A Voxel Graph CNN for Object Classification with Event Cameras

Yongjian Deng[1] Hao Chen[2*] Hai Liu[3] Youfu Li[4*]

[1]College of Computer Science, Beijing University of Technology

[2]School of Computer Science and Engineering, Southeast University

[3]The National Engineering Research Center for E-Learning, Central China Normal University

[4]Department of Mechanical Engineering, City University of Hong Kong

yjdeng@bjut.edu.cn, haochen303@seu.edu.cn, hailiu0204@ccnu.edu.cn, meyfli@cityu.edu.hk

Abstract

Event cameras attract researchers' attention due to their low power consumption, high dynamic range, and extremely high temporal resolution. Learning models on event-based object classification have recently achieved massive success by accumulating sparse events into dense frames to apply traditional 2D learning methods. Yet, these approaches necessitate heavy-weight models and are with high computational complexity due to the redundant information introduced by the sparse-to-dense conversion, limiting the potential of event cameras on real-life applications. This study aims to address the core problem of balancing accuracy and model complexity for event-based classification models. To this end, we introduce a novel graph representation for event data to exploit their sparsity better and customize a lightweight voxel graph convolutional neural network (EV-VGCNN) for event-based classification. Specifically, (1) using voxel-wise vertices rather than previous point-wise inputs to explicitly exploit regional 2D semantics of event streams while keeping the sparsity; (2) proposing a multi-scale feature relational layer (MFRL) to extract spatial and motion cues from each vertex discriminatively concerning its distances to neighbors. Comprehensive experiments show that our model can advance state-of-the-art classification accuracy with extremely low model complexity (merely 0.84M parameters).

1. Introduction

Each pixel of event cameras is independent and only report lightness changes at the correspondent location (Fig. 1). This novel working principle enables their output to be sparse and non-redundant [28, 37]. Consequently, event cameras hold advantages of low power consumption, high response speed, and high dynamic range compared to traditional cameras [17]. How to tailor models for event data with the particular format to perform core vision tasks, such as object classification, has been a trending topic. Motivated by the huge success of learning-based methods on vision tasks, developing data-driven approaches for event data becomes a leading choice. For instance, many studies [5, 12, 19, 50, 51] resort to 2D convolutional neural networks (CNNs) by converting sparse events to dense frames. These works achieve advanced performance utilizing well-pretrained 2D CNNs. Yet, the constructed dense representation and large size models sacrifice the sparsity of event data (Fig. 2 (a)) while limiting the potential of event cameras on mobile or wearable applications.

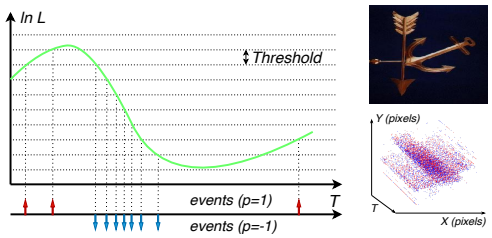

Figure 1. **Left:** A sketch of the working principle of event cameras (the detailed working principle is introduced in supplementary material). Events are produced asynchronously according to the lightness ($\ln L$) changes. Red and blue arrows represent positive and negative events, respectively. **Right:** The RGB image captured from the traditional RGB camera (top) and event signals in the original format produced from an event camera (bottom).

To exploit the sparsity of event data and build low-complexity models, recent researchers migrate learning models initially designed for point clouds to event data, such as the works [43, 47] migrate models from pointnet-like architecture [38] and the approaches [3, 31] utilize graph neural networks. Although these approaches advance in exploiting the sparsity advantage of event data, a fundamental question has never been investigated: Is point-wise

*: Corresponding author

This work was supported by the National Natural Science Foundation of China (61873220, 92167102, 62102083, 62173286, 61875068, 62177018), the Natural Science Foundation of Jiangsu Province (BK20210222), the Research Grants Council of Hong Kong (CityU 11213420), and the Science and Technology Development Fund, Macau SAR (0022/2019/AKP).

input (taking event points as processing units) proper for event-based vision tasks?

Intuitively, each point in 3D point clouds can be used as a key point to building the external structure of an object [8]. Therefore, these points do well in describing the geometry, which is the key for classifying 3D objects. Instead, event data are more like 2D videos recorded asynchronously. Event-based classification models require the ability to accurately extract 2D semantics from the event data rather than their "geometry" (Fig. 1), which usually contains motion cues or motion trajectories. Thus, we believe that using raw events as input is not suitable, as it is difficult for sparse event points to provide decisive features (*e.g.*, local 2D semantics) for event-based models.

To overcome the lacking of characterizing local 2D semantics in the popular point-based solutions, this study proposes a novel voxel-wise representation for event data. Inspired by the traditional image domain: it is challenging to extract decisive features from images using discrete or discontinuous pixels (like sparse point-wise input). Thus, we propose a representation to encode locally coherent 2D semantics by describing the regional events contained in each voxel. In specific, we build a graph by voxelizing event points, selecting representative voxels as vertices (Fig. 2 (c)), and connecting them according to their spatio-temporal relationships for further processing. Each voxel in the graph can be analogous to frame patches of still images that contain essential cues such as local textures and contours [14], which can help the network recognize 2D scenes effectively.

Besides the proposed representation, a lightweight graph-based learning architecture (*EV-VGCNN*) is introduced. The critical problem for designing an event-based graph model is how to learn the embeddings for the edges and vertices' features. First, we learn a scoring matrix for each vertex according to spatio-temporal geometry and utilize the learned matrix to achieve feature aggregation across its neighbors attentively. Moreover, for a vertex in the event-based graph, its adjacent neighbors usually convey local spatial messages, while distant neighbors are more likely to carry motion cues or global changes. Inspired by this variation, we design a multi-scale feature relational layer (*MFRL*) to extract semantic and motion cues from each vertex discriminatively. Specifically, two learnable blocks in *MFRL* are applied to adjacent and distant neighbors respectively, and the obtained features are aggregated as the joint representation of a vertex. Finally, we cascade multiple *MFRL* modules with graph pooling operations and a classifier as the *EV-VGCNN* to perform end-to-end object classification. The proposed model achieves SOTA accuracy while holding surprisingly low model complexity.

