Pushing the boundaries of event subsampling in event-based video classification using CNNs

Hesam Araghi, Jan van Gemert, and Nergis Tomen

Computer Vision Lab, Delft University of Technology
{h.araghi, j.c.vangemert, n.tomen}@tudelft.nl

Abstract. Event cameras offer low-power visual sensing capabilities ideal for edge-device applications. However, their high event rate, driven by high temporal details, can be restrictive in terms of bandwidth and computational resources. In edge AI applications, determining the minimum amount of events for specific tasks can allow reducing the event rate to improve bandwidth, memory, and processing efficiency. In this paper, we study the effect of event subsampling on the accuracy of event data classification using convolutional neural network (CNN) models. Surprisingly, across various datasets, the number of events per video can be reduced by an order of magnitude with little drop in accuracy, revealing the extent to which we can push the boundaries in accuracy vs. event rate trade-off. Additionally, we also find that lower classification accuracy in high subsampling rates is not solely attributable to information loss due to the subsampling of the events, but that the training of CNNs can be challenging in highly subsampled scenarios, where the sensitivity to hyperparameters increases. We quantify training instability across multiple event-based classification datasets using a novel metric for evaluating the hyperparameter sensitivity of CNNs in different subsampling settings. Finally, we analyze the weight gradients of the network to gain insight into this instability. The code and additional resources for this paper can be found at: https://github.com/hesamaraghi/pushing-boundaries-event-subsampling.

Keywords: Event cameras · Event-based video classification · Sparsity of frame representation in event processing · Accuracy vs. event rate trade-off · Hyperparameter sensitivity in event data

1 Introduction

Event cameras are power-efficient visual sensors thanks to capturing only *changes* in light intensity. They combine exceptionally high temporal resolution and low power consumption, making them suitable for edge-device applications [2,12,19]. Examples include real-time interaction scenarios such as simultaneous localization and mapping (SLAM) in small aerial vehicles [38] and obstacle avoidance [8]. In these applications, the visual classification task is essential for identifying, avoiding, or following objects, and navigating through them, for which the convolutional neural network (CNN) is a popular and well-established model. Considering the total load of the visual pipeline in such edge applications, event cameras have the potential to significantly enhance the overall bandwidth and power efficiency of the system [5]. However, how to maximize the efficiency gains in the complete framework without compromising task accuracy is not well-studied.

In high-motion scenarios, the high temporal resolution of event cameras generates a large number of events, and the maximum event rate can reach up to one billion events per second [10,14,34]. Consequently, transmitting and processing such high-rate event data requires substantial bandwidth and computational resources, reducing the power efficiency of pipelines with event-based sensors in high-motion scenarios. Increasing the data bandwidth is not always straightforward in edge-device applications, motivating workarounds like filtering out events in hardware [10] or using event rate control [7] to prevent the bus saturation. Another approach is subsampling events to reduce the computational cost. However, subsampling risks information loss and can adversely affect the downstream tasks. The relationship between subsampling level and its effects on task performance is not straightforward and has been little explored in literature. Determining the minimum necessary number of events for particular tasks enables us to tailor the computational processes for edge applications, thereby enhancing the power efficiency and computational effectiveness of the event processing unit.

Here, we study the effect of subsampling on the performance of event data classification using a CNN model. Surprisingly, our findings show that across multiple datasets, the number of events per video can be reduced by an order of magnitude without sacrificing considerable accuracy. Moreover, we observe that classification accuracy often remains significantly above chance level even when using as few as 8 or 16 events per video in total (see Figure 1). This shows that an unexpected amount of task-relevant information can be retained in just a few events and showcases the potential for greatly reducing the processing power requirements for event-based vision.

At the same time, we uncover unexpected challenges associated with subsampling. Specifically, we discover that higher subsampling rates can make the training of CNNs unstable. We explore this subsampling instability and analyze the gradient flow and hyperparameter sensitivity.

The contributions of the paper can be summarized as follows:

- We investigate, for the first time, how subsampling affects CNN task performance across various event classification datasets.
- We highlight the increased hyperparameter sensitivity in CNN training, particularly at higher subsampling rates.
- We introduce a novel metric to quantify the CNN hyperparameter sensitivity in extremely sparse input regimes.
- We perform a detailed analysis of the gradients to evaluate the effect of subsampling on CNN training stability.

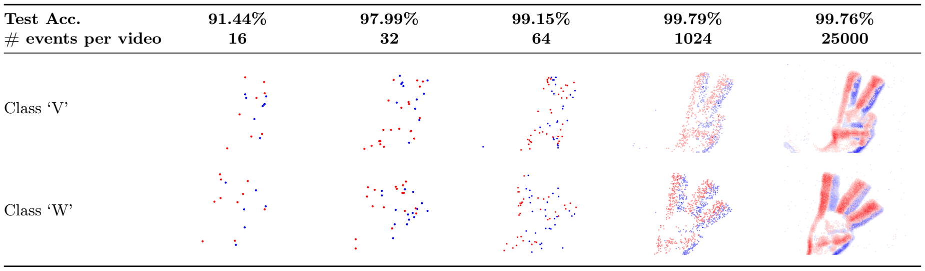


Fig. 1: Illustration of two classes (‘V’ and ‘W’) from the Neuromorphic American Sign Language (N-ASL) dataset, which includes 24 classes, for different numbers of events per video. The events are accumulated into a single frame, with red and blue indicating the two polarities. Even with a significantly reduced number of events, a CNN-based classifier can still achieve very high accuracies.

2 Related work

2.1 Event Classification Using CNNs

Many tasks in event-based machine vision rely on deep neural networks (DNNs) for processing the events [40]. To use a DNN model, we need to represent the events in a format which is compatible with the network. One of the commonly used DNNs for event-based machine vision is the convolutional neural network (CNN). We concentrate on CNN models in this study since they can offer a balance between task performance and computational efficiency: While computationally ‘lighter’ methods like handcrafted approaches [23, 29, 31] and spiking neural networks [24, 32, 37] show limited performance in complex applications, more recent solutions such as visual transformers [22, 39, 41] come with a high computational cost, diminishing the low power advantages of event cameras during processing.

For processing the events in a CNN, we first need to convert events into frames where events are aggregated into an image-like format. Various types of frame-based representations are proposed. Time surface representation is an image, with pixel values representing timestamps. For instance, [23] stores the timestamp of the most recent event for each pixel. Time surface enables asynchronous updates and retains temporal information of last events. Event count/histogram is another image representation generated by counting events in a specific time window for each pixel [26]. This representation can have the spatial information of the edges, but it may result in blurry outputs for fast-moving objects. Another approach is the voxel grid representation [42], which discretizes the time window into multiple bins and computes an image for each bin through a weighted summation of events near that bin. In contrast to predefined representations, [13] proposed the EST algorithm, which is a learning-based method to generate a voxel grid representation which can be trained flexibly for various tasks. Due to its end-to-end representation learning property, we use the EST algorithm [13] to assess the classification performance of CNNs across different levels of sparsity.

We specifically investigate the effects of input *sparsity* on training and classification accuracy, since input frames constructed from subsampled events lead to increased sparsity and thus to more efficient processing.

