EventNet: Asynchronous Recursive Event Processing

Yusuke Sekikawa[†], Kosuke Hara[†], and Hideo Saito[‡]

[†]Denso IT Laboratory, [‡]Keio University

Abstract

Event cameras are bio-inspired vision sensors that mimic retinas to asynchronously report per-pixel intensity changes rather than outputting an actual intensity image at regular intervals. This new paradigm of image sensor offers significant potential advantages; namely, sparse and non-redundant data representation. Unfortunately, however, most of the existing artificial neural network architectures, such as a CNN, require dense synchronous input data, and therefore, cannot make use of the sparseness of the data. We propose EventNet, a neural network designed for real-time processing of asynchronous event streams in a recursive and event-wise manner. EventNet models dependence of the output on tens of thousands of causal events recursively using a novel temporal coding scheme. As a result, at inference time, our network operates in an event-wise manner that is realized with very few sum-of-the-product operations—look-up table and temporal feature aggregation—which enables processing of 1 mega or more events per second on standard CPU. In experiments using real data, we demonstrated the real-time performance and robustness of our framework.

1. Introduction

Existing frame-based paradigms—dense synchronous video stream acquisition and dense/batch processing—cannot scale to higher frame rates or finer temporal resolutions because the computational complexity grows linearly with the processing rate or temporal resolution (Fig. 1 top). The grows of the computational complexity comes from the redundant synchronous measurement/transmission of dense intensity frames for unchanged pixels and the following redundant signal processing algorithm, such as convolutional neural networks (CNNs) [24, 7, 4, 20, 28], which computes the sum of the products, even for the unchanged pixels. Furthermore, the same frames are computed multiple times (temporal sliding window operation) to model temporal dependencies [24].

The event-based camera [10] discards the frame-based paradigm and instead adopts a bio-inspired approach of independent and asynchronous pixel brightness change measurement without redundancy. This new type of data acquisition has the potential to enable a new paradigm of high-speed, non-redundant signal processing using the naturally compressed non-redundant event stream.

Our research goal was to develop a neural network architecture that can process an extremely high-rate,[1] variable length, and non-uniform raw event stream in real time. To this end, we proposed EventNet, a neural network designed for the real-time processing of an asynchronous event stream in an event-wise manner. Our main contributions are summarized as follows:

Recursive Architecture
We proposed a recursive algorithm by formulating dependence on causal events (which could be tens of thousands) to the output recursively using a novel temporal coding and aggregation scheme that comprises a complex phase rotation and complex $\max$ operation.

Lookup Table Realization of MLP
The deep multi-layer-perceptron (MLP) appears in the recursive formula and dominates most of the computation. It was replaced by a lookup table (LUT) at inference time by the factorization of the temporal term. This replacement removed most of the sum-of-product operations of the MLP.

Asynchronously Two Module Architecture
The entire network was separated into two modules working asynchronously; an event-driven module that updates the global feature immediately as it receives a new event, and the on-demand module that computes the final output with a lightweight MLP. This separate architecture avoids a wasteful computation of output that is not used by applications (Fig. 1 bottom).

Applicability to Real-World Data
We demonstrated the applicability of EventNet for real-world applications using publicly available datasets. We applied EventNet to event-wise semantic segmentation, ob-

[1]Maximum event rate of the iniVation DAVIS240 camera is 12 mega events per second.

