University of Zurich[UZH]
Department of Informatics

University of Zurich[UZH]
ETH zürich
Institute of Neuroinformatics

ROBOTICS & PERCEPTION GROUP

Kexin Shi & Yifei Liu

Efficient Spatio-Temporal Processing of Event Data

Semester Thesis

Robotics and Perception Group
University of Zurich

Supervision

Mathias Gehrig
Nico Messikommer
Prof. Dr. Davide Scaramuzza

Feb 2022

Contents

Abstract **iii**

1 Introduction **1**
1.1 Related Work 1

2 Preprocessing **2**
2.1 Downsample 2
2.2 Voxelization 3
2.3 Point Masking 3

3 Comparisons of Different Methods **4**
3.1 Dataset 5
3.2 Classification 5
3.2.1 Number of PVCNN blocks 5
3.2.2 Resolution 5
3.2.3 MLP 6
3.2.4 Concatenation 6
3.3 Regression 7
3.4 Model Comparison 7

4 Explorations in PVCNN **9**
4.1 MLP is not useful inside Point Voxel Block 9
4.2 Devoxelization is useful 10

5 Another Attempt **12**
5.1 Sparse convolution 12

6 Discussion **13**
6.1 Conclusion 13
6.2 Outlook 13

A Downsampling **15**
A.1 Downsample Code 15
A.2 The Effectiveness of Downsample Method 15
A.3 A new downsampling method 15

Abstract

Event cameras are novel sensors that record a stream of asynchronous events and offer advantages of high dynamic range and no motion blur. Events can be converted to voxel grids and be processed by conventional neural networks, or be directly processed by point-based model. Voxel-based approaches are used more often and point-based models are less used, and we want to investigate why this is the case and see the pros and cons between these two types of models. By fairly comparing them within the same datasets and tasks, and within the similar preprocessing methodology, we show that point-based methods can get better performance than voxel-based models, but voxel-based models with 2D convolution have a more reasonable trade-off between performance and speed.

Chapter 1

Introduction

Event cameras are bio-inspired sensors that captures change of intensities asynchronously and results in a stream of events. A stream of events are a sequence of events and each event is represented as {x,y,t,p}, where x and y are pixel location, p is polarity indicating the positive or negative change in intensity, and t the timestamp.
There are many ways to process the event data. One can either directly process the events with a point-based model, or convert the events to voxel grids and process them using conventional neural networks (such as CNNs), or use other methods such as spiking neural networks to process the events asynchronously. The voxel-based method is used most often but point-based method is used less, and we are investigating why this is the case and show the pros and cons between these two types of models.

1.1 Related Work

The point-based models are not often used in processing events, they are more often used in 3D scenarios like Lidars. PointNet [9] is a network that directly processes points, and PointNet++ [10] recursively use PointNet to learn local features from points. One network that uses point-based models on events is EventNet [12], which achieved real-time processing. However, there are few other work using point-based models on events.

Voxel-based models are used often in event data. One can convert points to voxels by either accumulating the polarities or recording the timestamps [3]. After converting events to voxels, one can apply typical convolution networks on the voxels. An off-the-shelf Resnet can be used to learn the voxel representation and can get state-of-art performance [3]] [2].Voxel representations can also be applied by recurrent neural networks, E-RAFT [5] applied recurrent networks on the voxel grids and got high precision in predicting optical flow.

Another type of model uses both points and voxels. PVCNN [6] uses MLPs to learn point features and convolutions to learn voxel features and combine them in one network.

Chapter 2

Preprocessing

2.1 Downsample

Point models are often used in lidar scenarios which have thousands of points, but events have hundreds of thousands or million points and requires more space and time to process. Point-based models can not directly take as input all the events because of memory issues, so downsampling is needed.

We first voxelize points to voxels by interpolating or summing the polarities in each voxel, and then set a threshold to filter out voxels which has low summed polarities. Then we extract points from voxels whose summed value is beyond that threshold. The extraction is done by taking each valid voxel as a single point, using its coordinate and summed polarity. Figure 2.1 visualizes the raw events, randomly downsampled events, and the proposed downsampled events. The randomly downsampled events are noisy, while the proposed downsampling preserves a clear structure and removes noisy or isolated events.

