Thesis Research

Optimizing Event-Based Vision by Realizing Super-Resolution in Event-Space: an Experimental Approach

RO57035: RO MSc Thesis

Mahir Sabanoglu

Delft University of Technology

Thesis Research

Optimizing Event-Based Vision by Realizing Super-Resolution in Event-Space: an Experimental Approach

by

Mahir Sabanoglu

Student:	MM Sabanoglu
Student Number:	4494172
Supervisor:	Prof. dr. ir. J.C.F. de Winter
Daily Supervisor:	Dr. ir. N. Tömen
Business Supervisor:	Ir. E.M. Souman
Institution:	Delft University of Technology
Place:	Faculty of Mechanical, Maritime and Materials Engineering, Delft
Project Duration:	April, 2022 - January, 2023

Cover Image: 3D Phase portrait visualization from https://wallpaperbat.com/data-science-wallpapers

Contents

List of Figures

List of Tables

Optimizing Event-Based Vision by Realizing Super-Resolution in Event-Space: an Experimental Approach

M.M. Sabanoglu
Cognitive Robotics
Delft University of Technology
Delft, Netherlands

Dr. ir. N. Tömen
Computer Science
Delft University of Technology
Delft, Netherlands

Prof. dr. ir. J.C.F. de Winter
Cognitive Robotics
Delft University of Technology
Delft, Netherlands

***Abstract*—An event-based camera enables capturing a video at a high temporal resolution, high dynamical range, reduced power consumption and minimal data bandwidth while the camera has minimal physical dimensions compared to a frame-based camera with the same vision properties. The limiting factor, however, of an event-based camera is the spatial resolution which ranges between 40×40 and 1280×960. To counter this deficiency, a method is researched to super resolve event-based vision in order to enhance spatial resolution. A selection of different neural network types and configurations are researched in a step-by-step fashion. Subsequent experiments tested the selected networks on their ability to process event-based data and extract features from it. Followed by experiments that exploited the limitations of the networks to super resolve at different ratios, lengths of eventstreams and more complex event-based data.**

Results of various experiments showed that a network configuration that utilizes a transformer architecture was best able to super resolve event-based vision. This type of network leverages the ability to extract features based on dependencies between events which aligns with the characteristics of event-based vision. Based on the obtained results from the experiments, a pipeline is proposed to super resolve event-based vision and consists of a combination of a transformer network, multilayer perceptrons and a *k*-nearest-neighbor algorithm. Using this pipeline, eventstreams can be super resolved in the spatial resolution at a scaling ratio of 4. Visually, these super resolved eventstreams resemble more detailed and enhanced version to the low-resolution input. This proposed pipeline can be considered as a starting point in further research toward the super-resolution of event-based data and thereby contributes to the extension of application possibilities of event-based vision.

***Index Terms*—event-based vision, event-space, *k-NN*, machine learning, MLP, N-Caltech-101, N-MNIST, neural network, neuromorphic vision, self-attention, super-resolution, transformers**

1. Introduction

Capturing a video with a high temporal resolution (in the order of MHz), with a reduced data bandwidth and energy consumption while having a higher dynamical range (140 dB vs 60 dB) with a camera the size of a small handheld recorder is unthinkable with a frame-based camera, but made possible with an event-based camera [1], [2]. Applications for this new technology are among others: a flying drone that navigates by video input [3], [4], [5], a camera mounted on a vehicle [6], [7], [8], [9], or for super slow motion visualization [10]. Inspired by the biological eyes, researchers have come up with a different paradigm called neuromorphic or event-based vision that enables these properties [11], [12]. Both names are interchangeable; neuromorphic vision indicates that the paradigm is inspired by neuroscience, while event-based vision refers to the principle of the paradigm in which visual input is translated into events. In this research, the name event-based vision will be used as the name is derived from the technical principle.

Research towards event-based vision has come to a point at which it is possible to build Dynamic Vision Sensors (DVS) which enables event-based vision. However, currently, one of the limitations of event-based vision is due to the large physical size of pixels. Large pixel sizes inherently increases the needed distance between pixels (also called pixel pitch) and thus fewer pixels can be fitted on the sensor which reduces the resolution. Event-based vision sensors generaly have a resolution between 40×40 and 240×180 [13], [14] although there is progress in reducing the pixel pitch to realize larger resolution sensors of 1280×960 [15]. Nevertheless, the resolution of event-based vision is much lower compared to what is possible with frame-based vision. Therefore, depending on the purpose, diminishing this shortcoming can be beneficial. There is a class of techniques to enhance the resolution of imaging after being captured. This class is referred to as super-resolution or super resolving. The task of these techniques is to find higher-resolution representatives of low-resolution inputs. A high-resolution representative is a plausible high-resolution imaging that reassembles a low-resolution input but with enhanced resolution. Super-resolution is an ill-proposed problem meaning that the solution is not deterministic since data that results in the high-resolution is missing and ought to be found. There are proposed solutions in which event based vision is super resolved. Still, all of these models inflict a conversion from events to frames [16], [17], [18], [19]. This conversion counters its beneficial properties of having a minimal data bandwidth and a high temporal resolution. Therefore, it should be omitted.

In this research, a method will be proposed that performs super-resolution directly on the data structure of event-based data. Super resolving in this fashion does not compromise the event-based vision properties of minimal data bandwidth and a high temporal resolution and enables converting events-towards at a variable frame-rate after being super resolved. In addition, with respect to earlier proposed solutions that inflict a conversion to a frame-based representation, this method makes super-resolving events potentially less power-demanding and faster to process since eventstreams occupy less memory and more accurate since temporal information is not discretized.

1.1. Characteristics of event-based vision

The paradigm behind event-based vision is only discussed on a high level in the introduction. The principles of event-based vision will be deliberated in more detail in this section to give a better understanding. The sensor used in event-based vision is continuously exposed to light. The pixels in this sensor will trigger an event whenever the observed brightness (I_t) has surpassed a preset threshold (θ) with respect to the previously observed event (I_{t-1}). Therefore, each event only visualizes relative brightness changes. This can be mathematically described as:

$$\log(I_t) - \log(I_{t-1}) \geq \theta. \tag{1}$$

This happens independently and asynchronously for each pixel continuously in time. It results in pixelwise visual observation only when there is a change in scenery (and thus a brightness change) and no visual observation when there is no visual change in that particular pixel. This is contrary to frame-based vision in which the absolute value of brightness is captured at a fixed frame rate regardless of the change in scenery.

A visualization of the differences between event- and frame-based vision is made by [20] and is depicted in Figure 1. It shows a schematic overview of a test setup. Above is a Standard camera (frame-based camera) and beneath is an Event camera, both observing a spinning disc with a black dot on its surface. Successively, the disc spins at high, no, and slow angular velocities. Obtained data by both cameras are depicted in a timeline in which the spatial plane reflects the camera sensor. Noticeably, the data of the Standard camera is prone to motion blur which is visible when the disc rotates at a high velocity, while in the period when the disc did not rotate it produces redundant data as all the frames include the same visual information. The data of the Event camera is color-coded blue or red for positive and negative brightness changes respectively. It shows a high temporal resolution when the disc spins fast. Yet, it does not produce data when there is no change in scenery, this is also visible in the period when the disc did spin since the events only occur in the area in which the black dot moves and not its surroundings. Another characteristic feature of event-based vision is that moving objects will result in a manifold in space-time. The surface of this manifold could be a useful feature for super-resolving as it marks a region where brightness changes.

The obtained data structure of event-based vision is a 4-dimensional tuple sequence:

$$\{e_i\}_N = \{x_i, y_i, p_i, t_i\}_N \tag{2}$$

also referred to as eventstream containing N events, where $(x_i, y_i) \in \mathbb{R}^{H \times W}$ indicates the position on the sensor grid, $p_i \in \{0, 1\}$ the event polarity respectively positive or negative brightness change and t_i the timestamp at which event e_i is observed. Events can be plotted in a 3-dimensional space-time continuum as depicted in Figure 1. This continuum is called event-space and consists of both spatial- (x, y) and time dimension t. The term event-space is also used as an indication that the event-based vision is in its original data format.

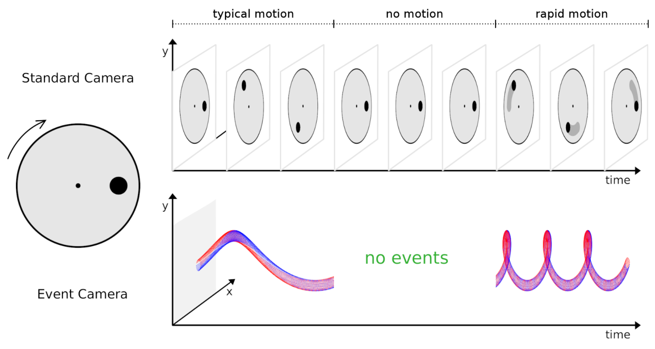


Figure 1. Visualization of differences in perceived data by a frame-based camera (top) and an event-camera (bottom) when filming a spinning disc with a black dot [20].

Eventstreams are usually converted towards a frame-based representation since a raw tuple sequence nor the representation in event-space are visually easy to interpret. This conversion can be made in various fashions. In this research, the most elementary fashion will be used, namely event-frames. Eventstreams are converted using the pseudocode as depicted in Algorithm 1. In short, all events occurring in a set time-bin are accumulated according to their polarity (with +1 or -1) and spatial address onto an array of the same spatial dimension with initialized values of a gray tint (brightness equals 128 for 8-bit representation). A visualization of this process is depicted in Figure 2.

Algorithm 1 converting eventstreams ($\{e_i\}_N$) to event-frames (F)

```
Input: $\{e_i\}_N = \{x_i, y_i, t_i, p_i\}_N, t_{bin}$
Output: $F_{(W,H,\#bins)}$
 1: $\#bins = t_N / t_{bin}$
 2: $F = \frac{255}{2} \times 1_{(W,H,\#bins)}$
 3: for $e_0$ to $e_N$ do
 4:     if ($p_i == 1$) then
 5:         $bin = \text{floor}(t_i / t_{bin})$
 6:         $F_{(x_i,y_i,bin)} += 1$
 7:     else
 8:         $F_{(x_i,y_i,bin)} -= 1$
 9:     end if
10: end for
11: return $F$
```

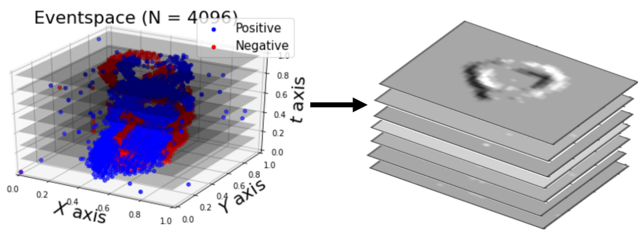


Figure 2. Visualization of the converting process to obtain event-frames (right) from eventstreams depicted in event-space (left).

1.2. Super-resolution

The problem of super resolving data should be seen as finding missing data. Although it is an ill-proposed problem, there are differences that make one solution perform better than other solutions in a given circumstance. The most elementary analogy is: given a set of data points $x_1[0, 2, 4, 6]$ results in $y_1[0, 2, 4, 6]$, which values would belong to $x_2[0, 1, 2, 3, 4, 5, 6]$? Without further knowledge in this example, the trivial prediction would be $y_2[0, 1, 2, 3, 4, 5, 6]$ but this does not necessarily have to be true (e.g. given x_1, the function $f(x) = \sin \pi x + x$ would also result in y_1). This is just a brief example that clarifies what an ill-proposed problem defines and highlights the need for an algorithm to have a policy that aligns the dynamics of a system in order to super resolve it properly.

There are two global categories that will be featured in this research, namely naive and learning-based. In a naive algorithm, the dynamics of a system are modeled by a fixed set of rules. In the previous analogy, this could very well have been $f(x) = x$ or $f(x) = \sin \pi x + x$. For a naive algorithm, it is key that the model is according to the underlying characteristics of the system. It is therefore crucial to have an understanding of the system in order to formulate this model. While in a learning-based (also referred to as machine learning) approach, the characteristics of the system, analogous to the name, are learned during a process called training. In short, during training, the algorithm is fed both inputs x and outputs y numerous times at which it fits a model that seems to project the input to the output the best according to the set criterion which is defined as a loss function. This process will be deliberated in detail later in this report. It is therefore not key to have a thorough understanding of the underlying characteristics of a system, which makes it ideal to implement on systems with complex or unknown characteristics. Yet, a good understanding is needed to formulate a loss function that quantifies the performance of the network appropriately in order to make the algorithm converge toward the desired solution. While a high-level understanding is granted to make decisions on the type and shape of the machine learning algorithm to find the best solution.

In frame-based vision, the amount of to be predicted data points is deterministic to the scaling factor. For example, $4 \times H \times W$ data points need to be predicted when scaling an image of resolution (H, W) by factor 2 to $(2H, 2W)$. Super-resolution in event-based vision does not translate to a predetermined number of events to be present in the higher resolution. This is a direct consequence of the asynchronous method in which each pixel observes vision in the form of events independently from other pixels. It requires a method that enables variable length sequence to sequence prediction. However, since the primary scope is to find suitable fundamental methods, super-resolution of events is considered as a fixed length vector to vector prediction in order to prevent a too-narrow initial scope for solutions.

1.3. Related Work

Event-based vision is a novel research area that is reflected by the amount of found literature on this research topic. In [21] a naive sampling technique named Thinning is used to super resolves events. This sampling technique is based on the principles of Poisson Point Process and samples points (or events) based on probability. Both[17] and [18] propose a model which uses converted event-frames combined with images. A model which uses eventstreams and an image is proposed in [22]. These models which use a combination of events and images leverage the advantages of both data sources. Yet, these models are limited to the use of captures made by event-cameras which utilize both a DVS and an Active Pixel Sensor (APS). What makes our research unique is the input and output both comply with the eventstream datastructure whereas available solutions use a frame-based datastructure.

There is an undeniable similarity upon projecting the datastructure of eventstreams ($\{e_i\}_N = \{x_i, y_i, p_i, t_i\}_N$) on pointclouds ($\{p_i\}_N = \{x_i, y_i, z_i\}_N$). Namely, both datastructures consists of 3 Euclidean dimensions the polarity of events is neglected. In literature, various upsampling algorithms are proposed for the super-resolution for pointclouds. This makes the proposed models in the pointcloud domain a potential inspirational source. Used architectures for these pointcloud upsampling networks are Convolutional Neural Networks (CNN) [23], [24], Graph Convolutional Network (GCN) [25], [26] and Transformer Neural Networks (Transformer) [27], [28]. Both GCN and Transformer architectures have been used for classification tasks of event-based data in event-space [29], [30], [31], but not for super-resolving like done with pointclouds.

1.4. Research questions

The proposed solutions in literature for the task of super-resolution applied to event-based vision involve a discretization somewhere in the process. However, to cherish the advantage that comes along with the data sparsity and embedded temporal information of the event representation, discretization and conversion towards a frame-based representation should be omitted. Therefore, a solution is researched which enables super-resolution of eventstreams in event-space. For this research, a wide range of computer vision techniques is considered to find a solution. The most prominent technique used in computer vision is neural networks (also referred to as networks). This is a sub-section of machine learning and is an overarching term for every computing system which consists of a network with a topological resemblance to the biological brain [32]. This means that these networks consist of a likewise structure (also referred to as architecture) with layers of neurons that are connected to each other to surpass data and perform abstractions of data, together they form a large system that realizes a powerful computation tool. There are various different neural network types, and combinations of these lead to an even more vast variety of advanced networks. This research suggests the applicability of a solution in the form of a pipeline that is divided into three distinctive modules. A schematic of this pipeline, consisting of a Feature extractor, Coordinate reconstructor, and Polarity reconstructor is depicted in Figure 3.

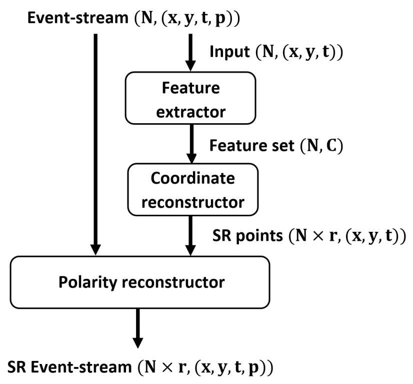


Figure 3. Schematic overview of the suggested super-resolution pipeline consisting of 3 modules with an eventstream length (N), 2-dimensional coordinate(x, y) of the concerning event with respect to the sensor grid, polarity (p) of the event, feature channel (C), scaling ratio (r) between input and output length of eventstream.

The Feature extractor module extracts features from the given input. For this module, it is key to handle the data representation efficiently while being able to extract the right features from it. Features are then encoded toward a feature space. The Coordinate reconstructor predicts the coordinates of the super-resolution based on the extracted features in the previous module. After this step, solely the locations of the events are predicted in the (x, y, t) dimension. In the third step, the Polarity reconstructor module predicts the polarity at predicted event locations. This step can be seen as a supervised classification problem in which the polarity is represented by a positive $\{1\}$ or negative $\{0\}$ label. Features that can be used to predict the polarity are events with the polarity of the low-resolution input. k-Nearest-Neighbors (*k-NN*) is often used for these types of tasks, although it has never been used before for this specific case to predict the polarity of events. In this study, different experiments are conducted in which multiple methods are tested and compared in order to find suitable solutions for this distinctive module.

In order to validate a found solution, a baseline method will be reproduced which is based on Poisson Point Process sampling, specifically Thinning, as is proposed by [21]. Although this method initializes a small discretization, it is the next closest option to super resolve event-based vision in an event-to-event fashion. If the found techniques are performing poorer to a specific metric with respect to the baseline, the found technique has a small chance to be suitable for further investigation and vice versa.

This raises the following research question: ***To what extent and by which method can event-based vision be super resolved in event-space?***

With sub-questions:

1) *To what extent can a naive algorithm be used to super resolve events or would it suffice to have a learning-based algorithm?*
2) *Which type of neural network is best capable to extract features in event-based data when used in the Feature extractor module?*
3) *To what extent can we predict the coordinates of events of high-resolution eventstream based on a given feature-set using a Coordinate reconstructor module?*
4) *To what extent can we retrieve polarity after super-resolving using k-NN as a Polarity reconstruction module?*

1.5. Approach

The goal of this research is to find a proof of concept to super resolve event-based vision in event-space which can be used as a stepping stone in further research. The expected solution has the form of a pipeline as defined in Figure 3. This pipeline consists of 3 separate parts, a Feature extractor, Coordinate reconstructor and Polarity reconstructor. Solutions for the first two modules will be found by researching currently available methods for pointcloud super-resolution and for event-based vision classification and evaluating their applicability based on results from conducted experiments. This study focuses mainly on the development of the feature extractor module since it is expected to demand the most unique and tailor-made solution whereas more generic solutions are suitable for the Coordinate reconstructor and Polarity reconstructor modules.

The initially used Coordinate reconstructor is based on a module used in [33] with a similar utility. Premature experimenting during the literature study prior to this research proved the suitability of a *k-NN* algorithm to be used as a Polarity reconstructor. With the use of this algorithm, predictions about the polarity of newly predicted events can be made based on their location in relationship with the k nearest neighbors with known polarity from the low-resolution eventstream. In Experiment 1 the suitability limitations of *k-NN* as a Polarity reconstructor will be thoroughly researched under different super-resolution scaling ratios and settings for *k*.

The research starts by defining a baseline method which is based on the Poisson Point Process (PPP). Then, solutions for the suggested pipeline are researched using a step-by-step approach in which sequential experiments are conducted with increasing complexity. Using this approach makes it possible to take a broad set of initial proposed solutions into consideration which can be narrowed down early on in the experiments. Experiment 2 concerns a classification task. For this test, the network only consists of a Feature extractor and tests the ability of a Feature extractor to (1) handle event-based data and (2) extract features in general from the eventstreams data structure. Although the extracted features needed for classification are most likely to be different with respect to super-resolution of event-based data, it will give a first indication of its ability to feature in this data structure. In Experiment 3 selected networks - based on the results from the previous experiment - are extensively tested on their ability to super resolve under various test configurations (values for scaling ratio (r) and input sequence length (N)) and different datasets. Comparison between different methods must be fair in order to enable valid consideration between them based on the results of the experiments. Each experiment in which a comparison ought to be made should thus only have a deviation in one of the modules. To realize this, the modules have to be interchangeable i.e. all Feature extractors modules should be compatible with identical Coordinate reconstructors. A boundary condition is set in order to secure the validity of comparison, the designed feature extractor should output a feature set with the dimensions $N = 1024$ and $C = 128$. After this test, a network type will be selected based on analyzing the obtained results and complemented by arguments based on theory. The final experiment is conducted in Experiment 4 in which the obtained solution is compared to the defined baseline solution which is based on the PPP.

2. Methods

2.1. Baseline method

In [21], event-based vision is super resolved using a naive algorithm named Thinning. Thinning [34], [35] is a sampling method based on the characteristics of a one-dimensional nonhomogeneous Poisson Point Process (PPP). It enables generating a specified amount of events (S^*) on a set timeline $(0, T]$ according to a given probability density function. This algorithm is of the naive kind. To use this algorithm on event-based data, the probability density function and the accumulated amount of events S^* are properties that need to be calculated per pixel address.

The probability density function $(\lambda(t))$ is derived from a histogram Peristimulus Time Histogram (PSTH). The rate function at the spatial grid location i, j is $P_{i,j}$ and resembles a histogram divided in time-bins with length t_{bin} in which the value of each time-bin depicts the concerning value of occurring events. This division into small non-overlapping bins is the part that establishes a discretization into the process. A pseudocode for the calculation of the rate-function is depicted in Algorithm 2. The number of occurring events S^* is calculated per spatial address i, j, together they form a countmap (C). This calculation is depicted in pseudocode in Algorithm 3. For example: consider a sensor consisting of only two neighboring pixels (resolution of 1×2) named p_0 and p_2 respectively which can be super resolved to obtain a resolution of 1×3. PPP characteristics S_0^*, S_2^*, $\lambda_0(t)$ and $\lambda_2(t)$ are derived after which $S_1^* = \frac{S_0^*+S_2^*}{2}$ events can be sampled according to the probability function $\lambda_1(t) = \frac{\lambda_0+\lambda_2}{2}$ using thinning to find the predicted observations of p_1. The same principle is used in [21] but then in two dimensions, width and height.

These two characteristics ($\lambda(t)$ and S^* of each event) are then used to predict the PPP characteristics of neighboring pixels from a super resolved perspective by interpolation, after which the thinning algorithm is used to sample events. This procedure is executed separately for positive and negative events. Per spatial address i, j with $0 < i < W$ and $0 < i < H$, S^* is set to the concerning value in the countmap $C_{i,j}$. The nonhomogeneous rate function $\lambda(t)$ is derived from the concerning PSTH $(P_{i,j})$, and the homogeneous rate function (λ^*) is set to the maximum value of $\lambda(t)$ factored by 1.2. Then a value t_{gen} is generated as a fictive event in the range $[0, T]$, and v is generated in the range $[0, 1]$. If and only if $v \leq \lambda(t_{gen})/\lambda^*$, the fictive event is accepted and appended to the list of generated events (S) with value $e\{i, j, t_{gen}, pol\}$ from which pol equals 1 or 0 depending on the concerning polarity. This process is repeated until the list S contains S^* events. The pseudocode of this process is depicted in Algorithm 4. A visualization of the Thinning process is depicted in Figure 4.

