EV-Mask-RCNN: Instance Segmentation in Event-based Videos

Ana Bălțăreț
Supervisor(s): Nergis Tömen, Ombretta Strafforello, Xin Liu
EEMCS, Delft University of Technology, The Netherlands

A Dissertation Submitted to EEMCS faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 19, 2022

Abstract

Instance segmentation on data from Dynamic Vision Sensors (DVS) is an important computer vision task that needs to be tackled in order to push the research forward on these types of inputs. This paper aims to show that deep learning based techniques can be used to solve the task of instance segmentation on DVS data. A high performing model was used to solve this task, using event-based data that was transformed into RGB-D images. The chosen model for this work was Mask R-CNN, with an alteration for depth images, because of its high performance on frame based data. The N-MNIST dataset provides the event-based input, and the transformation of such an input is presented in this study. Furthermore, the masks are generated with the help of the MNIST dataset and heuristics are used for placing them at the correct positions. The results are promising and comparable to other results from literature on the task of semantic segmentation. The code is available on GitHub[1] and some qualitative results can be viewed on YouTube[2].

Keywords: Event Cameras, Instance Segmentation, Dynamic Vision Sensor, Mask R-CNN.

1 Introduction

Instance segmentation is the computer vision task that deals with finding masks for certain objects in images. For context, bounding box detection finds boxes that surround some objects, and semantic segmentation distinguishes what class each pixel belong to (e.g. balloon, background). Instance segmentation is the combination of bounding box detection and semantic segmentation, and it returns the mask of each particular instance as an output (i.e. balloon1, balloon2), a visual comparison is shown in Fig 1.

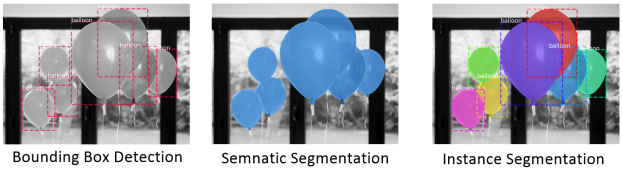

Figure 1: Bounding Box Detection VS Semantic Segmentation VS Instance Segmentation, image taken from [1].

Event cameras (Fig 2) are asynchronous sensors that capture changes in light intensity at every pixel. These event-based cameras have been available to the public since 2008 [2], and more recently larger companies have shown more interest in the technology [3]. The cameras work as follows: for each pixel an event is created when the difference in the logarithm of light intensities crosses a certain threshold. The direction of an event can be either positive or negative, depending on the direction of the light change. The output of these Dynamic Vision Sensors (DVS) is a stream of events that include the pixel position (x,y), polarity (p), i.e. the pixel is changing to either a darker or lighter value, and a timestamp (t), denoted as $e(x, y, p, t)$. The reason for having a stream of events is to allow each pixel to record events asynchronously, which provides various advantages in comparison to a frame based approach.

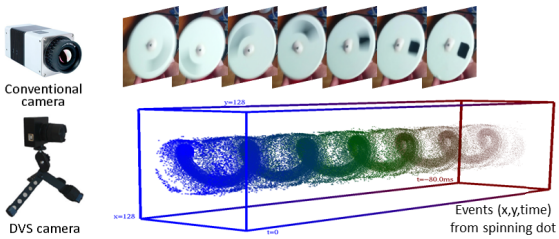

Figure 2: Conventional Frame-based camera (top) vs Event-based camera (bottom) along with the data they record, [4].

Some of the advantages of using these types of inputs are: recording at a high temporal resolution and low latency because each pixel responds asynchronously to light stimuli, and a pixel does not have to wait for other pixels in order to trigger an event. Another advantage is the high dynamic range [5], which means it can handle very bright or very dark shots better than frame-based cameras. On the other side, there are some challenges with these cameras, because they are relatively new [2]. So far they have been mainly used in robotics related tasks and not so much for solving other computer vision problems, as mentioned in [5], thus there are not that many sources or algorithms developed specifically for solving tasks such as instance segmentation. Another challenge is the lack of labeled data, which can pose huge delays when attempting to train and compare the performance of models. Lastly, there is no official representation for the stream of events that is fully compatible with neural networks and which utilizes the advantages of event-based cameras.

The main reason behind using event-based data is that it contains a more sparse representation than frame-based data, which can make it easier and faster for a deep network to recognize some object, because it has less information that could confuse the model. Another reason is the fact that event-based cameras have a high dynamic range, which means they can still properly function in brighter or darker environments. This advantage could provide, for example, more accurate object detection during the night, which could lead to improvements in the autonomous driving sector.

In this paper the objective is to show that event-based cameras can be used more widely in the field of computer vision, not only in robotics, and more specifically to solve the task of instance segmentation. To the best of our knowledge, there are currently no models that do instance segmentation on event-based cameras, and the papers with the closest topic are on semantic segmentation [6], [7].

The aim of this research is to answer the following question: *"Can we train deep networks to do instance segmentation on event-based cameras?"*. This requires making decisions in terms of the way the event-based data is represented, what model is used for training and how the performance is evaluated.

[1]Code is available on GitHub: https://github.com/ana-baltaretu/instance-segmentation

[2]Qualitative results: https://youtu.be/MtkTWbTHgzQ

The main contributions from this paper are:

1. Generating masks for digits from the event-based dataset, N-MNIST [8], by making use of the original frame based dataset, MNIST [9].
2. Converting the event-based data to an RGB-D format such that the Mask R-CNN model [10] could be trained on it.
3. Generating and training a dataset with several digits in the same image to demonstrate that the model still performs well, even with additional data.

