Leveraging Asynchronous Spiking Neural Networks for Ultra Efficient Event-Based Visual Processing

DingYi Zeng[1*], Yuchen Wang[1*], Honglin Cao[1], Wanlong Liu[1], Yichen Xiao[1], ChengzhuoLu[1], Wenyu Chen[1], Malu Zhang[1†], Guoqing Wang[1], Yang Yang[1]

[1]School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China
zengdingyi@std.uestc.edu.cn, yuchenwang@std.uestc.edu.cn, 202422081203@std.uestc.edu.cn liuwanlong@std.uestc.edu.cn, xiaoyichen@std.uestc.edu.cn, 2019270101012@std.uestc.edu.cn, cwy@uestc.edu.cn, maluzhang@uestc.edu.cn, gqwang0420@uestc.edu.cn, yang.yang@uestc.edu.cn

Abstract

Event cameras encode visual information by generating asynchronous and sparse event streams, which hold great potential for low latency and low power consumption. Despite many successful implementations of event camera-based applications, most of them accumulate the events into frames and then utilize conventional frame-based computer vision algorithms. These frame-based methods, though typically effective, diminish the inherent advantages of the event camera's low latency and low power consumption. To solve the above problems, we propose ASGCN, which efficiently processes data on an event-by-event basis and dynamically evolves into a corresponding dynamic representation, enabling low latency and high sparsity of data representation. The sparsity computation is further improved by introducing brain-inspired spiking neural networks, resulting in low power consumption for ASGCN. Extensive and diverse experiments demonstrate the energy efficiency and low latency advantages of our processing pipeline. Especially on real-world event camera datasets, our pipeline consumes more than 10,000 times less energy and achieves similar performance compared to current frame-based methods.

Introduction

Benefiting from their unique characteristics, event cameras, also known as bio-inspired neuromorphic cameras, have extraordinary advantages in many high-barrier fields such as autonomous driving (Zhu et al. 2018), robotics (Falanga, Kleber, and Scaramuzza 2020), computational imaging (Rebecq et al. 2019) and Internet of Things surveillance (Mitra, Fusi, and Indiveri 2008). The independent measurement of the brightness change of each pixel enables event cameras to have an ultra-high dynamic range (up to 140 dB), while the event stream-based data processing pipeline enables event cameras to have ultra-low latency and ultra-high time resolution (both in the order of μs) (Gallego et al. 2020). Furthermore, due to the highly sparse and asynchronous data pipeline based on the event stream, the power consumption and bandwidth required by the event camera are exponentially lower than those of the frame-based camera.

Methods	Accuracy	MOP	Energy (MJ)
EST	0.925	1050	4830
YOLE	0.927	328.16	1509.53
AsyNet	**0.944**	21.5	98.9
NVS-S	0.915	5.2	23.92
Ours	0.939	**0.04**	**0.036**

Table 1: Compared with the performance and power consumption of some SOTA models on N-Cars dataset.

Filtering-based (Gallego et al. 2017; Kim, Leutenegger, and Davison 2016; Lagorce et al. 2016; Orchard et al. 2015b) approaches are the forerunners of event processing pipelines, but the technical route based on handcrafted filter limits their ability to be broadly applied to a variety of tasks. Indeed, a distinct disparity exists in the efficacy of real-world task performance between filtering techniques and deep neural network approaches.

In order to better utilize the advantages of the event camera, the method of event-by-event processing includes Time-surface-based methods (Mitrokhin et al. 2020; Lagorce et al. 2016; Sironi et al. 2018) and Graph-based methods (Bi et al. 2019, 2020; Li et al. 2021b; Wang et al. 2019; Deng et al. 2022; Sekikawa, Hara, and Saito 2019; Schaefer, Gehrig, and Scaramuzza 2022; Zhu, Hou, and Lyu 2022) are proposed. Graph-based methodologies predominantly leverage the duality inherent in events to uphold the sparsity of the event stream. This entails treating each event as a graph node, culminating in the computation of node representations that collectively form the comprehensive event stream representation. Nevertheless, it's noteworthy that the network architecture of graph-based approaches tends to be dense, resulting in heightened computational demands for graph neural networks, particularly when confronted with a substantial volume of events.

The brain-inspired Spiking Neural Networks (SNNs) are proposed for asynchronous computing with ultra-low power consumption(Xu et al. 2022). Unlike conventional Artificial Neural Networks (ANNs), neurons in SNNs will fire a spike when the accumulated membrane potential exceeds a threshold, otherwise, they will remain silent. Benefiting from this way of completely imitating the operation of real biological

*These authors contributed equally.
†Corresponding author: Malu Zhang

neurons, SNNs have richer spatial-temporal dynamic characteristics than ANNs.

In addition, SNNs are promising to provide a low-power consumption alternative due to their binary and sparse-asynchronous spikes(Xu et al. 2023). Inspired by these, there are some existing works that try to utilize SNNs to process event stream data (Mitra, Fusi, and Indiveri 2008; Amir et al. 2017; Orchard et al. 2015b; Liu et al. 2020). However, most SNN-based methods convert event streams into frame sequences, losing the sparsity and time dependence in event data. Moreover, these methods weaken the intrinsic benefits offered by event cameras, such as their low latency and low power consumption attributes.