The main contributions of this paper are summarized as follows: (1) We introduce a novel method to construct event-based graph representation with correspondence to the properties of event data, which can effectively utilize informative features from voxelized event streams while maintaining the sparse and non-redundant advantage of event data. (2) We introduce the *MFRL* module composed of several *SFRL*s to discriminatively learn spatial semantics and motion cues from the event-based graph according to spatio-temporal relations between vertices and their neighbors. (3) Extensive experiments show that our model enjoys noticeable accuracy gains with extremely low model complexity (merely 0.84M parameters).

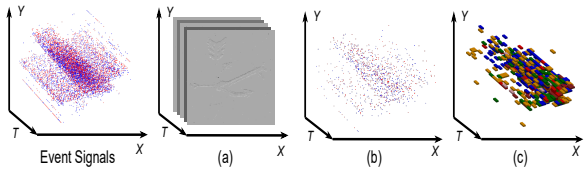

Figure 2. Visual comparison of three types of event-based representation: **(a)** the frame-based representation by integrating events into dense frames; **(b)** the point-based representation generated by sampling a subset of event signals; **(c)** representative event voxels selected as vertices in the proposed graph.

2. Related Work

Event-based approaches can be distinguished into two categories: frame-based method and point-based method.

(i) Frame-based method: integrating event signals into frame-wise 2D representations to directly adopt 2D CNNs for event data. For example, studies such as [11, 12, 29, 30, 50, 51] accumulating event points along the time axis to frames *w.r.t* events' polarities, locations and timestamps. To adaptively embed events to frames, [19] and [5] introduce learnable neural blocks to weigh the importance of each event in the frame. Besides, The success of methods [10, 18, 23, 40, 46] in knowledge transfer also benefits from the frame-based representation. While these frame-based methods achieve encouraging performance on event-based classification, the conversion from sparse points to dense 2D maps introduces much redundant information [31]. As a result, they typically require heavy-weight models to extract high-level features, limiting the potential of event cameras on mobile or wearable applications.

(ii) Point-based method: taking a single event or a set of regional events as input units for feature extraction and aggregation. Early studies falling in this category focus on designing handcraft descriptors for event data to perform multiple vision tasks, such as corner detection [6, 33], edge/line extraction [29, 42], optical flow prediction [4, 7], denoising [48] and object classification [26, 44]. However, the quality of their extracted features is usually sensitive to the noise and scene variation, limiting their generalizability to complex scenarios. To enable models to respond adaptively to various samples of event data, the development of point-based learning methods is gradually becoming main-

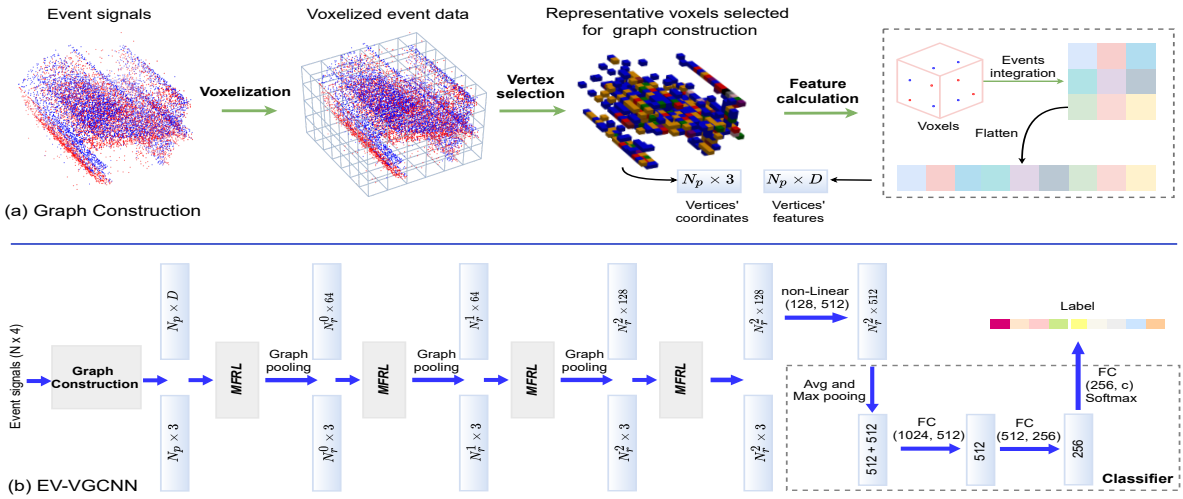

Figure 3. (a) **Graph construction**: We first voxelize event data and then select N_p representative voxels as the graph's input vertices according to the event points number inside each voxel. Finally, we attain features of vertices by integrating their internal events along the time axis. (b) **EV-VGCNN**: It consists of multiple multi-scale feature relational layers (*MFRL*) followed by graph pooling for learning global features from the input graph, and several linear layers for object classification. The *MFRL* module is to aggregate features for each vertex from k-nearest neighbors. In the figure, non-Linear(x, y) is a block with the input channel of x and output channel of y. It contains a linear layer, a Batch Normalization, and a ReLU layer. N_r^n represents the output number of vertices of each graph pooling operation.

stream. There are many visual tasks have been performed successfully, *e.g.*, motion estimation [43], motion segmentation [31] and object classification [1, 3, 13, 35, 36, 47]. Among these approaches, the closest studies to ours are methods inspired by 3D point cloud learning models, such as [3, 31, 43, 47], where the RG-CNNs in [3] acquires superior results over other point-based methods on various datasets. These models are lightweight and show their potential in multiple tasks. However, all these algorithms take the original events as input units for further processing. As stated in Section 1, this input representation is challenging to extract regional 2D semantics effectively from event data. As a result, their models' performance on object classification tasks is far behind the frame-based solutions. Instead of using the original events as input units, this paper proposes a novel voxel-wise representation. Our constructed representation retains more semantic and motion cues while maintaining the sparsity advantage. Moreover, the *MFRL* module allows us to flexibly aggregate features from different neighbors of each vertex in the graph. Consequently, our model outperforms RG-CNNs [3] with a large margin in terms of accuracy and model complexity. In the supplementary, we also include connections between our models with voxel-based approaches for 3D point cloud learning.

