Asynchronous Event Processing with Local-Shift Graph Convolutional Network

Linhui Sun[1,2,3], **Yifan Zhang**[1,2,3,*], **Jian Cheng**[1,2,3], **Hanqing Lu**[1,2]

[1] Institute of Automation, Chinese Academy of Sciences, 100190, Beijing, China
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences, 100049, Beijing, China
[3] AIRIA, 211135, Nanjing, China
sunlinhui2018@ia.ac.cn, yfzhang@nlpr.ia.ac.cn, jcheng@nlpr.ia.ac.cn, luhq@nlpr.ia.ac.cn

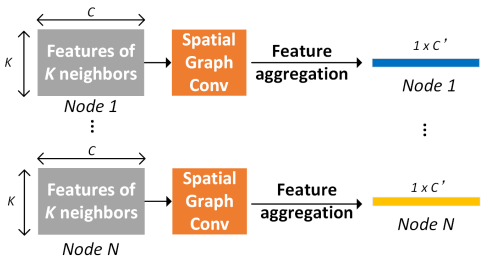
Abstract

Event cameras are bio-inspired sensors that produce sparse and asynchronous event streams instead of frame-based images at a high-rate. Recent works utilizing graph convolutional networks (GCNs) have achieved remarkable performance in recognition tasks, which model event stream as spatio-temporal graph. However, the computational mechanism of graph convolution introduces redundant computation when aggregating neighbors features, which limits the low-latency nature of the events. And they perform a synchronous inference process, which can not achieve a fast response to the asynchronous event signals. This paper proposes a local-shift graph convolutional network (LSNet), which utilizes a novel local-shift operation equipped with a local spatio-temporal attention component to achieve efficient and adaptive aggregation of neighbors features. To improve the efficiency of pooling operation in feature extraction, we design a node-importance based parallel pooling method (NIPooling) for sparse and low-latency event data. Based on the calculated importance of each node, NIPooling can efficiently obtain uniform sampling results in parallel, which retains the diversity of event streams. Furthermore, for achieving a fast response to asynchronous event signals, an asynchronous event processing procedure is proposed to restrict the network nodes which need to recompute activations only to those affected by the new arrival event. Experimental results show that the computational cost can be reduced by nearly 9 times through using local-shift operation and the proposed asynchronous procedure can further improve the inference efficiency, while achieving state-of-the-art performance on gesture recognition and object recognition.

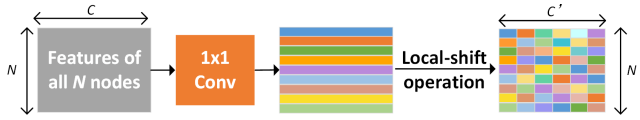
1 Introduction

Event camera (Lichtsteiner et al. 2008; Brandli et al. 2014) is bio-inspired sensor that utilizes asynchronous and sparse event streams to represent spatio-temporal visual information. It asynchronously triggers an individual event when the brightness changes of a pixel exceed a preset threshold, which distinguishes it from frame-based cameras. Benefiting from the mechanism of generating data, compared with traditional cameras, event cameras own many attractive properties, including low latency, low power, high temporal resolution (s), and high dynamic range (140 dB vs 60 dB of traditional cameras). Due to these excellent properties, event cameras are more advantageous over traditional cameras for visual tasks that require fast response, limited power consumption, robustness to high speed motion, and variable lighting. Therefore, event cameras have a wide range of applications, such as gesture recognition (Amir et al. 2017; Li et al. 2021; Yao et al. 2021), object recognition (Cannici et al. 2019; Sironi et al. 2018; Deng et al. 2022), optical flow estimation (Ding et al. 2022; Hu et al. 2022), pose relocalization (Nguyen et al. 2019; Sekikawa et al. 2019), video reconstruction (Rebecq et al. 2021a,b), autonomous driving (Cheng et al. 2019; Chen et al. 2020a), etc.

For taking advantage of event-based data in downstream tasks, designing models to effectively extract features from sparse and asynchronous event streams is a key step. Previous methods (Liu et al. 2020; Amir et al. 2017) introduce spiking neural networks (SNNs) for event processing, which exploit the sparsity nature of event data and can asynchronously respond to events. However, these methods have limited performance in high-level tasks mainly due to the lacking of general and robust learning rules. Since GCNs and PointNet-like frameworks are suitable for feature extraction from sparse inputs, recent works (Li et al. 2021;



(a) Spatial graph convolutional network.



(b) Local-shift graph convolutional network.

Figure 1: Feature aggregation way of spatial graph convolutional network and local-shift graph convolutional network.

*Corresponding author
Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Sekikawa et al. 2019; Messikommer et al. 2020) combine GCNs or PointNet with event-by-event processing. However, when aggregating features for each node, the spatial graph convolutions involved in these methods need to process the features of all its neighbors. Specifically, as shown in Fig.1a, if a node is a neighbor of multiple nodes, its features will be processed multiple times, which incurs significant computation consumption.

Based on these observations, to effectively extract spatio-temporal information from sparse event streams, this paper proposes a local-shift graph convolutional network, namely LSNet, which processes event sequence as a spatio-temporal graph. To improve feature extraction efficiency, we replace the spatial graph convolution with lightweight point-wise convolution and a novel local-shift operation which shifts information from neighbor nodes to the current one for feature aggregation. Specifically, each node in the graph only needs to be performed point-wise convolution operation one time to obtain its independent features. Then, the local-shift operation aggregates neighbors features across channel dimensions, as shown in Fig.1b. However, shifting features of different neighbors to the current node equally ignores the relationship between it and different neighbors. Therefore, a local spatio-temporal attention component is proposed, which adaptively determines the contribution degree of each neighbor when performing the local-shift operation.

