Making Every Event Count:
Balancing Data Efficiency and Accuracy in Event Camera Subsampling

Hesam Araghi   Jan van Gemert   Nergis Tomen
Computer Vision Lab, Delft University of Technology
{h.araghi, j.c.vangemert, n.tomen}@tudelft.nl

Abstract

Event cameras offer high temporal resolution and power efficiency, making them well-suited for edge AI applications. However, their high event rates present challenges for data transmission and processing. Subsampling methods provide a practical solution, but their effect on downstream visual tasks remains underexplored. In this work, we systematically evaluate six hardware-friendly subsampling methods using convolutional neural networks for event video classification on various benchmark datasets. We hypothesize that events from high-density regions carry more task-relevant information and are therefore better suited for subsampling. To test this, we introduce a simple causal density-based subsampling method, demonstrating improved classification accuracy in sparse regimes. Our analysis further highlights key factors affecting subsampling performance, including sensitivity to hyperparameters and failure cases in scenarios with large event count variance. These findings provide insights for utilization of hardware-efficient subsampling strategies that balance data efficiency and task accuracy. The code for this paper will be released at: https://github.com/hesamaraghi/event-camera-subsampling-methods.

1. Introduction

Event cameras are visual sensors equipped with pixel arrays capable of capturing changes in brightness intensity with a time resolution in the order of microseconds. Their power efficiency and high temporal resolution make them ideal candidates for edge AI applications [14, 23]. However, due to their temporal resolution, event cameras can generate exorbitant events per unit time, placing a heavy load on both the transmission and processing systems. This high event rate is typically associated with increased power consumption [7], potentially compromising the camera's power efficiency.

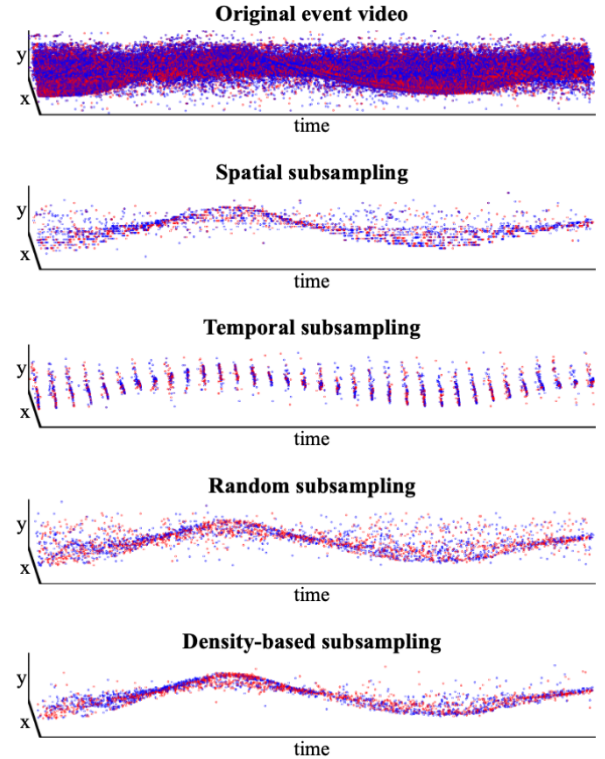A simple and effective way to reduce the number of

Figure 1. Spatial, temporal, random, and density-based subsampling applied to an example video from the DVS-Gesture [2] dataset, along with the original video. Positive and negative polarity events (red, blue) are plotted in 3 dimensions spanned by pixel coordinates (x, y) and time. Although all subsampled outputs contain the same number of events (3000), their structures vary due to the distinct characteristics of each subsampling method. We evaluate their impact on downstream tasks.

events is through subsampling. In [3], the authors have shown that even at higher levels of subsampling, the accuracy of event video classification can remain high. Nevertheless, there are numerous subsampling methods to consider, some of which are shown in Figure 1. In addition,

at the hardware level, event cameras may adopt different strategies to manage the event rate. For example, the event camera introduced in [13] limits the event rate to a predefined threshold by subsampling events both spatially and temporally. Similarly, the event rate controller method in [9] discards events for a specific time period if the event rate becomes too high. However, the downstream effects of these subsampling techniques on visual task performance have not been well-studied. Choosing the subsampling strategy for a given task remains a challenge, particularly when aiming to minimize subsequent accuracy loss.

This study investigates the effect of different subsampling methods on a downstream task. Specifically, we consider object classification as an exemplary downstream task, which plays a key role in many edge AI applications [4, 6, 15, 27, 44]. We focus on hardware-friendly subsampling methods which use simple computing operations without demanding high processing power. In addition, a good subsampling method should reduce the number of events while preserving as much task-relevant information as possible, thereby minimizing accuracy degradation. We hypothesize that most relevant information is contained in regions with a high density of events in spatiotemporal space. These regions likely correspond to motion in the scene, which typically carries substantial task-relevant information, while isolated events are often triggered by noise. To test this hypothesis, we propose a density-based subsampling method. We adopt a causal approach which does not depend on future events, preserving the low latency and requiring no memory-intense event buffering. The key contributions of this work can be summarized as follows:

- We study the accuracy-#events trade-off for six different subsampling methods on the downstream task of event video classification using convolutional neural networks. Our evaluation is conducted on three benchmark datasets: N-Caltech101, DVS-Gesture, and N-Cars.
- We test the hypothesis that events in high-density regions carry more informative content by proposing a simple causal density-based subsampling method.
- We conduct an in-depth analysis of the factors contributing to performance limitations of subsampling methods. Our findings indicate that sensitivity to hyperparameters, such as phase offset, can significantly degrade performance in naive spatial subsampling.

2. Related work

2.1. Event Rate Reduction

Reducing the event rate can be achieved through different approaches, including downsampling events by reducing the spatial and/or temporal resolution or filtering out the events. In [8], the authors propose a downsampling method that reduces both temporal and spatial resolution.

However, their evaluation focuses on classification using a spiking neural network (SNN), which lags behind state-of-the-art methods. This potentially makes their findings less generalizable to other architectures, such as CNNs or transformers. In [21], the authors use random event subsampling as a data augmentation technique, but they do not study it as a method for subsampling event input. The work in [3] investigates the impact of random subsampling on classification accuracy and the challenges of training CNNs in extremely sparse event regimes. In [19], authors propose a specific spatial subsampling by normalizing event counts within spatially downscaled regions. An event is then triggered when the normalized count exceeds a predefined threshold. Other spatial subsampling methods involve using feedforward SNNs to generate downsampled events [17, 20]. However, practical use of these methods would require specialized hardware for SNNs in the camera.

Beyond naive spatial, temporal, and random subsampling, more complex methods take into account interactions between events when deciding which ones to keep. Several studies focus on denoising background activity (BA) by filtering out low-density events [12, 22, 26, 33]. However, the goal is noise removal rather than event subsampling, and the effect of density-based methods on the subsampling is unexplored.

Another relevant line of research involves event-based corner detection methods [1, 18, 32, 38, 45], where most approaches adapt the Harris corner detector [24] for event streams. However, these works primarily focus on corner detection rather than using corners for event subsampling. Given the informativeness of corners in standard images [34, 35], they could also be leveraged for efficient subsampling of event data. Despite the widespread use of subsampling methods, such as spatial and temporal subsampling, a systematic comparison of the accuracy performance is lacking. Thus, we introduce an evaluation study on assessing the effect of different subsampling methods—including density-based and corner-based approaches—across multiple datasets.

