Data Augmentation for Event Camera Data

Otto Mäkinen

19 August 2021

I3S / CNRS — Polytech Nice Sophia Université Côte d'Azur, Nice, France

1 Introduction

Event cameras or *DVS (Dynamic Vision Sensor)* are asynchronous sensors that pose a potential shift in the way visual information is acquired. Event cameras sample light based differences and do not have a shutter. Event cameras capture visual information continuously through time. Their pixels operate independently from one another and suppress redundant information by remaining silent unless a change in brightness is detected. The output of an event camera results in spikes with each event representing a change in brightness (log-intensity). When the change of brightness exceeds a certain threshold, the event camera sends an event, that is then transmitted from the chip with the x,y location, the time t and the 1-bit polarity of the change when the brightness threshold is met ("ON") or ("OFF"). However, there are some shortcomings that have been solved by adding some static output and consequently newer versions of event cameras output both static and dynamic information. The key advantages of an event camera is high temporal resolution and low latency. Additionally, very high dynamic range 140db vs. 60db of standard cameras and low power consumption.

Event cameras have become commercially available only since 2008, the recent availability of literature on event cameras as well as the recent plans for mass production claimed by companies, such as Samsung and Prophesee highlight a significant commercial interest and ways to apply this technology to robotics and VR/AR. Subsequently, Event cameras are used for object tracking and gesture recognition. [1]

Data augmentation is a way to increase both the amount and diversity of data from existing data and can help improve the generalization ability of deep learning models. For images, common augmentation techniques include Translation, Rotating, Flipping, Cropping, Contrast, Sharpness, Shearing. If the issue of data scarcity is faced, the simple yet effective techniques such as transformations may pose a limited solution. If a dataset is too small, then a transformed image set via rotation and mirroring etc. may still be too small for a given problem. Another solution is the sourcing of entirely new and synthetic images through various techniques, for example the use of Generative adversarial networks to create new synthetic images for data augmentation. However, event data is fundamentally different for frame-like data (images made up from pixels) and normal data augmentation techniques cannot be directly applied to event based data, as the data is asynchronous. [2]

2 Datatype of event camera data

The DVSGesture dataset was used to build the real-time, gesture recognition system described in the CVPR 2017 paper titled "A Low Power, Fully Event-Based Gesture Recognition System." The DVS Gesture dataset comprises 1,342 instances of a set of 11 hand and arm gestures, grouped in 122 trials collected from 29 users under 3 different lighting conditions. Each trial contains one user standing up against a stationary background and performed all 11 gestures sequentially under the same lighting condition. The gestures include hand waving (both arms), large straight arm rotations (both arms, clockwise and counter-clockwise), forearm rolling (forward and backward), air guitar, air drums, and an "Other" gesture invented by the subject. The 3 lighting conditions are combinations of natural light, fluorescent light,and LED light, which control the effects of shadows and fluorescent light flicker on the DVS128 event camera. Each gesture lasts about 6 seconds. To evaluate classifier performance, 23 subjects are designated as the training set, and the remaining 6 subjects are reserved for out-of-sample validation. The dataset is in the file format of AEDAT and can be converted to AEDAT4 using the Dynamic Vision Viewer. The dataset is available at `http://research.ibm.com/dvsgesture/`. [3], [4]

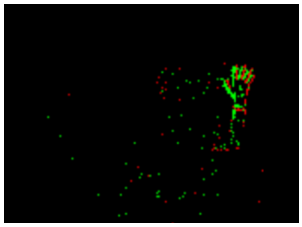

Figure 1: Left hand wave from AEDAT4 file

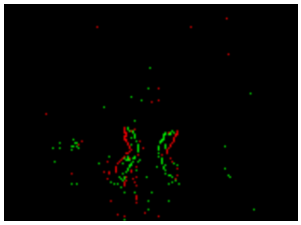

Figure 2: Handclap from AEDAT4 file

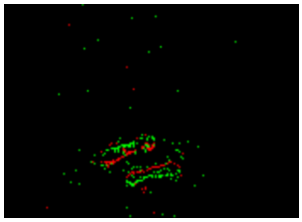

Figure 3: Hand roll from AEDAT4 file

The files in the dataset are in AEDAT3.1 format as polarity events ("ON") or ("OFF") and were converted to AEDAT4. The files were generated by iniVation event-based processing software, such as DV and jAER that use the custom AEDAT (=Address Event DATA) format to store the event data and its timestamp information. The recordings were made with a DVS128 event camera. AEDAT4 uses Google Flatbuffers to serialize data in a convenient and efficient format, with an easy path to extend existing data structures and introduce new ones. Flatbuffers also allow quick and easy support for other languages, such as Python or Java, to be added, by auto-generating the appropriate support files for them.

