EventDrop: Data Augmentation for Event-based Learning

Fuqiang Gu[1] , **Weicong Sng**[2] , **Xuke Hu**[3] and **Fangwen Yu**[4*]
[1] College of Computer Science, Chongqing University, China
[2] School of Computing, National University of Singapore, Singapore
[3] Institute of Data Science, German Aerospace Center, Germany
[4] Department of Precision Instrument, Tsinghua University, China
gufq@cqu.edu.cn, sngweicong@comp.nus.edu.sg, xuke.hu@dlr.de, yufangwen@tsinghua.edu.cn

Abstract

The advantages of event-sensing over conventional sensors (e.g., higher dynamic range, lower time latency, and lower power consumption) have spurred research into machine learning for event data. Unsurprisingly, deep learning has emerged as a competitive methodology for learning with event sensors; in typical setups, discrete and asynchronous events are first converted into frame-like tensors on which standard deep networks can be applied. However, over-fitting remains a challenge, particularly since event datasets remain small relative to conventional datasets (e.g., ImageNet). In this paper, we introduce EventDrop, a new method for augmenting asynchronous event data to improve the generalization of deep models. By dropping events selected with various strategies, we are able to increase the diversity of training data (e.g., to simulate various levels of occlusion). From a practical perspective, EventDrop is simple to implement and computationally low-cost. Experiments on two event datasets (N-Caltech101 and N-Cars) demonstrate that EventDrop can significantly improve the generalization performance across a variety of deep networks.

1 Introduction

Event sensors, such as DVS event cameras [Patrick *et al.*, 2008] and NeuTouch tactile sensor [Taunyazov *et al.*, 2020], are bio-inspired devices that mimic the efficient event-driven communication mechanisms of the brain. Compared to conventional sensors (e.g., RGB cameras), which synchronously capture the scene at a fixed rate, event sensors asynchronously report the changes (called events) of the scene. For example, DVS cameras capture the changes in luminosity over time for each pixel independently rather than intensity images as RGB cameras do. Event sensors usually have the advantages of higher dynamic range, higher temporal resolution, lower time latency, and higher power efficiency [Gehrig *et al.*, 2019]. These advantages have stimulated research into machine learning for event data. Unsurprisingly, deep learning, which performs extremely well on a variety of tasks, remains a competitive method for learning with event sensors.

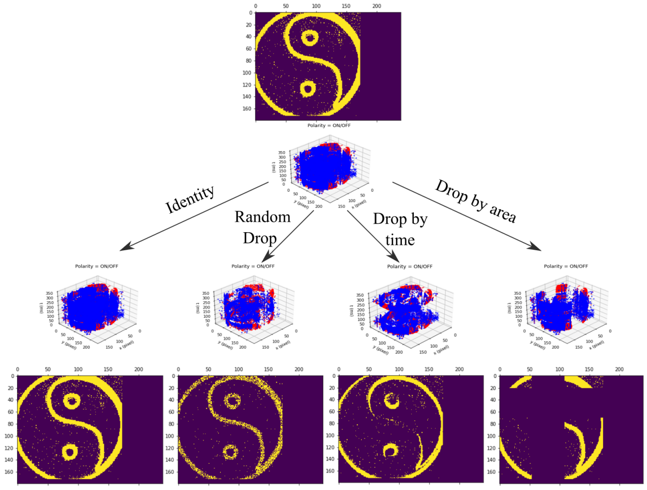


Figure 1: An example of augmented events with EventDrop. For better visualization, the event frame representation is used to visualize the outcome of augmented events.

A challenging problem in deep learning is over-fitting, which causes a model that exhibits excellent performance on training data to degrade dramatically when validated against new and unseen data. A simple solution to the over-fitting problem is to significantly increase the amount of labeled data, which is theoretically feasible but may be cost-prohibitive in practice. The over-fitting problem is more severe in learning with event data since event datasets remain small relative to conventional datasets (e.g., ImageNet).

Data augmentation is a way to increase both the amount and diversity of data from existing data, which can improve the generalization ability of deep learning models. For images, common augmentation techniques include Translation, Rotating, Flipping, Cropping, Contrast, Sharpness, Shearing, etc [Cubuk *et al.*, 2019]. Event data are fundamentally different from frame-like data (e.g., images), and hence we cannot directly use these augmentation techniques that are originally developed for frame-like data to augment asynchronous event data.

In this paper, we present EventDrop, a novel method to

[*] Corresponding Author

augment event data by dropping events. This is motivated by the observations that the number of events significantly changes over time even for the same scene and that occlusion often occurs in many vision tasks. To address these issues, we propose three strategies to select certain events to be dropped, including *Random drop*, *Drop by time*, and *Drop by area*. Figure 1 shows a specific example of augmented events with different operations of EventDrop. Through these augmentation operations, we can enlarge both the amount of training data as well as the data diversity, which will benefit deep learning models. To the best of our knowledge, EventDrop is the first work to augment asynchronous event data by dropping events.