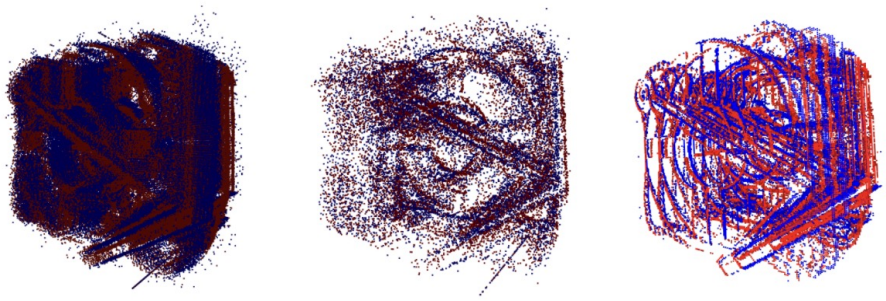


Figure 2.1: Downsample effect. From left to right: raw events, random downsample, proposed downsample

There are three hyperparameters: summing or interpolating polarities, the number of bins and the threshold. We plan to discuss more in Appendix later.

2.2 Voxelization

Point-based models do not need voxelization. For voxel-based models, the input need to have voxel representations. In the classification task, We pre-voxelize the events in the data loading by interpolating the events in to pre-defined temporal bins. In the optical flow regression task, we do not voxelize events in data loading, but use the differentiable and gpu-efficient voxelization layer introduced in PVCNN [6] to voxelize points, in the first layer of the network.

2.3 Point Masking

The voxel representation has a fixed size and can easily form batches for training. But the number of points varies in different samples, and we can not directly collect a batch from different samples. Inspired from machine translation, We propose a method to pad the points to the maximum length of the samples in the batch and generate a point mask for them. In the point mask , 1 indicates original and valid points, while 0 indicates padded and unvalid points.

During training, we pass both the points and the point masks to Point-Voxel Networks. For voxelization layers, we adapt the CUDA code so that the padded points with mask 0 do not contribute to voxelization. Similarly during de-voxelization, padded points are assigned nothing but 0 values. Moreover, we multiply the output of Shared MLPs, i.e. 1D convolution, by the broadcasted point masks, ensuring that the positions in the output feature corresponding to padded points have zero values. These operation ensures that padded points have zero features along the whole network, and stops gradient from passing through these points.

The point masking is very important for Point-Voxel Networks. Figure 2.2 shows our initial experiment using point masks. Before we use masks, we padded points at random locations with fixed zero polarity, and the training was unstable and hard to converge. After we used the point masks, the training was stablized and the validation accuracy got improved. In the context of this project when we mention Point-Voxel Networks, including PVCNN and PVUNET, we implicitly mean we use this point masking strategy in preprocessing.

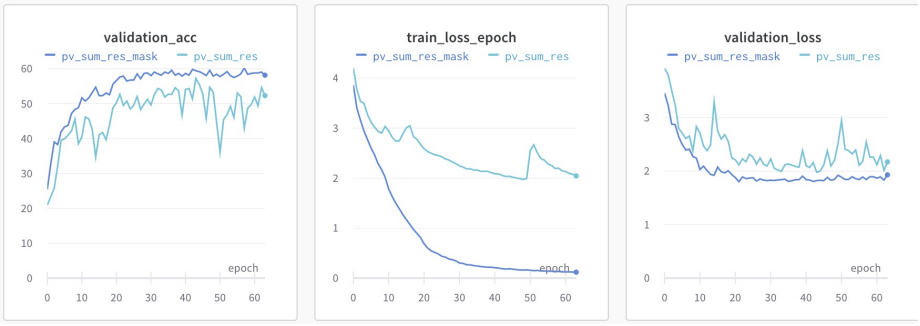


Figure 2.2: Effect of adding point masks. Light blue: before adding masks. Deeper blue: after adding masks.

Chapter 3

Comparisons of Different Methods

In this chapter, we will discuss the performances comparisons of different methods. We have four different types models to compare in total: Voxel-based 2D models (2D Resnet, 2D Unet [11]), Voxel-based 3D models (3D Resnet, 3D Unet), Point-based 3D models (PointNet++ [10]), Point-Voxel CNN (PVCNN [6]). We will compare their performances and speeds in classification and regression tasks respectively.