3. The Proposed Method

We propose a novel graph construction method for event data and design a lightweight network *EV-VGCNN* for object classification. The pipeline of our graph-based solution is shown in Fig. 3, which can be depicted as follows. (i) The event stream is organized into voxels, where each voxel may contain several events. (ii) We regard each voxel as a vertex in our graph. In specific, the coordinate of a vertex is determined by voxel positions. The feature (temporal and regional semantics) at each vertex is obtained by accumulating event point features inside its corresponding voxel. (iii) We utilize the *EV-VGCNN* for sequentially aggregating each vertex's features to generate global representations. In the following, we detail two crucial components in our learning architecture sequentially, including the graph construction (Sec. 3.1) and the *EV-VGCNN* (Sec. 3.2).

3.1. Graph construction

In this part, we build a directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{1, ..., N_p\}$ and $\mathcal{E}$ represent vertices and edges respectively. Each vertex $\mathcal{V}_i$ has two attributes which are the spatio-temporal coordinate $\mathcal{U}_i \in \mathbb{R}^{1\times 3}$ and the feature vector $\mathcal{F}_i \in \mathbb{R}^{1\times D}$. As the network *EV-VGCNN* is capable of finding neighbors and calculate edges' weights for vertices, we only need to determine the coordinate and features of each vertex in the graph construction stage.

Voxelization. Event streams can be expressed as a 4D tuple:

$$\{e_i\}_N = \{x_i, y_i, t_i, p_i\}_N, \tag{1}$$

where e_i is a single event. The first two dimensions x_i and y_i are constrained with the range of the spatial resolution ($\{H, W\}$) of event cameras and t_i is the timestamp when the event is triggered. Hence, (x_i, y_i, t_i) represents the spatio-temporal coordinate of an event, and the last dimension p_i can be seen as the attribute. Given an event

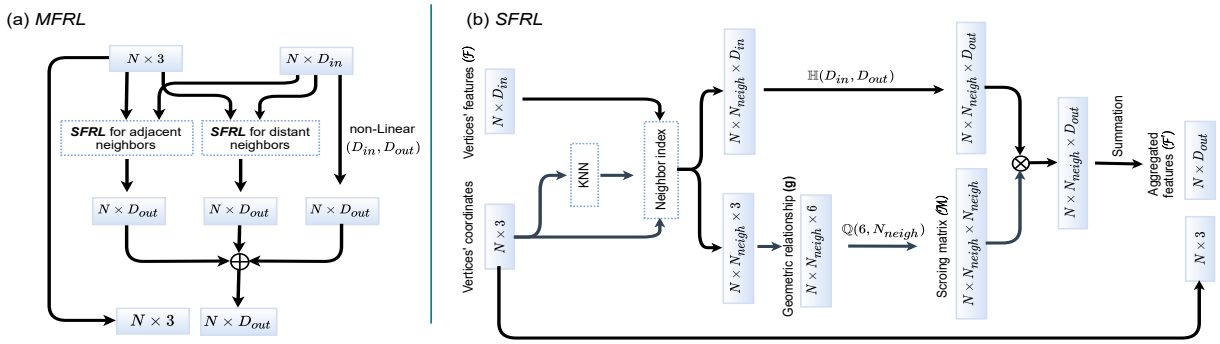

Figure 4. Structure of the *MFRL* and its base component *SFRL*. (a) ***MFRL***: The *MFRL* is composed of two *SFRL* modules and a shortcut connection, in which the two SFRL modules are to encode features from adjacent and distant neighbor vertices respectively. This design allows us to explore motion and spatial messages behind event signals flexibly. $\bigoplus$: element-wise addition. (b) ***SFRL***: This module realizes the Eq. (4) and Eq. (5) using neural networks. Particularly, the *SFRL* module takes vertices with their coordinates and features as input. For each vertex, the *SFRL* builds edges between the vertex and its N_{neigh} neighbors, computes the scoring matrix for its neighbors then aggregates the features from neighbors using the matrix to obtain the representation of this vertex. $\bigotimes$: matrix multiplication.

stream, we can subdivide the 3D x-y-t space into voxels, as shown in Fig. 3 (a). Considering the value discrepancy between (x_i, y_i) and (t_i), we first normalize the time dimension with a compensation coefficient *A* using:

$$\{t_i\}_N = \frac{\{t_i - t_0\}_N \times A}{t_{N-1} - t_0}. \tag{2}$$

After normalization, event signals encompass 3D space with range H, W, A along the x, y, t axes respectively. Then, we voxelize this 3D space with the size of each voxel as v_h, v_w and v_a. The resulting voxels in spatio-temporal space is of size $H_{voxel} = H/v_h$, $W_{voxel} = W/v_w$ and $A_{voxel} = A/v_a$, where each voxel may contain several events. In this work, we refer voxel locations to define the coordinate of each vertex in the graph. Accordingly, the coordinate of *i*-th vertex $\mathcal{U}_i = (x_i^v, y_i^v, t_i^v)$ is with the range of $\{H_{voxel}, W_{voxel}, A_{voxel}\}$. For simplicity, we assume that H, W, A are divisible by v_h, v_w and v_a.

Vertex selection. In practice, even though we only consider non-empty voxels as vertices, there are still tens of thousands of vertices that will be contained in a graph. Constructing a graph with all these vertices imposes a substantial computational burden. In addition, due to noise points (also known as hot pixels) produced by event cameras typically occurs isolated without any connections with other events, plenty of vertices may only be composed of single or a few noise points only. Accordingly, taking these uninformative vertices as input would inevitably introduce interference to the whole learning system. Therefore, we propose to keep only representative vertices for graph construction. To this end, we adopt a simple yet effective selection strategy, which is to find N_p vertices with the largest number of event points inside and feed them into the learning system. On the one hand, this selection strategy can work as a noise filter for input data purification. On the other hand, this procedure can help us save computational costs. Please refer to our supplementary material for more comparisons with different selection approaches.

