Event Transformer. A sparse-aware solution for efficient event data processing

Alberto Sabater [1] Luis Montesano[1,2] Ana C. Murillo[1]
[1]DIIS-I3A, Universidad de Zaragoza, Spain [2]Bitbrain Technologies, Spain.

Abstract

Event cameras are sensors of great interest for many applications that run in low-resource and challenging environments. They log sparse illumination changes with high temporal resolution and high dynamic range, while they present minimal power consumption. However, top-performing methods often ignore specific event-data properties, leading to the development of generic but computationally expensive algorithms. Efforts toward efficient solutions usually do not achieve top-accuracy results for complex tasks. This work proposes a novel framework, Event Transformer (EvT)[1]*, that effectively takes advantage of event-data properties to be highly efficient and accurate. We introduce a new patch-based event representation and a compact transformer-like architecture to process it. EvT is evaluated on different event-based benchmarks for action and gesture recognition. Evaluation results show better or comparable accuracy to the state-of-the-art while requiring significantly less computation resources, which makes EvT able to work with minimal latency both on GPU and CPU.*

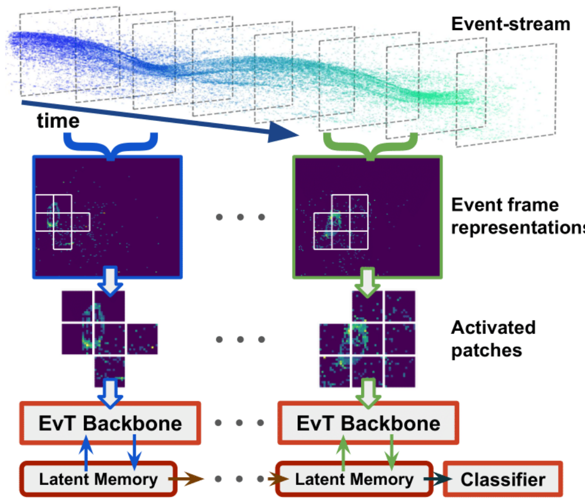


Figure 1. **Framework overview**. Areas (*activated patches*) from event-streams with sufficient event information are extracted from event frame representations and processed by the EvT backbone to update a set of *latent memory vectors*. The latest version of this memory is used for a final event-stream classification.

1. Introduction

Event cameras are bio-inspired sensors that register changes in intensity at each pixel of the sensor array. Contrary to traditional cameras, they work in a sparse and asynchronous manner, with an increased High Dynamic Range and high temporal resolution (in the order of microseconds) with minimal power consumption. These characteristics have pushed the research of many event-based perception tasks such as action recognition [6, 19], body [7, 36] and gaze tracking [2], depth estimation, [14, 48] or odometry [24, 35], interesting for many applications that involve low-resource environments and challenging motion and lightning conditions, such as AR/VR or autonomous driving.

Processing information from event-based cameras is still an open research problem. Top-performing approaches transform event-streams into frame-like representations, throwing away their inherent sparsity, and using heavy processing algorithms such as Convolutional Neural Networks [1, 4, 8, 19] or Recurrent Layers [8, 19]. Other methods that better exploit this sparsity, such as PointNet-like Neural Networks [47], Graph Neural Networks [6, 10] or Spike Neural Networks [23, 39], are more efficient but do not reach the same accuracy. As a result, there is a need for methods that put to good use all the potential of event-based cameras for efficiency and low energy consumption while maintaining high performance.

This work introduces *Event Transformer* (*EvT*), a novel framework (summarized in Fig. 1) designed to tackle the event data sparsity to be highly efficient while obtaining top-accuracy results. We propose: 1) the use of a sparse *patch-based event data representation*, that only accounts for the areas of event-streams with registered information, and 2) a compact transformer-like backbone based on attention mechanisms [44] that naturally work with this patched information. The later, in contrast to previous frame-based methods, requires minimal computational resources by using a set of *latent memory vectors* to bound the quadratic

[1]Code, trained models and supplementary video can be found in: https://github.com/AlbertoSabater/EventTransformer

computational complexity associated to transformer architectures [21, 22], but also, to encode the seen information.

EvT evaluation is run on three public real event data benchmarks of different complexity for long and short event-stream classification (i.e., action and gesture recognition). The results show that EvT achieves better or comparable results to the state-of-the-art. More importantly, our approach presents a significant decrease in computational resource requirements with respect to previous works, with the corresponding power consumption savings, making EvT able to work with minimal latency both in GPU and CPU.

2. Related work

This section summarizes the most common approaches for event data representation as well as event-based Neural Network architectures that process them. It also includes a brief description of different kind of event datasets.

2.1. Event data representation

Event data representations encode the event information related to a time-interval or temporal-window extracted from an event-stream. These representations can be divided in two categories: **event-level representations** usually treat the event data as graphs [5, 6, 10, 47] or point-clouds [37, 45] with minimal pre-processing and keeping the event data sparsity; differently, **frame-based representations** group incoming events into dense frame-like arrays, ignoring the event data sparsity but easing a later learning process. Our work is built on the top of frame-based representations, where we find plenty of variations in the literature. The *time-surfaces* [25] build frames encoding the last generated event for each pixel. SP-LSTM [32] builds frames where each pixel contains a value related to the existence of an event in a time-window and its polarity. The *Surfaces of Active Events* [30] builds frames where each pixel contains a measurement of the time between the last observed event and the beginning of the accumulation time. *Motion-compensated* [34, 46] generate frames by aligning events according to the camera ego-motion. [15] binarizes frame representations in the temporal dimension, achieving a better time-resolution. TBR [19] aggregates binarized frame representations into single-bins frames. M-LSTM [8] uses a grid of LSTMs that processes incoming events at each pixel to create a final 2D representation.