A basic PVCNN block is shown in Fig. 3.1. It contains two parts. The upper part represents voxel-based feature aggregation to extract coarse-grained features, which includes voxelizing points into voxels, 3D convolutions on voxels, and devoxelizing voxels into points again. The lower part represents point-based feature transformation to extract fine-grained features, which only includes Multi-Layer Perception (MLP) on points directly.

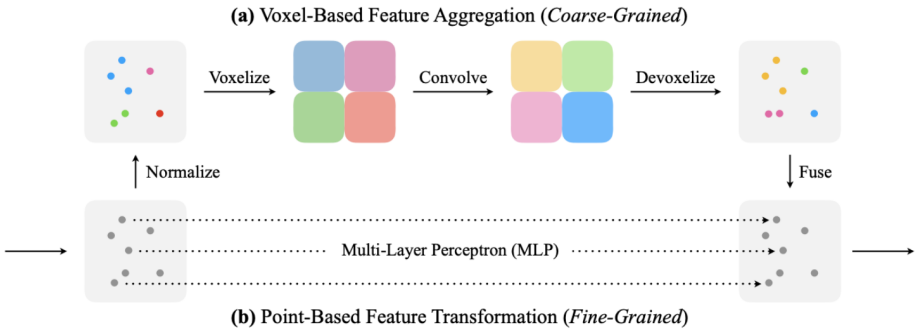


Figure 3.1: A basic PVCNN block.

3.1 Dataset

In this project we use two datasets for two tasks. We use N-Caltech101 [8] for object classification task, and DSEC [4] for optical flow regression. We only use events as input in both dataset. And we report model performance on N-Caltech101 in 3.2, and report performance on DSEC in 3.3.

3.2 Classification

In classification task, a single basic Point-Voxel block is not representative enough. So we stack multiple basic Point-Voxel blocks sequentially at different resolutions to predict labels.

3.2.1 Number of PVCNN blocks

The first question we need to solve is to find how many blocks do we need per resolution. We compared the results of PVCNN 1x (one PVCNN block per resolution) and PVCNN 2x (two PVCNN blocks per resolution). The results are shown in Table 3.1.

Nb. blocks	**Accuracy**	**Nb. params**	**Speed(Instances/sec)**
PVCNN 1x	67.85	**5.6M**	**29.72**
PVCNN 2x	**71.81**	10.4M	24.4

Table 3.1: Performances of Nb. of PVCNN blocks.

Definitely, the more layers we use, the better performance we may achieve. But it will also have more parameters and lower speed. To make a trade-off, we decide not to add more blocks and use PVCNN 2x to finish following experiments.

3.2.2 Resolution

In sequential model, the resolutions in different PVCNN blocks will be smaller and smaller while the channels will increase gradually. The next question is which resolution to start will be the best? We have tried several experiments to find it. The results are shown in Table 3.2.

Resolutions	**Accuracy**	**Nb. params**	**Speed(Instances/sec)**
1/2→1/4→1/8	69.89	10.4M	18.32
1/4→1/8→1/16	71.81	10.4M	24.4
1/8→1/16→1/32	**78.506**	10.4M	25.36
1/16→1/32→1/64	69.89	10.4M	**29.24**

Table 3.2: Performance of different resolutions.

In this section, we only change the resolution of voxels. Voxelization and devoxelization do not contribute parameters and that is why the parameters are same. On the other hand, the smaller resolution, the less convolutions we need

to perform. That is why the speed is faster. But it does not always hold that the smaller resolution we use, the larger receptive field, the better performance. The best resolution is starting from 1/8 of original resolution and then with 1/16, 1/32.

3.2.3 MLP

In the original PVCNN paper, the author used two additional MLP structures after all PVCNN blocks to aggregate information in classification task. As for our event object classification task, we also did same experiments to explore whether these two MLPs are useful or not. The results are shown in Table 3.3.

MLPs	**Accuracy**	**Nb. params**	**Speed(Instances/sec)**
w/ MLPs	**78.506**	10.4M	25.36
w/o MLPs	76.897	**9.1M**	**36.12**

Table 3.3: Effects of MLPs

From the results, we can conclude that the last two MLPs can help integrate information effectively and improve performances a lot with sacrificing speed.