Algorithm 2 PSTH (P^+, P^-)

```
Input: ${e_i}_N = {x_i, y_i, t_i, p_i}_N$, $t_{bin}$
Output: $P^+_{(W,H,\#bins)}, P^-_{(W,H,\#bins)}$
 1: $\#bins = t_N/t_{bin}$
 2: for $e_0$ to $e_N$ do
 3:     if ($p_i == 1$) then
 4:         $bin = \text{floor}(t_i/t_{bin})$
 5:         $P^+_{(x_i,y_i,bin)} += 1$
 6:     else
 7:         $P^-_{(x_i,y_i,bin)} += 1$
 8:     end if
 9: end for
10: $P^+_{(2W,2H,\#bins)} \leftarrow interpolation(P^+_{(W,H,\#bins)})$
11: $P^-_{(2W,2H,\#bins)} \leftarrow interpolation(P^-_{(W,H,\#bins)})$
12: return $P^+, P^-$
```

Algorithm 3 positive and negative countmap (C^+, C^-)

```
Input: ${e_i}_N = {x_i, y_i, t_i, p_i}_N$
Output: $C^+_{(W,H)}, C^-_{(W,H)}$
 1: $C^+_{(W,H)}, C^-_{(W,H)} = 0_{(W,H)}$
 2: for $e_0$ to $e_N$ do
 3:     if ($p_i == 1$) then
 4:         $C^+_{x_i,y_i} += 1$
 5:     else
 6:         $C^-_{x_i,y_i} += 1$
 7:     end if
 8: end for
 9: $C^+_{(2W,2H)} \leftarrow interpolation(C^+_{(W,H)})$
10: $C^-_{(2W,2H)} \leftarrow interpolation(C^-_{(W,H)})$
11: return $C^+, C^-$
```

2.2. Neural Networks

Neural networks (referred to as networks) belong to a subset of machine learning algorithms which are self-learning algorithms. A network can consist of a combination of different neural network types. Input to the network is processed through a set of mathematical operations, the different types of networks result in different mathematical operations each with its unique properties. A network should leverage these properties in order to be optimal. Background information in this section concerning different network

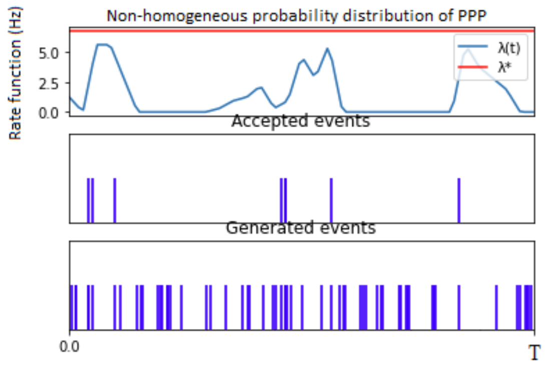


Figure 4. Visualization of Thinning according to the rate functions $\lambda(t)$ and λ^* (top), generated fictive event locations t_{gen} (bottom) and accepted event locations v (middle).

Algorithm 4 super resolved events sampling $\{e_i\}_N \rightarrow \{e_i\}_M$

```
Input: ${e_i}_N, C, P, pol \in [0, 1]$
Output: ${e_i}_M$
 1: $E_{sr} = 0_{(1,4)}$
 2: $i, j = 0$
 3: for 0 to $W$ do
 4:     for 0 to $H$ do
 5:         $S^* = C^{pol}_{(i,j)}$
 6:         $S = 0_{(1,4)}$
 7:         $\lambda(t) = f(P^{pol}_{(i,j,:)})$
 8:         $\lambda^* = 1.2 \max(\lambda(t))$
 9:         while $\text{len}(S) < S^*$ do
10:             generate $t_{gen}$ random in time interval $[0, T]$
11:             generate $v$ random between $[0, 1]$
12:             if $v \leq \lambda(t_{gen})/\lambda^*$ then
13:                 if (pol == 1) then
14:                     append$(S, {i, j, t_{gen}, 1})$
15:                 else
16:                     append$(S, {i, j, t_{gen}, 0})$
17:                 end if
18:                 append$(E_{sr}, S)$
19:             end while
20:         end if
21:         $i += 1$
22:         $j += 1$
23:     end for
24: end for
25: ${e_i}_M \leftarrow E_{sr}$
26: return ${e_i}_M$
```

types is primarily focused on encountered network types in the literature of related work and includes Multilayer Perceptron (MLP), CNN, GCN and Transformers. How well a network performs is quantified by a loss function and is tailored for the task that is ought to be performed.

2.2.1. Multi-Layer Perceptron (MLP). An MLP is the most elementary type of neural network and consists of layers with artificial neurons [36]. In each layer, each input is connected to each neuron and each neuron is in turn connected to each output entry. The mathematical operations are basic but together they form a network that enable the detection of complex patterns in data. Detected patterns can be used as features. Networks that consists of multiple layers are referred to as deep networks; the deeper a network the higher the ability to detect complex (or referred to as hidden) features. The mathematical operation of a MLP layer can be described as:

$$z = XW + b \tag{3}$$

where X is the input vector, W the weight vector and b the bias and z the output. This is a linear operator, a activation function $\sigma(z)$ is needed in order to detect non-linear patterns.

2.2.2. Convolutional Neural Network (CNN). A CNN layer was first proposed in [37]; it extracts features by convolving a set of kernels with learnable parameters over input in a sliding window fashion and prevents global spatial bias to where features are positioned in the input and a reduced amount of learnable parameters when compared to MLPs. CNNs are designed to extract features from data that have a grid-like topology and have features embedded in local spatial relationships. ResNet [38] is a deep CNN with residual connections. Deep refers to the existence of multiple layers in the network's architectures, and residual connections to the branched data flow through a CNN layer and a data flow around this layer which is concatenated afterward. It has applications for both super-resolution and classification tasks, although being initially designed to classify images. There are different variations concerning the number of convolutional layers often indicated as a postfix [18, 34, 50, 101, 154] after ResNet. Despite the fact that eventstreams are expected to have no explicit spatial ordered relationships presented, it is quite possible that CNNs can extract features if they are deep enough. The mathematical operation of a 1-D convolution layer can be described as:

$$z[n] = (X * K)[n] = \sum_{m} X[n-m]K[m]. \tag{4}$$

where X is the input vector of size n, K the kernel of size m and $z[n]$ the output at position n. The output is padded with zeros in order to prevent the kernel to reduce the dimensionality of the output by $m-1$.

2.2.3. Graph Convolutional Neural Network (GCN). A graph is a data structure existing of nodes and vertices. A vertex connects a node to its neighbor. A well-known graph is a map of train connections in which train stations are translated as nodes and the connections between train stations as graphs. Convolution operations as used in CNN can not be applied directly to this data since it is not structured in a matrix form. Graph Convolutional Neural Networks (GCN) solve this problem by applying a convolution operator on each node with a specified amount of closest neighboring nodes and resulting in neighbor-based feature extraction instead of grid-like spatial based as extracted using CNNs. Events presented in event-space (x, y, t, p) have a similar format as graphs which makes GCNs applicable to process events. In [39], a GCN architecture (specifically EdgeConv module from PyTorch Geometric) is used for the classification and segmentation of point clouds.

2.2.4. Transformer neural networks (Transformers). Transformers were first proposed in [40] and showed state-of-the-art (SOTA) results in natural language processing. Processing natural language or other types of sequences was done using Recurrent Neural Networks (RNN) [**recurrent**] and Long-Short-Term-Memory (LSTMs) networks prior to the invention of transformers. Transformers make use of self-attention (SA) mechanisms that selectively focus (attention) on the most relevant information. SA extracts features based on found dependencies between each independent element in a sequence to each other element of a sequence in the form of cross-relevance without being affected by the relative distance contrary to RNNs and LSTMs. In addition, transformers feature the ability to parallelizable computations (which decreases computation time) resulting in decreased computation time which is not possible using RNNs and LSTMs. Models using a transformer architecture for event-based data classification have been proposed by [30] and [31]. A major benefit of using transformers to process event-based data is that it is already encoded into quantitative values (x, y, t) which makes it directly usable as input, unlike natural language which has to be transformed into so-called tokens. The input of the self SA-layer (X with length N) is processed to obtain the Query ($Q \in \mathbb{R}^{N\times C}$), Key ($K \in \mathbb{R}^{N\times C}$) and Value ($V \in \mathbb{R}^{N\times N}$) matrices with channel (C) using convolutional layers. The self-attention mechanism ($\mathrm{Attention(Q,K,V)}$) is described as the following mathematical operation:

$$\mathrm{Attention(Q,K,V)} = \mathrm{softmax}(\frac{QK^T}{\sqrt{N}}V) \tag{5}$$

where ^{T} denotes a transpose operator and N is used as a scaling factor to prevent values from becoming too large. The dot product between Q and K realizes obtaining the cross relevance between events.

2.2.5. Training. The answer (or output) of a network, when given an input, is based on a prediction that follows from a series of mathematical operations with the learnable parameters (or simply referred to as parameters) of the network. To the set of parameters belongs W, b and K. The right set of parameters results in the right policy for successful predictions and is found during a process named training. This process requires 1) data consisting of paired input (X) and ground truth (Y) and 2) loss function $L(Y, \tilde{Y})$ which is a method to quantify how well or how poor related the predicted output ($\tilde{Y}$) is compared to Y. In the training process, the network is fed X after which $L(Y, \tilde{Y})$ is calculated. Then, the parameters of the network are adjusted according to the loss during back-propagation. This cycle is repeated multiple times. How much a network learns from each training cycle is controlled by the learning rate (*lr*) and is implemented as a factor to the loss. A network is said to be fully trained when it is converged towards an optimal solution indicated by the seemingly lowest possible obtained loss.

2.2.6. Loss functions. The loss function is used as a quantitative metric to evaluate how a prediction $\hat{Y}$ is related to the ground truth Y. Therefore, it is key that the used loss function reflects the essence of the aimed task in order for the network to converge to an optimal solution. Different tasks require different loss functions. In this research Cross-Entropy (CE) 6 will be used in the training of networks for the classification task. Chamfer Distance 7 will be used in the training networks for the task of super resolving. Mean Squared Error (MSE) 8 will be used as a similarity metric between the converted event-frames of the ground truth and predictions, it calculates the avarge quadratic error between corresponding entries and will not be used as a loss function during training.

$$L_{CE}(Y, \tilde{Y}) = -\sum_{i=1} y_i \log(\tilde{y}_i) \tag{6}$$

$$L_{CD}\left(Y, \tilde{Y}\right) = \frac{1}{n} \sum_{y \in Y} \min_{\tilde{y} \in \tilde{Y}} \|\tilde{y} - y\|_2^2 + \frac{1}{n} \sum_{\tilde{y} \in \tilde{Y}} \min_{y \in Y} \|\tilde{y} - y\|_2^2 \tag{7}$$

$$L_{MSE}(Y, \tilde{Y}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{y}_i)^2 \tag{8}$$

2.3. Suggested Pipeline

2.3.1. Feature extractor. The Feature extractor is the first module in this pipeline. As the name suggests, this module's priority is to extract features from eventstreams. Based on these features, the subsequent Coordinate reconstructor module predicts the coordinates of super resolved events and is thus dependent on the ability of the Feature extractor to extract a complete set of features. Currently, the Feature extractor is a black box, for which this study aims to find a solution. Yet, the researched solution has to comply with the set boundary conditions of input dimension $(N, C) = (N, 3)$ and output dimension $(N, C) = (N, 128)$.

2.3.2. Coordinate reconstruction. The Coordinate reconstruction module is linked to the Feature extraction module. It reconstructs the coordinates based on the extracted features which are directly fed as an input. Inspired by point-cloud upsampling architecture from PU-GCN [33], the input of dimensionality (N, C) is periodically shuffled (Reshape) towards the shape $(Nr, C/r)$. It is then fed into successive multibranch MLP-layers with shared parameters in which the channel dimension is contracted in the following order: 64, 16, 3. It has an output of dimension $(Nr, 3)$, a schematic of this module is depicted in Figure 5.

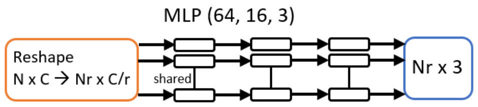


Figure 5. Overview of the Coordinate reconstruction module with Input eventstream length (N), feature channel (C), scaling ratio (r).

2.3.3. Polarity Reconstruction. The predicted coordinates of events by the Coordinate reconstruction module only entail a coordinate. The module Polarity reconstructor will be designed to predict the polarity of those events. Available information about the polarity of events in the low-resolution eventstream (E_{LR}) can be used in this task. One of the most simple and commonly used algorithms for classification (or labeling) in a certain latent space is *k-NN*. When this algorithm is fitted on the E_{LR}, it predicts the polarity of given events based on the k most frequent labels of its nearest neighbors in E_{LR}. Distance is calculated as Euclidean distance.

As *k-NN* classifies based on labels of nearest neighbors, it is key to assure those nearest neighboring events indeed indulge information. The suitability of this algorithm to be used as a Polarity reconstructor was tested in a small experiment prematurely as part of the literature study prior to this thesis study found in the Hypothesis section in Appendix F. In this small experiment, *k-NN's* ability to predict the polarity has been tested. Half of the events were masked after which the polarity was retrieved by *k-NN* with an accuracy of 93.8 %. Initially, this accuracy indicates that *k-NN* could be usable for predicting the polarity of events.

2.4. Experiments

The experiments are conducted using the GPU of Google Colab[1] The neural network architectures are defined using PyTorch[2] and extended 3rd party libraries Pytorch3d[3] and torch_geometric[4].

There are two types of tasks that will be performed during our experiments. The types are classifications in

1. https://colab.research.google.com/
2. https://pytorch.org/
3. https://pytorch3d.org/
4. https://pytorch-geometric.readthedocs.io/en/latest/

which the category (or label) of the data instance will be predicted, and super-resolution in which a higher resolution representation of a given input will be predicted. Classifying is often found an easier task compared to super resolving in computer vision. Therefore, we will use the classification of event-based vision as a starting point for our experiments to obtain insight into the applications of different networks on eventstreams. This insight enables a first selection of network types to be used as Feature extractors for the super-resolution task. The selected Feature extractors will be exploited on both N-MNIST and N-Caltech-101 in order to form a valid argumentation that the found solution is proper.

2.4.1. Data. The two neuromorphic datasets used in this research are N-MNIST and N-Caltech-101 [41]. Both datasets, N-MNIST and N-Caltech-101 [41], are neuromorphic vision conversions of two renowned datasets in computer vision; MNIST [42] and Caltech-101 [43] respectively. MNIST contains 70,000 images of handwritten digits (0, 1,.. 9) of resolution 28×28, it is used as a 'toy dataset' meaning that it is frequently used as a starting point for testing potential computer vision solutions early on. Caltech-101 is a more complex dataset that consists of 101 categories of images ranging from airplanes to dolphins and pizza with a resolution of 245×302 and is often issued in a late development stadium of computer vision solutions. The datasets are converted using a physical setup with an event camera recording a monitor which displays the image of the datasets. Since events are triggered by changing brightness in the pixels, the scenery has to be dynamic. To make the scenery dynamic, the event camera is moved over a micro distance at a fast pace, also named saccade, in a triangular motion. The duration of each recording is 300 ms and the spatial resolution of N-MNIST is 37×37 and N-Caltech-101 is 180×240. The N-MNIST dataset contains eventstreams with lengths in the order of 10^3 while N-Caltech-101 contains eventstreams with a length in the range of 10^4 to 10^5. For the experiments in this study, eventstreams will be preprocessed to make them have equal lengths and normalized values to lay in [0, 1]. These resulting eventstreams lengths depend on the testing condition. Also, each available category inside the dataset is represented by the same number of eventstreams to prevent categorical biases. For N-MNIST this will be in the range of $[1024, 4096]$ and for N-Caltech-101 in the range of $[12, 288, 49, 152]$.

2.4.2. Experiment 1. During the literature study prior to this thesis, an initial experiment was conducted to indicate the applicability of a *k-NN* algorithm to predict polarities of events which is included in the Hypothesis section of Appendix F. In that experiment, performance was measured by the accuracy at which *k-NN* could predict polarities of an eventstream from which half of the events had a masked polarity. Events to be masked were chosen at random. The result was an accuracy of 93.8%. In this experiment, additional testing will be conducted to measure the accuracy at which *k-NN* predicts the polarity of eventstreams from which a different portion is masked and at different values for k. The masked portion illustrates the super resolved coordinates and the portion with polarity illustrates the low-resolution input and the ratio between the masked portion and the portion with polarity illustrates the scaling ratio r. Tests will be conducted at test circumstances $r \in [2, 4, 8]$ and $k \in [1, 3, \cdots 9]$, the length of the masked portion eventstreams are fixed at $N = 4096$ and the portion with polarity is shaped proportional the the ratio. Results will indicate which setting for k results in the highest accuracy and how applicable the use of *k-NN* is in the suggested pipeline.

2.4.3. Experiment 2. Prior to this experiment, there is little known about the appliances of available standardized neural network architectures on eventstreams. This experiment is the starting point of this research in which we will perform classification on the N-MNIST dataset. With the purpose to obtain insight into which architectures can cope efficiently with the data structure of eventstreams and which are able to extract information from evenstreams. Argumentation as to which extent an architecture can efficiently cope with eventstreams can be deducted from the time it takes to do a full epoch in the training loop and also how many eventstreams can be processed parallel to each other (batch size). Insight into the ability of the network to extract information (also referred to as features) can be obtained by the measured accuracy at which the network can classify. These architectures are: ResNet [38], PointNet [44], Dynamical Graph CNN (DGCNN) [39] and Point Cloud Transformer (PCT) [27] and Transformer heavily based on [30].

X consists of eventstreams of equal lengths of $N = 1024$ with coordinates (x, y, t) and Y consists of one-hot encoded labels. There are in total 10,000 evenstreams, 1,000 of each class. The networks are trained for 10 epochs using Adam as an optimizer, $lr = 0.005$ and Cross-Entropy (CE) Equation 6 as loss function.

2.4.4. Experiment 3. This experiment contains all conducted tests concerning the task of super-resolution under various test circumstances and data. The results of these conducted tests will provide insight into the suitability and limitations of different architectures which will motivate design choices for further development of a solution. For super-resolution, the focus lies upon the discovery of network architectures to extract features that are needed to predict missing data in a low-resolution to form a complementary higher resolution. We will refer to these network architectures as Feature extractors 2.2. The Feature extractors are implemented as a module into the pipeline as depicted in Figure 3. The used Coordinate reconstructor is based on PU-GCN [33] and for the Polarity reconstructor 2.3.3 *k-NN* is used. The basic idea is that a neural network - consisting of Feature extractor and

Coordinate reconstruction 2.3.2 modules - is used to predict coordinates of the events in event-space (x, y, t), the polarity of these predicted coordinates are then labeled by applying K-NN fitted on the initial low-resolution eventstream. This results in an end-to-end pipeline that predicts a E_{HR} of dimension $(Nr, (x, y, t, p))$ with scaling ration r based on low-resolution input E_{LR} of dimension $(N, (x, y, t, p))$.

In this experiment, different network architectures will be tested on their performance to super resolve events, the performance is quantified by the Chamfer Distance (CD) Equation 7. The choice of Feature extractors is based on the findings in Experiment 2, these include PointNet [44], ResNet-18, ResNet-34 and PCT [27]. Overviews of the resulting network configurations are included in Appendix A. The capabilities of these Feature extractors will be exploited through testing on both N-MNIST and N-Caltech-101 under various test conditions. This test condition concerns the scaling ratio r between X and Y, the eventstream length N of X and different subsample techniques to sample $X \in Y$ from Y. The purpose is to get a better insight into both the behavior and limitations. This insight will enable the formation of stronger arguments about the legitimacy of our found solutions.

Variating r tests to which scale a network can super resolve and how this factor affects its performance and variating N tests how the performance of a network is relatable to the number of events in the input eventstream. Fewer events in the input limit the ability of the network to have a semantic understanding of the data e.g. in the case of cropping the eventstream by a half (output eventstream N = 2048) or a quarter (output eventstream N = 1024) the network has less information about what the evenstream is portraying. Variating N will give insight if the network is extracting features based on the underlying manifold/semantics or uses a policy in which it predicts point just around the input events. The last policy is motivated by the fact that since $x_i \in X$ are subsampled from Y, there is a higher chance that y_i are located around the x_i in the more dense areas in the event-space.

The two different subsample techniques include subsample at random and subsample based on the spatial coordinate address of the events. The first subsample technique results in a sparse representation of Y, the task for the algorithm is then to predict a more dense representation. Although it lacks a realistic comparison to the essence of super-resolving, this sample fashion enables subsample by an unbounded to integer scaling ratio which makes it favor exploiting the limitations of the Feature extractors at ease. The second subsample technique results in more realistic counterparts of low- and high-resolution eventstreams. In which events are sampled based on their address with respect to the observed position with respect to the physical sensor (x, y), this fashion of sampling is closely related to the essence behind super-resolving. For example, N-MNIST is observed by a sensor with dimensions (34 $\times$ 34), we will sub-sample to imitate how a sensor with dimensions of (17 $\times$ 17) would have observed. A visualization of sampling events by address is depicted in Figure 6. This sub-sampling method is refered to as True downsampling (TDS). By using TDS, downsampling E_{LR} from E_{HR} is based on pixel-address. A visualization of sampling by pixel-location from the spatial perspective is depicted in Figure 6. It can be described as $e_i \in E_{LR}$ if $e_{ix} \mod 2 = 0$ and $e_{iy} \mod 2 = 0$. The difference between subsampling by TDS and at random is observable in event-frames. However, as the data is processed directly in event-space, differences between these two sub-sampling techniques are expected to be minor.

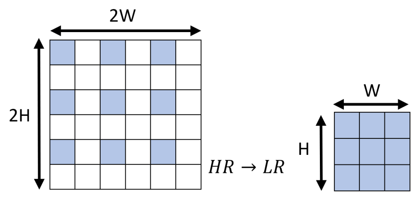


Figure 6. Visualization of the subsample process with respect to the sensor/spatial plane to obtain the low-resolution (LR) from the high-resolution (HR) eventstream.

Initially, the test condition for super resolving event-based vision is set at a scaling factor (r) of 4 on the input eventstream with length (N) of 1024 events sub-sampled at random on the N-MNIST dataset. These results are used to compare the results obtained in the altered test conditions later on in this experiment.

The altered test conditions can be divided into four sections. 1) r is altered while having a fixed output length of Y, this results in the following paired test circumstances $r \in [8, 4, 2]$ and $N \in [512, 1024, 2048]$. 2) r is fixed to be 4 but input length N is altered, this results in the following test circumstances $N \in [256, 512, 1024]$. 3) X is obtained by the TDS sub-sample technique at $r = 4$ and $N = 1024$. 4) The network configurations are trained and tested on a more complex dataset N-Caltech-101, which includes eventstreams with longer sequence lengths. In this test, $r = 4$ and $N = 1024$. For the fourth part, none of the network configurations are optimized nor adjusted for usage on this data in order to test how generalizable this research potentially is. In this test, $r = 4$ and $N = 3072$.

The different networks will be trained for 100 epochs using Adam optimizer and $lr = 0.005$ and CD Equation 7 as loss function. The network is combined with the Polarity reconstructor with $k = 3$ after it is trained to predict E_{HR}. Visualizations of both event-space and event-frames will be made. Event-frames are converted using Algorithm 1.

2.4.5. Experiment 4. In this experiment, the found solution for the pipeline will be compared to the earlier defined benchmark which is based on PPP [21]. Before testing, the network part of the solution will be optimized during hyperparameter tuning. The test will be conducted on the N-MNIST dataset which is downsampled by TDS at $r = 4$ and $N = 1024$. Quantification is made in two ways. (1) By CD on the predicted eventstream, this enables to compare the ability of the algorithm to predict the coordinates of the events. (2) MSE on converted event-frames, this metric is dependent on the algorithm's (complete pipeline including Polarity reconstructor) ability to predict coordinates and polarity. Together, these two metrics deliver an overcomplete quantization of performance to super resolve event-based vision. Based on the results, conclusions will be drawn about the applicability of the found solution to super resolve event-based vision in event-space.

3. Results

This section depicts the obtained results from the conducted experiments. Results will be further deliberated in the discussion chapter 4.

3.1. Experiment 3: *k-NN*

The results of this experiment in which the accuracy is measured at which *k-NN* can predict the polarity of an event stream from which a portion has a masked polarity are depicted in Table 1. An overview of the process is depicted in Figure 7.