The remainder of this paper is structured as follows: in Section 2 there is a deeper dive into the already existing research that relates to this paper. Afterwards, Section 3 explains the idea behind combining Mask R-CNN with event-based data. Following that, in Section 4 the methodology which was used in order to answer the question is presented. Section 5 explains the setup of experiments and the results are showcased. In Section 6 and 7 there is a reflection on the ethical aspects of the research and some comments on the work. Finally Section 8 concludes the paper and mentions possibilities of future research.

2 Related work

In order to get acquainted with the topic of event-based cameras and how they work the paper [5] was used as a starting point for this research. It provides a comprehensive overview of what has been attempted in the field of DVS cameras as of August 2020. The advantages and challenges of these cameras are explained in the paper, as well as how the different brands of cameras record and store data. It also showcases multiple ways in which event-based data can be represented, some of which have been considered when exploring the topic of this paper. For example the "Time surfaces" are very similar to the Depth Frames that were created in this paper to make use of the time information provided by the cameras. Some differences are that the Depth Frames are normalized to a fixed size interval and they contain all of the events, not only negative events. The paper [5] also shows that most of the focus for DVS camera applications was set on robotics tasks, and less on tasks such as instance segmentation, with the closest mention of it describing motion segmentation.

One relevant paper is [6] which shows promising results by using a CNN and doing semantic segmentation on the DDD17 dataset [11]. The paper shows how to generate labels for the dataset, which provided some inspiration to attempt to auto-generate labels in this research, instead of manually labelling the dataset. Their results are further improved by [7] in which a Teacher network is used to train a Student network through knowledge distillation, which is further explained in that paper, and attempts to solve the tasks of semantic segmentation and object detection on the same dataset. Although semantic segmentation is closely related to instance segmentation, the latter also needs to identify each instance of a class, not only classify the pixels in an image. Therefore the results in this paper cannot be compared directly with the results in the previously mentioned papers, but they can be at least used to set some expectations in terms of performance. The previously mentioned papers were also used to gather some ideas on how to solve such a task and they helped with the decision of which metrics should be used to compare results. A comparison of the results using the various metrics is shown later in this paper.

The dataset that is used in this paper is based on the N-MNIST (Neuromorphic MNIST) dataset from [8]. MNIST is a large dataset of hand-written numbers, which has been used in the past as a point of comparison when training models. The neuromorphic version of the dataset, N-MNIST, converts the initial 28x28 pixel images into event-based data by making use of a DVS sensor attached to a mechanism which moves it to set locations. This movement is formed by three saccades (Fig 3), and events get generated at the edges of the numbers. The choice to move the camera instead of moving the MNIST digit on the screen was made to take advantage of properties that the event based cameras offer, and is further explained in the N-MNIST paper [8].

Figure 3: Saccades on the N-MNIST dataset, red=on (pixel changes from black to white), blue=off (pixel changes from white to black), from [8].

The contents of paper [12] were of great help when deciding on the representation of the input. For this paper the two-channel images, shown in Fig 4, were chosen because they represent the event-based data in a CNN-friendly format. More specifically the two-channel images have constant shapes, no matter how many events get triggered in the selected time window, and they keep track of the polarity of events. In their paper the Red color channel was used to represent the negative polarity and the Blue color channel was used to represent the positive polarity, and maximum values on the respective channel were given to the pixels that get triggered during the time interval of creating the images.

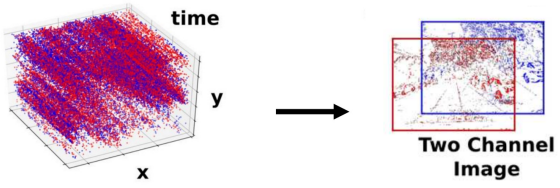

Figure 4: Event representation as two-channel image, [12].

Mask R-CNN [10] is a popular framework that has been used for many tasks that are related to instance segmentation. It is based on another network, called Faster R-CNN [13], which solves the task of proposing approximate regions where objects are located. In addition to that, Mask R-CNN has a branch which predicts segmentation masks (Fig 5).

Newer developments with this algorithm have used 4-channel images, to take advantage of depth information provided by RGB-D cameras [14].

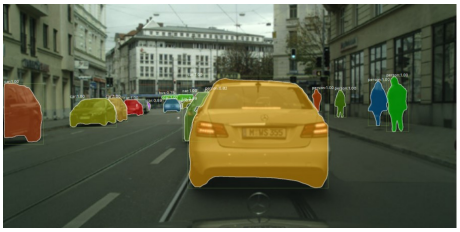

Figure 5: Instance segmentation with Mask R-CNN, [10].

3 Intuition behind EV-Mask-RCNN

The hypothesis when starting this study was that the event-based data could be transformed in such a way that it could be fed into a deep network and have it predict the masks of the digits from the N-MNIST dataset [8]. To prove this hypothesis the input was transformed into RGB-D images, ground truth masks were automatically generated for each frame, then this information was fed into the already existing Mask R-CNN model [10]. The results were then compared to results from different training settings for the same model, but also with results from papers that tried to solve a similar task, semantic segmentation.