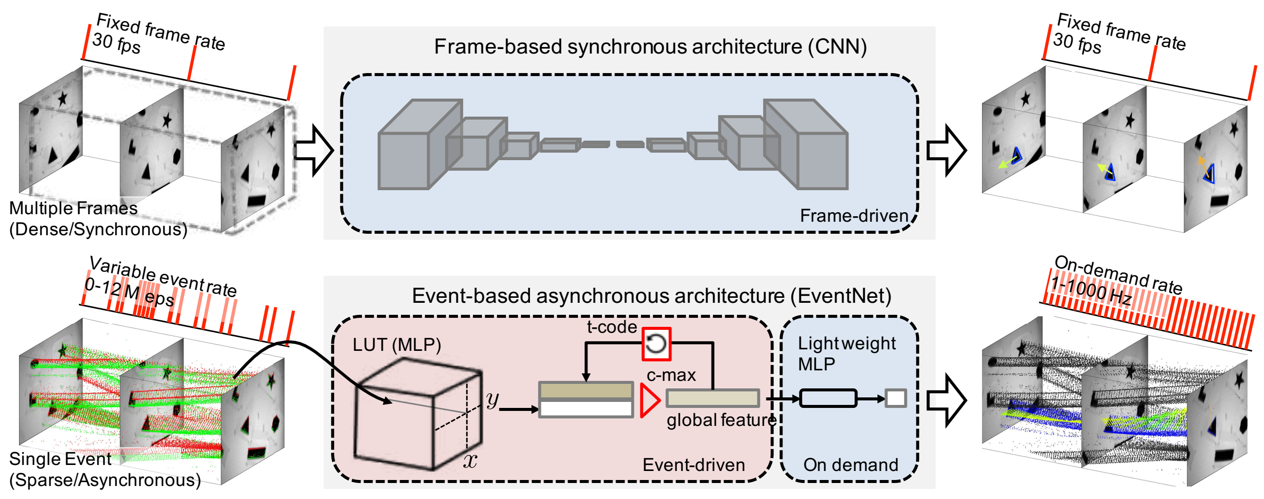

Figure 1. **Overview of asynchronous event-based pipeline of EventNet (inference) in contrast to the conventional frame-based pipeline of CNN.** Top: Conventional frame-based paradigm (CNN). Bottom: Proposed event-based paradigm (EventNet). In the conventional frame-based paradigm, the computation is redundant, which prevents the algorithm from scaling to higher frame rates. Due to the slow output rate, applications are obliged to use old results. Our EventNet can directly process sparse events in an event-wise manner without densification; it processes only changed pixels, and thus, there is no redundancy. When a new event arrives, global features (summary of current states) are updated with a highly efficient recursive operation using an LUT at the event rate. And, when the application requests output, it is computed on demand by feeding the global-features to a lightweight MLP. This pipeline is extremely efficient and can process more than 1 mega events per second (MEPS); it can also respond to the on-demand request at 1 kHz or more with a standard CPU. Examples of input event streams are shown (*green* and *red* indicate positive and negative events, respectively) on the right, and the results of the global object motion estimation and event-wise semantic segmentation are shown on the right (events classified as triangles are shown in *blue*, and others are shown in *grey*, and the estimated motion vector of a triangle is shown as an arrow where the color encodes the angle of the motion). The data from [13] are used for this experiment. Notes: MLP = multi-layer perceptron, t-code = temporal coding, and c-max = complex max pooling.

ject motion estimation, and ego-motion estimation. These demonstrated real-time performance in the CPU. The event-driven module—comprising an event-wise LUT, a temporal code, and $\max$—worked extremely fast and it could process about 1 mega event per second (MEPS) on a standard CPU. Further, the on-demand inference module was capable of responding to a request from an application at 1 kHz or more on the CPU.

2. EventNet

2.1. Event-Based Camera

Each pixel of an event-based camera asynchronously measures intensity levels and reports an event—(x, y, p, t) pixel location, polarity indicating the positive or negative changes in intensity, and time stamp—when changes in intensity are detected. We considered a sequence of events within a τ ms interval based on the time stamp t_j of the j-th event as $\mathbf{e}_j := \{e_i | i = j - n(j) + 1, ..., j\}$, where each event e_i is a quartet of its $(x_i, y_i, p_i, \Delta t_{j,i})$, where $\Delta t_{j,i}$ represents the time difference $\Delta t_{j,i} = (t_j - t_i)$, and $\mathbf{e}_j$ is updated when a new $(j+1)$-th event arrives by adding e_{j+1} and removing events that come out of the τ ms interval. Thus the length of the sequence $n(j)$ changes dynamically (Fig. 3 right).

2.2. Problem Statement

We considered the function f, realized by neural networks, to estimate target value y_j given event stream $\mathbf{e}_j$: $y_j = f(\mathbf{e}_j)$. Events come at a nonuniform rate, ranging from 0 to millions per second, and the network needs to process the variable-rate data in an event-wise manner. Our goal was to realize a trainable event-driven neural network f that satisfies all of the following conditions:

i) End-to-End Trainable
To realize the practical supervised learning of applications, the network needs to be trainable end-to-end by using error backpropagation (BP) using a supervised signal.

ii) Event-Wise Processing
The events are spatially sparse and temporally nonuniform. The dependence of output y_j on the sparse signal $\mathbf{e}_j$ needs to be processed in an event-wise manner without densification to voxel representation to avoid redundant computation.

iii) Efficient Recursive Processing
The event rate could be very high (more than 1 MEPS), de-