In this work, we apply sparsity to event stream representation and network architecture at the same time and build the most efficient SNNs-based event stream processing pipeline. Our processing pipeline transforms the event stream into a graph and reduces the resources required for computation through a subgraph extraction strategy. From a network architecture perspective, we greatly reduce the computational cost of new events by using asynchronous node updates and sparse spike message passing. We validate the efficiency of the proposed model with some popular event datasets. Compared with the other state-of-the-art models, it has more than 10,000 times lower energy consumption while still achieving similar accuracy, as shown in Table 1. In summary, the main contributions of this paper are as follows:

- We propose an Asynchronous Spiking Graph Convolutional Network (ASGCN) as a novel approach for graph representation learning. To the best of our knowledge, *our study represents the first attempt to utilize spiking neural networks for real-world graph-level tasks*.
- We built a processing pipeline that directly processes raw neuromorphic data in a fully event-driven manner. Moreover, by modeling event streams as graphs and utilizing the ASGCN for computation, the pipeline can effectively maintain event data's sparsity and asynchrony characteristics throughout the graph representation and computation stages.
- Extensive and diverse experiments demonstrate the effectiveness of our processing pipeline. In particular, our framework achieves performance close to state-of-the-art methods on real tasks while requiring more than 10,000 times less energy consumption compared to frame-based methods.

Preliminaries and Related Work

Event Cameras

As bio-inspired sensors, event cameras generate asynchronous event streams by independently measuring logarithmic-scale brightness changes in each individual pixel, which enables event cameras with ultra-high dynamic range (up to 140 dB).

In addition, the data processing pipeline based on asynchronous event streams enables event cameras with ultra-low latency and ultra-high time resolution (both are on the order of μs). Different from traditional frame-based cameras, the sparseness and asynchrony of event cameras are embodied by event streams, where events are the basic unit in the pipeline. Formally, a set of events can be defined as

$$\mathcal{E} = \{e_k\}_{k=1}^{N} = \{(x_k, y_k, t_k, p_k)\}_{k=1}^{N}, \tag{1}$$

where the coordinates of the pixel are (x, y), the timestamp is t, and the polarity of the event is p. More specifically, an event is generated when there is a logarithmic change in brightness, which can be expressed as:

$$L(x, y, t) - L(x, y, t - \Delta t) \geq pC, \tag{2}$$

where Δt is the time since the last event at coordinate (x, y), and C is the contrast threshold.

Spiking Neural Networks

Due to its simplicity and ability to reflect the basic characteristics of real neurons, the Integrate-and-Fire (IF) neuron is employed in this work. The membrane potential $v_j^l(t)$ of a IF neuron j in layer l at time step t can be described as:

$$\tilde{v}_j^l(t) = v_j^l(t-1) + \sum_i w_{ij} s_i^{l-1}(t) + b_j, \tag{3}$$

$$s_j^l(t) = \begin{cases} 1, & \tilde{v}_j^l(t) \geq \theta_j^l \\ 0, & \text{otherwise} \end{cases}, \tag{4}$$

$$v_j^l(t) = (1 - s_j^l(t))\tilde{v}_j^l(t) + s_j^l(t) v_{\text{rest}}, \tag{5}$$

where s_i^{l-1} is the spike from presynaptic neuron i and v_{rest} donates the resting potential. When $\tilde{v}_j^l(t)$ exceeds firing threshold θ_j^l, neuron j will send a spike to its postsynaptic neuron.

At present, the most commonly used learning methods of SNNs are surrogate gradient learning (Wu et al. 2018; Neftci, Mostafa, and Zenke 2019; Xu et al. 2024) and ANN-SNN conversion (Li et al. 2021a; Wang et al. 2022). Surrogate gradient learning solves the problem of non-differentiable binary activation function, and can directly train SNN from scratch. However, their training procedure requires enormous resources and suffers from unsatisfactory accuracy. ANN-SNN conversion methods directly map the parameters of a pre-trained ANN to SNN with the same structure.

These methods not only bypass the non-differentiable issue of the spike activity, but also save training time and resources compared with the direct training rules. Since the output rate of IF neurons cannot exceed 1, which requires the output range of the ReLU function to be $[0, 1]$, it is needed to transform the parameters of ANN. The layer-wise parameter normalization (LayerNorm) (Rueckauer et al. 2017) rescales all parameters by the maximum output value λ^l of layer l as follows:

$$W^l = W^l \frac{\lambda^{l-1}}{\lambda^l}, \quad b^l = \frac{b^l}{\lambda^l}. \tag{6}$$

After replacing the neurons of ANN with IF neurons, and rescaling the network parameters, a pre-trained ANN is transformed into SNN that can be used directly.

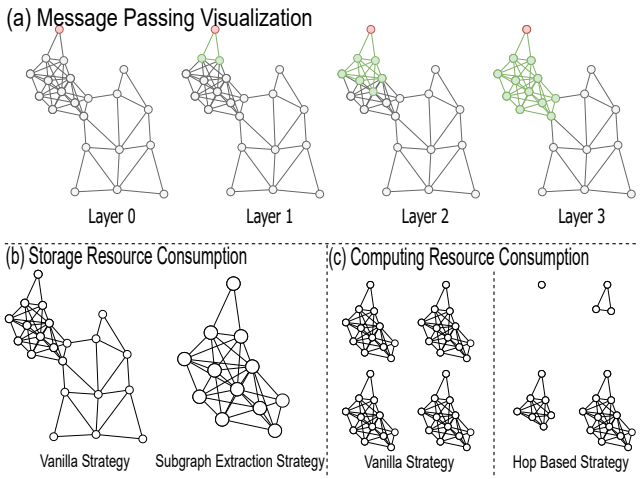

Figure 1: (a) The message passing visualization in graphs, where the red node is a new node, and the green nodes are the nodes affected by the new node. (b) The comparison of the storage resource consumption of the vanilla strategy and the subgraph extraction strategy. (c) The comparison of computing resource consumption between the hop-based strategy and the vanilla strategy.