In order to make use of the properties of event cameras, such as recording the time at which an event happens and keeping track of the polarity of an event, the decision was made to use the RGB-D channels of an image. The polarity of each event is represented through its own color channel, with the Red color channel representing off-events (pixel changing from white to black). This usually happens at the edges of the digit in the direction in which it is moving, although sometimes events do randomly get triggered on the screen because of sensor noise. The Blue color channel was used to represent on-events (pixels changing from black to white) which happens at the opposite edges of the red events. Lastly the Depth channel was used to represent the position in time at which some events happen. "Older" events have lower values in the generated depth frames and "younger" events have larger values, with the idea that the latter ones would contribute more when deciding where to place the mask.

4 Methodology

This section starts with a more in-depth look at how the instance segmentation dataset and masks were created, then moves on to a description of how the model was trained and evaluated, while the concrete findings are presented in Subsection 5.2.

4.1 Single digit dataset generation

The event-based data comes from the N-MNIST dataset [8], which was created by recording the MNIST digits displayed on a screen, while moving the DVS camera. The N-MNIST dataset was used to create the RGB-D images and masks used in this paper. The N-MNIST dataset contained labels for which class an entry belongs to, which could be used for solving a classification task, but these were not enough for training a model to do instance segmentation.

Because the Mask R-CNN model requires actual images as an input, a first step was to create the 34x34 RGB frames. This was done by taking the events from a fixed sized time window and depending on whether their polarity was positive or negative the frame was colored with red or blue at that pixel position, shown in Fig 6. If an event was triggered at the same location during that time window, the latest value was kept.

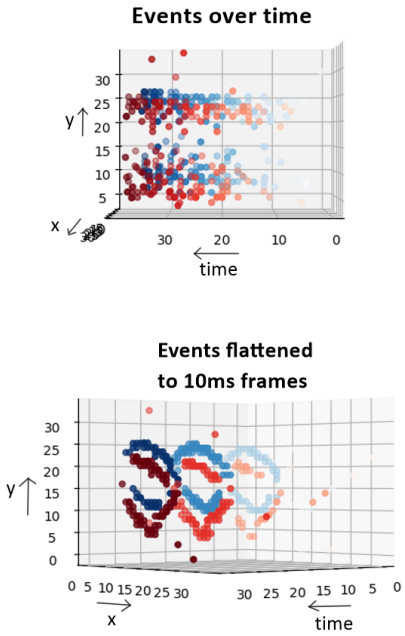

Figure 6: Generating frames by flattening the events from a 10ms time window, first image contains events as they are recorded by the DVS sensor, while the second one shows them flattened.

Furthermore, in order to make use of the time information that event cameras record when an event is detected a depth frame is created, shown in Fig 7. In these frames events that happened first have lower values (closer to black) and events that occur later have higher values (closer to white). The depth frames were normalized through contrast stretching, such that even if the length of the time window is changed later on, the depth frames have values in the range of 0 to 255.

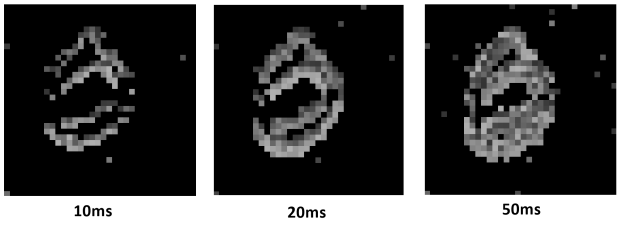

Figure 7: Depth frames for 10, 20 and 50 ms time windows, lighter pixels are more recent events, darker pixels are older events, and black pixels mean no event was triggered there.

An important step prior to training was generating masks for the given data. First of all the frames were denoised and the center of the digit was calculated based on the mean of the location of the minimum and maximum events detected on the X and Y axis. Afterwards the dimension of a digit was determined through taking an average over multiple frames which contained the same digit. To create realistic masks the MNIST dataset was overlaid at these calculated positions. This was achieved because the authors of [8] kept the digits in the same order, allowing the N-MNIST indices to be matched to their MNIST masks.

A more difficult part was aligning the masks with the event data, since it was not feasible to have a human manually placing the masks, this had to be automated. The main idea was to use the centers previously generated and slightly shift the masks vertically, horizontally or diagonally to better "match" them with the events. To determine how well the masks match when placed at some position the sum of pixels from the intersection between the mask and the red pixels (events triggered by changing from white to black) was calculated. This is because the edges of the digit are located at the location in which the latest red pixel appear and for short time windows previously triggered events are a good indication of where the digit was. Some examples of the final generated RGB-D frames and their masks can be seen in Fig 8 and in Fig 13.

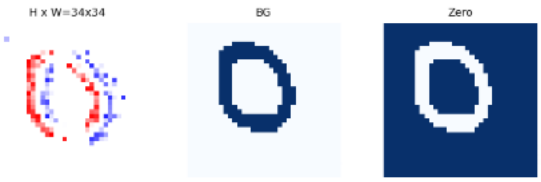

Figure 8: Training masks generated for one entry, left is the RGB-D input frame, middle is the mask of the background and right is the mask of the digit.