Feature calculation for vertices. The performance of graph-based neural networks heavily relies on the quality of each vertex's input features. Since each vertex in our graph contains several event points inside, an appropriate method to encode the features for these points is required. Event streams can be seen as a 2D video recorded asynchronously. As an event voxel (vertex) is generally with a short time span, we think it is rational to represent the 2D semantics of voxels simulating the imaging principle of the traditional cameras. That is, accumulating event points into 2D frame patches along the time axis. Particularly, given a graph with vertices $\{\mathcal{V}_i\}_{N_p}$ and coordinates $\{(x_i^v, y_i^v, t_i^v)\}_{N_p}$, we can attain 2D features $\mathcal{F}_i^{2d} \in \mathbb{R}^{1 \times v_h \times v_w}$ of its *i*-th vertex $\mathcal{V}_i$ as formulated in Eq. (3).

$$\mathcal{F}_i^{2d}(x, y) = \sum_i^{N_v} p_i^{in} \delta(x - x_i^{in}, y - y_i^{in}) t_i^{in}, \tag{3}$$

where N_v denotes the number of events inside the vertex $\mathcal{V}_i$. For each internal event, its coordinate $(x_i^{in}, y_i^{in}, t_i^{in})$ is constrained with the size of voxels $(\{v_h, v_w, v_a\})$. By linearly integrating the events *w.r.t* their temporal coordinates and polarities, we want to encode the 2D semantics of each vertex while retaining temporal (motion) cues to a certain extent [19, 49, 51]. Eventually, we flatten the resulting 2D features $\{\mathcal{F}_i^{2d}\}_{N_p}$ to obtain feature vectors $\{\mathcal{F}_i\}_{N_p} \in \mathbb{R}^{N_p \times D}$ of all vertices in the graph, where $D = v_h v_w$.

3.2. EV-VGCNN

The proposed learning architecture comprises three main components: the multi-scale feature relational layer, the graph pooling operation, and the classifier. In this section,

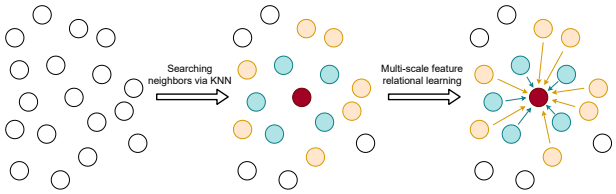

Figure 5. An intuitive illustration of how the *MFRL* aggregate features for a vertex from its adjacent and distant neighbors. For a vertex (the red point) in our graph, we firstly determine its N_{neigh}^{adj} (blue points) and N_{neigh}^{dis} (yellow points) neighbors $w.r.t$ distances. Then, we aggregate features from these neighbors utilizing two independent network branches, where yellow arrows and blue arrows represent learned weights from two *SFRL*s.

we show how to design them and assemble these modules into the *EV-VGCNN*.

Multi-scale feature relational layer (*MFRL*). Unlike the traditional 3D point clouds whose coordinates only express geometric messages, event data contains two different types of cues, $i.e.$, 2D spatial messages, and motion information. For a vertex in the event-based graph, its adjacent neighbors usually carry local spatial cues. In contrast, its distant neighbors are more likely to comprise much motion information. Notably, though adjacent and distant neighbors carry motion and spatial cues simultaneously in most cases, the motion and spatial variances between a vertex and its adjacent and distant neighbors are different, $i.e.$, adjacent neighbors hold small and local variance while distant neighbors carry more global changes. Given this disparity, it is difficult to use a shared CNN branch to learn all neighbors. Inspired by the multi-scale learning strategy in [39], we introduce the *MFRL* module to extract motion and semantic messages from vertices distinguishably depending on the distance between vertices and their neighbors. As shown in Fig. 4 (a), the *MFRL* consists of one shortcut connection and two single-scale feature relational layers (*SFRL*) to extract correlations from adjacent and distant neighbors respectively. More specifically, for a vertex, we define N_{neigh}^{adj} and N_{neigh}^{dis} as the numbers of its adjacent and distant neighbors, respectively. The results obtained from these three branches are then aggregated as the output. We intuitively illustrate this learning process in Fig. 5.

We then detail how to design the *SFRL* module. In contrast to aggregating neighbors' features only using pooling operations like PointNet++ [39], we introduce a scoring matrix computed with correspondence to spatio-temporal relationships between each vertex and its neighbors to achieve feature aggregation attentively. In specific, through the construction procedure depicted in Sec. 3.1, we have obtained features ($\mathcal{F}$) and coordinates ($\mathcal{U}$) of vertices ($\mathcal{V}$) in the graph. The *SFRL* takes these features and coordinates as inputs and is entailed to accomplish three functions: (i) building connections among vertices with edges, (ii) computing the scoring matrix for neighbors of the vertex, and (iii) integrating features from the neighborhood for each vertex. As shown in Fig. 4 (b), to achieve these goals, we first utilize the K-Nearest Neighbor algorithm (K-NN) to determine N_{neigh} neighbors for each vertex and link them with edges [16]. The graph includes a self-loop, meaning that each vertex also links itself as a neighbor [25]. Then, for the i-th vertex $\mathcal{V}_i$ with edges $\mathcal{E}_i$ ($\mathcal{E}_i \in \mathbb{R}^{1\times N_{neigh}}$) and coordinates $\mathcal{U}_i$, the scoring matrix can be calculated using:

$$\mathcal{M}_i = \mathbb{Q}(\underset{j:(i,j)\in\mathcal{E}_i}{\mathbb{S}_G}(g_{i,j}); W_m), \tag{4}$$

where $g_{i,j} = [\mathcal{U}_i, \mathcal{U}_i - \mathcal{U}_j] \in \mathbb{R}^6$ represents the geometric relation between a vertex and its neighbors. $[\cdot,\cdot]$ denotes the concatenation of two vectors. $\mathbb{S}_G(\cdot)$ is a function that stacks all geometric relations of the vertex's neighbors ($\{g_{i,j} : (i,j) \in \mathcal{E}_i\}$) and its output is in $\mathbb{R}^{N_{neigh}\times 6}$. $\mathbb{Q}$ is parameterized by W_m and comprises a linear mapping function, a Batch Normalization, and a Tanh function to explore geometric messages from neighbor vertices. The output $\mathcal{M}_i \in \mathbb{R}^{N_{neigh}\times N_{neigh}}$ is the scoring matrix for $\mathcal{V}_i$, which aims to re-weight features from its neighbors based on spatio-temporal relationships when aggregating the neighbors' features for the central vertex. Finally, we formulate the function of aggregating features from neighbors into the vertex as:

$$\mathcal{F}_i^{'} = \sum \mathcal{M}_i(\mathbb{H}(\underset{j\in\mathcal{E}_i}{\mathbb{S}_F}(\mathcal{F}_j); W_f)), \tag{5}$$

where $\mathbb{S}_F(\cdot)$ is a function that stacks all features ($F_j \in \mathbb{R}^{1\times D_{in}}$) from neighbor vertices and its output is in $\mathbb{R}^{N_{neigh}\times D_{in}}$. $\mathbb{H}$ is a non-linear transform function with parameters W_f and consists of a linear mapping function, a Batch Normalization and a ReLU. The function $\mathbb{H}$ takes the stacked features as input and produces transformed features in $\mathbb{R}^{N_{neigh}\times D_{out}}$. After that, the scoring map $\mathcal{M}_i$ is utilized to re-weight neighbors' features. We then apply a summation operation on the feature space over all neighbors to generate the aggregated features $\mathcal{F}_i^{'} \in \mathbb{R}^{1\times Dout}$ for the i-th vertex.

