Event Transformer+. A Multi-Purpose Solution for Efficient Event Data Processing

Alberto Sabater, Luis Montesano, and Ana C. Murillo

Abstract—Event cameras record sparse illumination changes with high temporal resolution and high dynamic range. Thanks to their sparse recording and low consumption, they are increasingly used in applications such as AR/VR and autonomous driving. Current top-performing methods often ignore specific event-data properties, leading to the development of generic but computationally expensive algorithms, while event-aware methods do not perform as well. We propose Event Transformer+, that improves our seminal work EvT with a refined patch-based event representation and a more robust backbone to achieve more accurate results, while still benefiting from event-data sparsity to increase its efficiency. Additionally, we show how our system can work with different data modalities and propose specific output heads, for event-stream classification (i.e. action recognition) and per-pixel predictions (dense depth estimation). Evaluation results show better performance to the state-of-the-art while requiring minimal computation resources, both on GPU and CPU.

Index Terms—Computer vision, image analysis, image classification.

I. INTRODUCTION

Event cameras register changes in intensity at each pixel of the sensor array providing, with minimal power consumption, asynchronous sparse information with an increased High Dynamic Range and a high temporal resolution (in the order of microseconds). Many applications such as AR/VR or autonomous driving can take advantage of this type of cameras, especially when computational power is limited or when dealing with challenging motion and lighting conditions. Although this type of sensors are relatively recent, they have already shown good results in action recognition [4], [15], tracking [6], [28], depth estimation [10], [41] or odometry [18]. The most common way to process event streams converts them into frame representations and use state of the art algorithms based on Convolutional Neural Networks [1], [3], [7], [15] and/or Recurrent Layers [7], [15]. These frame-like representations ignore the natural sparsity of event cameras. There are also methods that try to exploit this sparsity and, consequently, are more efficient, e.g. PointNet-like Neural Networks [40], Graph Neural Networks [4], [8] or Spike Neural Networks [17], [32]. However, they usually obtain lower accuracy.

This paper proposes Event Transformer+, a novel solution (overview in Fig. 1) for efficient event data processing without sacrificing performance. It extends our previous work on Event Transformer [29] in three ways. First, using a finer patch-based event data representation with richer spatio-temporal information, while still benefiting from its sparsity. Second, improving the Event Transformer backbone with a more robust data processing, adapted to jointly use information from different data modalities (e.g., event data and grayscale images). In addition to this, Event Transformer+ can be combined with different output heads to perform either event-stream classification, i.e., action recognition, or per-pixel predictions, i.e. dense estimation.

We evaluate the new EvT+ in real event data benchmarks of two different tasks. First, event-stream classification (i.e. gesture recognition), where it improves prior work performance, including our previous EvT solution. Second, dense per-pixel estimation tasks (i.e. dense depth estimation), including also multi-modal inputs, events and grayscale images. In all cases, our validation demonstrates that EvT+ obtains better results than the state-of-the-art for the different tasks, while performing very efficiently.

II. 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 summary of available event-based datasets.

A. 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 [4], [4], [8], [40] or point-clouds [30], [37] 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 [19] build frames encoding the last generated event for each pixel. SP-LSTM [24] 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 [22] 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 [27], [38] generate frames by aligning events according to the camera ego-motion. [11] binarizes frame representations in the temporal dimension, achieving a better time-resolution. TBR [15] aggregates binarized frame representations into single-bins frames. M-LSTM [7] uses a grid of LSTMs that processes incoming events at each pixel to create a final 2D representation. TORE [3] uses FIFOs to retain the last events for each pixel. EvT [29] build histogram-like representations for each pixel and divide the frame representation into patches, ignoring the ones with not enough event information.

The present work builds patch representations as EvT, but from frames constructed using FIFOs to retain events distributed sparsely on time. The proposed solution benefits from the sparsity of the event data, but also benefits from the robustness of using frame-based representations.

B. Neural Network Architectures for Event Data

Event-stream classification has been addressed in different ways in the literature. First, we find some efficient architectures that process sparse event representations such as Spike Neural Networks [17], [32], [43], PointNet-style Networks [40] or Graph neural Networks [4], [8]. Most common, other rely on CNNs to process event-frame representations [1], [3], [7], [15]. For long event-stream processing, they are split into shorter time-windows that are frequently processed independently, and then aggregated with Recurrent Networks [15], [42], CNNs [1], [15], temporal buffers [3], [8], or voting between the intermediate results [15].