To perform hierarchical learning and improve feature extraction efficiency, previous graph-based methods (Li et al. 2021; Bi et al. 2020; Deng et al. 2022) introduce graph pooling operations to obtain a few representative nodes, such as cluster-based pooling, random pooling and pooling through farthest point sampling (FPS). However, cluster-based pooling obtains the downscaling factor based on the resolution of events and preset cluster size, like pooling operation in convolutional neural networks (CNNs), which is not suitable for the sparse event stream. Although random pooling can fast obtain the pooling results, the sampled nodes cannot retain the diversity of events. Pooling through FPS can cover the spatio-temporal coordinates of sparse events, but the serial sampling process introduces huge time consumption. This paper proposes a node-importance based parallel pooling method (NIPooling) which calculates the importance of each node and obtains uniform sampling results covering the importance range in parallel. In this way, we can efficiently obtain representative nodes that retain the diversity of events.

To achieve a fast response to asynchronous event signals, an asynchronous event processing procedure is designed for the LSNet. When a new event arrives, the proposed asynchronous procedure can reuse the previous calculations according to the connectivity relationship between nodes to avoid recalculating all nodes features in the reconstructed graph. Specifically, our method only needs to compute the changed activations around the new added node at the lowest layer and propagates these changes to higher layers, which achieves asynchronous response to event streams in a nearly latency-free and power-saving way.

The contributions are summarized as follows:

- We propose a local-shift graph convolutional network (LSNet) to adaptively shift information from neighbor nodes to the current one to reduce computational cost.
- We propose a node-importance based parallel pooling method, namely NIPooling, which efficiently obtains uniform sampling results covering the importance range.
- We propose an asynchronous event processing procedure that achieves a fast asynchronous response to event data.
- Experiments show that the proposed method achieves state-of-the-art results on object recognition and gesture recognition, while significantly reducing the computational complexity compared with previous methods.

2 Related Work

2.1 Event-Based Approaches

Motivated by the huge success achieved by CNNs in traditional computer vision, many previous methods compress event streams into 2D frames (Cannici et al. 2020; Nguyen et al. 2019; Rebecq et al. 2019) or 3D voxel grids (Gehrig et al. 2019; Pan et al. 2020) to utilize existing efficient methods based on dense CNNs. However, such data representation transformations discard the sparse and asynchronous nature of event streams, resulting in redundant computation.

Some methods propose to exploit the sparse nature of events through sparse network architectures, such as SNNs, PointNet-like frameworks, and GCNs. Although SNNs are bio-inspired designed and tailored to process asynchronous event streams, SNNs-based methods (Liu et al. 2020; Shrestha et al. 2018) obtain limited performance due to the lacking of training technology. Some methods treat event streams as event clouds and utilize PointNet-like frameworks or GCNs for better performance. (Wang et al. 2019) utilize PointNet++ (Qi et al. 2017) to extract local and global features for gesture recognition. (Bi et al. 2020; Chen et al. 2020b; Mitrokhin et al. 2020; Rebecq et al. 2021b) utilize GCNs to exploit the topological structure of event clouds by interpreting them in the form of spatio-temporal graphs. (Bi et al. 2020) utilize the properties of B-spline bases to filter the graph inputs and prove that such graph representation requires less computation and memory than CNNs. (Deng et al. 2022) propose a voxel-wise graph representation and a multi-scale feature relational layer to extract spatial and motion cues. However, these methods perform a synchronous inference process, which means that for each new arrival event, they need to recompute all network activations, but a single event only indicates a per-pixel change independently.

To avoid the redundant computation caused by recalculating all activations, recent methods propose to reuse network activations and achieve event-by-event inference. EventNet (Sekikawa et al. 2019) designs a recursive and event-wise manner to process event streams. (Messikommer et al. 2020) propose a general framework to convert synchronous models into asynchronous models through applying efficient recursive update rules. However, EventNet (Sekikawa et al. 2019) lacks hierarchical learning and local information extraction, limiting its scalability to high-level tasks, and the input representation of (Messikommer et al. 2020) discards the temporal information of event data. (Li et al. 2021) introduce a graph-based recursive algorithm for efficient event-

wise processing. However, when aggregating features for each node, the spatial graph convolution involved in it needs to process all neighbors features, which introduces repeated computation of node features. Therefore, this paper proposes the LSNet to effectively extract features from sparse event streams, and an asynchronous event processing procedure to achieve a fast asynchronous response to event data.

2.2 Shift Convolution Operators

Recently, shift operations (Jeon et al. 2018; Wu et al. 2018; Zhong et al. 2018) are utilized to replace expensive spatial convolutions to reduce the number of parameters and computational complexity. However, these methods are designed for standard CNNs and are not compatible with sparse event data. Shift-GCN (Cheng et al. 2020) successfully applied shift to GCN for skeleton-based action recognition. However, for the proposed non-local shift graph operation, the receptive field of each node covers the full skeleton graph, which can not be used to process events with a large number. And the proposed shift operation solves the computational redundancy in computing features using three kinds of adjacent matrix for each node. We propose a novel local-shift operation, which avoids processing a node multiple times when aggregating features to reduce computational complexity.

3 Preliminaries

3.1 Event Camera Model

Event cameras respond to changes of logarithmic brightness $L(u_i, t_i) = logI(u_i, t_i)$ between timestamp t_i and t_{i-1} asynchronously and independently at each pixel $u_i = (x_i, y_i)$:

$$\Delta L = L(u_i, t_i) - L(u_i, t_{i-1}) \tag{1}$$

An event $e_i = (x_i, y_i, t_i, p_i)$ will be triggered immediately when the ΔL exceeds a preset threshold C (with $C > 0$) which is a camera parameter. Each event encodes the pixel location (x_i, y_i), trigger time t_i, and polarity of the brightness change $p_i \in \{-1, 1\}$:

$$p = \begin{cases} 1, \Delta L \geqslant C \\ -1, \Delta L \leqslant -C \end{cases} \tag{2}$$

A group of events in a time window T can be expressed as a sequence of events:

$$\{e_i\}_{N_e} = \{x_i, y_i, t_i, p_i\}_{N_e} \tag{3}$$

where N_e is the number of events in the sliding window.