Graph pooling operation. The pooling operation is to reduce the vertex number in the network progressively. The pooling layer in 3D vision models commonly aggregates local messages of each point and then selects a subset of points with the dominant response for the following processing. In our case, the feature aggregation step has been fulfilled by the *MFRL* module. Hence, in our pooling layers, we only need to randomly select vertices from the graph to enlarge the receptive field for the feature aggregation of each vertex. We denote the output number of vertices of the graph pooling operation as N_r.

Classifier. Following the operation used in [38, 39], we apply symmetry functions on the high-level features to achieve a global representation for the input. Specifically,

we use max and average pooling operations to process these high-level features respectively, and then concatenate them to form a one-dimensional feature vector. Finally, we feed the global feature vector to three fully connected layers for classification.

Network architecture. The structure of *EV-VGCNN* is the same for all datasets. As shown in Fig. 3 (b). We embed four relational learning modules (*MFRL*) into our model to obtain the global representation of an event stream. Besides, we apply the pooling operation after each of the first three *MFRL*s. We further apply a non-linear block consisting of a linear layer, a Batch Normalization, and a ReLU after the fourth *MFRL* and then feed the output feature of this non-linear block to the classifier.

4. Experimental Evaluation

In this section, we use several benchmark datasets to evaluate the proposed method on the classification accuracy, the model complexity (measured in the number of trainable parameters), and the number of floating-point operations (FLOPs). Also, please refer to the supplementary material for details about the effectiveness of our method on the action recognition task.

4.1. Datasets

We validate our method on five representative event-based classification datasets: N-MNIST (N-M) [34], N-Caltech101 (N-Cal) [15, 34], CIFAR10-DVS (CIF10) [27], N-CARS (N-C) [44] and ASL-DVS (ASL) [2]. In general, there are two alternatives to generate these datasets. Particularly, N-M, N-Cal and CIF10 are obtained by recording traditional images displayed on monitors with fixed motion trajectories. This recording method may suffer from artificial noise introduced by the shooting and emulation environments. On the contrary, N-C and ASL are recorded in the real-world environment using event cameras, which means that the evaluation results on these two datasets should better reflect the performance of event-based models in practice. We train models separately for each dataset and evaluate the performance of our approach on their testing sets. For those datasets (N-Cal, CIF10, and ASL) without official splitting, we follow the experiment setup adopted in [3, 44], in which 20% data are randomly selected for testing, and the remaining is used for training and validation. We average over five repetitions with different random seeds as our reported results.

4.2. Implementation details

Graph construction. We fix the compensation coefficient A for all datasets as 9 to normalize event data. We set the input vertex number N_p as 512 for N-MNIST and ASL as the objects in them are small size, set N_p as 1024 for N-CARS, and $N_p = 2048$ for more complex datasets N-Cal and CIF10. According to the spatio-temporal discrepancy across different datasets, we set the voxel size as $(v_h, v_w, v_a) = (2, 2, 1)$ for N-MNIST, $(7, 7, 1)$ for CIF10 and $(5, 5, 3)$ for other datasets. Please refer to supplementary materials for details about voxel size settings.

Network. The values of N_{neigh}^{adj} and N_{neigh}^{dis} for all *MFRL* modules are fixed as 10 and 15 respectively. We set the output vertex number of three graph pooling operations as 896, 768, and 640 for N-Cal, N-C, and CIF10 datasets. For the other two datasets, which only take 512 vertices as input, we remove the pooling operation between *MFRL* modules. We add dropout layers with a probability of 0.5 after the first two fully-connected layers in the classifier to avoid overfitting. Each fully-connected layer is followed by a LeakyReLU and a Batch Normalization except for the prediction layer.

Training. We train our model from scratch for 250 epochs by optimizing the cross-entropy loss using the SGD [45] optimizer (except for the ASL) with a learning rate of 0.1 and reducing the learning rate until 1e-6 using cosine annealing. As for the ASL, we experimentally find that using the Adam [24] optimizer with a learning rate of 0.001 and decaying the learning rate by a factor of 0.5 every 20 epochs contributes to better performance. The batch size for training is set to 32 for all datasets.

Table 1. Comparison of the classification accuracy between ours and other point-based methods. † Using our proposed representation as input. Blue and green color indicate the first and second best performance.

Method	N-M	N-Cal	N-C	CIF10	ASL
H-First [35]	0.712	0.054	0.561	0.077	-
HOTS [26]	0.808	0.21	0.624	0.271	-
HATS [44]	0.991	0.642	0.902	0.524	-
EventNet [43]	0.752	0.425	0.750	0.171	0.949
PointNet++ [47]	0.841	0.503	0.809	0.465	0.947
PointNet++ [47]†	0.955	0.621	0.907	0.533	0.956
RG-CNNs [3]	0.990	0.657	0.914	0.540	0.901
Ours (w/ SFRL)	**0.992**	**0.737**	**0.944**	**0.652**	**0.962**
Ours	**0.994**	**0.748**	**0.953**	**0.670**	**0.983**

4.3. Classification accuracy

In this section, we report the comparison to two mainstream event-based object classification solutions, namely point-based and frame-based methods, to show the advantages of our model comprehensively.

Comparison with point-based methods. As our proposed work also falls under the category of point-based methods, we firstly compare it to SOTA point-based models. As shown in Table 1, the proposed method outperforms SOTA point-based models consistently. Especially, our approach improves the model's performance by a large margin on four challenging datasets such as N-Cal, N-C, CIF10, and ASL. Surprisingly, when alternative the input from point-wise to voxel-wise representation, the method

Table 2. Comparison of different event-based classification models on the model complexity (#Params) and the number of FLOPs. † GFLOPs = 10^9 FLOPs. ‡ Using our proposed representation as input. ¶T(CPU) represents that both input generation and inference process run on CPU. T(GPU) means that input construction is on CPU and network inference is on GPU.