Depending on the aggregation strategy, we consider that an event-processing algorithm is able to perform online inference if it can record 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.

When it comes to event-stream dense estimation, dense event representations and CNNs to process them are the most common scenario. LMDDE [12] uses fully convolutional networks to process the event-data and ConvLSTMs to handle their temporality. ULODE [47] trains a CNN to deblur event representations and predict optical flow, egomotion, and depth. ECN [44] uses an Evenly-Cascaded Convolutional Network to predict optical flow, egomotion and depth. DTL [39] use CNNs to translate events to images for semantic segmentation and depth estimation. RAM-Net [10] uses CNNs to encode both grayscale and event frames and ConvGRUs to update a hidden state with the temporal information, used later to perform multi-modal depth estimation. LMDDE [12] and RAM-Net [10] propose synthetic datasets to be used as pre-training.

Differently, we complement our sparse tokens (already processed by our backbone) with dummy tokens to create a dense representation that is then updated with the information from our latent memory vectors. Then, similar to RGB solutions, [26] we use skip connections between the self-attention blocks in the encoder and the dense output head to generate the final dense output.

C. Event Dataset Recordings

Large-scale public datasets recorded with event cameras in real scenarios are scarce. This lack has motivated many works to propose different approaches to translate RGB datasets to their event-based counterpart. Earlier approaches [4], [13], [20], [25], [31] display RGB data in a LCD monitor and then record the display with an event-camera. More recent approaches introduce the use of learning-based emulators [9], [14], [23] to generate event data. Unfortunately, these translated datasets cannot fully mimic the event-data nature and introduce certain artifacts, specially on their sparsity and latency. 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+ for event-stream classification [1], [34], and multi-modal dense estimation [46].

III. EVENT Transformer FRAMEWORK

Different to traditional RGB cameras, event cameras log the captured visual information in a sparse and asynchronous manner. Each time the event camera detects an intensity change, it triggers an event \( e = \{x, y, t, p\} \) defined by its location \((x, y)\) within the sensor grid \((H \times W)\), its timestamp \(t\) (in the order of \(\mu s\)) and its polarity \(p\) (either positive or negative change). In the following, we detail the \(\text{EvT}^+\) contributions in terms of event-data representation and processing for classification and dense estimation tasks.

A. Patch-Based Event Data Representation

Similarly to previous work [1], [3], [4], [15], [29], [40], we create a frame representation for each time-window \(\Delta t\) that covers a time-span from \(t_i\) to \(t_e\). Like TORE [3], we model this event information with queues \(\text{FIFO}(x, y, p, k)\) (see Fig. 2(a)) that retain \(keK\) events for each pixel \((y\in H, x\in W)\) within the sensor array and polarity \((p\in\{0, 1\})\). But differently, for each pixel we do not retain the last \(K\) events but the last \(K\) events that are separated by at least a minimum time of \(T_m = \frac{\Delta t}{K}\). This threshold is intended to avoid the over-representation of the information provided by the events that happen consecutively in time. And additionally, when no events are registered in this time-window span for a certain pixel, we account for the ones triggered up to a maximum time \(T_M (T_M \gg \Delta t\) and \(T_m \ll t_e)\).

Once the events are queued for a given time-window, we build an intermediate frame representation \(F^{H \times W \times K \times 2}\) with their time-stamps (see Fig. 2(b)). We normalize the pixels to have a value in the range from 0 to \(T_M\) and then to scale their values to a 0 – 1 range (1):

\[
F = F - (t_e - T_M), \quad F = F / T_M
\]

Therefore, the events queued at the end of the time-window will have values close to 1 and the ones close to \(T_M\) will have a value close to 0.
Then, similar to EvT, we split the generated frame-representations into non-overlapping patches of size $P \times P$ (see Fig. 2(c)), and we set each patch as activated if it contains a minimum $m$ percent of pixels that have information of events triggered between $t_1$ and $t_e$. Note that events triggered between $T_{inv}$ and $t_e$ are not involved in the patch activation decision, since they have been considered in previous time-windows, but they complement the patch information to ease later their processing. Activated patches are finally flattened to create tokens $T$ of size $(P^2 \cdot K \cdot 2)$, input of the transformer backbone detailed in the next subsection.