2.2. Event Processing

Deep neural networks have significantly advanced event data processing across various applications [47]. Among them, convolutional neural networks (CNNs) are widely used for event cameras [14–16, 47]. CNNs are particularly attractive due to their computational efficiency compared to more complex models like vision transformers (ViTs) [29, 41, 49] and their improved accuracy over SNNs [8, 10, 46] or handcrafted approaches [31, 40, 43, 44]. Since CNNs require grid-like inputs, event data in address-event representation [36] must first be transformed. Various representations exist to do this. The time surface representation [31] assigns the most recent event timestamp to each pixel,

keeping the temporal information of last events. Another type of representation is generating frames using counting the events [30, 37], which aggregates events at each pixel. This representation is simple and it can keep the spatial information such as edges of the scene, but may discard fine-grained temporal details, potentially causing blurring. Another approach is voxel grid representation [48], which divides the time axis into bins and applies predefined kernels to compute bin values. A more advanced method, Event Spike Tensor (EST) [15], extends this idea by learning a kernel for aggregating the surrounding events into bin values. The end-to-end learning property of EST allows the model the flexibility of extracting more relevant information from the input event data. In this paper, we adopt the EST algorithm [15] to compare the classification accuracy between the different subsampling methods.

3. Method

We represent an event video as a set of events, $\mathcal{E} = \{e_i\}_{i=1}^{N}$, where N is the total number of events in the video. Each event e_i consists of four values: $e_i = (x_i, y_i, t_i, p_i)$, where $x_i \in \{1, \ldots, W\}$ and $y_i \in \{1, \ldots, H\}$ represent the horizontal and vertical spatial positions with H and W the height and width in pixels, $t_i \in \mathbb{R}$ is the timestamp, and $p_i \in \{-1, 1\}$ denotes the polarity of the i-th event. We assert that the time stamps are ordered $t_i \leq t_j$ for $i \leq j$.

3.1. Event Representation and Training Procedure

For object classification in event videos, convolutional neural networks (CNNs) have been well established [4, 6, 15, 44] Here, we use CNNs as a proxy for evaluating the information content retained by subsampling methods in the downstream classification task. We use the EST algorithm [15] to represent events in a voxel grid format, making them compatible with convolutional neural networks (CNNs) for classification. The voxel grid representation in EST divides the temporal dimension of an event stream into B equally-spaced bins. For each bin, events are accumulated into two 2D grid frames—one for each event polarity $p_i \in \{-1, 1\}$. As a result, the final event representation is a voxel grid of size $V \in \mathbb{R}^{2B \times H \times W}$. The mapping of the events to each voxel in the EST representation is computed by a multilayer perceptron (MLP). The resulting voxel grid V is then fed into a CNN with $2B$ input channels. The MLP and the CNN is updated *jointly* during training the model. This end-to-end training makes the event representation specific to the task, and allows the model to adapt event aggregation based on the input event stream. We set the number of bins to B=9, following the original EST paper [15]. We use ResNet34 [25] as the CNN architecture, initialized with pretrained weights from `ImageNet-1k_v1`. The input layer is modified to accept $2B$=18 channels instead of the original 3. The weights of

the modified input layer are initialized randomly. For implementation, we use the original code from the EST paper[1].

For each subsampling method, we apply the same subsampling level to both the training and test event videos. For each video, a subset of events is kept based on the subsampling level and type, which remains fixed throughout the training process.

3.2. Subsampling Types

We select subsampling methods which can be easily implemented in hardware. To this end, we prioritize the following characteristics:

- The method must be *causal*, meaning it cannot depend on future events. This is important for two reasons: 1) A dependence of the subsampling method on future events can severely hamper the latency of the visual task, which is crucial for many real-time, closed-loop or edge AI applications, and 2) Accessing future events would require memory units for buffering, which are known to be high-power components. A primary goal of subsampling is to improve power efficiency, and incorporating additional memory units would counteract this objective.
- We only allow methods composed of simple computing operations, which are computationally light and do not require significant processing power or memory. Using complex algorithms, such as event compression or autoencoders to compute latent representations of the event stream, is not practically feasible for direct hardware implementation.

3.2.1. Spatial Subsampling

As illustrated in Fig. 2a, we remove events both horizontally and vertically in spatial subsampling. Specifically, we retain events from every r_y-th row vertically and every r_x-th column horizontally. Horizontal and vertical offsets are denoted by $0 \leq r_{x,0} \leq r_x - 1$ and $0 \leq r_{y,0} \leq r_y - 1$, respectively. An event $e = (x, y, t, p)$ is kept if $(x - r_{x,0})$ mod $r_x = 0$, and $(y - r_{y,0})$ mod $r_y = 0$, where mod denotes modulus notation. The offset values are chosen randomly with equal probability for each training run.

3.2.2. Temporal Subsampling

The temporal subsampling is illustrated in Fig. 2b. We define a temporal window of size w_t and divide it by r_t to obtain the subsampling interval $\Delta t = \frac{w_t}{r_t}$. We also introduce a time offset, $0 \leq \Delta t_0 \leq w_t - \Delta t$, for subsampling. Thus, as shown in Figure 2b, an event $e = (x, y, t, p)$ is kept if there exists an integer k such that the event's timestamp t satisfies the following inequality:

$$kw_t + \Delta t_0 \leq t < kw_t + \Delta t + \Delta t_0. \tag{1}$$

In each training run, the offset Δt_0 is uniformly drawn from the interval $[0, w_t - \Delta t]$.

---

[1] https://github.com/uzh-rpg/rpg_event_representation_learning
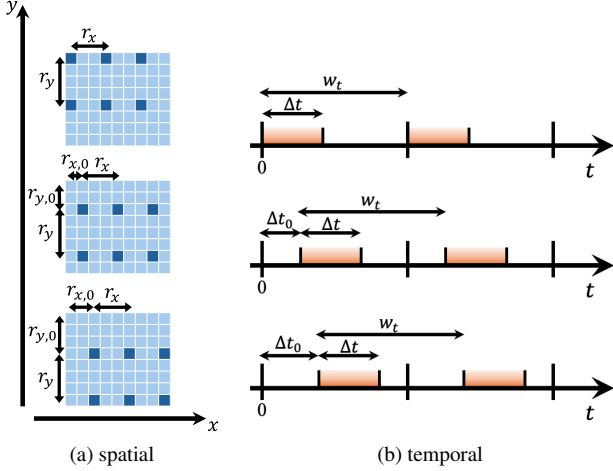
(a) spatial   (b) temporal

Figure 2. **(a) Spatial subsampling:** we keep events from every r_y-th row vertically and every r_x-th column horizontally (dark blue pixels). The horizontal and vertical offsets are denoted by $r_{x,0}$ and $r_{y,0}$, respectively. **(b) Temporal subsampling:** we keep the events within the sampling interval Δt (colored) in a temporal window of size w_t, where Δt_0 is the time offset. In both cases, the topmost subsampling example has zero offset(s).

3.2.3. Random Subsampling

Each event $e = (x, y, t, p)$ is independently retained with probability $0 \leq \rho \leq 1$. In each experiment, we first apply random subsampling to the events with probability ρ and then use the subsampled events consistently throughout training.