TABLE 1. OBTAINED ACCURACY IN PERCENTAGES (%) WHEN USING *k-NN* TO PREDICT THE POLARITIES OF A EVENTSTREAMS

r	k				
	1	3	5	7	9
2	**98.2**	97.3	96.6	95.8	95.1
4	93.1	**93.7**	92.3	90.7	89.2
8	**90.3**	87.4	84.6	82.1	80.0

This table shows the obtained accuracy when using *k-NN* to predict the polarity of eventstreams with masked polarity based on a smaller subset with polarity. Tests are conducted at different scaling ratios r between the masked eventstream and its subset wit and set values for k.

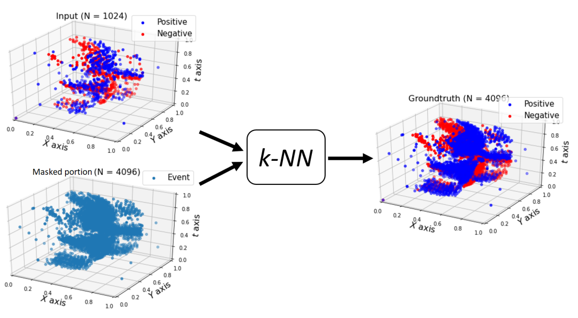


Figure 7. Overview of the predicting process of the polarities of event coordinates based (bottom left) on low-resolution (top left) input using *k-NN*.

3.2. Experiment 2: Classification N-MNIST

The results of the first experiment in which data from the N-MNIST dataset is classified are depicted in Table 2. Noticeable is the high accuracy of PointNet in combination with the lowest amount of parameters and a short epoch time. A second observation is the highest epoch calculation time of DGCNN in combination with the lowest accuracy which indicates its inability to process from event-streams and extract features. These observations encourage further investigation of PointNet, PCT and ResNet for their application to be used as a Feature extractor to vision super resolve eventstreams.

TABLE 2. RESULTS OF DIFFERENT NETWORKS ON THE CLASSIFICATION OF DATA FROM N-MNIST

Network	Param.[a] 10^6	Time[b] s	Acc.[c] %	B. size[d]
ResNet-18	7.69	15	63.5	**512**
ResNet-34	14.4	30	62.7	256
ResNet-50	16.0	87	56.6	64
ResNet-101	28.3	132	52.4	64
ResNet-152	38.5	215	42.4	32
PointNet	**0.68**	17	77.4	256
PCT	1.47	**14**	61.3	256
Transformer	1.47	65.2	59	64
DGCNN	1.8	320	41.2	8

[a]Number of learnable parameters
[b]Time it takes for 1 epoch to be calculated
[c]Final accuracy to predict labels in the testset
[d]Batch-size which is the maximum number of data-instance to be parallel processed overloading the GPU

This table shows various quantities obtained from the tested network configurations during the task of classifying data from N-MNIST.

3.3. Experiment 3: SR of eventstreams

3.3.1. Results the initial experiment on N-MNIST.

Performance to predict locations of the super resolution of the different Feature extractors measured by CD and are depicted in Table 3. The Feature extractor which uses the PCT-based architecture results in the highest performance. In several instances, the epoch calculation time of the PCT in the given test circumstances of input sequence length ($N = 1024$) resulted in the longest computation time contrary to Experiment 2 in which the input sequence length was shorter ($N = 256$). This difference is important to note since it reveals insight into the scalability of the network type as a function of input sequence lengths. This is important since an input sequence length of 1024 is small compared to the expected sequence lengths of eventstreams of more realistic event-base footage which is several times larger.

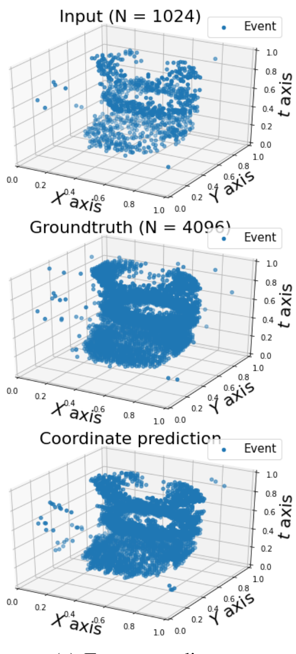
Visualization of the coordinates prediction of events of X_{coor}, Y_{coor}, and $\bar{Y}_{coor}$ by the PointNet Feature extractor are depicted respectively in Figure 8a. The polarity of $\bar{Y}_{coor}$ is retrieved using the *k-NN* module, and the result is visualized in Figure 8b. Event-frames are converted from event-space using Algorithm 1. Visualizations of event-frames predicted by the networks using the different Feature extractors are depicted in Figure 9. Visually, converted event-frames from the predictions of the networks using PCT and PointNet modules are similar to the same degree to its ground truth while predictions made using a ResNet module only slightly resemble the ground truth. These visual observations made from the event-frames are analog to the resulting losses.

TABLE 3. RESULTS OF DIFFERENT FEATURE EXTRACTORS ON SR OF N-MNIST

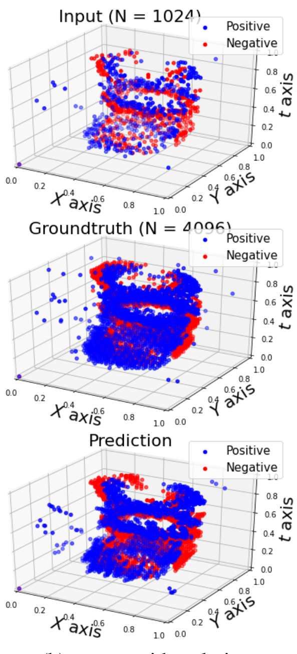
Feature extractor	Param.[a] 10^6	Time[b] s	CD[c] 10^{-3}	B. size[d]
ResNet-18	7.71	71	1.67	**256**
ResNet-34	14.46	121	1.83	128
PointNet	**0.88**	**55**	1.30	**256**
PCT	0.94	179	1.22	64

[a]Number of learnable parameters
[b]Time it takes for 1 epoch to be calculated
[c]Measured Chamfer Distance on testset
[d]Batch-size which is the maximum number of data-instance to be parallel processed overloading the GPU

This table shows various quantities obtained from the tested network configurations during the task of super resolving data from N-MNIST at scaling ratio $r = 4$ with eventstream length $N = 1024$.



(a) Event coordinates



(b) events with polarity

Figure 8. Visualization of input (X), Ground truth (Y) and Prediction ($\bar{Y}$) made using a PointNet based Feature exractor in event-space.

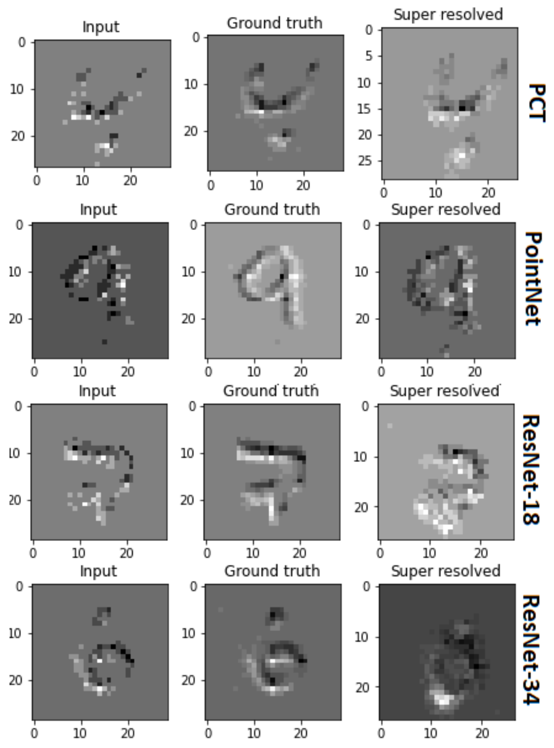


Figure 9. Event-frames converted at $t_{bin} = 10^{-3}s$ from super-resolution predictions made on data from N-MNIST with $r = 4, N = 1024$ using networks utilizing a PCT, PointNet, ResNet-18 and ResNet-34 Feature extractor.

3.3.2. Exploiting limitations of networks to super resolve. In order to exploit the limitations of networks to super resolve, an altered test was conducted which consists of 4 parts. Results are shown in Table 4.

In the first part, it was tested how the performance of a network, measured by the CD loss function, is affected by the scaling ratio r. In line with triviality, the resulting loss becomes larger when r becomes larger and vice versa. The test circumstances in which $r = 8$ and $N = 512$ resulted in the highest loss. Visualizations of the predicted eventstreams are shown in Figure 19 of Appendix B for both event-space and event-frames. This figure shows the difference in density between the input sequence and the ground truth which emphasizes the level of difficulty to super resolve at this scaling ratio.

In the second part correlation between input eventstream lengths N and the ability to super resolve was tested. Data shows all network configurations benefit from having a longer sequence length since it results in a lower loss. Test circumstances in which $r = 4$ and $N = 256$ resulted in the highest loss. Visualization of the predicted eventstreams are depicted in Figure 23 and converted event-frames in Figure 22 and 25.

In the third part network configurations were trained and tested on a dataset in which X is sampled by the true downsample (TDS) technique from Y. It was expected that performance would not be affected significantly, this expectation is supported by the quantified results in Table 4. Yet, Figure 11 depicts converted event-frames that show a checkerboard pattern to appear on the predictions made by the network which uses a PCT Feature extractor. Predicted events in event-space are shown in Figure 10.

In the fourth and final part of this experiment, a test is performed in which the networks were trained on the N-Caltech-101 dataset with $N = 3072$ and $r = 4$ and TDS as downsampling technique. Comparing the obtained loss between each network configuration gives a similar ranking as previous parts. Figure 12 depicts the event-space representation of a data instance from N-Caltech-101. Figure 13 depicts predicted event-frames on some of the more simple data instances from N-Caltech-101. Predictions of the network configuration on more complex datainstances of the N-Caltech-101 dataset are depicted in Figure 24 of Appendix B.

Solely based on the quantified loss obtained by the Chamfer Distance in extensive tests that have been conducted in this experiment, it can be concluded that PCT-based Feature extractor results in the best performing network and PointNet-based Feature extractor as second best. ResNet would be the least-suited Feature extractor to be implemented in our network.

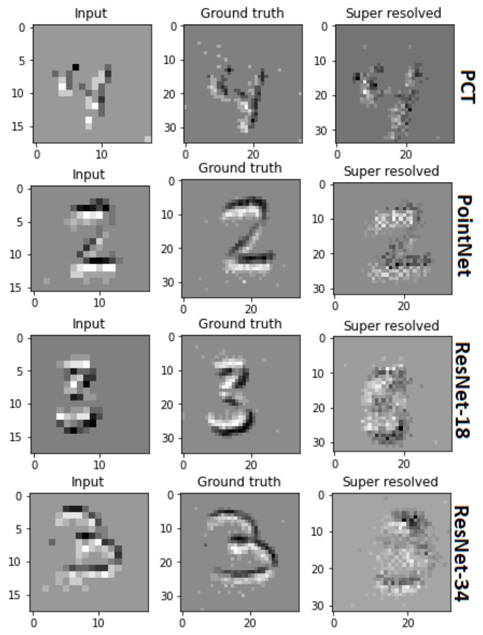


Figure 10. Event-frames converted at $t_{bin} = 10^{-3}s$ from super-resolution predictions made on data from N-MNIST with $r = 4, N = 1024$ subsampled by TDS using networks utilizing a PCT, PointNet, ResNet-18 and ResNet-34 Feature extractor.

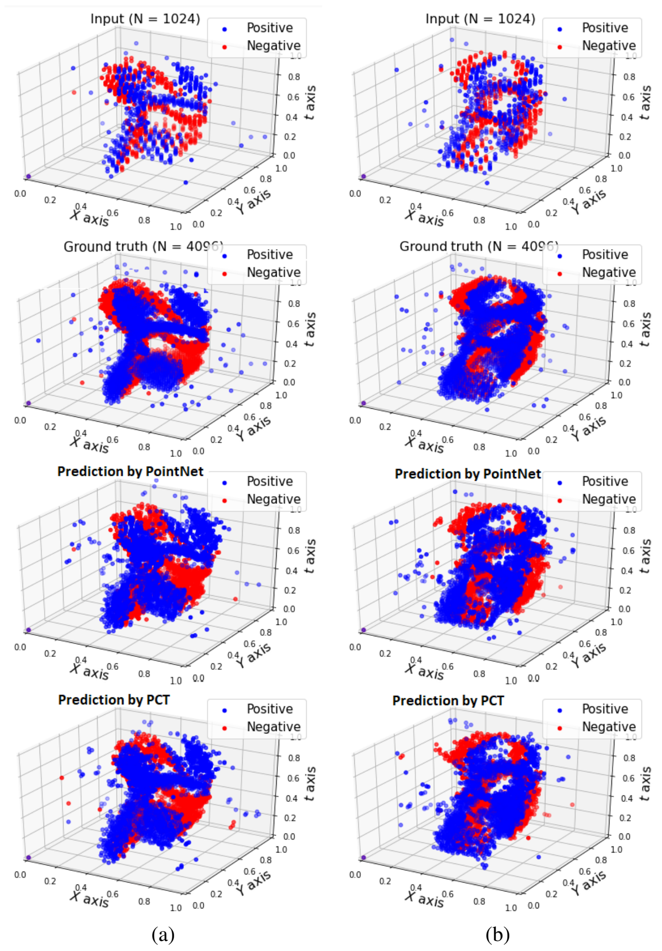


Figure 11. Super-resolution predictions made on data from N-MNIST with $r = 4, N = 1024$ subsampled by TDS using networks utilizing a PCT, PointNet, ResNet-18 and ResNet-34 Feature extractor.

TABLE 4. Resulting loss of Feature extractors on SR of N-MNIST under various test conditions

Test conditions	PointNet	ResNet-18	ResNet-34	PCT
$r = 8,\ N = 512$	1.50	1.81	1.70	**1.33**
$r = 4,\ N = 1024$	1.30	1.67	1.83	**1.22**
$r = 2,\ N = 2048$	0.83	1.30	1.42	**0.74**
$r = 4,\ N = 256$	3.34	4.68	4.4	**3.20**
$r = 4,\ N = 512$	**1.83**	2.53	2.64	1.85
$r = 4,\ N = 1024$	1.30	1.67	1.83	**1.22**
$r = 4,\ N = 1024$	1.30	1.67	1.83	**1.22**
TDS	**1.21**	1.74	1.76	1.18
N-Caltech-101	1.10	1.89	1.75	**1.04**

This table shows obtained Chamfer Distance (10^{-3}) between the super resolved- and the ground truth eventstream when using PointNet, Resnet-18, ResNet-34 and PCT as a Feature extractor in the network. Tests are conducted using different test conditions defined by scaling ratio r, input eventstream N, subsample technique TDS and dataset N-Caltech-101.

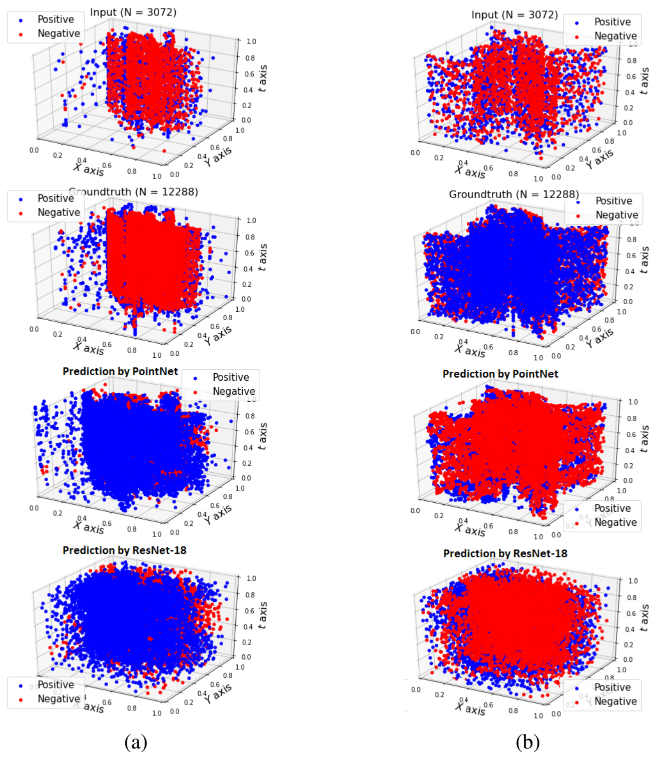


Figure 12. Super-resolution predictions made on N-Caltech-101 with $r = 4, N = 3072$ subsampled by TDS using networks utilizing a PCT, PointNet, ResNet-18 and ResNet-34 Feature extractor.

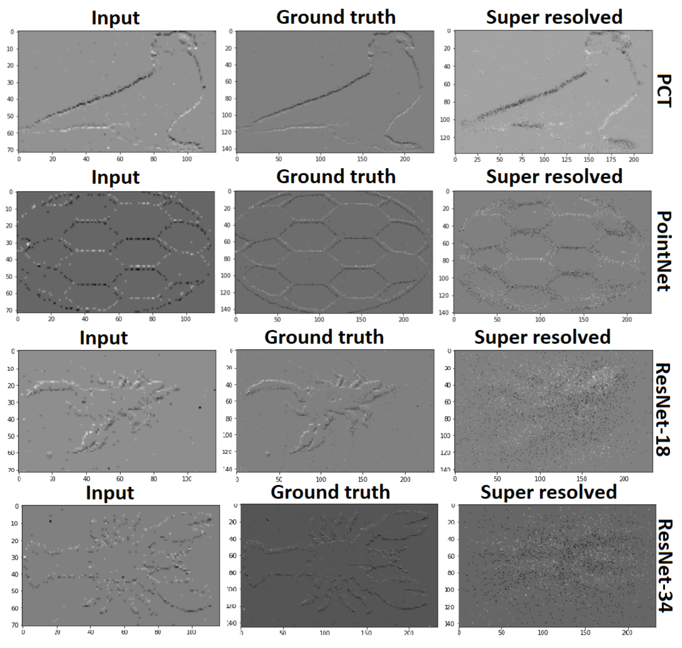


Figure 13. Event-frames converted at $t_{bin} = 10^{-3}s$ from super-resolution predictions made on data from N-Caltech-101 with $r = 4, N = 3072$ subsampled by TDS using networks utilizing a PCT, PointNet, ResNet-18 and ResNet-34 Feature extractor.

3.4. Experiment 4: Benchmark solution to the baseline method

The obtained pipeline to super resolve event-based vision will be compared in this final test to the defined baseline method. The baseline method uses PPP characteristics to supersample given input as described in section 2.1. The pipeline final configuration of the obtained network architecture obtained after hyperparameter tuning is depicted in Figure 14. The process of hyperparameter tuning is included in Appendix D in which altered configurations of hyperparameters were exploited. However, the tested altered network configuration did not result in performance gains. Consequently, the tuned architecture is the same as the initial (vanilla) architecture. The network was then trained on an augmented dataset which led to convergence to a lower loss value, for consecutive 50 epochs with a lr=0.005, 50 epochs with lr=0.0035 and 50 epochs with lr=0.002 using *Adam* as optimizer. The trained network is used next to the PPP-based method in the final test from which its results are depicted in Table 5. Visualization of the predictions made by PPP and trained network (referred to as proposed model) in combination with the given Input and Ground truth is depicted in Figure 15.

Data shows that the found network, as opposed to the baseline method, was able to predict the coordinate of super resolved events more similar to the ground truth with a resulting Chamfer Distance of 0.779e-3 vs 2.034e-3 at a smaller standard deviation 0.779e-3 vs 2.034e-3 in the given test circumstances with $r = 4$ and $N = 1024$.

TABLE 5. RESULTS OF DIFFERENT NETWORKS ON THE CLASSIFICATION OF N-MNIST

Algorithm	CD		MSE	
	avg[a] (μ) 10^{-3}	std[b] (σ) 10^{-4}	avg (μ)	std (σ)
PPP	2.034	1.869	1.511	0.5491
PCT	0.779	0.549	0.336	0.0620

[a]average
[b]standard deviation

This table shows obtained averages and the standard deviations of the measured Chamfer Distance and Mean Squared Error between the ground truth and predictions.
obtained on N-MNISTS validation set under conditions defined by scaling ratio r, input eventstream N, using the PPP basline method and the proposed pipeline with PCT implemented as Feature extractor.

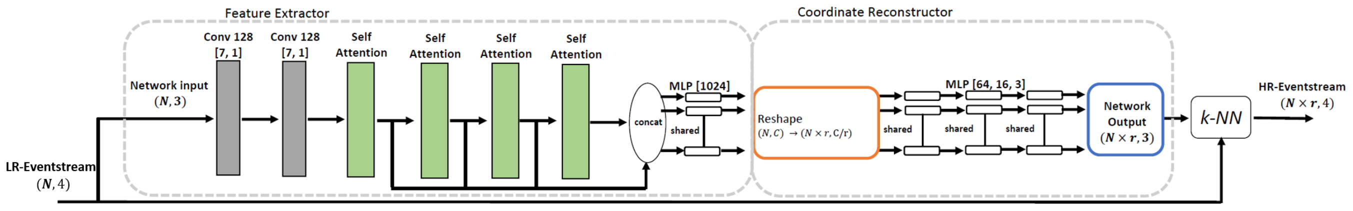


Figure 14. Overview of the obtained network configuration implemented in the proposed pipeline.

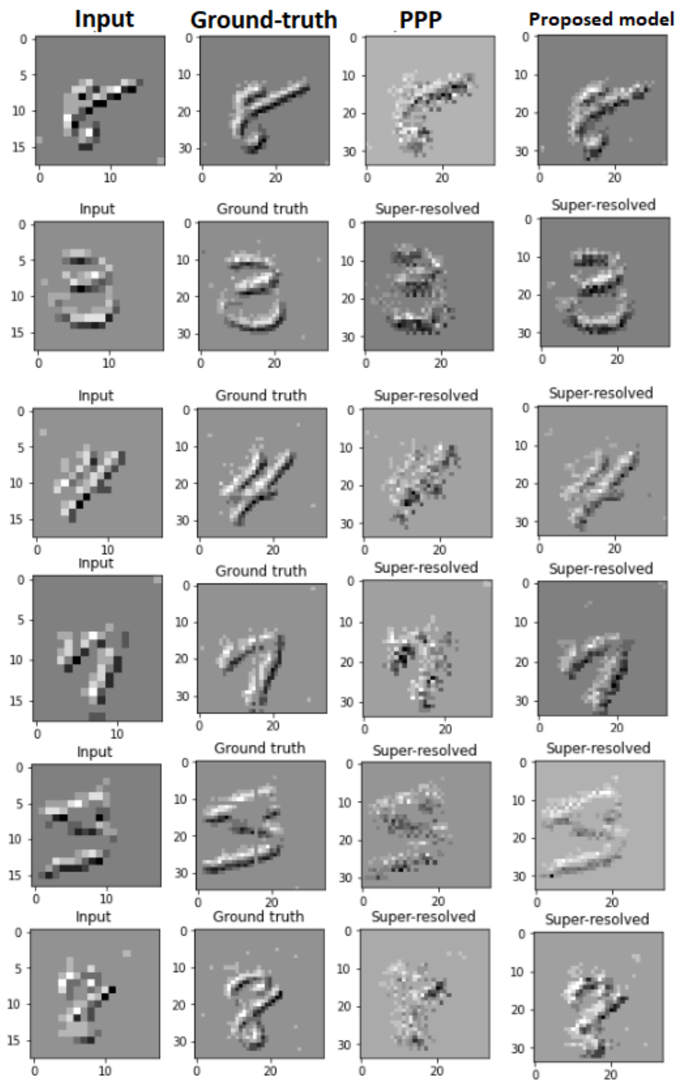


Figure 15. Event-frames converted at $t_{bin} = 10^{-3}s$ from super-resolution predictions made on data fromN-MNIST with $r = 4, N = 1024$ subsampled by TDS using PPP and the proposed model.

4. Discussion

In this study, methods are explored to super resolve event-based vision in event-space. The research gap was broad as no solution had been proposed to this problem prior to this research. This resulted in a large solution space in which various methods had to be considered.