B. Event Transformer

Transformers [36] 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 and process them with attention mechanisms. The later, different to convolutions, focus on the whole input data (structuring it as a Query ($Q$), Key ($K$) and Value ($V$)) to capture both local and long-range token dependencies:

$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{Q K^T}{\sqrt{d_k}} \right) V.$$  \hspace{1cm} (2)

The processing core of our work, Event Transformer+ (EvT+), is motivated by these ideas. As the patches $P$ (whose length varies on the event sparsity, as described in Section III-A) from new time-windows are being generated, they are processed by this backbone with attention mechanisms. This process ends with the update of a set of $M$ latent vectors. These vectors act as a memory that logs the key information seen so far and its final processing allows to perform tasks such as event-stream classification or dense per-pixel estimations. This whole process (detailed in Fig. 3) is divided in the following steps:

1) Patch pre-processing: In this step, the set patch tokens related to single time-windows are processed by analyzing their spatial affinity. For this purpose, each one of the $T$ input activated patches is mapped to a vector of dimensionality $D$, constant along all the network. This transformation ($FF1$) consists of an initial Feed Forward layer (FF), the concatenation of 2D-aware positional embeddings, and a last FF layer. 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. An initial set of $N_1$ Self-Attention blocks is then used to analyze long and short-range spatial dependencies between tokens. In the case of multi-modal data processing, a different patch pre-processing branch is used for each data modality (as summarized in Fig. 1(b)).

2) Backbone processing: In this step, the incoming spatial information from the pre-processed patch tokens is fused with the one from the latent memory vectors. This is done with a single Cross-Attention Module that processes the latent memory vectors $M'$ (as $Q$) with token information (as $K-V$). The resulting $M'$ vectors are then refined with $N_2$ Self-Attention layers.

3) Memory update: The latent memory vectors $M$ are updated given the new generated vectors $M'$ with a simple sum operation and normalization:

$$M = ||M + M'||$$  \hspace{1cm} (3)

This augmented version of the latent vectors encodes longer spatio-temporal information and is used to perform the final downstream task, for which we have implemented the following two options.

4.a) Classification head: Event-stream classification is performed by processing the refined latent memory vectors, that contain the key spatio-temporal information of the event-stream seen so far. This processing consists of $N_3$ Self-Attention modules and then, similar to EvT, processing the resulting vectors with two Feed Forward layers and Global Average Pooling (GAP).

4.b) Dense estimation head: Given the sparse set of tokens processed at step 1 and their initial location in the frame representation, we convert them back to a dense representation by adding dummy tokens (initialized with zeroes) that take the place of the patches filtered out due to their event-data sparsity. We then add positional information to each patch of this dense representation and update it with the information contained in the latent memory vectors (used as $K-V$) with a Cross-Attention layer. The resulting set of tokens is then refined with $N_1$ Self-Attention layers, with skip-connections from the $N_1$ Self-Attention layers of the patch pre-processing step. In this process, dense information at non-activated patches is inferred by jointly processing their positional information, their surrounding activated and non-activated (dummy) tokens, and the latent memory vectors, which encode the information processed in previous time-windows.

In the case of multi-modal data processing, the skip connections propagate the information jointly for each data modality, whose tokens are merged with a simple addition and normalization operation.

Attention Modules. All the Cross and Self-Attention modules from EvT+ share the same architecture, similar to previous transformer related works [2, 16, 29, 36], composed of a Multi-Head Attention layer [36], normalization layers, skip connections and Feed Forward layers.

C. Optimization

EvT+ is optimized differently for different downstream tasks. In the case of event-stream classification, EvT+ is optimized with the
for the depth issue to 2. All takes advantage from the and using gradient clipping. The batch-size overview. The input is a set of time-window representations (e.g., event frames or images) that are processed sequentially. Each takes advantage from the and the groundtruth labels in two tasks (event-stream classification and monocular dense estimation), analyze its efficiency, and main design choices.