Graph Neural Networks

MPNNs (Gilmer et al. 2017) provide a methodology to abstract GNNs with a unified view of message passing mechanisms. In the case of this paradigm, node-to-node information is propagated by iteratively aggregating neighboring node information to a central node. Formally, given a node i in graph G, its hidden representation $\mathbf{h}_i^t$ at t iteration, its neighboring nodes $N(i)$, edge $\mathbf{e}_{ij}$ connecting node i to node j, an iteration of standard message passing paradigm can be expressed as:

$$\mathbf{c}_i^t = \sum_{j \in N(i)} \phi^t \left(\mathbf{h}_i^t, \mathbf{h}_j^t, \mathbf{e}_{ij} \right), \tag{7}$$

$$\mathbf{h}_i^{t+1} = \sigma^t(\mathbf{c}_i^t, \mathbf{h}_i^t), \tag{8}$$

where ϕ^t and σ^t are the $aggregate$ and $update$ function at t iteration. (Bi et al. 2020) is the pioneering work that transforms neuromorphic spike events into graphs, leading to significant computational reduction compared to traditional ResNet50 (He et al. 2016). Additionally, they establish a processing pipeline involving Sampling-Graph Construction-Graph Convolution. (Li et al. 2021b) introduces the SlideGCN framework for early recognition, making better use of the event cloud's partial regular structure and enhancing early recognition speed based on event graph stability. (Schaefer, Gehrig, and Scaramuzza 2022) extends the standard GNN to handle evolving spatio-temporal graphs through an asynchronous, event-based graph neural network, reducing computational complexity through synchronous training and asynchronous testing. (Zhu, Hou, and Lyu 2022) pioneers an end-to-end learning paradigm utilizing event clouds, integrating a density-insensitive downsampling strategy and motion flow-based key event backtracking via learning.

Methodology

In this section, we will first introduce how the event graph is constructed. Then, we will introduce the proposed asynchronous spiking graph convolution, which can process spike-based graph data.

Event Graph Construction

Formally, a set of events streams can be defined as:

$$\mathcal{E} = \{e_i\}_{i=1}^N = \{(x_i, y_i, t_i, p_i)\}_{i=1}^N, \tag{9}$$

where the coordinates of the pixel are (x, y), the timestamp is t, and the polarity of the event is p.

Sampling Events are sparse in the field of view of the entire event camera sensor, but extremely dense in terms of Event Stream. In order to reduce the consumption of computing resources by a large number of events, we uniformly sample events by a factor R to obtain the sampled event stream $\widetilde{\mathcal{E}}$.

Graph Construction For the sampled event stream $\widetilde{\mathcal{E}}$, we regard each event as a node, which contains coordinates, timestamps, and polarity features. For each node pair ij, when the Euclidean distance between node i and node j is less than D, an edge E_{ij} is created between them, and the feature of the edge is the Euclidean distance between node i and node j. Let $G = (V, E, \mathbf{X})$ be a simple, undirected, connected graph with a finite set of $nodes$ V and a finite set of $edges$ E, where the node feature $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^\top$. Formally, feature for node i is defined as:

$$\mathbf{x}_i = (x_i, y_i, t_i, p_i). \tag{10}$$

Subgraph Extraction Among Graph Neural Networks, in order for a node to receive information from other nodes at a radius of K, GNNs needs to stack at least K layers, which significantly limits their capabilities (Alon and Yahav 2021). However, we innovatively take advantage of this defect from another side, and reduce the storage space required for calculation by directly obtaining the feature of the nodes participating in the calculation.

Specifically, for a K-layer stacked Graph Neural Network, only the information of the receptive field with a radius of K is needed to calculate the representation of a specific node. As shown in Fig. 1 (a), a new node is added to the graph and computed through each layer of the graph neural network. In the first layer, a newly added node can only affect its first-order neighbor nodes, and as the number of layers deepens, the order of the nodes it affects also increases. The representations of other unaffected nodes have nothing to do with the newly added node representations, so the calculation of the graph neural network only needs to be performed in the affected nodes.

To this end, we extract the K-hop $Egonetwork$ of the root node i, and perform message passing within the $Egonetwork$ subgraph to reduce storage resource consumption. As shown in Fig. 1 (b), our subgraph extraction strategy allows us to store less graph data during computation. Formally speaking, $N(v)$ denotes the set of neighboring nodes of the root node v and a more generalized notation $N_K(v)$ denotes the set of nodes within k-hop from the

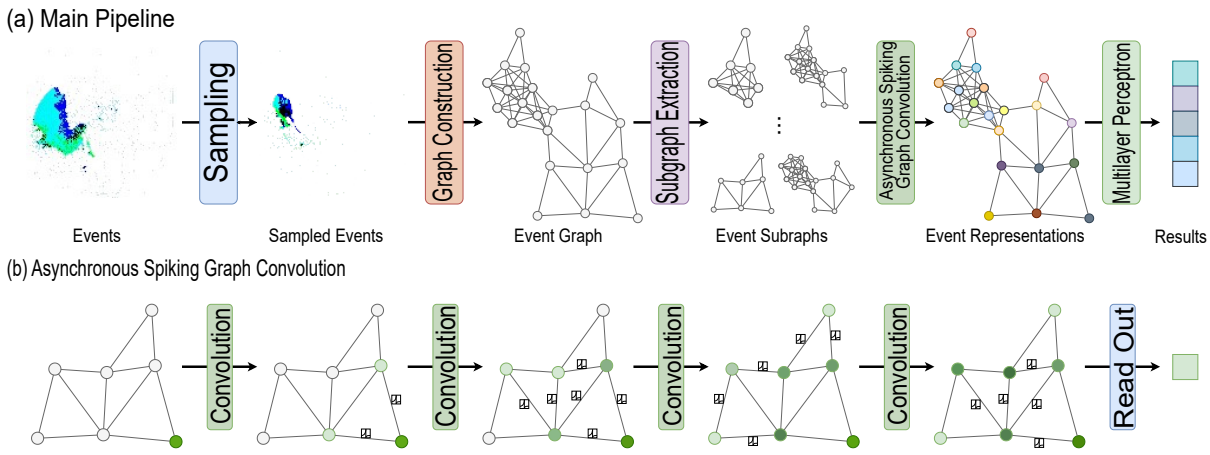

Figure 2: (a) The overview of our processing pipeline. We first uniformly sample events by a factor R to obtain the sampled event stream $\widetilde{\mathcal{E}}$. Then we construct a graph over the sampled event stream via the Euclidean distance between pairs of nodes. Finally, graph representation is obtained by constructing subgraphs and performing Asynchronous Spiking Graph Convolution on the subgraphs. (b) A schematic diagram of Asynchronous Spiking Graph Convolution. Unlike traditional graph neural networks, ASGC does not necessarily transmit information in every message passing.