4.1. Experiment 1

Experiment 1 was conducted in order to test the applicability of *k-NN* when used to predict the polarity of events, results are depicted in Table 1. The main focus of this study is to super resolve at a scaling ratio for 4 ($r = 4$) for which the highest accuracy is obtained (93.7%) when k= 3. A secondary choice would be the use of k= 1 since it performs second best when in test circumstances with $r = 4$ and best when $r = 2$ and $r = 8$. Another observation is the strong correlation between a higher r and a decrease in the accuracy at which *k-NN* can predict the polarity. Although this correlation can be explained as a result of more information missing when r becomes larger. Yet, this correlation should be noted as it indicates that alternatives should be researched when the aim is to super resolve at higher scaling ratios.

4.2. Experiment 2

The aim of Experiment 2 was to filter out potential networks from a large considered selection. Initial filtering was based the network's ability to process event-based data efficiently and extract features from it measured by respectively the calculation time per epoch and the accuracy of the network to classify N-MNIST. Results from this initial experiment, shown in Table 2, indicate the potential applicability of ResNet-18, ResNet-34, PointNet and PCT for further research.

Notably, the PointNet configuration resulted in higher accuracy than the ResNet configuration while having significantly fewer parameters. This result is counter-intuitive since both configurations rely on convolutional layers as a backbone and the fact that ResNet consists of ten times more parameters which should enhance the ability to extract features and thus logically predict more accurately accordingly. This observation indicates that PointNet extract features more effectively. Also, data suggest an inverse relationship between the complexity of the used ResNet variant and the obtained accuracy. This inverse relationship may be due to a combination of training with an insufficient large dataset (containing of 10,000 data instances) and a limited number of training epochs which leads to insufficient resources to train larger networks. Another explanation can be that convolutional networks do not benefit from deep architectures (i.e. multiple layers) when used for classifying a relatively simple event-based dataset like N-MNIST. However, it is plausible that these deeper architectures would be beneficial for other tasks such as super-resolution. Therefore ResNet-18 and ResNet-34 were included in this initial selection despite lower measured performance compared to the PointNet configuration.

4.3. Experiment 3

In Experiment 3 various tests were conducted to exploit the limitations of the selected network types when used as a Feature extractor module in the suggested pipeline. The data in Table 3 shows that, contrary to the results of Experiment 2, the PCT configuration has the longest epoch time. This can be attributed to the fact that the computational complexity of transformers increases quadratically to the sequence length of the input. Since the sequence length of the input was increased from 256 in Experiment 2 to 1024 in this experiment, a transformer architecture needs roughly 16 times more time to do the same calculation. This property of transformers is important to note since it results in poor scalability with respect to longer sequence lengths and could potentially make this type of architecture impractical due to long calculation times.

When comparing the converted event-frames in Figure 9, it can be seen that the PCT and PointNet configurations performed similarly. Both enable the prediction of event coordinates to match the appearance of the digit in the ground truth. Yet, the predicted event-frame had a noisy appearance which does not reflect the same smooth gradient as appeared in the ground truth. The noisy appearance is possible due to either wrongly predicted polarities or an inconsistent number of events occurring at neighboring spatial addresses. In the case of wrongly predicted coordinates, this would be the result of the used polarity reconstructor and is instantiated outside of the network. If this appearance is due to an inconsistent number of predicted events, it would be caused by the network and could be vanished in the final solution when the network is optimized and fully trained. As for the converted event-frames from the ResNet predictions, the digit appears on different coordinate locations as ground truth. Also, it misses out on details and only has a similar shape from a high-level point of view.

In additional tests consisting of four parts, the ability of the networks to super resolve under different scaling ratios (r), input sequence lengths (N), data that is downsampled with respect to the spatial address, as well as more complex data from N-Caltech-101, was evaluated.

4.3.1. First part. The first part of the additional tests researched to what extent the scaling ratio affects the networks. It was found that lower scaling ratios result in a lower number of data points to be predicted, which is inherently easier. Nevertheless, the obtained loss did not vary significantly between tests with $r = 4$ and $r = 8$. Furthermore, ResNet-34 even obtained a lower loss in the test with $r = 8$ than in $r = 4$. This variation in results can be attributed to the volatility of the obtained learning curve in combination with the non-deterministic nature of the training-process. Despite the small difference in the obtained result between $r = 8$ then in $r = 4$, it indicates tested network configurations' indifference to higher scaling ratios. However, there is a catch that can explain this observation, the used technique to downsample the data was at random as opposed to based on spatial address. When a network is trained on this resulting dataset, it learns how to predict a more dense event stream based on a sparse input. The sparse dataset still holds a certain level of information about the perimeter of the depicted digit, which can be seen in the input column in Figure 18. It is likely that the visible perimeter is used by the network as a feature to predict features, thus leading to indifference in predicting under higher scaling ratios. A downsample technique based on the spatial address of events was utilized in the third and fourth parts of this experiment and the final experiment to test under more realistic test circumstances.

4.3.2. Second part. In the second part of the additional tests, the relationship between input sequence lengths and the obtained loss was tested. The results in Table 4 indicate that all networks benefit from having a larger input sequence length. This observation supports the argument that all networks do take semantics into account when super resolving. Another plausibility is that the calculation of the loss is biased towards spatial correctness when N is larger. In both cases (relatively shorter and longer input sequence lengths) the maximum values are bounded to equal 1 which is not compensated in the loss calculation, resulting in the calculation of the loss being more biased towards spatial correctness when N is larger. Since spatial coordinate is assumable to be easier to predict as opposed to temporal coordinate, the loss would be probably lower. While the essence of this test seems to be ambiguous, it still has relevance as a sanity check and is insightful in the behavior of the tested network configurations.

4.3.3. Third part. In the third part of Experiment 3, networks were tested on a dataset that was downsampled with respect to the spatial address of events (TDS). This downsampling technique was different from the previously used downsample technique which was based on sampling at random. The obtained loss from this test, with the same test circumstances ($r = 4$, $N = 1024$), were similar to the results obtained from the previous test as can be seen in Table 4. However, upon inspection of the visualized results shown in Figure 11 and Figure 25 in Appendix B, it was observed that the shown predictions made by PointNet Feature extractor displayed distortions. Since only the sub-sample technique prior to these tests changed, it is likely that this effect is highlighted as a result of the TDS subsample technique. This checkerboard pattern suggests a failing in the policy of the PointNet network configurations to super resolve. Assumable, this pattern indicates that the network uses a policy in which it super resolves by simply predicting the location of the events to fall into the same coordinates as the events in the input.

4.3.4. Fourth part. In the final part of Experiment 2, a different dataset is used that includes eventstreams with much larger sequence lengths that are captured at a higher spatial resolution. The results are depicted in Table 4. It should be noted that the resulting loss should not be compared to the obtained losses in previous tests as this would result in misleading interpretations due to the differences in N-MNIST and N-Caltech-101 datasets. These differences originate in the fact that events in eventstreams in N-Caltech-101 are to a higher degree scattered distributed compared to N-MNIST which is observable when comparing Figure 10 to Figure 12. As the scattered distribution is more closely related to a homogeneous distribution, predictions are inherently closer related which yields a lower loss. Both PointNet and PCT network configurations seem capable of super resolving more complex eventstreams like N-Caltech-101. Nevertheless, it should be noted that the quadratic scaled computational complexity of the transformer architecture resulted in 12 hours for PCT to train for 100 epochs as opposed to 1 hour for the PointNet-based network configuration.

To summarize the observations, the PointNet and PCT-based configurations perform on par with respect to the resulting loss throughout all conducted tests in Experiment 3. still, it is important to note that both have their own shortcomings. The usage of a PCT module results in quadratic scaled computational complexity which possibly makes this module obsolete when super-resolving much larger eventstreams ($N >> 10,000$). While the predictions made by the PointNet configuration show, specifically in the event-frames, distortions to appear in a checkerboard pattern. Speculations about the origin of the distortions are formulated which argue it to be due to a too simplistic found policy by the PointNet configurations. But explanations of observed behavior based on theoretical knowledge of the concerning network dynamics are necessary in order to form solid argumentation and be able to decide between the PCT- and PointNet-based Feature extractor for further development. This is done in Appendix C.

The results from the additional tests in this experiment and the detailed theoretical knowledge in Appendix C support the hypothesis that the PointNet-based network lacks semantic awareness. This lack of semantic awareness results in a too simplistic policy to super resolve events which is motivated by the perceived checkerboard effect on predictions made by the PointNet-based network. This policy rather predicts super resolved points around the coordinates of the given inputs instead of predicting based on semantics. The lack of semantic awareness is caused by kernel sizes in the convolution layers (only the first convolutional layer has a kernel of 9 while the rest of 1) in combination with the channel-wise MLP layers which isolates data in each neuron and avoid data to flow trough parallel neurons. Attempts to prevent this isolation in order to enhance semantic awareness did not seem to work out but can be used for further research. While transformers, like PCT, use layers named self-attention modules which selectively focus on the most relevant information per event with respect to each other event. Self-attention incorporates the semantics of the input into the predictions and enables the network to use a more advanced prediction policy. A small ablation study was performed in an attempt to gain insight and explain the used policy by the obtained network, this is included in Appendix E. Unfortunetaly, the abblation study did not result in a explicit understanding of the used policy. Based on the theoretical knowledge and additional test, the decision was made to choose PCT for further development.

4.4. Experiment 4

After the hyperparameter tuning process which is included in Appendix D, the solution in the form of a pipeline consisting of a PCT-based Feature extractor, Coordinate reconstruction based on PU-GCN and a Polarity reconstructor based on *k-NN* with $k = 3$ is our proposed solution. This solution is depicted in Figure 14 and was compared to the baseline method, based on principles from PPP, in Experiment 4. It should be noted that the proposed solution is compatible with CUDA[5] which enables parallel computations which is not possible with the baseline method. This deviancy results in a large computation time of approximately 15 seconds for a single data instance while the proposed solution can process data instances almost instantaneously. Results show that the proposed solution performed superior compared to the baseline method in every quantified aspect and provides a proof of

5. CUDA (Compute Unified Device Architecture) is a parallel computing platform

concept of this pipeline. Also from a visual perspective when looking at the converted event-frames, the proposed solution retrieves details from the low-resolution input and thereby contributes towards enhanced quality of the event-stream which is debatable for baseline method as can be seen in Figure 15. However, the baseline method enables sequence-to-sequence data handling, which results in the ability to handle arbitrary lengths of eventstreams and predict variable lengths of SR eventstreams. While the proposed pipeline is restricted to vector-to-vector data handling which means that it is both configured and trained for fixed input and output lengths. In this research, the scope was set to super-resolve with respect to the spatial resolution by factor 2. This translate to the practical interpretation that a solution is researched which enables eventstream captured by a low-resolution sensor of size (W, B) to be predicted as it was captured by a high-resolution sensor of size $(2W, 2B)$. Although there are 4 times as many pixels, the resulting sequence lengths of the eventstreams captured by an HR sensor are not strictly fixed by factor 4. The inability of the proposed solution to process a sequence is a disadvantage but can be overcome when processing large eventstreams in parts in a convolving/sliding fashion. Another caveat of the comparison, is the degree of underdevelopment of the baseline method as opposed to the suggested pipeline. Significant performance gains are expected to be realized from the baseline method if it is developed further. This expectation is based on the fact that the publishers of this method claimed far better performance in their paper [21] as we were able to realize.

5. Conclusion

Event-based vision has many promising vision properties which makes it potentially applicable for a wide range of computer vision implementations. The limiting factor of an event-based camera, however, is the spatial solution. This research aimed to overcome this limiting factor by super resolving the spatial resolution of eventstreams. The solution was suggested to be in the form of a pipeline. From a high-level point of view, this pipeline consists of three distinctive modules, namely: Feature extractor, Coordinate reconstructor and Polarity reconstructor. In order to find solutions for these modules, the following sub-questions were formulated:

1) To what extent can a naive algorithm be used to super resolve events or would it suffice to have a learning-based algorithm?

Literature showed a proposed naive algorithm that uses the Thinning sampling technique to super resolve eventstreams [21]. Implementing their proposed method did yield a solution that could super resolve events, but visually this did not match the proclaimed results from the paper. This naive method was used as a baseline method and functioned as a benchmark of the solution that was eventually found. This suggests that a learning-based algorithm would result in a more sufficient solution.

2) Which type of neural network is best capable to extract features in event-based data when used in the Feature extractor module?

Various experiments have been conducted in Experiment 2 and Experiment 3 in order to find the answer to this sub-question. After analyzing the results, it was concluded that a PCT-based network was most capable to extract features from event-based data. This conclusion was supported by theoretical knowledge that transformers enable the finding of dependencies in data that do not follow a grid-like structure like eventstreams.

3) To what extent can we predict the coordinates of events of high-resolution eventstream based on a given feature-set using a Coordinate reconstructor module?

Throughout this research, a module inspired by PU-GCN [33] was issued which performed reasonably well when implemented in the suggested pipeline. Arguably, this research lacks a dedicated test to benchmark the used solution. At the same time, it was not part of the primary scope of this research as it performed reasonably well right from the start and was not considered a bottleneck.

4) To what extent can we retrieve polarity after super-resolving using k-NN as a Polarity reconstruction module?

Initial tests in the literature research as part of this study proved the applicability of the use of *k-NN* to retrieve polarity and additional tests were conducted in Experiment 1. Results from the additional tests confirmed this initial proof of concept in a more extensive fashion. Nevertheless, the use of *k-NN* as a Polarity reconstructor is limited in this pipeline which scales at $r = 4$. For larger r, the *k-NN* algorithm is not suitable as the accuracy drops significantly for which alternative methods should be researched.

The answers to the sub-questions enable answering the main question which was formulated as:

To what extent and by which method can event-based vision be super resolved in event-space?

A solution that enables spatial super-resolution of event-based vision in event-space was found after implementing the used methods used to answer sub-questions 2, 3, 4 as modules in the stated pipeline. The found pipeline was compared with the baseline solution in Experiment 4 in which performance was measured in the form of two metrics. Firstly, Chamfer Distance measured the distance between the predicted and ground truth eventstreams and thereby quantified how similar the predictions were. Secondly, Mean Square Error measured the similarity between the converted event-frames from the super resolved and ground truth eventstreams which is dependent on the ability to predict both the coordinates and the polarity of the events and thereby quantified the overall similarity. Based on results from both metrics, it was concluded that the found solution was superior to the baseline method. In addition, from a visual perspective, the converted event-frames from the super resolved eventstreams depict a more detailed version of the low-resolution input. Therefore, the found solution is considered to enhance the spatial resolution from a practical perspective also. The found solution should be considered as a starting point in further research toward the super-resolution of event-based data and thereby contributes to the extension of application possibilities of event-based vision.

6. Recommendations

The proposed model in this solution is a mere first contribution towards super-resolution of event-based vision. The main focus of this study was on finding the most suitable type of neural network architecture to be used as a feature extractor. As a result, the other modules in our proposed pipeline, Coordinate reconstructor and Polarity reconstructor, received less attention. These modules were optimized during the hyper-parameter tuning, but this was limited to the settings within the algorithm itself and lacks consideration of different types of algorithms. This leaves room for research towards alternative solutions for the Coordinate reconstructor and Polarity reconstructor. Therefore, in further research, we highly recommend considering our proposed pipeline as a starting point and researching alternative methods in order to optimize the Coordinate reconstructor and Polarity reconstructor.

Also, solutions can be researched to counter the caveat of quadratic scalability of the computational complexity with respect to the input sequence length which makes usage of its transformers on long sequence lengths possibly obsolete. We expect that sliding box operation can be used to process long sequence eventstreams. Another possibility is to use an operation like *k-NN* before the transformer which limits the calculation of dependencies of each event to only the k nearest neighbor. A similar method is applied in PU-GCN [33]. Unfortunately, the publishers have not made the code of PU-GCN publicly available yet and an attempt to reproduce this architecture failed due to the absence of the needed functions in supporting standard libraries in order to define the needed modules. It is, however, expected that the concerning code will be made publicly available at the time their paper is accepted and standard libraries will adopt these functions which enable reproducing PU-GCN.

Lastly, to make the solution generalize better, a downsample technique with inflicted noise into X can be considered.

Acknowledgments

Hereby I would like to take the opportunity to express my gratitude to N. Tömen. Her support and guidance were always valuable. Her patience was essential to keep me motivated for the completion of this thesis. My gratitude is also extended to J. de Winter for his role as a formal supervisor. His supervision with respect to the MSc Robotics program kept me on track. Yet, besides his formal role, I want to thank him for his genuine interest in this research topic and his understanding of the problems I encountered.

M. Souman provided valuable insight into my research by having weekly meetings and discussing my work on a high level and personal level which I am sincerely thankful for. His input and support have been instrumental in helping me to see the big picture of my activities throughout my research. His support was made possible by the company Alten which provided me with a place to work and the opportunity to interact with colleagues. Working at the office countered the feeling of being on my own.

Lastly, I would like to thank H. Argahi for his willingness to meet with me. His listening ear and sharp questions helped me to have a critical attitude toward my own work. It was a privilege to have the opportunity to ask for feedback and consulting from both N. Tömen and H. Argahi.

References

[1] Guillermo Gallego et al. "Event-based Vision: A Survey". In: (Apr. 2019). DOI: 10.1109/TPAMI.2020.3008413. URL: http://arxiv.org/abs/1904.08405%20http://dx.doi.org/10.1109/TPAMI.2020.3008413.

[2] Tobi Delbrück et al. "Activity-driven, event-based vision sensors". In: *ISCAS 2010 - 2010 IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems*. 2010, pp. 2426–2429. ISBN: 9781424453085. DOI: 10.1109/ISCAS.2010.5537149.

[3] Florian Mahlknecht et al. "Exploring Event Camera-based Odometry for Planetary Robots". In: (Apr. 2022). URL: http://arxiv.org/abs/2204.05880.

[4] Bas J. Pijnacker Hordijk, Kirk Y.W. Scheper, and Guido C.H.E. de Croon. "Vertical landing for micro air vehicles using event-based optical flow". In: *Journal of Field Robotics* 35.1 (Jan. 2018), pp. 69–90. ISSN: 15564967. DOI: 10.1002/rob.21764.

[5] Antoni Rosinol Vidal et al. "Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High Speed Scenarios". In: (Sept. 2017). DOI: 10.1109/LRA.2018.2793357. URL: http://arxiv.org/abs/1709.06310%20http://dx.doi.org/10.1109/LRA.2018.2793357.

[6] Ana I. Maqueda et al. "Event-based Vision meets Deep Learning on Steering Prediction for Self-driving Cars". In: (Apr. 2018). DOI: 10.1109/CVPR.2018.00568. URL: http://arxiv.org/abs/1804.01310%20http://dx.doi.org/10.1109/CVPR.2018.00568.

[7] Yuxuan Chen, Igor Gilitshenski, and Alexander Amini. *Real World Application of Event-based End to End Autonomous Driving*. Tech. rep.

[8] Guang Chen et al. "Event-Based Neuromorphic Vision for Autonomous Driving: A Paradigm Shift for Bio-Inspired Visual Sensing and Perception". In: *IEEE Signal Processing Magazine* 37.4 (July 2020), pp. 34–49. ISSN: 15580792. DOI: 10.1109/MSP.2020.2985815.

[9] Guang Chen et al. "NeuroIV: Neuromorphic Vision Meets Intelligent Vehicle Towards Safe Driving with a New Database and Baseline Evaluations". In: *IEEE Transactions on Intelligent Transportation Systems* 23.2 (Feb. 2022), pp. 1171–1183. ISSN: 15580016. DOI: 10.1109/TITS.2020.3022921.

[10] Henri Rebecq et al. "High Speed and High Dynamic Range Video with an Event Camera". In: (June 2019). URL: http://arxiv.org/abs/1906.07165.

[11] Christoph Posch et al. "Retinomorphic event-based vision sensors: Bioinspired cameras with spiking output". In: *Proceedings of the IEEE* 102.10 (Oct. 2014), pp. 1470–1484. ISSN: 15582256. DOI: 10.1109/JPROC.2014.2346153.

[12] C. Posch. "Bio-inspired vision". In: *Journal of Instrumentation* 7.1 (Jan. 2012). ISSN: 17480221. DOI: 10.1088/1748-0221/7/01/C01054.

[13] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. "A 128 $\times$ 128 120 dB 15 μs latency asynchronous temporal contrast vision sensor". In: *IEEE Journal of Solid-State Circuits* 43.2 (Feb. 2008), pp. 566–576. ISSN: 00189200. DOI: 10.1109/JSSC.2007.914337.

[14] Christian Brandli et al. "A 240 $\times$ 180 130 dB 3 μs latency global shutter spatiotemporal vision sensor". In: *IEEE Journal of Solid-State Circuits* 49.10 (Oct. 2014), pp. 2333–2341. ISSN: 00189200. DOI: 10.1109/JSSC.2014.2342715.

[15] IEEE Circuits and Systems Society and Institute of Electrical and Electronics Engineers. *2020 IEEE International Symposium on Circuits and Systems (ISCAS) : proceedings : ISCAS 2020 : Virtual Conference, October 10-21, 2020.* ISBN: 9781728133201.

[16] Peiqi Duan et al. *EventZoom: Learning to Denoise and Super Resolve Neuromorphic Events*. Tech. rep. URL: https://sites.google.com/view/EventZoom.

[17] Jin Han et al. *EvIntSR-Net: Event Guided Multiple Latent Frames Reconstruction and Super-resolution*. Tech. rep.

[18] Lin Wang, Tae-Kyun Kim, and Kuk-Jin Yoon. *EventSR: From Asynchronous Events to Image Reconstruction, Restoration, and Super-Resolution via End-to-End Adversarial Learning*. Tech. rep. URL: https://github.com/.

[19] S I Mohammad Mostafavi GIST et al. *Learning to Super Resolve Intensity Images from Events https://github.com/gistvision/e2sri*. Tech. rep. URL: https://github.com/gistvision/e2sri.

[20] Hanme Kim, Stefan Leutenegger, and Andrew J Davison. *Real-Time 3D Reconstruction and 6-DoF Tracking with an Event Camera*. Tech. rep. URL: https://youtu.be/LauQ6LWTkxM?t=35s.

[21] Hongmin Li et al. *Super-resolution of spatiotemporal event-streamimagecaptured bytheasynchronous temporal contrast vision sensor*. Tech. rep.

[22] Yongcheng Jing et al. *Turning Frequency to Resolution: Video Super-resolution via Event Cameras*. Tech. rep. URL: https://osf.io/6c3d9/,.

[23] Lequan Yu et al. *PU-Net: Point Cloud Upsampling Network*. Tech. rep.

[24] Shuquan Ye et al. "Meta-PU: An Arbitrary-Scale Upsampling Network for Point Cloud". In: *IEEE Transactions on Visualization and Computer Graphics* (2021). ISSN: 19410506. DOI: 10.1109/TVCG.2021.3058311.

[25] Ruihui Li et al. "PU-GAN: a Point Cloud Upsampling Adversarial Network". In: (July 2019). URL: http://arxiv.org/abs/1907.10844.

[26] Bochen Xie et al. "VMV-GCN: Volumetric Multi-View Based Graph CNN for Event Stream Classification". In: *IEEE Robotics and Automation Letters* 7.2 (Apr. 2022), pp. 1976–1983. ISSN: 23773766. DOI: 10.1109/LRA.2022.3140819.

[27] Meng-Hao Guo et al. "PCT: Point cloud transformer". In: (Dec. 2020). DOI: 10.1007/s41095-021-0229-5. URL: http://arxiv.org/abs/2012.09688%20http://dx.doi.org/10.1007/s41095-021-0229-5.

[28] Xumin Yu et al. "PoinTr: Diverse Point Cloud Completion with Geometry-Aware Transformers". In: (Aug. 2021). URL: http://arxiv.org/abs/2108.08839.

[29] Yin Bi et al. *Graph-Based Object Classification for Neuromorphic Vision Sensing*. Tech. rep.

[30] Zhihao Li, M Salman Asif, and Zhan Ma. *Event Transformer*. Tech. rep.

[31] Alberto Sabater, Luis Montesano, and Ana C Murillo. *Event Transformer. A sparse-aware solution for efficient event data processing*. Tech. rep. URL: https://github.com/AlbertoSabater/EventTransformer.

[32] Samira Pouyanfar et al. *A survey on deep learning: Algorithms, techniques, and applications*. Aug. 2018. DOI: 10.1145/3234150.