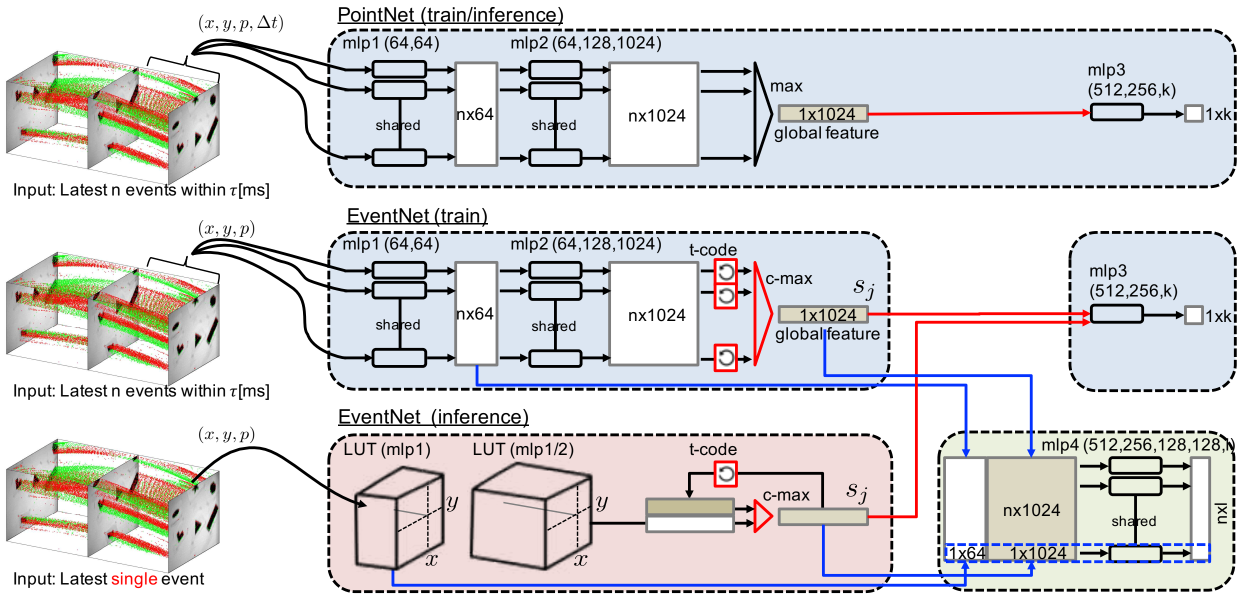

Figure 2. **EventNet architecture** The network architecture of our EventNet is shown in comparison with PointNet. Our network has the novel temporal coding layer of Eq. 3. Thanks to this layer, the dependence on the sequence of events is computed recursively. Furthermore, the most computationally significant part (mlp1 and mlp2, which are trained using standard error backpropagation) are realized as a lookup table (LUT) after training, which is significantly faster than an MLP. As a result, EventNet processes streams of events efficiently in an event-driven manner—compute per-event feature by LUT, apply the temporal code to the global feature, and aggregate global feature by max pooling of the two vectors—which is repeated recursively as it receives a new event. Numbers in bracket are layer sizes. Batch normalization [5] is used for all layers except output layers. Similar to PointNet, EventNet has a variant of architecture that can output on a per-event basis, which is realized with mlp4 by a concatenation of local feature and global feature(*blue* line).

pending on the scene, and thus, the size of $\mathbf{e}_j$ could be large (tens of thousands depending on the event rate and window size τ). It is quite impossible to process such a large variable length nonuniform event-stream $\mathbf{e}_j$ in a batch manner at a very high event rate. Thus, an efficient algorithm that can recursively process the stream is required.

iv) Local Permutation Invariance
An ideal event-stream recognizer f needs to be invariable against data-point permutations in a *short time* window, while having sensitivity to capture *longer time* behaviors. Invariance against a temporally local permutation is essential because the order that event data emerges in a temporal vicinity can be generally thought of as stochastic due to the limited temporal resolution (e.g., 1 μs) and the noise of time stamps (Fig. 3 left). Thus, the order of incoming events may change even if the same scene is observed with the same camera motion. On the other hand, the long-range temporal evolution of an event needs to be modeled to capture motion in a scene.

2.3. Symmetric Function

To cope with the permutation, a simple MLP or RNN with randomly permuted sequences is not a feasible choice because it is difficult to scale to thousands or tens of thousands of input elements [17]. In addition, it is impossible to be totally invariant to the permutation [26]. The PointNet architecture [17] solves the problem in a theoretical and concise way by approximating the function as

$$y_j = f(\mathbf{e}_j) \approx g(\max(h(e_{j-n(j)+1}), ..., h(e_j))), \qquad (1)$$

where , $h : \mathbb{R}^4 \to \mathbb{R}^K$, $\max : \underbrace{\mathbb{R}^K \times ... \times \mathbb{R}^K}_{n(j)} \to \mathbb{R}^K$, and $g : \mathbb{R}^K \to \mathbb{R}$. They approximate h and g using an MLP. Because of the symmetric function $\max$, the permutation of events does not change the output y_j. Note that $\max$ operates independently for each dimension of $\mathbb{R}^K$.

2.4. EventNet

PointNet focuses on processing sets of vectors such as a 3D point cloud in a batch manner. When we attempt to use PointNet to sequentially process a stream of events, a huge amount of computation of h realized by the MLP and $\max$ makes real-time processing difficult: For all $n(j)$ events in $\mathbf{e}_j$, when a new $(j+1)$-th arrives, the time difference $\Delta t_{j,i} = (t_j - t_i)$ changes to $(t_{j+1} - t_i)$; thus, most of the $n(j)$ event that has already been processed by h (realized by deep MLP) needs to be processed again by h when we receive the new event as long as it is within ther τ ms time window. If the function $\max$ is the function of the set of $n(j)$ high-dimensional feature vectors and the vector changes with the same situation as above, we need to compute $\max$ within all $n(j)$ feature vectors at the event rate. The single cycle of these computations are themselves intensive since $n(j)$ may be thousands or tens of thousands in common scenarios. Furthermore, these two computations should run on the event rate (could be more than 1 MEPS). These issues make it impossible to use PointNet to process event streams in real time.

To overcome the above difficulty in processing the event streams, we proposed EventNet, which processes the sparse signal recursively rather than processing large numbers, $n(j)$, of events in batch manner.

Temporal Coding

Since the function h is a function of time difference Δt, the network is required to compute h tens of thousands of times for the same event (as long as the event exists within the time window) as the new event is being received. Simply removing Δt from the input e creates a loss of important temporal information, resulting in a deterioration of the performance (this will be discussed in Section 3.5). To avoid the multiple time computations of h for the same events while keeping the temporal information, we removed the dependence on Δt from h and instead introduced a temporal coding function c to encode the information of Δt as $h(e_i) = c(h(e_i^-), \Delta t_{j,i})$, where $e^- := (x, y, p)$. Then Eq. 1 becomes

$$f(\mathbf{e}_j) \approx g(\max(c(z_{j-n(j)+1}, \Delta t_{j,j-n(j)+1}), ..., c(z_j, 0))), \tag{2}$$

where, $z_i = h(e_i^-) \in \mathbb{C}^K$. Using this formulation, we need to compute h only once for each observed event; however, c and $\max$ need to be computed for all events in the time window every time a new event arrives.