3.2 Event Graph and Spatial Graph Convolution

A directed event graph can be constructed for each sliding window, which is denoted as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with $\mathcal{V}$ nodes and $\mathcal{E}$ directed edges. Each event is a node v_i in the event graph, which contains the spatio-temporal coordinate (x_i, y_i, t_i) and the node attribute (p_i). For constructing the connectivity relationship between nodes, this paper introduces the ball-query strategy (Qi et al. 2017). Specifically, the neighbor nodes connected to node v_i are the first K neighbor nodes whose Euclidean distance between v_i is less than radius distance R. And each edge connecting v_i and v_j has its own attribute e_{ij}, which is obtained by calculating the relative Cartesian coordinates of the linked nodes v_i and v_j.

After constructing the event graph, spatial graph convolution in previous methods (Li et al. 2021; Bi et al. 2020) operate on local neighborhood graphs, which aggregates features for each node by weighting its neighbors features through a trainable function. However, it can be found that if a node is a neighbor of multiple nodes, its features will be processed multiple times for feature aggregation, which brings redundant computation. The proposed local-shift operation can solve this problem.

4 Method

For effectively extracting the spatio-temporal information from sparse event streams, this paper proposes a local-shift convolutional network (LSNet), which adaptively aggregates features for each node through a local-shift operation equipped with a local spatio-temporal attention component. In addition, a node-importance based parallel pooling method (NIPooling) is proposed to efficiently sample representative nodes covering the importance range, based on the calculated importance of each node. Furthermore, for achieving a fast response to asynchronous event signals, we propose an asynchronous event processing procedure.

4.1 Local-Shift Convolutional Network

The proposed LSNet comprises three kinds of layers: local-shift layer, global aggregation layer, and classifier layer, as shown in Fig.2.

Local-shift layer. To obtain multi-scale features for each node, the local-shift layer contains two parallel feature extraction branches, the upper and lower paths shown in Fig.2. Each branch contains two 1×1 convolutions for extracting features of all nodes in the event graph. Then the local-shift operation is performed to aggregate neighbors features. There are two differences between the two branches. The first one is that the receptive fields of nodes are different, that is, different R and K are set in the ball-query strategy when constructing event graph. The second one is that the numbers of channels in the 1×1 convolutions are different, which are set larger for the branch with larger receptive field. Features extracted from the two branches are concatenated to obtain the multi-scale features of each node.

In local-shift operation, let v_i denotes a node in the graph and $F_i^{nei} \in \mathbb{R}^{N_K \times C_{n1}}$ represents a feature map of its N_K neighbor nodes. Neighbors are sorted in ascending order of distance from v_i and are numbered starting with 0. The feature in c_{th} channel of v_i is replaced by feature in the same channel location of the c_{th} mod N_K neighbor node, as shown in Fig.3. After shifting, the aggregated features of v_i are formed by alternating the features of its neighbors in the channel dimension.

However, there are two shortcomings of shifting features as described above. Firstly, the shifting process ignores the relationship between v_i and its different neighbor nodes.

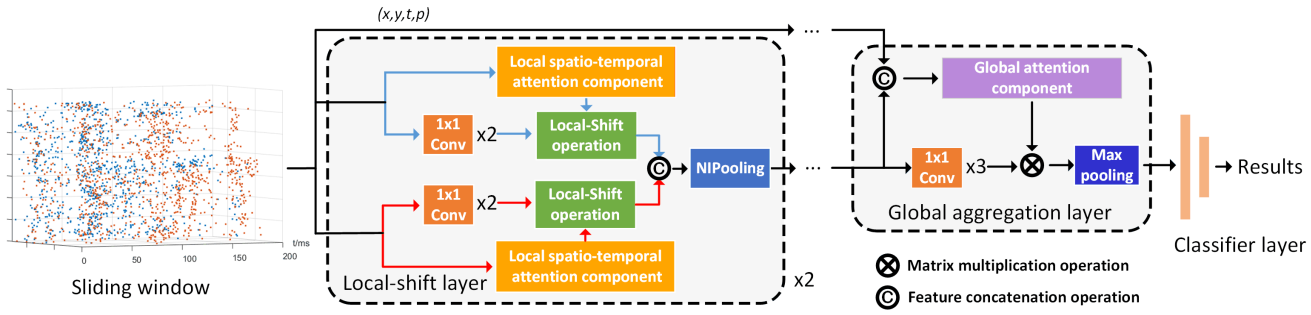


Figure 2: Overview of the proposed LSNet, which contains two local-shift layers, one global aggregation layer and one classifier layer. The local-shift layer extracts multi-scale features for each node and performs the local-shift operation to aggregate features. Blue arrows represent feature extraction and local-shift operation are performed on small receptive field, and red arrows represent large receptive field. The global aggregation layer aims at obtaining high-level global features through global max pooling operation. The classifier layer contains two fully connected layers for predicting results.

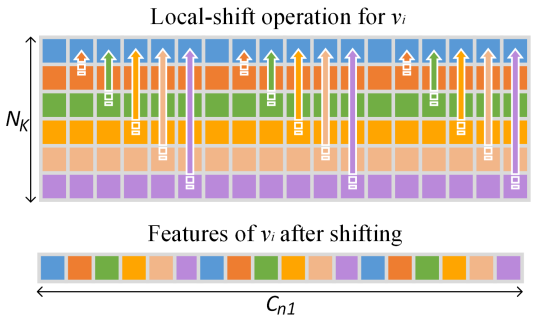


Figure 3: Local-shift operation for v_i. For intuitive representation, v_i has 6 neighbors including itself, and the number of feature channels is 18.

Secondly, the neighbor information is asymmetric. Specifically, different nodes have different frequencies of becoming neighbors of other nodes. Nodes with higher frequency appear more times in the shifting process, while the features of nodes with low frequency may be abandoned.

