Benchmarking Event-Based Object Detection in Lossy Environments

Anonymous ICCV submission

Paper ID ****

Abstract

Event-based object detection is currently one of the primary applications of biologically-inspired neuromorphic cameras. These cameras offer many advantages over traditional cameras in the world of object detection, but due to their high data rates, object-detection typically takes place at a separate location from the camera itself. This work is motivated by the necessity of streaming event data from camera to processor and encoding it into suitable representations for object detection models. In this project, we aimed to test the resilience of an object detection model under simulated lossy streaming conditions to enable real-time object detection. We developed a pipeline to benchmark the state-of-the-art Recurrent Vision Transformer model with one event representation and varying streaming bandwidths and types of loss, finding that the model is quite resilient to streaming loss down to a bandwidth of 25 Mbps, after which the model accuracy rapidly decreases. This work serves as a starting point for more thorough evaluation of additional models and event representations, so that optimal event representations and models can be identified and standardized in real-world applications.

1. Introduction

Recently, biologically inspired event cameras have come into increased use in applications such as object detection, object tracking, and robotics due to their low latency, high dynamic range, and low power consumption. In many cases, event cameras can be deployed in remote operating environments away from a centralized server, for example on an unmaed aerial vehicle (UAV) or robot [7]. However, their high data rates often necessitate off-device processing, as on-device processing would require power-hungry hardware, negating the cameras' inherent low-power advantage. This highlights the critical need for a system for streaming data from event cameras to more capable processing units.

Such an event streaming system must allow continued data transfer under lossy conditions. To optimize event streaming under lossy conditions, we need to understand how different degrees and types of loss effect the downstream applications, and in particular, how loss affects the accuracy of object detection models.

Since event streams do not natively resemble the image frames that traditional object detection models ingest as input, they must be converted into some form of event representation, which transforms the event stream into a tensor. Many different event representations have been proposed [1, 3, 5, 8, 12, 14, 15, 17, 18, 20–22], each optimized for slightly different aspects of the event streaming and processing pipeline. To optimize event streaming under lossy conditions, we need to test object detection models utilizing these event representations under lossy conditions.

This work seeks to make the following contributions:

- An analysis of an event-based object detection model under lossy conditions, with future work analyzing additional event representations and object detection models
- An open-source pipeline for future analysis of additional models and event representations

2. Background and Related Work

2.1. Event Cameras

Event cameras are biologically inspired cameras that record changes in scene intensity by measuring the intensity of light at each pixe, and firing output events if the log of the intensity increases or decreases by more than a threshold ammount. Each event generated takes the form $\langle x, y, t, p \rangle$, where x and y are the x and y coordinates of the pixel, t is the timestamp of the event, precise to the microsecond, and p is the event polarity, taking the value of -1 for an event of decreasing light intensity, and +1 for an event of increasing light intensity [7].

Event cameras offer the advantages of high temporal precision, high dynamic range ($\geq$120 dB), sub-milisecond latency, and low power consumption [7], while suffering from high data rates, exceeding 500 Mbps during scenes with high amounts of motion [6]. These high data rates make it necessary for event data to be streamed to off-camera devices for processing, rather than attempting to process the data locally.

2.2. Event Representations

Event cameras generate streams of events, which are usually encoded as 4-byte chunks within a bytestream. Most computer vision applications in use today were designed for analyzing 2D images of dimension H x W. To analyze video, these models accept a sequence of image frames in the form of a H x W x B tensor, where B is the number of frames in the video. To convert a stream of events into a tensor of this shape, various event representations are used.

Most event representations discretize the temporal domain into B time bins, and accumulate the events at each pixel within each time bin [1, 3, 5, 15, 18, 22].

The Event Histogram representation generates a histogram of positive events at each pixel and time bin, followd by a histogram of negative events at each pixel and time bin [15].

The Mixed Density Event Stack representation generates overlapping temporal windows with decreasing numbers of events in each stack to capture the movements of objects of varying speeds [16].

The Event Temporal Image representation generates a histogram of events that cancels polarities, so that the value at each location is the difference between the number of positive and negative events at a particular pixel and time bin, mapped to the range [0, 255] [5].