The closely-related works to this study are Dropout [Srivastava *et al.*, 2014], Cutout [DeVries and Taylor, 2017], RE [Zhong *et al.*, 2020] and SpecAugment [Park *et al.*, 2019], all of which introduce some noise to improve the generalization ability of deep learning models. However, Dropout drops the units and their connections in the models that can be in the intermediate layers, while our method only drops events in the input space. EventDrop can be considered as an extension of Dropout in the input space. Compared to Cutout and RE, both of which deal with images by considering occlusion, EventDrop works with event data and deals with both sensor noise and occlusion. SpecAugment works on audio, while our method deals with event-based data.

In summary, our main contributions are as follows:

- We propose EventDrop, a novel method for augmenting asynchronous event data, which is simple to implement, computationally low-cost, and can be applied to various event-based tasks.
- We evaluate the proposed method on two public event datasets with different event representations. Experimental results show that the proposed method significantly improves the generalization performance across a variety of deep networks.

2 Related Work

2.1 Event-based Learning

Event-based learning has been increasingly popular due to the advantages of event sensors (e.g, low time latency, low power consumption, and high dynamic range) [Gallego *et al.*, 2020; Lee *et al.*, 2020]. Event-based learning algorithms can be grouped into two major approaches. One approach is to first convert asynchronous events into frame-like data, such that frame-based learning methods can be applied directly (e.g., state-of-the-art DNNs). Some representative works include event frame [Rebecq *et al.*, 2017], Event Count Image [Maqueda *et al.*, 2018], Voxel Grid [Zhu *et al.*, 2019], and Event Spike Tensor (EST) [Gehrig *et al.*, 2019]. While such methods can make use of the powerful ability of modern DNNs through event conversion, they may discard some useful information about the events (e.g., polarity, temporal information, density).

The other approach is to directly use spiking neural networks (SNNs) on the asynchronous event data. The event-driven property of SNNs makes them inherently suitable for dealing with event data. Compared to standard DNNs, SNNs are more biologically plausible and more energy efficient when implemented on neuromorphic processors. Event-based learning with SNNs has been used for object recognition [Gu *et al.*, 2020], visual-tactile perception [Taunyazov *et al.*, 2020], etc. While SNNs are attractive for dealing with event data, the spike function is not differentiable and hence one cannot directly use backpropagation methods to train the SNNs. Several solutions have been proposed to address this issue, such as converting DNNs to SNNs and approximating the derivative of the spike function [Wu *et al.*, 2019]. However, the overall performance of SNNs is often inferior to standard DNNs.

In this study, we focus on data augmentation for event-based learning with DNNs, but the proposed method can be also applied for SNN-based methods.

2.2 Regularization

Regularization is a key technique for mitigating over-fitting in the training of deep learning models. Common regularization strategies include weight decay, and Dropout [Goodfellow *et al.*, 2016]. The basic idea of weight decay is to penalize the model weights by adding a term to the loss function. Popular forms of weight decay are L^1 and L^2 regularization. Dropout is also a widely-used regularization technique, which simulates sparsity for the layer it is applied to. In the standard Dropout method [Srivastava *et al.*, 2014], units and their connections are randomly dropped out from the model with a certain probability (e.g., 0.5) during training. Many variants have been proposed to further improve the speed or regularization effectiveness [Labach *et al.*, 2019]. Compared to Dropout, this study drops events in the input space rather than drops the units and their connections in the models.

2.3 Data Augmentation

Data augmentation can be also regarded as a regularization method that improves the generalization ability of deep learning models. It is widely accepted that deep learning models over-fit, and benefit strongly from larger datasets. Data augmentation is a practical technique to increase the amount of training data as well as the data diversity. Many studies have demonstrated that deep learning models can significantly improve their generalization ability by applying some transforms on the input images [Krizhevsky *et al.*, 2012], such as Translation, Rotation, Flipping, and Cropping. Recently, A popular augmentation technique called SamplePair [Inoue, 2018] is proposed for image classification, which creates a new sample from one image by overlaying another image that is randomly selected from training data.