2.2 Sparsity in Event Cameras

Previous works considering sparsity to improve computational efficiency in event processing include modeling events as point-clouds [31], graphs [15], and employing sparse CNNs [27]. These models can reduce the computational costs by increasing the sparsity. However, in this work, we focus on accuracy-sparsity trade-off instead of computational efficiency. To reduce the bandwidth and computation costs, [11] proposes using a small subset of pixels generating events for visual place recognition. However, this method relies on event counting for selected pixels and may not be suitable for other tasks like image classification. In [18], the authors introduce spatial downscaling methods based on either averaging events or estimating luminance. Similarly, in [16], two additional spatial downscaling methods are proposed, using SNN pooling. These techniques effectively reduce the number of events, and their effect on classification accuracy using SNNs is studied in [16, 17]. The work by [6] examines the effect of event downsampling on event rate reduction and classification accuracy, suggesting that reducing the spatial and temporal resolution of input data can improve classification accuracy and lower data rates while retaining the number of events. They employ a synaptic kernel inverse method (SKIM) [35]. However, the performance of SKIM and SNN-based methods used in [6, 16, 17] does not match state-of-the-art methods, raising questions about the generalizability of their findings to more accurate models like CNNs. In contrast, our work focuses on the impact of event subsampling, where the number of events is significantly reduced. We examine the effect of subsampling across multiple event datasets and introduce the challenges of training networks with severe subsampling by studying hyperparameter sensitivity.

3 Method

3.1 The EST Algorithm for Converting Events to Frames

The EST algorithm [13] automatically learns the mapping between the events and the frames from the data, making it adaptable to specific tasks. Suppose we have a set of events in a video denoted by $\mathcal{E} = \{e_i\}_{i=1}^N$, where $e_i = (x_i, y_i, t_i, p_i)$ contains the spatial position of the event (x_i, y_i), occurring time t_i, and the polarity of the event $p_i \in \{-1, +1\}$. The temporal dimension of the video is divided into C equally spaced timestamps $\{t^{(c)}\}_{c=1}^C$, and for each timestamp there are two frames one for each polarity, resulting in total $2C$ frames. The algorithm trains a multilayer perceptions (MLP) layer as a filter function $f_\theta(\cdot) : \mathbb{R} \to \mathbb{R}$ with learnable parameters θ to compute the frame representation V.

Particularly, the value at position (x, y) of frame c with polarity $p \in \{0, 1\}$ equals

$$V(p, c, x, y) = \sum_{i=1}^{N} t_i \, f_\theta \left(t_i - t^{(c)} \right) \mathcal{I}(x_i - x)\mathcal{I}(y_i - y)\mathcal{I}(p_i - p), \tag{1}$$

where $\mathcal{I}(\cdot) : \mathbb{R} \to \{0, 1\}$ is an indicator function which outputs 1 for zero input and 0 otherwise. After generating the $2C$ frames from N number of events, a CNN model performs the classification task, where the number of input channels for the CNN model equals the number of frames, i.e. $2C$.

3.2 Event Subsampling Procedure

For event subsampling, we randomly select from the events within a video. During training, in every epoch, we subsample a new subset of events for each video rather than fixing the subsampled events for each video during the entire training process. This approach helps to mitigate overfitting on the training set, which could occur if a fixed subset of events is used in each epoch. It's particularly important in scenarios with sparse input, where the subsampling rate is high. Additionally, we do not restrict the subsampling of events to specific spatial pixels or time intervals. Random subsampling provides better augmentation of the videos during training, improving the model's robustness to different subsets of events in a video. This provides better generalization to sparse inputs at test time, which might be crucial for real-time applications. By introducing different subsets of events to the network in every epoch, we try to minimize the impact of information loss and isolate the effect of sparsity on classification accuracy. The practical implications of this training approach are discussed in Section 5. During testing, the evaluation is conducted 20 times with different random subsamples for the test set, and the average accuracy is reported.

3.3 Training Procedure

To convert the events into frames, we adopt the settings from the original EST paper [13] regarding the number of frames. We choose $C = 9$ frames for each positive and negative polarity for any video, yielding a total of 18 channels for the input of the CNN classifier. As in Equation (1), pixels that do not correspond to any event's spatial location are assigned zero values. Thus, in this conversion method, higher subsampling rates of the events in the video lead to increased sparsity of the input frames for the CNN. Therefore, studying the effect of subsampling is similar to studying the impact of sparsity. Throughout the paper, we train a ResNet34 [20] as the CNN classifier, using the training set of the corresponding datasets. Before training, the weights are always initialized from a ResNet34 model pre-trained on ImageNet1k_v1. More details of the EST algorithm are in the supplementary material.

3.4 Event Classification Datasets

N-Caltech101 [28]: This dataset is derived from Caltech101 [9], which comprises 101 categories of images of various objects. To capture events from the images, [28] employed an ATIS event camera [30] mounted in front of an LCD monitor displaying the images. The camera was then moved in three directions to generate events. The camera motion pattern is identical for all classes, meaning that the temporal information derived from the camera's movement may not provide useful features for distinguishing between different object classes.

N-Cars [33]: This dataset captures real-world scenes by mounting an ATIS camera behind the windshield of a car and recording videos while driving in urban environments. It contains two classes of cars and background scenes, featuring various background scenarios. Unlike N-Caltech101, motion has a stronger correlations with the class labels.

N-ASL [3]: This dataset consists of handshape movements recorded by a DAVIS240c event camera, featuring 24 classes corresponding to 24 letters (excluding J and Z) from American Sign Language (ASL). The camera was stationary while subjects performed the handshape of each letter.

DVS-Gesture [1]: This dataset demonstrates 11 classes of hand and arm gestures from 29 subjects under 3 different lighting conditions. The gestures were performed in front of a DVS128 event camera against a stationary background. Some examples of gestures are hand waving, arm rotating, air guitar, and an “other" gesture invented by the subject. We used the Tonic library [25] for downloading and loading the dataset.

Fan1vs3: We introduce a new dataset to evaluate motion to better address the microsecond temporal resolution of event cameras. We placed a Prophesee Gen4 event camera [10] in front of a rotating fan, with the blades set to two different speeds, slow (level 1) vs. fast (level 3), corresponding to the two classes of the dataset. Distinguishing between the classes in this dataset relies more on the temporal details of the events rather than their spatial information, which makes it different than the previously mentioned datasets, where spatial information plays a more important role in classifying the objects. Further details of this toy dataset are provided in the supplementary material.

4 Experiments

4.1 Accuracy vs. Sparsity in Event Data Classification

We test the resilience of classification accuracy against reducing number of events per video across various datasets, including **N-Caltech101** [28], **N-ASL** [3], **N-Cars** [33], **DVS-Gesture** [1], and the proposed **Fan1vs3**. We use Adam [21]

Table 1: CNN classification accuracy for decreasing the number of events per video across various datasets. We can reduce the number of events per video by an order of magnitude without sacrificing considerable accuracy. Even under extreme sparsity (8 or 16 total events per video with 18 frames) the accuracies remain significantly above chance level.