3 Transformations for event based datasets

The tonic package provides popular datasets and transformations for spike-based/event-based data. The package delivers an interface for converting event based data into python in terms of numpy arrays that can be manipulated. The package is modeled after PyTorch Vision without depending on it to give you the flexibility to use other frameworks as well. Spike-based datasets are published in numerous data formats, scattered around the web and containing different data such as events, images, inertial measurement unit readings and many more. For example the Gesture dataset. The package tries to streamline the download, manipulation and loading of such data, as the files are already converted into .npy format. In addition to downloading datasets, custom transforms can be applied to events and images to pre-process data before feeding them to a classification algorithm such as ANN or SNN. [5]

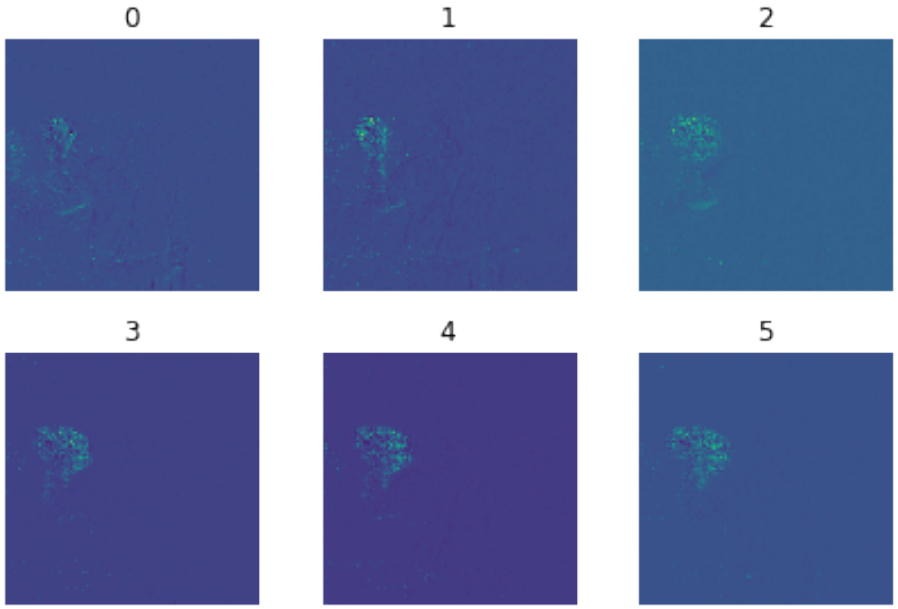

Figure 4: Events accumulated in a grid view for visual inspection.

4 Data Augmentation in Event based Data

ANNs 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. Let ϵ be a sequence of events, which encode the location (x_i, y_i), time (t_i), polarity (p_i) (sign) of the changes, where the event camera produces the sequence of events during some time interval $\Delta(T)$. [6]

$$\epsilon = \epsilon_{i=1} = (x_i, y_i, t_i, p_i)$$

However, event data can be decoded into a matrix representation as a raw and compact format of $(N \times E)$, where N is the number of events and E is the number of event channels, for data manipulation. This is a mapping. The sensor resolution (W, H), depends on the sensor size where DVSGesture has an input sensor size of (128×128), Furthermore, the matrix can be transformed and different data augmentation techniques can be applied to the array. Examples of different techniques to transform and represent event data, such as an event frame, voxel grid, HATS, HOTS and a 4-dimensional sparse tensor. [9], [10]

Representation	Array Shape
Event Frame	$(N \times P \times W \times H)$
Voxel Grid	$(B \times W \times H)$
HATS: Averaged time-surfaces	$(\bar{T} \times W \times H)$
HOTS: Time-surfaces	$(T \times W \times H)$
Sparse Tensor 4D	$(B \times T \times W \times H)$

Table 1: Event representations of sensor data.

Furthermore, other transforms can be applied to augment the data such as randomly dropping events with some drop probability p. 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. Also, combining different strategies for example randomly dropping events, dropping events by time and also by area can reduce a model from overfitting.