Different from existing works that deal with images, this study works on the augmentation of event data, which remains unexplored to the best of our knowledge. We focus particularly on the object occlusion problem and noisy event data. The closely-related works that also deal with occlusion are Cutout [DeVries and Taylor, 2017], RE [Zhong *et al.*, 2020], and SpecAugment [Park *et al.*, 2019]. Specifically, Cutout applies a fixed-size zero mask to a random location of each input image, while RE erases the pixels in the randomly selected region with random values. Compared to Cutout and

RE, which deal with images, our approach deals with event data and considers both object occlusion and sensor noise. SpecAugment is an augmentation method for audio, which operates on the log mel spectrogram of the input audio. Both SpecAugment and our method are inspired by Cutout. However, SpecAugment works on audio, while our method deals with event data (which are asynchronous events).

3 Event Representation

State-of-the-art DNNs usually deal with frame-like data (e.g., images, videos) and cannot be directly used for event data since event data are a stream of asynchronous events. An individual event alone, contains little information about the scene. To make use of event data, certain methods have been proposed to learn useful frame-like representations that can be exploited by DNNs. Popular event representations include Event Frame [Rebecq *et al.*, 2017], Event Count Image [Maqueda *et al.*, 2018], Voxel Grid [Zhu *et al.*, 2019], and EST [Gehrig *et al.*, 2019], which we will introduce in the following. Figure 2 shows the general framework of converting asynchronous event data into popular event representations.

Let ε be a sequence of events, which encode the location, time, polarity (sign) of the changes. It can be described as:

$$\varepsilon = \{e_i\}_{i=1}^{I} = \{x_i, y_i, t_i, p_i\}_{i=1}^{I}, \tag{1}$$

where (x_i, y_i) is the coordinate of the pixel triggering the event e_i, t_i is the timestamp when the event is generated, and p_i is the polarity of the event. The polarity takes two values: 1 and -1, representing positive and negative events, respectively. I is the number of events.

Event Frame represents events using the histograms of events for each pixel, which can be written as (denoted by V_{EF}):

$$V_{EF}(x_l, y_m) = \sum_{e_i \in \varepsilon} \delta(x_l - x_i)\delta(y_m - y_i), \tag{2}$$

$$\delta(a) = \begin{cases} 1, \text{ if } a = 0 \\ 0, \text{ otherwise}, \end{cases} \tag{3}$$

where $\delta(\cdot)$ is an indicator function. (x_l, y_m) is the pixel coordinate in the Event Frame representation, and $x_l \in \{0, 1, \cdots, W-1\}$, $y_m \in \{0, 1, \cdots, H-1\}$. The Event Frame can be regarded as a 2D image with a resolution of $H \times W$.

Event Count Image is similar to Event Frame, but it uses separate histograms for positive events and negative events. Event Count Image V_{EC} is described as:

$$V_{EC}(x_l, y_m, \pm) = \sum_{e_i \in \varepsilon_\pm} \delta(x_l - x_i)\delta(y_m - y_i), \tag{4}$$

where ε_+ and ε_- are event sequences with positive polarity and negative polarity, respectively. The Event Count Image can be regarded as a two-channel image with each channel corresponding to one polarity.

Voxel Grid V_{VG} considers the temporal information of the events, which is not explicitly handled in Event Frame and Event Count Image. It is written as

$$V_{VG}(x_l, y_m, c_n) = \sum_{t_{n-1} < t_i \le t_n} \delta(x_l - x_i)\delta(y_m - y_i)\mathbf{1}_{t_i}, \tag{5}$$

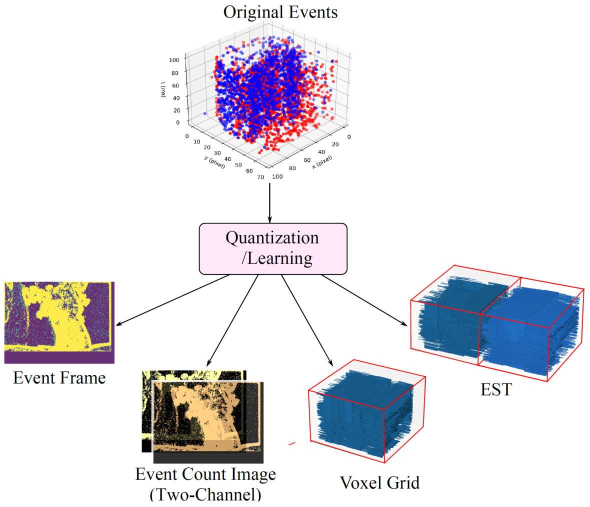


Figure 2: General framework of converting events into popular event representations. The original asynchronous events can be transformed into frame-like data through quantization or learning (e.g., neural networks).

$$t_n = t_1 + (c_n + 1)\Delta T, \tag{6}$$

where $\mathbf{1}_{t_i}$ is an indicator function, which takes 1 when t_i is in the interval $(t_{n-1}, t_n]$ and 0 otherwise. c_n is the temporal index of the Voxel Grid representation, and $c_n \in \{0, 1, \cdots, C-1\}$. ΔT is temporal bin size.