4.2 Multiple digits dataset generation

Generating a dataset that contains multiple digits in a single frame was comparatively easier to do once all of the frames, masks and depth images were generated, as explained in Subsection 4.1. To make this new dataset, new images of size 64 by 64 pixels were created, 1 digit was taken in order of the IDs of the images and 3 other digits were chosen randomly from the base dataset. These four digits were then placed at the set locations of $[(0,0),(0,32),(32,0),(32,32)]$, and because the initial N-MNIST images were of size 34x34, 1 row and 1 column were removed from each edge of an image to not lose much information from either one of the digits. Lastly, to easily match each mask with its digit when loading them: the digits were stored and the masks were color-coded, with the color indicating which digit they should be used for, examples can be seen in Figure 9.

4.3 Training the model

The MNIST dataset was already split into training and testing, and to further separate the training results from the final results, the decision was made to divide the training dataset into 80% training data, and 20% validation data, while keeping the testing dataset entirely separate and using it to calculate the results after the model was trained. The training and testing was done on multiple time windows, more specifically on 10 ms, 20 ms and 50 ms time windows such that it could be compared with other models. The decision was made to not go over a 50 ms time window because the size of an image is relatively small and at 50 ms a number is clearly visible, usually without any sparsity. The difference of how the frames look like for various time windows is shown in Fig 10.

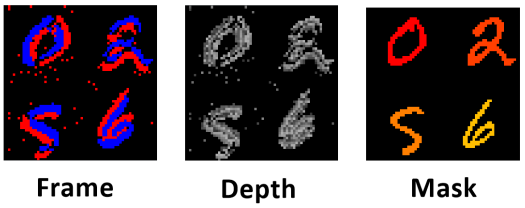

Figure 9: Example of polarity frames, depth images and color-coded masks, generated for images with multiple digits.

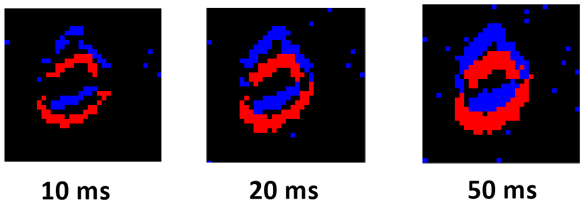

Figure 10: Comparison between how frames look for time windows of 10 ms, 20 ms and 50 ms, polarity is shown only, no depth applied.

To avoid detecting masks which are not significant enough, the "DETECTION_MAX_INSTANCES" variable from the model configuration was set to maximum 1 mask for the training with individual digits in one frame, and it was set to 4 when training with multiple digits.

4.4 Comparing results

The comparison to other sources is limited because of how niche the performed experiment is, and at the time of writing this paper, no direct comparisons could be found in the literature. Therefore, the evaluation mainly focuses on seeing how different adjustments to the input relate to the output scores. Another point of comparison is seeing how other models trained to do semantic segmentation compare to the Ev-Mask-RCNN model trained to do instance segmentation, because some metrics can be related between the two tasks.

The evaluation metrics that were used in the experiment are accuracy (Acc), mean intersection over union (MIoU) and mean average precision (mAP), because these metrics are widely used in other papers, especially those that focus on semantic segmentation, like [6] and [7].

Accuracy, inspired by [6], is defined as the sum of pixels that have the same class (indicated by the Kronecker delta function, δ) in the prediction (y_i) and in the ground truth ($\hat{y_i}$), which is then divided by the total number of pixels in the image (N), shown in Equation 1. This metric was used to give an indication of how many pixels are properly categorized. It is not that representative for singular digits because most

pixels are categorized as "background", but it is more representative for a scene with more entries, for example for the multiple digits dataset.

$$Accuracy(y_i, \hat{y_i}) = \frac{1}{N} \sum_{i=1}^{N} \delta(y_i, \hat{y_i}) \quad (1)$$

Intersection over Union (IoU) is defined as the area of intersection divided by the area of union between the ground truth and predicted mask, similarly to how it is defined in [15] for bounding boxes. The metric used in this paper is the Mean Intersection over Union (MIoU), which is just the average over all classes between these predictions, or 0 if the correct digit is not predicted at all. The reasoning behind using this metric was that it could better show when a mask is properly placed on a digit, in comparison to accuracy.

The mean average precision (mAP) metric is based on the IoU and calculates the score using precision and recall. This metric was used in the Mask R-CNN paper [10] and to evaluate the performance of a model on the COCO dataset [16], explained in more detail in this article [17]. This metric was used because it was already implemented from the Mask R-CNN code, it is based on MIoU and it could be used as a point of comparison for future papers.

The testing dataset, mentioned in the previous subsection, was used to create the results that can be seen in Subsection 5.2, and this entire dataset was kept separate from the training data.

5 Experimental Setup and Results

The concrete setup that was used for the experiment, including configuration settings for Mask R-CNN is described in Subsection 5.1, followed by quantitative and qualitative results that were found on various time windows shown in Subsection 5.2.

5.1 Experiment setup

The model was trained and tested on a HP ZBook Studio x360 G5 laptop, using the Central Processing Unit Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz.

The model used for training is Mask R-CNN [10], and some relevant parts of the custom configuration are: its backbone is based on Resent-50 [18], it uses 4 image channels (RGB-D) anchor scales of 4, 8, 16, 32, 64. The size of an image that Mask R-CNN gets trained on is 64x64 even though the images with individual digits are 34x34, they get scaled up because they need to have powers of 2 as the anchors. The images that contain multiple digits are generated as 64x64 images to begin with and digits trimmed to 32x32 are placed in each corner. The number of classes used for training is 11 (10 digit classes and background) and the model was trained using a learning rate of 0.001 over 5 epochs, which was then reduced to 0.0001 over 10 more epochs. The weights were saved after epoch 2, 5 and 15, to be able to compare the performance on the different metrics.