Our work introduces a *patch-based event representation*: we first build a simple frame representation (similar to [15]), and later we divide the resulting frames into a grid of non-overlapping patches, inspired by ideas from Visual Transformers [11]. Generated patches with not sufficient event information are discarded, while the rest are keep as the final event data representation. The proposed hybrid solution presents benefits from both the event-level representations, since we can to a certain extent tackle the sparsity of the event data, but also from the robustness of the frame-based representations.

2.2. Neural Network architectures for event data

Deep learning based techniques have shown promising results working with event-camera data. This section discusses the main existing architectures to process different types of event representations, as well as to aggregate the processed information from several time-windows. Besides, we provide a short overview of Visual Transformers, since they are actually one of the pillars of the architecture proposed in this work to process events.

Architectures for event-representations processing. Methods designed to process event-level representations process the information within each time-window with architectures that take advantage of the event sparsity such as Spike Neural Networks [23, 39, 50], PointNet-style Networks [47] or Graph neural Networks [5, 6, 10]. Differently, frame-based methods often rely on the use of Convolutional Neural Networks to process the frames built for each time-window [1, 4, 8, 19]. Sometimes using also Transformers to process long-range spatial dependencies among CNN-generated features [49]. These methods process whole frames, even if no events are triggered in large parts of this frame. Despite usually achieving higher accuracy than event-level representations, this unnecessary processing makes frame-based approaches consume more computational resources than needed.

Architectures to aggregate several time-windows data. Solutions aimed to analyze long event-streams divide them into different time-windows that are represented and processed with the methods previously described, and then aggregated to perform the final visual recognition task. This aggregation is performed differently in the related work, including the use of Recurrent Networks [19, 49], CNNs [1, 19], temporal buffers [4, 10], or voting between the intermediate results of each time-window [19].

Depending on the aggregation strategy, we consider that an event-processing algorithm is able to perform **online inference** if it can evaluate the information within each time-window incrementally, as it is generated, and then perform the final visual recognition with minimal latency, as opposed to the processing of all the captured information in a large batch. Our approach performs online inference by updating incrementally a set of latent memory vectors with simple addition operations, and processing the resulting vectors with a simple classifier.

Visual Transformers. Transformer architectures, initially introduced for Natural Language Processing [44], have recently gained popularity for Visual Recognition tasks. Vision transformers, work on the features generated by CNNs [9] from RGB images or, more commonly, are directly applied to patches extracted from the input images [11,27]. When working on video data, some methods [3,28] process patches from different video frames together or aggregate the temporal information with LSTM layers [18]. Of special relevance for this work is Perceiver [21, 22], a transformer that uses latent vectors to process the input data and bound the quadratic complexity of transformers.

In this work, we use a Transformer-like architecture to analyze sets of activated patches of variable length, extracted from time-windows within an event-stream. Inspired by [21,22], we use latent vectors for this processing, but differently, we also refine them incrementally with the information extracted from different time-windows.

2.3. Event dataset recordings

Despite their promising applicability, there are still not many large-scale public datasets recorded with these cameras in real scenarios. Therefore, some methods seek for the translation of RGB datasets to their event-based counterpart. Earlier event-based solutions [6, 16, 26, 33, 38] display RGB data in a LCD monitor that they record with an event-camera. More recent works introduce the use of learning-based emulators [13,17,31] to generate event data. Still, these translated datasets cannot fully mimic the event-data nature and introduce certain artifacts, specially on their sparsity and latency. This happens because events are triggered by unrealistic lightning conditions and are frequently dependent on the fixed low frame-rate of a monitor or the movement of a camera over static images. In order to have a more reliable evaluation setup, we focus our experimentation on datasets recorded with event-cameras on real scenarios. More specifically, we train and evaluate EvT in the classification of both short [5] and long [1,42] event-streams related to action and gesture recognition.

3. Event Transformer framework

Different to traditional RGB cameras, event-cameras log the captured visual information in a sparse and asynchronous manner. Each time an intensity change is detected, the camera triggers an event $e = \{x, y, t, p\}$ defined by its location (x, y) within the space of the sensor grid ($H \times W$), the timestamp t of the event (in the order of μs) and its polarity p (either positive or negative change).

This work proposes a novel solution (overview in Fig. 1) for efficient event data processing. Our framework introduces a **patch-based event data representation** that extracts the areas of intermediate frame representations with sufficient logged event information. The resulting information is then processed by our proposed **Event Transformer (EvT)**, a compact Neural Network that uses a set of latent memory vectors to process the incoming event data as well as to encode the information seen so far. The final version of these latent vectors is processed to perform the final visual recognition task, event-stream classification in our case. Next subsections detail the proposed patch-based event data representation and Event Transformer processing, and how the memory vectors are used for the classification of a stream of events.