			# events per video							
Dataset	# classes		8	16	32	64	512	1024	4096	25000
N-ASL	24	Test Acc. (%)	24.33	91.44	97.99	99.15	99.80	99.79	99.81	99.76
		Std. Dev. (%)	±28.52	±0.37	±0.23	±0.08	±0.01	±0.09	±0.15	±0.28
		p-value	0	0	0	0	0	0	0	0
N-Cars	2	Test Acc. (%)	72.51	78.74	83.23	86.58	92.97	93.23	92.46	91.87
		Std. Dev. (%)	±0.18	±0.12	±0.17	±0.16	±0.13	±0.20	±0.95	±0.81
		p-value	0	0	0	0	0	0	0	0
DVS-Gesture	11	Test Acc. (%)	47.98	55.99	75.23	84.81	93.37	94.69	95.18	95.33
		Std. Dev. (%)	±0.21	±0.23	±2.21	±0.34	±0.65	±0.30	±0.49	±0.52
		p-value	1.08e-205	3.25e-282	0	0	0	0	0	0
Fan1vs3	2	Test Acc. (%)	53.35	54.17	56.62	58.53	75.29	94.00	98.24	99.40
		Std. Dev. (%)	±0.56	±1.77	±0.42	±0.99	±1.19	±5.69	±0.79	±0.51
		p-value	3.67e-01	2.86e-01	1.54e-01	1.06e-01	9.76e-06	7.48e-17	1.02e-20	2.61e-22
N-Caltech101	101	Test Acc. (%)	25.20	31.37	39.00	46.19	67.85	74.06	82.87	88.62
		Std. Dev. (%)	±0.17	±0.15	±0.21	±0.19	±0.42	±0.54	±0.47	±0.51
		p-value	0	0	0	0	0	0	0	0

for optimization with the 'Reduce on Plateau' learning rate scheduler for all experiments. The reduction factor is set at 2, and the patience parameter is determined based on the number of epochs, with values of 20 for 100 epochs, 40 for 250 epochs, and 75 for 500 epochs. The batch size, learning rate, weight decay coefficient, and number of epochs are chosen to be the same across all sparsity levels of the same dataset. For each dataset, we picked a set of hyperparameters which could converge to a high accuracy at every sparsity level based on a small preliminary experiments, except N-Cars, for which we adopt the values from the EST paper [13]. For the exact values of the hyperparameters, please see the supplementary material. For the test set, the evaluation is conducted 20 times with different random subsamples, and the average accuracy is reported.

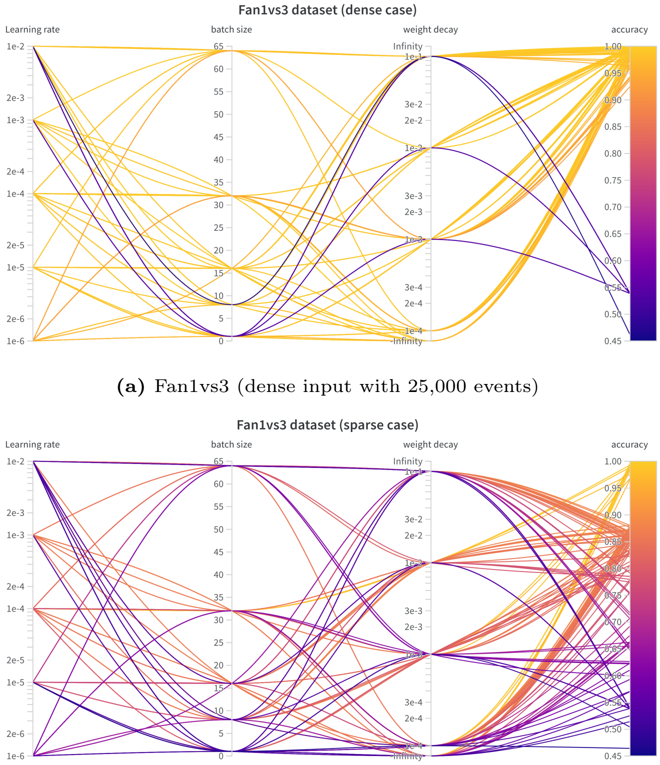
In Tab. 1, we present the classification accuracy of the EST algorithm with the ResNet34 model with varying numbers of subsampling for different datasets. Each entry represents the average accuracy over 5 independent runs with different seeds (in black), and their standard deviation (in red). The results demonstrate a decrease in accuracy as the number of events per video decreases, which is not surprising due to the loss of information incurred by removing events. However, it's notable that the network maintains accuracy levels close to those achieved with denser cases for even unexpectedly small numbers of events per video. For instance, in the N-ASL dataset, even with only 64 events per video, distributed over 18 frames, the accuracy remains at 99 percent. Similarly, across other datasets, we observe that the CNN can still perform classification above chance level while reducing total events per video considerably. To show that the accuracies significantly exceed the corresponding chance levels, we compute the

p-value for each mean accuracy in Tab. 1 using the one-tailed binomial test (in blue). For all datasets, except Fan1vs3, the p-values are almost zero for different subsampling levels. Thus, even with just 8 or 16 events per video, distributed over 18 frames, the accuracies are significantly higher than chance level. Here, the Fan1vs3 dataset serves as an extreme case where CNNs struggle with the high sparsity of the input. Because the classes are distinguished by the fan speed, the temporal details of the events play a more crucial role than the spatial details. Our procedure of randomly subsampling the events then makes it challenging for the network to accurately classify speeds, especially as the number of events is decreased. This highlights the subsampling sensitivity in applications where temporal details are more important because as we reduce the number of events, temporal information is lost faster than spatial information. For better visualization, the accuracy curves from Tab. 1, along with their standard deviations, are presented in the supplementary material.

4.2 Hyperparameter Sensitivity with Increasing Sparsity

In this subsection, we study the effect of subsampling on the *training procedure of a CNN model*. While experiments of the previous subsection have shown that CNNs can maintain high accuracies even with sparse input data, it is essential to recognize that sparse input in CNNs introduces new challenges. As we sample less events per video, we inevitably lose information from the removed events, and the trivial consequence is a drop in classification accuracy. However, in addition to this effect, we find that the sensitivity of training to hyperparameters may also increase, making the training more challenging. To investigate it further, we compare the distributions of classification accuracy for different optimization hyperparameter sets in two scenarios: 1. with sparse input and 2. with dense input.

In our analysis, we focus on four datasets — Fan1vs3, DVS-Gesture, N-Caltech101, and N-ASL — to analyze how varying levels of subsampling affect the training process and sensitivity to optimization hyperparameters. We select three optimization hyperparameters (HPs) to tune: learning rate, batch size, and weight decay. To cover a wide range of possibilities, we choose learning rates from the set $\{10^{-i}\}_{i=2}^{6}$, ensuring that the range is large enough to have performance drops at both the upper and lower bounds. Weight decay values are selected from the set $\{0\} \cup \{10^{-i}\}_{i=1}^{4}$. For batch sizes, we consider $\{1, 8, 16, 32, 64, 128\}$ for all datasets except Fan1vs3, where we exclude a batch size of 128 due to GPU memory limitations. We determine the number of epochs such that the loss curves reach convergence. Specifically, we choose 250 epochs for Fan1vs3, DVS-Gesture, and N-Caltech101, and 100 epochs for N-ASL. Other HPs, such as the optimizer (Adam), learning rate scheduler, reduce factor, patience, and number of epochs remain consistent with those used in Sec. 4.1. We randomly select 50 sets of HPs from the specified settings. For each set, we conduct training with 3 different seeds for both sparse and dense cases, resulting in a total of 300 different training instances per dataset. In the dense case, we choose 25,000 events per video, while for the sparse case, we select 1,024 events per video



(a) Fan1vs3 (dense input with 25,000 events)

(b) Fan1vs3 (sparse input with 1,024 events)

Fig. 2: Parallel coordinate plots showing HP tuning results for Fan1vs3 dataset using dense and sparse inputs. HPs: learning rate, batch size, and weight decay. In the dense setting, we observe test accuracies concentrated near the maximum accuracy, while in the sparse setting, we observe a small number of runs achieving the maximum accuracy. The plots are from the Weight and Biases website [4].

for Fan1vs3, 64 for DVS-Gesture, 512 for N-Caltech101, and 8 for N-ASL. The dataset-specific choice of the number of events per video in sparse case is based on the sparsity levels where we started observing instability in training for each dataset. For example, for N-ASL, the number of events per video had to be decreased to below 16 to observe a substantial drop in accuracy.