3.2.4. Causal Density-based Subsampling

We introduce a density-based subsampling method with the constraints of being causal, memory-efficient, and computationally inexpensive.

First, we compute a density value $f_i^{(p_i)}$ for each incoming event $e_i = (x_i, y_i, t_i, p_i)$, separately for each polarity p_i, using the following causal spatiotemporal filtering:

$$f_i^{(p_i)} = \sum_{j=1|p_j=p_i}^{i} s(x_i - x_j, y_i - y_j) \exp\left(\frac{t_i - t_j}{\tau}\right), \quad (2)$$

where $s(\cdot, \cdot)$ is a spatial kernel with filter size $w_d \times w_d$, while temporal filtering is applied using an exponential kernel with decay parameter τ. The formulation in (2) allows for separate computation of of spatial filtering $s(\cdot, \cdot)$ and temporal filtering. Moreover, the exponential temporal filtering enables recursive computation of the density value $f_i^{(p_i)}$ using the previous value of $f_{i-1}^{(p_{i-1})}$, eliminating the need for buffering past events. For the spatial kernel $s(\cdot, \cdot)$, we use a two-dimensional Gaussian kernel with a standard deviation of $w_d/5$. The spatial filter size is set to $w_d = 7$, and the temporal decay is set to $\tau = 30$ milliseconds. These

(a) Original   (b) Random threshold   (c) Fixed threshold
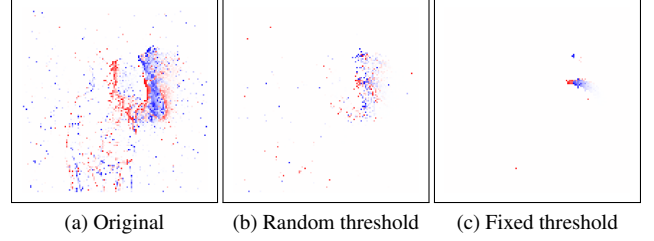
Figure 3. Causal density-based subsampling using fixed and random thresholding, retaining a similar number of events. **(a)** shows the original unfiltered events. Random thresholding **(b)** preserves the overall shape of the arm and hand movement, while fixed thresholding **(c)** focuses greedily on a small region near the hand.

values were selected based on the scene dynamics and the camera resolution in the datasets.

A higher density value, $f_i^{(p_i)}$, shows that an event comes from a denser region. After computing the density value for the incoming event $e_i = (x_i, y_i, t_i, p_i)$, we determine whether to keep the event based on a predefined threshold $f^{(\text{thresh})}$. A higher threshold value results in filtering out more events. We observe that filtering using a fixed threshold $f^{(\text{thresh})}$ for all events can cause greedy selection from very dense regions only. To prevent this, we apply a random thresholding approach. Instead of using a fixed cutoff, we introduce a random coefficient u, uniformly distributed between 0 and 1, i.e. $u \sim \mathcal{U}(0, 1)$. Particularly, we keep an event e_i if:

$$f_i^{(p_i)} \geq u \, f^{(\text{thresh})}. \quad (3)$$

Figure 3 compares the effects of fixed and random thresholding on the representation of subsampled events from a video in the DVS-Gesture dataset. As seen in the figure, fixed thresholding results in events being predominantly selected from the densest areas, such as around the hand, while random thresholding allows for a more diverse selection, capturing movement from the entire arm.

3.2.5. Event Count Subsampling

As a baseline method, we consider the Event Count subsampling approach proposed in [19], which performs subsampling in the spatial domain. In this method, the full-scale spatial image is divided into non-overlapping windows of size (r_x, r_y). For each window in the full-scale image, there is one corresponding pixel in the subsampled output, effectively spatially downscaling the events. The averaged polarity of all incoming events within a window is computed by summing the event polarities and dividing by the number of pixels in the window, $r_x \times r_y$. If this normalized event count value crosses a predefined threshold $p_{EC}^{(\text{thresh})}$, an event is triggered at the corresponding output pixel. The polarity of the triggered event is determined by whether the threshold is crossed in an increasing or decreasing manner.

3.2.6. Corner-based subsampling

Corners in an image are key interest points that carry high information content [34, 35]. Therefore, events that correspond to image corners in an event video can be good candidates for subsampling. To identify the corners, we adopt the two-dimensional Harris corner detector proposed in [18]. This method first introduces an efficient computational event representation called the Threshold-Ordinal Surface (TOS), which produces an 8-bit grayscale representation. The TOS representation can be updated per each incoming event. Harris corner detection is then performed using the `cornerHarris` function from the OpenCV library [5]. Similarly, we use the TOS representation to create a 2D event representation. With each incoming event, the TOS representation is updated, and a patch of size $w_c \times w_c$, centered at the event's spatial location, is extracted. This patch is then passed to the `cornerHarris` function to compute the Harris score h_c at the center of the patch. An event is retained as a corner if its Harris score exceeds a predefined threshold, i.e., $h_c > h_c^{(\text{thresh})}$. For parameter selection, we follow the settings from [18]. The patch size for the TOS representation is set to $w_c=7$, while the `cornerHarris` function parameters are kept at their default values: `blockSize=2`, `ksize=3`, and `k=0.04`. The Sobel operator is used to compute the horizontal and vertical derivatives.

3.3. Event Classification Datasets

N-Caltech101 [39]. This dataset is generated by displaying static images from the Caltech101 [11] dataset in front of an ATIS event camera [42] and moving the camera in three directions to trigger events. It consists of 101 classes. As all videos follow the same predefined motion pattern, the temporal details of the events do not correlate strongly with class information.

DVS-Gesture [2]. This dataset consists of various hand and arm gestures in 11 classes, recorded from 29 subjects under three different lighting conditions. The gestures were captured using a fixed DVS128 event camera with a stationary background. This dataset contains real dynamic motion generated by actual hand and arm movements rather than predefined camera motion.

N-Cars [44]. This dataset was recorded using an ATIS event camera mounted on a car driving through urban environments. It contains two classes: cars and background. Similar to the DVS-Gesture dataset, the events capture real motion dynamics. However, the camera is not fixed, leading to large background variations in the dataset.

4. Experiments

In this section, we examine the effect of subsampling methods introduced in Subsection 3.2 on classification accuracy
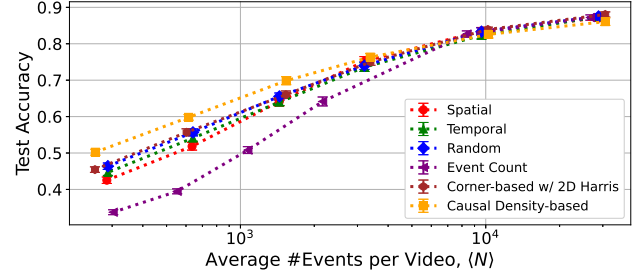

Figure 4. Classification accuracy at different subsampling levels for six subsampling methods on the **N-Caltech101** dataset. Each curve is the average of 18 independent runs with different seeds. The x-axis represents the *average* number of events per video, $\langle N \rangle$. Error bars show the standard deviation across runs.

for the datasets **N-Caltech101** [39], **DVS-Gesture** [2], and **N-Cars** [44]. Our choice of datasets spans a range of event-based inputs, from data with little class-related temporal information to recordings with diverse motion dynamics. The selection includes both static and moving cameras, as well as varying background conditions.