Recursive Processing

We considered processing $\mathbf{e}_{j-1}$ and $\mathbf{e}_j$ sequentially. Let the time difference of latest time stamp be $\delta t_j := t_j - t_{j-1}$, and assume the norm of each element of z_i is less than 1 (by $\tanh$). We want to make Eq. 2 recursive, cf, computing $\max$ at j using $\max$ at $j-1$ and event e_j. For this, the composition of $\max$ and c need to be recursive. The $\max$ is not recursive for general time series vectors unless the window size is ∞, so we propose temporal coding function c of Eq. 3, which guarantees recursiveness:

$$a_{j,i} = c(z_i, \Delta t_{j,i}) = \left[|z_i| - \frac{\Delta t_{j,i}}{\tau}\right]^+ \exp\left(-\mathrm{i}\frac{2\pi \Delta t_{j,i}}{\tau}\right), \tag{3}$$

where the first term decays the input linearly to the elapsed time, and the second term encodes the temporal information by complex rotation. As c decays the input linearly on elapsed time (constant decay within the fixed time window), it satisfies the relation $\Delta t_{j,i} = \sum_{k=i,...,j} \delta t_k$. The norm of each element of a that is older than τ is always zero, and the complex rotation of each feature represents elapsed time by sequentially rotating the element by $2\pi\delta t_j/\tau$. This coding function makes the composition of $\max$ and c recursive for a finite window size,

$$\begin{aligned}\max((c(z_{j-n(j)}, \Delta t_{j-n(j)}), ..., c(z_{j+1}, 0))) \\ = \max(c(s_j, \delta t_j), z_{j+1}),\end{aligned} \tag{4}$$

where s_j is the result of $\max$ at time t_j. Putting Eq. 3 into Eq. 2, we get the following recursive algorithm:

$$f(\mathbf{e}_{j+1}) \approx g(\max(c(s_j, \delta t_j), h(e_{j+1}^-)), \tag{5}$$

where global feature $s_j := \max(c(s_{(j-1)}, \delta t_{j-1}), h(e_j^-))$ is updated recursively. This formulation has favorable characteristics for sequential event processing: i) we only need to compute $h(e_i^-)$ once for each event; ii) c is computed only for s, not for all $n(j)$ vectors; and iii) $\max$ is computed between only two vectors instead of between all $n(j)$ vectors. A permutation of events that has the same time stamp does not change the result as desired, and because the coding function of Eq. 3 is approximately constant within a small temporal interval, small perturbations of time stamps from noise will result in small changes in output y_j as long as the function g is smooth.

Note that $\max$ in the above equation is defined in a set of complex values. Let $a_1^k, ..., a_n^k \mid a_i^k \in \mathbb{C}$ be a k-th channel of temporal sequence (i.e., sequence of feature vectors after the temporal coding), then the complex $\max$ is defined as;

$$\max(a_1^k, ..., a_n^k) = a_i, \textit{ where } i = \arg\max_i(|a_i^k|). \tag{6}$$

LUT Implementation of MLP

Because the spatial position and polarity from event-based cameras are discrete, there are only $W \times H \times 2$ patterns in inputs e^- (spatial position and polarity). Therefore, we can precompute results of h, and utilize LUT in inference time for computing high-dimensional vectors. This is considerably faster than Deep MLP h, which contains a large number of product-sum operations.

Summary

EventNet satisfies all the conditions described in Section 2.2. It is trainable using BP in supervised manner because it uses differentiable MLP, it can efficiently process sparse event signals without densification with a novel recursive architecture of Eq. 5. It can also capture a long-range temporal evolution by a complex rotation while also being invariant to the small temporal vicinity due to the max operation.

2.5. Network Architecture

The EventNet model of Eq. 5 is realized in the architecture shown in Fig. 2. The function h is realized as mlp1 and mlp2, the function g is realized as mlp3 for global-estimations, and mlp4 for event-wise estimation.

Depending on the application, the required rate of output varies, and computing a final output at the event rate (1 MEPS) is a waste of computation as most of the results are not used by the application. In addition, 1000 Hz may be more than enough for many applications. To achieve real-time event processing and high-rate estimation without wasteful computation, our network comprises two separate modules that work asynchronously with each other.

Event-Drive Processing Module
This module operates in an event-driven manner: when a new event e_j arrives at the network asynchronously, it is immediately processed to update the global feature vector s_j, which is realized by the recursive algorithm of Eq. 5. Furthermore, mlp1 and mlp2 are realized as an LUT, which is much faster than feed-forwarding deep MLP.

On-Demand Processing Module
This module operates on demand from an application. When the application requires the latest estimation, the output is computed on demand with mlp3 or mlp4. Because the input to mlp3 is a single vector, its computation is reasonably fast, and 1000 Hz with a standard CPU is easily achieved.

Asymmetric Training/Inference
As shown in Fig. 2 the network structure of EventNet differs in training and inference; it utilizes a batch structure when training. In principle, EventNet can be trained recursively using Eq. 5 as long as batch normalization (BN) is absent. The recurrent structure does not allow for the computation of batch statistics. However, we adopt the batch structure during training for two reasons: i) to use BN, which plays a very important role, based on our experience; and ii) to parallelize the computation of MLP for efficiency. The structure of EventNet when training are similar to PointNet [17] except for the temporal coding layer (Fig. 2 middle), and they are trained in largely the same way.[2] However, the temporal coding makes a huge structural difference and a large computational gain in inference time (Fig. 2 bottom, Table 2).