The Voxel Grid representation employs a similar strategy, but uses a bilinear sampling kernel to maintain the temporal distribution of events within each time bin [22].

2.3. Event-based Object Detection

Many vision transformer models, such as [9] and [18] have been applied to the domain of event cameras, leveraging the transformer network architecture to detect objects from event representations. Other models harness specific qualities of event-based data to detect objects moving at different speeds and to detect objects that generate a high or low number of events, both spatially and temporally [5, 20].

Most related projects in this area focus on contributing a model or event representation to the field, and often both, where the event representation is designed to work specifically with the author's model. For example, the authors of [18] devised a unique event representation and model pairing for event-based object detection. In some cases, such as [9], the authors implemented two different event representations for their model, but the majority of projects do not investigate the pairings of different event representations with their models. In future additions to this work, we will apply multiple different event representations to multiple object detection models in order to find the pairings that yield the best results under lossy conditions.

To reduce latency for downstream applications, event data should be streamed in its most consise form, that is, the events themselves should be streamed, and then converted into an image-like event representation later. Converting the data to an event representation before streaming would result in a high degree of redundancy, requiring a high network bandwidth, and would scale poorly to low-latency applications. Then, streaming loss means that a subset of the events themselves will be lost during streaming. This works seeks to understand how resilient object detection models are to the loss of some events during streaming, and how various event representations handle this loss.

The authors of [2] analyzed the effect of six different subsampling strategies on ResNet34[11] with the Event Spike Tensor event representation [8]. Their subsampling strategies are applied at or near the camera itself, and thus would take place before the streaming of events, acting as a controlled form of loss. Subsampling reduces the number of events that are streamed to a downstream application, but in a predictable way. Of the six subsamping strategies tested, they found that density-based subsampling allows the downstream model to maintain the most accurate predictions despite a significant reduction in the number of events. These findings could guide future works towards intelligent subsampling strategies that reduce the number of events before streaming, making streaming loss more predictable. However, the subsampling strategies used do not offer a clear way to control the maximum allowable bandwidth of data transmission, a factor that is key to event-based video streaming. Our work benchmarks a model and event representation at various user-selected bandwidths, mimicking realistic conditions in video streaming.

3. Methods

3.1. Project Structure

To evaluate the effect of loss on different event representations and object detection models, we constructed a pipeline as shown in Figure 1. Events from an event camera are first fed into the loss module, which decodes the events, applies a configurable loss function to the events, and encodes the events back into a common file format. The raw events are then compressed into an HDF5 format using a modified version of Prophesee's OpenEB Raw to HDF5 conversion utility [1]. The events are then preprocessed into a user-selected event representation and delivered to the desired object detection model for evaluation.

3.2. Loss Module

The loss module is a Rust program that accepts as input a file of raw events from an event camera. The events are first decoded into structs, and the absolute timestamps are

[1] https://github.com/prophesee-ai/openeb

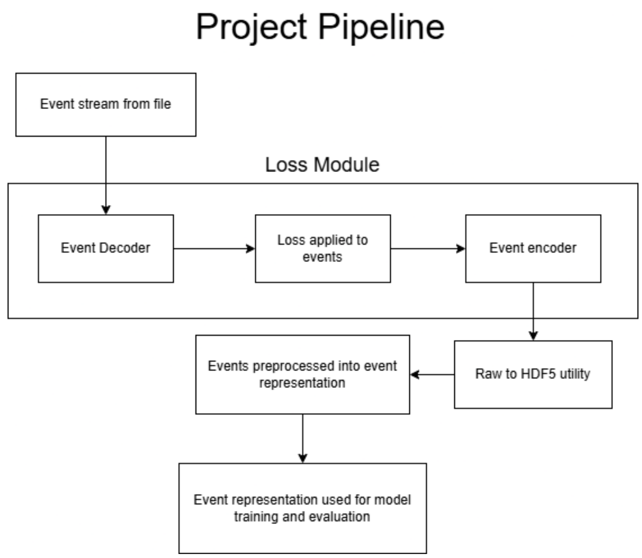

Figure 1. Structure of the project pipeline

calculated from the underlying events [2]. Based on the desired maximum allowable bandwidth and the size of each time chunk, the loss module calculates the maximum allowable number of events per chunk, K. Events exceeding K are discarded. For this project, the time chunk length is set at 50ms to match the time windows used in many object detection models. The loss module currently supports Prophesee's EVT2 file format, while future work on the loss module will expand support to Prophesee's EVT3 and DAT formats.