To solve the first problem, a local spatio-temporal component is proposed, which can adaptively determine the contribution degree of neighbors when performing the local-shift operation. It contains two concatenated 1×1 convolutions to obtain features of all nodes $F^{l_{att}} \in \mathbb{R}^{N \times C_{l_{att}}}$ in graph, and determines the contribution degree of different nodes in each local-neighbor graph according to the feature response strength. The symbol n denoting layer number is omitted for clarity. Let $f^{l_{att}}(i) \in \mathbb{R}^{1 \times C_{l_{att}}}$ denotes the features of v_i. $f_i^{l_{att}}(j)$ and $A_i(j)$ represent the features and the calculated contribution degree of its j_{th} neighbor, respectively:

$$A_i(j) = \frac{|f_i^{l_{att}}(j) - f^{l_{att}|}(i)}{\sum_{j \in \mathcal{N}(i)}[|f_i^{l_{att}}(j) - f^{l_{att}}(i)|]} \tag{4}$$

where $\mathcal{N}(i)$ is the neighbor nodes set of v_i. The calculated contribution degree of the neighbor nodes will be multiplied with the corresponding features during the local-shift operation to achieve adaptive feature aggregation.

To solve the second problem, for v_i, the max pooling operation is performed on the shifted features $f^{shift}(i)$ and the original features $f^{ori}(i)$ of it in the channel dimension to obtain the final features. In this way, the features with large responses in $f^{shift}(i)$ and $f^{ori}(i)$ at the same channel location will be retained to avoid that important original features are discarded due to the asymmetry of neighbor information.

Global aggregation layer. This layer adopts three 1×1 convolutions for feature extraction and introduces a global attention component for feature aggregation. The attention component utilizes two 1×1 convolutions to extract features $f^{g_{att}} \in \mathbb{R}^{1 \times C_{g_{att}}}$ for N_g input nodes, which are utilized to calculate the contribution degree:

$$G_A(j) = \frac{f^{g_{att}}(j)}{\sum_{j=0}^{N_g} f^{g_{att}}(j)} \tag{5}$$

Finally, each feature vector is multiplied by the calculated weight before being aggregated through a global max pooling operation to obtain a one-dimensional feature vector.

Classifier layer. Finally, the high-level global feature vector is fed into two fully connected layers for classification.

4.2 Node-Importance based Parallel Pooling Method

To perform hierarchical learning and improve feature extraction efficiency, graph-based methods introduce graph pooling operations to obtain a few representative nodes from the input. However, the serial sampling process in FPS introduces huge time consumption, and the sampling results of random pooling are clustered in the area with high density, which fails to retain the diversity of events. This paper proposes a node-importance based parallel pooling method (NIPooling), in which the importance of a node means its clustering degree. To be exact, if a node is close to its neighbors, it will have a high clustering degree and importance. The process of NIPooling can be divided into three steps.

1. The importance of each node is calculated based on the Euclidean distance obtained in the ball-query strategy.

Let N_{in} and N_{out} denote the number of nodes before and after pooling operation, respectively. Note that N_{in} is divisible by N_{out}. The distances between v_i and its $(N_{in}/N_{out})^2$ nearest neighbors are summed, the smaller the result, the higher the importance of v_i.

2. All nodes are ranked in decreasing order of importance.
3. To obtain uniform sampling results covering the importance range, indexes are generated starting from 0 with the step of N_{in}/N_{out}. According to the indexes and ranked results, the sampled nodes are finally obtained.

Through the proposed NIPooling, the pooling operation is performed in parallel, which improves the sampling efficiency. And the sampled nodes cover the importance range, which retains the diversity of event data.

4.3 Asynchronous Event Processing Procedure

GCNs-based methods process sparse events in the sliding window for constructing an event graph. When the time window is slid from $t-1$ to t, new events will enter the window, and old events will leave. Although an event only measures single pixel changes independently, previous methods need to recompute all network activations when a new event arrives, which introduces redundant calculations. This paper proposes an asynchronous event processing procedure for LSNet to restrict the nodes needed to be recomputed only to those affected by the new arrival event, based on the connectivity relationship between nodes.

As a new event arrives, a new node v_{new} and new edges connecting it with existing nodes will be added to the event graph, in which the oldest node v_{old} and its edges are abandoned. For reusing the previous calculated activations, a feature propagation rule is proposed, which guides the propagation of feature update from input layer to deeper layer. At the first layer $layer_1$ of the LSNet ($layer_0$ is the input layer), v_{new} or v_{old} only affects the nodes treating it as neighbor. Features of v_{new} are obtained through extracting multi-scale features and performing local-shift operation, according to the local-neighborhood graph of it. For other nodes affected by v_{new} or v_{old} at $layer_1$, they need to perform the shift operation based on the new neighbor relationship to update features. For deeper layers $layer_n$, the affected nodes are those that treat the updated nodes at $layer_{n-1}$ as neighbors.

Pooling operation aims at obtaining a few representative nodes to perform hierarchical learning and improve feature extraction efficiency. Therefore, two types of nodes are regarded as the newly added nodes, including the new sampled nodes and the nodes both sampled at $t-1$ and t whose features are updated. For the nodes sampled at $t-1$ and are not sampled at t, they and their edges are discarded from the graph. Then the features of the affected nodes can be updated according to the above feature propagation rule.

Thanks to the asynchronous event processing procedure, when a new node is inserted into the event graph, the network activations are updated locally, which greatly reduces computation complexity to achieve asynchronous response to event in a nearly latency-free and power-saving way.

4.4 Training and Testing Strategies

In the training phase, since the feature update achieved by asynchronous event processing procedure is equivalent to recalculating the full graph nodes, the LSNet can be trained on batches of sliding windows through backpropagation. In this paper, sliding windows are obtained based on a fixed time interval T. The cross-entropy loss function with label smoothing is adopted for training. In the inference phase, for each event stream, events triggered before T construct the initial event graph, and the LSNet is applied to the full graph to initialize the features of each node. Then when new event arrives, the asynchronous event processing procedure is introduced to update the features, which achieves a fast response to the asynchronous event signals.