3. Experiments

The purpose of the experiments was to evaluate the computational efficiency and robustness of EventNet in a practical application. To the best of our knowledge, there are no end-to-end trainable neural networks capable of processing raw event streams in real time. Nevertheless, we compare EventNet with PointNet as it is a successful method that can directly process reasonably good-sized point set. We want to emphasize that because PointNet cannot process event streams in a recursive manner, as EventNet does, it is thousands of times slower (depending on the event rate) than EventNet at such processing and cannot process an event stream in real-time.

We will first describe the learning procedure (Section 3.1) and datasets used for the experiments (Section 3.2). Next, we demonstrate the real-time processing capability of EventNet in several real-world tasks (Section 3.3 and Section 3.4). And last, we provide the results of an ablation study to reveal the effects of core components of EventNet (Section. 3.5).

3.1. Learning Procedure

Mini-Batch Construction

Training MLPs in EventNet requires processing events

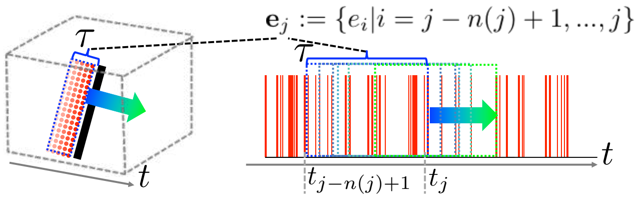

Figure 3. **Event data composition** Left: An event stream from moving rod-like objects is illustrated. Intensity changes caused by the movements of the objects generated an event stream. Within a short period of time, different pixels of the sensor detect the intensity changes almost simultaneously. Thus permutation between events happens even if the camera captures the same scene. Right: We considered the event-stream $\mathbf{e}^t := \{e_i | i = n(j) + 1, ..., j\}$ within a fixed temporal window τ. The number of events in the stream, $n(j)$, changed dynamically.

streams in a batch manner (Fig. 2 EventNet (train)), although it can process event streams recursively in inference time. Given an entire event-stream for training, the single event stream $\mathbf{e}_j$ for training was composed as follows. An event that corresponds to the latest event within the temporal window is randomly selected from the entire event stream (let the index of the event be j). Then, cut out an event stream on a τ ms interval based on t_j, $\mathbf{e}_j := \{e_i | i = j - n(j) + 1, ..., j\}$ (Fig. 3). Optionally, to increase the variety in training data, it can then be randomly spatially cropped. Note that the length of each sequence $n(j)$ is different since event rates change depending on the scene or the motion of the camera.

Optimization
All networks, including the one from the ablation study, were trained using the same protocol. Single-epoch consisted of randomly composed $8,000$ event stream, and the training was carried out for 500 epochs. For the optimization, we used Adam [6] with the recommended hyper-parameter settings of: $\beta_1 = 0.9, \beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The initial learning was 0.0002, and it was divided by 2 every 20 epochs until epoch 100, after which it was kept con-

[2]In terms of implementation, they are greatly different because EventNet needs to handle variable-length data The implementation details are described in the supplemental material.

Table 1. **Quantitative evaluation using ETHTED+ and MVSEC.** Quantitative evaluations of our EventNet and PointNet [17] are shown. For experiments using ETHTED+, we report global accuracy (GA) and mean intersection of union (mIoU) for semantic segmentation and L2 regression error for object motion. For experiments using MVSEC, we evaluated the L2 regression error for ego-motion estimation. Our EventNet achieved a comparable performance to PointNet while achieving real-time performance. On the bottom, the results of EventNet when disabling each term of temporal coding in Eq. 3 are shown (see main text for the abbreviation). The number in parentheses indicates the standard deviation of the results.

		ETHTED+			MVSEC	Real-time processing at 1 MEPS
		Semantic segmentation		Object-motion	Ego-motion	
		GA [%]	mIoU [%]	error [pix/τ]	error [deg/sec]	
PointNet		98.9	97.4(0.13)	3.14(0.08)	4.55	NO
EventNet		99.2	97.5(0.22)	3.11(0.28)	**4.29**	YES
Ablation	w/o TD	**99.4**	**98.8**(0.16)	**3.08**(0.32)	—	NO
	w/o TR	98.1	97.9(0.11)	3.74(0.06)		YES
	w/o ALL	98.3	97.1(0.25)	4.14(0.32)		NO

Table 2. **Computational complexity.** Computational times (μs) for processing a single event with our EventNet and PointNet [17] are shown. To compute this statistic, we considered the case of an event rate of 1 MEPS, which approximately corresponds to the highest event-rate scenario in the ETHTED+ dataset. The spatial position of the synthetic event was generated randomly and was temporarily uniform at the rate. The statistic was measured by a single core 3.2GHz Intel Core-i5. The temporal window size of $\tau = 32$ ms is assumed. To update the global feature s_j using a newly received event e_j, mlp1/2 of PointNet needed to process a stream $\{\mathbf{e}_j | i = j - n(j) + 1, ..., j\}$ where $n(j)$ can be thousands or tens of thousands. Similarly, $\max$ needs to process the $n(j)$ high-dimensional vector. Conversely, input to EventNet was a single event e_j because of the recursive formula of Eq. 5. Furthermore deep MLP is replaced with an LUT, which results in an extremely fast computation time of processing a single event in about 1 μs. Thus, it can process (update global feature s_j) events of , at most, 1 MEPS. The computation time for mlp3/mlp4 is less than 1 ms meaning the application can query the results at more than 1000 Hz. We also report the computation time of naive mlp1/2 in parentheses for EventNet and observed that the processing of the MLP was accelerated about $45\times$ by the LUT.