There are two types of loss currently being investigated in this project. Building on to the work of the authors in [10], the loss module copies events into the output buffer during each time chunk until the threshold K is reached, at which point any additional CD events are discarded. This means that all of the lost events are congregated at the end of each time chunk. We will refer to this type of loss later on as "end biased" loss.

For the second type, the loss module aggregates events in each time chunk, then estimates how many events must be removed in order to maintain the desired bandwidth. It then removes events at roughly equal intervals throughout the chunk, so that the lost events are equally dispersed throughout each time chunk. We will refer to this type of loss later on as "evenly distributed" loss.

Finally, the events are encoded back into the Raw data format, and compressed to an HDF5 format.

[2]https://docs.prophesee.ai/stable/data/file_formats/raw.html

3.3. Event Representations

Before the stream of events can be ingested by an object detection model, it must first be preprocessed into an appropriate event representation. This step typically involves splitting the temporal domain into B temporal bins, and computing a "framed" representation of the events occurring in that temporal bin. In this project, we used $B = 10$ temporal bins, follwing the work of [9].

In this project, we are first investigating the effect of one event representation, Event Histogram [15]. In future works, we will implement the Mixed Density Event Stack [16], Voxel Grid [22], Event Temporal Image [5], and Group Token [18] representations, as well as and others to be determined from community feedback.

3.4. Object Detection Models

For this work, the first model investigated was the state-of-the-art Recurrent Vision Transformer (RVT) object detection model [9]. This model, introduced in 2023, was purpose built for event-based object detection and achieved high accuracy (47.2% mAP on the Gen1 Automotive dataset) while reducing inference time to about 10 ms.

Future works will seek to add additional models, such as the S5-ViT State Space model [23], the Group Event Transformer [18], and YOLOv11 [13].

3.5. Dataset

In this project, we used the eTraM dataset [19] to evaluate the performance of the models and event representations. The eTraM dataset offers 10 hours of video, filmed with a stationary event camera positioned at intersections and roads. It includes both daytime and nighttime footage, at a resolution of 1280x720 pixels. The dataset features annotations for various objects, such as vehicles, pedestrians, and micro-mobility devices like bicycles, wheelchairs, and scooters.

We a subset of the test set for model evaluation. Most files had between 100 and 300 million events and lasted between 1 and 3 minutes in duration. Due to the variable density of events within a file, enforcing a set bandwidth limit affected the total number of events lost in each file differently.

4. Experiments

4.1. Setup

The preprocessing step was implemented as a Python script, adapted from [9]. The stream of events is processed into the user-selected event representation, before being fed into the RVT model. For the evaluation step, we reused the pre-trained model weights from [19].

To evaluate the performance of the RVT model, we performed validation on our test split of the eTraM datase and

Table 1. RVT mAP with the Stacked Histogram event representation and various loss parameters

RVT Mean Average Precision (mAP)									
Loss Parameters			Bandwidth (Mbps)						
Representation	Loss Type	Metric	75	50	25	15	10	5	1
Stacked Histogram	End Biased	mAP	0.3823	0.3707	0.3376	0.2899	0.2432	0.1673	0.0423
		mAP @ 50% IoU	0.7499	0.7389	0.7008	0.6231	0.5398	0.3997	0.1329
		mAP @ 75% IoU	0.3583	0.3399	0.2998	0.2500	0.2029	0.1225	0.0140
		mAP Large	0.5445	0.5295	0.4815	0.3959	0.3135	0.1917	0.0390
		mAP Medium	0.4139	0.4060	0.3825	0.3461	0.3013	0.2178	0.0608
		mAP Small	0.1824	0.1675	0.1236	0.0670	0.0470	0.0312	0.0100
	Evenly Distributed	mAP	0.3811	0.3686	0.3340	0.2899	0.2485	0.1703	0.0468
		mAP @ 50% IoU	0.7481	0.7365	0.6915	0.6156	0.5468	0.4034	0.1382
		mAP @ 75% IoU	0.3551	0.3375	0.2961	0.2538	0.2112	0.1296	0.0179
		mAP Large	0.5446	0.5265	0.4696	0.3980	0.3269	0.2011	0.0348
		mAP Medium	0.4133	0.4048	0.3798	0.3439	0.3067	0.2216	0.0673
		mAP Small	0.1792	0.1615	0.1213	0.0660	0.0482	0.0291	0.0127