[33] Guocheng Qian et al. "PU-GCN: Point Cloud Upsampling using Graph Convolutional Networks". In: (Nov. 2019). URL: http://arxiv.org/abs/1912.03264.

[34] Yuanda Chen. *Thinning Algorithms for Simulating Point Processes*. Tech. rep. 2016.

[35] P A W Lewisr and G S Shedler. *SIMULATION OF NONHOMOGENEOUS POISSON PROCESSES BY THINNING*. Tech. rep.

[36] Warren S Mcculloch and Walter Pitts. *A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY* n*. Tech. rep. 2. 1990, pp. 99–115.

[37] Yann Lecun. *Convolutional Networks for Images, Speech, and Time-Series Oracle Performance for Visual Captioning View project Unsupervised Learning of Speech Representations View project*. Tech. rep. 1995. URL: https://www.researchgate.net/publication/216792820.

[38] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: (Dec. 2015). URL: http://arxiv.org/abs/1512.03385.

[39] Yue Wang et al. "Dynamic graph Cnn for learning on point clouds". In: *ACM Transactions on Graphics* 38.5 (Oct. 2019). ISSN: 15577368. DOI: 10.1145/3326362.

[40] Ashish Vaswani et al. "Attention Is All You Need". In: (June 2017). URL: http://arxiv.org/abs/1706.03762.

[41] Garrick Orchard et al. "Converting static image datasets to spiking neuromorphic datasets using saccades". In: *Frontiers in Neuroscience* 9.NOV (2015). ISSN: 1662453X. DOI: 10.3389/fnins.2015.00437.

[42] Yann LeCun and Corinna Cortes. "MNIST handwritten digit database". In: (2010). URL: http://yann.lecun.com/exdb/mnist/.

[43] Fei-Fei Li et al. *Caltech 101*. Apr. 2022. DOI: 10.22002/D1.20086.

[44] Charles R. Qi et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: (Dec. 2016). URL: http://arxiv.org/abs/1612.00593.

Appendix A. Network configurations

TABLE 6. Architecture overview of the PCT configuration

Layer (type)	Output Shape	Param #
Conv1d-1	[-1, 128, 1024]	2,688
BatchNorm1d-2	[-1, 128, 1024]	256
ReLU-3	[-1, 128, 1024]	0
Conv1d-4	[-1, 128, 1024]	114,688
BatchNorm1d-5	[-1, 128, 1024]	256
ReLU-6	[-1, 128, 1024]	0
Conv1d-7	[-1, 32, 1024]	4,096
Conv1d-8	[-1, 32, 1024]	4,096
Conv1d-9	[-1, 128, 1024]	16,512
Softmax-10	[-1, 1024, 1024]	0
Conv1d-11	[-1, 128, 1024]	16,512
BatchNorm1d-12	[-1, 128, 1024]	256
ReLU-13	[-1, 128, 1024]	0
SA_Layer-14	[-1, 128, 1024]	0
Conv1d-15	[-1, 32, 1024]	4,096
Conv1d-16	[-1, 32, 1024]	4,096
Conv1d-17	[-1, 128, 1024]	16,512
Softmax-18	[-1, 1024, 1024]	0
Conv1d-19	[-1, 128, 1024]	16,512
BatchNorm1d-20	[-1, 128, 1024]	256
ReLU-21	[-1, 128, 1024]	0
SA_Layer-22	[-1, 128, 1024]	0
Conv1d-23	[-1, 32, 1024]	4,096
Conv1d-24	[-1, 32, 1024]	4,096
Conv1d-25	[-1, 128, 1024]	16,512
Softmax-26	[-1, 1024, 1024]	0
Conv1d-27	[-1, 128, 1024]	16,512
BatchNorm1d-28	[-1, 128, 1024]	256
ReLU-29	[-1, 128, 1024]	0
SA_Layer-30	[-1, 128, 1024]	0
Conv1d-31	[-1, 32, 1024]	4,096
Conv1d-32	[-1, 32, 1024]	4,096
Conv1d-33	[-1, 128, 1024]	16,512
Softmax-34	[-1, 1024, 1024]	0
Conv1d-35	[-1, 128, 1024]	16,512
BatchNorm1d-36	[-1, 128, 1024]	256
ReLU-37	[-1, 128, 1024]	0
SA_Layer-38	[-1, 128, 1024]	0
Conv1d-39	[-1, 1024, 1024]	524,288
BatchNorm1d-40	[-1, 1024, 1024]	2,048
LeakyReLU-41	[-1, 1024, 1024]	0
Linear-42	[-1, 1024, 128]	131,200
Linear-43	[-1, 4096, 64]	2,112
Linear-44	[-1, 4096, 16]	1,040
Linear-45	[-1, 4096, 3]	51
Total params: 944,515		
Input size (MB): 0.01		
Forward/backward pass size (MB): 87.59		
Params size (MB): 3.60		
Estimated Total Size (MB): 91.21		

Overview the successive layers used in the PCT-based network for super-resolution of eventstreams at $r = 4$ and $N = 1024$.

TABLE 7. Architecture overview of the PointNet configuration

Layer (type)	Output Shape	Param #
Conv1d-1	[-1, 64, 1024]	960
BatchNorm1d-2	[-1, 64, 1024]	128
Conv1d-3	[-1, 64, 1024]	20,480
BatchNorm1d-4	[-1, 64, 1024]	128
Conv1d-5	[-1, 64, 1024]	20,480
BatchNorm1d-6	[-1, 64, 1024]	128
Conv1d-7	[-1, 128, 1024]	40,960
BatchNorm1d-8	[-1, 128, 1024]	256
Conv1d-9	[-1, 1024, 1024]	655,360
BatchNorm1d-10	[-1, 1024, 1024]	2,048
Linear-11	[-1, 1024, 128]	131,200
Linear-12	[-1, 4096, 64]	2,112
Linear-13	[-1, 4096, 16]	1,040
Linear-14	[-1, 4096, 3]	51
Total params: 875,331		
Input size (MB): 0.01		
Forward/backward pass size (MB): 24.59		
Params size (MB): 3.34		
Estimated Total Size (MB): 27.94		

Overview the successive layers used in the PointNet-based network for super-resolution of eventstreams at $r = 4$ and $N = 1024$.

TABLE 8. ARCHITECTURE OVERVIEW OF THE RESNET-18 CONFIGURATION

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 3, 512]	512
MaxPool2d-2	[-1, 64, 3, 256]	0
BatchNorm2d-3	[-1, 64, 3, 256]	128
ReLU-4	[-1, 64, 3, 256]	0
Conv2d-5	[-1, 64, 3, 256]	28,736
BatchNorm2d-6	[-1, 64, 3, 256]	128
Conv2d-7	[-1, 64, 3, 256]	20,544
BatchNorm2d-8	[-1, 64, 3, 256]	128
ResBlock-9	[-1, 64, 3, 256]	0
Conv2d-10	[-1, 64, 3, 256]	28,736
BatchNorm2d-11	[-1, 64, 3, 256]	128
Conv2d-12	[-1, 64, 3, 256]	20,544
BatchNorm2d-13	[-1, 64, 3, 256]	128
ResBlock-14	[-1, 64, 3, 256]	0
Conv2d-15	[-1, 128, 3, 128]	24,704
BatchNorm2d-16	[-1, 128, 3, 128]	256
Conv2d-17	[-1, 128, 3, 128]	57,472
BatchNorm2d-18	[-1, 128, 3, 128]	256
Conv2d-19	[-1, 128, 3, 128]	82,048
BatchNorm2d-20	[-1, 128, 3, 128]	256
ResBlock-21	[-1, 128, 3, 128]	0
Conv2d-22	[-1, 128, 3, 128]	114,816
BatchNorm2d-23	[-1, 128, 3, 128]	256
Conv2d-24	[-1, 128, 3, 128]	82,048
BatchNorm2d-25	[-1, 128, 3, 128]	256
ResBlock-26	[-1, 128, 3, 128]	0
Conv2d-27	[-1, 256, 3, 64]	98,560
BatchNorm2d-28	[-1, 256, 3, 64]	512
Conv2d-29	[-1, 256, 3, 64]	229,632
BatchNorm2d-30	[-1, 256, 3, 64]	512
Conv2d-31	[-1, 256, 3, 64]	327,936
BatchNorm2d-32	[-1, 256, 3, 64]	512
ResBlock-33	[-1, 256, 3, 64]	0
Conv2d-34	[-1, 256, 3, 64]	459,008
BatchNorm2d-35	[-1, 256, 3, 64]	512
Conv2d-36	[-1, 256, 3, 64]	327,936
BatchNorm2d-37	[-1, 256, 3, 64]	512
ResBlock-38	[-1, 256, 3, 64]	0
Conv2d-39	[-1, 512, 3, 32]	393,728
BatchNorm2d-40	[-1, 512, 3, 32]	1,024
Conv2d-41	[-1, 512, 3, 32]	918,016
BatchNorm2d-42	[-1, 512, 3, 32]	1,024
Conv2d-43	[-1, 512, 3, 32]	1,311,232
BatchNorm2d-44	[-1, 512, 3, 32]	1,024
ResBlock-45	[-1, 512, 3, 32]	0
Conv2d-46	[-1, 512, 3, 32]	1,835,520
BatchNorm2d-47	[-1, 512, 3, 32]	1,024
Conv2d-48	[-1, 512, 3, 32]	1,311,232
BatchNorm2d-49	[-1, 512, 3, 32]	1,024
ResBlock-50	[-1, 512, 3, 32]	0
Linear-51	[-1, 512, 3, 128]	4,224
Linear-52	[-1, 1024, 128]	24,704
Linear-53	[-1, 4096, 64]	2,112
Linear-54	[-1, 4096, 16]	1,040
Linear-55	[-1, 4096, 3]	51
Total params: 7,714,691		
Input size (MB): 0.01		
Forward/backward pass size (MB): 24.22		
Params size (MB): 29.43		
Estimated Total Size (MB): 53.66		

Overview the successive layers used in the ResNet-18-based network for super-resolution of eventstreams at $r = 4$ and $N = 1024$.

TABLE 9. ARCHITECTURE OVERVIEW OF THE RESNET-34 CONFIGURATION

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 3, 512]	512
MaxPool2d-2	[-1, 64, 3, 256]	0
BatchNorm2d-3	[-1, 64, 3, 256]	128
ReLU-4	[-1, 64, 3, 256]	0
Conv2d-5	[-1, 64, 3, 256]	28,736
BatchNorm2d-6	[-1, 64, 3, 256]	128
Conv2d-7	[-1, 64, 3, 256]	20,544
BatchNorm2d-8	[-1, 64, 3, 256]	128
ResBlock-9	[-1, 64, 3, 256]	0
Conv2d-10	[-1, 64, 3, 256]	28,736
BatchNorm2d-11	[-1, 64, 3, 256]	128
Conv2d-12	[-1, 64, 3, 256]	20,544
BatchNorm2d-13	[-1, 64, 3, 256]	128
ResBlock-14	[-1, 64, 3, 256]	0
Conv2d-15	[-1, 64, 3, 256]	28,736
BatchNorm2d-16	[-1, 64, 3, 256]	128
Conv2d-17	[-1, 64, 3, 256]	20,544
BatchNorm2d-18	[-1, 64, 3, 256]	128
ResBlock-19	[-1, 64, 3, 256]	0
Conv2d-20	[-1, 128, 3, 128]	24,704
BatchNorm2d-21	[-1, 128, 3, 128]	256
Conv2d-22	[-1, 128, 3, 128]	57,472
BatchNorm2d-23	[-1, 128, 3, 128]	256
Conv2d-24	[-1, 128, 3, 128]	82,048
BatchNorm2d-25	[-1, 128, 3, 128]	256
ResBlock-26	[-1, 128, 3, 128]	0
Conv2d-27	[-1, 128, 3, 128]	114,816
BatchNorm2d-28	[-1, 128, 3, 128]	256
Conv2d-29	[-1, 128, 3, 128]	82,048
BatchNorm2d-30	[-1, 128, 3, 128]	256
ResBlock-31	[-1, 128, 3, 128]	0
Conv2d-32	[-1, 128, 3, 128]	114,816
BatchNorm2d-33	[-1, 128, 3, 128]	256
Conv2d-34	[-1, 128, 3, 128]	82,048
BatchNorm2d-35	[-1, 128, 3, 128]	256
ResBlock-36	[-1, 128, 3, 128]	0
Conv2d-37	[-1, 128, 3, 128]	114,816
BatchNorm2d-38	[-1, 128, 3, 128]	256
Conv2d-39	[-1, 128, 3, 128]	82,048
BatchNorm2d-40	[-1, 128, 3, 128]	256
ResBlock-41	[-1, 128, 3, 128]	0
Conv2d-42	[-1, 256, 3, 64]	98,560
BatchNorm2d-43	[-1, 256, 3, 64]	512
Conv2d-44	[-1, 256, 3, 64]	229,632
BatchNorm2d-45	[-1, 256, 3, 64]	512
Conv2d-46	[-1, 256, 3, 64]	327,936
BatchNorm2d-47	[-1, 256, 3, 64]	512
ResBlock-48	[-1, 256, 3, 64]	0
Conv2d-49	[-1, 256, 3, 64]	459,008
BatchNorm2d-50	[-1, 256, 3, 64]	512
Conv2d-51	[-1, 256, 3, 64]	327,936
BatchNorm2d-52	[-1, 256, 3, 64]	512
ResBlock-53	[-1, 256, 3, 64]	0
Conv2d-54	[-1, 256, 3, 64]	459,008
BatchNorm2d-55	[-1, 256, 3, 64]	512
Conv2d-56	[-1, 256, 3, 64]	327,936
BatchNorm2d-57	[-1, 256, 3, 64]	512
ResBlock-58	[-1, 256, 3, 64]	0
ResBlock-58	[-1, 256, 3, 64]	0
Conv2d-59	[-1, 256, 3, 64]	459,008
BatchNorm2d-60	[-1, 256, 3, 64]	512
Conv2d-61	[-1, 256, 3, 64]	327,936
BatchNorm2d-62	[-1, 256, 3, 64]	512
ResBlock-63	[-1, 256, 3, 64]	0
Conv2d-64	[-1, 256, 3, 64]	459,008
BatchNorm2d-65	[-1, 256, 3, 64]	512
Conv2d-66	[-1, 256, 3, 64]	327,936
BatchNorm2d-67	[-1, 256, 3, 64]	512
ResBlock-68	[-1, 256, 3, 64]	0
Conv2d-69	[-1, 256, 3, 64]	459,008
BatchNorm2d-70	[-1, 256, 3, 64]	512
Conv2d-71	[-1, 256, 3, 64]	327,936
BatchNorm2d-72	[-1, 256, 3, 64]	512
ResBlock-73	[-1, 256, 3, 64]	0
Conv2d-74	[-1, 512, 3, 32]	393,728
BatchNorm2d-75	[-1, 512, 3, 32]	1,024
Conv2d-76	[-1, 512, 3, 32]	918,016
BatchNorm2d-77	[-1, 512, 3, 32]	1,024
Conv2d-78	[-1, 512, 3, 32]	1,311,232
BatchNorm2d-79	[-1, 512, 3, 32]	1,024
ResBlock-80	[-1, 512, 3, 32]	0
Conv2d-81	[-1, 512, 3, 32]	1,835,520
BatchNorm2d-82	[-1, 512, 3, 32]	1,024
Conv2d-83	[-1, 512, 3, 32]	1,311,232
BatchNorm2d-84	[-1, 512, 3, 32]	1,024
ResBlock-85	[-1, 512, 3, 32]	0
Conv2d-86	[-1, 512, 3, 32]	1,835,520
BatchNorm2d-87	[-1, 512, 3, 32]	1,024
Conv2d-88	[-1, 512, 3, 32]	1,311,232
BatchNorm2d-89	[-1, 512, 3, 32]	1,024
ResBlock-90	[-1, 512, 3, 32]	0
Linear-91	[-1, 512, 3, 128]	4,224
Linear-92	[-1, 1024, 128]	24,704
Linear-93	[-1, 4096, 64]	2,112
Linear-94	[-1, 4096, 16]	1,040
Linear-95	[-1, 4096, 3]	51
Total params: 14,459,651		
Input size (MB): 0.01		
Forward/backward pass size (MB): 39.22		
Params size (MB): 55.16		
Estimated Total Size (MB): 94.39		

Overview the successive layers used in the ResNet-34-based network for super-resolution of eventstreams at $r = 4$ and $N = 1024$.

Appendix B.
Figures

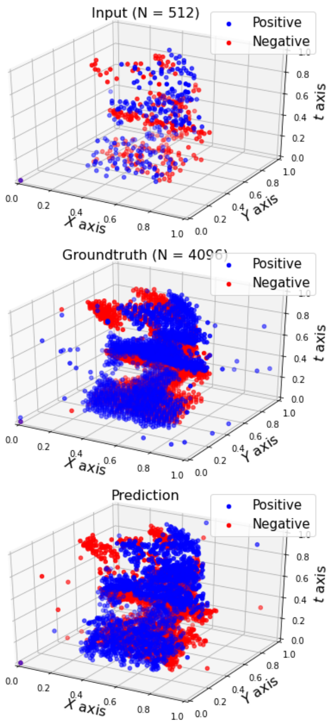


Figure 16. Prediction of PCT depicted in event-space.

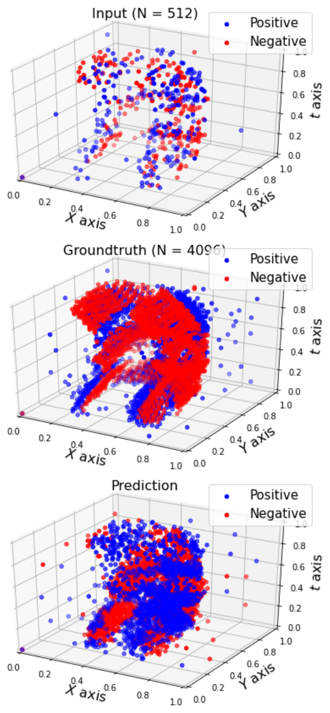


Figure 17. Prediction of ResNet-34 depicted in event-space.

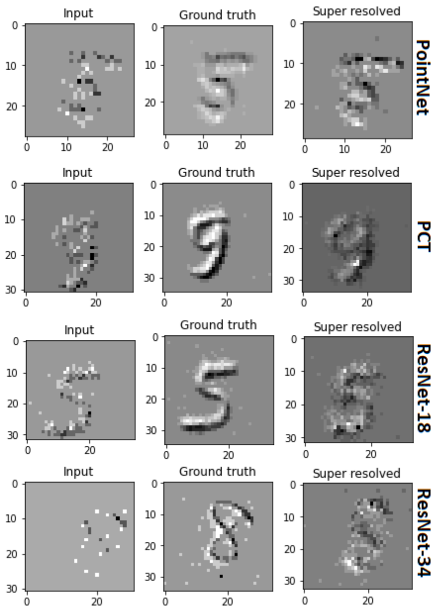


Figure 18. Converted event-frames at $t_{bin} = 10^{-1}$ from super-resolution predictions.

Figure 19. Super-resolution predictions made on data from N-MNIST with $r = 8, N = 512$ using networks utilizing a PCT, PointNet, ResNet-18 or ResNet-34 basedFeature extractor.

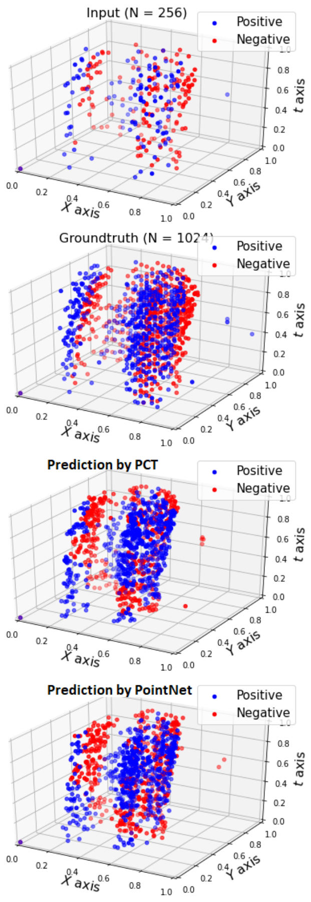


Figure 20. Prediction of PCT and PointNet depicted in event-space.

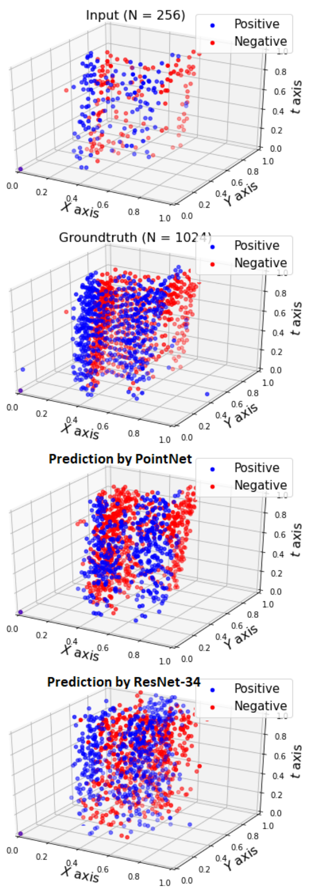


Figure 21. Prediction of PointNet and ResNet-34 depicted in event-space.

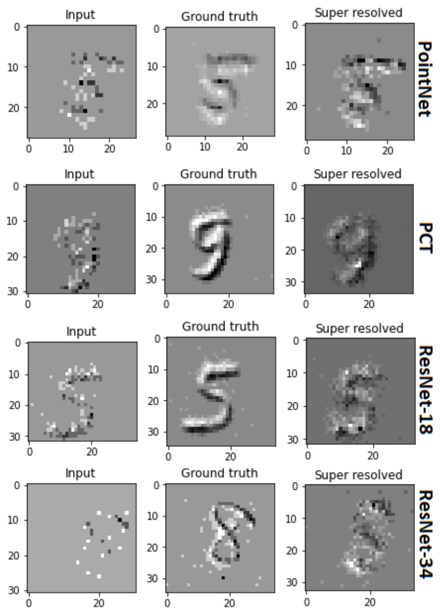


Figure 22. Converted event-frames at $t_{bin} = 2.5 \times 10^{-4}s$ from super-resolution predictions.

Figure 23. Predictions made on data from N-MNIST with $r = 8, N = 256$ using networks utilizing a PCT, PointNet, ResNet-18 or ResNet-34 basedFeature extractor.

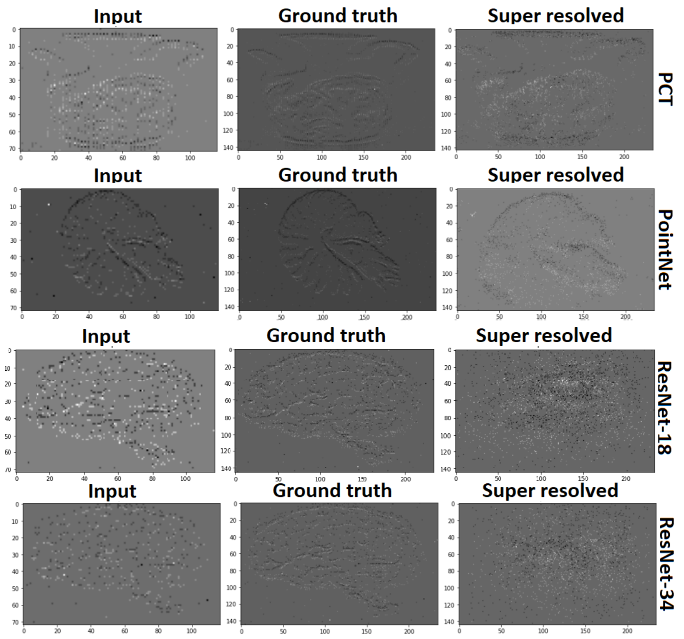


Figure 24. Event-frames converted at $t_{bin} = 10^{-3}s$ from super-resolution predictions on data from N-Caltech-101 with with $r = 4, N = 3072$ subsampled by TDS using networks utilizing a PCT, PointNet, ResNet-18 and ResNet-34 based Feature extractor.