Figure 3 illustrates histograms of accuracies for all 300 experiments with different HP sets. For datasets except N-Caltech101, we observe a notable distinction between the dense and sparse cases. In the dense case, runs for different HP sets tend to cluster near the highest achievable accuracy, while in the sparse case, we observe a noticeable gap between the main cluster of runs and those achieving the highest accuracy.

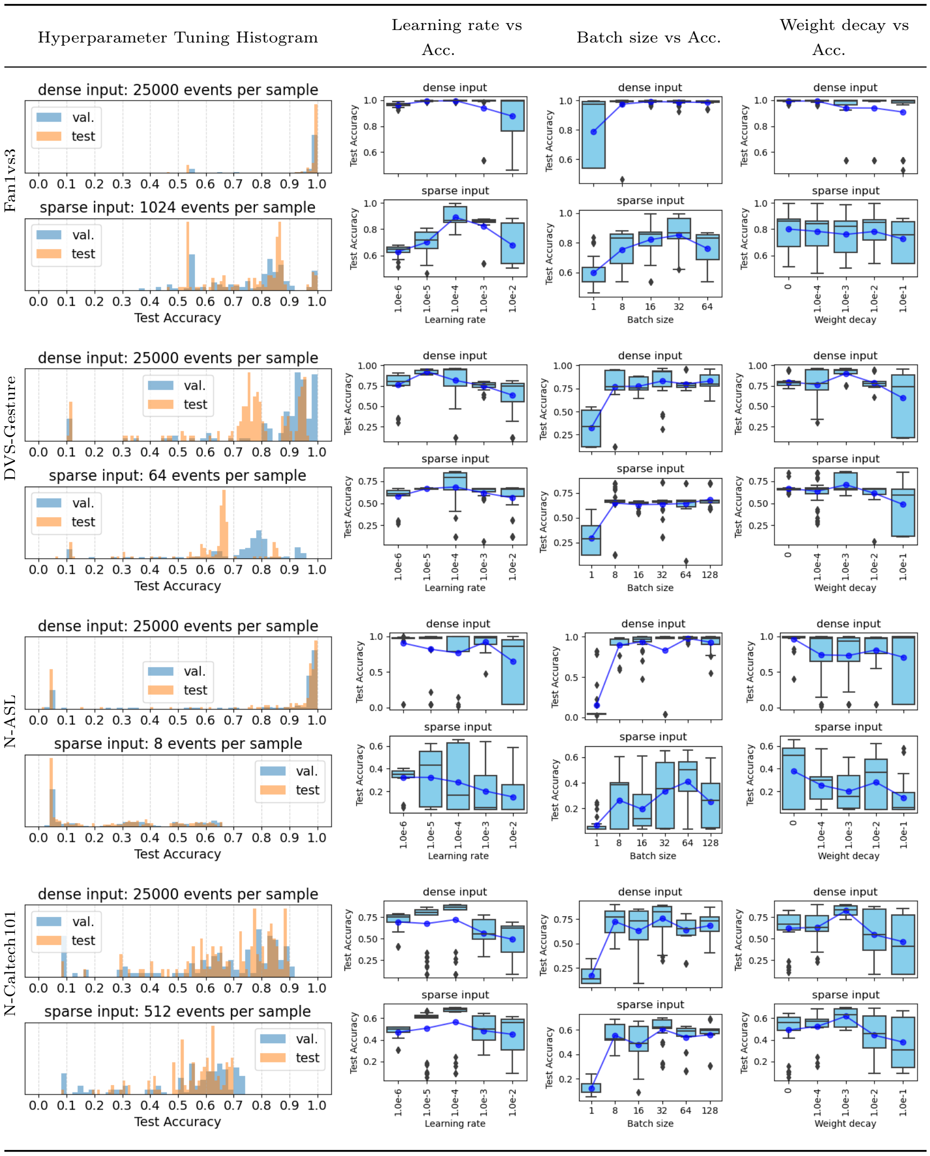


Fig. 3: Histograms of all 300 accuracies attained during hyperparameter tuning for both dense and sparse input scenarios (first column), and boxplots of test accuracies obtained using each individual hyperparameter: learning rate, batch size, and weight decay (second to fourth columns). The blue curve represents the mean test accuracies.

Figure 3 also shows the dependencies of test accuracies to each HP individually. Each box in the boxplot figures of the second to fourth columns shows the test accuracies corresponding to all the experiments with specific values of that individual hyperparameter Notably, in the sparse case, sensitivity to individual HPs is higher compared to the dense case. Among the HPs, the variation of learning rate more significantly affects the test accuracies. The behavior of the N-Caltech101 dataset differs from that of other datasets, suggesting that sensitivity to HPs is not a universal behavior in sparse input cases. Nevertheless, we argue that such sensitivity may arise in some datasets, and propose to consider extensive hyperparameter tuning during model development in sparser scenarios. In the N-Caltech101 dataset, it appears that the loss of information plays a more significant role in the accuracy drop than sensitivity to HPs.

Figure 2 presents parallel coordinate plots of hyperparameter (HP) tuning for the Fan1vs3 dataset using dense and sparse inputs from the Weight and Biases [4] website. Each string represents a set of HPs, and we observe that in the dense case, the concentration of test accuracies is near the peak. Conversely, in the sparse case, only a small number of HP sets manage to achieve accuracy levels close to the peak. This suggests the increased importance of HP tuning during training in sparse cases to ensure that performance reaches its maximum potential.

To quantify the level of sensitivity to HPs, we propose a *hyperparameter sensitivity metric* based on the clustering of accuracies for different HP sets. First, we employ the K-means algorithm to cluster the accuracies into K clusters. Next, we compute the distance between the centers of the most populated cluster (i.e., the cluster with the most runs) and the cluster with the maximum accuracy. To account for the different maximum accuracies in dense and sparse cases, we normalize the obtained distance by dividing it by the maximum achieved accuracy. In scenarios where the number of runs concentrated near the maximum accuracies is low, such as in the Fan1vs3 dataset, we need to select a larger number of K to prevent the high-accuracy cluster from merging with other clusters incorrectly. For this, we compute the normalized metric for a range of K values, i.e. from 2 to 10. Then, we select the maximum value over different K values in search of at least one popular cluster falling far apart from the high-accuracy cluster. Table 2 displays the HP sensitivity metric, as well as the mean and maximum test accuracy and the percentage of improvement observed from mean to maximum accuracy for each dataset for dense and sparse cases. In most datasets, excluding N-Caltech101, the metric indicates a higher sensitivity to HPs when the input is sparse, and the improvement from mean to maximum accuracy is more substantial in the sparse case.

4.3 Gradient Diversity in the Sparse vs. Dense Case

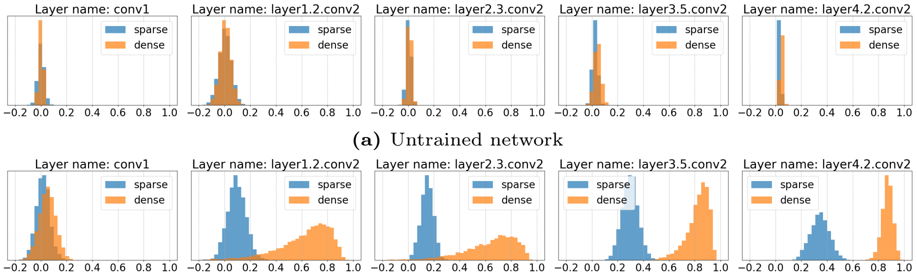
In this subsection, we examine how the subsampling level of an event video affects the gradients in the network. We first randomly select a video from the training set of the Fan1vs3 dataset and generate $M = 100$ different subsamples from it for both the sparse (1,024 events) and the dense (25,000 events) case.