Method	#Params	GFLOPs†	T(CPU)¶	T(GPU)¶
EST [19]	21.38 M	4.28	27.1 ms	6.41 ms
M-LSTM [5]	21.43 M	4.82	34.8 ms	10.89 ms
MVF-Net [11]	33.62 M	5.62	42.5 ms	10.09 ms
AsyNet [30]	3.69 M	0.88	-	-
EventNet [43]	2.81 M	0.91	9.3 ms	3.35 ms
PointNet++ [47]	1.76 M	4.03	174.3 ms	103.85 ms
PointNet++‡ [47]	1.77 M	4.17	178.4 ms	107.97 ms
RG-CNNs [3]	19.46 M	0.79	1236 ms	-
Ours	**0.84 M**	**0.70**	26.1 ms	7.12 ms

in [47] achieves a significant performance gain, suggesting the effectiveness of our newly introduced event-based representation. Furthermore, Table 2 illustrates that our method shows huge advantages on the model and computational complexity, $e.g.$, our model can achieve 20 times parameter reduction and are with fewer FLOPs compared to the SOTA method RG-CNNs. We attribute the two-sided improvement to two designs in our model. (i) Our graph is constructed by setting an event voxel instead of a single event point as the vertex. This modification retains more powerful regional semantics than other strategies. The resulting compact and informative inputs largely ease the following network to learn distinguishable features in various scenarios. (ii) The *MFRL* module in our network can exploit semantics and motion cues from each vertex distinguishably concerning its distances to neighbors, allowing us to construct a shallow and lightweight network while achieving SOTA accuracy.

Moreover, we introduce a baseline model which replaces the *MFRL* with the *SFRL* module in the *EV-VGCNN*. In contrast to learning local and distant cues discriminatively, all neighbors of a vertex are equally treated in the *SFRL* to learn their correlations with shared parameters. For a fair comparison, we set N_{neigh} for each *SFRL* as the summation of N_{neigh}^{adj} and N_{neigh}^{dis} in *MFRL*. As shown, the *MFRL* consistently improves the performance on listed datasets, suggesting that the adopted mutli-scale learning strategy can effectively enhance the discriminativeness of features by considering the spatial-temporal variation across neighbors.

Comparison with frame-based methods. To have a comprehensive analysis, we also compare our method with several representative frame-based approaches as shown in Table 3. In particular, MVF-Net has achieved SOTA performance on different classification datasets. From the table, we can see that EST, M-LSTM, and MVF-Net have been consistently improved after using pretrained networks, especially on two datasets (N-Cal, CIF10) converted from traditional images. This is because that the frame-based classification models can take advantage of the weights pretrained on large-scale traditional image datasets ($e.g.$, ImageNet) [41]. However, without utilizing the prior knowledge from conventional images, our approach can still achieve comparable accuracy to these frame-based methods on N-M, N-C, and ASL datasets. More importantly, the proposed method obtains better results on all evaluated datasets than most frame-based methods trained from scratch. These comparisons demonstrate that our architecture and designs are greatly suitable for extracting distinguishable representations from event data.

Table 3. Comparison of the classification accuracy between ours and frame-based methods. † Results are acquired by using the classifier with Resnet-34 [21] as the backbone. ‡ We train these approaches from scratch and adopt the same training and testing sets used in this paper. Blue and green color indicate the first and second best performance.

Method	N-M	N-Cal	N-C	CIF10	ASL
Pretrained on ImageNet [9]					
EST [19]	0.991	0.837	0.925	0.749	0.991
M-LSTM [5] †	0.989	0.857	0.957	0.730	0.992
MVF-Net [11]	0.993	0.871	0.968	0.762	0.996
Without pretraining					
EST [19] ‡	**0.990**	**0.753**	0.919	0.634	0.979
M-LSTM [5] ‡	0.986	0.738	0.927	0.631	0.980
MVF-Net [11] ‡	0.981	0.687	0.927	0.599	0.971
AsyNet [30]	-	0.745	**0.944**	**0.663**	-
Ours	**0.994**	**0.748**	**0.953**	**0.670**	**0.983**

4.4. Complexity and computation analysis

We follow the calculation method described in [3, 20, 22, 32] to compute FLOPs for these methods. Since models' architecture may vary when evaluated on different datasets, we obtain results from these models on the same dataset N-Cal. The model complexity and the number of FLOPs of these frame-based methods are listed in Table 2, in which our approach is capable of performing classification with lower computational cost and fewer parameters. Compared to frame-based solutions which introduce much redundant information, our graph network learns decisive features directly from the sparse inputs, thus effectively relieving the learning pressure of the neural network. For instance, the 18-channel frame-based representation of samples from the N-Cal dataset with a spatial resolution of 180×240 used in [19] has 777600 input elements, while our proposed graph only contains 2048 ones.

In addition, we compute the averaged computation time for processing each sample in the dataset N-C and list the results in Table 2. We implement various methods on a workstation with a CPU (Intel i7), a GPU (GTX 1080Ti), and 64GB of RAM. From the table, we can find that the processing speed of our lightweight model is the same level with frame-based methods ($e.g.$ EST). However, our method shows weakness in computation speed compared to Event-

Table 4. Impact of different value of N_{neigh}^{adj} and N_{neigh}^{dis} on the performance evaluated on the N-Cal dataset.

	Value						
Variants	A	B	C	D	E	F	G
N_{neigh}^{adj}	10	10	10	5	20	5	20
N_{neigh}^{dis}	15	10	20	15	15	20	5
Accuracy	0.748	0.742	0.751	0.737	0.740	0.743	0.730

Net [43], which is developed on PointNet. We attribute this phenomenon to two points. (i) The integration operations for graph construction cost much computation time. (ii) The neighbor searching and feature embedding functions that do not exist in EventNet, though enlarge our model's performance by a large margin, also increase our computation time. Considering the low computational complexity (FLOPs) of our approach, we believe that there will be a large room for our method to speed up with the help of coding optimization. Moreover, though our approach cannot reach the temporal resolution of event data, the processing rate (only need 7.12 ms for each sample, which is equivalent to 140 Hz as frame-rate) is fast enough for most high-speed applications.

4.5. Ablation study

In this part, we conduct ablation studies to verify advantages of our voxel-wise graph representations and discuss the impact of hyper-parameters N_{Neigh}^{adj} and N_{Neigh}^{dis} to the system. Also, please refer to the supplementary material for details about the robustness of our model to the input vertex density.