3.2.4 Concatenation

In the original PVCNN paper, the author concatenates all point features from all PVCNN blocks and also the last two MLP parts. It will help further aggregate information intuitively but also slow down the process. Therefore, we want to explore the necessity of concatenation in event data. The results are listed in Table 3.4.

Concatenation	**Accuracy**	**Nb. params**	**Speed(Instances/sec)**
w/ Concatenation	**78.506**	10.4M	25.36
w/o Concatenation	76.322	**9.7M**	**28.88**

Table 3.4: Effects of Concatenation

The result of model with concatenation is significantly better than model without concatenation, because it is useful for integrating information from point-based features at different resolutions, especially in classification task.

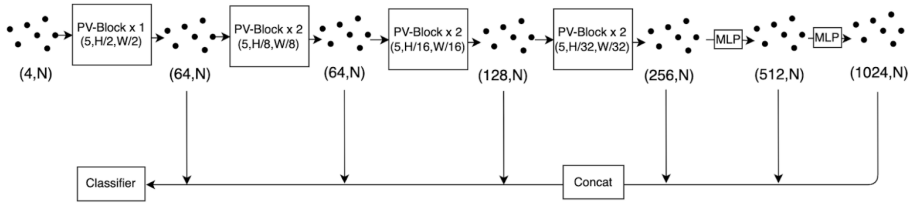


Figure 3.2: Final PVCNN model.

By summarizing all we have discussed before, the final PVCNN model we used for classification task is shown in Fig. 3.2. This model finally can achieve accuracy of 78.506% with 25.36 instances per second.

3.3 Regression

In regression task, we need to predict the optical flow for each pixel of the entire image. Thus the sequential model does not fit for this task. Inspired by Unet, we modify PVCNN model to encoder-decoder structure and the hype-parameters are also similar to Unet, which is shown in Fig. 3.3. We call it PV Unet.

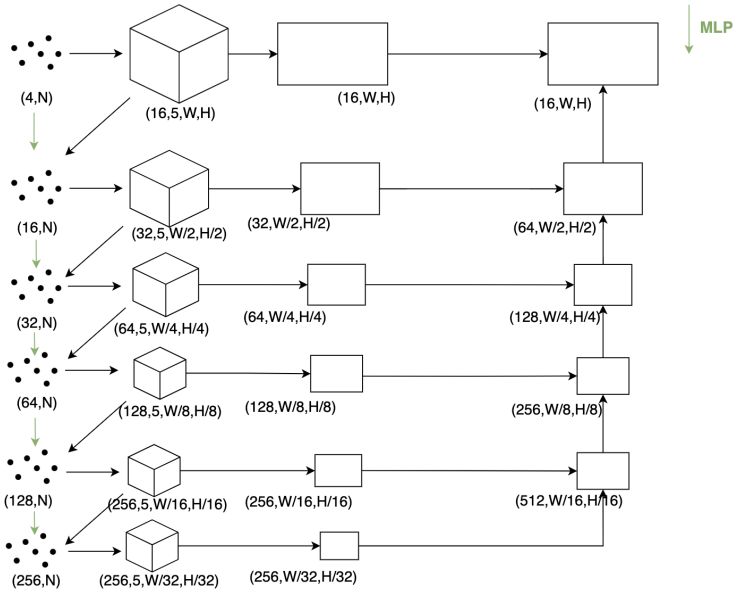


Figure 3.3: PV Unet.

3.4 Model Comparison

In classification task, we compare point-based method (PointNet++), 2D voxel-based method (2D ResNet34), 3D voxel-based method (3D ResNet18) and point-voxel method (PVCNN). The results are shown in Table 3.5.

Models	Accuracy	Nb. params	Speed(Instances/sec)
PointNet++	51.956	**1.8M**	9.4
3D ResNet18	74.943	33.2M	58.16
2D ResNet34	76.092	21.3M	**58.2**
PVCNN	**78.506**	10.4M	25.36

Table 3.5: Performances of Classification Models

In regression task, we compare 2D voxel-based method (2D Unet), 3D voxel-based method (3D Unet) and point-voxel method (PV Unet). The evalution

criteria are:

EPE: 1-pixel-error, the percentage of ground truth pixels with optical flow magnitude error > N. N is either 1, 2 or 3.