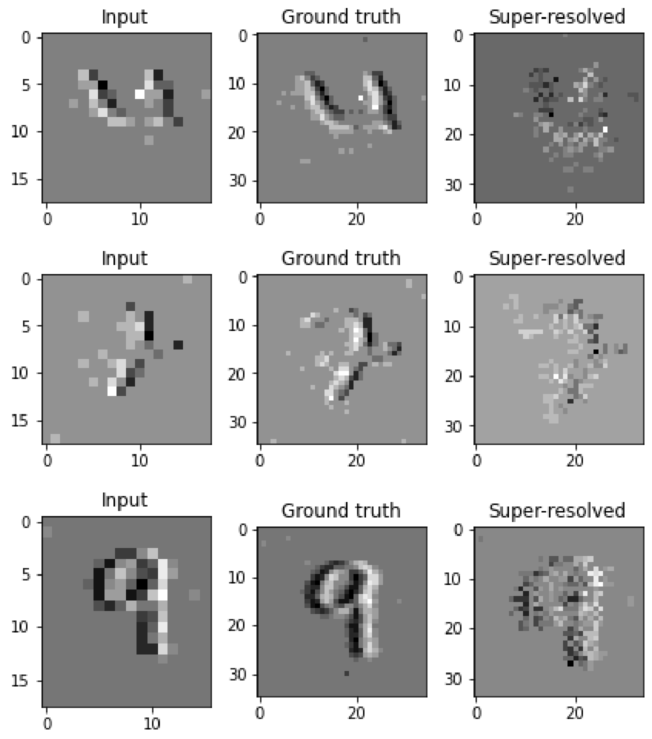


Figure 25. Event-frames converted at $t_{bin} = 10^{-3}s$ from super-resolution predictions on data from N-Caltech-101 with with $r = 4, N = 1024$ subsampled by TDS using PointNet-based Feature extractor. Data shows the appearance of a checkerboard pattern on the predictions.

Appendix C. Comparing PCT to PointNet Feature extractors

In Experiment 3, the PCT-based module performed comparably to the PointNet-based module when used as a Feature extractor in the researched network. Yet the PN-Net resulted in a far lower computation time ranging between factor 4 and 10 depending on the length of the event sequence. Also, it was noticed that the PN-Net is prone to visualize a checkerboard pattern in its predictions. One-one comparison of the obtained converted event-frames from Experiment 3 are depicted in Figure 30 in which these checkerboard patterns are observable in the prediction column made by PN-Net. This section will go into detail about the differences between both networks to find explanations of the occurred behavior in order to make a structured verdict as to which network will be elaborated further on. Both PCT and PointNet are developed for the classification and segmentation of 3D pointclouds, and the example given in the paper [27] is depicted in Figure 28. Although the purpose of both networks differs from super-resolution it processes 3D point cloud data and has similarities to events. This set the interest in this research to experiment with the Feature extractor parts of these networks for super-resolution purposes of events.

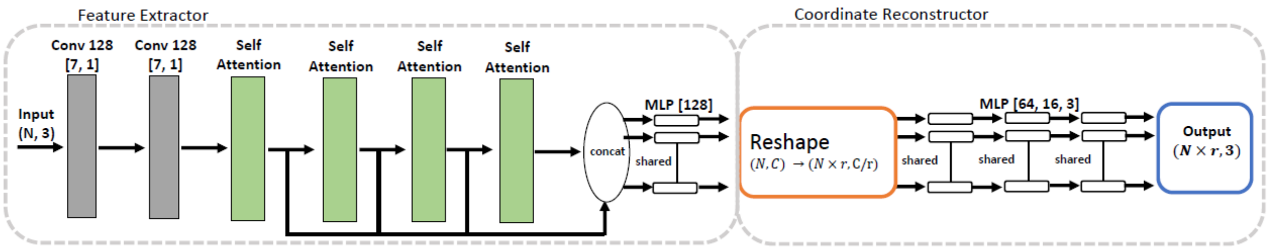


Figure 26. PCT-based network for super-resolution

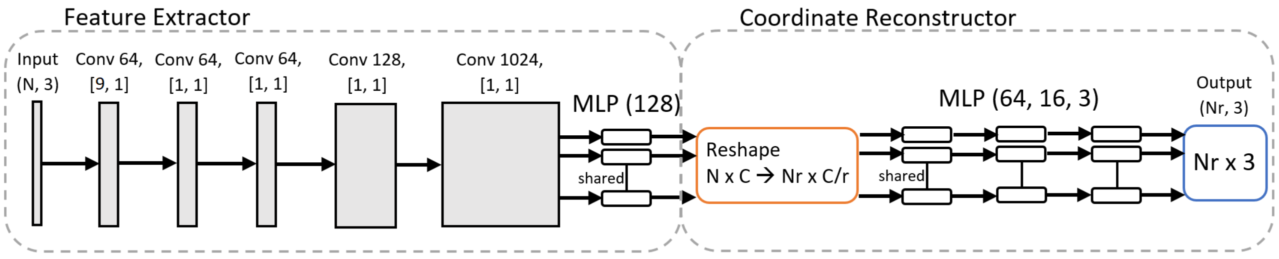


Figure 27. PointNet-based network for super-resolution

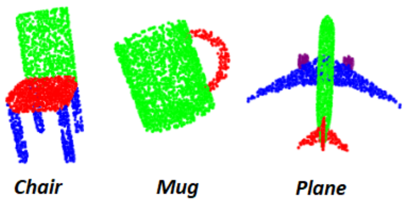


Figure 28. Classification and segmentation of pointclouds by PCT source: [27]

C.1. In-dept PointNet Feature extractor

PN-Net is based on the Feature extractor sub-module inside PointNet [44] named T-Net. It uses five convolutional from which the first layer has a kernel size of $k = 9$ while the rest of $k = 1$ is followed by a channel-wise MLP layer. The channel-wise MLP-layers isolate data in each neuron and avoid data to flow through parallel neurons. From this point of view, it is questionable if this type of network filters semantics into its feature extraction. This suspicion is based on the fact that its limited receptive[6] field of 9. Also, convolutional layers are designed to process data with a grid-like topology like images or a 1D sequence. Eventstreams are tuple sequences consisting of 3 euclidean dimensions but can only be ordered in one dimension. Meaning, the first convolutional layer can filter data from 8 neighbors (9 minus itself) from a time (t) perspective, but these neighbors do not necessarily entail the most useful information needed to super-resolve. We argue that this implementation of the PN Feature extractor can not incorporate semantics into its extraction but uses a simplistic policy in which it predicts super resolved points around the coordinates of the given inputs which results in the perceived checkerboard effect. We proved this argument by experimenting with how the network would predict if it only has kernels of size $k = 1$ bringing its receptive field equally to 1, it resulted in almost identical predictions although being even more prone to this checkerboard effect. Afterward, we experimented by initiating kernel sizes of $k = 5$ in each convolution layer to enhance the network's understanding of semantics. This configuration gives the network a receptive field of the whole input length ($(k-1)^5 = 1024$), yet this configuration performed worse compared to the previous one. This can be due to the kernels having fixed values that are designed for grid-like data which the characteristics of the data structure of eventstreams do not feature, this insight indicates a design flaw in our implementation. This raises questions as to how the use of this Feature extractor scored the highest accuracy in Experiment 2, how its policy to predict coordinates around the coordinates of the given inputs was not detected before, why it is able to extract features when implemented in PointNet? As for the first question, we have to take a look at the architecture that was used for the classification task which is depicted in Figure 29. From this overview it can be said that different from the super-resolution implementation, the MLP layers are not channel-wise operated and thereby provide the network the ability to obtain semantic awareness.

As for the second question, sub-sampling technique at random was used to obtain the low-resolution eventstream (input) from the high-resolution eventstream (ground truth) in the initial experiments. This sub-sample technique enables unlimited scaling ratio possibilities since it was not restricted by spatial address as the later used true downsampling (TDS) technique 2.4.4. Unfortunately, sub-sample at random 'camouflaged' the checkerboard pattern since the predictions were likewise randomly distributed which is the reason why it was only detectable in the final part of Experiment 3. A second reason is an apparent loophole in the calculation in the CD loss function which left occurrences of predicted events on the same coordinate unpunished. This loophole 'allowed' the network to converge to this simplistic policy and is undetected by the quantification of the loss function.

As for the latter question, this Feature extractor is issued as a residual connection in a vast network. The semantics are obtained outside this Feature extractor and thus are not obstructive in that sense.

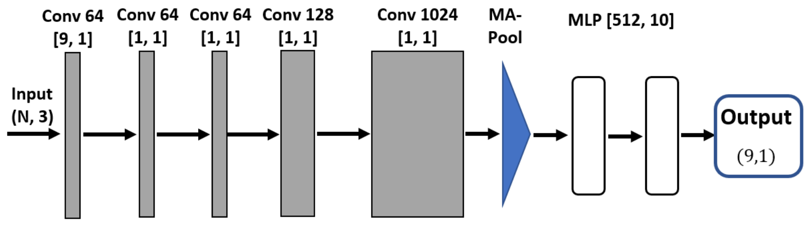


Figure 29. Architecture for classification of N-MNIST using a PointNet based Feature extractor

C.2. In-dept PCT Feature extractor

The PCT Feature extractor in our network is an exact copy of the one proposed in [27] and uses it as a core-component self-attention layers. These self-attention layers, contrary to convolutional layers, allow to selectively focus on the most relevant information given in a sequence of each event to each other event. This cross-relevance is referred to as dependencies between data points. This allows self-attention mechanisms to capture dependencies in sequential data. We see an alignment in need when we project the ability of selective focus onto the characteristics of events and the process of super-resolution. Namely, in order to super-resolve, the network must have an understanding of the spatial and temporal dependencies between each event. These dependencies are not grid-like structures like an image but depend on deeper data structures and semantics. The downside of self-attention is its time complexity being $O(n^2)$ per layer which is inherently caused by the calculation of dependencies by the self-attention mechanism.

6. The receptive field of a convolutional neural network is the region of the input data that is processed by a given neuron in the network. It is determined by the size and stride of the convolutional filters that are applied to the input data, as well as the number of layers in the network.

C.3. Verdict

In conclusion, the PointNet-based Feature extractor as implemented in our network does not provide the ability to extract sufficient features for super-resolution. The reason for this originates in the false implementation in our network in which it is used as a primary module to extract all features while the original PointNet implementation is used to process data in a residual connection. The PCT-based Feature extractor seems to be a perfect fit to extract features by dependencies between each event. Although the self-attention results in a time complexity of $O(n^2)$, in future research there can be search methods to work around this single deviancy.

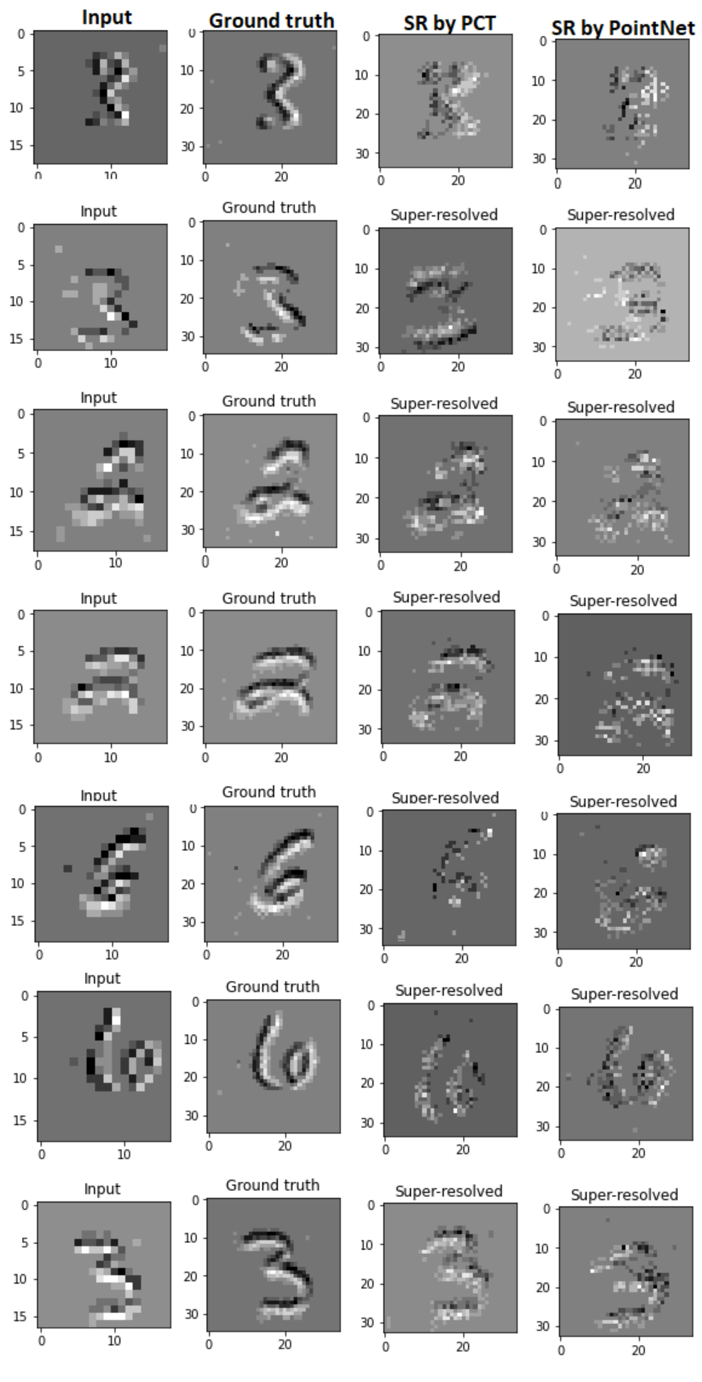


Figure 30. One-on-one comparison of converted event-frames from super-resolution prediction made by networks using a PCT and PointNet Feature extractor

Appendix D.
Hyperparameter Tuning

Various experiments have been conducted to compare various types of networks from which the network which utilizes the Feature extractor as used in PCT performed the best. Up until now, experiments were conducted with the initial architectural configuration used in the PCT network. This configuration is referred to as vanilla. To enhance performance right set of hyperparameters has to be found. The search for this set is referred to as hyperparameter tuning and covers fine-tuning all non-learnable parameters in order to improve its performance according to the loss function. We are bound to make a selection of hyperparameters to be considered due to the time constraints of this research. Among the selection of hyperparameters are:

1) Number of MLP layers
2) Number of self-attention layers
3) Number of neurons in the MLP layers
4) Regularization methods
5) Data augmentation

The resulted network will be fully trained after the hyperparameter tuning process.

D.1. Number of MLP layers

Having more layers leverages the network's ability to extract hidden features. But it also makes the network more complex and has a larger amount of parameters to train which causes the network to converge slower. On the other hand, you can remove layers which makes the network less complex and thus faster converging but possibly, not able to extract the right hidden features which results in an overfitting network. There is an optimum for a network between converging at a reasonable pace and the value of the loss it is converging to. In this experiment, we define two extra configurations named Shallow Coordinate Reconstructor (SCR) and Deep Coordinate Reconstructor (DCR). The SCR configuration has two MLP layers in the coordinate reconstructor module with channels [64, 3]. The DCR configuration has 5 MLP layers in the coordinate reconstructor module with channels [64, 64, 64, 16 3].

The resulting learning curves of these configurations together with the vanilla network are depicted in Figure 31. From this result, it can be concluded that the DCR is much slower converging with respect to the other configurations and is also prone to an unstable learning course. Also, the SCR is converging in a similar way as the vanilla configuration although being more prone to volatility. These results suggest it is best to keep the amount of MLP layers in the coordinate reconstructor to 3.

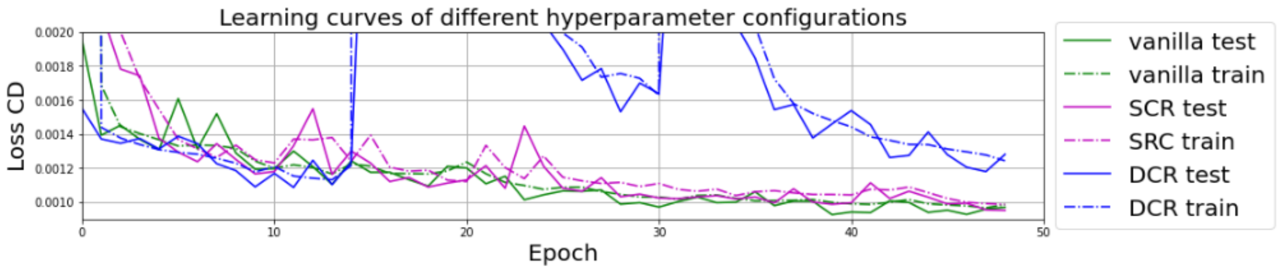


Figure 31. Number of MLP layers configurations

D.2. Number of self-attention layers

Increased number of self-attention (SA) layers in the network result in the increased ability of the network to capture more complex features in the data but also the computational complexity. Finding the right balance between the number of SA layers and the ability to extract appropriate features is key especially since the SA layers form the bottleneck in the network. The configurations that are tested are Shallow Self-Attention (SSA) with 2 SA layers and Deep Self-Attention

(DSA) with 6 SA layers.

The resulting learning curves of these configurations together with the vanilla network are depicted in Figure 32. Both SSA and DSA have a peak around the 10th epoch after which SSA converges to the same value as the vanilla architecture and DSA converges to a constant higher but decreasing value. Plausible does the DSA configuration converges to the same value or even lower when trained for more epochs. But since SSA converges to almost the same value but over a more volatile course, we conclude that SSA is probably too shallow and is less capable to extract the required deeper features. From these observations, we continue with the vanilla architecture since it seems to have the right balance between extracting the required deep features while preventing it from becoming overly computationally complex.

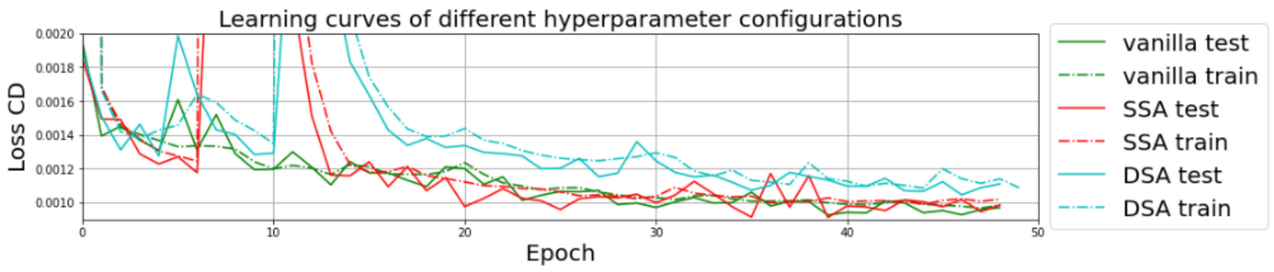


Figure 32. Number of SA layers configurations

D.3. Number of neurons in the MLP layers

Increasing the number of channels will result in a higher ability to reconstruct coordinates since there it entails more features for reconstructing the coordinates. We test a Wide Coordinate Reconstructor (WCR) configuration with MLP(128, 64, 3) and the vanilla configuration with MLP(64, 16, 3). Results are depicted in Figure 33. From this result, it can be concluded that the WCR configuration does enhance the loss value with respect to the vanilla architecture. From this, we conclude that the vanilla channel configuration in the coordinate reconstruction is near perfect.

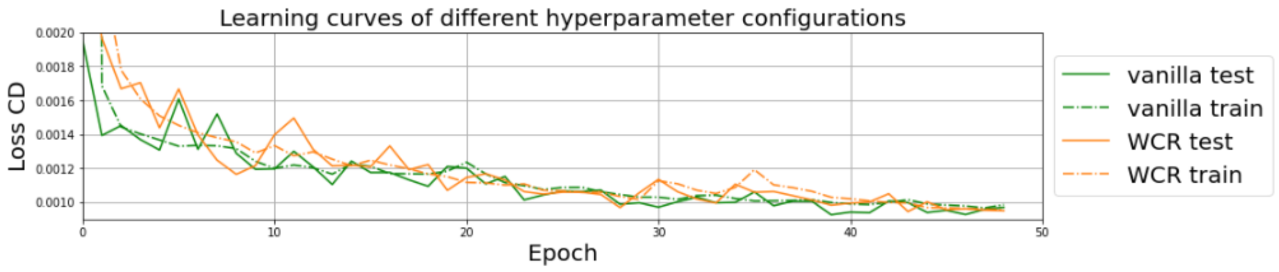


Figure 33. Number of neurons in the MLP layers of the coordinate reconstructor configurations

D.4. Regularization methods

Regularization methods improve the networks to generalize their performance to unseen data. Therefore it is best used in cases in which there is a gap between the test and train learning curves. Looking at the learning curve we can conclude with confidence that there is no gap between the train and test curves. Unless there appears a gap between these curves during the final training, regularization methods would be obsolete for research.

D.5. Data augmentation

Data was augmented by rotating the eventstreams spatially by swapping the x and y dimensions and also with downsampling with respect to different spatial addresses. Theoretically, there are 4 different spatial addresses to respectively can be downsampled from. We only used two addresses to prevent memory overloading of the hardware of Google Colab. Together with the spatially rotated augmentation resulted in a four times larger dataset.

D.6. Final network configuration and training of the network

The resulting network after the hyperparameter tuning is equal to the vanilla architecture. There are still some optional hyperparameters left to be tuned which makes an interesting research for further research. The obtained network with learned parameters from the initial 50 epochs long training loop on the normal dataset will be trained further on the augmented dataset. The result is depicted in Figure 34. After the first 50 epochs, the network was trained on the augmented dataset in combination with a lowered learning rate of 0.0035 and show a large drop in loss from $9.5e-4$ to converging at $8.5e-4$. From epoch 100 and onward we trained the network with a learning rate of 0.002, it shows another drop in the loss function after which it converges to a value of between $7.8e-4$ and $8e-4$ eventually. Remarkably, compared to the train learning curve, the test learning curve is consequently higher which indicates that the network can predict the test dataset better although it is trained on the training dataset. This also means that the network does not need any addition of regularization methods. In the end, the network converges towards a loss of $7.79e-4$, the final obtained network is depicted in Figure 35.

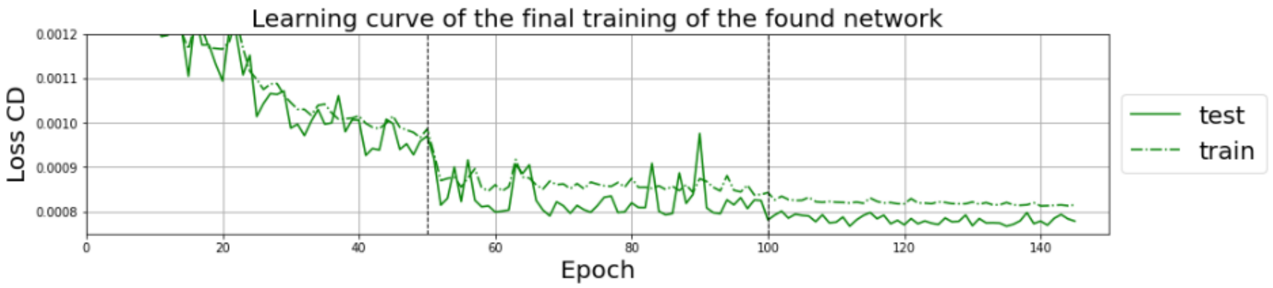


Figure 34. Learning curve of the final training, black dashed line indicates a change of set learning rate [0.005, 0.0035, 0.002] at epoch number [0, 50, 100]

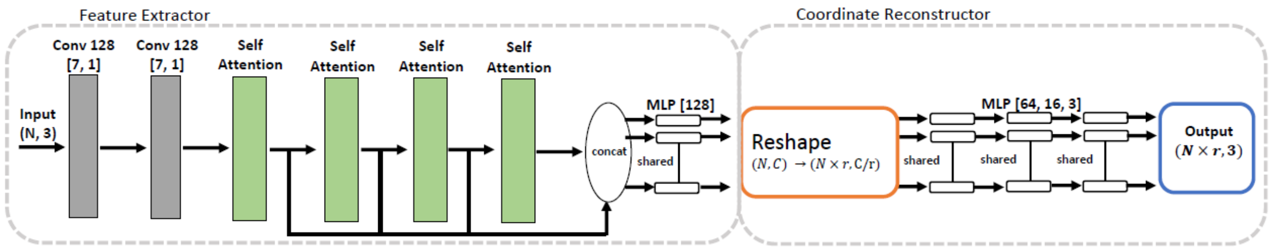


Figure 35. Overview of PCT-based network configuration

Appendix E.
Ablation study

Insight into the learned policy used by a trained network enables explaining perceived behavior. Unfortunately, a well-known caveat of neural networks is the inability to acquire this insight. Obtaining insight is complicated by the paradigm of neural networks in which a network as a whole is able to apply a policy, but each parameter on its own does not. This section attempts to obtain insight by investigating the self-attention mechanisms by the means of an ablation study. The final network consists of 4 consecutive SA-layers as depicted in Figure **??**. As mentioned before, the SA-layers calculate dependencies between events, this results in a SA map (A) of the size (N, N) with N being the sequence length. Value $a_{i,j}$ symbolizes the dependency between event e_i and e_j, the higher the value the higher the dependency. It is expected that there is a correlation between both spatial and temporal proximity. With in mind the fact that eventstreams are ordered in the temporal dimension, the expectation would result in a diagonal region in A with relatively higher values. Yet, visualized A of each SA-layer in Figure 36 do not align with our expectations as there is no observable diagonal region with increased interest which would take the form of a brighter diagonal region compared to the rest of the matrix. Instead, these self-attention maps show horizontal and vertical lines, which suggests that certain events are valued as important in principle independent of their relationship to other events.