reported the overall mean average precision (mAP), as well as mAP at 50% Intersection over Union (IoU), mAP at 75% IoU,and mAP on large, medium, and small objects.

We evaluated one object detection model (RVT), with one event representations (Stacked Histogram), with maximum allowable bandwidths of 75, 50, 25, 15, 10, 5, and 1 Mbps, and with both kinds of loss as described in Section 3.2.

The experiments were run on a high performance computing cluster running CentOS with NVIDIA P100 and V100 GPUs.

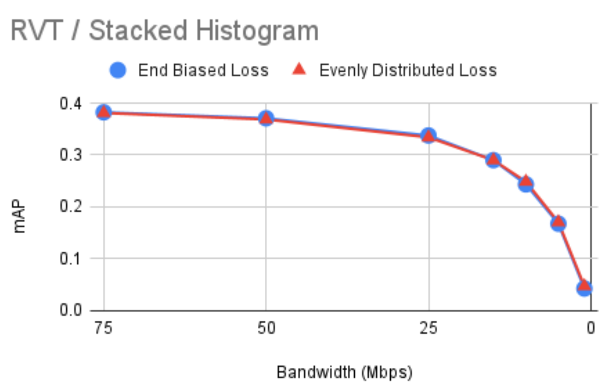

Figure 2. Object detection performances decreases faster as the bandwidth decreases

4.2. Preliminary Results

Table 1 presents the resulting mAP of the RVT model with Stacked Histogram event representation on the eTraM test set with varying degrees and types of loss. From the table, and from Figure 2, we see that the mAP decreases slowly at first as bandwidth decreases, indicating that RVT is resilient to loss at high bandwidths. As the maximum allowable bandwidth drops below 25 Mbps, the mAP drops off steeply. This suggests that a fair amount of events can be dropped from the event stream, while having a minimal effect on downstream accuracy. A large raw data file streamed at a bandwidth of 75 Mbps required 5558 Mbits, while the same file streamed at 25 Mbps required only 3096 Mbits, a 44% decrease in data transmitted. From Table 1, we see that there is only a 4% decrease in mAP overall between those two bandwidths.

We also see a very slight difference of mAP between end-biased and evenly-distributed loss. At higher bandwidths, end-biased loss gives slightly better accuracy, while at lower bandwidths, evenly-distributed loss gives slightly better accuracy. This difference is negligible at a 50ms time chunk, but could become relevant for larger time chunks, where end-biased loss becomes more imbalanced.

5. Conclusion

This paper presented an open-source pipeline for evaluating event-based object detection models with different event representations and loss parameters. We investigated the RVT model with the Stacked Histogram event representation, finding that a large proportion of events can be discarded resulting in only a small decrease in downstream accuracy.

Future work will expand on this project to include additional models, types of streaming loss and subsampling techniques, like those suggested in [2], so that a more complete understanding of lossy event-based object detection can learned. Finally, we will use an additional dataset, the

Gen1 Automotive dataset from Prophesee [4]. This dataset contains 39 hours of footage, but at a lower resolution of 304x240, and with just 2 object classes, pedestrian and car. This dataset will allow us to investigate the resilliency of object detection models to low-resolution data streaming loss, where events lost in the spatial domain could have a larger detrimental effect to overall object detection. The field is ripe for future research in this area, but this work provides a solid baseline for other projects to build off of and compare against, helping researchers around the world better understand the nature of event-based streaming loss.