root node v. Then the $Ego(v)_K$ is a K-hop $Egonetwork$ rooted at node v and its corresponding nodes are the neighbors within K-hop $N_K(v)$ of the root node v.

Asynchronous Spiking Graph Convolutional Networks

Unlike other frame-based image data, event streams are asynchronous and sparse. We solve the sparsity problem by transforming event streams into event stream graphs, but conventional Graph Neural Networks are not suitable for asynchronous graph data. Thus, we generalize graph convolutions to asynchronous subgraph convolutions for updating node-specific representations. Subsequently, in order to let the model calculate in a more sparse way, we propose the Asynchronous Spiking Graph Convolution Network (ASGCN).

Specifically, we treat event stream graphs as a kind of dynamic graphs with sliding time windows. In a given time window $\triangle^T$, there exists a graph $G^{\triangle^T}$ consisting of events contained within that time window. When the time window slides, events continue to come, and new events, that is, nodes are continuously added to the existing graph through the graph construction strategy.

For a new node i, we extract it's K-hop $Egonetwork$, and perform message passing operation on its K-hop $Egonetwork$ to obtain its representation. Our model has different calculations during the training phase and the inference phase. In the training phase, we decompose the graph neural network applied to subgraphs into graph neural networks applied to each hop, as:

$$\mathbf{c}_i^l = \frac{1}{|Ego(i)_l|} \sum_{j \in Ego(i)_l} \mathbf{h}_j^l \cdot \varphi^l\left(\mathbf{e}_{ij}\right), \tag{11}$$

$$\mathbf{h}_i^{l+1} = \sigma^l(\mathbf{c}_i^l, \mathbf{h}_i^l), \tag{12}$$

where σ^l is the $update$ function at l layer of our network, φ^l is the kernel function defined over the weighted B-Spline tensor product basis (Fey et al. 2018). In the training phase, the ReLU activation function is still used in σ^l and relies on floating-point operations. Subsequently, we convert the network into an SNN with the same structure for the network inference phase, which can reason in an asynchronous and sparse manner.

In the inference phase, we first fold Batch Normalization (BN) layers into conventional graph convolution in Eq. (11). Because only binary spikes are used to characterize node features, BN cannot be used directly. The calculation of BN is:

$$\mathbf{x}' = \frac{\mathbf{x} - E[\mathbf{x}]}{Std[\mathbf{x}]} \odot \gamma + \beta, \tag{13}$$

where γ and β are trainable parameters. $E[\mathbf{x}]$ and $Std[\mathbf{x}]$ are mean and standard deviation of $\mathbf{x}$, respectively. So the Eq. (11) should updated as:

$$\begin{aligned}\mathbf{c}_i^l = &\frac{\gamma}{|Ego(i)_l| \cdot Std[\mathbf{c}]} \sum_{j \in Ego(i)_l} \mathbf{h}_j^l \cdot \varphi^l\left(\mathbf{e}_{ij}\right) \\ &+ \frac{\gamma \cdot (-E[\mathbf{c}])}{Std[\mathbf{c}]} + \beta.\end{aligned} \tag{14}$$

It should be noted that the $\mathbf{h}_j^l$ in Eq. (14) is the spike train. The result calculated by Eq. (14) is the input information of the spiking neuron, and then the membrane potential will be updated as:

$$\tilde{\mathbf{v}}_i^l(t) = \mathbf{v}_i^l(t-1) + \mathbf{c}_i^l(t) + \mathbf{h}_i^l(t-1). \tag{15}$$

$$\mathbf{h}_i^l(t) = \begin{cases} \theta_i^l, & \tilde{\mathbf{v}}_i^l(t) \geq \theta_i^l \\ 0, & \text{otherwise} \end{cases}. \tag{16}$$

Parameter normalization is equivalent to transmitting threshold value in SNN (Li et al. 2021a). Since the firing

Methods	Network	Representation	Asynchrony	Network Sparsity	Representation Sparsity	N-MNIST	CIFAR10-DVS	N-Caltech101	N-Cars
HOTS	ANN	TimeSurface	✓	×	×	0.808	0.271	0.210	0.624
HATS	ANN	TimeSurface	✓	×	×	0.991	0.524	0.642	0.902
DART	ANN	TimeSurface	✓	×	×	0.985	0.658	0.664	-
YOLE	ANN	VoxelGrid	✓	×	×	0.961	-	0.702	0.927
AsyNet	ANN	VoxelGrid	✓	×	×	0.994	0.663	0.745	0.944
EST	ANN	Event-Histogram	×	×	×	-	-	0.817	0.925
SSC	ANN	Event-Histogram	×	×	×	-	-	0.761	0.945
GIN	ANN	Graph	×	×	✓	0.754	0.423	0.476	0.846
ChebConv	ANN	Graph	×	×	✓	0.949	0.452	0.524	0.855
GCN	ANN	Graph	×	×	✓	0.781	0.418	0.530	0.827
AEGNN	ANN	Graph	✓	×	✓	-	-	0.668	0.945
MoNet	ANN	Graph	×	×	✓	0.965	0.476	0.571	0.854
RG-CNNs	ANN	Graph	×	×	✓	0.990	0.540	0.657	0.914
NVS-S	ANN	Graph	✓	×	✓	0.986	0.602	0.670	0.915
TA-SNN	SNN	Event-Histogram	✓	✓	×	0.982	0.72	-	-
H-First	SNN	Event-Histogram	✓	✓	×	0.712	0.077	0.054	0.561
Gabor-SNN	SNN	Event-Histogram	✓	✓	×	0.837	0.245	0.196	0.789
AER	SNN	Event-Histogram	✓	✓	×	0.963	0.322	-	-
DECOLLE	SNN	Event-Histogram	✓	✓	×	0.96	-	-	-
Ours	SNN	Graph	✓	✓	✓	0.977	0.594	0.662	0.939