EPE: Endpoint error. The average of the L2-Norm of the optical flow error.

AE: Angular error.

The results are shown in Table 3.6.

Models	**EPE**	**AE**	**1PE**	**2PE**	**3PE**	**Nb. params**	**Speed**
2D Unet	1.334	**4.513**	33.043	14.143	7.935	19.9M	**23.4**
3D Unet	1.326	4.676	31.888	13.556	7.648	**10.5M**	7.36
PV Unet	**1.296**	4.570	**30.270**	**12.996**	**7.530**	**10.5M**	6.36

Table 3.6: Performances of Regression Models

We can draw the following conclusions:

1. Point-based method is inaccurate and slow;
2. Point-voxel method has the best performance in all unpretrained models with sacrificing some speed;
3. 3D voxel-based method is either bad at performance or speed. The most time-consuming part is 3D convolutions;
4. 2D voxel-based method has a good trade-off between performance and speed.

Chapter 4

Explorations in PVCNN

From the previous chapter, point-voxel methods have the best performances. In this chapter, we will explore the reasons why it performs best. Which part of point-voxel methods is useful and why does it work?

4.1 MLP is not useful inside Point Voxel Block

One innovation about PVCNN model is the fusion of voxel-based features and point-based features. The main difference between point-voxel method and pure voxel-based method is the point features. In the original PVCNN paper, the authors also have proved that the MLPs inside PVCNN blocks are important and useful. However, their tasks are based on lidar points and our tasks are based on event data. The results may be not consistent. In this section, we try to cancel MLP or increase the layers of MLP.

Models	**Accuracy**	**Nb. params**	**Speed(Instances/sec)**
No MLP	**78.506**	**10.4M**	**25.36**
Single Layer	77.816	10.5M	24.32
Two Layers	76.322	10.7M	22.12

Table 4.1: Effects of MLP parameters in PVCNN

Models	**EPE**	**AE**	**1PE**	**2PE**	**3PE**	**Nb. params**	**Speed**
No MLP	**1.296**	**4.570**	**30.270**	**12.996**	**7.530**	**10.5M**	**6.36**
Single Layer	1.366	4.687	32.473	14.712	8.405	10.6M	5.73

Table 4.2: Effects of MLP parameters in PV Unet

From Table 4.1 and Table 4.2, we can observe that point-based features generated by MLP is useless for event data, even bring some noise. The more layers we increase, the more negative effects it has. MLP structure can not extract useful information form event data for both classification and regression tasks.

We are also curious about the reason. Fig. 4.1 is a driving scene from DSEC dataset. We can observe that in event dataset, a few hundred milliseconds are taken as an example at most, and then the points in the time dimension tend to be straight without too many complex 3D structures, which means there is little information along time dimension. It is super different from lidar dataset, which always has abundant depth information. That is why the point-feature can not work at all.

Figure 4.1: An example from DSEC dataset. Left: Image. Right: Events

4.2 Devoxelization is useful

After removing MLP structures, the only difference between point-voxel method and pure voxel-based method is the devoxelization part. In this section, we try to remove the devoxelization and see the performance.

In the classification task, the original model in Fig. 4.2 voxelize and devoxelize at each PVCNN block.

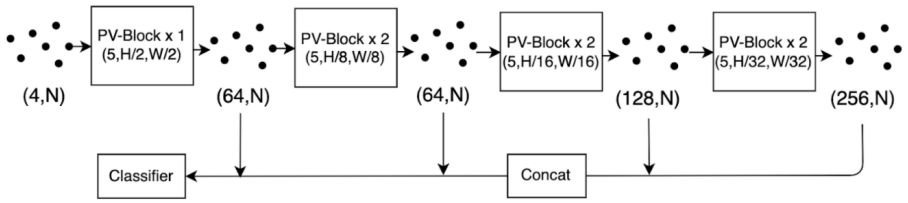


Figure 4.2: Orignial PVCNN model.

Firstly we try to remove only voxelization in Fig.4.3, which means what we pass through different resolutions are voxel-based features, but we still use point-based features to predict labels.