Classification performance is evaluated across different subsampling levels. For each subsampling level, the parameters of the subsampling methods are selected to ensure that the average number of events per video remains similar between the methods. Specific parameter values for each method are provided in the supplementary material. For all experiments, we use Adam optimizer [28] for training. The learning rate scheduler is 'Reduce on Plateau' with reducing factor 2. The patience parameter is set to 40 for N-Caltech101 and DVS-Gesture, and 50 for N-Cars. Each model is trained for 250 epochs. For the batch size and learning rate, we follow the suggested parameters in [3] for each datasets. We used the same hyperparameters across all subsampling levels.

4.1. N-Caltech101 Dataset

In Figure 4, we present the classification test accuracy of six subsampling methods for N-Caltech101 dataset. The horizontal axis shows the average number of events per video $\langle N \rangle$. The accuracy–#events curves are generated by evaluating classification test accuracy across different subsampling levels. Each curve represents the average performance over 18 independent runs with different random seeds.

The results show that input independent methods of spatial, temporal, and random subsampling perform better than the baseline method proposed in [19][2]. This highlights the importance of considering simple subsampling techniques as baselines while evaluating more complex approaches. Interestingly, random subsampling achieves slightly better

---

[2]For the implementation of the Event Count method, we used the repository: https://github.com/ameliegruel/EvVisu
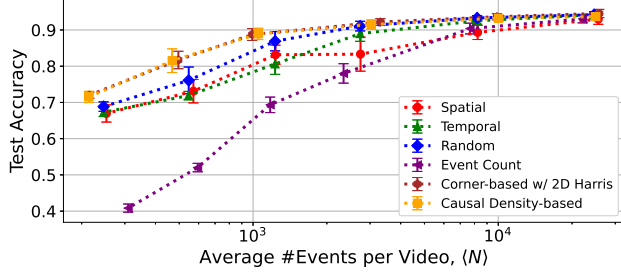
Figure 5. Classification accuracy at different subsampling levels for six subsampling methods on the **DVS-Gesture** datasets. Each curve is the average of 18 independent runs with different seeds. Error bars show the standard deviation across runs.
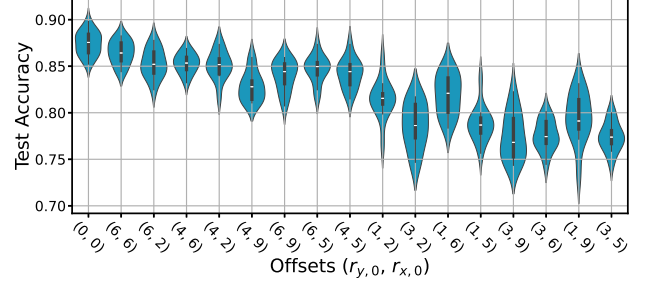

Figure 6. Effect of different sets of horizontal and vertical offsets in spatial subsampling on the test accuracy for DVS-Gesture dataset. The horizontal subsampling is $r_x = 10$ and vertical is $r_y = 8$.

(a) Spatial subsampling

(b) Temporal subsampling

Figure 7. Comparing the offset sensitivity between **(a)** spatial and **(b)** temporal subsampling in DVS-Gesture dataset for different levels of sparsity. We observe that temporal subsampling is less sensitive to changes in the offset.

performance than spatial and temporal subsampling. This is noteworthy because most event camera hardware reduces event rates through spatial and/or temporal subsampling, while our results suggest that they may be suboptimal compared to random subsampling.

Among the more complex, input dependent methods—casual density-based and Harris corner-based subsampling—the density-based method achieves the highest accuracy, particularly in the sparser event regimes, while the corner-based method performs comparably to random subsampling. The differences are insightful considering the small magnitude of the standard deviations (error bars). This result supports the hypothesis that denser regions in event data carry more information, and subsampling from these areas leads to better classification accuracy.

4.2. DVS-Gesture Dataset

The accuracy curves for DVS-Gesture dataset is presented in Figure 5 for the six subsampling methods. The curves are the average of 18 independent runs. The results for this dataset align with those of N-Caltech101 in Subsection 4.1. Specifically, the trivial, input-independent methods—spatial, temporal, and random subsampling—outperform the baseline Event Count method [19]. Among these, random subsampling has better accuracy than spatial and temporal. However, in this experiment, the corner-based method achieves similar accuracy to the causal density-based method, with both methods surpassing the other subsampling strategies. This suggests that corner events are particularly informative for classifying DVS-Gesture.

An important observation in Figure 5 is the relatively high standard deviation of accuracies (i.e., diverse performance) for the spatial subsampling method. By investigating this high variance, we find that it is due to the high sensitivity of spatial subsampling to offset selection. Figure 6 illustrates this sensitivity, where we subsample horizontally with $r_x = 10$ and vertically with $r_y = 8$. Test accuracies
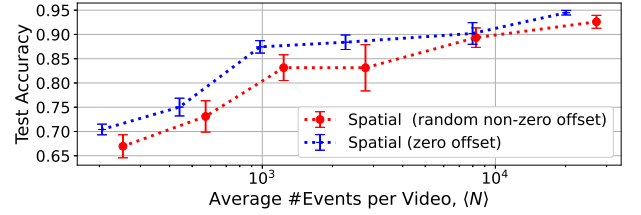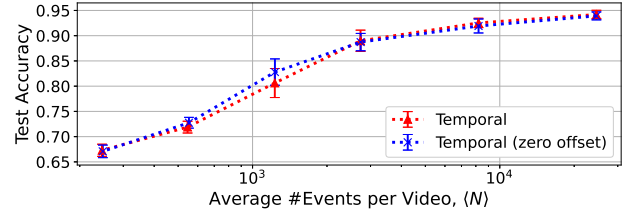
are computed for different offset configurations, each evaluated with multiple random seeds. The high performance differences for various offsets shows the high sensitivity of the spatial subsampling to offset selection. This result suggests that spatial subsampling not only removes high-frequency spatial information but can also distort low-frequency information due to spatial aliasing effects. Combined, our results indicate that naive spatial subsampling at the hardware level may hamper downstream tasks depending on the frequency content of the video.

Figure 7 compares classification accuracy for spatial and temporal subsampling under two conditions: zero offset and random non-zero offset, across different subsampling levels. The results show that temporal subsampling has lower sensitivity to offset variations compared to spatial subsampling. This highlights an advantage of temporal sub-
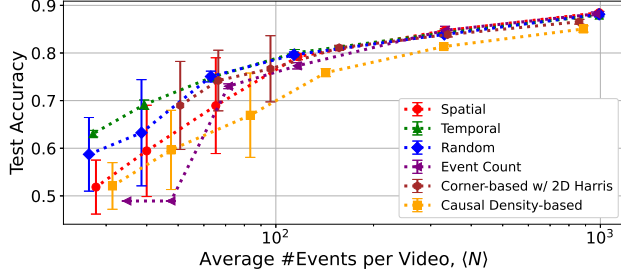
Figure 8. Classification accuracy at different subsampling levels for six subsampling methods on the **N-Cars** dataset. Each curve is the average of six independent runs with different seeds. Error bars show the standard deviation across runs.
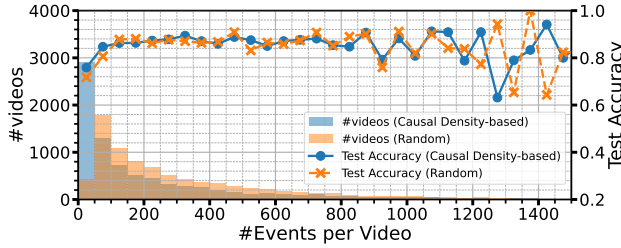

Figure 9. Causal density-based subsampling results in many videos with very low number of events (bar plot, blue) compared to random subsampling (bar plot, orange). Such videos (e.g. first blue bar, with < 50 events per video, distributed over 18 input channels) achieve lower classification test accuracy (line plots), which in turn lead to lower mean test accuracy for the causal density-based method.

sampling in terms of offset robustness. Given that spatial subsampling is commonly used for event rate reduction—typically by discarding events from specific rows and/or columns—we highlight that temporal subsampling may be a more stable alternative, considering the greater susceptibility of spatial subsampling to the effect of offset.