3.1. Patch-based event data representation

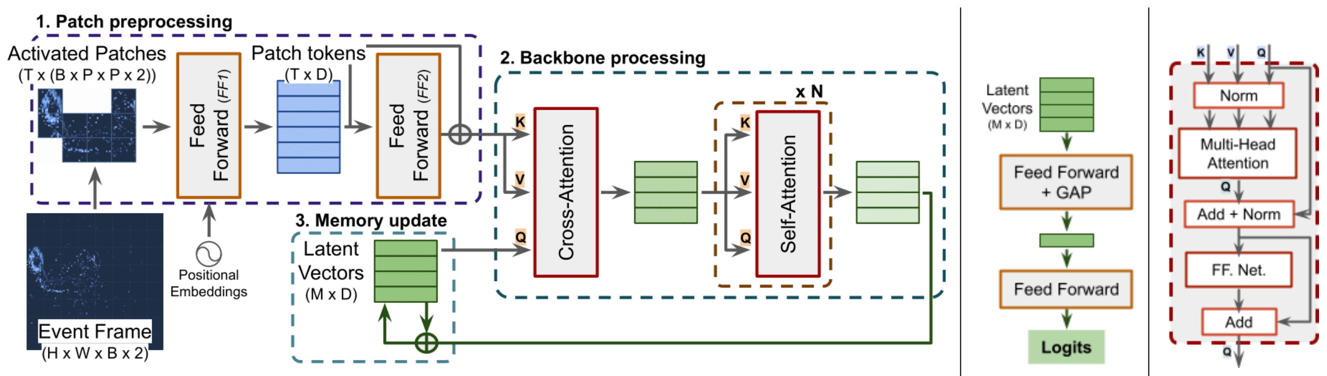
Similar to previous frame-based methods [1, 4, 19], we aggregate the event data $\varepsilon = \{e^1, e^2, e^3, ...\} \mid e^i_t \epsilon \Delta t$ generated during a time-window Δt into a frame-like representation $F^{H\times W\times B\times 2}$. Each location $(x, y) \mid y \epsilon H, x \epsilon W$ in F is represented with two histogram-like vectors of B bins, one for each polarity $p \epsilon \{0, 1\}$. Each histogram discretizes Δt in each bin and counts the number of positive or negative events occurring in the corresponding period $\Delta t/B$. Final representations are transformed as $F' = log(F + 1)$ to smooth extreme high values in highly activated areas.

Frame representations, are then split into non-overlapping patches of size $P \times P$. Then we set each patch as activated if it contains at least m percent of non-zero elements, i.e., if at least a m percent of the pixels (x, y) within the patch have registered events. Activated patches are kept for further processing by EvT, while the non-activated patches are discarded, reducing significantly the following computation cost and implicitly the ambient noise. If the amount of activated patches is below a threshold n, i.e., there is not enough visual information to be processed, we expand the time-window Δt and increase the event set ε with the new incoming events, recompute the frame representation and extract the activated patches. This last step is repeated until we get at least n activated patches. As a final step, we flatten the T activated patches to create tokens of size $(P^2 \times B \times 2)$, input of the transformer backbone detailed in the next section.

This patch-based event representation is therefore designed to take advantage from different event-data properties for a later more efficient event-data processing. Of special relevance, the **spatio-temporal event sparsity** is addressed by dropping non-informative patches, and the **low latency** of this data is addressed by using short time-windows (whose length is adapted to the events density), which are also binarized for a finer representation.

3.2. Event Transformer

Transformers are a natural way to process the patch-based representation we propose. Different to other architectures, they are able to ingest lists of tokens of variable length that are processed with attention mechanisms. The later, different to convolutions, focus on the whole input data (structuring it as a Query (Q), Key (K) and Value (V))



(a) Backbone overview

(b) Classification architecture

(c) Cross and Self-Attention architecture

Figure 2. Event Transformer (EvT) overview. (a) For each new event frame built from an input event-stream, *Activated Patches* are detected and pre-processed to build patch tokens. These tokens are processed by our backbone, and the resulting output is used to update a set of latent vectors. These latent vectors encode all the information received so far. (b) The final version of latent vectors is used to perform the final event-stream classification. (c) Architecture shared by the Cross and Self-Attention modules.

to capture both local and long-rage token dependencies.

The processing core of our work, Event Transformer (EvT), is motivated by these ideas. This backbone processes, with attention mechanisms and a set of M learned latent vectors, lists of patch tokens (whose length varies depending on the event-data sparsity). These latent vectors serve also as a memory that is incrementally refined with the processing of activated patches calculated, as described in Section 3.1, from consecutive time-windows within the event-stream. The final version of the refined latent vectors is processed with a simple classifier to perform the final event-stream classification. The whole process (detailed in Figure 2) is divided in the following steps:

Patch pre-processing. Each one of the T input activated patches is mapped to a vector of dimensionality D. This vector length D is constant along all the network. This transformation (*FF1*) consists of an initial single-layer Feed Forward Network (FF), the concatenation of 2D-aware positional embeddings, and a last single-layer FF Network. The use of positional embeddings to augment the patch information is required since Transformers, unlike CNNs, cannot implicitly know the locality of the input data. Before being processed by the core of our backbone, transformed patches, i.e., patch tokens, undergo another transformation (*FF2*) composed of a double-layer Feed-Forward Network and a skip-connection to achieve a finer representation while preserving the information about its locality.

Backbone processing. The core of our backbone is composed by a single Cross-Attention and N Self-Attention modules that share the same architecture (detailed in Fig. 2c). Similar to previous transformer related works [3, 22, 44], it is composed of a Multi-Head Attention layer [44], normalization layers, skip connections and Feed Forward layers. Our backbone first processes the latent memory vectors Q on the basis of the patch token information K-V (Cross-Attention) and resulting vectors are then refined Q-K-V with no external information (Self-Attention).

Memory update. Once the patch tokens T and latent vectors have been processed by the backbone, the resulting latent vectors in this iteration are combined with the existing memory latent vectors with a simple sum operation. This augmented version of latent vectors encodes a broader spatio-temporal information and will be used to process the activated patches from the following time-window.