	#input mlp1	#input max	mlp1/2	max pool(+t-code)	total	mlp3	mlp4
PointNet	$n(j)$	$n(j)$	936.9×10^3	16.47×10^3	953.3×10^3	$\mathbf{0.58 \times 10^3}$	0.59×10^3
EventNet	1	2	**0.65**(29.27)	**0.36**	**1.01**	0.61×10^3	0.61×10^3

stant. The decay rate for BN started at 0.5 and gradually increased to 0.99. All implementations used the MatConvNet [25] library, and we carried out all our training on a single NVidia V100 GPU.

3.2. Datasets

We used publicly available datasets captured with real event-based cameras for evaluation.

ETHTED+

The first and second applications used the datasets from [13] with our additional hand-annotated segmentation label[3] for *shape rotation sequence* (ETHTED+). The supplemental annotation is given on a per-event basis, indicating whether it came from the triangle or not. The camera used for this dataset was the DAVIS 240, which has an array size of 180 × 240. Each training event stream was randomly cropped spatially to 128 × 128 from the original 180 × 240 to increase the variety of spatial displacements. The average event rate of this data set (center 128 × 128 region) was 0.16 MEPS. We used the first 50 seconds of the sequence for training, and the other 10 seconds was used for testing. The temporal window size of $\tau = 32$ ms was used for this dataset.

MVSEC

The third application used MVSEC [30], consisting of event sequences captured in road scenes, which are much larger than ETHTED+; thus, it had more variation in the input data space. The camera used for this dataset was the DAVIS 346, which has an array size of 260 × 346. The average event rate of this data set was 0.35 MEPS. To remove noise, we applied the nearest neighbor filter followed by refractory period filter, as described in [16], where we used 5 ms and 1 ms as temporal window size respectively. We used *outdoor day1*, *outdoor day2*, *outdoor night1*, and *outdoor night2* sequences for training and used *outdoor night3* for testing. A temporal window size of $\tau = 128$ ms was used for this dataset.

[3]The additional label will be published with this paper.

3.3. Applications

We demonstrated the applicability of EventNet for several real-world tasks.

Target Motion Estimation

In this application, the network estimates the motion of a specific object (triangle) using global-estimation network (see *red* line in Fig. 2 with mlp3). We used ETHTED+ for this experiment. Using the class label of events, we computed the motion $[u, v]$ of the triangle by linearly fitting the centroid position of events within 33 ms intervals. The input event stream was processed to compute global features at the event rate, and the target values $[u, v]$ were computed on demand at 1000 Hz using the global feature. When testing, the cropping region was fixed to the center region.

Semantic Segmentation

In this application, the network estimates the class label of each event (triangle or others) using event-wise-estimation network (see *blue* line in Fig. 2 with mlp4). We used ETHTED+ for this experiment. The input event stream was processed to compute per-event features from mlp1 and global features at the event rate. The target values which was the probability of each class, was computed on demand using concatenation of global features and local features. We note that temporal information may be less important for this task because we can still determine the shape from the event stream even if the temporal information was lost. The network for this application was trained jointly with the global-estimation network sharing mlp1/2. An example of processing results for object-motion estimation and semantic segmentation is shown in Fig. 1.

Ego-Motion Estimation

In this application, the network estimates ego-motion of the camera using global-estimation network. We used MVSEC for this experiment and the yaw-rate, which was provided with the dataset was used as a target values. The qualitative results are shown in Fig. 4.

3.4. Quantitative Comparison

Estimation accuracy for the three applications are reported in Table 1 on top, and computational times are reported in Table 2. In summary, EventNet achieved a performance that is comparable to PointNet while realizing less than 1 μs processing for a single event, proving it can achieve real-time processing up to 1 MEPS, which covers most practical scenarios.

Estimation Accuracy

As shown in Table 1, our model achieved comparable performance to PointNet in all three of the experiments, while also achieving real-time performance using much less memory usage. For the object motion estimation task, we conducted preliminary experiments to see the effects of different time windows of 8, 16, and 32 ms. The largest (32 ms) performed the best with our EventNet, but the performance of PointNet was almost the same across the different time windows. The results and a discussion of the results are reported in the supplemental material.

Computation Time

We assumed the system displayed in Fig. 1 for the comparison shown in Table 2. The required number of operations for the MLP and $\max$ was about $n(j)\times$ less than processing in batch manner due the novel recursive formula used in EventNet. Furthermore, the LUT realization of mlp1/2 improved the computation speed by about $45\times$. Consequently, EventNet processed input event rates of 1 MEPS with standard CPU, covering most of the practical scenarios. Conversely, PointNet cannot process event streams recursively and is, thus, required to process $n(j)$ events within a τ ms time window in a batch manner every time it receives a new event by reprocessing the events again and again. Real-time processing is, therefore, entirely impossible. Note that the per-event computation time of EventNet is not affected by the event rate, while the computational complexity grows linearly with the rate in the case of PointNet.

Further, EventNet is much more memory efficient than PointNet because it requires storage of only one global feature vector as it processes incoming event streams recursively (Eq. 5); and it operates with an LUT and is, thus, not required to store intermediate feature maps.