5 Experiments

5.1 Experimental Setup

Datasets. Our methods are evaluated on four commonly used event-based datasets, including DVS128 Gesture Dataset (Amir et al. 2017), N-Cars (Sironi et al. 2018), MNIST-DVS (Orchard et al. 2015a), and CIFAR10-DVS (H et al. 2017). The DVS128 Gesture Dataset records 1342 instances of 11 gestures. The N-Cars dataset contains 12336 car samples and 11693 background samples. These two datasets are collected by event cameras in real world environments. Differently, MNIST-DVS and CIFAR10-DVS are converted from the existing frame-based datasets MNIST dataset (Lecun et al. 1998) and CIFAR10 dataset (Krizhevsky 2009), respectively. The MNIST-DVS contains 30000 samples, which are obtained by displaying 10000 converted samples at three scales. The CIFAR10-DVS dataset converts 10000 samples of 10 categories.

Implementation details. The local-shift layer contains two parallel feature extraction branches to obtain multi-scale features from different receptive fields. For the first local-shift layer, K and R in ball-query strategy are set as 8, 0.06 and 16, 0.12 for the upper and low paths, respectively. For the second layer, K and R are set as 16, 0.12 and 32, 0.24, respectively. In pooling layers, the number of representative nodes are 512 and 256. The proposed method is implemented by PyTorch, which is trained on TITAN RTX GPU. The batch size is 64 and Adam optimizer (Kingma and Ba 2015) is adopted with an initial learning rate of 0.001 multiplied by 0.5 every 20 epochs. More details about the network architecture are shown in the supplementary material.

Metrics. For object recognition and gesture recognition tasks, prediction accuracy is adopted. For evaluating the computational complexity, the giga and million floating-point operations of the network (GFLOPs and MFLOPs) and the million floating-point operations per event (MFLOPs/event) are introduced.

5.2 Ablation Study

For verifying the reduction of computational complexity brought by the proposed LSNet and asynchronous event processing procedure, as well as the performance gain brought

Method	Local-shift	L-att	G-att	Acc(%)	MFLOPs
Baseline				95.45	940.9
Shift	✓			95.51	**105.1**
	✓	✓		95.78	110.8
	✓	✓	✓	**96.10**	117.2

Table 1: Contribution of local-shift operation and each attention component, evaluated on N-Cars dataset. Local-shift, L-att, and G-att refer to the proposed local-shift operation, local spatio-temporal attention component, and global attention component, respectively.

by the local spatio-temporal attention component, global attention component and NIPooling, we conduct ablation experiments on the N-Cars dataset. Each sliding window with T set as $50ms$ in N-Cars is randomly sampled 1024 events for processing.

Local-shift convolutional network. For verifying the efficiency and accuracy improvement brought by the local-shift operation and each attention component, architectures with various settings are introduced, as shown in Table 1. Baseline means the network has a similar structure as LSNet removing two attention components, except that it performs spatial graph convolutions on the local-neighborhood graph of each node to aggregate features. Note that for a fair comparison, architectures in Table 1 all adopt the proposed NIPooling. As shown in the first two rows of Table 1, the proposed local-shift operation reduces the MFLOPs by $9\times$, while achieving better accuracy (95.51% vs 95.45%). And after introducing the proposed two attention components, the accuracy is further improved (96.10%) with a small increase in computational complexity, which confirms that the two lightweight attention components can adaptively guide the feature aggregation to obtain a better performance.

Node-importance based parallel pooling method. For verifying the effectiveness of the NIPooling, two additional pooling methods are introduced to the LSNet, as shown in Table 2. Random performs random sampling from input to obtain representative nodes. FPS means that the pooling results are obtained by utilizing farthest point sampling. Table 2 shows that the NIPooling achieves better accuracy than FPS (96.10% vs 95.88%) while improving the speed by nearly $780\times$. The results prove that based on the importance of each node, the NIPooling obtains higher quality representative nodes than only relying on node coordinates. And the parallel operation significantly reduces the sampling time, which is beneficial to the low-latency nature of event data. In addition, the accuracy of NIPooling is 0.29% higher than that of Random, which proves that the sampled nodes preserving the diversity of events obtained by NIPooling can provide richer features for prediction.

Asynchronous event processing procedure and hyperparameter settings. Since K and R in ball-query strategy influence the connectivity of nodes in the graph, which affect the accuracy and efficiency of the network as well as the number of nodes needed to be updated in the asynchronous procedure, ablation experiments are conducted. Because the LSNet consists of tow shift-layers containing multi-scale feature extraction, for reducing the number of hyperparameters, we set the ratio between the four K and the ratio between K and R. Following the setting in (Qi et al. 2017), let K_{min} denotes the minimum K at the first layer, then the other K of the first layer is $2K_{min}$. And the two K at the second layer are $2K_{min}$ and $4K_{min}$, respectively. For the setting of R, we adjust R according to the ratio of $K = 16, R = 0.12$. In experiments, SLSNet means that the LSNet performs a synchronous update process in inference, that is, when a new event arrivals all nodes activations are recomputed. ALSNet refers to utilizing the asynchronous event processing procedure. As shown in Table 3, since ALSNet introduces asynchronous processing, it reduces MFLOP/s by $78\times$, $51\times$, and $33\times$ compared to SLSNet, respectively. With K increasing, due to ALSNet processing the affected nodes, the MFLOPs/event is increased. Although the MFLPOs/event is only 1.1 when K_{min} is set as 4, the prediction accuracy is sacrificed (95.74%), due to insufficient neighbor information. By setting K_{min} to 8, ALSNet can extract sufficient local information and obtains the best performance 96.10%, striking a balance between accuracy and computation cost. If K_{min} is set too large, interference information may be introduced when aggregating features, resulting in decreased accuracy 95.95%. In subsequent experiments, K_{min} is set to 8 to balance accuracy and efficiency.