Classification output. The final output of our proposed framework is obtained by processing the latest version of the latent memory vectors, that contain the key spatio-temporal information of the event-stream seen so far. In our case, we perform a multi-class classification by simply processing the latent vectors with two Feed Forward Layers and Global Average Pooling (GAP), as detailed in Fig. 2b.

4. Experiments

This section includes the implementation and training details of the proposed Event Transformer (EvT), and the experimental validation. We evaluate EvT in two tasks (long and short event-stream classification for action and gesture recognition), analyze its efficiency, and justify the EvT design choices with a thorough ablation study.

4.1. Implementation and training details

Patch-based event representation. We set a common patch size of 6×6 pixels, but specific values for time-window Δt and number of bins B are specific for each dataset and defined in the following subsections. For all cases, to consider a patch as *activated* we set $m = 7.5$, i.e., we require a 7.5% of the pixels within the patch to log events. Besides, we set $n = 16$, i.e., an event frame must have at least 16 active patches to continue the processing. As previously detailed, if the frame does not have enough ($\geq n$) active patches we increase the time-window covered by the frame and repeat the search of active patches.

Event Transformer. The dimensionality D of the latent vectors and the pre-processed patch tokens is set to 128. The latent memory is composed of 96 latent vectors randomly initialized at the beginning of the training with a Normal distribution of mean 0.0 and deviation 0.2. Similarly, the positional encodings are initialized with 16 bands of 2D Fourier Features [41] (dimensionality of $\frac{H}{P} \times \frac{W}{P} \times 64$, being H and W the specific sensor height and width from each dataset). Both the latent vectors and positional encodings are learned as the rest of the parameters of the network during training. Patch tokens are processed by a single Multi-Head Cross-Attention layer and 2 Multi-Head Self-Attention layers, all of them using 4 attention heads.

Training details. The whole framework is optimized with a Negative Log Likelihood and AdamW [29] optimizer, in a single NVIDIA Tesla V100. The initial learning rate is set as $1e-3$, and we use Stochastic Weight Averaging [20] and gradient clipping. Due to computation resource constraints, we perform the ablation experiments with a batch size of 64 and we reduce the learning rate by a factor of 0.5 after 10 epochs with no loss reduction. Differently, the benchmark results are obtained using a batch size of 128 and a *1cycle learning rate* policy [40] for 240 epochs. As for the data augmentation, we use spatial and temporal random cropping, dropout, drop token, and we repeat each sample within the training batch twice with different augmentations.

4.2. Evaluation

The performance of the proposed Event Transformer (EvT) is evaluated in two scenarios (visualized in the supplementary video) of real event-camera recordings that represent different use cases. First, we evaluate EvT to classify long event-streams, where the analysis and aggregation of the information from different sections (temporal-windows) of the stream is key. Second, we demonstrate that our solution is also suitable to classify short event-streams, where the recorded target has little motion and less video information is available. Note that, since the methods we compare EvT with are not always designed for both long and short event-stream classification, we cannot show their accuracy in all the evaluated datasets. We discuss the EvT efficiency on each classification task with respect to other solutions and we show the ability of EvT to take advantage of the event data sparsity with an efficiency analysis on different kinds of datasets.

4.2.1 Long event-stream classification

EvT is evaluated in two benchmarks for long event-stream classification. The **DVS128 Gesture Dataset** [1] is composed of 1342 event-streams capturing 10 different human gestures (plus an extra category for random movements) and recorded with 29 different subjects under three different illumination conditions. The **SL-Animals-DVS Dataset** [42] is composed of 1121 event-streams capturing 19 different sign language gestures, executed by 58 different subjects, under different illumination conditions. Recordings from these two datasets last 1-6 s. Recordings are split, as detailed in Section 4.3, into time-windows Δt of 24ms for the DVS128 dataset and 48ms for the SL-Animals-DVS. Similar to previous works [4, 6, 19, 47], we build intermediate event representations for each time-window extracted from the event-stream. Then, we process and aggregate them iteratively for the final event-stream classification.

Model	**10 Classes**	**11 Classes**	**Online**
RG-CNN [6]	N/A	97.2	x
3D-CNN + Voting [19]	99.58	99.62	x
CNN [1]	96.49	94.59	✓
Space-time clouds [47]	97.08	95.32	✓
CNN + LSTM [19]	97.5	**97.53**	✓
TORE [4]	N/A	96.2	✓
EvT (Ours)	**98.46**	96.20	✓

Table 1. Classification Accuracy in DVS128 Gesture Dataset (long event-streams). N/A = Not Available at the source reference

Model	**3 Sets**	**4 Sets**
SLAYER [43]	78.03	60.09
STBP [43]	71.45	56.20
DECOLLE [23]	77.6	70.6
TORE [4]	N/A	85.1
EvT (Ours)	**87.45**	**88.12**

Table 2. Classification Accuracy in SL-Animals-DVS (long event-streams). N/A = Not Available at the source reference

Table 1 shows the accuracy of top-performing models in the DVS128 Dataset, with and without including the extra additional distractor of random movements (11 and 10 classes classification respectively). The column *Online* highlights the ability of each model to perform online inference, i.e., incremental processing of the event data and classification with low latency. Similarly, Table 2 shows the accuracy of top-performing methods evaluated on the SL-Animals-DVS Dataset, a more demanding benchmark with lower state-of-the-art accuracy. *3 Sets* results exclude the samples recorded indoor with artificial lighting from a neon light source, since they include noise related to the reflection of clothing and the flickering of the fluorescent lamps. *4 Sets* evaluates all the samples within the dataset.