Table 2: Comparison of experimental performance with different technical routes for object recognition. Our pipeline has similar performance to multiple state-of-the-art technical routes, including different neural network architectures, different representations and whether they are asynchronous.

rate of neurons in SNN cannot exceed 1, the firing threshold can be set to the maximum activation value of ANN. In order to distinguish different node features, we use the maximum activation value in a single feature dimension when setting the neuron firing threshold θ_i^l.

Finally, we use IF neurons with the soft-reset mechanism in ASGCN, which describes Eq. (5) as:

$$\mathbf{v}_i^l(t) = \tilde{\mathbf{v}}_j^l(t) - \mathbf{h}_i^l(t). \tag{17}$$

In contrast to the hard-reset mechanism in Eq. (5), soft-reset neurons retain residual potential at firing instants, resulting in a better performance. Furthermore, we regarded the input of SNN as the input current of spiking neurons, so we don't apply additional encoding.

For the node feature $\mathbf{h}_j^l$ that already sufficiently contains high-level features, we need to convert it into a graph representation. To further reduce the computational burden, we reduce the number of nodes through graph pooling. We obtain the cluster center C_p through node clustering, and construct a more compact graph through the cluster center. For a node v belonging to cluster C_p, we obtain the features of new cluster centroids through average pooling:

$$\mathbf{h}_p = \frac{1}{|C_p|} \sum_{i \in C_p} \mathbf{h}_i. \tag{18}$$

Clustering reduces the number of nodes, so we convert the edges of the original graph to edges in the new graph through the ψ function:

$$E_c = \psi(E, C). \tag{19}$$

We then proceed to perform convolutions on the new graph.

Experiments and Analysis

In this section, we conduct a detailed analysis of our pipeline, including comparison experiments, ablation experiments, resource consumption analysis and spiking neuron analysis.

Experimental Setup

Datasets We primarily validate our pipeline on the task of event-based object recognition which is predicting the category of the object through the input event stream. We primarily validate our pipeline on the task of event-based object recognition which is predicting the category of the object through the input event stream. Due to their high dynamic range and high temporal resolution, event cameras have the potential to detect ultra-high-speed objects in harsh lighting environments, while frame-based methods struggle to detect objects in similar situations.

Therefore we verify the effectiveness of our pipeline on two types of datasets: synthetic neuromorphic datasets such as N-MNIST (Orchard et al. 2015a), CIFAR10-DVS (Li et al. 2017), N-Caltech101 (Orchard et al. 2015a) and native neuromorphic datasets such as N-Cars (Sironi et al. 2018).

Synthetic neuromorphic datasets such as N-MNIST (Orchard et al. 2015a), CIFAR10-DVS (Li et al. 2017), N-Caltech101 (Orchard et al. 2015a) are converted from frame-based datasets to event-based data by displaying a moving image on a monitor and capturing it with an event camera.

Native neuromorphic datasets such as N-Cars (Sironi et al. 2018) are constructed by directly using event cameras to record objects in real-world environments.

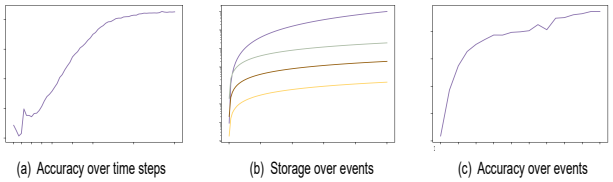

Figure 3: (a) The inference top-1 accuracy on N-Cars dataset w.r.t different simulation lengths. (b) The required storage space w.r.t the different number of events in Dense Vanilla, Sparse Vanilla, Dense Subgraph, Sparse Subgraph strategy. (c) The inference top-1 accuracy on N-Cars dataset w.r.t the different number of events.

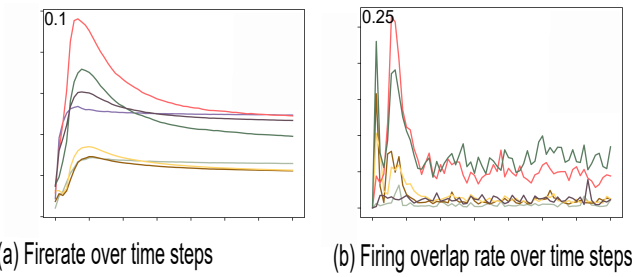

Figure 4: (a) The firing rate of our 6-layer ASGCN on N-Cars dataset. Here we tested 64 different simulation lengths (from 1 to 64). (b) Overlap ratio of firing neurons when simulation length is 64.