Transfer learning was used from the weights of the COCO dataset [16] because this model was already trained to detect other classes in an image, and with pre-trained weights the training process requires less time or training data. This was not possible on all layers because the shape of the input didn't match, so the "mrcnn_class_logits", "mrcnn_bbox_fc", "mrcnn_bbox", "mrcnn_mask", "conv1" are initialized with random weights and trained from scratch, as suggested in the Wiki page[3] from an implementation of Mask R-CNN.

5.2 Results

Results for the base dataset. The results in Table 1 are measured by using the model to predict the mask on a never-before-seen entry, and repeating this 500 times on random entries from the testing dataset, then taking an average of the result. These results so far are promising and show a trend that with more training time the model performs better in terms of Accuracy, MIoU and mAP, metrics which were explained in Subsection 4.4.

Table 1: Results from running the model over multiple epochs (for each metric the possible values are between 0-100, with 100 being the best).

Metrics \Epochs	2 epochs	5 epochs	15 epochs
Acc (10 ms)	93.33	95.04	95.64
Acc (20 ms)	94.23	95.63	96.29
Acc (50 ms)	94.76	95.27	96.51
MIoU (10 ms)	14.19	41.05	55.47
MIoU (20 ms)	20.48	47.24	58.01
MIoU (50 ms)	27.82	41.73	60.29
mAP (10 ms)	13.4	32.8	42.3
mAP (20 ms)	18.7	37.1	43.2
mAP (50 ms)	23.7	35.2	44.6

The results of training Mask R-CNN on RGB-D images that contain event based data with time windows of 10, 20 and 50 ms are shown in Table 2, and they are presented along with results from papers [6] and [7] to provide a better comparison in terms of performance expectations. Even though it is hard to make a one-to-one comparison because of the different datasets that are used, similarities in terms of Accuracy and MIoU are still important for proper reflection. It might seem from the table that Ev-Mask-RCNN performs better than other models, but this should be taken with a grain of salt because the DDD17 dataset is comparatively more complex than the N-MNIST dataset.

Table 2: Comparison of results from EV-Mask-RCNN with other similar results from literature.

Model & Dataset	Accuracy 10ms	MIoU 10ms	Accuracy 20ms	MIoU 20ms	Accuracy 50ms	MIoU 50ms
EV-SegNet [6] (on DDD17)	86.46	45.85	-	-	89.76	54.81
EVDistill [7] (on DDD17)	-	48.68	-	-	-	57.16
EV-Mask-RCNN (on N-MNIST)	95.64	55.47	96.29	58.01	96.51	60.29

Currently the best MIoU that was achieved through this model is 60.29%, and the best accuracy is 96.51%, both on

[3] Wiki page with suggestions on how to use RGB-D images in Mask R-CNN: https://github.com/matterport/Mask_RCNN/wiki#training-with-rgb-d-or-grayscale-images

50 ms time windows. Qualitative results with relatively good predictions for the average case are shown in Fig 11 and in Fig 14.

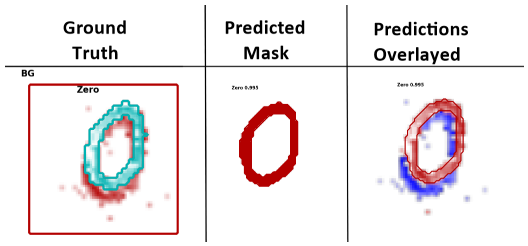

Figure 11: Predicted masks for one entry with a single digit.

Results for the dataset with multiple digits. A new model was trained to predict multiple digits in the same frame, with a small tweak to the settings shown in Subsection 5.1, having the maximum number of predicted masks set to 4 instead of 1. The dataset itself was generated based on the single digit dataset, using the procedure explained in Subsection 4.2. The results from this model are calculated after training it on 15 epochs, and an average of the predictions of each frame is taken for 500 random samples from the validation dataset. The metrics used for this dataset are the same as the ones used for the base dataset: Accuracy, MIoU and mAP, mainly because we want them to be comparable with other papers in the future. The best results for the multiple digits dataset are Accuracy of 95.38%, MIoU of 31.70%, and mAP of 42.48%, with qualitative results being shown in Fig 12 and Fig 15.

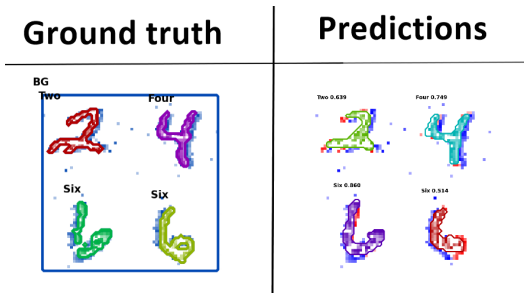

Figure 12: Predicted masks for one entry with multiple digits.

6 Responsible Research

It is of utmost importance to conduct research responsibly because it builds trust in the completed work and enables for collaboration between researchers.

Even though this research does not have any people or sensitive data involved, it is still important to mention that future research might need to consider topics such as responsible machine learning. For example, a possible application of instance segmentation with event based cameras could be pedestrian detection, especially because during the night these cameras might perform better. Special care must be taken for recognizing darker colored skin tones since it might be harder for event based cameras to register an event in such cases. Therefore it is very important to have a balanced dataset and do proper field testing before publicly releasing such a model. No such testing was done in this paper because it focuses on a more simplistic dataset.