Next we remove both voxelization and devoxelization in Fig.4.4, which means we pass voxel-based features between different resolutions and we also use voxel-based features to do prediction. Actually, without concatenation, it pretty similar to a shallow ResNet.

In the regression task, we do the same thing. After removing devoxelization, the model actually becomes 3D Unet.

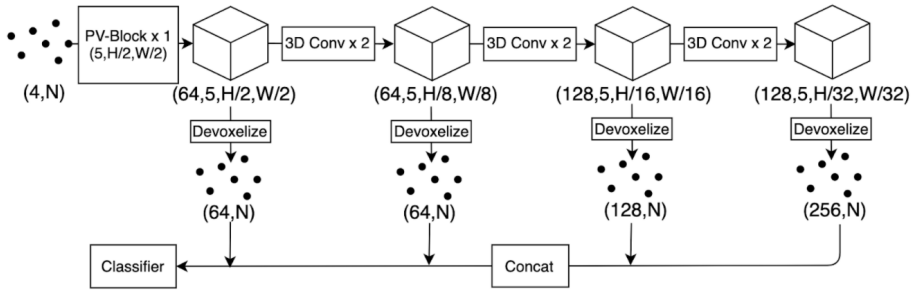


Figure 4.3: Remove voxelization in PVCNN.

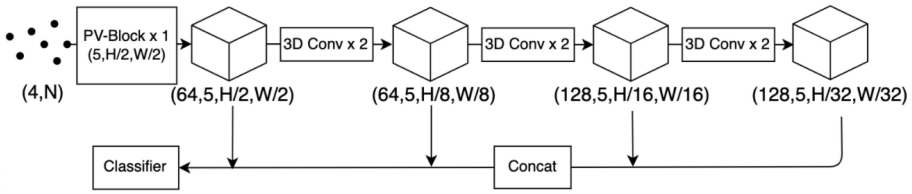


Figure 4.4: Remove both voxelization and devoxelization in PVCNN.

Models	Accuracy	Nb. params	Speed
Original	**76.897**	9.1M	36.12
w/o voxelize	69.885	9.1M	42.16
w/o both voxelize and devoxelize	70.115	9.1M	**67**

Table 4.3: Effects of devoxelization in PVCNN.

Models	EPE	AE	1PE	2PE	3PE	Nb. params	Speed
w/ devoxelize	**1.296**	**4.570**	**30.270**	**12.996**	**7.530**	10.5M	**6.36**
w/o devoxelize	1.326	4.676	31.888	12.556	7.648	10.5M	7.36

Table 4.4: Effects of devoxelization in PV Unet.

From the results shown in Table 4.3 and Table 4.4 , we can draw the conclusion that passing point-based features between different resolutions performs better than only passing voxel-based features. In other words, devoxelizing to point-based features is necessary for a better performance.

It is because when trilinearly devoxelize, we use the position information of the original points implicitly. But in pure 3D voxel-based methods, the original position information will lose gradually because the resolutions become smaller and smaller.

Chapter 5

Another Attempt

5.1 Sparse convolution

Sparse convolution have been used on events for fewer computations and faster inference speed [7]. In our project, we replace the original dense convolution in Point Voxel Networks by sparse convolution and show that the inference speed increases, but the performance drops under this model structure. Specifically, in PVCNN we replace every point-voxel block by sparse convolution, in PVUNET we replace the point-voxel blocks in the encoder. There exist differnt types of sparse convolutions and we use Minkowski1[1] sparse convolution.

	Accuracy	**Nb. params**	**Speed(Instances/sec)**
Dense	**78.506**	10.4M	25.36
Sparse	56.122	**9.7M**	**36.72**

Table 5.1: Sparse vs. Dense convolution in classification

Table 5.1 shows the classification results in N-Caltech101 when we replace the dense convolution by sparse convolution. The sparse convolution in this case decreases the accuracy. And the same result is showed in optical flow regression in Table 5.2.

	EPE	**Nb. params**	**Speed(Instances/sec)**
Dense	**1.296**	10.5M	6.36
Sparse	1.393	**11.2M**	**15.6**

Table 5.2: Sparse vs. Dense convolution in regression. Lower EPE is better.