3.5. Ablation Study

In this section, we discuss our study of the contribution of temporal coding of Eq. 3, the key component of EventNet, which enables highly efficient recursive event-wise processing. For this, we ran the object motion estimation and semantic segmentation experiments using ETHTED+, which is the same as the ones discussed in Section 3.3. We examined the contribution of the temporal decay term $([|z_i| - \frac{\Delta t_i}{\tau}]_+)$ (TD) and the complex temporal rotation term $\exp(-\mathrm{i}\frac{2\pi\Delta t_i}{\tau})$ (TR) from the equation. The results are summarized at the bottom in Table 1. Because segmentation accuracy was equally good for all variants, we discuss the performance in terms of the object motion estimation accuracy, in which the temporal information may be more informative for the estimation.

EventNet Without Temporal Decay

This variant disabled only decay term. Without the decay, the recursion of Eq. 5 was not satisfied; thus, it is necessary to compute $\max$ for all $n(j)$ vectors. Therefore, this variant cannot process 1 MEPS of events in real-time. The estimation accuracy was the best among all variants, including full EventNet.

EventNet Without Temporal Rotation

This variant disabled only the complex temporal rotation term. Because this variant has linear decay term, the recursion of Eq. 5 was satisfied; thus, it can process event streams in an event-wise manner. Actually, this variant is

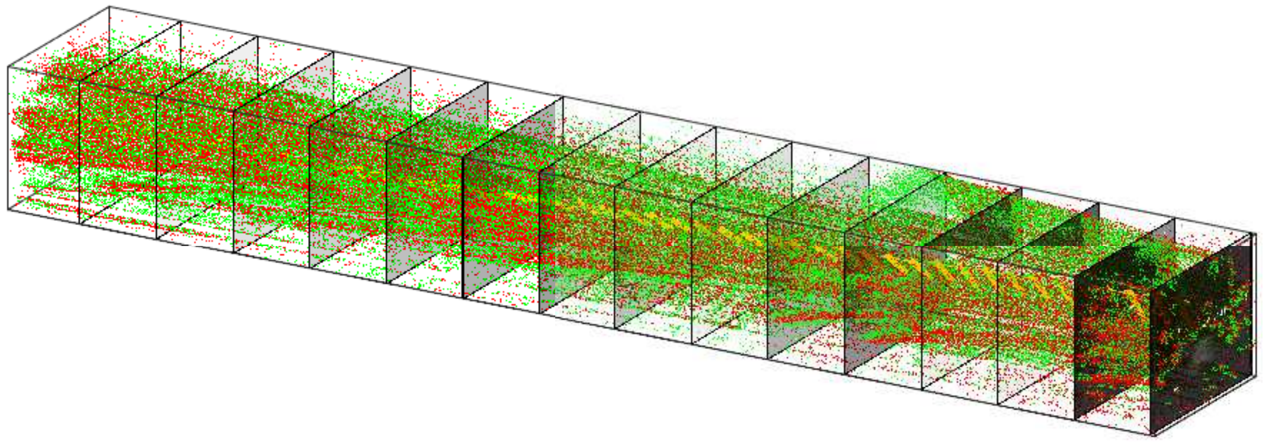

Figure 4. **Snapshot from the MVSEC.** Snapshot from the experiments using MVSEC for the ego-motion (yaw rate) estimation task, where estimated ego motions are shown as arrows (color encoded by the angle). In this application, variable length event sequences are processed recursively in an event-wise manner updating the global feature at the variable event rate, and the ego motion is estimated at the rate of 1000 Hz using the system shown in Fig. 1. Processing the input at this rate in real time is infeasible with the frame-based paradigm.

slightly faster than the full EventNet because it does not compute complex rotation. The performance was not as good as EventNet or EventNet without TD, which may be attributed the lack of temporal information without explicit coding of temporal information as complex rotation.

EventNet Without All

This variant disabled both terms. As a result, the network structure of this variant was the same as PointNet, and the difference to PointNet was that this variant did not include temporal term Δt as input. This variant cannot operate in real time for the same reason that EventNet without TD cannot. This variant showed the worst performance of the variants, which may explained by this variant not having any temporal information.

4. Related Literature

Frame-Based DNNs for Modeling Event Data.

There have been a few studies [11, 31] attempting to model data from event cameras using DNNs. The authors of [11] performed one of the pioneering works using CNNs to model event data. In order to take advantage of existing 2D-CNN architecture, they converted raw spatiotemporal event data into an *event frame* consisting of 2D histograms of positive and negative events. The authors in [31] additionally concatenated *time stamp images* to incorporate temporal information. Most of the existing DNN-based approaches densify the sparse event signal to make use of the architecture of a frame-based paradigm such as with a CNN, which cannot make good use of the sparsity of the event stream.

Spiking Neural Networks

Spiking neural networks (SNNs) [8, 21, 2, 23, 12, 14, 9, 27, 1, 22, 3] are third generation neural networks that are expected to process sparse asynchronous data efficiently. The phased-LSTM [15] can also be interpreted as a kind of SNN. It is specifically designed to process asynchronous data such as event data in event-driven manner, and end-to-end supervised learning using a BP is possible similar to ours. However, due to the architectural difference, its computational cost may be tens or hundreds of times more than our recursive LUT.

Deep Learning on Unordered Sets

PointNet [17] is a pioneering and successful method of dealing with unordered input sets, making use of a permutation invariant operation (such as $\max$) to deal with the unordered data in a concise and structured way. PointNet and subsequent studies [18, 29, 29] work remarkably well for many kind of tasks that require dealing with unordered point sets such as 3D point cloud data. However, since PointNet focuses on processing a set of points in a batch manner, its algorithm cannot process a sparse spatiotemporal event-stream recursively.