4.3. N-Cars Dataset

In Figure 8, we present the results of different subsampling methods on the two-class object classification dataset N-Cars. Due to the larger dataset size and higher training time, we average over six independent runs. The number of events in N-Cars is significantly lower than in N-Caltech101 and DVS-Gesture. As a result, we see that subsampling low-rate datasets like N-Cars can lead to performance failure. As shown in Figure 8, below 100 events per video, most subsampling methods fail to classify the videos. This is both evidenced by the drop in test accuracy towards 50% (chance level) and the large accuracy variation across runs (error bars). Only temporal subsampling seems to perform slightly better than the others in the sparse regime, and is more stable across runs.

Another key observation is the relative accuracy drop in

the causal density-based method. This can be attributed to the causal nature of the method, which is agnostic to the total number of events per video. Since the method cannot know how many events will be recorded before the end of each video, the filter parameters (Eq. 2) as well as the threshold $f^{(\text{thresh})}$ is fixed for the whole dataset. This means that, in the case of the N-Cars dataset, where the variance of the total number of events across samples is large, the causal density-based subsampling leads to a large number of samples with close to zero events in the video. This effect is illustrated in Figure 9, which shows the histogram (bar plot) of the number of events per video *after* both random and causal density-based subsampling. Since density values $f_i^{(p_i)}$ in sparse videos are low, more events are removed after applying a fixed threshold to the density values.

To mitigate this issue, we normalize the density values before thresholding, preventing excessive subsampling of sparser videos. As shown in Figure 10, normalizing the density values improves accuracy. This normalization cost is to lose the causal property of the subsampling process, however, it demonstrates that the density-based subsampling approach can still capture informative events in the N-Cars dataset. In practice, a similar improvement can be achieved by simply employing an adaptive threshold, which can, for example, be lowered when the event rate is low and vice versa. In this case, the additional memory and power requirements of keeping a running average of the event rate would be implementation-dependent.

4.4. Performance Overview Across Datasets

To provide a quantitative comparison between the different subsampling methods, we define a metric evaluating their task performance across all subsampling levels. Given that the primary objective of subsampling is to maintain information effectively in settings with a low number of events, we want to place greater emphasis on accuracy in this regime. To achieve this, we compute the area under the curve (AUC) for the test accuracy as a function of the logarithm of number of events:

$$\text{AUC}_{\text{acc-\#events}} = \int \text{acc}(\text{\#events}) d\left(\log_{10} \text{\#events}\right). \quad (4)$$

To normalize this value, we divide the AUC in (4) by $\text{AUC}_{\text{acc-\#events}}^{(oracle)}$, which represents the AUC of an ideal classifier achieving an accuracy of 1.0 across all number of events:

$$\text{nAUC}_{\text{acc-\#events}} = \frac{\text{AUC}_{\text{acc-\#events}}}{\text{AUC}_{\text{acc-\#events}}^{(oracle)}}. \quad (5)$$

The normalized AUC provides a quantitative measure of the overall classification accuracy of subsampling methods, with an increased weight on the low-event regime. We compute $\text{nAUC}_{\text{acc-\#events}}$ for each subsampling method using 18 independent runs for N-Caltech101 and DVS-Gesture and
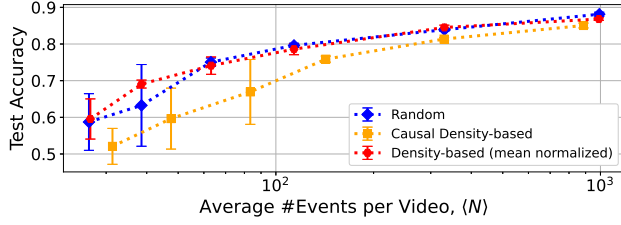
Figure 10. Normalizing the density values $f_i{}^{(p_i)}$ before thresholding (red) improves the performance of density-based subsampling (yellow). Random subsampling (blue) given for comparison.

6 for N-Cars. For N-Cars, we only consider experiments where $\langle N \rangle > 50$ to filter out runs with highly variant accuracy due to very low total event counts.

Table 1 presents the mean and standard deviation of nAUC$_{acc\text{-}\#events}$ for different subsampling methods on the three datasets. In N-Caltech101, the causal density-based method outperforms other approaches, with random and corner-based subsampling coming in second. In DVS-Gesture, the causal density-based and corner-based subsampling methods achieve similarly high performance compared to other methods. For N-Cars, temporal subsampling performs slightly better than other methods, while most approaches achieve similar performance with nAUC ≈ 0.82. The causal density-based method initially performs suboptimally with nAUC ≈ 0.79. However, as analyzed in Subsection 4.3, its performance improves after normalizing the density values, bringing it back to 0.82.

Taken together, within the class of more advanced subsampling methods we evaluate, corner-based and density-based methods seem to demonstrate the best accuracy-#event efficiency balance. In terms of computational complexity the causal density-based method displays a slight advantage. Table 2 shows the memory usage and computational cost in multiply–accumulate operations (MACs) for an event video with an spatial resolution of $H \times W$, and total event number of N. Simple methods such as spatial, temporal, and random subsampling require only $\mathcal{O}(1)$ memory and no specific MAC operation. The Event Count method [19] has higher memory usage compared to these simpler methods. For more complex methods, such as corner-based and causal density-based subsampling, memory usage scales with the camera resolution ($\mathcal{O}(HW)$), while computational cost depends on the square of filter sizes used in these algorithms. However, for the same filter size ($w_d = w_c$), the computational cost of the corner-based method is higher due to the additional operations required for the Harris corner detection algorithm. More details are provided in supplementary material.

Table 1. Mean and standard deviation of nAUC$_{acc\text{-}\#events}$ for different subsampling methods. The highest value for each dataset is highlighted in **bold**, and the second highest with an <u>underline</u>.

Method	N-Caltech101	DVS-Gesture	N-Cars
Spatial	0.697 ± 0.005	0.827 ± 0.016	0.818 ± 0.009
Temporal	0.697 ± 0.004	0.841 ± 0.007	$\mathbf{0.827 \pm 0.002}$
Random	$\underline{0.708} \pm 0.003$	0.867 ± 0.009	$\underline{0.825} \pm 0.003$
Event Count	0.648 ± 0.006	0.748 ± 0.007	0.822 ± 0.005
Corner-based w/ 2D Harris	0.707 ± 0.003	$\mathbf{0.886 \pm 0.008}$	0.823 ± 0.001
Causal Density-based	$\mathbf{0.723 \pm 0.004}$	$\underline{0.883} \pm 0.009$	0.789 ± 0.011
Density-based (mean normalized)	—	—	0.821 ± 0.008

Table 2. Memory usage (#units) and computational complexity (MACs) for different subsampling methods. Note that the complexity of the corner-based method is higher than that of the causal density-based method for the same filter size ($w_d = w_c$).