In terms of reproducibility, the methodology, documented code[4] and pre-trained weights[5] were made publicly available and the data can be generated by making use of this code, and the N-MNIST dataset [8]. More specifically, the methodology is documented in Section 4 along with the concrete setup that was used to run the experiment. Furthermore, the results are shown in Section 5.

7 Discussion

The choice to use the N-MNIST dataset was taken early on and it was an important one mainly because of the duration of the project. The training time was a constraint because of a lack of a GPU and the small size of images was a positive in this regard. The dataset itself was very simplistic, with minimal noise around the digit, which was usually centered, and mostly visible in all of the generated frames. This visibility could pose problems in a real world setting (for example with a live-recorded dataset) and the choice of a time window, could affect how many frames contain all the required contours for proper detection. For this project it was also very important that the masks could be generated and placed automatically. The benefit of having a single entry in a denoised frame meant that heuristics could be applied for aligning the masks, and the already existing MNIST dataset could be used to create them. The results drastically increased when changing the mask placement heuristics, and, as it is now, the masks seem to be properly placed in almost all of the frames. On a negative note, the results should not be considered as generalizable, for reasons mentioned in Subsection 7.1, mainly because the dataset is very "empty" compared to the real-world setting, but it could be used as a baseline to compare other methods by making use of the same dataset and masks, as MNIST was used in the past for frame-based methods.

7.1 Comments on results

The results for the base dataset as well as the multiple digit dataset are promising and are comparable to results from semantic segmentation on different datasets, shown in Table 2. Even though the setup of this experiment is simplistic, it is noticeable that the model made significant improvements after training on 15 epochs, as it was shown in Table 1. From the experiments it seems that the results are comparatively better than the results of other papers [6], [7], but that might be mainly because of the simplicity of the dataset that is used in this research. The other papers use the DDD17 dataset [11], which is not only more complex because of the size of a frame, but also because the background contains more noise. Therefore the results from Table 2 should not be compared directly, but rather taken as a general guideline of what results we expected.

[4]Link to code on GitHub: https://github.com/ana-baltaretu/instance-segmentation

[5]Pre-trained weights: https://github.com/ana-baltaretu/instance-segmentation/tree/main/release_logs

7.2 Contours or entire objects

One problem with drawing conclusions from these results is that the dataset itself might have been too simplistic to accurately measure the performance of instance segmentation. More concretely, all of the digits are more or less contours, so after training, the model only needs to indicate where these contours are located. One possible point of concern is that for time windows of 20 and 50 ms the masks themselves were just as thick as the red pixels. For these time windows the model got trained to determine that when there are red pixels, the digit is more likely to be present. On the positive note, for the 10 ms time window, the model not only recognized the red pixels as part of the mask, but also the white pixels (where no event got triggered) up until the blue pixels and categorized them correctly. If the objects being detected were larger and no events are triggered in the middle of them, this could indicate that the model could still learn to detect the pixels in between red and blue that help define the shape of the mask. It is clear that the model is good at detecting edges as being part of an object, but it should definitely be investigated further to see whether it can detect entire objects with constant color, rather than just their contours.

8 Conclusions and Future Work

In conclusion the experiment itself provided relatively good results given that the masks themselves are not always perfectly placed, and it shows that instance segmentation is possible on event based data. For thin edged objects, like the numbers, it performs well and it can be used to at least predict the contours if not the entire object.

In terms of future work, one recommendation would be to look into generating a dataset with simplistic shapes, such as a triangle, square, pentagon or others, and using the method from [8] to transform it into an event-based dataset. Following that, the contents of this paper could be directly applied to see if a model can actually be trained to detect the masks of entire objects or just the contours.

Another suggestion would be to either figure out a way to automatically label, or just manually label the DDD17 dataset such that this model could be more or less directly compared to the results of semantic segmentation from [6] and [7].

Lastly a final recommendation would be to take a look at direct comparisons between training on event-based data versus training on frame-based data. For example future projects could look into comparing differences in training time, how fast each model predicts a mask per frame and how accurate each of them are. It would be advisable to use a more advanced dataset for this task, such that the difference in results would be more obvious, and as a starting point, the model used in this paper, Mask R-CNN [10], could be used directly on either frame or event-based data.

References

[1] W. Abdulla, "Splash of color: Instance segmentation with mask r-cnn and tensorflow," Mar 2018. https://engineering.matterport.com/splash-of-color-instance-segmentation-with-mask-r-cnn-and-tensorflow-7c761e238b46.

[2] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 × 128 120 db 15μs latency asynchronous temporal contrast vision sensor," *IEEE journal of solid-state circuits*, vol. 43, no. 2, pp. 566–576, 2008. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4444573.

[3] B. Son, Y. Suh, S. Kim, H. Jung, J.-S. Kim, C. Shin, K. Park, K. Lee, J. Park, J. Woo, *et al.*, "4.1 a 640× 480 dynamic vision sensor with a 9μm pixel and 300meps address-event representation," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 66–67, IEEE, 2017. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7870263.

[4] F. Barranco, C. L. Teo, C. Fermuller, and Y. Aloimonos, "Contour detection and characterization for asynchronous event sensors," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 486–494, 2015. https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Barranco_Contour_Detection_and_ICCV_2015_paper.pdf.