The results show that using sparse convolution to replace point voxel structure harms the performance, under the same network structure. In regression task, sparse convolution hurts performance because the output of sparse convolution is sparse, while the task we want to predict is a dense optical flow.

Chapter 6

Discussion

6.1 Conclusion

We introduced downsampling and point masking strategies to apply Point Voxel Networks on event data. The PVCNN can achieve high performance on events classification as an unpretrained model. The PV Unet model can achieve similar performance as 2D Resnet in optical flow regression. The limitation of Point-Voxel Network is that they are not fast compared to 2D Resnet, and they need large memory space for training. We also showed that purely point-based models and 3D convolution models are not suitable for event data, and 2D convolution models have a good trade-off between performance and inference speed.

6.2 Outlook

In the future it is worthwhile to explore more about the downsample method, because it is relevent to the performance of Point Voxel Networks. The current downsample method sums or interpolates polarities into voxel bins, ignoring the fact that there may be positive and negative events in the same bin that cancel each other out. And when extracting points from voxels we use the time, i.e. the discrete bin number, of the voxel bins to recover the timestamps of extracted points. However, more accurate ways to recover the timestamps of points could be used, for example one could record point timestamps for voxels during voxelization, and use them to recover the more fine-grained timestamp.

Appendix A

Downsampling

A.1 Downsample Code

The code of downsample is in *utils/vis3d.py* , and one can visualize the downsampled points by using function *vis3d* , which is in the same python file.

A.2 The Effectiveness of Downsample Method

We show that our downsampling method is much better than random downsampling. But we do not claim it is the best downsample method, actually we discussed possibilities to improve it in the last chapter.

The following Figure A.1 shows the effect of downsampling using the same model. The figure is organized such that the upper righter the better. We can see from the red vector that our downsample method outperforms random downsample by a large margin, especially in optical flow regression task. Under certain hyperparameter setting: summing polarities, 10 temporal bins, threshold=1, the PVCNN model gets 78.5% accuracy, which is highest among unpretrained models.

A.3 A new downsampling method

In 6.2 we mentioned there could be better downsampling method, and we provide a possible implementation as follows. The code is the function *downsample_sum_new* in python file *utils/vis3d.py.*

During voxelization, we have separate voxel-grids for positive and negative events respectively. And we have separate voxel-grids capturing polarities and timestamps respectively. Therefore we have 4 voxel-grids during voxelization. For voxel-grids that capture polarities, we accumulate, i.e. sum up the polarities of events in the corresponding voxel. However, accumulating polarities by interpolation is also possible. For voxel-grids that capture timestamps, we compute the average timestamp of events in each voxel. Finally we filter out voxels using a threshold and recover events from remaining voxels. When recovering

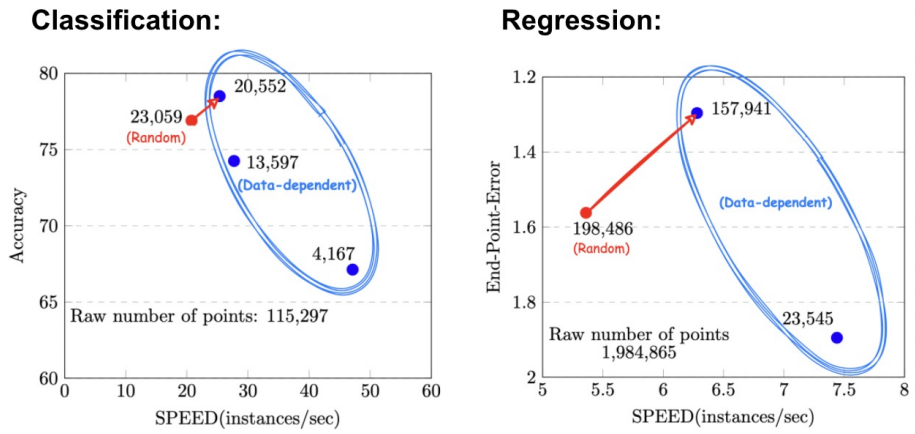


Figure A.1: PVCNN performance under different downsample strategies. Upper right corner is better in both plots. Blue dots represent different hyperparameters with the proposed method, while red point represents random downsample.

events, we use the average timestamp of each voxel instead of the discreet bin number, and thus we can retain more temporal information. Figure A.2 shows the difference between the downsample method used in this project and the new downsample method discussed in appendix.