Negative Log-Likelihood loss:

\[ \mathcal{L}_{NLL} = - \frac{1}{n} \sum_i \left( y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i) \right) \]

between the predicted labels \( \hat{y}_i \) and the groundtruth labels \( y_i \) and label smoothing [45] for regularization.

In the case of monocular dense estimation, we train EvT+ in the sparse depth labels measured by a LiDAR sensor. These ground-truth depth maps, similar to other methods [10, 12], are clipped to a range \([D_m - D_M]\) captured by the sensor \((2 - 80)\) and \([224 - 1881]\) in our case of the MVSEC and EV-IMO2 datasets) and we train EvT+ to predict its normalized log depth representation \( \bar{Y} \[0-1] \):

\[ \bar{Y} = \frac{\log(Y) - \log(D_M)}{\log(D_m) - \log(D_M)} \]

We optimize EvT+ as in [10] with a scale-invariant loss

\[ \mathcal{L}_{si} = \frac{1}{n} \sum_i (R_i)^2 - \frac{1}{n^2} \left( \sum_k R_k \right)^2, \]

and a multi-scale invariant loss

\[ \mathcal{L}_{msi} = \frac{1}{n} \sum_k \sum_i \left( \| \nabla_x R_{ik}^2 \| + \| \nabla_y R_{ik}^2 \| \right), \]

where \( n \) are the valid depth ground-truth points, \( R_i \) is the log-depth difference map \( \| \bar{Y}_i - \bar{Y}_j \| \) at the point \( i \), \( R_{ik} \) is the log-depth difference map at the scale \( k e[0-4] \). Both losses are combined as \( \mathcal{L} = \mathcal{L}_{si} + \lambda \mathcal{L}_{msi} \), with \( \lambda = 0.25 \).

IV. EXPERIMENTS

This section includes the implementation and training details of the proposed Event Transformer+ (Event+) along with its experimental validation. We evaluate Event+ in two tasks (event-stream classification and monocular dense estimation), analyze its efficiency, and main design choices.

A. Implementation and Training Details

A. Patch-based event representation: We set a patch size of \( 10 \times 10 \) for event-stream classification and \( 12 \times 12 \) for the depth estimation task, that has larger frame representations. In all cases the number of events in the FIFO, \( K \), is set to 3, \( M_F \) is set to 256 ms, and the threshold \( m \) for the patch activation is set as in EvT (7.5%). Dataset-specific hyperparameters are discussed in the following Section IV-B.

A. Event Transformer: The latent vectors and the vector dimensionality \( D \) is set to 160. The latent memory is composed of 32 latent vectors. The positional encodings are initialized with 6 bands of [33], 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. The amount of Attention layers \( N_1, N_2 \) and \( N_3 \) are set to 1, but in the case of depth estimation, \( N_1 \) is set to 2. All the Multi-Head Attention layers use 8 heads, but in the case of depth estimation that has bigger input size, we use only 4 in the pre-processing and decoding steps to increase their efficiency.

A. Training details: The whole framework is optimized with the AdamW optimizer [21] in a single NVIDIA Tesla V100, with the learning rate set to \( 1e-3 \) and using gradient clipping. The batch-size is 128 for event-stream classification and 24 for depth estimation. Data augmentation used consists of spatial and temporal random cropping, dropout, drop token, and repetition of each sample within the training batch twice with different augmentations.

B. Evaluation

The proposed Event Transformer+ (Event+) is evaluated in two scenarios of real event-camera recordings that represent different use cases. First, we evaluate EvT+ to classify event-streams of human actions and gestures. Second, we demonstrate that our solution is also suitable for dense inference from a sparse input, in particular, using multi-modal (grayscale image and event data) dense depth estimation. Finally, we provide a detailed analysis on how EvT+ takes advantage from the event-data sparsity to increase its efficiency, performing inferences with minimal latency.
TABLE I
CLASSIFICATION ACCURACY IN DVS128 GESTURE DATASET

| Model              | 10 Classes | 11 Classes | Online |
|--------------------|------------|------------|--------|
| RG-CNN [4]         | N/A        | 97.2       | x      |
| 3DCNN + Voting [15]| 99.58      | 99.62      | x      |
| CNN [1]            | 97.19      | 94.39      | ✓      |
| Space-time clouds [40]| 97.08    | 95.32      | ✓      |
| CNN + LSTM [15]    | 97.3       | 97.33      | ✓      |
| TORE [3]           | N/A        | 96.2       | ✓      |
| EvT                | 98.46      | 96.20      | ✓      |
| EvT↑ (Ours)        | 99.24      | 97.57      | ✓      |

N/A = not available at the source reference.