Table 2: Hyperparameter (HP) sensitivity metric along with mean and maximum test accuracies for dense and sparse cases for different datasets. Higher values for the metric ↑ indicate a higher sensitivity to HPs. For most datasets, the HP sensitivity metric as well as the improvement from mean to maximum test accuracy is larger in the sparse case (in bold) compared to the dense case.

Dataset	N-ASL		Fan1vs3		DVS-Gesture		N-Caltech101	
	dense	sparse	dense	sparse	dense	sparse	dense	sparse
# events	25000	8	25000	1024	25000	64	25000	512
Hyperparameter sensitivity metric								
val. metric	0.000	**0.915**	0.000	**0.471**	0.060	**0.191**	**0.122**	0.098
test metric	0.021	**0.887**	0.000	**0.145**	0.196	**0.230**	**0.279**	0.101
Mean and maximum test acc.								
mean test acc. (%)	80.74	25.09	95.14	76.62	75.22	61.50	69.23	55.41
max. test acc. (%)	99.95	65.57	100.00	99.62	96.37	86.34	89.55	69.88
max. to mean improvement (%)	23.79	**161.34**	5.10	**30.01**	28.13	**40.38**	**29.34**	26.11

For each subsample, we compute the loss and obtain the gradient of the loss with respect to the weights of $L = 5$ different layers, corresponding to the last convolutional layer of each block in the ResNet34 network. We pick these layers to study the gradients at different depths of the network. The specific layers chosen for gradient computation are detailed in the supplementary material. For subsample $i \in \{1, \ldots, M\}$ and layer $l \in \{1, \ldots, L\}$, we denote the gradient vector as $v_i^{(l)} \in \mathbb{R}^{p_l}$, where p_l is the number of learnable weights in layer l. Then, we compute the pairwise cosine similarity between $v_i^{(l)}$ and $v_j^{(l)}$ for all unique pairs in $\{(i, j) | 1 \leq i < j \leq M\}$ for each layer. The total number of cosine similarities for each layer equals $\frac{M\,(M-1)}{2} = 4950$. To demonstrate the diversity of the gradients, we illustrate the histogram of the cosine similarities separately for each layer for both sparse and dense cases.

Figure 4 displays the histograms of cosine similarity between unique pairs of gradient vectors. We show histograms for both untrained and fully trained networks. Values closer to 1 in the histograms show greater alignment among the gradients, while values close to 0 show higher gradient diversity. At the beginning of training, when the network is untrained, the gradients are diverse for both sparse and dense cases. For the trained network, we observe that gradients are more aligned in the dense case compared to the sparse case, which is in line with the results of Sec. 4.2 about the difficulty of training networks with sparse inputs. Higher gradient diversity suggests that the network may not travel far from the initial weights during gradient descent, instead taking a random walk around the initialization point. Conversely, greater consensus among the gradients of different subsamples during training, i.e. closer values to 1, contributes to faster convergence of the network towards final weight values. Furthermore, as we progress to deeper layers of the trained network, gradients tend to be more aligned for both sparse and dense inputs. This can be attributed to the larger receptive field in deeper layers, allowing the network to develop a global



(a) Untrained network

(b) Trained network

Fig. 4: Histogram of cosine similarities for both dense and sparse inputs for the (a) untrained network and (b) trained network. Values close to 1 indicate greater alignment among gradients. For the trained network, gradients are more aligned in the dense case, and as we progress to deeper layers (from left to right), the alignments increase for both sparse and dense cases.

understanding of the input video, thereby reducing the influence of different subsampling on these layers.

5 Discussion and Conclusion

In this paper, we study the effect of subsampling on the sparsity vs. accuracy trade-off in event video classification using a CNN model. Our findings reveal that accuracy can remain high even under severe sparsity in many datasets. However, special attention must be given to the training challenges arising from hyperparameter (HP) sensitivity in sparse input regimes.

Practical Implications for the Subsampling Procedure The training procedure described in Sec. 3.2, where a new subset of events is selected in each epoch, may not be feasible in all scenarios. It is important to note that when dense data is available, this subsampling procedure can help mitigate overfitting. In practice, data collection for training occurs less frequently than in the utilization phase of the model, and in the former case, we can leverage higher sampling rates using capturing devices with greater bandwidth or computational capacity. These dense datasets can then be used to train the models following our procedure, helping the model maintain higher accuracies for sparse event input at test time.

Limitations It is important to note that keeping accuracies high with sparse input is not universal across all datasets. As a specific example, we observe that in the N-Caltech101 dataset, accuracies drop faster with increased subsampling, and, unlike in other datasets, hyperparameter tuning does not help much in the sparse case compared to the dense case. This dataset is different from others we study because it is derived from static images captured by moving the camera, unlike the others, which are recordings of real-world scenes. The high sensitivity to sampling level in N-Caltech101 likely stems from its greater reliance on spatial information rather than temporal information.

The results of this paper rely on the subsampling procedure of selecting random subsets of events at each epoch. However, dense event data may not always be available to subsample from during the training process. Therefore, maintaining high accuracies may not be achievable, and the model is prone to overfitting if the training set itself is sparse, and limited in sample size.

Future Work We limited our analysis to classification using CNNs as a proof-of-concept. However, our training methodology simply allows for increasing the sparsity in favor of reducing the bandwidth at train and test time, which might be a viable training procedure for other popular event-based vision models, including transformers and graph neural networks (GNNs). Similarly, looking at the gradient analysis, the instabilities seem to emerge, not due to the nature of the CNNs or the classification task, but simply due to the variability of information in the training set in the sparse setting. This suggests that our results could potentially be generalized to other models and vision tasks, making it a future direction for empirical testing. Additionally, adapting methods such as [36], which aim to mitigate the impact of input sparsity in conventional images, can be an interesting line of research to improve the robustness of CNNs to sparse event data.

References

1. Amir, A., Taba, B., Berg, D., Melano, T., McKinstry, J., Di Nolfo, C., Nayak, T., Andreopoulos, A., Garreau, G., Mendoza, M., Kusnitz, J., Debole, M., Esser, S., Delbruck, T., Flickner, M., Modha, D.: A Low Power, Fully Event-Based Gesture Recognition System. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7388–7397 (Jul 2017). https://doi.org/10.1109/CVPR.2017.781, https://ieeexplore.ieee.org/document/8100264, iSSN: 1063-6919
2. Andersen, K.F., Pham, H.X., Ugurlu, H.I., Kayacan, E.: Event-based Navigation for Autonomous Drone Racing with Sparse Gated Recurrent Network (Apr 2022), http://arxiv.org/abs/2204.02120, arXiv:2204.02120 [cs]
3. Bi, Y., Chadha, A., Abbas, A., Bourtsoulatze, E., Andreopoulos, Y.: Graph-Based Object Classification for Neuromorphic Vision Sensing. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 491–501. IEEE, Seoul, Korea (South) (Oct 2019). https://doi.org/10.1109/ICCV.2019.00058, https://ieeexplore.ieee.org/document/9010397/
4. Biewald, L.: Experiment tracking with weights and biases (2020), https://www.wandb.com/, software available from wandb.com
5. Censi, A., Mueller, E., Frazzoli, E., Soatto, S.: 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). pp. 3319–3326. IEEE, Seattle, WA, USA (May 2015).

https://doi.org/10.1109/ICRA.2015.7139657, http://ieeexplore.ieee.org/document/7139657/