Similar to Voxel Grid, EST is also a grid-based representation that is learned end-to-end directly from asynchronous event data through differentiable kernel convolution and quantization. EST considers both temporal information and polarity about events, which is described as:

$$\begin{aligned} &V_{EST}(x_l, y_m, c_n, \pm) \\ &= \sum_{e_i \in \varepsilon_\pm} f_\pm(x_i, y_i, t_i) k(x_l - x_i, y_m - y_i, t_n - t_i), \end{aligned} \tag{7}$$

where $f_\pm(x, y, t)$ is the normalized timestamp, and $f_\pm(x, y, t) = \frac{t - t_1}{\Delta T}$ where t_1 is the first timestamp, ΔT is the bin size. $k(x, y, t)$ is a trilinear kernel, which is written as

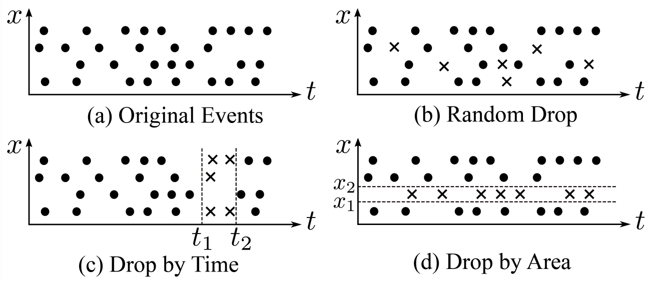
$$k(x, y, t) = \delta(x, y)\max(0, 1 - \left|\frac{t}{\Delta T}\right|). \tag{8}$$

In addition to the above representations, there are other event representations such as HOTS [Lagorce *et al.*, 2016], HATS [Sironi *et al.*, 2018], and Matrix-LSTM [Cannici *et al.*, 2020]. In this study, we take the four representations as representatives to analyze how EventDrop enhances the performance of DNNs.

4 Proposed Method: Augmenting Event Data by Dropping Events

4.1 Motivation

This work is motivated by two observations. The first observation is that the output of event cameras for the same scene



(a) Original Events

(b) Random Drop

(c) Drop by Time

(d) Drop by Area

Figure 3: Strategies used by EventDrop, where t indicates time dimension, x denotes the pixel coordinate (only one dimension is shown here for clarity). Dots represent original events, and × denotes the events to be dropped. Dashed lines represent threshold borders. (a) Original events that are triggered asynchronously. (b) *Random drop* strategy. (c) *Drop by time* strategy. (d) *Drop by area* strategy.

under the same lighting condition may vary significantly over time. This may be because event cameras are somehow noisy, and events are usually triggered when the change about the scene reaches or surpasses a threshold. By randomly dropping a proportion of events, it is possible to improve the diversity of event data and hence increase the performance of downstream applications.

The second observation is that occlusion often occurs in many tasks such as object recognition and tracking. The generalization ability of a machine learning model depends highly on the diversity of training data, including various levels of occlusion. However, the available training data usually suffer from limited variance in the occlusion level. A machine learning model trained on the data with limited or no (totally visible) occlusion variance may generalize poorly on new samples that are partially occluded. By generating new samples that simulate partially occluded cases, the model is able to better recognize objects with partial occlusion.

4.2 Strategies of Dropping Events

To address the above issues, we propose three strategies of dropping events to augment event data, namely *Random drop*, *Drop by time*, and *Drop by area*. The first strategy is to prepare the model for noisy event data. The other two strategies are for simulating the occlusion problem. Figure 3 illustrates the idea of different dropping strategies. We describe the three strategies in the following.

- **Random drop**. The basic idea of *Random drop* is to randomly drop a proportion of events in the sequence. This is to overcome the noise from the event sensors.
- **Drop by time**. *Drop by time* is to drop events triggered within a random period of time. It tries to increase the diversity of training data by stimulating the case that objects are partially occluded during certain time period.
- **Drop by area**. *Drop by area* is to drop events triggered within a randomly selected pixel area. It also aims to improve the data diversity by simulating various cases that some parts of the objects are partially occluded.