TABLE II
CLASSIFICATION ACCURACY IN SL-ANIMALS-DVS

| Model    | 3 Sets | 4 Sets |
|----------|--------|--------|
| SLAYER [35]| 78.03  | 60.09  |
| STBP [35] | 71.45  | 56.20  |
| DECOLLE [17]| 77.6   | 70.6   |
| TORE [3]  | N/A    | 85.1   |
| EvT       | 87.45  | 88.12  |
| EvT↑ (Ours)| 92.34  | 94.39  |

N/A = not available at the source reference.

1) Event-Stream Classification: EvT↑ is evaluated in two benchmarks for event-stream classification. The DVS128 Gesture Dataset [1] is composed of 1342 event-streams capturing 10 different human gestures (plus an optional extra category for random movements) and recorded with 29 different subjects under three different illumination conditions. The SL-Animals-DVS Dataset [34] 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 between 1 and 6 seconds and contain continuous repetitions of shorter human gestures. As detailed in the supplementary material, available online, these recordings are cropped to 120 to 1792 ms and split into time-windows of 24 and 48 ms for the DVS128 and SL-Animals-DVS datasets respectively.

Table I shows the accuracy of top-performing models in the DVS128 Dataset, with and without including the extra additional distractor class of random movements (11 and 10 classes 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 II 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 I show how our approach obtains better results than prior works, improving EvT in both data set-ups. Only [15] is more accurate than EvT↑ but it uses offline inference and 3D-CNNs, which are computationally expensive but have a good inductive bias, useful when training with small datasets like DVS128 and with random movements (as it is the case of 11 Classes). As for the more challenging SL-Animals Dataset, EvT↑ achieves a new state-of-the-art, outperforming prior methods by a large margin. Interestingly, our solution presents higher robustness to the different lighting conditions of the 4 Sets, being able to take advantage of larger training set to achieve better accuracy.

2) Monocular Multi-Modal Dense Depth Estimation: To evaluate EvT↑ for dense depth estimation we use the MVSEC Dataset [46]. This dataset includes stereo automobilistic recordings with event-data, grayscale images and depth maps captured by a LiDAR, captured by day (2 recordings) and at night (3 recordings). Similar to previous work, we use the outdoor_day_2 sequence for training and the remaining 4 sequences for testing. Due to the corruption of different sequences, we limit the training and validation to the data recorded from the left sensor. Depth maps are always recorded at 20 Hz, but grayscale images are recorded at 45 Hz by day and 10 Hz by night, therefore, they are not synced with the depth maps.

Since the recorded sequences are very long (262 to 653 seconds), due to computational restrictions during training, we only consider the previous 512 ms (as detailed in the supplementary material, available online) of information before the timestamp of a depth map for its inference. The event information from these sequences is split in time-windows of 50 ms that are synced with the depth maps, and is complemented with a grayscale image generated at least 15t/2 ms away from the end of each time-window. Therefore, all time-steps contain event-data but might not contain grayscale image information. This issue is more frequent in the night sequences, where the grayscale image frequency is lower. When there is no grayscale information, only the event tokens update the memory and are used to the later dense depth estimation. Table III shows the average depth error of different models at different cut-off depths, i.e., pixels whose groundtruth depth information is under the specified threshold (10, 20, or 30 meters). As observed, EvT↑ is able to largely outperform previous methods in most of the set-ups, with no specific pre-training. More importantly, when including image data EvT↑ improves its accuracy to achieve higher robustness even in the most challenging scenarios.