Results from Table 1 show how our approach obtains comparable results to state-of-the-art. Only [19] is more accurate than EvT but it uses offline inference and CNNs, which are computationally more expensive but have a good inductive bias, useful when training with small datasets like DVS128 and with random movements (as is the case of 11 Classes). As for the most challenging SL-Animals Dataset, EvT achieves a new state-of-the-art. Interestingly, our solution presents higher robustness to different lightning conditions. Note how the performance does not decrease when using the *4 Sets* instead of 3. Instead, it is able to take advantage of larger training set and achieves better accuracy.

Regarding the **model efficiency**, there is not available information to directly compare our approach to the other methods reported in these two datasets. However, the best methods in Tables 1 and 2 use CNN-based models [1,4,19] to process frame-event representations and/or complex aggregation architectures such as Recurrent Layers [19] or CNNs [6] to process intermediate time-window results. Our approach processes event representations with minimal cost compared to other approaches (see Table 4 for a similar comparison in the next dataset), and incrementally aggregates intermediate results on the latent vectors used for the final classification with negligible cost. Given this, EvT is able to perform online inference, while being significantly more resource-efficient than the related methods for long event-stream classification.

4.2.2 Short event-stream classification

This experiment tackles the problem of classifying event-streams of scenes with low motion, typically shorter than those of the previous subsection. This experiment is ran on the **ASL-DVS Dataset** [5]. It contains 100,800 event-streams capturing 24 letter signs from the American Sign Language, performed by 5 different subjects. Each sample lasts about 100 ms. The dataset is randomly split with 80% of the data for training and the remaining 20% for testing, as proposed by the authors. As many other methods that tackle this problem [4, 5, 8, 10], we treat it as a simple instance classification problem where a single event representation (from a single time-window Δt) is built from the whole event-stream.

To perform this short event-stream classification, we represent each event-stream with a single frame representation that encodes 100 ms of the sample ($\Delta t = 100ms, B = 2$). Then, the final classification is performed by processing the generated activated patches with the EvT backbone and computing the logits from the output latent vectors, with no need to perform any memory update. Table 3 shows that EvT is able to get excellent performance.

Model	Accuracy
RG-CNN [5]	90.1
EV-VGCNN [10]	98.3
M-LSTM [8]	99.73
TORE [4]	99.6
EvT (Ours)	**99.93**

Table 3. Classification Accuracy in ASL-DVS (short event-streams)

Although this particular dataset is already solved satisfactorily by prior work, we find that EvT perform with lower computational resources than them. Unfortunately, there are no **model efficiency** statistics published from prior work on real short event-stream benchmarks. However, several works provide them on the simulated dataset **N-Caltech-101** [33], so we use this dataset for the following analysis. This dataset is the event counterpart of the RGB images from Caltech-101 [12]. It contains short event recordings of RGB images displayed on a LCD monitor. Table 4 shows the model complexity (measured in FLOPs and number of model parameters, which is directly related to energy consumption) required by different methods evaluated on the N-Caltech-101 dataset. EvT presents much lower requirements than frame-based methods, that run heavy computations. More importantly, EvT also presents lower computational requirements than point-based methods, that are also designed to tackle the sparsity of the event data, but achieve less computational savings than our approach. Note that since EvT is designed with real-event processing in mind, it does not improve the state-of-the-art accuracy on the N-Caltech-101 dataset. Therefore, this experiment only intends to present a common setup to measure efficiency (not accuracy). Previous Table 3 already showed an accuracy comparison with these short-stream classification approaches when using real event camera data.

Model	Type	(G)FLOPs	#Params.
RG-CNN [5]	Point-based	0.79	N/A
EV-VGCNN [10]	Point-based	0.70	0.84 M
M-LSTM [8]	Frame-based	4.82	21.43 M
EvT (Ours)	**Patch-based**	**0.20**	**0.48 M**

Table 4. Average FLOPs for all validation samples in N-Caltech-101 and number of parameters per model. N/A = Not Available at the source reference

4.2.3 Event sparsity and model efficiency analysis.

In the previous Section, we have compared the computational cost and model size of EvT with prior work in the non-real N-Caltech-101 dataset (Table 4). We now provide a deeper analysis of the computational cost of EvT showing, on four datasets, how it takes advantage of event data sparsity and a compact network design to improve its efficiency.

The computational cost of EvT is determined by the Cross-Attention Layer, which has $O(T \times M)$, where T stands for the amount of activated patches and M for the amount of latent memory vectors. Therefore, the less activated patches there are, the less resources EvT needs. But also, in the worst case, when many activated patches are found ($T \gg M$), the latent vectors prevent EvT from incurring a quadratic cost. Note that the Self-Attention layers have then a reduced cost of $O(M^2)$ instead of $O(T^2)$.

Table 5 shows the average computational cost (time and FLOPs) per time-window Δt of EvT, calculated for the different datasets of our previous experimentation. As observed, short event-stream datasets (N-Caltech-101 and ASL) generate many patches ($T \gg M$) since they are recorded with a bigger sensor (240 × 180 pixels) and, in the case of N-Caltech-101, because of its synthetic nature. Otherwise, short event-stream datasets (Sl-Animals and DVS128) generate less patches ($T < M$) since they use a smaller sensor (128 × 128 pixels) and shorter time-windows. Although the amount of activated patches T of the first group has an order of magnitude more than the second, we do not observe this trend in its computational cost (both in time and FLOPs), that grows smoothly with T.