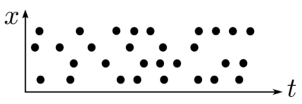

Figure 5: Events with time and spatial location

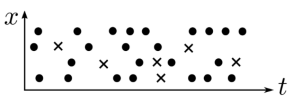

Figure 6: Random drop with probability p

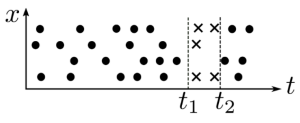

Figure 7: Drop events by timestamp

Also denoising a dataset can improve and clean noisy recordings. Furthermore, denoising can improve the quality of the dataset where event data that is 'not sufficiently connected to other events in the recording.' In practise that means that an event is dropped if no other event occurred within a spatial neighbourhood of 1 pixel and a temporal neighbourhood of filter time time units. Useful to filter noisy recordings where events occur isolated in time.

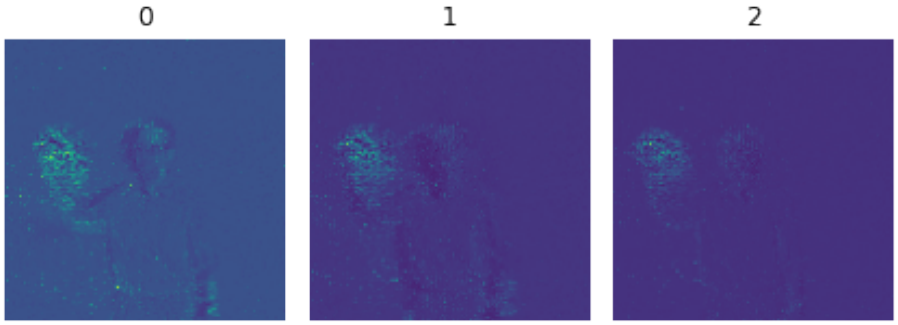

Figure 8: Events with sensor noise.

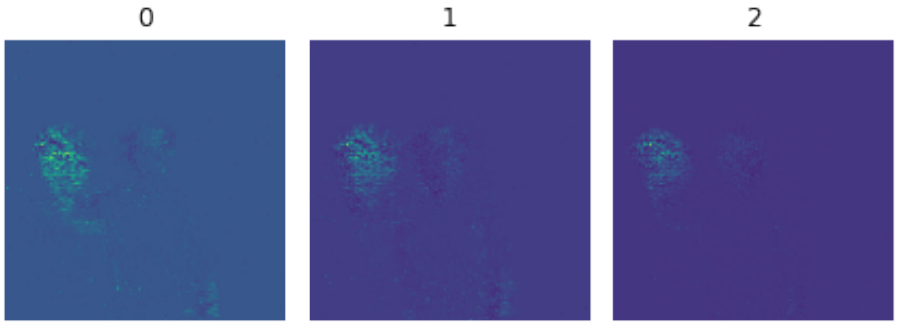

Figure 9: Denoised events.

The function to denoise events in event based data can be expressed in the following way, where the function and the thresholds are defined and an integer value can adjusted [11] to filter time:

$$f(x) = \begin{cases} \alpha, & x > 0 \text{ and } \tilde{t} > t \\ \beta, & x < W - 1 \text{ and } \tilde{t} > t \\ \gamma, & y > 0 \text{ and } \tilde{t} > t \\ \delta, & y < H - 1 \text{ and } \tilde{t} > t \end{cases}$$

4.1 Tensor operations

Tensors are generalisations of matrices for higher-order data and provide useful data representation format. Fueled by increased computing capacity during the past decade, tensors have expanded to many domains, including statistics, data science, and machine learning. Therefore, tensor-based models that preserve the higher-order structure of imaging data, such as event camera data are useful in making different statistical computations efficiently. [12]

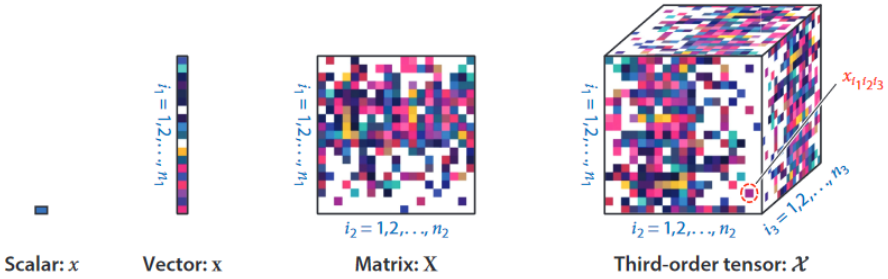

Figure 10: Arrays with different dimensions corresponding to a scalar, a vector, a matrix, and a third-order tensor, where different colors represent different values.

