3.42
(Spike + Accum)	✓	(Hybrid)	✓	10	88.22	4.68
(Direct Quant)	✓	✗	✗	10	86.61	**1.94**
EST	✗	-	-	9	86.50	6.26

TABLE 5. Ablation study on BET. ACE is used in all experiments as the representation for event data. Both frame and video branches improve the accuracy effectively.

BackBone	Frame branch	Video branch	NCal101 Acc (%)	Time (ms)
Res18	✓	✓	**89.25**	3.28
Res18	✓	✗	88.45	3.26
Res18	✗	✓	87.24	3.27
Res18	✗	✗	86.90	**3.24**
Res34	✓	✓	89.95	4.56

replacing the ACE module with EST [16] degrades the accuracy by 2.75% to 86.50%, showing the significance of the overall ACE module.

Moreover, in the 'ACE variants' part, we show the contributions of the ACE sub-modules. In this part, except for specific changes, other settings are the same as ACE. In the last row, 'Direct Quant' directly quantizes events into ten bins without the remaining sub-modules of accumulative voxelization and aligned compression, resulting in a 2.64% drop. The variant 'Spike' omits the accumulative voxelization step, hence each voxel represents event spikes as in EST, rather than accumulative intensity changes in the original design. 'Spike+Accum' denotes a hybrid of the two voxelization methods. Without accumulative voxelization, the accuracy decreases by 0.68% and 1.03%, respectively, showing the contribution of the accumulative voxelization. Finally, the 'Avg Comp' replaces the aligned compression by pixel-wise average pooling along the temporal axis, to compress the frames in groups. Without the spatial-temporal aligned compression, the accuracy drops by 2.06%.

From the ablation study results in Table 4, we find that our ACE method is an efficient representation for event data, improving accuracy in both object classification and action recognition tasks. Moreover, the three sub-modules of ACE all demonstrate their contributions in the ablation studies.

2) ABLATION ON BET

Next, we also study the contribution of the BET framework. Compared with ACE, BET is a simple design that extends the backbone CNN by adding two prediction branches, the frame and the video branch.

We ablate the two branches respectively in Table 5. Removing the video branch leads to 0.8% drop in the accuracy, while removing the frame branch results in 2.01% drop. When removing both branches, the BET rollback to the conventional Res18, having an accuracy 2.35% lower.

Another technical choice of BET is the CNN backbone. Compared with Res18, using Res34 as the backbone improves the accuracy by 0.7%. We also evaluate other different backbones in the next section.

The ablation studies on BET demonstrate the contributions of both video and frame branches. While each of them can effectively improve the performance, a combination of the two can further boost the accuracy.

3) BACKBONE CNN

BET adopts a CNN backbone as the encoder to learn a vector representation for each ACE frame. Previous methods [16], [18], [19] adopt the popular ResNet [38] as their backbone, which is also the default choice of BET. To study the performance of different CNN models, we run ablation studies on the N-Caltech101 dataset. As shown in Table 6, BET benefits from more powerful CNN backbones consistently. Compared with BET + Res18 configuration, the adoption of the EfficientNet_B2 [54] increases the accuracy by 2.7% to 91.95%. With similar or smaller model sizes, other choices such as EfficientNet_lite and DenseNet [55] can also improve the accuracy. These results confirm that the improvement brought by ACE-BET is universal. Even better, the BET framework is complementary to its backbone and can harvest from more advanced CNN models.

TABLE 6. Backbone ablation study on N-Cal101.

Backbone CNN	#params (Mb)	NCal101 Acc (%)
Res18 [38]	11.18	89.25
Res34 [38]	21.28	89.95
Dense121 [55]	7.48	90.12
ENet_lite0 [54]	4.03	90.75
ENetB2 [54]	8.42	**91.95**

F. INFLUENCE OF HYPER-PARAMETERS

To study the impact of the hyper-parameters $\hat{M}$ (number of quantization bins), G (frame compression ratio) and K (spatial kernel size), we conduct experiments on N-Caltech101 as in Table 7. We fix the seed to reduce randomness that may affect the results.