Subsampling method	Memory	Computational complexity (MACs)
Spatial	$\mathcal{O}(1)$	0
Temporal	$\mathcal{O}(1)$	0
Random	$\mathcal{O}(1)$	0
Event Count	$\mathcal{O}\left(\frac{HW}{r_y r_x}\right)$	N
Corner-based	$\mathcal{O}(HW)$	$(2\,\texttt{ksize}^2 + 3\,\texttt{blockSize}^2 + 10)\,w_c{}^2\,N$
Causal density-based	$\mathcal{O}(HW)$	$4\,w_d{}^2\,N$

5. Discussion and Conclusion

In this paper, we study the trade-offs between hardware-friendly subsampling methods and their impact on object classification accuracy. To the best of our knowledge, no prior work has systematically compared these methods in terms of task performance. Our analysis offers insights into selecting event rate reduction methods, whether implemented in hardware or software. Among simpler methods with little computation requirements, random subsampling tends to perform better in more scenarios compared to temporal and spatial subsampling. While spatial subsampling can be highly sensitive to offset effects, more advanced methods, such as density-based and corner-based subsampling, achieve superior classification performance by selectively preserving more informative events from dense regions or corner areas. We also observe that high variance in event counts across videos can harm the subsampling methods. To mitigate this effect, adaptive thresholding or an on-off mechanism for dynamically activating the subsampling method could be helpful.

Limitations. Our analysis is limited to classification tasks using CNNs on three datasets. However, the training procedure for different subsampling methods can be applied to other vision tasks and models as a direction for future research. The proposed density-based method primarily serves as a proof-of-concept to test the hypothesis of density-based subsampling, rather than aiming at computational optimization. Future exploration can be considered for optimizing the density-based filtering using techniques from [22, 26, 33] to improve efficiency in both memory usage and computational complexity.

References

[1] Ignacio Alzugaray and Margarita Chli. Asynchronous Corner Detection and Tracking for Event Cameras in Real Time. *IEEE Robotics and Automation Letters*, 3(4):3177–3184, 2018. Conference Name: IEEE Robotics and Automation Letters. 2

[2] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, Jeff Kusnitz, Michael Debole, Steve Esser, Tobi Delbruck, Myron Flickner, and Dharmendra Modha. A Low Power, Fully Event-Based Gesture Recognition System. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7388–7397, 2017. ISSN: 1063-6919. 1, 5

[3] Hesam Araghi, Jan van Gemert, and Nergis Tomen. Pushing the boundaries of event subsampling in event-based video classification using CNNs, 2024. arXiv:2409.08953 [cs]. 1, 2, 5

[4] R. Wes Baldwin, Ruixu Liu, Mohammed Almatrafi, Vijayan Asari, and Keigo Hirakawa. Time-Ordered Recent Event (TORE) Volumes for Event Cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):2519–2532, 2023. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence. 2, 3

[5] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 5

[6] Marco Cannici, Marco Ciccone, Andrea Romanoni, and Matteo Matteucci. A Differentiable Recurrent Surface for Asynchronous Event-Based Data, 2020. arXiv:2001.03455 [cs]. 2, 3

[7] Andrea Censi, Erich Mueller, Emilio Frazzoli, and Stefano Soatto. A Power-Performance Approach to Comparing Sensor Families, with application to comparing neuromorphic to traditional vision sensors. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3319–3326, Seattle, WA, USA, 2015. IEEE. 1

[8] Gregory Cohen, Saeed Afshar, Garrick Orchard, Jonathan Tapson, Ryad Benosman, and André van Schaik. Spatial and Temporal Downsampling in Event-Based Visual Classification. *IEEE Transactions on Neural Networks and Learning Systems*, 29(10):5030–5044, 2018. Conference Name: IEEE Transactions on Neural Networks and Learning Systems. 2

[9] Tobi Delbruck, Rui Graca, and Marcin Paluch. Feedback control of event cameras. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1324–1332, Nashville, TN, USA, 2021. IEEE. 2

[10] Wei Fang, Zhaofei Yu, Yanqi Chen, Timothée Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2661–2671, 2021. 2

[11] 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*, pages 178–178. IEEE, 2004. 5

[12] Yang Feng, Hengyi Lv, Hailong Liu, Yisa Zhang, Yuyao Xiao, and Chengshan Han. Event Density Based Denoising Method for Dynamic Vision Sensor. *Applied Sciences*, 10(6):2024, 2020. Number: 6 Publisher: Multidisciplinary Digital Publishing Institute. 2

[13] Thomas Finateu, Atsumi Niwa, Daniel Matolin, Koya Tsuchimoto, Andrea Mascheroni, Etienne Reynaud, Pooria Mostafalu, Frederick Brady, Ludovic Chotard, Florian LeGoff, Hirotsugu Takahashi, Hayato Wakabayashi, Yusuke Oike, and Christoph Posch. A 1280×720 back-illuminated stacked temporal contrast event-based vision sensor with 4.86µm pixels, 1.066GEPS readout, programmable event-rate controller and compressive data-formatting pipeline. In *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*. IEEE, 2020. 2

[14] Guillermo Gallego, Tobi Delbruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J. Davison, Jorg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-Based Vision: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(1):154–180, 2022. 1, 2

[15] Daniel Gehrig, Antonio Loquercio, Konstantinos Derpanis, and Davide Scaramuzza. End-to-End Learning of Representations for Asynchronous Event-Based Data. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5632–5642, Seoul, Korea (South), 2019. IEEE. 2, 3

[16] Mathias Gehrig, Manasi Muglikar, and Davide Scaramuzza. Dense Continuous-Time Optical Flow from Events and Frames. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(7):4736–4746, 2024. arXiv:2203.13674 [cs]. 2

[17] Anindya Ghosh, Thomas Nowotny, and James Knight. Ev-Downsampling: a robust method for downsampling event camera data. In *ECCV workshop*, 2024. 2

[18] Arren Glover, Aiko Dinale, Leandro De Souza Rosa, Simeon Bamford, and Chiara Bartolozzi. luvHarris: A Practical Corner Detector for Event-Cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):10087–10098, 2022. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence. 2, 5, 1

[19] Amélie Gruel, Jean Martinet, Teresa Serrano-Gotarredona, and Bernabé Linares-Barranco. Event Data Downscaling for Embedded Computer Vision:. In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, pages 245–253, Online Streaming, — Select a Country —, 2022. SCITEPRESS - Science and Technology Publications. 2, 4, 5, 6, 8

[20] Amelie Gruel, Jean Martinet, Bernabe Linares-Barranco, and Teresa Serrano-Gotarredona. Performance comparison of DVS data spatial downscaling methods using Spiking Neural Networks. In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 6483–6491, Waikoloa, HI, USA, 2023. IEEE. 2

[21] Fuqiang Gu, Weicong Sng, Xuke Hu, and Fangwen Yu. EventDrop: data augmentation for event-based learning, 2021. arXiv:2106.05836 [cs]. 2