Additionally, we have evaluated EvT↑ in the newly introduced EV-IMO2 dataset [5], which presents a different use case of indoor depth estimation and a larger sensor size. Although there is no prior work tested in this dataset, EvT↑ is able to perform this indoor depth estimation with a mean absolute error of 176, 121, 176, and 181 mm for the different imo, imo_ll, sfm, and sfm ll dataset splits respectively. For this experiment, we have reduced the sensor size by half by filtering out one out of two consecutive pixels from the sensor array.

3) Event Sparsity and Model Efficiency Analysis: We now provide a deeper analysis of the theoretical computational cost of EvT↑ and how it benefits from the data sparsity. In the case of event-stream classification, different from EvT where the computational cost (O(|T| × M)) depends on the cross-attention layer, the computational cost of EvT↑ depends on the initial self-attention pre-processing (O(|T|^2)), where |T| stands for the amount of activated patches and M for the amount of latent memory vectors. This means that similar to EvT, the cost lowers as the input data is more sparse (less activated tokens |T| are generated), but it is not bounded by M. Although this cost is theoretically higher, as observed in Table IV, in practice different implementation improvements such as bigger patch sizes P and less latent vectors M, make EvT↑ even more efficient (FLOPs and latency) than EvT for event-stream classification.

In the case of depth estimation, the existence of a dense output head that does not work with sparse information increases the computational cost to O(M^2W^2 + P^2). In the following, we use |P| to refer to this cost, defined as the number of the total generated patches. This higher cost and the use of a bigger sensor size that generates more activated patches, make EvT↑ to demand more computational resources than for event-stream classification. Still, the rest of the Neural Network benefits from the data sparsity by suppressing non-activated patches (details in Table V). This is also true for the ones generated
TABLE III
EVALUATION ON THE MVSEC DATASET

| Model     | Events | Images  | outdoor day 1 | outdoor night 1 | outdoor night 2 | outdoor night 3 |
|-----------|--------|---------|---------------|-----------------|-----------------|-----------------|
| ULODE [47]| ✓      | X       | 2.72          | 3.84            | 4.40            | 3.13            |
| LMODE [12]| ✓      | X       | 2.70          | 3.46            | 3.84            | 3.56            |
| LMODE [12]*| ✓      | ✓       | 1.85          | 2.64            | 3.13            | 3.38            |
| EvT+ (Ours) | ✓  | ✓       | 1.31          | 1.92            | 2.32            | 1.54            |
| RAM Net [10]** | ✓   | ✓      | 1.39          | 2.17            | 2.76            | 2.50            |
| EvT+ (Ours) | ✓  | ✓       | 1.24          | 1.91            | 2.36            | 1.45            |

* Pre-training on DENSE [12] dataset  ** Pre-training on EventScape [10] dataset

Average absolute depth error in meters (lower is better) at different cut-off depth distances in meters (10, 20, 30). First block shows models trained just on event data. Second block shows models trained jointly with event and image (grayscale) data.

TABLE IV
EVT+ EFFICIENCY ANALYSIS: EXECUTION TIME AND FLOPS PER Δt

| Model     | Sensor size | Dataset     | P   | [T]/|P| | Δt ms | Latency (GPU/CPU) | FLOPs ([T]/|P|) | #Params |
|-----------|-------------|-------------|-----|-----|----|--------|------------------|----------------|--------|
| EvT       | 128 x 128   | SL-Animals  | 8   | 80  | 256| 48     | 3 / 5 ms         | 0.09 / - G    | 0.48 M  |
| EvT       | 128 x 128   | DVS128      | 8   | 45  | 296| 24     | 2 / 4 ms         | 0.08 / - G    | 0.48 M  |
| EvT+      | 128 x 128   | SL-Animals  | 10  | 52  | 169| 48     | 2 / 4 ms         | 0.04 / 0.08 G (x2) | 0.66 M |
| EvT+      | 128 x 128   | DVS128      | 10  | 18  | 169| 24     | 3 / 3 ms         | 0.03 / 0.07 G (x2.3) | 0.66 M |
| EvT**     | 348 x 260   | MVSEC       | 12  | 318 | 638| 50     | 10 / 25 ms       | 2.94 / 3.94 G (x1.35) | 1.98 M |
| EvT**     | 346 x 260 x 2 | MVSEC    | 12  | 318 + 319 | 638 + 638| 50 | 15 / 37 ms       | 3.68 / 4.98 G (x1.35) | 2.50 M |
| EvT+      | 220 x 240   | EV-MO2      | 12  | 237 | 540| 50     | 10 / 42 ms       | 2.27 / 3.11 G (x1.57) | 1.99 M |