The value setting for N_{Neigh}^{adj} and N_{Neigh}^{dis}. In this part, we set up a series of experiments on the N-Cal dataset to discuss how the variation of N_{Neigh}^{adj} and N_{Neigh}^{dis} affects the final performance of our model. Results of the controlled experiment are listed in Table 4. Comparing the settings A, B and C, we can find that when N_{Neigh}^{adj} is fixed, a larger value of N_{Neigh}^{dis} results in better performance. Intuitively, when we involve more distant neighbors to aggregate the features of a vertex, a denser neighborhood, carrying more global messages and cross-vertex spatio-temporal relationships, is encoded to aggregate the features for the central vertex. Differently, when we fix the N_{Neigh}^{dis} and change the value of N_{Neigh}^{adj} (e.g., settings A, D, and E) from 10 to 20, the final performance drops considerably. We argue that this is due to only a small number of adjacent neighbors being informative to characterize the local semantic information of a vertex. If a considerable part of adjacent neighbors is actually with large distance, then these "adjacent" neighbors are difficult to characterize this vertex's local semantics and tend to be interference. For the value chosen of these two hyper-parameters in this work, we firstly fix the summation of neighbors as 25 considering computational budget, then experimentally set N_{Neigh}^{adj} and N_{Neigh}^{dis} as 10 and 15 respectively according to the comparison among settings A, F and G.

Table 5. Comparison between voxel-wise and point-wise graph construction strategies with the same network architecture.

Vertex type	#Vertex	Accuracy	GFLOPs
Original events	2048	0.565	0.63
Original events	4096	0.601	0.66
Original events	8192	0.619	0.72
Event voxels (Ours)	2048	0.748	0.70

Comparison of graph construction strategies. To validate that our proposed voxel-wise graph is more effective in semantics encoding over point-wise inputs, this section performs comparisons of our voxel-wise graph to point-wise graph [3], which is constructed by selecting a random subset of event data as vertices and assigning the polarity of events to vertices as their features. We feed these two graphs to the same model *EV-VGCNN* and test their performance on the N-Cal dataset. The results in Table 5 show that with the same number of vertices (2048) inside, our graph construction strategy contributes to a significant accuracy gain, indicating that it encodes more informative features from the event data than the point-wise graph. We then increase the vertex number of the point-wise graph to 8192, which is much larger than ours. Even so, our method still has a considerable accuracy leading. We credit these superiorities to that our voxel-wise graph construction strategy enables the vertex to encode local correlations among points, thus carrying a more powerful representation for a point region. In contrast, although the compared point-wise graph reduces the complexity of the input, it leads to a severe loss of local 2D appearance and motion information in each vertex.

5. Limitation

First, this study is based on the assumption that the input event data to the model always relate to objects. When this assumption does not hold, our neighbor searching strategy in Euclidean space may not be able to find proper neighbors for vertices. Second, the potential of our model has not been fully facilitated at the current stage due to the lack of a prior knowledge support from large datasets. Third, the adopted synchronous processing pattern can normally provide more robust global features compared to asynchronous methods [30, 43] while inevitably sacrifice real-time performance.

6. Conclusion

In this work, we introduce a novel graph-based learning framework for event data. The proposed voxel-wise graph construction method retains more local information than previous point-wise methods while maintaining the sparsity of event data. Moreover, we tailor a *MFRL* module to explore spatial-temporal relationships between vertices and their neighbors discriminatively. Extensive experiments show the advantages of our designs, as well as the elegant improvement on both accuracy and model complexity achieved by the proposed lightweight *EV-VGCNN*.

References

[1] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, et al. A low power, fully event-based gesture recognition system. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7243–7252, 2017. 3

[2] Yin Bi, Aaron Chadha, Alhabib Abbas, Eirina Bourtsoulatze, and Yiannis Andreopoulos. Graph-based object classification for neuromorphic vision sensing. In *IEEE/CVF Int. Conf. Comput. Vis.*, pages 491–501, 2019. 6

[3] Y. Bi, A. Chadha, A. Abbas, E. Bourtsoulatze, and Y. Andreopoulos. Graph-based spatio-temporal feature learning for neuromorphic vision sensing. *IEEE Trans. Image Process.*, pages 1–1, 2020. 1, 3, 6, 7, 8

[4] Tobias Brosch, Stephan Tschechne, and Heiko Neumann. On event-based optical flow detection. *Front. Neurosci.*, 9:137, 2015. 2

[5] Marco Cannici, Marco Ciccone, Andrea Romanoni, and Matteo Matteucci. A differentiable recurrent surface for asynchronous event-based data. In *Eur. Conf. Comput. Vis.*, August 2020. 1, 2, 7

[6] Xavier Clady, Sio-Hoi Ieng, and Ryad Benosman. Asynchronous event-based corner detection and matching. *Neural Netw.*, 66:91–106, 2015. 2

[7] Xavier Clady, Jean-Matthieu Maro, Sébastien Barré, and Ryad B Benosman. A motion-based feature for event-based pattern recognition. *Front. Neurosci.*, 10:594, 2017. 2

[8] Paulo de Oliveira Rente, Catarina Brites, João Ascenso, and Fernando Pereira. Graph-based static 3d point clouds geometry coding. *IEEE Trans. Multimedia*, 21(2):284–299, 2019. 2

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 248–255, 2009. 7

[10] Yongjian Deng, Hao Chen, Huiying Chen, and Youfu Li. Learning from images: A distillation learning framework for event cameras. *IEEE Trans. Image Process.*, pages 1–1, 2021. 2

[11] Yongjian Deng, Hao Chen, and Youfu Li. Mvf-net: A multi-view fusion network for event-based object classification. *IEEE Trans. Circuits Syst. Video Technol.*, pages 1–1, 2021. 2, 7

[12] Y. Deng, Y. Li, and H. Chen. Amae: Adaptive motion-agnostic encoder for event-based object classification. *IEEE Robot. Autom. Lett.*, 5(3):4596–4603, 2020. 1, 2

[13] Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International Joint Conference on Neural Networks*, pages 1–8. IEEE, 2015. 3

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *Int. Conf. Learn. Represent.*, 2021. 2

[15] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(4):594–611, 2006. 6

[16] Zeqing Fu and Wei Hu. Dynamic point cloud inpainting via spatial-temporal graph learning. *IEEE Transactions on Multimedia*, pages 1–1, 2021. 5