Implementation Details We use the PyTorch framework (Paszke et al. 2017) and the PyTorch Geometric library (Fey and Lenssen 2019) to implement our Asynchronous Spiking Graph Convolution Network pipeline. In the training phase of our model, we adopt the Adam algorithm with gradient centralization. The L2 penalty with a value of $5e^{-3}$ is also added. We set the initial learning rate as $1e^{-3}$, followed by milestones learning rate decay. In the inference phase, we initialize the membrane potential of spiking neurons as half of firing threshold for better performance. As for the input of SNN, which can be regarded as the input current of spiking neurons, we don't apply additional encoding.

Baselines We compare our pipeline with several state-of-the-art methods of different technical routes, both asynchronous and synchronous, with different event representations, and with different network architectures including: HFirst (Orchard et al. 2015b), HOTS (Lagorce et al. 2016), HATS (Sironi et al. 2018), DART (Ramesh et al. 2019), YOLE (Cannici et al. 2019), DECOLLE (Kaiser, Mostafa, and Neftci 2020), AER (Liu et al. 2020), TA-SNN (Yao et al. 2021), EST (Gehrig et al. 2019), SSC (Graham, Engelcke, and Van Der Maaten 2018), AsyNet (Messikommer et al. 2020), NVS-S (Li et al. 2021b), AEGNN (Schaefer, Gehrig, and Scaramuzza 2022), Gabor-SNN (Sironi et al. 2018), GIN (Xu et al. 2019), ChebConv (Defferrard, Bresson, and Vandergheynst 2016), GCN (Kipf and Welling 2017), MoNet (Monti et al. 2017), and RG-CNNs (Bi et al. 2019).

Comparison with other Technical Routes

Table 2 compares our results with other state-of-the-art methods of different technical routes. We emphasize three key features across different approaches: network architecture, representation method, and asynchrony. Network architecture encompasses ANN and SNN, with SNN leveraging neuromorphic computing for energy-efficient inference. Representation methods span diverse approaches, including converting event streams to TimeSurfaces to capture asynchrony. The final aspect is asynchrony, indicating if the method can process event streams event-by-event. We mainly focus on three characteristics of different technical routes: network architecture, representation method, and asynchrony.

The network architecture mainly includes ANN and SNN, among which SNN benefits from neuromorphic computing methods with lower energy consumption in the inference. Representation methods include a variety of technical routes, such as converting event stream to TimeSurface to obtain asynchrony. The last characteristic is asynchrony, which expresses whether the method can process event stream event-by-event.

HFirst (Orchard et al. 2015b) and Gabor-SNN (Sironi et al. 2018) adopt the network architecture of SNN, but the representation based on Event-Histogram seriously drags down its consumption of computing resources. MoNet (Monti et al. 2017)and RG-CNNs (Bi et al. 2019) introduce graphs into event stream representations, pioneering geometric learning for event cameras. However, the lack of asynchrony makes MoNet and RG-CNNs less valuable for practical deployment. NVS-S (Li et al. 2021b) AEGNN (Schaefer, Gehrig, and Scaramuzza 2022) introduce an asynchronous, event-based graph neural network, which has made progress in computational consumption and inference efficiency.

As shown in Table 2, our pipeline chooses SNN for the network architecture, uses graph as the representation of event stream, and maximizes the resource saving through the simultaneous sparseness of data representation and network architecture. The performance of our method surpasses most graph-based methods on real-world tasks and is close to that of frame-based methods. Especially in the N-Cars dataset collected by the real event camera, the performance gap between our method and the state-of-the-art method is less than one percent.

Computational Consumption Analysis

To prove the energy efficiency of the proposed method, we further quantitatively simulate the power consumption of the proposed model, referring to (Horowitz 2014; Rathi and Roy 2020). In ANNs, each dot-product computation involves one floating-point Multiply Accumulate (MAC) operation, while in SNNs each operation is one floating-point addition due to binary spikes. Moreover, in the SNN, the calculation will only occur when the neuron receives the spike, and there will be no energy consumption in the neuron's resting state.

Method	Accuracy	MOP	Energy (MJ)
ASGCN	0.939	0.04	0.036
Modified BN $\rightarrow$ Vanilla BN	0.867	0.04	0.036
Graph $\rightarrow$ Event-Histogram	0.947	424	1950

Table 3: Ablation experiments on different components.

R	10	20	30	40	50	60	70	80
Acc.(%)	93.9	92.1	91.7	91.0	88.4	87.7	84.5	83.3
D	2	3	4	5	6	7	8	9
Acc.(%)	93.89	93.45	92.87	92.22	91.93	91.68	91.34	91.11

Table 4: Hyperparameter experiments on the N-Cars dataset.

So we can estimate the energy consumption by the number of spikes, the number of operations of layer l in SNN is:

$$\text{OP}^l = \begin{cases} E \times f_{node} + N \times f_{node}^2, & \text{Conv} \\ f_{in} \times f_{out}, & \text{FC} \end{cases}, \tag{20}$$

$$\text{OP}^l_{\text{SNN}} = \text{FiringRate}^l \times T \times \text{OP}^l, \tag{21}$$

where E is the number of edges, N is the number of nodes, and f_{node} is the node feature dimension. f_{in} and f_{out} are the number of input features and output features. FiringRate^l is the average neuron firing rate in layer l, and T is the simulation length. The energy cost of 32-bit MAC operation is $4.6pJ$ in a 45nm CMOS, and an addition operation needs $0.9pJ$ (Horowitz 2014). Leveraging this method, we simulated the energy consumption of both our proposed model and some state-of-the-art (SOTA) works on the N-Cars dataset, and the corresponding outcomes are presented in Table 1.

Scalability Analysis with Storage Consumption

For a GNN with N nodes, E edges, L layers, and F node feature dimensions, dense inference has a space complexity of LN^2+LF^2+LNF, and sparse inference has a complexity of $LE + LF^2 + LNF$. While acceptable for small-scale static graphs, this complexity becomes problematic for event streams due to continuous event accumulation.