[5] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, *et al.*, "Event-based vision: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 1, pp. 154–180, 2020. https://arxiv.org/pdf/1904.08405.pdf.

[6] I. Alonso and A. C. Murillo, "Ev-segnet: Semantic segmentation for event-based cameras," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019. https://arxiv.org/pdf/1811.12039.pdf.

[7] L. Wang, Y. Chae, S.-H. Yoon, T.-K. Kim, and K.-J. Yoon, "Evdistill: Asynchronous events to end-task learning via bidirectional reconstruction-guided cross-modal knowledge distillation," 2021. https://openaccess.thecvf.com/content/CVPR2021/papers/Wang_EvDistill_Asynchronous_Events_To_End-Task_Learning_via_Bidirectional_Reconstruction-Guided_Cross-Modal_CVPR_2021_paper.pdf.

[8] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers in neuroscience*, vol. 9, p. 437, 2015. https://www.garrickorchard.com/datasets/n-mnist.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=726791.

[10] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017. https://openaccess.thecvf.com/content_ICCV_2017/papers/He_Mask_R-CNN_ICCV_2017_paper.pdf.

[11] J. Binas, D. Neil, S.-C. Liu, and T. Delbruck, "Ddd17: End-to-end davis driving dataset," *arXiv preprint arXiv:1711.01458*, 2017. https://arxiv.org/pdf/1711.01458.pdf.

[12] D. Gehrig, A. Loquercio, K. G. Derpanis, and D. Scaramuzza, "End-to-end learning of representations for asynchronous event-based data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5633–5643, 2019. https://openaccess.thecvf.com/content_ICCV_2019/papers/Gehrig_End-to-End_Learning_of_Representations_for_Asynchronous_Event-Based_Data_ICCV_2019_paper.pdf.

[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015. https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

[14] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3d: Learning from rgb-d data in indoor environments," *arXiv preprint arXiv:1709.06158*, 2017. https://arxiv.org/pdf/1709.06158.pdf.

[15] R. Padilla, S. L. Netto, and E. A. Da Silva, "A survey on performance metrics for object-detection algorithms," in *2020 international conference on systems, signals and image processing (IWSSIP)*, pp. 237–242, IEEE, 2020. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9145130.

[16] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014. https://link.springer.com/content/pdf/10.1007/978-3-319-10602-1_48.pdf.

[17] J. Hui, "map (mean average precision) for object detection - jonathan hui - medium," Mar 2018. https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. https://openaccess.thecvf.com/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf.

A Training masks

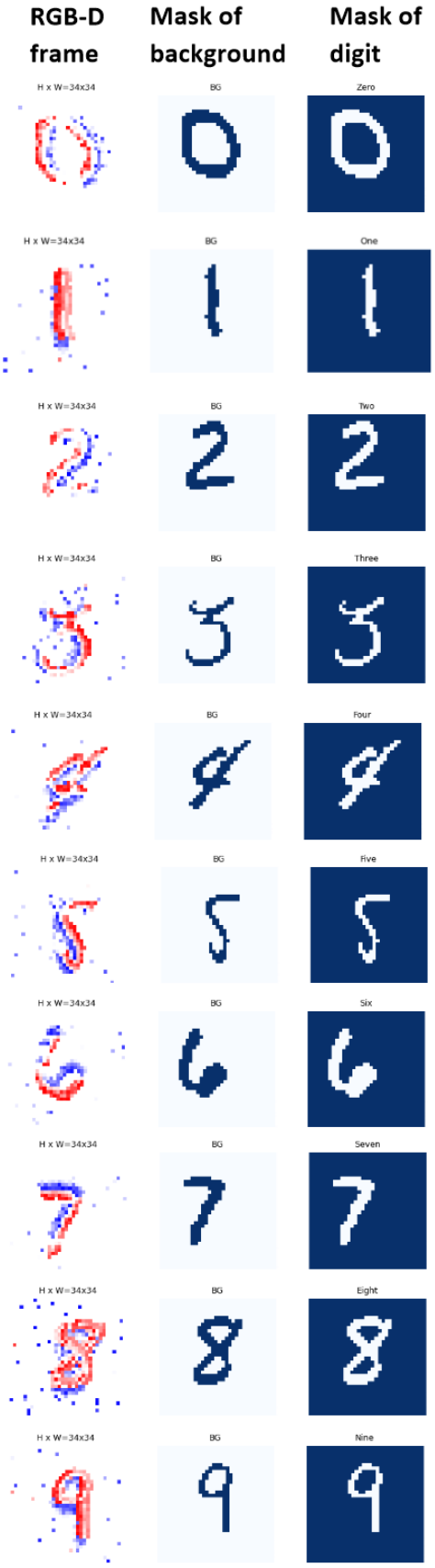

Figure 13: Masks generated for training from each time window of 10 ms, based on event location and MNIST dataset.