A tensor is defined as a multidimensional array; specifically, a dth-order tensor is an array with d dimensions, denoted by $X \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$, which is an extension of a matrix to higher order. Furthermore, a sparse tensor is a high-dimensional extension of a sparse matrix where non-zero elements are represented as a set of indices and Various sparse storage formats (such as COO, CSR/CSC, LIL, etc.) have been developed that are optimized for a particular structure of non-zero elements in sparse arrays as well as for specific operations on the arrays. [13]

For example, the memory consumption of a 10 000 x 10 000 tensor with 100 000 non-zero 32-bit floating point numbers is at least (2 * 8 + 4) * 100 000 = 2 000 000 bytes when using COO tensor layout and 10 000 * 10 000 * 4 = 400 000 000 bytes when using the default strided tensor layout. Notice the 200 fold memory saving from using the COO storage format. Memory saving is advantageous when running storing a large amount of information to make operations.

5 Pipeline

Machine learning pipelines consist of multiple sequential steps that do everything from data extraction and preprocessing to model training and deployment. Whether you are maintaining multiple models in production or supporting a single model that needs to be updated frequently, an end-to-end machine learning pipeline is a must. A machine learning pipeline starts with a manual iterative process and in the end can be automated for larger implementation in larger organisations. [14]

1. Data as input
2. Data Augmentation: Drops, Flips, Denoise, Transformations
3. Defining the ANN or SNN architecture
4. Building the network with the architecture
5. Training the model
6. Output is the classifier for the machine learning model
7. ANN can also be converted to a SNN, depending on the task

Given an ANN model written in some neural network library, the toolbox parses the provided network files by extracting the relevant information and creating an equivalent Keras model from it. This parsed model serves as common abstraction stage from the input and is internally used by the toolbox to perform the actual conversion to a spiking network. Conversion toolbox currently supports input networks generated with Keras, PyTorch, Lasagne, or Caffe. It is possible to convert an ANN to a SNN. After conversion the SNN can be transfered to a neuromorphic device simulation in a spiking simulator or deployment on dedicated spiking neuron chips. [15]

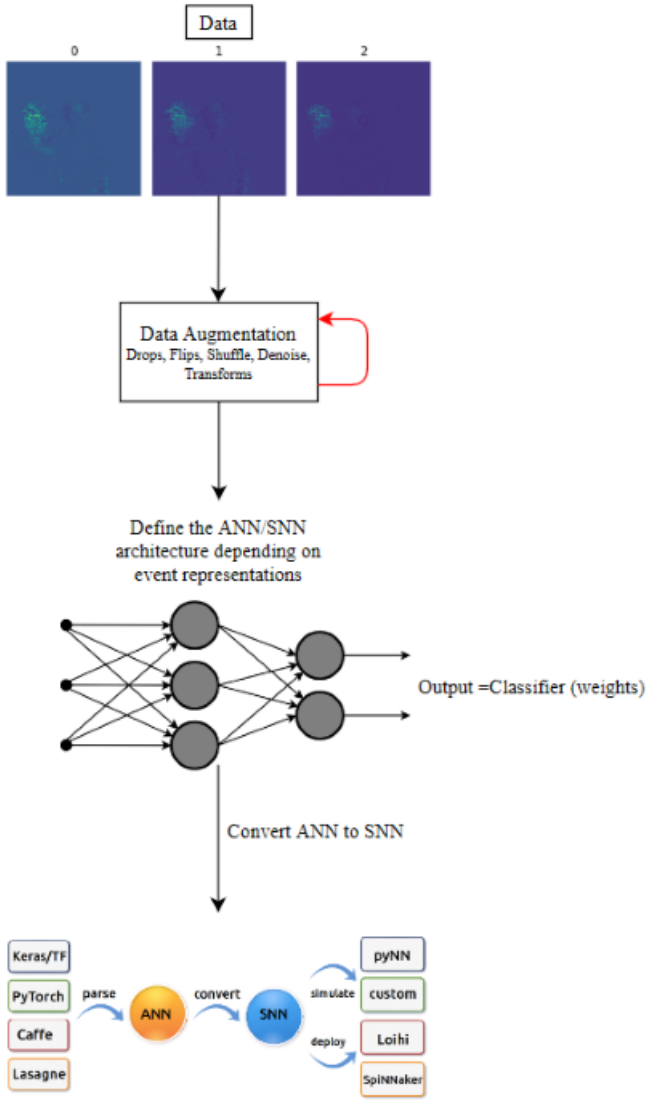

Figure 11: Pipeline of event data, data augmentation methods and transforms, ANN/SNN architecture and possible conversion of ANN to SNN.