* uses multi-modal processing (events + grayscale images)  
Δt: time-window length  
T: amount of activated patches  
P: total amount of generated patches (no filtering)

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

TABLE V
AVERAGE FLOPS REQUIRED TO PROCESS A SINGLE TIME-WINDOW Δt AND GENERATE A DENSE OUTPUT FOR DEPTH ESTIMATION

| Sparse token pre-processing | 0.61 G |
| Backbone processing and Latent Vectors update | 0.06 G |
| Dense output head | 2.27 G |

EvT+ Step  FLOPs

(a) DVS128 Dataset - 10 classes (blue line), SL-Animals-Dataset - 4 Sets (green line).
(b) MVSEC (multi-modal processing), outdoor_day_1 (blue line) and outdoor_night_2 (green line).

Fig. 4. Avg. number of activated patches (vertical axis) generated at each time window on different datasets with different patch sizes (horizontal axis). Stars: the selected hyperparameter value.

from grayscale images, when pixels are black, especially in the night sequences.

However, since our network is very shallow, in all cases the final latency of EvT+ is minimal, being able to perform inference in a time-span significantly shorter than the time-window Δt processed, both in GPU and CPU.

As observed in Table IV, the practical computational cost (FLOPs) depends mainly on the number of tokens generated, which depends on the sensor and patch sizes, but also on the event data sparsity. It is interesting to notice that when not filtering the non-activated patches, i.e. using the whole set of generated patches |P|, the FLOPs that EvT+ demands to process a single time-window scale up between 1.35 and 2.3 times. Fig. 4 also shows, for different datasets, how the demanded FLOPs depend on the event data sparsity and the patch size P. In particular, smaller patch sizes, bigger recording sensors (i.e. MVSEC dataset), and denser event information generate more activated patches, increasing the computational cost.

The supplementary material, available online, extends this efficiency analysis including efficiency statistics and comparisons with standard visual backbones (ResNets), and our seminal EvT work [29] examines the benefits of our proposed representation against other non-frame event representations and non-transformer processing models.

C. Framework Design Study

The main hyperparameters that are key for achieving high efficiency while maintaining a high performance are the patch size P and the number of self-attention layers N1.

The patch size P influences the number of activated patches that are generated. Larger patch sizes generate fewer activated patches, benefiting efficiency. In our experiments, we set the patch size P as 10 for the classification task and a value of 12 for the dense estimation task, whose datasets present a larger recording sensor and more dense
event information. These values allow to achieve high performance while being highly efficient.

The number of self-attention layers \(N_1\) determine the quadratic computational cost of EvT\(^+\), so a large amount will significantly decrease the framework efficiency. In our experiments, we set the number of MHSA layers to 1 in the case of event-stream classification and a number of 2 MHSA layers in the case of dense estimation, which requires more detailed event data processing.

A thorough analysis of the key EvT\(^+\) hyperparameters can be found in the supplementary material, available online.

V. Conclusion

This work presents the Event Transformer\(^+\) (EvT\(^+\)) framework for event data processing, improving the seminal version of EvT. The proposed refined patch-based event representation and backbone compared to EvT are shown to provide more accurate results and increase further its efficiency for event-stream classification. This framework is demonstrated with a more complete validation, including the usage of data from different modalities and dense per pixel estimation tasks (in particular dense depth estimation) in addition to event-stream classification tasks. Evaluation results show better or comparable accuracy to the state-of-the-art while requiring minimal computational resources, which makes EvT\(^+\) able to work with minimal latency both on GPU and CPU. This work shows how patch-based representations and transformers are a promising line of research for efficient event-data processing and opens opportunities for further contributions with different kinds of sparse data such as LiDAR data.