Algorithm 1: Procedures of augmenting event data with EventDrop

```
Input  : A sequence of events
         $\varepsilon = \{e_i\}_{i=1}^{I} = \{x_i, y_i, t_i, p_i\}_{i=1}^{I}$, pixel
         resolution $(W, H)$.
Output: Augmented event sequences $\varepsilon^*$.
Initialize the $\varepsilon^*$ to an empty set, namely $\varepsilon^* = \{\}$;
$Operation \leftarrow$ Random.choice($identity$,
  $drop\_by\_time$, $drop\_by\_area$, $random\_drop$);
if $Operation == identity$ then
    $\varepsilon^* \leftarrow \varepsilon$;
end
if $Operation == drop\_by\_time$ then
    $\rho \leftarrow Rand(1, 10)/10$;
    $T_{min} \leftarrow Rand(t_1, t_I)$;
    $T_{max} \leftarrow max(t_I, T_{min} + \rho * (t_I - t_1))$;
    for $e_i \in \varepsilon$ do
        if $(t_i < T_{min}) \| (t_i > T_{max})$ then
            Add $e_i$ into $\varepsilon^*$;
        end
    end
end
if $Operation == drop\_by\_area$ then
    $\rho \leftarrow Rand(1, 6)/20$;
    $x_0 \leftarrow Rand(0, W)$;
    $y_0 \leftarrow Rand(0, H)$;
    for $e_i \in \varepsilon$ do
        if
          $(x_i \in [x_0, x_0 + \rho * W]) \& (y_i \in [y_0, y_0 + \rho * H]$
          then
            Do nothing;
        end
        else
            Add $e_i$ into $\varepsilon^*$;
        end
    end
end
if $Operation == random\_drop$ then
    $\rho \leftarrow Rand(1, 10)/10$;
    $\varepsilon^* = Random.choices(\varepsilon, I * (1 - \rho))$
end
return $\varepsilon^*$.
```

4.3 Implementation

In this section, we describe the implementation of EventDrop. Algorithm 1 gives the procedures of augmenting event data with EventDrop. This algorithm takes as input a sequence of asynchronous events and corresponding image resolution (W, H). We first define four augmentation techniques, namely *Identity*, *Random drop*, *Drop by time*, and *Drop by area*, and conduct one augmentation technique that is randomly selected on the event sequence. The random policy exploration in [Cubuk *et al.*, 2020] is adopted in this study due to its simplicity and excellent performance. The probability p of each of these augmentation operations being chosen is set to equal (namely, $p = 0.25$). The magnitude of

Random drop and *Drop by time* is discretized into 9 different levels and that of *Drop by area* into 5 levels. Specifically, when conducting the *Drop by time* operation, a magnitude is first randomly selected and then a period of time is selected. The events triggered within the selected time period would be deleted from the event sequence, the remaining event sequence will be returned as the output of the algorithm. In the *Drop by area* operation, a pixel region is first selected by a random magnitude and a random location, and then the events within the selected region would be dropped. In the *Random drop* operation, a proportion of events are randomly selected to be dropped. Overall, EventDrop is simple to implement and computationally low-cost. We have implemented EventDrop in PyTorch and the source code is available at *https://github.com/fuqianggu/EventDrop*.

5 Experiments and Results

5.1 Datasets

We evaluate the proposed EventDrop augmentation technique using two public event datasets: N-Caltech101 [Orchard *et al.*, 2015] and N-Cars [Sironi *et al.*, 2018]. N-Caltech101 (Neuromorphic-Caltech101) is the event version of the popular Caltech101 dataset [Fei-Fei *et al.*, 2004]. To convert the images to event sequences, an ATIS event camera was installed on a motorized pan-tilt unit, and automatically moved while pointing at images from the original dataset (Caltech101) that were shown on a LCD monitor. N-Cars (Neuromorphic-Cars) is a real-world event dataset for recognizing whether a car is present in a scene. It was recorded using an ATIS camera that was mounted behind the windshield of a car.

5.2 Experiment Setup

We evaluate the proposed method using four state-of-the-art deep learning architectures, namely ResNet-34 architecture [He *et al.*, 2016], VGG-19 [Simonyan and Zisserman, 2014], MobileNet-V2 [Sandler *et al.*, 2018], and Inception-V3 [Szegedy *et al.*, 2016]. All the networks are pretrained on ImageNet [Russakovsky *et al.*, 2015]. Since the number of input channels and output classes for our case are different from these pre-trained models, we adopt the approach used in [Gehrig *et al.*, 2019] and replace the first and last layer of the pre-trained models with random weights, and then fine-tune all the parameters on the task.

Since event data are a stream of asynchronous events and cannot be directly applied to deep nets, we consider and implement the four event representations introduced in Section 3. For the implementation of EST, we replace the neural network with a trilinear kernel to convolve with the normalized timestamps for computational efficiency. Note that the considered deep learning models take as input 2D images, while some event representations we considered (e.g., Voxel Grid and EST) are 3D or 4D tensor. To adapt to these pretrained model, we concatenate the event representation along the polarity and/or temporal dimension as channels.