TABLE 7. Impact of hyper-parameters on N-Caltech101. Our method is robust with a range of settings.

$\hat{M}$	Acc(%)	G	Acc(%)	K	Acc(%)
50	88.86	5	89.25	3	88.23
100	89.25	10	89.25	5	89.25
150	89.26	20	88.91	9	89.43
200	88.91	50	88.05		

TABLE 8. Inference Speed of ACE-BET. The second column indicates whether the method processes new events asynchronously or synchronously in packages. For fairness, we list the results for both Res18 and Res34. The N-Cars dataset is used following the baseline methods. Same as Reference [16], [19], we also use the metrics of inference time, and event processing speed (in thousand events per second). Equivalent frame rates are also included (in frame per second).

Method	GPU type	Asynch.	Time (ms)	Speed (kEv/s)	Eqv. FPS
Gabor-SNN [18]	-	✓	285.95	14.15	3.50
HOTS [17]	-	✓	157.57	25.68	6.35
HATS [18]	-	✓	7.28	555.74	137.36
Matrix-LSTM [19]	GTX 1080 Ti	✗	8.25	468.36	121.21
EST [16]	RTX 2080 Ti	✗	*6.26*	*632.9*	*159.74*
ACE-BET (Res34) (Ours)	GTX 1080 Ti	✗	**4.56**	**891.80**	**219.30**
ACE-BET (Res18) (Ours)	GTX 1080 Ti	✗	**3.28**	**1240.96**	**304.87**

Our method is robust under different hyper-parameter settings. The maximum accuracy differences for $\hat{M}$, G and K are 0.4%, 1.2% and 1.2% respectively. This suggests that ACE benefits from the workflow and the architecture design, more than specific hyper-parameter settings.

For $\hat{M}$, a moderate number (100 or 150) yields higher accuracy than either long duration (50) or short duration (200). This is consistent with our observation in Fig.1b and Reference [37]. Furthermore, the accuracy tends to keep improving as the spatial kernel size K increases. We hypothesize that a larger kernel size would result in a larger receptive field during the spatial-temporal aligned compression procedure. Nonetheless, a larger kernel also increases the model size and slows down the inference speed, so there is a trade-off between efficiency and accuracy.

G. EFFICIENCY OF OUR APPROACH

Besides higher accuracy, the proposed method also outperforms other methods in terms of efficiency. Following Reference [16], [19], we use the inference time and the average event processing speed as metrics, and the N-Cars dataset is used. We use one GTX 1080Ti GPU and predict one sample at a time, same as in Reference [19] for fairness. Although EST [16] uses RTX 2080 Ti, our method is faster even with a less powerful GPU. The methods are categorized into asynchronous and synchronous classes. The asynchronous methods, such as Gabor-SNN [18], HOTS [17] and HATS [18], process the input events asynchronously upon their arrival. On the other hand, synchronous methods, such as Matrix-LSTM [19] and EST [16], process the events in packets.

From Table 8, the efficiency of ACE-BET is the highest among competing methods. To fairly evaluate the speed of BET, we use Res34 as the CNN backbone, same as other baselines. Benefiting from the efficient ACE representation, our approach achieves the lowest inference latency and fastest event processing speed, at 4.56 ms and 891.80 kEvents/s, respectively. Moreover, with the Res18 backbone, the ACE-BET is even faster at 3.28 ms, equivalent to an impressive 304 frames per second.

FIGURE 7. Visualizations of ACE (left) and EST [16] (right) in pairs. Our method enjoys sharper edges, clearer texture and finer details.

VI. VISUALIZATION

In Fig. 7 we visualize ACE in comparison with the EST representations. It is commonly observed that ACE generates clear edges and textures. Moreover, the differences are even more obvious when there are texts, as shown in the second row. This observation proves the effectiveness of the proposed aligned compression method in ACE. Moreover, we also notice that ACE demonstrates a higher contrast than EST. Because ACE tracks the historical intensity changes instead of short-term spikes, it has a larger range for the voxel values than EST. However, some waveform patterns similar to the moiré effect are also amplified compared to the EST representations.