References

[1] A. Amir et al., “A low power, fully event-based gesture recognition system,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 7388–7397.
[2] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Luˇc, and C. Schmid, “ViVT: A video vision transformer,” 2021, arXiv:2103.15691.
[3] R. Baldwin and R. Liu, M. Almattri, V. Asari, and K. Hirakawa, “Time-ordered recent event (TORE) volumes for event cameras,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 45, no. 2, pp. 2519–2532, Feb. 2023.
[4] Y. Bi, A. Chadha, A. Abbas, E. Bouroussalatte, and Y. Andreopoulos, “Graph-based spatio-temporal feature learning for neuromorphic vision sensing,” IEEE Trans. Image Process., vol. 29, pp. 9084–9098, 2020.
[5] L. Burner, A. Mitrokhin, C. Fermüller, and Y. Aloimonos, “EVIMOS: An event camera dataset for motion segmentation, optical flow, structure from motion, and visual inertial odometry in indoor scenes with monocular or stereo algorithms,” 2022, arXiv:2205.03467.
[6] E. Calabrese et al., “DHP19: Dynamic vision sensor 3D human pose dataset,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops, 2019, pp. 1695–1704.
[7] M. Cannici, M. Ciccone, A. Romanoni, and M. Matteucci, “A differentiable recurrent surface for asynchronous event-based data,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 136–152.
[8] Y. Deng, H. Chen, H. Chen, and Y. Li, “EV-VGNN: A voxel graph CNN for event-based object classification,” 2021, arXiv:2106.00216.
[9] D. Gehrig, M. Gehrig, J. Hidalgo-Carrió, and D. Scaramuzza, “Video to events: Recycling video datasets for event cameras,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 3583–3592.
[10] D. Gehrig, M. Ruegg, M. Gehrig, J. Hidalgo-Carrió, and D. Scaramuzza, “Combining events and frames using recurrent asynchronous multimodal networks for monocular depth prediction,” IEEE Robot. Automat. Lett., vol. 6, no. 2, pp. 2822–2829, Apr. 2021.
[11] R. Ghosh, A. Gupta, A. Nakagawa, A. Soares, and N. Thakor, “Spatiotemporal filtering for event-based action recognition,” 2019, arXiv:1903.07067.
[12] J. Hidalgo-Carrió, D. Gehrig, and D. Scaramuzza, “Learning monocular dense depth from events,” in Proc. Int. Conf. 3D Vis., 2020, pp. 534–542.
[13] Y. Hu, H. Liu, M. Pfeffer, and T. Delbruck, “DVS benchmark datasets for object tracking, action recognition, and object recognition,” Front. Neurosci., vol. 10, 2016, Art. no. 405.
[39] L. Wang et al., “Dual transfer learning for event-based end-task prediction via pluggable event to image translation,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2021, pp. 2115–2125.

[40] Q. Wang et al., “Space-time event clouds for gesture recognition: From RGB-camera to event cameras,” in Proc. IEEE Winter Conf. Appl. Comput. Vis., 2019, pp. 1826–1835.

[41] Z. Wang, L. Pan, Y. Ng, Z. Zhuang, and R. Mahony, “Stereo hybrid event-frame (SHEF) cameras for 3D perception,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2021, pp. 9758–9764.

[42] W. Weng, Y. Zhang, and Z. Xiong, “Event-based video reconstruction using transformer,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2021, pp. 2543–2552.

[43] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, “Spatio-temporal backpropagation for training high-performance spiking neural networks,” Front. Neurosci., vol. 12, 2018, Art. no. 331.

[44] C. Ye, A. Mitrokhin, C. Fernmüller, J. A. Yorke, and Y. Aloimonos, “Unsupervised learning of dense optical flow, depth and egomotion with event-based sensors,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2020, pp. 5831–5838.

[45] C.-B. Zhang et al., “Delving deep into label smoothing,” IEEE Trans. Image Process., vol. 40, pp. 5984–5996, 2021.

[46] A. Z. Zhu, D. Thakur, T. Özaslan, B. Pfommer, V. Kumar, and K. Daniilidis, “The multivehicle stereo event camera dataset: An event camera dataset for 3D perception,” IEEE Robot. Automat. Lett., vol. 3, no. 3, pp. 2032–2039, Jul. 2018.

[47] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, “Unsupervised event-based learning of optical flow, depth, and egomotion,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 1694–1694.