There is, however, a possibility that the expected diagonal region does not appear in A due to the existence of multiple SA-layers and that there could be such a diagonal region hidden in a higher dimension. As a sanity check of the hypothesis of attention to be located in the diagonal region, a simpler configuration with a singular SA-layer is trained and tested. The obtained SA-maps from a network configuration containing a singular SA-layer are depicted in Figure 37. From these SA-maps we have to conclude that the expected correlation between both spatial and temporal proximity is either false or the focus on this relationship is established outside of the SA-mechanism.

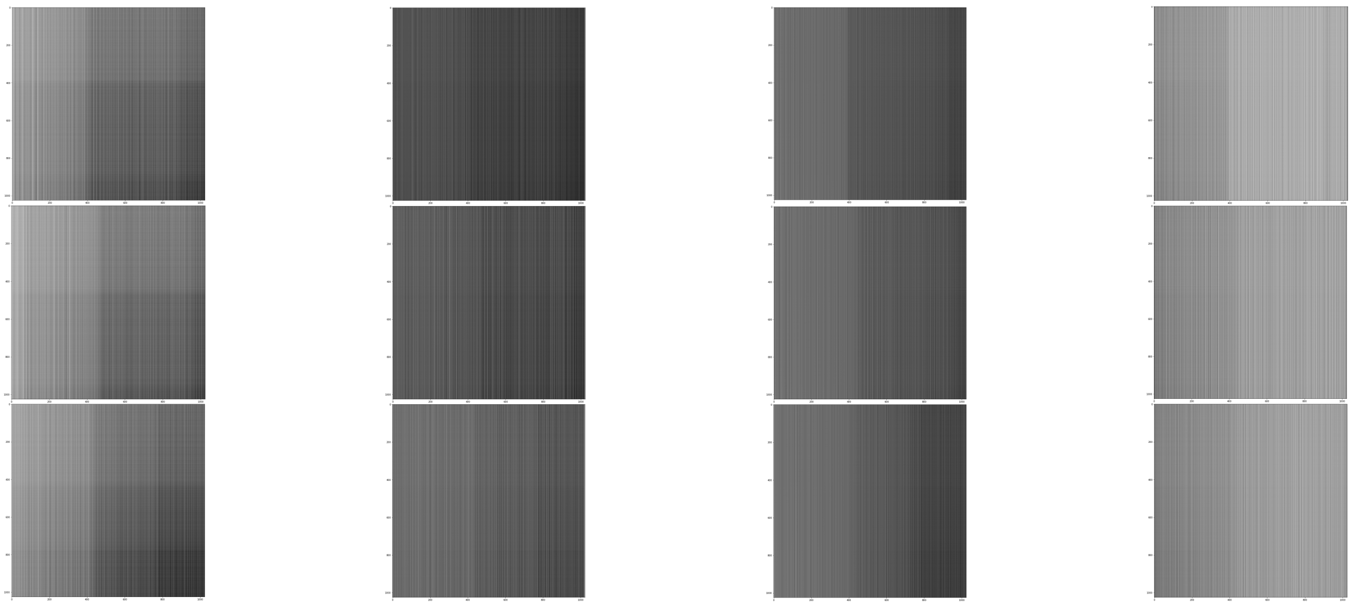


Figure 36. Visualization of SA-maps, each row shows from left to right the SA-maps of the successive layers processed by from the same input. Each row depicts results obtained of different inputs.

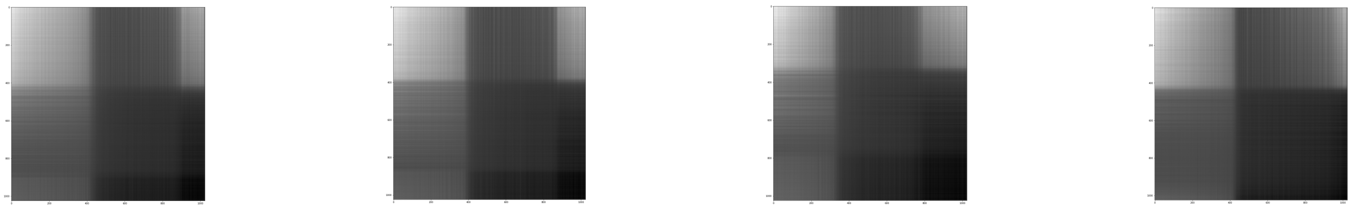


Figure 37. Visualization of four different SA-maps, obtained from a train network containing a singular SA-layer on four different datainstances.

Appendix F.
Literature study

Literature Review

Optimizing Event-Based Vision
by Realizing Super-Resolution in Event-Space

RO57010-20: RO Literature Research

Mahir Sabanoglu

Delft University of Technology

Literature Review

Optimizing Event-Based Vision by Realizing Super-Resolution in Event-Space

by

Mahir Sabanoglu

Student:	MM Sabanoglu
Student Number:	4494172
Supervisor:	Dr.ir. JCF (Joost) de Winter
Daily Supervisor:	N. Tömen
Buisuness Supervisor:	E.M. Souman
Institution:	Delft University of Technology
Place:	Faculty of Mechanical, Maritime and Materials Engineering, Delft
Project Duration:	Februari, 2022 - April, 2022

Cover Image: 3D Phase portrait visualization from https://wallpaperbat.com/data-science-wallpapers

Summary

The human brain is capable of performing extremely complex computations while it only has a power consumption as low as 20 Watts [29], this makes the brain to form a major source of inspiration for designing a computer that shares the same specifications. This research topic is named neuromorphic computing and a subsection, named neuromorphic vision, is occupied with mimicking the way humans perceive vision. The retina is the part of the human eye that translates vision to signals that the human brain can process. The vision sensor in neuromorphic vision consists of pixels that trigger an event $e(t, x, y, p)$ (neuromorphic vision is often referred to as event-based vision contrary to the frame-based vision of conventional video cameras) timestamp t, at pixel location (x, y) with polarity $p\{-1, +1\}$ whenever the observed brightness surpasses a threshold. This is done asynchronously for each pixel just like the photosensitive cells in the retina. This paradigm results in some interesting properties that make it besides neuromorphic computing also interesting to be implemented directly on computer vision tasks. Compared to frame-based vision, these properties of neuromorphic cameras include a greater dynamic range (140 dB versus 60 dB), no motion-blur, high temporal resolution as it can observe at a frequency of $1 \sim 10$ MHz, a one-hundreds times lower power consumption, and a high data sparsity in which redundant data is omitted. However, the main limitation of neuromorphic cameras is their resolution of 128×128 pixels which is quite low.

Super-resolution is a computer vision problem in which it is tried to predict a higher resolution counterpart of a lower resolution input. Various research is done on the super-resolution of frame-based videos and images. But found solutions for frame-based super resolution are not directly interchangeable for event-based vision since data representation of frame-based vision differs from neuromorphic vision. There are some proposed models that claim to super resolve event-based vision, but all of these models inflict a conversion from event representation -also referred to as event-space- to frames. This conversion is a discretization in which (to some extent) temporal information is lost while extra data is introduced to fill in pixels in which no events have occurred. This counters the sparse data representation which is needed for power-efficient applications. In this literature research possibilities are exploited to realize super-resolution on event-based vision in event-space. For a complete overview of the problem, state-of-the-art super-resolution models in frame-based vision, event-based vision, and pointclouds (which share similar data representation) are reviewed. The reviewed models are then used as inspiration to find options to incorporate temporal information while keeping the data representation sparse.

Reviewed state-of-the-art image super-resolution algorithms make use of convolutional neural networks. These types of networks are commonly used in computer vision tasks as they have beneficial properties for processing image representations in the form of an array. The same type of network is also used to realize super-resolution in conventional videos, but with the addition that consecutive frames are also used as they embed some extra information about the super-resolution. When looking at current state-of-the-art super-resolution models for the event-based vision, it can be concluded that these models are inspired by the frame-based vision. Events are discretized into a frame-based vision-like representation to make them fit those algorithms. Doing so results in the following cons: (i) temporal information is lost in the discretization, and (ii) the transformation from events to frames counters the sparsity of the data which is one of the key properties of event-based vision.
To exploit the possibilities to incorporate temporal information while keeping the data representation sparse, we reviewed papers about implementations of spiking neural networks. This is a special class of neural networks that is optimized to process data representations like events. Nevertheless, this class of network is not far enough developed to be used in more complex computer vision tasks -like super-resolution-. This is due to both lacks of performance on even simple computer vision tasks as well as the lack of available mature toolboxes to implement spiking neural networks.
Except for polarity information, pointclouds share the same data representation compared to events from a euclidean perspective. There are different methods reviewed regarding how pointcloud upsample models cope with the data representation. From the reviewed methods, Graph convolution network-

based algorithms seem to be the most applicable class of networks to realize super-resolution on events. This type of network is used in models to super-resolve noisy and sparse pointclouds. Yet, this type of network is not used to realize super-resolution on events.

We see a promising option to realize super-resolution of events without the loose of temporal information by the use of a graph convolutional neural network. This raises the following research question: **To what extent can event-based vision be super-resolved in event-space?**

With sub-questions:

1. To what extent can naive algorithm be used to super-resolve events or should a learning-based algorithm be used instead?
2. To what extent is it power and computational efficient to realize super resolution in event-space instead of super-revolve event-frames?
3. Do events converted to frame-based video result in a lower mean squared error with respect to a ground truth when the events are super-resolved in eventspace rather then after the conversion?

During the thesis, research will be done to find answers to the research questions. The model that is capable to realize super-resolution directly on event representation will be evaluated in threefold. This threefold include evaluation (i) in event space by a combination of a pointcloud similarity metric like Chamfer distance or Hausdorff distance and a precision metric for the binary classification of the polarities. (ii) Computation time will be compared with other models. (iii) frame-based conversion will be compared by performing super-resolution in event space and then converting events to frames using E2VID to review models that first convert to event frames and then super-resolve.

Contents

List of Figures

List of Tables

1 Introduction

The human brain is capable of performing extremely complex computations while it only has a power consumption as low as 20 Watts [29], this makes the brain to form a major source of inspiration for designing a computer that shares the same specifications. Neuromorphic computing describes this field of study that tries to mimic the physics of the human brain. The first contribution of neuromorphic computing was made in 1940 [17] in which a simple artificial neuron model was created that was able to make computations which we would call an artificial neural network (ANN) to date. During this time span, huge progress has been made and we are now able to design deep neural networks (DNN) -which are ANNs but then consist of multiple layers of neurons- with a wide diversity of different architectures and properties. In combination with the large computation power we possess and the option to do parallel computations Graphical Process Unit (GPU) and Tensor Processing Unit (TPU), algorithms can be trained that sometimes even out-compete the human brain especially when the data have abstract correlation or when there is a massive amount of data that needs to be calculated. Although performances with respect to the calculated results are perfected, the power efficiency is downright abysmal. For example, it can cost up to 650 MWh to train a single state-of-the-art DNN which is a carbon dioxide emission equivalent to the lifespan of 5 cars [31].

So, despite the fact that we are able to mimic the physics of the brain, the power efficiency of state-of-the-art DNNs are not comparable with the human brain. Although DNNs are topological inspired by the human brain, the data representation differs in paradigm which is one of the reasons why it is not as energy efficient. One way to obtain data with a representation more close related to its biological counterpart is by mimicking how it is instantiated. Neuromorphic vision -often referred to as event-based vision- is a subsection of neuromorphic computing in which the way humans perceive vision is mimicked.

Research towards neuromorphic vision has come to a point at which it is possible to build vision sensors that work with the same paradigm as the retina in the human eye. This type of sensor captures vision in bio-inspired event-space, which has some interesting properties which will be deliberated on in the literature review. However, currently, the main limitation of neuromorphic vision is its low resolution of 128×128 pixels. Finding higher resolution representatives of low-resolution input is called super-resolution (or super resolving) and could overcome this shortcoming in event-based vision. There are some proposed models that claim to super resolve event-based vision, but all of these models inflict a conversion from event-space to frames and thus counter the data representation needed for power-efficient applications.

Therefore, the aim of this literature review is to find an answer to the following research question: **Is it possible to super resolve event-based vision in event-space?**

With the following sub-questions to exploit all possibilities:

1. How does event-based vision differ from frame-based vision and why should we bother?
2. To what extent is super-resolution acquired on conventional frame-based vision?
3. Which different techniques are proposed to realize super-resolution on frame-based vision and how can this be used for event-based vision?

4. Which algorithms are proposed to realize super-resolution on event-based vision to date, and how does it cope with the data representation?
5. Are there data representations that are comparable with events, and are there proposed algorithms to realize super-resolution on this data?

2 Literature Review

2.1. Neuromorphic vision

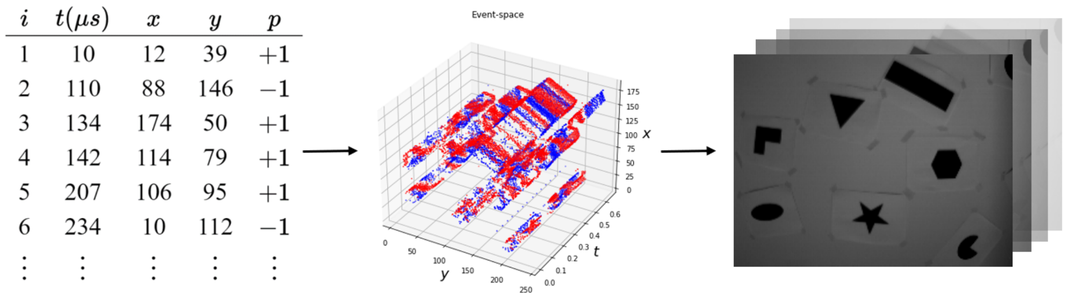
The way human eyes process visual input is different from how a conventional video camera works. The retina is a part of the eye that makes it possible to translate observed vision into signals that are both transmittable through the nervous system and interpretable by our brains. This is due to the photosensitive cells named photoreceptors -which are located in the retina-, it can be seen as the counterpart of an independent pixel in a photosensitive sensor of a camera. Each of the photoreceptors transmits signals to the brain whenever a brightness change is observed that surpasses a certain relative threshold which is done independent of other photoreceptors and continuous in time. This results in the fact that eyes do not have a shutter time nor a frame rate at which vision is processed. Additionally, the signal does not even include absolute brightness values but so-called spike trains and patterns that embed observed brightness changes. This data representation results in a highly compressed form of vision when compared with conventional video cameras in which absolute brightness is processed at a fixed frame rate regardless of occurring scenery [25]. Due to photoreceptors operating in an asynchronous way, the exposure time is inherently asynchronous which prevents motion blur in the observation whenever a projection on the retina is moving with a high velocity. In addition, the brightness threshold that is used to trigger signals to the brain is scaled logarithmically which inflicts the advantage that eyes have a high dynamic range (140 dB versus 60 dB). This translates in practice to enhanced performance in distinguishing bright and darker objects in the same scene.

A lot of research is performed towards the realization of neuromorphic cameras which make use of a similar paradigm as the way retinas process vision. This research area is named neuromorphic vision and the type of sensor is often referred to as retinamorphic. In 1980 the first retinamorphic sensor was made in the form of a silicon model [18] with a resolution of 48×48. Since then, they have been improved and currently possess the desired properties of eyes which were mentioned previously. These are summarized to be: no motion blur due to asynchronized exposure time of pixels, a high dynamic range of 140 dB due to the logarithmic scale, highly compressed visual representation, a high temporal resolution -although not continuous- in the order of microseconds which is equal to a frame rate of 1 MHz [6], and a power consumption of ~ 10 mW which is one-hundreds of the power consumption of a frame-based camera.

Whenever a pixel observes a brightness change that surpasses the lower or higher threshold, a signal in the form of an event is triggered. This kind of neuromorphic vision is named event-based vision. An event $e_i(t_i, x_i, y_i, p_i)$ is triggered at timestamp t_i, at pixel location (x_i, y_i) with polarity $p_i\{-1, +1\}$ whenever the observed brightness (I_t) surpasses a threshold (θ) with respect to the previous observed event (I_{t-1})

$$\log(I_t) - \log(I_{t-1}) \geq \theta. \tag{2.1}$$

N events together forms a stream of events or eventstream $E^{N\times 4}(t, x, y, p)$. This eventstream can be depicted as a list and also visualized when plotted in a continuous space-time which is called the eventspace. It is also possible to convert the events towards frames using conversion algorithms which in its turn can be used to depict a video in a conventional way. Figure 2.1 depicts the eventstream representation, events plotted in eventspace in which the blue and red color-coding indicates the polarity -positive or negative observed brightness change-, and conversion of the events to frames captured by the DAVIS 240C camera [20]. DAVIS stands for "Dynamic and Active-pixel Vision sensor", it is a special type of sensor enabling it to capture vision as events as well as grayscaled frames named APS.



i	$t(\mu s)$	x	y	p
1	10	12	39	$+1$
2	110	88	146	-1
3	134	174	50	$+1$
4	142	114	79	$+1$
5	207	106	95	$+1$
6	234	10	112	-1
⋮	⋮	⋮	⋮	⋮

Figure 2.1: Different kinds of event-based vision representation and its conversion from eventstream to plot in eventspace to frames

Two concepts are often used in computer vision namely: spatial and temporal. These words are used to indicate if it is respectively time- or space-relating. A more practical explanation would be if something is temporal, it is only variable in time and not in space. These two concepts are requisite in order to study event-based cameras as events have space-time relatedness and thus are spatial-temporal (or spatio-temporal).

Although the temporal resolution is dense, meaning that the refresh rate at which the pixels can observe events with a rate of $1 \sim 10$ MHz is considered to be high, the spatial resolution of the sensor is sparse 128×128 pixels [16]. In addition, off-the-shelf event cameras have a maximum bandwidth at which they can process events, deviancy in the captured data collection occur when the observed amount of events exceeds this bandwidth. Namely, photoreceptors on every 2nd row do not process any events as long as the bandwidth is exceeded. This results in a corrupted data collection on which no solution is found yet. On both shortcomings -low spatial resolution as well as retrieving corrupted data- potential solutions can be thought of which inherits the same backbone. In the first case, spatial resolution is increased by finding the values of a higher resolution representative should have. While in the second case, values of the missing blank column in the spatial plane are retrieved.

2.2. Super Resolution

There is various research done to enhance spatial resolution by reconstructing a higher spatial resolution representation of the original lower resolution. A representative in the higher dimension is referred to as super-resolution or super-revolve (SR) and the original resolution as low resolution (LR). This problem of obtaining SR from LR is a classical computer vision problem wherein it is tried to obtain an SR that is as similar as possible to the ground truth high resolution (HR) and research is done extensively both on images as well as frame-based vision [14].

To start with, reconstructing the SR is an ill-posed problem as the ground truth of the SR is not explicitly embedded in the LR which leads to a variety of possible solutions. So the predicted SR is in all cases an approximation of the actual HR. But some techniques result in better performance than their counterparts. Performance is most often evaluated by the mean squared error between the super-resolved image and the groundtruth. Datasets used for this research topic consists of paired LR- and HR frames which are respectively the input (x) to predict SR and ground truth (y) for evaluation. y is usually the original sized image while x is derived from the y by downsampling with kernel (k) by factor (s) and applying some noise (n) to it. This is mathematically described as $x = (y \otimes k) \downarrow_s +n$. In super-resolution, a method is sought to reverse this process, a high-level overview of this process is depicted in Figure 2.2.

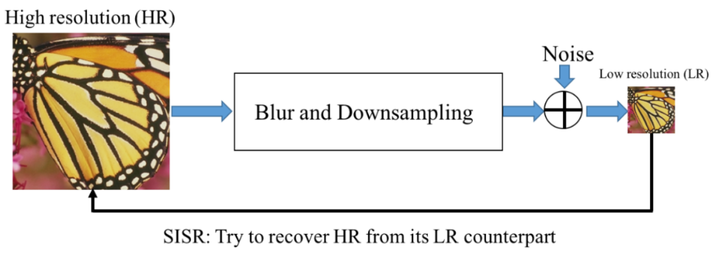


Figure 2.2: High-level overview of creating low resolution image from original high resolution image and reversing this process [36] which is called super revolving

2.2.1. Image Super Resolution

An LR image can be described as an array $I^{\text{LR}} \in [0,1]^{H\times W\times C}$ and a SR is then said to be $I^{\text{SR}} \in [0,1]^{H\times W\times C}$ in which height is denoted as pixel height H, pixel width W, color-channel C and scale factor s. The process of approximating SR for images is referred to as Single Image Super-Resolution (SISR). The dimension of the color channel is 1 when grey scaled and 3 when color is encoded for example Red Green Blue (RGB). Note that an image has only spatial relatedness.

Interpolation-based SISR method

The most elementary method to find I^{SR} is to find the intermediate pixel values by averaging the surrounded values. This is referred to as bilinear interpolating in the x and y direction, but there are also more advanced techniques like bicubic interpolation. These interpolation methods are straightforward and computationally easy to perform. Yet, interpolation tends to neglect the existence of sharp edges as it introduces unwanted fading between all pixels.

Reconstruction-based SR method

Reconstruction-based SR methods consists of a class of algorithms that can cope with both smooth reconstruction of gradients and sharp edges [3] [10]. These methods have great performance and are applicable to a wide range of different scenery as they do not have a bias like learning-based methods do. Unfortunately, reconstruction-based SR method still struggle when the scaling factor is large and are computational far more complex than the interpolation-based SISR method.

Learning-based SISR method

Over the years, many different learning-based approaches have been developed in order to perform SISR. Learning-based refers to the parameters being found in a learnable fashion in which the SR output (y_{pred}) of the network is evaluated by a defined metric -often called loss function ($\mathcal{L}$)- to the HR ground truth (y). Depending on the loss value between y_{pred} and y the parameters are optimized by a process called backpropagation. During backpropagation, the gradient is calculated between each output entry, and the accountable parameters are then adjusted according to the negative direction of the gradient.

One of the disadvantages of applying learning-based methods to computer vision tasks is the number of input arguments as this determines the number of neurons in the input layer of the learning-based method in a default NN. The amount of input arguments is equal to the number of pixels of the image multiplied by the number of channels in each pixel -3 for RGB-. This results in an enormous amount of trainable parameters which makes it difficult to find the right values that make the NN converge towards a solution. This makes it prone to have spatial bias whenever it is trained on a data set in which the images have the focus in the middle of an image and vague background on the border. Both disadvantages are countered by a special class of NN named convolutional neural networks (CNN). This special kind of neural network consists of a set of filters with trainable parameters which are convolved over the input image. A CNN has the property to have a reduced amount of learnable parameters due to the same filters being used in a convolution over the whole image instead of being designated to be applied to a single input argument. At the same time, this prevents the possible spatial bias which can occur in a default NN.