We want to stress that no extra sharpness loss is introduced in our ACE-EFN framework, and the visual difference is a by-product during the classification and action recognition tasks. Therefore, the sharper ACE visualization results may not directly prove that ACE is a better representation for the static and dynamic tasks covered in this research. Nonetheless, we find a correlation between the sharper representations and the improvements in accuracy brought by ACE. Together with the results in Section V-E1, we believe the visualization results provide an intuitive interpretation to our method.

VII. CONCLUSION

This work proposes ACE-BET for event-based vision under both static and dynamic scenes. Based on the observation of the misalignment problem in the event data, as well as the mechanism of event cameras, we introduce ACE as a tensor representation, which takes the spatial-temporal relation and relative intensity into consideration. Moreover, BET is shown to be versatile for both static and dynamic scenes with a uniform network structure, which is the first of its kind to the best of our knowledge. Extensive experiments have confirmed that the integration of ACE and BET consistently outperforms all other baseline methods by large margins. The contributions of ACE and BET have been extensively quantified through ablation studies. ACE demonstrates itself to be a generically better voxel-based representation for event data under different scenes, and can be adopted for various event-vision tasks in the future. BET then improves the prediction accuracy further. Compared with existing approaches, ACE-BET is provably superior in inference time and promising for real-time online tasks. Nevertheless, the best choice of hyperparameters, such as the number of quantized frames and ratio of compression, might vary from scene to scene, requiring manual tuning for deployments. It would be appealing to let the model adjust the temporal resolution automatically, as a direction for future work.

REFERENCES

[1] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, and S. Leutenegger, "Event-based vision: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 1, pp. 154–180, Jul. 2022.

[2] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 × 128 120 dB 15μs latency asynchronous temporal contrast vision sensor," *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Feb. 2008.

[3] Z. Ge, N. Meng, L. Song, and Y. E. Lam, "Dynamic laser speckle analysis using the event sensor," *Appl. Opt.*, vol. 60, no. 1, pp. 172–178, Jan. 2021.

[4] J. Xu, M. Jiang, L. Yu, W. Yang, and W. Wang, "Robust motion compensation for event cameras with smooth constraint," *IEEE Trans. Comput. Imag.*, vol. 6, pp. 604–614, 2020.

[5] S. Al-Obaidi, H. Al-Khafaji, and C. Abhayaratne, "Making sense of neuromorphic event data for human action recognition," *IEEE Access*, vol. 9, pp. 82686–82700, 2021.

[6] E. Perot, P. de Tournemire, D. Nitti, J. Masci, and A. Sironi, "Learning to detect objects with a 1 megapixel event camera," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 16639–16652.

[7] C. Iaboni, H. Patel, D. Lobo, J.-W. Choi, and P. Abichandani, "Event camera based real-time detection and tracking of indoor ground robots," *IEEE Access*, vol. 9, pp. 166588–166602, 2021.

[8] Z. Ge, Y. Gao, K.-H. H. So, and Y. E. Lam, "Event-based laser speckle correlation for micro motion estimation," *Opt. Lett.*, vol. 46, no. 16, pp. 3885–3888, Aug. 2021.

[9] Z. Ge, P. Zhang, Y. Gao, K.-H. H. So, and Y. E. Lam, "Lens-free motion analysis via neuromorphic laser speckle imaging," *Opt. Exp.*, vol. 30, no. 2, pp. 2206–2218, Jan. 2022.

[10] H. C. Duwek, A. Bitton, and E. E. Tsur, "3D object tracking with neuromorphic event cameras via image reconstruction," in *Proc. IEEE Biomed. Circuits Syst. Conf. (BioCAS)*, Oct. 2021, pp. 1–4.

[11] B. Su, L. Yu, and W. Yang, "Event-based high frame-rate video reconstruction with a novel cycle-event network," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2020, pp. 86–90.

[12] J. Yang, Q. Zhang, B. Ni, L. Li, J. Liu, M. Zhou, and Q. Tian, "Modeling point clouds with self-attention and Gumbel subset sampling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3323–3332.

[13] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 652–660.

[14] Y. Liu, B. Fan, S. Xiang, and C. Pan, "Relation-shape convolutional neural network for point cloud analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 8895–8904.