6. Cohen, G., Afshar, S., Orchard, G., Tapson, J., Benosman, R., van Schaik, A.: Spatial and Temporal Downsampling in Event-Based Visual Classification. IEEE Transactions on Neural Networks and Learning Systems **29**(10), 5030–5044 (Oct 2018). https://doi.org/10.1109/TNNLS.2017.2785272, https://ieeexplore.ieee.org/document/8260859, conference Name: IEEE Transactions on Neural Networks and Learning Systems
7. Delbruck, T., Graca, R., Paluch, M.: Feedback control of event cameras. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 1324–1332. IEEE, Nashville, TN, USA (Jun 2021). https://doi.org/10.1109/CVPRW53098.2021.00146, https://ieeexplore.ieee.org/document/9522899/
8. Falanga, D., Kleber, K., Scaramuzza, D.: Dynamic obstacle avoidance for quadrotors with event cameras. Science Robotics **5**(40), eaaz9712 (2020)
9. Fei-Fei, L., Fergus, R., Perona, P.: 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. pp. 178–178. IEEE (2004)
10. Finateu, T., Niwa, A., Matolin, D., Tsuchimoto, K., Mascheroni, A., Reynaud, E., Mostafalu, P., Brady, F., Chotard, L., LeGoff, F., Takahashi, H., Wakabayashi, H., Oike, Y., Posch, C.: 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 (Feb 2020)
11. Fischer, T., Milford, M.: How Many Events Do You Need? Event-Based Visual Place Recognition Using Sparse But Varying Pixels. IEEE Robotics and Automation Letters **7**(4), 12275–12282 (Oct 2022). https://doi.org/10.1109/LRA.2022.3216226, conference Name: IEEE Robotics and Automation Letters
12. Gallego, G., Delbruck, T., Orchard, G., Bartolozzi, C., Taba, B., Censi, A., Leutenegger, S., Davison, A.J., Conradt, J., Daniilidis, K., Scaramuzza, D.: Event-Based Vision: A Survey. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**(1), 154–180 (Jan 2022). https://doi.org/10.1109/TPAMI.2020.3008413, https://ieeexplore.ieee.org/document/9138762/
13. Gehrig, D., Loquercio, A., Derpanis, K., Scaramuzza, D.: End-to-End Learning of Representations for Asynchronous Event-Based Data. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 5632–5642. IEEE, Seoul, Korea (South) (Oct 2019). https://doi.org/10.1109/ICCV.2019.00573, https://ieeexplore.ieee.org/document/9009469/
14. Gehrig, D., Scaramuzza, D.: Are High-Resolution Event Cameras Really Needed? (Mar 2022), http://arxiv.org/abs/2203.14672, arXiv:2203.14672 [cs]
15. Gehrig, D., Scaramuzza, D.: Pushing the limits of asynchronous graph-based object detection with event cameras. arXiv preprint arXiv:2211.12324 (2022)
16. Gruel, A., Martinet, J., Linares-Barranco, B., Serrano-Gotarredona, T.: Performance comparison of DVS data spatial downscaling methods using Spiking Neural Networks. In: 2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). pp. 6483–6491. IEEE, Waikoloa, HI, USA (Jan 2023). https://doi.org/10.1109/WACV56688.2023.00643, https://ieeexplore.ieee.org/document/10030433/

17. Gruel, A., Carreras, L.T., Bueno García, M., Kupczyk, E., Martinet, J.: Frugal event data: how small is too small? A human performance assessment with shrinking data. In: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 4093–4100. IEEE, Vancouver, BC, Canada (Jun 2023). https://doi.org/10.1109/CVPRW59228.2023.00430, https://ieeexplore.ieee.org/document/10208584/
18. Gruel, A., Martinet, J., Serrano-Gotarredona, T., Linares-Barranco, B.: 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. pp. 245–253. SCITEPRESS - Science and Technology Publications, Online Streaming, — Select a Country — (2022). https://doi.org/10.5220/0010991900003124, https://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0010991900003124
19. Hanover, D., Loquercio, A., Bauersfeld, L., Romero, A., Penicka, R., Song, Y., Cioffi, G., Kaufmann, E., Scaramuzza, D.: Autonomous Drone Racing: A Survey (Jan 2023). https://doi.org/10.48550/arXiv.2301.01755, http://arxiv.org/abs/2301.01755, arXiv:2301.01755 [cs]
20. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition (Dec 2015). https://doi.org/10.48550/arXiv.1512.03385, http://arxiv.org/abs/1512.03385, arXiv:1512.03385 [cs]
21. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization (Jan 2017), http://arxiv.org/abs/1412.6980, arXiv:1412.6980 [cs]
22. Klenk, S., Bonello, D., Koestler, L., Cremers, D.: Masked Event Modeling: Self-Supervised Pretraining for Event Cameras (Dec 2022), http://arxiv.org/abs/2212.10368, arXiv:2212.10368 [cs]
23. Lagorce, X., Orchard, G., Galluppi, F., Shi, B.E., Benosman, R.B.: HOTS: A Hierarchy of Event-Based Time-Surfaces for Pattern Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**(7), 1346–1359 (Jul 2017). https://doi.org/10.1109/TPAMI.2016.2574707, conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence
24. Lee, J.H., Delbruck, T., Pfeiffer, M.: Training Deep Spiking Neural Networks Using Backpropagation. Frontiers in Neuroscience **10** (Nov 2016). https://doi.org/10.3389/fnins.2016.00508, https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2016.00508/full, publisher: Frontiers
25. Lenz, G., Chaney, K., Shrestha, S.B., Oubari, O., Picaud, S., Zarrella, G.: Tonic: event-based datasets and transformations. (Jul 2021). https://doi.org/10.5281/zenodo.5079802, https://zenodo.org/records/5079802
26. Maqueda, A.I., Loquercio, A., Gallego, G., Garcia, N., Scaramuzza, D.: Event-Based Vision Meets Deep Learning on Steering Prediction for Self-Driving Cars. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5419–5427. IEEE, Salt Lake City, UT (Jun 2018). https://doi.org/10.1109/CVPR.2018.00568, https://ieeexplore.ieee.org/document/8578666/
27. Messikommer, N., Gehrig, D., Loquercio, A., Scaramuzza, D.: Event-Based Asynchronous Sparse Convolutional Networks. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M. (eds.) Computer Vision – ECCV 2020, vol. 12353, pp. 415–431. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-58598-3_25, https://link.springer.com/10.1007/978-3-030-58598-3_25, series Title: Lecture Notes in Computer Science
28. Orchard, G., Jayawant, A., Cohen, G.K., Thakor, N.: Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades. Frontiers in Neu-

roscience **9** (2015), https://www.frontiersin.org/articles/10.3389/fnins.2015.00437