6 Conclusions

The work presented in this paper introduces the DVSGesture dataset and how event based data is organised with the different file types. Event data can be transformed and augmented using different techniques such as denoising noisy recordings or dropping events to improve a machine learning model and to reduce overfitting. Events can be transformed and represented in different ways, such as a voxels or sparse tensor representations, where a model can be built from these representations. Different data representations can be tested and compared using deep convolutional models or then a spiking neural network, where accuracy and efficiencies can be compared. Further work that can be explored is training models using event data and comparing accuracy for the classification task.

6.1 Acknowledgements

I thank professor Jean Martinet for supervising the project and this research was funded as an internship at the I3S research centre. The I3S research centre is the largest information and communication science centre located in the French Riviera. The I3S research fields cover many themes in the fields of Computer Science, Computer Engineering, Automation and Signal Processing.

References

[1] Gallego, G., Delbruck, T., Orchard, G., Bartolozzi, C., Taba, B., Censi, A., Leutenegger, S., Davison, A., Conradt, J., Daniilidis, K., Scaramuzza, D. *Event-based Vision: A Survey.* IEEE Trans. Pattern Anal. Machine Intell. (TPAMI), 2020.

[2] Fuqiang Gu, Weicong Sng, Xuke Hu, Fangwen Yu. *EventDrop: data augmentation for event-based learning.* IJCAI 2021.

[3] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza, J. Kusnitz, M. Debole, S. Esser, T. Delbruck, M. Flickner, and D. Modha. *A Low Power, Fully Event-Based Gesture Recognition System.* 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[4] *DVS128 Gesture Dataset.* `http://research.ibm.com/dvsgesture/`

[5] G. Lenz, K. Chaney, S. Shrestha, B. Sumit, O. Oubari, S. Picaud, Z. Guido. *Tonic: event-based datasets and transformations.* `https://tonic.readthedocs.io`

[6] N. Messikommer, D. Gehrig, A. Loquercio, D. Scaramuzza. *Event-Based Asynchronous Sparse Convolutional Networks.* European Conference on Computer Vision (ECCV), Glasgow, 2020.

[7] *Dvsgesture numpy conversion.* `https://github.com/neuromorphs/tonic/blob/main/tonic/datasets/dvsgesture.py`

[8] N. Messikommer, D. Gehrig, A. Loquercio, D. Scaramuzza *End-to-End Learning of Representations for Asynchronous Event-Based Data.* IEEE International Conference on Computer Vision (ICCV), 2019.

[9] A. Sironi, M. Brambilla, N. Bourdis, X. Lagorce, R. Benosman. *HATS: Histograms of Averaged Time Surfaces for Robust Event-based Object Classification.* CVPR 2018.

[10] X. Lagorce, G. Orchard, F. Galluppi, B. E. Shi, R. Benosman. *HOTS: A Hierarchy of Event-Based Time-Surfaces for Pattern Recognition.* IEEE Transactions on Pattern Analysis and Machine Intelligence 2017.

[11] *Denoise function.* `https://github.com/neuromorphs/tonic/blob/402e321a69fee27ddf9533a950a00a4f6a1629c5/tonic/functional/denoise.py`

[12] X. Bi, X. Tang, Y.Yuan, Y. Zhang, A. Qu. *Tensors in Statistics.* Annual Review of Statistics and Its Application 2020.

[13] F. Kjølstad. *Sparse Tensor Algebra Compilation.* PhD thesis, MIT 2020.

[14] Samad, M. Witherow, M. *A Machine Learning Pipeline to Optimally Utilize Limited Samples in Predictive Modeling.* 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)

[15] Rueckauer, B. and Hu, Y. and Lungu, I.A. and Pfeiffer, M. and Liu, S.-C. *Conversion of continuous-valued deep networks to efficient event-driven networks for image classification,Front.* Neurosci., 2017, doi: 10.3389/fnins.2017.00682