The Adam optimizer is used to train the model by minimizing the cross-entropy loss. The initial learning rate is set to 1×10^{-4} until the iteration reaches up to 100, after which the learning rate is reduced by a factor of 0.5 every 10 iterations. The total number of iterations is set to 200. We use a batch size of 4 for both datasets. To conduct a robust evaluation, we run the model on each dataset for multiple rounds with different random seeds, and report the mean and standard deviation values. We perform early stopping on a validation set using the splits provided by the EST [Gehrig *et al.*, 2019] on N-Caltech101 and 20% of the training data on N-Cars.

Model	**Representation**	**Average Accuracy (Std)**	
		Baseline	**EventDrop**
ResNet-34	Event Frame	77.39 (0.78)	78.20 (0.15)
	Event Count	77.75 (0.64)	78.30 (0.29)
	Voxel Grid	82.47 (0.80)	82.57 (0.42)
	EST	83.91 (0.44)	**85.15 (0.36)**
VGG-19	Event Frame	72.31 (1.38)	74.99 (0.67)
	Event Count	73.02 (1.05)	75.01 (0.57)
	Voxel Grid	76.63 (0.81)	77.28 (0.45)
	EST	78.88 (0.79)	**79.55 (1.25)**
MobileNet-V2	Event Frame	79.08 (0.84)	82.19 (0.63)
	Event Count	79.68 (1.09)	82.31 (0.72)
	Voxel Grid	83.12 (0.55)	85.56 (0.79)
	EST	84.76 (0.64)	**87.14 (0.54)**
Inception-V3	Event Frame	80.01 (0.81)	81.46 (0.55)
	Event Count	80.15 (0.56)	81.01 (0.81)
	Voxel Grid	82.68 (0.53)	84.54 (0.89)
	EST	84.60 (0.76)	**85.78 (0.63)**

Table 1: Object recognition accuracy (%) of different deep nets with varying representations on N-Caltech101.

5.3 Results on N-Caltech101

We first analyze the results of EventDrop on the N-Caltech101 dataset. The results from the same models without data augmentation are considered as the baselines. Table 1 compares the performance of EventDrop and the baselines. We can see that EventDrop improves the performance of all the models used with different event representations. The accuracy achieved with Voxel Grid and EST representations is much higher than that with Event Frame and Event Count representations. This is attributed to the fact that Voxel Grid and EST contain temporal information about the events that is discarded by Event Frame and Event Count. Since EST further considers the polarity information about the events, it behaves slightly better than Voxel Grid. The same trend can be found when comparing Event Frame (without polarity information) and Event Count (with polarity information). Among these deep nets, MobileNet-V2 seems to perform slightly better than ResNet-34 and Inception-V3, while VGG-19 performs the worst, probably because it is relatively old.

5.4 Results on N-Cars

We then compare the results of EventDrop with the baselines on N-Cars. As can be seen from Table 2, EventDrop outperforms the baselines with different deep learning architectures and event representations. The improvement on N-Cars dataset is generally greater than that on N-Caltech101 dataset. This might be because N-Cars is a real-world data dataset, and EventDrop works better with real-world cases where sensor noise and occlusion occur more likely than simulation

Model	Representation	Average Accuracy (Std)	
		Baseline	EventDrop
ResNet-34	Event Frame	91.83 (0.61)	94.04 (0.19)
	Event Count	92.18 (0.34)	95.20 (0.38)
	Voxel Grid	91.09 (0.46)	93.61 (0.82)
	EST	91.03 (1.30)	**95.50 (0.18)**
VGG-19	Event Frame	91.61 (0.82)	92.74 (0.81)
	Event Count	91.20 (0.53)	93.19 (1.00)
	Voxel Grid	91.45 (0.88)	**93.39 (0.86)**
	EST	91.72 (0.85)	93.12 (0.95)
MobileNet-V2	Event Frame	91.87 (0.54)	93.64 (0.50)
	Event Count	92.70 (1.22)	**95.19 (0.71)**
	Voxel Grid	91.18 (0.61)	94.05 (0.38)
	EST	91.71 (0.29)	94.55 (0.45)
Inception-V3	Event Frame	91.21 (0.56)	92.22 (2.73)
	Event Count	91.16 (0.74)	94.41 (0.99)
	Voxel Grid	90.67 (0.96)	92.12 (1.68)
	EST	90.91 (1.78)	**94.44 (0.72)**

Table 2: Object recognition accuracy (%) of different deep nets with varying representations on N-Cars.

(i.e., where the event camera looks at a projected scene rather than a real-world scene). The improvement of EventDrop can reach up to about 4.5% (by ResNet-34 with EST representation). The Event Count and EST representations, which consider polarity information, perform better than the Event Frame and Voxel Grid that do not take polarity into account. The performance of the four deep nets is similar among the baselines, while ResNet-34 and MobileNet-V2 achieve better accuracy when the training data is augmented with EventDrop.