It is important to remark that, in all cases, EvT processes the activated patches in a significantly shorter time-span than the corresponding time-window Δt, both in GPU and CPU. This highlights the ability of our method to do inference with minimal latency, being capable of processing the event data before the following batch (i.e., time-window data) is generated, therefore facilitating an online inference, even in low resource environments.

Dataset	**Activated Patches T**	**Δt (ms)**	**Time per Δt (GPU/CPU)**	**(G)FLOPs per Δt**
N-Caltech-101	532	100	4 / 16 ms	0.20
ASL	263	100	4 / 9 ms	0.13
SL-Animals	80	48	3 / 5 ms	0.09
DVS128	45	24	2 / 4 ms	0.08

GPU: NVIDIA GeForce RTX 2080 Ti. CPU: Intel Core i7-9700K
Δt: time-window length used to build the initial frame representations

Table 5. EvT efficiency analysis: execution time and FLOPs per Δt. Average results for all validation samples in each dataset.

4.3. Ablation study

This subsection analyzes the influence of key components in our framework, i.e., event data representation and transformer hyperparameters, to justify our design choices. This evaluation has been performed with both the DVS128 Gesture (10 classes) and the SL-Animals-DVS (4 sets) datasets. Note that, as detailed in Section 4.1, due to computation constraints the following experiments are performed with slightly different training conditions than the ones previously reported. Therefore, following results properly compare each component of our framework, but the reported accuracy might be inferior to the benchmark ones.

Event representation hyperparameters. The key components of our event representation are the time-window Δt used to aggregate events into a patch representations, and the number of bins B used to improve their time-resolution. Figure 3a shows the EvT accuracy using different time-window lengths Δt (and bins $B = 2$). As observed, the best Δt value is dependent of the evaluated dataset, requiring longer time-windows for SL-Animals-DVS (48 ms) than for DVS128 (24 ms); these time-window values are now set as default for these datasets. Intuitively, shorter time-windows are required to model motions that are executed at higher speeds, and longer ones fit better slower motions that register less event information. Figure 3b shows that using more bins B (3) is beneficial when using longer time-windows (SL-Animals-DVS) and using less bins B (2) helps with shorter time-windows (DVS128); these bins values are set now as default for these datasets. As observed, the use of more bins, helps us to use finer time-resolutions and therefore improve the final classification accuracy. However, too many bins represent very short time-intervals that do not contain representative event-information.

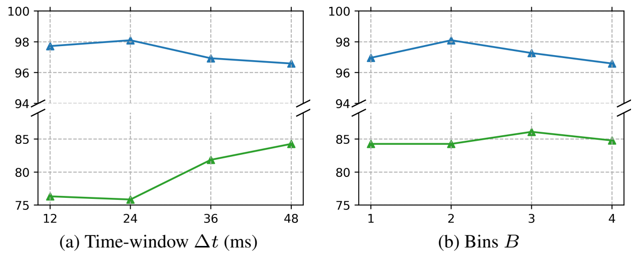


Figure 3. EvT accuracy with different time-windows Δt and bins B for the DVS128 Dataset - 10 classes (top blue line) and for the SL-Animals-Dataset - 4 Sets (bottom green line).

Regarding the generation of active patches from event representations, the patch size P is the most relevant hyperparameter, defining the granularity of the information we are working with. As observed in Figure 4, smaller patch sizes generate more active patches, making EvT to process more information, while bigger patches speed up the EvT inference, but provide less spatial information.

Besides the patch size, the generation of active patches also depends on the minimum amount of pixels m with logged events needed to activate a patch, and the minimum amount of patches n required to start the EvT backbone processing. Table 6 shows how filtering out patches with less event information improves the final accuracy, mainly be-

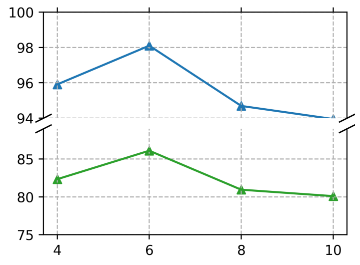


Figure 4. EvT Accuracy with different patch size value for the DVS128 Dataset - 10 classes (top blue line) and for the SL-Animals-Dataset - 4 Sets (bottom green line).

cause it removes the patches that contain noisy event information, not related with the motion in the scene itself. Additionally, setting a minimum amount of activated patches helps to avoid the processing of time-intervals with no motion and therefore with few activated patches.

n \ m	5%	7.5%	10%
8	97.6 84.6	97.9 83.9	96.6 83.3
16	96.9 85.9	98.1 86.1	98.0 84.3
24	97.2 84.3	97.2 82.4	96.9 82.0

Table 6. EvT accuracy with different combinations of m: min. pixels per patch and n: min. patches per Δt. Top cell value: DVS128 (10 classes). Bottom cell value: SL-Animals (4 Sets).

Transformer hyperparameters. The *number of latent vectors* is key for a good generalization. Figure 5a shows that using too few or too many vectors causes under or over-fitting, leading to suboptimal results. The *latent vectors dimensionality* (same dimension than the patch tokens) also affects the results. As observed in Fig. 5b, too short lengths lead to under-fitting results, and too long lengths lead to unstable learning that end with inaccurate results.

As for the attention hyperparameters, Figure 5c shows how the Self-Attention Fig. allow a better processing of the patch tokens up to a limit where the accuracy is no longer improved. Similarly, Fig. 5d shows how more attention heads allow to have a finer processing of the input patch tokens, but it can lead to unstable learning and over-fitting.

Attention scores generated for samples of the used datasets can be found in the supplementary video, and are helpful to understand which data features (patch tokens) lead to classification decisions. A confusion matrix of the most challenging dataset (SL-Animals) is also provided.