[17] G. Gallego, T. Delbruck, G. M. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza. Event-based vision: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 1–1, 2020. 1

[18] Daniel Gehrig, Mathias Gehrig, Javier Hidalgo-Carrió, and Davide Scaramuzza. Video to events: Recycling video datasets for event cameras. In *IEEE Conf. Comput. Vis. Pattern Recog.*, June 2020. 2

[19] Daniel Gehrig, Antonio Loquercio, Konstantinos G Derpanis, and Davide Scaramuzza. End-to-end learning of representations for asynchronous event-based data. In *IEEE/CVF Int. Conf. Comput. Vis.*, pages 5633–5643, 2019. 1, 2, 4, 7

[20] Kaiming He and Jian Sun. Convolutional neural networks at constrained time cost. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5353–5360, 2015. 7

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 770–778, 2016. 7

[22] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 7

[23] Yuhuang Hu, Tobi Delbruck, and Shih-Chii Liu. Learning to exploit multiple vision modalities by using grafted networks. In *Eur. Conf. Comput. Vis.*, pages 85–101, 2020. 2

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[25] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Int. Conf. Learn. Represent.*, pages 1–35, 2016. 5

[26] Xavier Lagorce, Garrick Orchard, Francesco Galluppi, Bertram E Shi, and Ryad B Benosman. Hots: a hierarchy of event-based time-surfaces for pattern recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(7):1346–1359, 2016. 2, 6

[27] Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: An event-stream dataset for object classification. *Front. Neurosci.*, 11:309, 2017. 6

[28] P. Lichtsteiner, C. Posch, and T. Delbruck. A 128× 128 120 db 15 μs latency asynchronous temporal contrast vision sensor. *IEEE J. Solid-State Circuit*, 43(2):566–576, Feb 2008. 1

[29] Ana I Maqueda, Antonio Loquercio, Guillermo Gallego, Narciso García, and Davide Scaramuzza. Event-based vision meets deep learning on steering prediction for self-driving cars. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5419–5427, 2018. 2

[30] Nico Messikommer, Daniel Gehrig, Antonio Loquercio, and Davide Scaramuzza. Event-based asynchronous sparse convolutional networks. In *Eur. Conf. Comput. Vis.*, pages 415–431. Springer, 2020. 2, 7, 8

[31] Anton Mitrokhin, Zhiyuan Hua, Cornelia Fermuller, and Yiannis Aloimonos. Learning visual motion segmentation using event surfaces. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 14414–14423, 2020. 1, 2, 3

[32] P Molchanov, S Tyree, T Karras, T Aila, and J Kautz. Pruning convolutional neural networks for resource efficient inference. In *Int. Conf. Learn. Represent.*, 2019. 7

[33] Elias Mueggler, Chiara Bartolozzi, and Davide Scaramuzza. Fast event-based corner detection. In *British Mach. Vis. Conf*, 2017. 2

[34] Garrick Orchard, Ajinkya Jayawant, Gregory K Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Front. Neurosci.*, 9:437, 2015. 6

[35] G. Orchard, C. Meyer, R. Etienne-Cummings, C. Posch, N. Thakor, and R. Benosman. Hfirst: A temporal approach to object recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(10):2028–2040, Oct 2015. 3, 6

[36] José Antonio Pérez-Carrasco, Bo Zhao, Carmen Serrano, Begona Acha, Teresa Serrano-Gotarredona, Shouchun Chen, and Bernabé Linares-Barranco. Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing–application to feedforward convnets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(11):2706–2719, 2013. 3

[37] Christoph Posch, Daniel Matolin, and Rainer Wohlgenannt. A qvga 143 db dynamic range frame-free pwm image sensor with lossless pixel-level video compression and time-domain cds. *IEEE J. Solid-State Circuit*, 46(1):259–275, 2010. 1

[38] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 652–660, 2017. 1, 5

[39] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Conf. Neural Inf. Process. Syst.*, pages 5099–5108, 2017. 5

[40] Henri Rebecq, Rene Ranftl, Vladlen Koltun, and Davide Scaramuzza. High speed and high dynamic range video with an event camera. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 1–1, 2019. 2

[41] 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.*, pages 1–1, 2019. 7

[42] Sajjad Seifozzakerini, Wei-Yun Yau, Bo Zhao, and Kezhi Mao. Event-based hough transform in a spiking neural network for multiple line detection and tracking using a dynamic vision sensor. In *British Mach. Vis. Conf*, 2016. 2

[43] Yusuke Sekikawa, Kosuke Hara, and Hideo Saito. Eventnet: Asynchronous recursive event processing. In *IEEE Conf. Comput. Vis. Pattern Recog.*, June 2019. 1, 3, 6, 7, 8

[44] Amos Sironi, Manuele Brambilla, Nicolas Bourdis, Xavier Lagorce, and Ryad Benosman. Hats: Histograms of averaged time surfaces for robust event-based object classification. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1731–1740, 2018. 2, 6

[45] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Int. Conf. Mach. Learn*, pages 1139–1147, 2013. 6

[46] Lin Wang, Yujeong Chae, Sung-Hoon Yoon, Tae-Kyun Kim, and Kuk-Jin Yoon. Evdistill: Asynchronous events to end-task learning via bidirectional reconstruction-guided cross-modal knowledge distillation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 608–619, June 2021. 2

[47] Qinyi Wang, Yexin Zhang, Junsong Yuan, and Yilong Lu. Space-time event clouds for gesture recognition: From rgb cameras to event cameras. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1826–1835, 2019. 1, 3, 6, 7

[48] Jinjian Wu, Chuanwei Ma, Leida Li, Weisheng Dong, and Guangming Shi. Probabilistic undirected graph based denoising method for dynamic vision sensor. *IEEE Trans. Multimedia*, pages 1–1, 2020. 2

[49] A. Zhu, Z. Wang, K. Khant, and K. Daniilidis. Eventgan: Leveraging large scale image datasets for event cameras. In *IEEE Conf. Comput. Photography*, pages 1–11, Los Alamitos, CA, USA, may 2021. IEEE Computer Society. 4

[50] Alex Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Ev-flownet: Self-supervised optical flow estimation for event-based cameras. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018. 1, 2

[51] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Unsupervised event-based optical flow using motion compensation. In *Eur. Conf. Comput. Vis.*, pages 0–0, 2018. 1, 2, 4