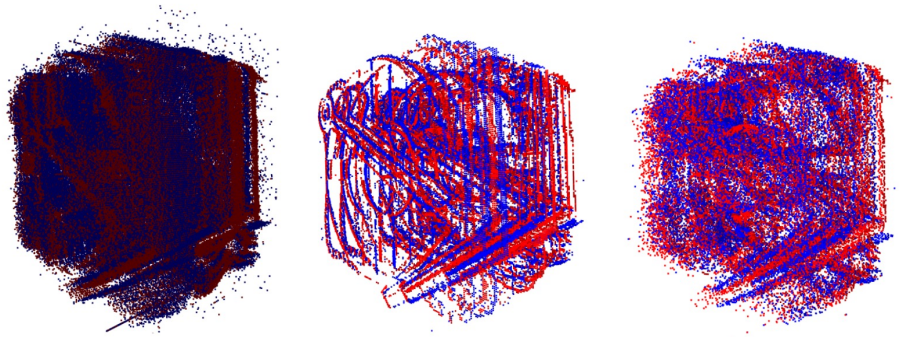


Figure A.2: Difference between downsample methods. From left to right: original events, the downsample method used in this project, the new downsample method

Bibliography

[1] Christopher B. Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. *CoRR*, abs/1904.08755, 2019.

[2] Daniel Gehrig, Mathias Gehrig, Javier Hidalgo-Carrió, and Davide Scaramuzza. Video to events: Bringing modern computer vision closer to event cameras. *CoRR*, abs/1912.03095, 2019.

[3] Daniel Gehrig, Antonio Loquercio, Konstantinos G. Derpanis, and Davide Scaramuzza. End-to-end learning of representations for asynchronous event-based data. *CoRR*, abs/1904.08245, 2019.

[4] Mathias Gehrig, Willem Aarents, Daniel Gehrig, and Davide Scaramuzza. DSEC: A stereo event camera dataset for driving scenarios. *CoRR*, abs/2103.06011, 2021.

[5] Mathias Gehrig, Mario Millhäusler, Daniel Gehrig, and Davide Scaramuzza. Dense optical flow from event cameras. *CoRR*, abs/2108.10552, 2021.

[6] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel CNN for efficient 3d deep learning. *CoRR*, abs/1907.03739, 2019.

[7] Nico Messikommer, Daniel Gehrig, Antonio Loquercio, and Davide Scaramuzza. Event-based asynchronous sparse convolutional networks. *CoRR*, abs/2003.09148, 2020.

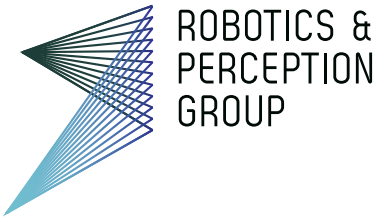
[8] Garrick Orchard, Ajinkya Jayawant, Gregory K. Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in Neuroscience*, 9, 2015.

[9] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016.

[10] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *CoRR*, abs/1706.02413, 2017.

[11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[12] Yusuke Sekikawa, Kosuke Hara, and Hideo Saito. Eventnet: Asynchronous recursive event processing. *CoRR*, abs/1812.07045, 2018.



Title of work:

Efficient Spatio-Temporal Processing of Event Data

Thesis type and date:

Semester Thesis, Feb 2022

Supervision:

Mathias Gehrig
Nico Messikommer
Prof. Dr. Davide Scaramuzza

Students:

Name:	Kexin Shi
E-mail:	kexin.shi@uzh.ch
Legi-Nr.:	20-744-538
Name:	Yifei Liu
E-mail:	yifei.liu@uzh.ch
Legi-Nr.:	20-742-250

Statement regarding plagiarism:

By signing this statement, we affirm that we have read the information notice on plagiarism, independently produced this paper, and adhered to the general practice of source citation in this subject-area.

Information notice on plagiarism:

`http://www.lehre.uzh.ch/plagiate/20110314_LK_Plagiarism.pdf`

Zurich, 11. 12. 2022: _____