[15] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5099–5108.

[16] D. Gehrig, A. Loquercio, K. Derpanis, and D. Scaramuzza, "End-to-end learning of representations for asynchronous event-based data," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 5633–5643.

[17] 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.

[18] 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.

[19] M. Cannici, M. Ciccone, A. Romanoni, and M. Matteucci, "A differentiable recurrent surface for asynchronous event-based data," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 136–152.

[20] H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza, "High speed and high dynamic range video with an event camera," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 6, pp. 1964–1980, Jun. 2021.

[21] L. Wang, T.-K. Kim, and K.-J. Yoon, "EventSR: From asynchronous events to image reconstruction, restoration, and super-resolution via end-to-end adversarial learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 8315–8325.

[22] L. Zhang, H. Zhang, J. Chen, and L. Wang, "Hybrid deblur net: Deep non-uniform deblurring with event camera," *IEEE Access*, vol. 8, pp. 148075–148083, 2020.

[23] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and A.-C. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 802–810.

[24] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.* Cham, Switzerland: Springer, 2015, pp. 234–241.

[25] C. Scheerlinck, H. Rebecq, D. Gehrig, N. Barnes, R. E. Mahony, and D. Scaramuzza, "Fast image reconstruction with an event camera," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2020, pp. 156–163.

[26] H. Rebecq, D. Gehrig, and D. Scaramuzza, "ESIM: An open event camera simulator," in *Proc. Conf. Robot Learn.*, 2018, pp. 969–982.

[27] Q. Wang, Y. Zhang, J. Yuan, and Y. Lu, "Space-time event clouds for gesture recognition: From RGB cameras to event cameras," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2019, pp. 1826–1835.

[28] 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.

[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[30] L. Annamalai, V. Ramanathan, and C. S. Thakur, "Event-LSTM: An unsupervised and asynchronous learning-based representation for event-based data," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 4678–4685, Apr. 2022.

[31] Z. Ge, T. Zeng, and E. Y. Lam, "Lensless sensing using the event sensor," in *Proc. OSA Imag. Appl.*, Jul. 2021, Paper ITu6B.5.

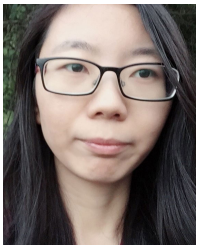
[32] Q. Liu, D. Xing, H. Tang, D. Ma, and G. Pan, "Event-based action recognition using motion information and spiking neural networks," in *Proc. 30th Int. Joint Conf. Artif. Intell.*, vol. 8, Z.-H. Zhou, Ed., Aug. 2021, pp. 1743–1749.

[33] 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.

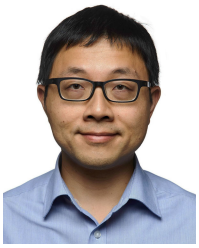
[34] X. She, S. Dash, and S. Mukhopadhyay, "Sequence approximation using feedforward spiking neural network for spatiotemporal learning: Theory and optimization methods," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–22.