Pooling method	Random	FPS	NIPooling
Accuracy(%)	95.81	95.88	**96.10**
Average pooling time(ms)	**0.11**	85.81	**0.11**

Table 2: The accuracy and average pooling time of choosing different pooling methods.

K_{min}	4	8	16
SLSNet	**86.3**	117.2	179.0
ALSNet	**1.1**	2.3	5.3
Accuracy(%)	95.74	**96.10**	95.95

Table 3: The MFLOPs/event and accuracy of SLSNet and ALSNet with different K_{min}.

5.3 Object Recognition

Experiments are conducted on three commonly used object recognition datasets. For CIFAR10-DVS and MNIST-DVS, T is set as $200ms$ and $100ms$. Each window is randomly sampled 4096 and 1024 events, respectively. The setting of N-cars is the same as in the ablation study.

Comparison with State of the Art. Table 4 compares the SLSNet and ALSNet with other methods. Compared with EvS-B (Li et al. 2021) which also utilizes a synchronous update process, thanks to the proposed local-shift operation, SLSNet reduces the computational complexity by $9.8\times$, $2.14\times$, and $12\times$, respectively. After introducing the asynchronous event processing procedure, ASLNet further re-

Methods	Representation	MNIST-DVS		N-Cars		CIFAR10-DVS	
		Acc%	MPs/ev	Acc%	MPs/ev	Acc%	MPs/ev
H-First (Orchard et al. 2015b)*	Spike	59.5	-	56.1	-	7.7	-
HATS(Sironi et al. 2018)*	TimeSurface	98.4	-	90.2	-	52.4	-
HOTS (Lagorce et al. 2017)*	TimeSurface	80.3	26	62.4	14.0	27.1	26
DART (Ramesh et al. 2020)*	TimeSurface	98.5	-	-	-	65.8	-
LIAF-Net (Wu et al. 2020)	Frame	99.1	-	-	-	70.4	-
YOLE (Cannici et al. 2019)*	VoxelGrid	96.1	-	92.7	328.1	-	-
Asynet (Messikommer et al. 2020)*	VoxelGrid	99.4	112	94.4	21.5	66.3	-
EV-VGCNN(Deng et al. 2022)	VoxelGrid	-	-	95.3	-	67.0	-
Dominic *et al.* (Jack et al. 2020)	Point-clouds	99.1	-	-	-	56.6	-
Bi *et al.* (Bi et al. 2020)	Graph	98.6	-	91.4	-	-	-
EvS-B (Li et al. 2021)	Graph	99.1	1152	93.1	251	68.0	3020
EvS-S (Li et al. 2021)*	Graph	99.1	15.2	93.1	6.1	68.0	33.2
SLSNet	Graph	**99.61**	117.2	**96.10**	117.2	**74.86**	251.2
ALSNet*	Graph	**99.61**	**3.3**	**96.10**	**2.3**	**74.86**	**2.0**

Table 4: Comparison with different methods on the MNIST-DVS dataset, N-Cars dataset, and CIFAR10-DVS dataset. * means that the results are obtained through event-by-event processing.

duces the computational cost because it only needs to update the activation of nodes affected by the new arrival event. Compared with SLSNet, ALSNet reduces the MFLOPs/event by $35\times$, $51\times$, and $125\times$ on the three datasets. Compared with EVS-S* (Li et al. 2021) and Asynet* (Messikommer et al. 2020), which achieve comparable accuracy, ASLNet obtains the best performance with the lowest complexity, which further proves the effectiveness of the proposed method. Note that CIFAR10-DVS is the most difficult dataset among the three datasets, due to high intra-class difference. ALSNet not only greatly improves the computational efficiency on this dataset, but also improves the prediction accuracy significantly, which proves the strong feature extraction and aggregation capabilities of the network.

5.4 Gesture Recognition

In DVS128 Gesture Dataset, T is set as $0.5s$. For each sliding window, 1024 events are sampled for training.

Comparison with State of the Art. Since the other methods in Table 5 do not provide results on MFLOPs/event, we compare the results on accuracy and GFLOPs. As shown in Table 5, SLSNet achieves the best performance with the lowest computational complexity. Although LIAF-Net (Wu et al. 2020), TA-SNN (Yao et al. 2021), and (Bi et al. 2020) have achieved comparable performance, the dense frame representation and CNNs used by the former two, and the spatial graph convolution adopted by the latter, all bring computational redundancy. Through introducing sparse graph representation and local-shift operation, SLSNet decreases GFLOPs by nearly $100\times$. In addition, these methods all need to perform synchronous update process when a new event arrives, which limits the low-latency nature of event data. By adopting the asynchronous event processing procedure, compared with SLSNet, MFLOPs/event of ALSNet is further reduced in inference (117.2 vs 3.1).

Methods	Representation	Acc(%)	GFLOPs
Amir *et al.* (Amir et al. 2017)	Spike	94.59	-
SpArNet (Khoei et al. 2020)	Spike	95.10	-
STBP-tdBN (Zheng et al. 2020)	Spike	96.87	-
Wang *et al.* (Wang et al. 2019)	Point-clouds	95.32	-
PAT (Yang et al. 2019)	Point-clouds	96.00	-
Kugele *et al.* (Kugele et al. 2020)	Frame	95.56	15.0
Massa *et al.* (Massa et al. 2020)	Frame	89.64	-
LIF-Net (He et al. 2020)	Frame	93.40	-
LIAF-Net (Wu et al. 2020)	Frame	97.56	13.6
TA-SNN (Yao et al. 2021)	Frame	98.61	-
Bi *et al.* (Bi et al. 2020)	Graph	97.20	13.7
SLSNet	Graph	**99.62**	**0.12**

Table 5: Comparison with different methods on the DVS128 Gesture Dataset.