29. Orchard, G., Meyer, C., Etienne-Cummings, R., Posch, C., Thakor, N., Benosman, R.: HFirst: A Temporal Approach to Object Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **37**(10), 2028–2040 (Oct 2015). https://doi.org/10.1109/TPAMI.2015.2392947, http://arxiv.org/abs/1508.01176, arXiv:1508.01176 [cs]
30. Posch, C., Matolin, D., Wohlgenannt, R.: 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)
31. Sekikawa, Y., Hara, K., Saito, H.: EventNet: Asynchronous Recursive Event Processing. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3882–3891. IEEE, Long Beach, CA, USA (Jun 2019). https://doi.org/10.1109/CVPR.2019.00401, https://ieeexplore.ieee.org/document/8954313/
32. Shrestha, S.B., Orchard, G.: SLAYER: Spike Layer Error Reassignment in Time. In: Advances in Neural Information Processing Systems. vol. 31. Curran Associates, Inc. (2018), https://proceedings.neurips.cc/paper_files/paper/2018/hash/82f2b308c3b01637c607ce05f52a2fed-Abstract.html
33. Sironi, A., Brambilla, M., Bourdis, N., Lagorce, X., Benosman, R.: HATS: Histograms of Averaged Time Surfaces for Robust Event-Based Object Classification. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1731–1740. IEEE, Salt Lake City, UT, USA (Jun 2018). https://doi.org/10.1109/CVPR.2018.00186, https://ieeexplore.ieee.org/document/8578284/
34. Suh, Y., Choi, S., Ito, M., Kim, J., Lee, Y., Seo, J., Jung, H., Yeo, D.H., Namgung, S., Bong, J., Yoo, S., Shin, S.H., Kwon, D., Kang, P., Kim, S., Na, H., Hwang, K., Shin, C., Kim, J.S., Park, P.K.J., Kim, J., Ryu, H., Park, Y.: A 1280×960 dynamic vision sensor with a 4.95-μm pixel pitch and motion artifact minimization. In: 2020 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE (Oct 2020)
35. Tapson, J.C., Cohen, G.K., Afshar, S., Stiefel, K.M., Buskila, Y., Wang, R.M., Hamilton, T.J., van Schaik, A.: Synthesis of neural networks for spatio-temporal spike pattern recognition and processing. Frontiers in neuroscience **7**, 153 (2013)
36. Uhrig, J., Schneider, N., Schneider, L., Franke, U., Brox, T., Geiger, A.: Sparsity Invariant CNNs (Aug 2017), http://arxiv.org/abs/1708.06500, arXiv:1708.06500 [cs]
37. Vicente-Sola, A., Manna, D.L., Kirkland, P., Di Caterina, G., Bihl, T.: Spiking Neural Networks for event-based action recognition: A new task to understand their advantage (Aug 2023). https://doi.org/10.48550/arXiv.2209.14915, http://arxiv.org/abs/2209.14915, arXiv:2209.14915 [cs]
38. Vidal, A.R., Rebecq, H., Horstschaefer, T., Scaramuzza, D.: Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High-Speed Scenarios. IEEE Robotics and Automation Letters **3**(2), 994–1001 (Apr 2018). https://doi.org/10.1109/LRA.2018.2793357, conference Name: IEEE Robotics and Automation Letters
39. Xie, B., Deng, Y., Shao, Z., Liu, H., Xu, Q., Li, Y.: Event Voxel Set Transformer for Spatiotemporal Representation Learning on Event Streams (May 2023), http://arxiv.org/abs/2303.03856, arXiv:2303.03856 [cs]
40. Zheng, X., Liu, Y., Lu, Y., Hua, T., Pan, T., Zhang, W., Tao, D., Wang, L.: Deep Learning for Event-based Vision: A Comprehensive Survey and Benchmarks (Feb 2023), http://arxiv.org/abs/2302.08890, arXiv:2302.08890 [cs]

41. Zhou, J., Zheng, X., Lyu, Y., Wang, L.: E-CLIP: Towards Label-efficient Event-based Open-world Understanding by CLIP (Sep 2023), http://arxiv.org/abs/2308.03135, arXiv:2308.03135 [cs]
42. Zhu, A.Z., Yuan, L., Chaney, K., Daniilidis, K.: Unsupervised Event-Based Learning of Optical Flow, Depth, and Egomotion. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 989–997. IEEE, Long Beach, CA, USA (Jun 2019). https://doi.org/10.1109/CVPR.2019.00108, https://ieeexplore.ieee.org/document/8953979/

Supplementary Material for the Paper: Pushing the boundaries of event subsampling in event-based video classification using CNNs

Hesam Araghi, Jan van Gemert, and Nergis Tomen

Computer Vision Lab, Delft University of Technology
{h.araghi, j.c.vangemert, n.tomen}@tudelft.nl

1 Details of Training Procedures

For all experiments, we use Adam [6] for optimization. The learning scheduler is the 'Reduce on Plateau' with a reduction factor of 2, and it monitors the validation accuracy. The cross-entropy loss function is used for the classification. During testing, the evaluation is conducted 20 times with different random subsamples for the test set, and the average accuracy is reported. We use PyTorch and PyTorch Lightning libraries for the training procedure and use Weights and Biases for logging the results [2].

1.1 EST Algorithm [4] Details

We use the implementation code provided in the original paper's GitHub Repository. [1] For all experiments, we choose $C = 9$ frames (addressed as the number of bins in the original paper) for each positive and negative polarity, yielding a total of 18 channels. As the CNN classifier, we use a ResNet34 [5] model pretrained on ImageNet1k_v1, modifying the number of input channels from 3 to 18. For the filter function $f_\theta(\cdot) : \mathbb{R} \to \mathbb{R}$, we employed an MLP model with two hidden layers, each containing 30 nodes, and used Leaky ReLU with a negative slope of 0.1 as the activation function.

1.2 Details for 'Accuracy vs. Sparsity in Event Data Classification' Experiment

Table 1 lists the hyperparameters used for the experiment 'Accuracy vs. Sparsity in Event Data Classification'. The hyperparameters are chosen to be the same across all sparsities of the same dataset. For each dataset, we picked a set of hyperparameters which could converge to a high accuracy at every sparsity level based on limited preliminary experiments. For N-Cars, we adopt the values from the EST paper [4].

[1] https://github.com/uzh-rpg/rpg_event_representation_learning

Table 1: Hyperparameters used for 'Accuracy vs. Sparsity in Event Data Classification' experiment

Hyperparameters	**N-ASL**	**N-Caltech101**	**N-Cars**	**DVS-Gesture**	**Fan1vs3**
Learning Rate	1e-4	1e-4	1e-5	1e-4	1e-4
Patients for Learning Scheduler	20	40	75	40	40
Weight Decay	0	0	0	0	0
Number of Epochs	100	250	500	250	250
Batch Size	32	16	100	64	32

Table 2: Characteristics for different event classification datasets used in this paper

Characteristics	**N-ASL**	**N-Caltech101**	**N-Cars**	**DVS-Gesture**	**Fan1vs3**
# classes	24	101	2	11	2
# videos	100,800	8,709	24,029	1,342	510
spatial resolution	[240,180]	[240,180]	[100,120]	[128,128]	[1280,720]
separate test set	✗	✗	✓	✓	✗

2 Accuracy curves for Table 1 of the paper

For better visualization of the accuracy behavior for the number of events per video, the accuracy curves for different datasets are depicted in Fig. 1. Except for the N-Caltech101 dataset, the accuracies remain relatively close to the dense case (rightmost points). The curves remain nearly flat until they reach a point where the accuracy drops significantly.

3 Details of Event Classification Datasets

Table 2 provides the characteristics of the different datasets we used for event classification in this paper. For splitting the data into training, validation, and test sets, we used the following approach: for datasets that do not come with a predefined test set, we randomly divided the videos of each class into 75% for training, 10% for validation, and 15% for testing. For datasets that already include a test set, we randomly divided the training set videos into 85% for training and 15% for validation.

For the DVS-Gesture dataset [1], we followed a procedure similar to [8]. We used the Tonic library [7] to download and load the dataset. The videos in the dataset were segmented by time into videos of length 1.7 seconds, without any overlap.