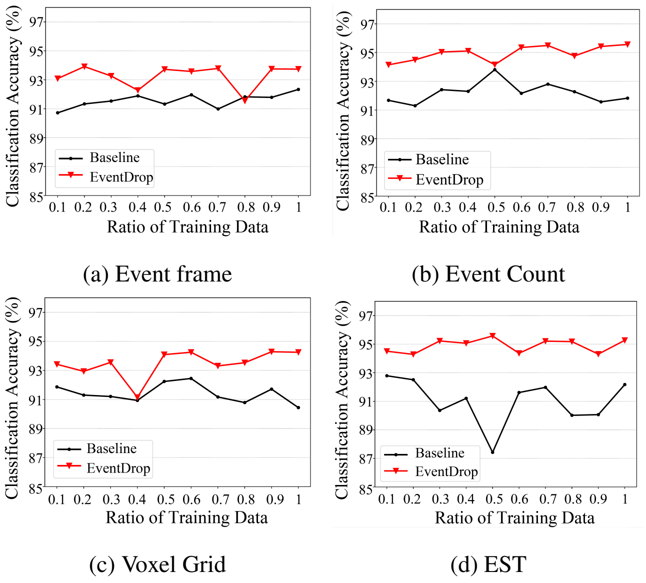
5.5 Comparison of Different Dropping Strategies

We also compare the performance of different dropping strategies on both datasets. In the implementation of *Drop by time*, *Drop by area*, and *Random drop* operations, the probability of each operation being conducted is set to 0.5, and the corresponding magnitude is randomly selected from the value set described in Section 4. For EventDrop, the three dropping strategies and *Identity* operations are randomly selected to conduct with equal probability. As demonstrated in Table 3, EventDrop that integrates different dropping strategies outperforms the baselines on both datasets, and the improvement of EventDrop over the baselines is bigger on N-Cars dataset than on N-Caltech101 dataset.

Specifically, for N-Caltech101 dataset, EventDrop and *Drop by area* operations have better performance than *Drop by time* and *Random drop* operations in general. The *Drop by time* operation seems not to improve the baselines when using Voxel Grid and EST representations but it improves the performance when using event frame and Event Count representations. This might be explained by N-Caltech101 being a simulated dataset in which the sensor noise and occlusion in time are negligible, and hence discarding some events that are selected randomly or triggered during a certain period of time does not increase the diversity of data. By contrast, for N-Cars dataset, all the dropping operations result in a better accuracy than the baselines. This might be because the real-world event dataset (N-Cars) suffers more from sensor noise and various occlusions, and dropping operations can better increase the data diversity.

Representation	Dropping Strategy	Average Accuracy (Std)	
		N-Caltech101	N-Cars
Event Frame	Baseline	77.39 (0.78)	91.83 (0.61)
	Drop by time	78.49 (0.70)	92.81 (1.27)
	Drop by area	77.49 (0.71)	92.59 (0.71)
	Random drop	77.19 (0.98)	92.23 (0.30)
	EventDrop	78.20 (0.15)	**94.04 (0.19)**
Event Count	Baseline	77.75 (0.64)	92.18 (0.34)
	Drop by time	78.12 (0.83)	93.91 (0.61)
	Drop by area	77.24 (0.80)	93.81 (0.49)
	Random drop	77.68 (0.54)	92.93 (0.87)
	EventDrop	78.30 (0.29)	**95.20 (0.38)**
Voxel Grid	Baseline	82.47 (0.80)	91.09 (0.46)
	Drop by time	80.80 (0.72)	92.97 (0.44)
	Drop by area	83.84 (1.09)	92.04 (0.66)
	Random drop	82.92 (1.00)	91.29 (0.79)
	EventDrop	82.57 (0.42)	**93.61 (0.82)**
EST	Baseline	83.91 (0.44)	91.03 (1.30)
	Drop by time	83.65 (0.59)	94.73 (0.38)
	Drop by area	85.18 (0.83)	92.71 (1.03)
	Random drop	84.07 (0.52)	93.84 (0.52)
	EventDrop	85.15 (0.36)	**95.50 (0.18)**

Table 3: Accuracy (%) comparison of different dropping strategies based on ResNet-34.



(a) Event frame

(b) Event Count

(c) Voxel Grid

(d) EST

Figure 4: Object classification accuracy using different ratios of training data on N-Cars with ResNet-34.