6 Conclusion

Although previous method adopting GCNs have achieved remarkable performance in event-based tasks, they face redundant computation brought by the computational mechanism of spatial graph convolution and can not fast respond to the asynchronous event signals, which limit the low-latency nature of event-based data. This paper proposes a novel local-shift convolutional network for efficient event processing, which aggregates neighbors features for each node through local-shift operation and significantly reduces the computation complexity. To improve the efficiency of pooling operation in feature extraction, a node-importance based parallel pooling method is designed for low-latency and sparse event-based data. NIPooling efficiently obtains uniform sampling results that retain the diversity of events in parallel. Furthermore, an asynchronous processing procedure is proposed, which achieves a fast response to event signals. Experiments demonstrate that the ALSNet achieves new state-of-the-art results with the lowest computation complexity.

7 Acknowledgments

This work was supported in part by the National Key Research and Development Program of China (2020AAA0103402), NSFC 62273347, Jiangsu Leading Technology Basic Research Project (BK20192004) and Jiangsu Key Research and Development Plan (No.BE2021012-2).

References

Amir, A.; Taba, B.; Berg, D. J.; Melano, T.; McKinstry, J. L.; di Nolfo, C.; Nayak, T. K.; Andreopoulos, A.; Garreau, G.; Mendoza, M.; Kusnitz, J.; DeBole, M.; Esser, S. K.; Delbrück, T.; Flickner, M.; and Modha, D. S. 2017. A Low Power, Fully Event-Based Gesture Recognition System. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 7388–7397. IEEE Computer Society.

Bi, Y.; Chadha, A.; Abbas, A.; Bourtsoulatze, E.; and Andreopoulos, Y. 2020. Graph-based spatio-temporal feature learning for neuromorphic vision sensing. *IEEE Transactions on Image Processing*, 29: 9084–9098.

Brandli, C.; Berner, R.; Yang, M.; Liu, S.; and Delbrück, T. 2014. A 240 x 180 130 dB 3 μs Latency Global Shutter Spatiotemporal Vision Sensor. *IEEE J. Solid State Circuits*, 49(10): 2333–2341.

Cannici, M.; Ciccone, M.; Romanoni, A.; and Matteucci, M. 2019. Attention Mechanisms for Object Recognition With Event-Based Cameras. In *IEEE Winter Conference on Applications of Computer Vision, WACV 2019*, 1127–1136. IEEE.

Cannici, M.; Ciccone, M.; Romanoni, A.; and Matteucci, M. 2020. A Differentiable Recurrent Surface for Asynchronous Event-Based Data. In Vedaldi, A.; Bischof, H.; Brox, T.; and Frahm, J., eds., *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XX*, volume 12365 of *Lecture Notes in Computer Science*, 136–152. Springer.

Chen, G.; Cao, H.; Conradt, J.; Tang, H.; Röhrbein, F.; and Knoll, A. C. 2020a. Event-Based Neuromorphic Vision for Autonomous Driving: A Paradigm Shift for Bio-Inspired Visual Sensing and Perception. *IEEE Signal Process. Mag.*, 37(4): 34–49.

Chen, J.; Meng, J.; Wang, X.; and Yuan, J. 2020b. Dynamic Graph CNN for Event-Camera Based Gesture Recognition. In *IEEE International Symposium on Circuits and Systems, ISCAS 2020*, 1–5. IEEE.

Cheng, K.; Zhang, Y.; He, X.; Chen, W.; Cheng, J.; and Lu, H. 2020. Skeleton-based action recognition with shift graph convolutional network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 183–192.

Cheng, W.; Luo, H.; Yang, W.; Yu, L.; Chen, S.; and Li, W. 2019. DET: A High-Resolution DVS Dataset for Lane Extraction. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2019*, 1666–1675. Computer Vision Foundation / IEEE.

Deng, Y.; Chen, H.; Liu, H.; and Li, Y. 2022. A Voxel Graph CNN for Object Classification With Event Cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1172–1181.

Ding, Z.; Zhao, R.; Zhang, J.; Gao, T.; Xiong, R.; Yu, Z.; and Huang, T. 2022. Spatio-temporal recurrent networks for event-based optical flow estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 525–533.

Gehrig, D.; Rebecq, H.; Gallego, G.; and Scaramuzza, D. 2019. EKLT: Asynchronous Photometric Feature Tracking Using Events and Frames. *International Journal of Computer Vision*, 128: 601–618.

H, L.; H, L.; X, J.; G, L.; and L, S. 2017. CIFAR10-DVS: An Event-Stream Dataset for Object Classification.Front Neurosci.

He, W.; Wu, Y.; Deng, L.; Li, G.; Wang, H.; Tian, Y.; Ding, W.; Wang, W.; and Xie, Y. 2020. Comparing SNNs and RNNs on Neuromorphic Vision Datasets: Similarities and Differences. *CoRR*, abs/2005.02183.

Hu, L.; Zhao, R.; Ding, Z.; Ma, L.; Shi, B.; Xiong, R.; and Huang, T. 2022. Optical Flow Estimation for Spiking Camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 17844–17853.

Jack, D.; Maire, F.; Denman, S.; and Eriksson, A. P. 2020. Sparse Convolutions on Continuous Domains for Point Cloud and Event Stream Networks. In Ishikawa, H.; Liu, C.; Pajdla, T.; and Shi, J., eds., *Computer Vision - ACCV 2020 - 15th Asian Conference on Computer Vision*, volume 12622 of *Lecture Notes in Computer Science*, 400–416. Springer.

Jeon, Y.; and Kim, J. 2018. Constructing fast network through deconstruction of convolution. *Advances in Neural Information Processing Systems*, 31.

Khoei, M. A.; Yousefzadeh, A.; Pourtaherian, A.; Moreira, O.; and Tapson, J. 2020. SpArNet: Sparse Asynchronous Neural Network execution for energy efficient inference. In *2nd IEEE International Conference on Artificial Intelligence Circuits and Systems, AICAS 2020*, 256–260. IEEE.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015*.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images. 32–33.

Kugele, A.; Pfeil, T.; Pfeiffer, M.; and Chicca, E. 2020. Efficient Processing of Spatio-Temporal Data Streams With Spiking Neural Networks. *Frontiers in Neuroscience*, 14: 439.