[35] M. Yao, H. Gao, G. Zhao, D. Wang, Y. Lin, Z. Yang, and G. Li, "Temporal-wise attention spiking neural networks for event streams classification," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 10221–10230.
[36] Z. Ge, Y. Zhu, Y. Zhang, and Y. E. Lam, "Dynamic speckle analysis using the event-based block matching algorithm," *Proc. SPIE*, vol. 11901, Oct. 2021, Art. no. 119010R.
[37] M. Iacono, S. Weber, A. Glover, and C. Bartolozzi, "Towards event-driven object detection with off-the-shelf deep learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 1–9.
[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
[40] G. Orchard, A. Jayawant, G. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers Neurosci.*, vol. 9, p. 437, Nov. 2015.
[41] Y. Hu, H. Liu, M. Pfeiffer, and T. Delbruck, "DVS benchmark datasets for object tracking, action recognition, and object recognition," *Frontiers Neurosci.*, vol. 10, p. 405, Aug. 2016.
[42] S. Miao, G. Chen, X. Ning, Y. Zi, K. Ren, Z. Bing, and A. Knoll, "Neuromorphic vision datasets for pedestrian detection, action recognition, and fall detection," *Frontiers Neurorobot.*, vol. 13, p. 38, Jun. 2019.
[43] G. Chen, F. Wang, W. Li, L. Hong, J. Conradt, J. Chen, Z. Zhang, Y. Lu, and A. Knoll, "NeuroIV: Neuromorphic vision meets intelligent vehicle towards safe driving with a new database and baseline evaluations," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 1171–1183, Feb. 2022.
[44] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–16.
[45] A. I. Maqueda, A. Loquercio, G. Gallego, N. Garcia, and D. Scaramuzza, "Event-based vision meets deep learning on steering prediction for self-driving cars," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5419–5427.
[46] S. Gao, G. Guo, H. Huang, X. Cheng, and C. L. P. Chen, "An end-to-end broad learning system for event-based object classification," *IEEE Access*, vol. 8, pp. 45974–45984, 2020.
[47] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, "Unsupervised event-based learning of optical flow, depth, and egomotion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 989–997.
[48] J. Kim, J. Bae, G. Park, D. Zhang, and Y. M. Kim, "N-ImageNet: Towards robust, fine-grained object recognition with event cameras," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 2146–2156.
[49] Y. Deng, H. Chen, and Y. Li, "MVF-Net: A multi-view fusion network for event-based object classification," *IEEE Trans. Circuits Syst. Video Technol.*, early access, Apr. 16, 2021, doi: 10.1109/TCSVT.2021.3073673.
[50] Y. Li, H. Zhou, B. Yang, Y. Zhang, Z. Cui, H. Bao, and G. Zhang, "Graph-based asynchronous event processing for rapid object recognition," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 934–943.
[51] L. Wang, Y. Chae, S.-H. Yoon, T.-K. Kim, and K.-J. Yoon, "EvDistill: Asynchronous events to end-task learning via bidirectional reconstruction-guided cross-modal knowledge distillation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 608–619.
[52] Q. Wang, Y. Zhang, J. Yuan, and Y. Lu, "ST-EVNet: Hierarchical spatial and temporal feature learning on space-time event clouds," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2020. [Online]. Available: https://www.researchgate.net/publication/343151693_ST-EVNet_Hierarchical_Spatial_and_Temporal_Feature_Learning_on_Space-time_Event_Clouds
[53] W. Fang, Z. Yu, Y. Chen, T. Huang, T. Masquelier, and A. Tian, "Deep residual learning in spiking neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 1–14.
[54] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.
[55] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.

CHANG LIU received the B.Eng. degree in computer engineering from The University of Hong Kong, Hong Kong, China, in 2019, where he is currently pursuing the Ph.D. degree with the Department of Electrical and Electronic Engineering, supervised by Dr. N. Wong. His research interests include event cameras, deep learning, 3D point cloud, and weakly supervised object detection.



XIAOJUAN QI (Member, IEEE) received the B.Eng. degree in electronic science and technology from Shanghai Jiao Tong University (SJTU), China, in 2014, and the Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong, Hong Kong, in 2018. She is currently an Assistant Professor with The University of Hong Kong, Hong Kong.



EDMUND Y. LAM (Fellow, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering from Stanford University. From 2010 to 2011, he was a Visiting Associate Professor at the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology. He is currently a Professor of electrical and electronic engineering and the Director of computer engineering program with The University of Hong Kong. His main research interest includes computational imaging, optics, image analysis, and machine intelligence. He is a fellow of Optica, SPIE, IS&T, and HKIE.

NGAI WONG (Senior Member, IEEE) received the B.Eng. and Ph.D. degrees in EEE from The University of Hong Kong (HKU). He was a Visiting Scholar at Purdue University, West Lafayette, IN, USA, in 2003. He is currently an Associate Professor with the Department of Electrical and Electronic Engineering, HKU. His research interests include electronic design automation (EDA), model order reduction, tensor algebra, linear and nonlinear modeling and simulation, and compact neural network design.