[22] Shasha Guo and Tobi Delbruck. Low Cost and Latency Event Camera Background Activity Denoising. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1): 785–795, 2023. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence. 2, 8, 1

[23] Drew Hanover, Antonio Loquercio, Leonard Bauersfeld, Angel Romero, Robert Penicka, Yunlong Song, Giovanni Cioffi, Elia Kaufmann, and Davide Scaramuzza. Autonomous Drone Racing: A Survey, 2023. arXiv:2301.01755 [cs]. 1

[24] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the Alvey Vision Conference*, pages 23.1–23.6, 1988. 2

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition, 2015. arXiv:1512.03385 [cs]. 3

[26] Alireza Khodamoradi and Ryan Kastner. O(N)O(N)-Space Spatiotemporal Filter for Reducing Noise in Neuromorphic Vision Sensors. *IEEE Transactions on Emerging Topics in Computing*, 9(1):15–23, 2021. Conference Name: IEEE Transactions on Emerging Topics in Computing. 2, 8, 1

[27] Junho Kim, Inwoo Hwang, and Young Min Kim. Ev-TTA: Test-Time Adaptation for Event-Based Object Recognition. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17724–17733, New Orleans, LA, USA, 2022. IEEE. 2

[28] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, 2017. arXiv:1412.6980 [cs]. 5

[29] Simon Klenk, David Bonello, Lukas Koestler, and Daniel Cremers. Masked Event Modeling: Self-Supervised Pretraining for Event Cameras, 2022. arXiv:2212.10368 [cs]. 2

[30] Jürgen Kogler, Christoph Sulzbachner, and Wilfried Kubinger. Bio-inspired stereo vision system with silicon retina imagers. In *International Conference on Computer Vision Systems*, pages 174–183. Springer, 2009. 3

[31] 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*, 39(7):1346–1359, 2017. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence. 2

[32] Ruoxiang Li, Dianxi Shi, Yongjun Zhang, Kaiyue Li, and Ruihao Li. FA-Harris: A Fast and Asynchronous Corner Detector for Event Cameras. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6223–6229, 2019. ISSN: 2153-0866. 2

[33] Hongjie Liu, Christian Brandli, Chenghan Li, Shih-Chii Liu, and Tobi Delbruck. Design of a spatiotemporal correlation filter for event-based sensors. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 722–725, 2015. 2, 8, 1

[34] Marco Loog. Information theoretic preattentive saliency: A closed-form solution. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1418–1424, 2011. 2, 5

[35] Marco Loog and Francois Lauze. The Improbability of Harris Interest Points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):1141–1147, 2010. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence. 2, 5

[36] Misha Mahowald. *VLSI analogs of neuronal visual processing: a synthesis of form and function*. PhD thesis, California Institute of Technology Pasadena, 1992. 2

[37] Ana I. Maqueda, Antonio Loquercio, Guillermo Gallego, Narciso Garcia, and Davide Scaramuzza. Event-Based Vision Meets Deep Learning on Steering Prediction for Self-Driving Cars. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5419–5427, Salt Lake City, UT, 2018. IEEE. 3

[38] Elias Mueggler, Chiara Bartolozzi, and Davide Scaramuzza. Fast Event-based Corner Detection. In *Procedings of the British Machine Vision Conference 2017*, page 33, London, UK, 2017. British Machine Vision Association. 2

[39] Garrick Orchard, Ajinkya Jayawant, Gregory K. Cohen, and Nitish Thakor. Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades. *Frontiers in Neuroscience*, 9, 2015. 5

[40] Garrick Orchard, Cedric Meyer, Ralph Etienne-Cummings, Christoph Posch, Nitish Thakor, and Ryad Benosman. HFirst: A Temporal Approach to Object Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(10):2028–2040, 2015. arXiv:1508.01176 [cs]. 2

[41] Yansong Peng, Yueyi Zhang, Zhiwei Xiong, Xiaoyan Sun, and Feng Wu. GET: Group Event Transformer for Event-Based Vision, 2023. arXiv:2310.02642 [cs]. 2

[42] Christoph Posch, Daniel Matolin, and Rainer Wohlgenannt. A qvga 143 db dynamic range frame-free pwm image sensor with lossless pixel-level video compression and time-domain cds. *IEEE Journal of Solid-State Circuits*, 46(1):259–275, 2010. 5

[43] Yusuke Sekikawa, Kosuke Hara, and Hideo Saito. EventNet: Asynchronous Recursive Event Processing. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3882–3891, Long Beach, CA, USA, 2019. IEEE. 2

[44] Amos Sironi, Manuele Brambilla, Nicolas Bourdis, Xavier Lagorce, and Ryad Benosman. HATS: Histograms of Averaged Time Surfaces for Robust Event-Based Object Classification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1731–1740, Salt Lake City, UT, USA, 2018. IEEE. 2, 3, 5

[45] Valentina Vasco, Arren Glover, and Chiara Bartolozzi. Fast event-based Harris corner detection exploiting the advantages of event-driven cameras. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4144–4149, 2016. ISSN: 2153-0866. 2

[46] Alex Vicente-Sola, Davide L. Manna, Paul Kirkland, Gaetano Di Caterina, and Trevor Bihl. Spiking Neural Networks

for event-based action recognition: A new task to understand their advantage, 2023. arXiv:2209.14915 [cs]. 2

[47] Xu Zheng, Yexin Liu, Yunfan Lu, Tongyan Hua, Tianbo Pan, Weiming Zhang, Dacheng Tao, and Lin Wang. Deep Learning for Event-based Vision: A Comprehensive Survey and Benchmarks, 2023. arXiv:2302.08890 [cs]. 2

[48] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Unsupervised Event-Based Learning of Optical Flow, Depth, and Egomotion. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 989–997, Long Beach, CA, USA, 2019. IEEE. 3

[49] Nikola Zubić, Daniel Gehrig, Mathias Gehrig, and Davide Scaramuzza. From Chaos Comes Order: Ordering Event Representations for Object Recognition and Detection, 2023. arXiv:2304.13455 [cs]. 2

Making Every Event Count:
Balancing Data Efficiency and Accuracy in Event Camera Subsampling

Supplementary Material

6. Details of Subsampling Parameters

Table 3 presents the specific parameters used for each subsampling method across various subsampling levels. The parameters are chosen to ensure that the average number of events $\langle N \rangle$ remains similar across different subsampling methods at each subsampling level for each dataset.

7. Memory Usage and Computational Complexity

In Table 2, we compare the memory usage and computational complexity across six different subsampling methods. We report the total memory units required for an event camera of size $H \times W$, and computational complexity in terms of multiply-accumulate operations (MACs) for a video with N number of events. Spatial, temporal, and random subsampling require only $\mathcal{O}(1)$ memory for storing a few method-specific parameters and essentially no specific MAC operation. Event Count method uses memory proportional to the downscaled spatial grid size $\left(\frac{H}{r_y}\right) \times \left(\frac{W}{r_x}\right)$, and need N MAC operation for computing the normalized event count per each incoming event. The corner-based subsampling method adapted from [18] requires $\mathcal{O}(HW)$ memory for the event representation frame. The MAC number contains vertical and horizontal Sobel filtering $2 \, \texttt{ksize}^2 \, w_c{}^2 \, N$, appyling filtering for computing the structural tensor $3 \, \texttt{blockSize}^2 \, w_c{}^2 \, N$, and $10 \, w_c{}^2 \, N$ for other computations including Harris score calculation. The causal density-based subsampling also requires $\mathcal{O}(HW)$ memory for the method explained in Subsection 3.2. The computational complexity is $4 \, w_d{}^2 \, N$ MAC operations.