5. Conclusion

We proposed EventNet, a trainable neural network designed for real-time processing of an asynchronous event stream in a recursive and event-wise manner. We experimentally showed usefulness for practical applications. We will evaluate it in more challenging scenarios using the recently released event-camera simulator [19]. Our current architecture is the single layer of EventNet, but we will extend this work to a hierarchical structure such as the ones proposed in PointNet++ [18]. Another direction would be the its applications with LiDAR data [29, 29], where we believe our model can be used to process point cloud data without waiting for the frame (360-degree rotation).

References

[1] A. Andreopoulos, H. J. Kashyap, T. K. Nayak, A. Amir, and M. D. Flickner. A low power, high throughput, fully event-based stereo system. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. viii

[2] Z. Bing, C. Meschede, K. Huang, G. Chen, F. Röhrbein, M. Akl, and A. Knoll. End to end learning of spiking neural network based on r-stdp for a lane keeping vehicle, to be appear. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018. viii

[3] M. Davies, T.-h. Lin, C.-k. Lin, S. Mccoy, Y.-h. Weng, A. Wild, and H. Wang. Loihi : A Neuromorphic Manycore Processor with On-Chip Learning. (February), 2018. viii

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. i

[5] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456. JMLR.org, 2015. iii

[6] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. v

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. i

[8] X. Lagorce, G. Orchard, F. Galluppi, B. E. Shi, and R. B. Benosman. Hots: A hierarchy of event-based time-surfaces for pattern recognition. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 39(07):1346–1359, jul 2017. viii

[9] J. H. Lee, T. Delbruck, and M. Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, 2016. viii

[10] P. Lichtsteiner, C. Posch, and T. Delbruck. A 128$\times$128 120 db 15μs latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566–576, Feb 2008. i

[11] A. I. Maqueda, A. Loquercio, G. Gallego, N. N. García, and D. Scaramuzza. Event-based vision meets deep learning on steering prediction for self-driving cars. *CoRR*, abs/1804.01310, 2018. viii

[12] H. Mostafa. Supervised learning based on temporal coding in spiking neural networks. *IEEE transactions on neural networks and learning systems*, 29(7):3227–3235, 2018. viii

[13] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam. *The International Journal of Robotics Research*, 36(2):142–149, 2017. ii, vi

[14] E. O. Neftci, C. Augustine, S. Paul, and G. Detorakis. Event-Driven Random Back-Propagation : Enabling Neuromorphic Deep Learning Machines. 11(June):1–18, 2017. viii

[15] D. Neil, M. Pfeiffer, and S.-C. Liu. Phased lstm: Accelerating recurrent network training for long or event-based sequences. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3882–3890. Curran Associates, Inc., 2016. viii

[16] V. Padala, A. Basu, and G. Orchard. A noise filtering algorithm for event-based asynchronous change detection image sensors on truenorth and its implementation on truenorth. *Frontiers in Neuroscience*, 12:118, 2018. vi

[17] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. iii, v, vi, viii

[18] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017. viii

[19] H. Rebecq, D. Gehrig, and D. Scaramuzza. Esim: an open event camera simulator. In A. Billard, A. Dragan, J. Peters, and J. Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 969–982. PMLR, 29–31 Oct 2018. viii

[20] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. i

[21] B. Z. Sajjad Seifozzakerini, Wei-Yun Yau and K. Mao. Event-based hough transform in a spiking neural network for multiple line detection and tracking using a dynamic vision sensor. In E. R. H. Richard C. Wilson and W. A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 94.1–94.12. BMVA Press, September 2016. viii

[22] E. Stromatias, M. Soto, T. Serrano-Gotarredona, and B. Linares-Barranco. An event-driven classifier for spiking neural networks fed with synthetic or dynamic vision sensor data. *Frontiers in Neuroscience*, 11:350, 2017. viii

[23] A. Tavanaei and A. S. Maida. Bio-inspired spiking convolutional neural network using layer-wise sparse coding and STDP learning. *CoRR*, abs/1611.03000, 2016. viii

[24] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015. i

[25] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. In *Proceeding of the ACM Int. Conf. on Multimedia*, 2015. vi

[26] O. Vinyals, S. Bengio, and M. Kudlur. Order Matters: Sequence to sequence for sets. *ArXiv e-prints*, Nov. 2015. iii

[27] R. M. Wang, T. J. Hamilton, J. C. Tapson, and A. van Schaik. A neuromorphic implementation of multiple spike-timing synaptic plasticity rules for large-scale neural networks. *Frontiers in neuroscience*, 9:180, 2015. viii

[28] L. Zhang, L. Lin, X. Liang, and K. He. Is faster r-cnn doing well for pedestrian detection? In *European Conference on Computer Vision*, pages 443–457. Springer, 2016. i

[29] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. viii

[30] A. Z. Zhu, D. Thakur, T. zaslan, B. Pfrommer, V. Kumar, and K. Daniilidis. The multivehicle stereo event camera dataset: An event camera dataset for 3d perception. *IEEE Robotics and Automation Letters*, 3(3):2032–2039, July 2018. vi

[31] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis. Ev-flownet: Self-supervised optical flow estimation for event-based cameras. *arXiv preprint arXiv:1802.06898*, 2018. viii