5.6 Effect of the Amount of Training Data

In this section, we analyze the effect of using different amounts of training data. The ratio we considered ranges from 0.1 to 1 where 0.1 represents only 10% training data that are randomly selected are used to train the network. To reduce the search space, we fix the random seed that is shared by the baselines and EventDrop, and then compare their performance. Figure 4 shows that EventDrop consistently improves the baselines in general. It can achieve about 94% accuracy even with only 10% training data compared to about 91.5% by the baseline. Although the improvement of EventDrop over the baseline is marginal when it is trained with certain ratios of training data, such a problem would obviously be reduced by averaging the results over more runs. It is also clear that the improvement of EventDrop is stable when using the more relatively robust EST representation.

6 Conclusion

In this paper, we propose a new augmentation method for event-based learning, which we call EventDrop. It is easy to implement, computationally low-cost, and does not involve any parameter learning. We have demonstrated that by dropping events selected with certain strategies, we can significantly improve the object classification accuracy of different deep networks on two event datasets. While we show the application of our approach for event-based learning with deep nets, our approach can be also applied to learning with SNNs. For future work, we will apply our approach to other event-based learning tasks, such as visual inertial odometry, place recognition, traffic flow estimation, and simultaneous localization and mapping.

References

[Cannici *et al.*, 2020] Marco Cannici, Marco Ciccone, Andrea Romanoni, and Matteo Matteucci. Matrix-lstm: a differentiable recurrent surface for asynchronous event-based data. *arXiv preprint arXiv:2001.03455*, 2020.

[Cubuk *et al.*, 2019] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 113–123, 2019.

[Cubuk *et al.*, 2020] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

[DeVries and Taylor, 2017] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[Fei-Fei *et al.*, 2004] 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.

[Gallego *et al.*, 2020] Guillermo Gallego, Tobi Delbruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Jörg Conradt, Kostas Daniilidis, et al. Event-based vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[Gehrig *et al.*, 2019] Daniel Gehrig, Antonio Loquercio, Konstantinos G Derpanis, and Davide Scaramuzza. End-to-end learning of representations for asynchronous event-based data. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5633–5643, 2019.

[Goodfellow *et al.*, 2016] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[Gu *et al.*, 2020] Fuqiang Gu, Weicong Sng, Tasbolat Taunyazov, and Harold Soh. Tactilesgnet: A spiking graph neural network for event-based tactile object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[Inoue, 2018] Hiroshi Inoue. Data augmentation by pairing samples for images classification. *arXiv preprint arXiv:1801.02929*, 2018.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[Labach *et al.*, 2019] Alex Labach, Hojjat Salehinejad, and Shahrokh Valaee. Survey of dropout methods for deep neural networks. *arXiv preprint arXiv:1904.13310*, 2019.

[Lagorce *et al.*, 2016] 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, 2016.

[Lee *et al.*, 2020] Chankyu Lee, Syed Shakib Sarwar, Priyadarshini Panda, Gopalakrishnan Srinivasan, and Kaushik Roy. Enabling spike-based backpropagation for training deep neural network architectures. *Frontiers in neuroscience*, 14, 2020.

[Maqueda *et al.*, 2018] Ana I Maqueda, Antonio Loquercio, Guillermo Gallego, Narciso García, and Davide Scaramuzza. Event-based vision meets deep learning on steering prediction for self-driving cars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5419–5427, 2018.

[Orchard *et al.*, 2015] Garrick Orchard, Ajinkya Jayawant, Gregory K Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in neuroscience*, 9:437, 2015.

[Park *et al.*, 2019] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*, 2019.

[Patrick *et al.*, 2008] Lichtsteiner Patrick, Christoph Posch, and Tobi Delbruck. A 128x 128 120 db 15μ s latency asynchronous temporal contrast vision sensor. *IEEE journal of solid-state circuits*, 43:566–576, 2008.

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

[Russakovsky *et al.*, 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[Sandler *et al.*, 2018] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[Sironi *et al.*, 2018] Amos Sironi, Manuele Brambilla, Nicolas Bourdis, Xavier Lagorce, and Ryad Benosman. Hats: Histograms of averaged time surfaces for robust event-based object classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1731–1740, 2018.

[Srivastava *et al.*, 2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[Szegedy *et al.*, 2016] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[Taunyazov *et al.*, 2020] Tasbolat Taunyazov, Weicong Sng, Hian Hian See, Brian Lim, Jethro Kuan, Abdul Fatir Ansari, Benjamin CK Tee, and Harold Soh. Event-driven visual-tactile sensing and learning for robots. In *Robotics: Science and Systems*, 2020.

[Wu *et al.*, 2019] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1311–1318, 2019.

[Zhong *et al.*, 2020] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *AAAI*, pages 13001–13008, 2020.

[Zhu *et al.*, 2019] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Unsupervised event-based learning of optical flow, depth, and egomotion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 989–997, 2019.