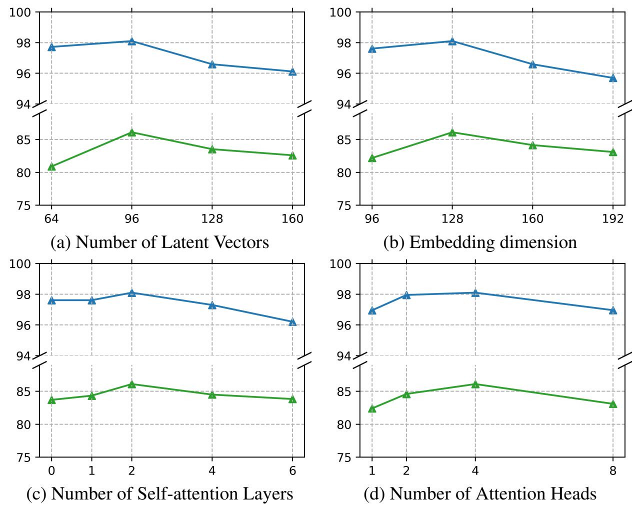


Figure 5. EvT accuracy for different architecture variations using the DVS128 Dataset - 10 classes (top blue line) and the SL-Animals-Dataset - 4 Sets (bottom green line).

5. Conclusions

The present work introduces a novel framework, Event Transformer (EvT), for event data processing. EvT effectively manages to take advantage of event data properties to minimize its computation and resource requirements, while achieving top-performing results. EvT introduces a new sparse patch-based event data representation that only accounts for parts of the streams with sufficient logged information, and an efficient and compact transformer-like architecture that naturally processes it. EvT achieves better or comparable accuracy than the current state-of-the-art on different benchmarks for action and gesture recognition. Most importantly, our solution requires significantly less computational resources than prior works, being able to perform inferences with minimal latency both in CPU and GPU, facilitating its use on low consumption hardware and real-time applications.

Based on the presented results, we believe that patch-based representations and transformers are a promising line of research for efficient event-data processing. This framework also offers promising benefits to other event-based perception tasks, such as body tracking or depth estimation, and to different kinds of sparse data, like LiDAR data.

Acknowledgments

This research has been funded by FEDER/Ministerio de Ciencia, Innovación y Universidades – Agencia Estatal de Investigación project PGC2018-098817-A-I00, DGA T45 17R/FSE and the Office of Naval Research Global project ONRG-NICOP-N62909-19-1-2027.

References

[1] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, et al. A low power, fully event-based gesture recognition system. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1, 2, 3, 5, 6

[2] Anastasios N Angelopoulos, Julien NP Martel, Amit P Kohli, Jorg Conradt, and Gordon Wetzstein. Event-based near-eye gaze tracking beyond 10,000 hz. *IEEE transactions on visualization and computer graphics*, 27(5):2577–2586, 2021. 1

[3] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. *arXiv preprint arXiv:2103.15691*, 2021. 3, 4

[4] R Baldwin, Ruixu Liu, Mohammed Almatrafi, Vijayan Asari, and Keigo Hirakawa. Time-ordered recent event (tore) volumes for event cameras. *arXiv preprint arXiv:2103.06108*, 2021. 1, 2, 3, 5, 6

[5] Yin Bi, Aaron Chadha, Alhabib Abbas, Eirina Bourtsoulatze, and Yiannis Andreopoulos. Graph-based object classification for neuromorphic vision sensing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019. 2, 3, 6

[6] Yin Bi, Aaron Chadha, Alhabib Abbas, Eirina Bourtsoulatze, and Yiannis Andreopoulos. Graph-based spatio-temporal feature learning for neuromorphic vision sensing. *IEEE Transactions on Image Processing*, 2020. 1, 2, 3, 5, 6

[7] Enrico Calabrese, Gemma Taverni, Christopher Awai Easthope, Sophie Skriabine, Federico Corradi, Luca Longinotti, Kynan Eng, and Tobi Delbruck. Dhp19: Dynamic vision sensor 3d human pose dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 0–0, 2019. 1

[8] Marco Cannici, Marco Ciccone, Andrea Romanoni, and Matteo Matteucci. A differentiable recurrent surface for asynchronous event-based data. In *European Conference on Computer Vision*. Springer, 2020. 1, 2, 6

[9] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*. Springer, 2020. 3

[10] Yongjian Deng, Hao Chen, Huiying Chen, and Youfu Li. Evvgcnn: A voxel graph cnn for event-based object classification. *arXiv preprint arXiv:2106.00216*, 2021. 1, 2, 6

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020. 2, 3

[12] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *2004 conference on computer vision and pattern recognition workshop*. IEEE, 2004. 6

[13] Daniel Gehrig, Mathias Gehrig, Javier Hidalgo-Carrió, and Davide Scaramuzza. Video to events: Recycling video datasets for event cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. 3

[14] Daniel Gehrig, Michelle Rüegg, Mathias Gehrig, Javier Hidalgo-Carrió, and Davide Scaramuzza. Combining events and frames using recurrent asynchronous multimodal networks for monocular depth prediction. *IEEE Robotics and Automation Letters*, 6(2):2822–2829, 2021. 1

[15] Rohan Ghosh, Anupam Gupta, Andrei Nakagawa, Alcimar Soares, and Nitish Thakor. Spatiotemporal filtering for event-based action recognition. *arXiv preprint arXiv:1903.07067*, 2019. 2

[16] Yuhuang Hu, Hongjie Liu, Michael Pfeiffer, and Tobi Delbruck. Dvs benchmark datasets for object tracking, action recognition, and object recognition. *Frontiers in neuroscience*, 2016. 3