B Results for singular digit

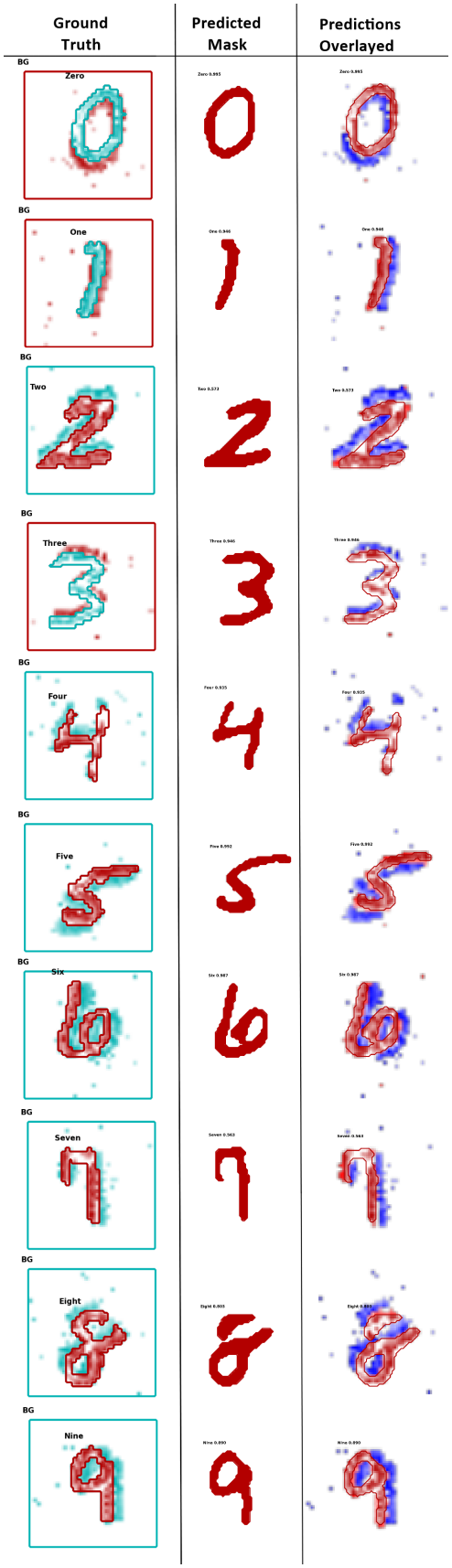

Figure 14: Qualitative results of the masks created by the trained model, for each digit the left column shows the ground truth and right column contains the predictions of the model, with a 10ms time window of events used to generate a frame.

C Results for multiple digits

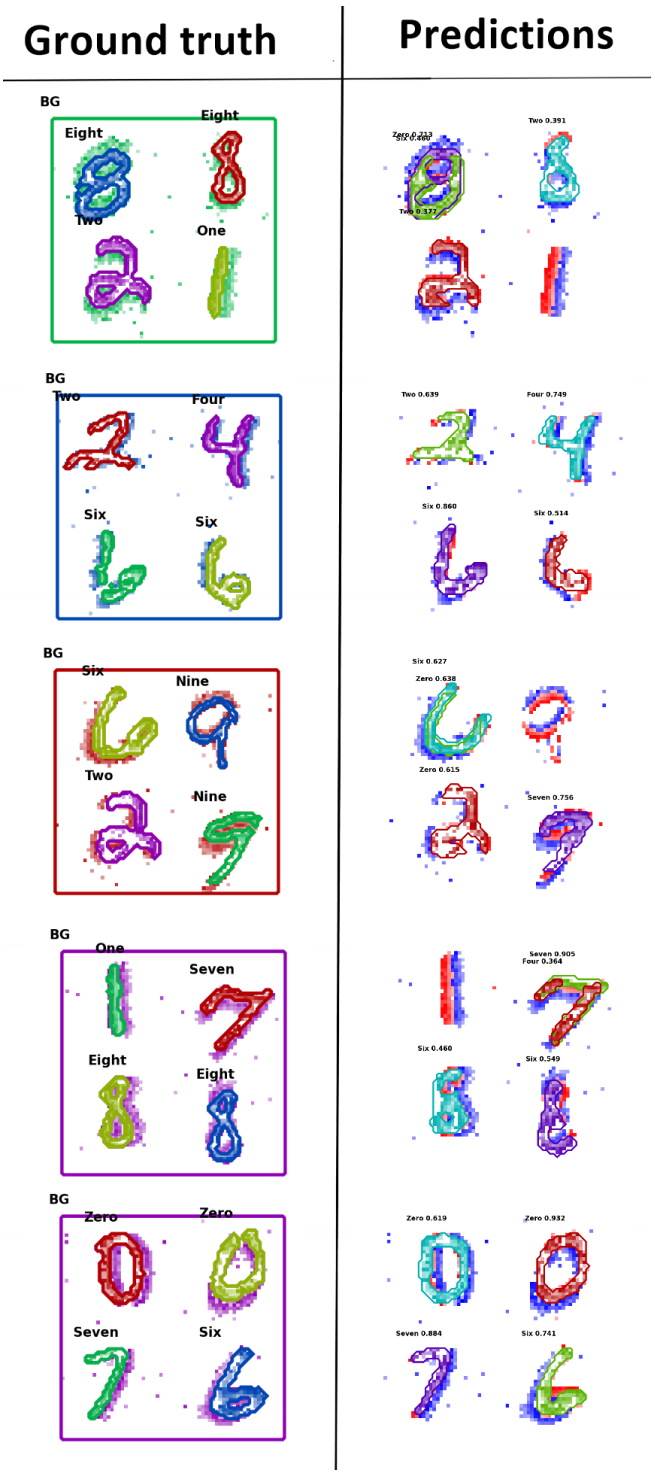

Figure 15: Qualitative results for multiple digits, left column being the ground truth and right column being the prediction, trained for 15 epochs.