References

[1] Iñigo Alonso and Ana C. Murillo. Ev-segnet: Semantic segmentation for event-based cameras, 2018. 1, 2

[2] Hesam Araghi, Jan van Gemert, and Nergis Tomen. Making every event count: Balancing data efficiency and accuracy in event camera subsampling, 2025. 2, 4

[3] Marco Cannici, Marco Ciccone, Andrea Romanoni, and Matteo Matteucci. A differentiable recurrent surface for asynchronous event-based data, 2020. 1, 2

[4] Pierre de Tournemire, Davide Nitti, Etienne Perot, Davide Migliore, and Amos Sironi. A large scale event-based detection dataset for automotive, 2020. 5

[5] Liangwei Fan, Yulin Li, Hui Shen, Jian Li, and Dewen Hu. From dense to sparse: Low-latency and speed-robust event-based object detection. *IEEE Transactions on Intelligent Vehicles*, 9(10):6298–6312, 2024. 1, 2, 3

[6] Andrew C. Freeman. Scalable event-based video streaming for machines with moq. In *Proceedings of the 4th Mile-High Video Conference*, page 60–66, New York, NY, USA, 2025. Association for Computing Machinery. 1

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

[8] Daniel Gehrig, Antonio Loquercio, Konstantinos G. Derpanis, and Davide Scaramuzza. End-to-end learning of representations for asynchronous event-based data, 2019. 1, 2

[9] Mathias Gehrig and Davide Scaramuzza. Recurrent vision transformers for object detection with event cameras, 2023. 2, 3

[10] Andrew Hamara, Benjamin Kilpatrick, Alex Baratta, Brendon Kofink, and Andrew C. Freeman. Low-latency scalable streaming for event-based vision, 2024. 3

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 2

[12] Timo Horstschaefer Henri Rebecq and Davide Scaramuzza. Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 16.1–16.12. BMVA Press, 2017. 1

[13] Rahima Khanam and Muhammad Hussain. Yolov11: An overview of the key architectural enhancements, 2024. 3

[14] Xavier Lagorce, Garrick Orchard, Francesco Galluppi, Bertram E. Shi, and Ryad B. Benosman. Hots: A hierarchy of event-based time-surfaces for pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(7):1346–1359, 2017. 1

[15] Ana I. Maqueda, Antonio Loquercio, Guillermo Gallego, Narciso Garcia, and Davide Scaramuzza. Event-based vision meets deep learning on steering prediction for self-driving cars. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, page 5419–5427. IEEE, 2018. 1, 2, 3

[16] Yeongwoo Nam, Mohammad Mostafavi, Kuk-Jin Yoon, and Jonghyun Choi. Stereo depth from events cameras: Concentrate and focus on the future. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6104–6113, 2022. 2, 3

[17] Paul K. J. Park, Baek Hwan Cho, Jin Man Park, Kyoobin Lee, Ha Young Kim, Hyo Ah Kang, Hyun Goo Lee, Jooyeon Woo, Yohan Roh, Won Jo Lee, Chang-Woo Shin, Qiang Wang, and Hyunsurk Ryu. Performance improvement of deep learning based gesture recognition using spatiotemporal demosaicing technique. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 1624–1628, 2016. 1

[18] Yansong Peng, Yueyi Zhang, Zhiwei Xiong, Xiaoyan Sun, and Feng Wu. Get: Group event transformer for event-based vision, 2023. 1, 2, 3

[19] Aayush Atul Verma, Bharatesh Chakravarthi, Arpitsinh Vaghela, Hua Wei, and Yezhou Yang. etram: Event-based traffic monitoring dataset, 2024. 3

[20] Dongsheng Wang, Xu Jia, Yang Zhang, Xinyu Zhang, Yaoyuan Wang, Ziyang Zhang, Dong Wang, and Huchuan Lu. Dual memory aggregation network for event-based object detection with learnable representation, 2023. 1, 2

[21] Yanxiang Wang, Bowen Du, Yiran Shen, Kai Wu, Guangrong Zhao, Jianguo Sun, and Hongkai Wen. Ev-gait: Event-based robust gait recognition using dynamic vision sensors. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6351–6360, 2019.

[22] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Unsupervised event-based learning of optical flow, depth, and egomotion, 2018. 1, 2, 3

[23] Nikola Zubić, Mathias Gehrig, and Davide Scaramuzza. State space models for event cameras, 2024. 3