For an exemplary comparison between the computational operations of the corner-based and causal density-based methods, based on the chosen parameters in Table 3, the per-event computing cost for the corner-based method is $40 \, w_c{}^2$, while for the causal density-based method, it is $4 \, w_d{}^2$, where $w_c = w_d = 7$.

It is important to note that in Subsection 3.2, our focus was not on optimizing memory usage or computational complexity but rather on analyzing the accuracy performance of density-based methods across different subsampling levels. There are existing approaches aimed at developing efficient spatiotemporal filtering methods [22, 26, 33] that can improve both memory efficiency and computational complexity. For instance, in [26], the authors proposed a spatiotemporal filtering technique that reduces memory usage from $\mathcal{O}(HW)$ to $\mathcal{O}((HW)^{\frac{1}{2}})$.

8. Visualization of subsampling methods

Figure 11 provides a visualization of the effect of different subsampling methods on event data. In the first row, we show the original event data without any subsampling. Starting from the second row, the figure illustrates the results for two different subsampling levels applied to each dataset. Each image is labeled with the corresponding number of subsampled events, which are consistent across the different subsampling methods.

Table 3. Parameters of different subsampling methods for various subsampling levels: from 1 (most #events) to 6 (least #events). **mS**: milliseconds.

| Subsampling methods	parameters	dataset	subsampling levels (1: most #events, 6: least #events)					
			1	2	3	4	5	6
Spatial	(r_x, r_y)	same for all	(2,2)	(4,3)	(6,6)	(12,10)	(15,12)	(25,16)
Temporal	r_t	same for all	4	12	36	120	180	400
	w_t (mS)	same for all	10	10	10	10	10	10
Random	ρ	same for all	$\frac{1}{4}$	$\frac{1}{12}$	$\frac{1}{36}$	$\frac{1}{120}$	$\frac{1}{180}$	$\frac{1}{400}$
Event Count	(r_x, r_y)	same for all	(2,2)	(4,3)	(6,6)	(12,10)	(15,12)	(25,16)
	$p_{EC}^{(\text{thresh})}$	same for all	0.75	0.75	1.0	1.0	1.0	1.0
Corner-based	w_c	same for all	7×7	7×7	7×7	7×7	7×7	7×7
	`ksize`	same for all	3	3	3	3	3	3
	`blockSize`	same for all	2	2	2	2	2	2
	`k`	same for all	0.04	0.04	0.04	0.04	0.04	0.04
	$h_c^{(\text{thresh})}$	N-Caltech101	0.067	0.23	0.68	1.52	3.85	9.10
		N-Cars	0.091	0.25	0.56	1.0	1.67	2.5
		DVS-Gesture	0.077	0.17	0.5	16.7	3.33	7.70
Causal density-based	τ (mS)	same for all	30	30	30	30	30	30
	w_d	same for all	7×7	7×7	7×7	7×7	7×7	7×7
	$f^{(\text{thresh})}$	N-Caltech101 & N-Cars	3.33	10.0	30.0	66.67	166.67	400.0
		DVS-Gesture	4.63	11.63	38.56	111.11	250.0	555.56
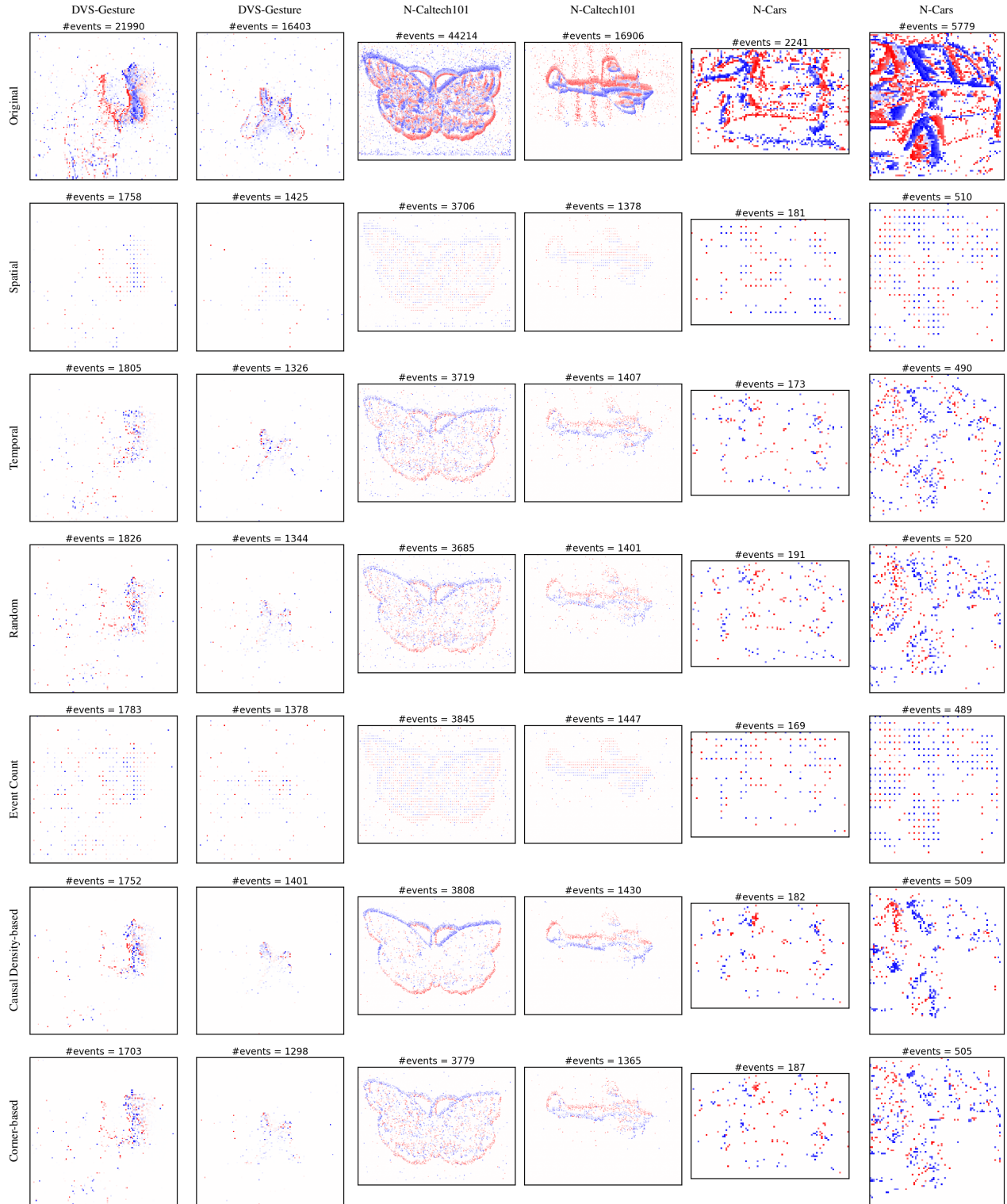
Figure 11. Visualization of different subsampling methods (starting from the second row). The first row shows the original data. We show for two videos for each dataset. The title of each image is the number of subsampled events. The number of events for different subsampling methods are similar.