Lagorce, X.; Orchard, G.; Galluppi, F.; Shi, B. E.; and Benosman, R. 2017. HOTS: A Hierarchy of Event-Based Time-Surfaces for Pattern Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(7): 1346–1359.

Lecun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Li, Y.; Zhou, H.; Yang, B.; Zhang, Y.; Cui, Z.; Bao, H.; and Zhang, G. 2021. Graph-based asynchronous event processing for rapid object recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 934–943.

Lichtsteiner, P.; Posch, C.; and Delbrück, T. 2008. A 128 x 128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor. *IEEE J. Solid State Circuits*, 43(2): 566–576.

Liu, Q.; Ruan, H.; Xing, D.; Tang, H.; and Pan, G. 2020. Effective AER Object Classification Using Segmented Probability-Maximization Learning in Spiking Neural Networks. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020*, 1308–1315. AAAI Press.

Massa, R.; Marchisio, A.; Martina, M.; and Shafique, M. 2020. An Efficient Spiking Neural Network for Recognizing Gestures with a DVS Camera on the Loihi Neuromorphic Processor. *CoRR*, abs/2006.09985.

Messikommer, N.; Gehrig, D.; Loquercio, A.; and Scaramuzza, D. 2020. Event-based asynchronous sparse convolutional networks. In *European Conference on Computer Vision*, 415–431. Springer.

Mitrokhin, A.; Hua, Z.; Fermüller, C.; and Aloimonos, Y. 2020. Learning Visual Motion Segmentation Using Event Surfaces. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020*, 14402–14411. Computer Vision Foundation / IEEE.

Nguyen, A.; Do, T.; Caldwell, D. G.; and Tsagarakis, N. G. 2019. Real-Time 6DOF Pose Relocalization for Event Cameras With Stacked Spatial LSTM Networks. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2019*, 1638–1645. Computer Vision Foundation / IEEE.

Orchard, G.; Jayawant, A.; Cohen, G.; and Thakor, N. 2015a. Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades. arXiv:1507.07629.

Orchard, G.; Meyer, C.; Etienne-Cummings, R.; Posch, C.; Thakor, N. V.; and Benosman, R. 2015b. HFirst: A Temporal Approach to Object Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(10): 2028–2040.

Pan, L.; Liu, M.; and Hartley, R. 2020. Single Image Optical Flow Estimation With an Event Camera. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020*, 1669–1678. Computer Vision Foundation / IEEE.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In Guyon, I.; von Luxburg, U.; Bengio, S.; Wallach, H. M.; Fergus, R.; Vishwanathan, S. V. N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, 5099–5108.

Ramesh, B.; Yang, H.; Orchard, G.; Thi, N. A. L.; Zhang, S.; and Xiang, C. 2020. DART: Distribution Aware Retinal Transform for Event-Based Cameras. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(11): 2767–2780.

Rebecq, H.; Ranftl, R.; Koltun, V.; and Scaramuzza, D. 2019. Events-To-Video: Bringing Modern Computer Vision to Event Cameras. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019*, 3857–3866. Computer Vision Foundation / IEEE.

Rebecq, H.; Ranftl, R.; Koltun, V.; and Scaramuzza, D. 2021a. High Speed and High Dynamic Range Video with an Event Camera. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(6): 1964–1980.

Rebecq, H.; Ranftl, R.; Koltun, V.; and Scaramuzza, D. 2021b. High Speed and High Dynamic Range Video with an Event Camera. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(6): 1964–1980.

Sekikawa, Y.; Hara, K.; and Saito, H. 2019. EventNet: Asynchronous Recursive Event Processing. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019*, 3887–3896. Computer Vision Foundation / IEEE.

Shrestha, S. B.; and Orchard, G. 2018. SLAYER: Spike Layer Error Reassignment in Time. *CoRR*, abs/1810.08646.

Sironi, A.; Brambilla, M.; Bourdis, N.; Lagorce, X.; and Benosman, R. 2018. HATS: Histograms of Averaged Time Surfaces for Robust Event-Based Object Classification. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*, 1731–1740. Computer Vision Foundation / IEEE Computer Society.

Wang, Q.; Zhang, Y.; Yuan, J.; and Lu, Y. 2019. Space-Time Event Clouds for Gesture Recognition: From RGB Cameras to Event Cameras. In *IEEE Winter Conference on Applications of Computer Vision, WACV 2019*, 1826–1835. IEEE.

Wu, B.; Wan, A.; Yue, X.; Jin, P.; Zhao, S.; Golmant, N.; Gholaminejad, A.; Gonzalez, J.; and Keutzer, K. 2018. Shift: A zero flop, zero parameter alternative to spatial convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 9127–9135.

Wu, Z.; Zhang, H.; Lin, Y.; Li, G.; Wang, M.; and Tang, Y. 2020. LIAF-Net: Leaky Integrate and Analog Fire Network for Lightweight and Efficient Spatiotemporal Information Processing. *CoRR*, abs/2011.06176.

Yang, J.; Zhang, Q.; Ni, B.; Li, L.; Liu, J.; Zhou, M.; and Tian, Q. 2019. Modeling Point Clouds With Self-Attention and Gumbel Subset Sampling. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019*, 3323–3332. Computer Vision Foundation / IEEE.

Yao, M.; Gao, H.; Zhao, G.; Wang, D.; Lin, Y.; Yang, Z.; and Li, G. 2021. Temporal-wise Attention Spiking Neural Networks for Event Streams Classification. *CoRR*, abs/2107.11711.

Zheng, H.; Wu, Y.; Deng, L.; Hu, Y.; and Li, G. 2020. Going Deeper With Directly-Trained Larger Spiking Neural Networks. *CoRR*, abs/2011.05280.

Zhong, H.; Liu, X.; He, Y.; and Ma, Y. 2018. Shift-based primitives for efficient convolutional neural networks. *arXiv preprint arXiv:1809.08458*.