Fan1vs3 dataset We introduce a new toy dataset designed to emphasize the temporal details captured by event cameras. To create this dataset, we placed a fan in front of a Prophesee Gen4 event camera [3]. The fan blades were set to two

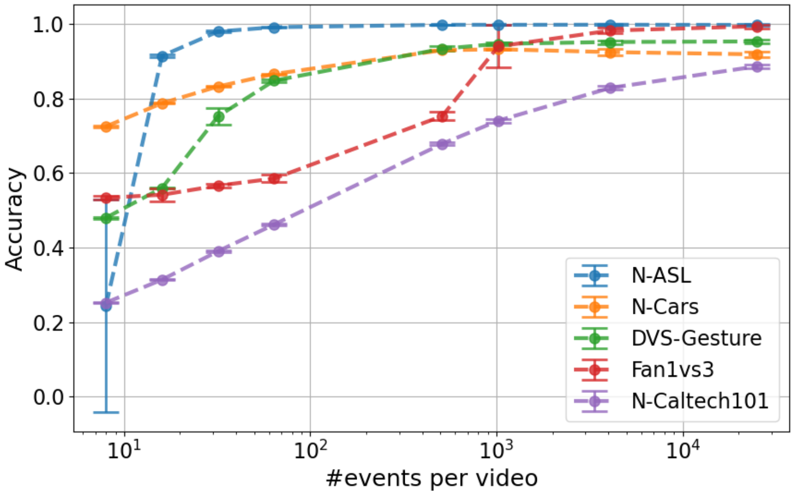


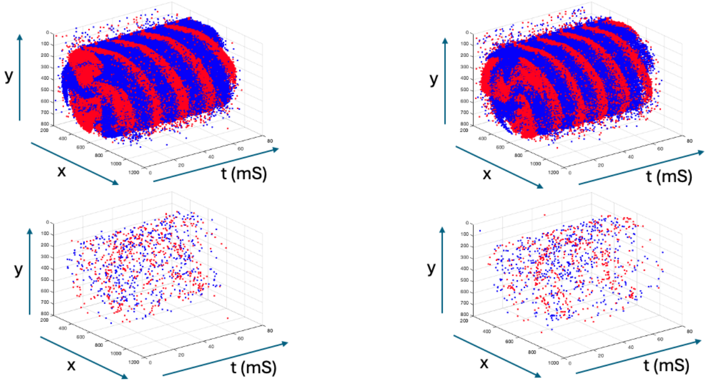
Fig. 1: Classification accuracy curves as the number of events per video decreases across various datasets. The error bars represent the standard deviation of accuracies across different runs. For many datasets, the accuracy curves do not significantly drop compared to the dense input case.

different speeds: the slowest (speed 1) and the fastest (speed 3), corresponding to the two classes of the dataset.

The videos for each class are segmented into 75-millisecond clips. The 'speed 1' class contains 235 videos, while the 'speed 3' class contains 275 videos. Figure 2 displays the events of an example video for each speed class in spatiotemporal space, along with the subsampled version of each, reduced to 1024 events. This dataset allows us to investigate the classification with an emphasis on the temporal data.

4 Details for 'Gradient Diversity in the Sparse vs. Dense Case' Experiment

For computing the gradients in the experiment 'Gradient Diversity in the Sparse vs. Dense Case', we consider five different layers, specifically the last convolutional layer of each block in the ResNet34 network. The specific names of these convolutional layers along with the number of parameters for each layer are provided in Tab. 3. Figure 3 also illustrates the position of the selected layer in the network architecture.



(a) Speed 1

(b) Speed 3

Fig. 2: Event illustration of a video for each speed class of Fan1vs3 dataset in spatiotemporal space. The second row displays the subsampled events, reduced to 1024 events.

Table 3: Layer names and the number of parameters in each layer used in gradient analysis of 'Gradient Diversity in the Sparse vs. Dense Case' Experiment

Layer name	# parameters
`conv1`	9,408
`layer1.2.conv2`	36,864
`layer2.3.conv2`	147,456
`layer3.5.conv2`	589,824
`layer4.2.conv2`	2,359,296

5 Details of Compute Resources

For the computing resources, we used a cluster of machines with GPUs, which it will be referenced later. We used one GPU per training execution. The GPUs used for the experiments were primarily NVIDIA A40 or NVIDIA V100 models. Each job requested 16 GB of RAM and 4 CPU cores. The specific computational times for the experiments in the 'Hyperparameter Sensitivity with Increasing Sparsity' subsection, which involved evaluating 300 hyperparameter sets per dataset, are detailed in Tab. 4. The computational times are reported from the Weight and Biases [2] website used for logging the results.

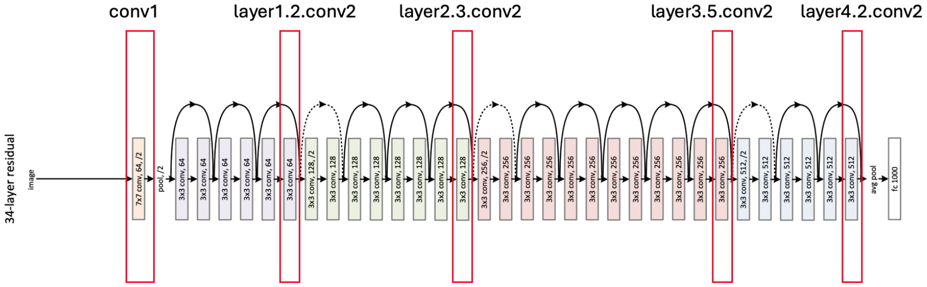


Fig. 3: Layer of the ResNet34 network which is used in gradient analysis of the 'Gradient Diversity in the Sparse vs. Dense Case' Experiment. The background image is adapted from the original ResNet paper [5].

Table 4: GPU computation time on the cluster in GPU days for the hyperparameter tuning used in the 'Hyperparameter Sensitivity with Increasing Sparsity' experiment.

Dataset	GPU computation time (GPU days)
N-ASL	280
N-Caltech101	67
DVS-Gesture	24
Fan1vs3	7

References

1. Amir, A., Taba, B., Berg, D., Melano, T., McKinstry, J., Di Nolfo, C., Nayak, T., Andreopoulos, A., Garreau, G., Mendoza, M., Kusnitz, J., Debole, M., Esser, S., Delbruck, T., Flickner, M., Modha, D.: A Low Power, Fully Event-Based Gesture Recognition System. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7388–7397 (Jul 2017). https://doi.org/10.1109/CVPR.2017.781, https://ieeexplore.ieee.org/document/8100264, iSSN: 1063-6919
2. Biewald, L.: Experiment tracking with weights and biases (2020), https://www.wandb.com/, software available from wandb.com
3. Finateu, T., Niwa, A., Matolin, D., Tsuchimoto, K., Mascheroni, A., Reynaud, E., Mostafalu, P., Brady, F., Chotard, L., LeGoff, F., Takahashi, H., Wakabayashi, H., Oike, Y., Posch, C.: 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 (Feb 2020)
4. Gehrig, D., Loquercio, A., Derpanis, K., Scaramuzza, D.: End-to-End Learning of Representations for Asynchronous Event-Based Data. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 5632–5642. IEEE, Seoul, Korea (South) (Oct 2019). https://doi.org/10.1109/ICCV.2019.00573, https://ieeexplore.ieee.org/document/9009469/
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition (Dec 2015). https://doi.org/10.48550/arXiv.1512.03385, http://arxiv.org/abs/1512.03385, arXiv:1512.03385 [cs]
6. Kingma, D.P., Ba, J.: Adam: A Method for Stochastic Optimization (Jan 2017), http://arxiv.org/abs/1412.6980, arXiv:1412.6980 [cs]
7. Lenz, G., Chaney, K., Shrestha, S.B., Oubari, O., Picaud, S., Zarrella, G.: Tonic: event-based datasets and transformations. (Jul 2021). https://doi.org/10.5281/zenodo.5079802, https://zenodo.org/records/5079802
8. Subramoney, A., Nazeer, K.K., Schöne, M., Mayr, C., Kappel, D.: Efficient recurrent architectures through activity sparsity and sparse back-propagation through time (Mar 2023), http://arxiv.org/abs/2206.06178, arXiv:2206.06178 [cs]