Our subgraph extraction strategy limits data for each graph convolution within a K-hop $Egonetwork$, significantly reducing space usage. In Fig. 3 (b), our method's storage space dropped significantly, by two orders of magnitude. This illustrates that our subgraph extraction strategy enhances neural network space efficiency, whether using dense or sparse matrix computations.

Response Time Analysis

Differing from prevailing event datasets, an event camera produces a continuous event stream in real environments. Therefore, when the model detects that there is an event stream input, it should give reliable results as soon as possible, otherwise, it will bring a large inference delay.

In order to further prove that our model can also give satisfactory results when there is only a small number of event inputs, i.e., the model has a lower inference delay, we verified the impact of the number of events on the accuracy of our pipeline. Illustrated in Fig. 3 (c), our pipeline's accuracy improves with accumulating events, yet our model maintains high accuracy even with partial information. Even with only 10% of the event input, our model did not drop more than 10 percent in accuracy. And when the number of events is accumulated to 2000, our accuracy rate can reach 90%. These all prove that our pipeline can make the correct response extremely fast.

Sparsity Analysis

We investigate spike sparsity in ASGCN. Specifically, we conducted experiments using a 6-layer ASGCN on the N-Cars dataset, recording average neuron firing rates across network layers (Fig 4 (a)). The model's maximum firing rate remains below 0.1, with most layers consistently under 0.05. Furthermore, our model's firing rate stabilizes after simulations surpass 28, highlighting network architecture sparsity.

This characteristic ensures ultra-low power consumption and rapid responsiveness. To further verify the asynchronous propagation of spikes on a computational graph shown in Fig. 2 (b), we record the firing neurons of each layer in the SNN at each time step. The results are shown in Fig 4 (b). It is easy to find that there are few neurons that repeatedly fire at adjacent time steps in the ASGCN, which reflects the flow of information between neurons.

Ablation and Hyperparameter Studies

For a more comprehensive analysis of our ASGCN, we examine: (a) the impact of correctly implementing BN; (b) substituting components' effects on performance and power consumption; (c) the impact of hyperparameters on performance. We assess component effectiveness, revealing in Table 3 that performance notably declined when our modified BN was substituted with vanilla BN. Table 3 also demonstrates changes in performance, operations, and power consumption upon substituting different pipeline components, underscoring each component's necessity. Incorporating Table 4, we conduct experiments to probe hyperparameters' impact on performance. Hyperparameter R notably directly influences performance, while hyperparameter D exerts a relatively minor effect on performance but significantly affects computational efficiency.

Conclusion

In this paper, we introduce an ultra efficient asynchronous neuromorphic event processing pipeline based on Asynchronous Spiking Graph Convolution Network (ASGCN). We take full advantage of graph data and spiking neural network to construct our processing pipeline, obtain the sparsity of data by directly reading the raw event stream and converting it into the dynamic graph, and obtain the sparsity of the network architecture through the spiking sparsity of the spiking neural network. Extensive and diverse experiments demonstrate the extreme energy efficiency of our processing pipeline, consuming 10,000 times less energy at comparable performance.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under grant U20B2063, 62220106008, and 62106038, the Sichuan Science and Technology Program under Grant 2024NSFTD0034 and 2023YFG0259, the Open Research Fund of the State Key Laboratory of Brain-Machine Intelligence, Zhejiang University (Grant No.BMI2400020).

References

Alon, U.; and Yahav, E. 2021. On the Bottleneck of Graph Neural Networks and its Practical Implications. In *Proc. of ICLR*.

Amir, A.; Taba, B.; Berg, D.; Melano, T.; McKinstry, J.; Di Nolfo, C.; Nayak, T.; Andreopoulos, A.; Garreau, G.; Mendoza, M.; et al. 2017. A low power, fully event-based gesture recognition system. In *Proc. of CVPR*.

Bi, Y.; Chadha, A.; Abbas, A.; Bourtsoulatze, E.; and Andreopoulos, Y. 2019. Graph-based object classification for neuromorphic vision sensing. In *Proc. of ICCV*.

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*.

Cannici, M.; Ciccone, M.; Romanoni, A.; and Matteucci, M. 2019. Asynchronous convolutional networks for object detection in neuromorphic cameras. In *Proc. of CVPR*.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Proc. of NeurIPS*.

Deng, Y.; Chen, H.; Liu, H.; and Li, Y. 2022. A Voxel Graph CNN for Object Classification with Event Cameras. In *Proc. of CVPR*.

Falanga, D.; Kleber, K.; and Scaramuzza, D. 2020. Dynamic obstacle avoidance for quadrotors with event cameras. *Science Robotics*.

Fey, M.; and Lenssen, J. E. 2019. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428*.

Fey, M.; Lenssen, J. E.; Weichert, F.; and Müller, H. 2018. SplineCNN: Fast Geometric Deep Learning With Continuous B-Spline Kernels. In *Proc. of CVPR*.

Gallego, G.; Delbrück, T.; Orchard, G.; Bartolozzi, C.; Taba, B.; Censi, A.; Leutenegger, S.; Davison, A. J.; Conradt, J.; Daniilidis, K.; et al. 2020. Event-based vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*.

Gallego, G.; Lund, J. E.; Mueggler, E.; Rebecq, H.; Delbruck, T.; and Scaramuzza, D. 2017. Event-based, 6-DOF camera tracking from photometric depth maps. *IEEE transactions on pattern analysis and machine intelligence*.

Gehrig, D.; Loquercio, A.; Derpanis, K. G.; and Scaramuzza, D. 2019. End-to-end learning of representations for asynchronous event-based data. In *Proc. of ICCV*.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *Proc. of ICML*.

Graham, B.; Engelcke, M.; and Van Der Maaten, L. 2018. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proc. of CVPR*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proc. of CVPR*.