[17] Yuhuang Hu, Shih-Chii Liu, and Tobi Delbruck. V2e: From video frames to realistic dvs events. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 3

[18] Jian Huang, Jianhua Tao, Bin Liu, Zheng Lian, and Mingyue Niu. Multimodal transformer fusion for continuous emotion recognition. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3507–3511. IEEE, 2020. 3

[19] Simone Undri Innocenti, Federico Becattini, Federico Pernici, and Alberto Del Bimbo. Temporal binary representation for event-based action recognition. In *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021. 1, 2, 3, 5, 6

[20] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*. Association For Uncertainty in Artificial Intelligence (AUAI), 2018. 5

[21] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021. 2, 3

[22] Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver: General perception with iterative attention. *arXiv preprint arXiv:2103.03206*, 2021. 2, 3, 4

[23] Jacques Kaiser, Hesham Mostafa, and Emre Neftci. Synaptic plasticity dynamics for deep continuous local learning (decolle). *Frontiers in Neuroscience*, 2020. 1, 2, 5

[24] Simon Klenk, Jason Chui, Nikolaus Demmel, and Daniel Cremers. Tum-vie: The tum stereo visual-inertial event dataset. *arXiv preprint arXiv:2108.07329*, 2021. 1

[25] Xavier Lagorce, Garrick Orchard, Francesco Galluppi, Bertram E Shi, and Ryad B Benosman. Hots: a hierarchy of event-based time-surfaces for pattern recognition. *IEEE transactions on pattern analysis and machine intelligence*, 2016. 2

[26] Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience*, 2017. 3

[27] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021. 3

[28] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. *arXiv preprint arXiv:2106.13230*, 2021. 3

[29] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5

[30] Elias Mueggler, Christian Forster, Nathan Baumli, Guillermo Gallego, and Davide Scaramuzza. Lifetime estimation of events from dynamic vision sensors. In *2015 IEEE international conference on Robotics and Automation (ICRA)*. IEEE, 2015. 2

[31] Jalees Nehvi, Vladislav Golyanik, Franziska Mueller, Hans-Peter Seidel, Mohamed Elgharib, and Christian Theobalt. Differentiable event stream simulator for non-rigid 3d tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 3

[32] Anh Nguyen, Thanh-Toan Do, Darwin G Caldwell, and Nikos G Tsagarakis. Real-time 6dof pose relocalization for event cameras with stacked spatial lstm networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019. 2

[33] Garrick Orchard, Ajinkya Jayawant, Gregory K Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in neuroscience*, 2015. 3, 6

[34] Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization. In *British Machine Vision Conference (BMVC)*, 2017. 2

[35] Juan Pablo Rodríguez-Gómez, Raul Tapia, Julio L Paneque, Pedro Grau, Augusto Gómez Eguíluz, Jose Ramiro Martínez-de Dios, and Anibal Ollero. The griffin perception dataset: Bridging the gap between flapping-wing flight and robotic perception. *IEEE Robotics and Automation Letters*, 2021. 1

[36] Viktor Rudnev, Vladislav Golyanik, Jiayi Wang, Hans-Peter Seidel, Franziska Mueller, Mohamed Elgharib, and Christian Theobalt. Eventhands: Real-time neural 3d hand pose estimation from an event stream. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12385–12395, 2021. 1

[37] Yusuke Sekikawa, Kosuke Hara, and Hideo Saito. Eventnet: Asynchronous recursive event processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. 2

[38] Teresa Serrano-Gotarredona and Bernabé Linares-Barranco. A 128×128 1.5% contrast sensitivity 0.9% fpn 3 μs latency 4 mw asynchronous frame-free dynamic vision sensor using transimpedance preamplifiers. *IEEE Journal of Solid-State Circuits*, 2013. 3

[39] Sumit Bam Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. In *NeurIPS*, 2018. 1, 2

[40] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*. International Society for Optics and Photonics, 2019. 5

[41] Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*, 2020. 5

[42] Ajay Vasudevan, Pablo Negri, Camila Di Ielsi, Bernabe Linares-Barranco, and Teresa Serrano-Gotarredona. Sl-animals-dvs: event-driven sign language animals dataset. *Pattern Analysis and Applications*, 2021. 3, 5

[43] Ajay Vasudevan, Pablo Negri, Bernabe Linares-Barranco, and Teresa Serrano-Gotarredona. Introduction and analysis of an event-based sign language dataset. In *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)*. IEEE, 2020. 5

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, 2017. 1, 2, 4

[45] Sai Vemprala, Sami Mian, and Ashish Kapoor. Representation learning for event-based visuomotor policies. *ArXiv*, abs/2103.00806, 2021. 2

[46] Antoni Rosinol Vidal, Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Ultimate slam? combining events, images, and imu for robust visual slam in hdr and high-speed scenarios. *IEEE Robotics and Automation Letters*, 2018. 2

[47] Qinyi Wang, Yexin Zhang, Junsong Yuan, and Yilong Lu. Space-time event clouds for gesture recognition: From rgb cameras to event cameras. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019. 1, 2, 5

[48] Ziwei Wang, Liyuan Pan, Yonhon Ng, Zheyu Zhuang, and Robert Mahony. Stereo hybrid event-frame (shef) cameras for 3d perception. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9758–9764. IEEE, 2021. 1

[49] Wenming Weng, Yueyi Zhang, and Zhiwei Xiong. Event-based video reconstruction using transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2563–2572, 2021. 2

[50] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 2018. 2