Horowitz, M. 2014. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*.

Kaiser, J.; Mostafa, H.; and Neftci, E. 2020. Synaptic plasticity dynamics for deep continuous local learning (DECOLLE). *Frontiers in Neuroscience*, 14: 424.

Kim, H.; Leutenegger, S.; and Davison, A. J. 2016. Real-time 3D reconstruction and 6-DoF tracking with an event camera. In *Proc. of ECCV*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proc. of ICLR*.

Lagorce, X.; Orchard, G.; Galluppi, F.; Shi, B. E.; and Benosman, R. B. 2016. Hots: a hierarchy of event-based time-surfaces for pattern recognition. *IEEE transactions on pattern analysis and machine intelligence*.

Li, H.; Liu, H.; Ji, X.; Li, G.; and Shi, L. 2017. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience*.

Li, Y.; Deng, S.; Dong, X.; Gong, R.; and Gu, S. 2021a. A Free Lunch From ANN: Towards Efficient, Accurate Spiking Neural Networks Calibration. *arXiv preprint arXiv:2106.06984*.

Li, Y.; Zhou, H.; Yang, B.; Zhang, Y.; Cui, Z.; Bao, H.; and Zhang, G. 2021b. Graph-based asynchronous event processing for rapid object recognition. In *Proc. of ICCV*, 934–943.

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 *Proc. of AAAI*.

Messikommer, N.; Gehrig, D.; Loquercio, A.; and Scaramuzza, D. 2020. Event-based asynchronous sparse convolutional networks. In *Proc. of ECCV*.

Mitra, S.; Fusi, S.; and Indiveri, G. 2008. Real-time classification of complex patterns using spike-based learning in neuromorphic VLSI. *IEEE transactions on biomedical circuits and systems*.

Mitrokhin, A.; Hua, Z.; Fermuller, C.; and Aloimonos, Y. 2020. Learning visual motion segmentation using event surfaces. In *Proc. of CVPR*.

Monti, F.; Boscaini, D.; Masci, J.; Rodolà, E.; Svoboda, J.; and Bronstein, M. M. 2017. Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs. In *Proc. of CVPR*.

Neftci, E. O.; Mostafa, H.; and Zenke, F. 2019. Surrogate gradient learning in spiking neural networks: Bringing the

power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*.

Orchard, G.; Jayawant, A.; Cohen, G. K.; and Thakor, N. 2015a. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in neuroscience*.

Orchard, G.; Meyer, C.; Etienne-Cummings, R.; Posch, C.; Thakor, N.; and Benosman, R. 2015b. HFirst: A temporal approach to object recognition. *IEEE transactions on pattern analysis and machine intelligence*.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch.

Ramesh, B.; Yang, H.; Orchard, G.; Le Thi, N. A.; Zhang, S.; and Xiang, C. 2019. Dart: distribution aware retinal transform for event-based cameras. *IEEE transactions on pattern analysis and machine intelligence*.

Rathi, N.; and Roy, K. 2020. DIET-SNN: Direct input encoding with leakage and threshold optimization in deep spiking neural networks. *arXiv preprint arXiv:2008.03658*.

Rebecq, H.; Ranftl, R.; Koltun, V.; and Scaramuzza, D. 2019. High speed and high dynamic range video with an event camera. *IEEE transactions on pattern analysis and machine intelligence*.

Rueckauer, B.; Lungu, I.-A.; Hu, Y.; Pfeiffer, M.; and Liu, S.-C. 2017. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*.

Schaefer, S.; Gehrig, D.; and Scaramuzza, D. 2022. AEGNN: Asynchronous Event-based Graph Neural Networks. In *Proc. of CVPR*.

Sekikawa, Y.; Hara, K.; and Saito, H. 2019. Eventnet: Asynchronous recursive event processing. In *Proc. of CVPR*.

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 *Proc. of CVPR*.

Wang, Q.; Zhang, Y.; Yuan, J.; and Lu, Y. 2019. Space-time event clouds for gesture recognition: From RGB cameras to event cameras. In *Proc. of WACV*.

Wang, Y.; Zhang, M.; Chen, Y.; and Qu, H. 2022. Signed Neuron with Memory: Towards Simple, Accurate and High-Efficient ANN-SNN Conversion. In *Proc. of IJCAI*.

Wu, Y.; Deng, L.; Li, G.; Zhu, J.; and Shi, L. 2018. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *Proc. of ICLR*.

Xu, Q.; Fang, X.; Li, Y.; Shen, J.; Ma, D.; Xu, Y.; and Pan, G. 2024. RSNN: Recurrent Spiking Neural Networks for Dynamic Spatial-Temporal Information Processing. In *Proceedings of the 32nd ACM International Conference on Multimedia*, 10602–10610.

Xu, Q.; Li, Y.; Shen, J.; Liu, J. K.; Tang, H.; and Pan, G. 2023. Constructing deep spiking neural networks from artificial neural networks with knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7886–7895.

Xu, Q.; Li, Y.; Shen, J.; Zhang, P.; Liu, J. K.; Tang, H.; and Pan, G. 2022. Hierarchical spiking-based model for efficient image classification with enhanced feature extraction and encoding. *IEEE Transactions on Neural Networks and Learning Systems*.

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. In *Proc. of ICCV*, 10221–10230.

Zhu, A. Z.; Thakur, D.; Özaslan, T.; Pfrommer, B.; Kumar, V.; and Daniilidis, K. 2018. The multivehicle stereo event camera dataset: An event camera dataset for 3D perception. *IEEE Robotics and Automation Letters*.

Zhu, Z.; Hou, J.; and Lyu, X. 2022. Learning Graph-embedded Key-event Back-tracing for Object Tracking in Event Clouds. In *Proc. of NeurIPS*.