SRCNN is a two-layered CNN architecture that super resolves images and was proposed by [4], an overview of this architecture is depicted in Figure 2.3. A key feature of this architecture is the ability to extract nonlinear patterns due to the non-linear mapping between the two layers and the second layer which enables to the extraction of hidden features.

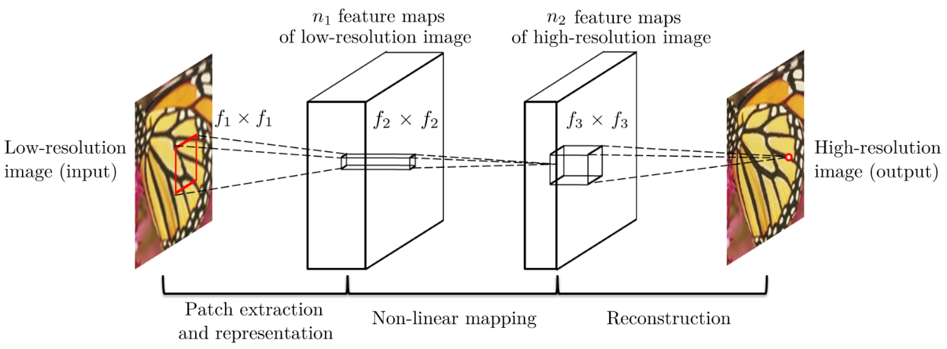

n_1 feature maps of low-resolution image
n_2 feature maps of high-resolution image
$f_1 \times f_1$
$f_2 \times f_2$
$f_3 \times f_3$
Low-resolution image (input)
High-resolution image (output)
Patch extraction and representation
Non-linear mapping
Reconstruction

Figure 2.3: Overview of the SRCNN architecture which uses a Convolutional Neural Network to super resolve images [4]

Due to the preferable properties of CNN on computer vision tasks, CNNs can be found in the base of state-of-the-art SR architectures. SRResNet is a residual network composed of CNNs. Further performance gains have been made by generative adversarial network (GAN) which is based on SRResNet [12]. With this network, it is possible to obtain photo-realistic results by even a scale factor as large as four.

2.2.2. Frame Based Vision Super-resolution

The conventional paradigm representation of a video is by a series of images $I_t \in [0, 1]^{H \times W \times C}$ in which t indicates the timestamp the frame is observed -which has constant intervals- and the amount of proceeded frames per second is referred to as frames per second (FPS). As independent frames have the same characteristics as images, algorithms that realize spatial SR for frame-based vision have commonly the same backbone as image SR. The existence of multiple frames through time introduces a time relevance which makes a video spatio-temporal. In combination with the fact that consecutive frames are taken at relatively short intervals, there is much overlap in the data which enables both spatial and temporal features can be used to super-resolve.

Besides spatial super-resolution, temporal relatedness introduces the option for temporal super-resolution. Whereas in spatial super-resolution intermediate pixels are predicted, in temporal super-resolution intermediate frames are predicted and thereby enhance the frame rate. Temporal super-resolution in combination with deblurring is done by [30] but will not get into detail as it is somewhat out of the scope for event-based vision purposes.

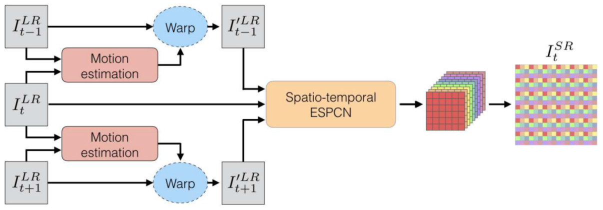
Spatio Super Resolution

Most video spatial SR algorithms have a base of an image SR algorithm that makes use of temporal relevance to further optimize its prediction. Temporal information enables for example flow estimation which can be used to have motion compensation (MC) to diminish blurriness but also boost the ability to predict texture or details that is exposed to motion in frames as is done in [1]. Their proposed architecture uses I_{t-1}^{LR}, I_t^{LR}, I_{t+1}^{LR} to predict I_t^{HR}. Using MC leads to good performance on video reconstruction as is shown in Figure 2.4. The downside of this approach, however, each frame is used three times. This results respectively in computational redundancy and the inability of the system to produce temporally consistent results which are noticeable by flickering inconsistent frames when visualizing the video. To these flaws, [28] has proposed a recurrent framework that realizes temporally consistent results while reducing the computational cost. The framework and the results are depicted in Figure 2.5. The consistency is secured due to the results of previously calculated SR frames being weighed in the calculation of the current SR frame.

(a) Ground truth

(b) No MC x3

(c) MC x3



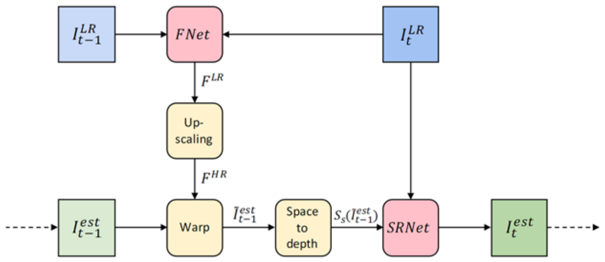
(d) Framework

Figure 2.4: An overview of the proposed framework (d) in comparison with the ground truth (a), spatial super-resolution without motion compensation (b) and motion compensated super-resolution (c) [1]

(a) Ground truth

(b) Bicubic SR x4

(c) Frame Recurrent SR x4



(d) Framework

Figure 2.5: An overview of the proposed Frame-Recurrent Video Super-Resolution results (c) in comparison with the ground truth (a) and super-resolution by bicubic interpolation (b), and an overview of their proposed framework (d) [28]

A less frequently applied deep learning technique to frame SR is via dictionary-based learning. This approach tends to learn a set of patterns of coupled LR and HR which is said to be the dictionary [2]. Then, when an LR frame is fed to the algorithm, small patches of the frame are "looked up" in the dictionary by comparing by similarity. The representative HR of the coupled LR -which is the most similar to the LR patch- is used in the reconstruction of the SR. All super-resolved patches together form the frame SR.

2.2.3. Event Based Video Super Resolution

The algorithms for realizing spatial super-resolution are not interchangeable between frame-based vision and event-based vision as they process vision in a totally different paradigm. This is a direct consequence of the difference in the data structure. To begin with, frame-based vision captures data at an explicit FPS regardless of there being a change in observed scenery. But even if the process rate of event-cameras of 1 MHz is considered as the FPS, it is still will not possible to make a simple CNN converge. This is due to the following property: whereas frame-based vision captures absolute brightness values, event-based vision only depicts relative brightness changes with respect to the previously observed event. In combination with the density of events at 1 μs interval being extremely sparse. This results in the chance of events being observed in neighbouring photoreceptors in the same microsecond being very low. When a convolution is applied at this time interval, the expected value is zero and thus CNNs will not be of any use.

To cope with the sparsity of the data, [14] proposed a model that discretizes the eventstream into temporal bins. In each time bin, the positive and negative events are separated and processed accordingly. Their model is a two-stage scheme mathematical model based on probability theory. The first step consists of two parallel actions, the number of occurred events per pixel is accumulated and denoted in the form of a countmap. This countmap is then upsampled by interpolation to the expected countmap of HR. The second action is to simulate the probability of occurring events per LR pixel by a rate function then use a convolution kernel to determine the new pixels of HR. In this, they assume that the rate function of a pixel is related to its neighbours.

In the second stage, an approximation of the HR is made by generating events according to a homogeneous Poisson process based on the interpolated event countmap and the found rate function. As eventstreams are not a viable option to represent results, especially not in 2-D, a conversion is made. The most straight forward conversion is to discretize the eventstream into k temporal bins $W_k = \{e_k, \ldots, e_{kN-1}\}$ each containing N. For each W_k the pixel values (x, y) are set to 0, 1, 0.5 in the case of respectively a negative, positive, or no events have occurred. This is formulated as

$$I_k(x, y) = \begin{cases} 0, & \text{if } p = -1 \\ 1, & \text{if } p = +1 \\ 0.5, & \text{otherwise} \end{cases} \tag{2.2}$$

and results in a intensity image $(I_k(x, y))$ [21]. This does however neglect the number of occurring events per pixel and even can lead to the cancellation of depicted events in the case of two events with opposite polarity occurring at the same pixel inside the temporal bin. Using this conversion, the results of the SR are depicted in Figure 2.6, and a high-level framework is depicted in Figure 2.7.

This proposed framework is quite simple, computationally efficient, and performs reasonably well as can be seen in Figure 2.6 and a high-level framework is depicted in Figure 2.7. However, the proposed method is not learning- but interpolation-based and although the results seem promising when comparing a 2-d render, it is arguable if performance is likewise when the predicted events are compared in event-space. This is due to the fact that temporal information between consecutive time-bins is not used, which will result in temporal inconsistencies. This is comparable to the case when frame-based vision SR is realized by applying SISR methods independent on frames. In addition, as was the case by interpolation-based SISR, interpolation does not have the ability to restore detailed patterns but rater has a fading effect which is most often not optimal.

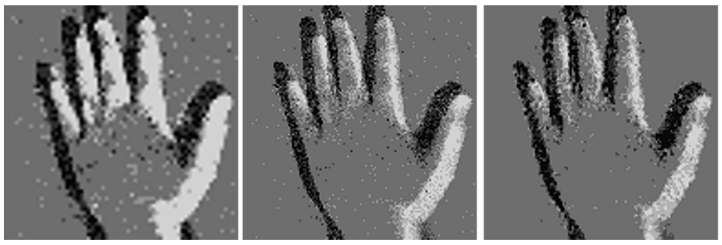


Figure 2.6: On the right side, the result of the proposed two-stage method [14] of super-resolved event-frame on low-resolution input data which is depicted on the left side. This is compared to the high-resolution ground truth which is depicted in the middle.

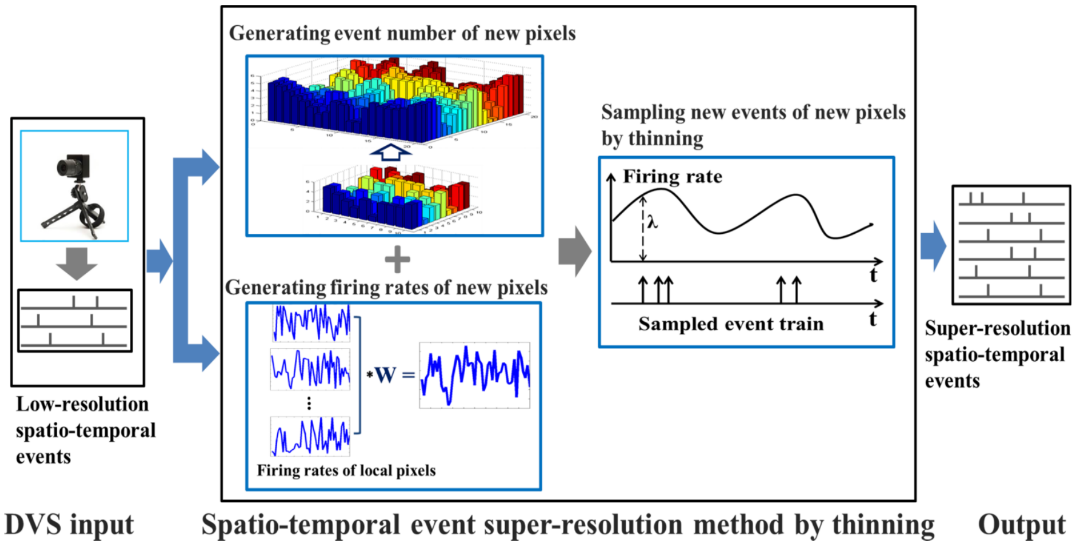

Generating event number of new pixels
Generating firing rates of new pixels
*W =
Firing rates of local pixels
Sampling new events of new pixels by thinning
Firing rate
λ
t
Sampled event train
t
Low-resolution spatio-temporal events
Super-resolution spatio-temporal events
DVS input
Spatio-temporal event super-resolution method by thinning
Output

Figure 2.7: An overview of the proposed two-stage framework which uses a probability-based methodology to realize super-resolution on events [14]

Yet, despite the potential of this proposed mathematical method, most research on SR of event-based vision is done by the use of deep learning. Frame-based SR networks are not suited for evenstreams, but it is a research area that has been developed extensively and has proven performance. So conversions, like used to depict the results in Figure 2.6, are often used as a preprocessing step in order to make an eventstream representation 'fit' the frame-based SR algorithms. Various ways have been developed to convert an eventstream to a representation that is interpretable by frame-based SR algorithms [7]. This section will deliberate on proposed solutions and the conversions they used as a preprocessing step.

The method with the most citations (32) is EventSR [19], they propose a framework that realizes SR on intensity images. The intensity images are compiled according to Equation 2.2, but with a little tweak to enable overlapping of the time bins to allow more temporal information which can lead to a better consistent prediction. An overview of the framework is depicted in Figure 2.8. Noticeable are the similar modules as the earlier discussed frame-recurrent video super-resolution framework 2.5d. The main difference is the conversion to intensity images in the preprocess step -which is needed to make the data fit the algorithm-, and the output is evaluated with respect to the observations of an active sensor pixel (APS). This framework is thereby optimized towards training on the data of a DAVIS camera but not on a plain event camera.

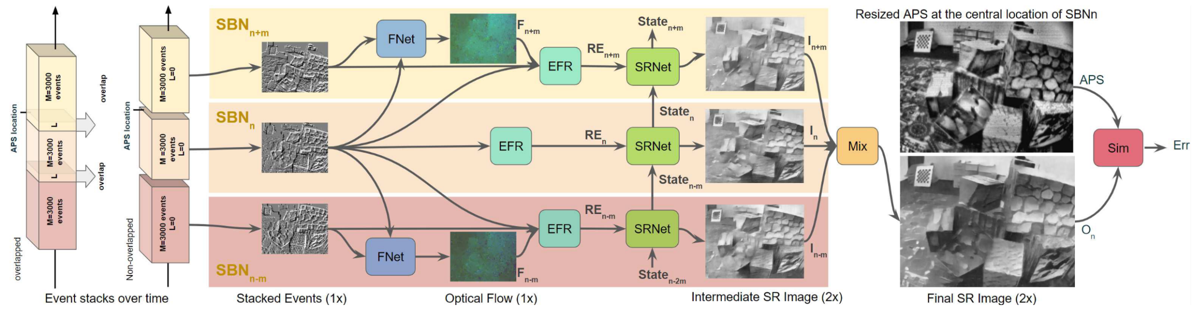


Figure 2.8: An overview of the framework of EventSR which uses intensity images as input to super-resolve it to a gray-scale frame [19]

Other conversions have invented such as, which results in a frame that yields the number of occurring events per pixel. Event-frames are almost identical to the introduced countmap in [14] but without a split between positive and negative events. EventZoom [5] is capable to perform super-resolution on these countmaps and even reconstruct these super-resolved countmaps to a super-resolved eventstream while it also denoises the events. The framework is depicted in Figure 2.10. First, they stack 16 consecutive event-frames, to take some temporal information into account, each event-frame is converted from a 10 ms time bin. The event-frames are converted to an image by E2VID which is then used in a concatenation in the early stage -after which it undergoes a 3D U-Net shaped architecture- and super revolved by FSRCNN to an HR image which is concatenated at the final stage of the framework. Finally, the super-resolved event count is redistributed to an HR eventstream. Keep in mind that although the input and output are both eventstreams, the way it is processed is still as a frame representation neglecting some degree of temporal information. There are two ambiguous claims in this paper. (i) They claim to use event-frames but the E2VID [27] module is made to directly convert events to frames not suited for. (ii) They claim to reconstruct super resolves event-frames to events but it is not explicitly noted how they do use a reference to [9] in section 5 (Applications) but their proposed conversion works on intensity images and not event-frames which the dimension of the output suggests. In addition, code is not provided with the paper which makes it harder to check the validity and it only has 3 citations which are low. However, the paper is published in 2021 so there was not much time for other researchers to cite this paper.

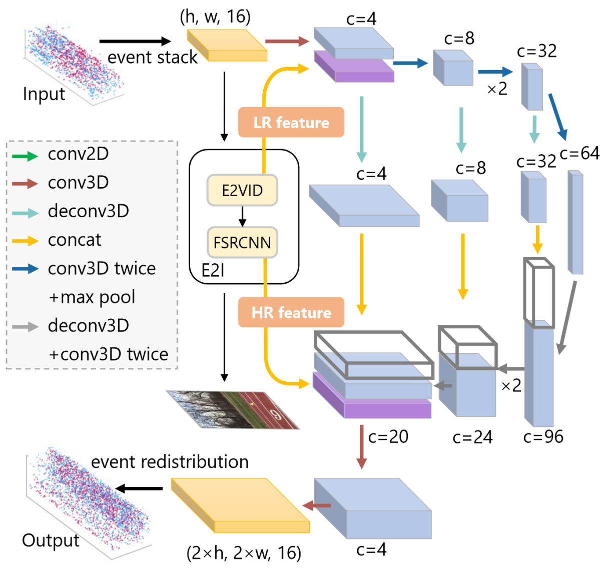
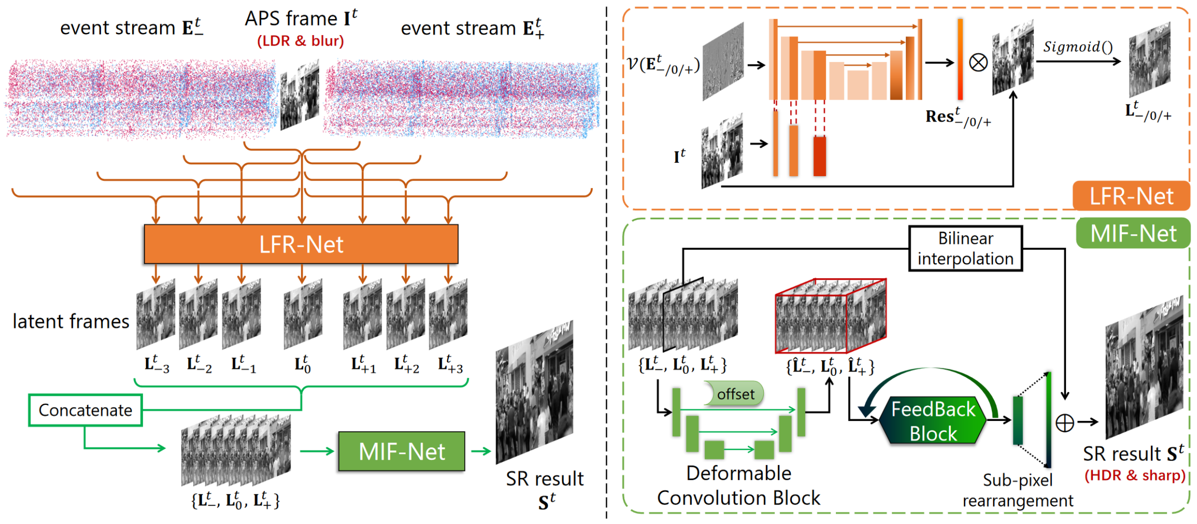


Figure 2.9: An overview of the framework of EventZoom which uses raw-event stream as input and super resolves it to a high-resolution eventstream [5]

EvIntSR-Net (Event Intensity Super Resolution-Net) [8] is another event super revolving architecture. Although the amount of citations is limited, the methodology is interesting as their method preserves temporal information by encoding it into the input. This is realized by transforming the events into a spatial-temporal voxel grid as they propose in [40]. Both the APS frame at time t and the voxel grid representation are used as input into their framework. The first stage consists of a U-Net and produces latent frames which are in the second step used to super resolve into the SR frame by multi-image-fusion net (MIF-Net). The disadvantages are (i) that it takes a lot of data as input which will make it computationally complex. (ii) As the network does not compute in a recurrent fashion, events are used multiple times which results in even more computational complexity. (iii) The voxel grid does allow for more temporal information but as the LFR-Net produces latent frames, it is arguable if it is that useful. This paper is like the previous paper published in 2021 and also only has 2 citations.



event stream $\mathbf{E}^t_-$
APS frame $\mathbf{I}^t$
(LDR & blur)
event stream $\mathbf{E}^t_+$
LFR-Net
latent frames
$\mathbf{L}^t_{-3}$ $\mathbf{L}^t_{-2}$ $\mathbf{L}^t_{-1}$ $\mathbf{L}^t_0$ $\mathbf{L}^t_{+1}$ $\mathbf{L}^t_{+2}$ $\mathbf{L}^t_{+3}$
Concatenate
$\{\mathbf{L}^t_-, \mathbf{L}^t_0, \mathbf{L}^t_+\}$
MIF-Net
SR result $\mathbf{S}^t$
$\mathcal{V}(\mathbf{E}^t_{-/0/+})$
$\mathbf{I}^t$
$\mathbf{Res}^t_{-/0/+}$
Sigmoid()
$\mathbf{L}^t_{-/0/+}$
Bilinear interpolation
$\{\hat{\mathbf{L}}^t_-, \mathbf{L}^t_0, \hat{\mathbf{L}}^t_+\}$
offset
Deformable Convolution Block
FeedBack Block
Sub-pixel rearrangement
(HDR & sharp)

Figure 2.10: An overview of the framework of EvIntSR-Net which uses both APS frame and events to super-resolve APS frames [8]

eSL-Net [34] stands for event Sparse Learning-Net. They manage to deblur, denoise and super resolve APS frames in combination with events to intensity images. It enables SR in both the temporal (with respect to the framerate of APS) as well as the spatial space by predicting the latent frames. The temporal SR can achieve a frame rate equivalent to the frequency of occurring events. It is published in 2020 and so far has already 17 citations. The performance seems promising and the code is made publicly available.

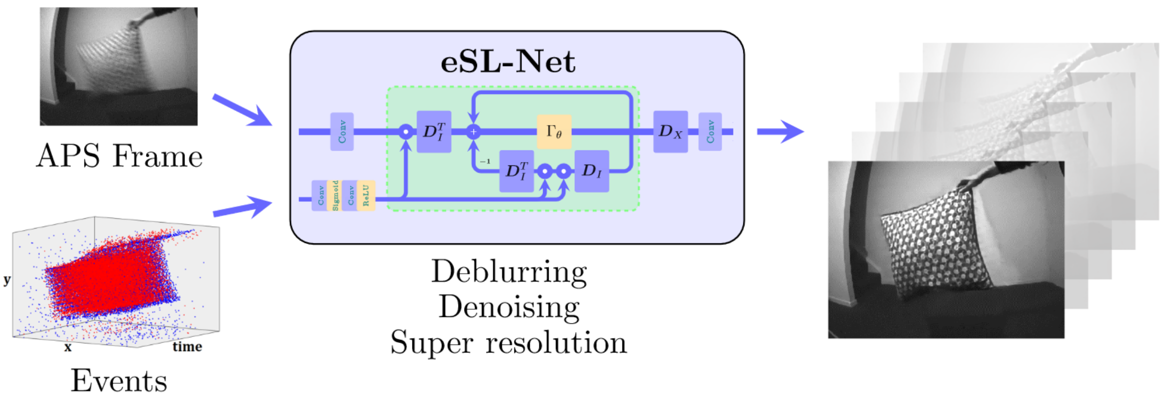


Figure 2.11: An overview of the framework of eSL-Net which uses both APS frame and events as input to deblur, denoise and super-resolve APS frame (in temporal and spatial context) [34]

As the DAVIS camera is the only event-based camera that is equipped with an APS, most research is done towards a combination of a grayscaled image from the APS with events. But there is already a model proposed which uses an input of colored RGB frames and intensity frames named Event-based VSR framework (E-VSR) [11]. The framework is depicted in Figure 2.12 and is quite overwhelming, but is actually a combination of a newly proposed event-based asynchronous interpolation (EAI) module from which its outputs are fed into a recurrent video super-resolution module [28].

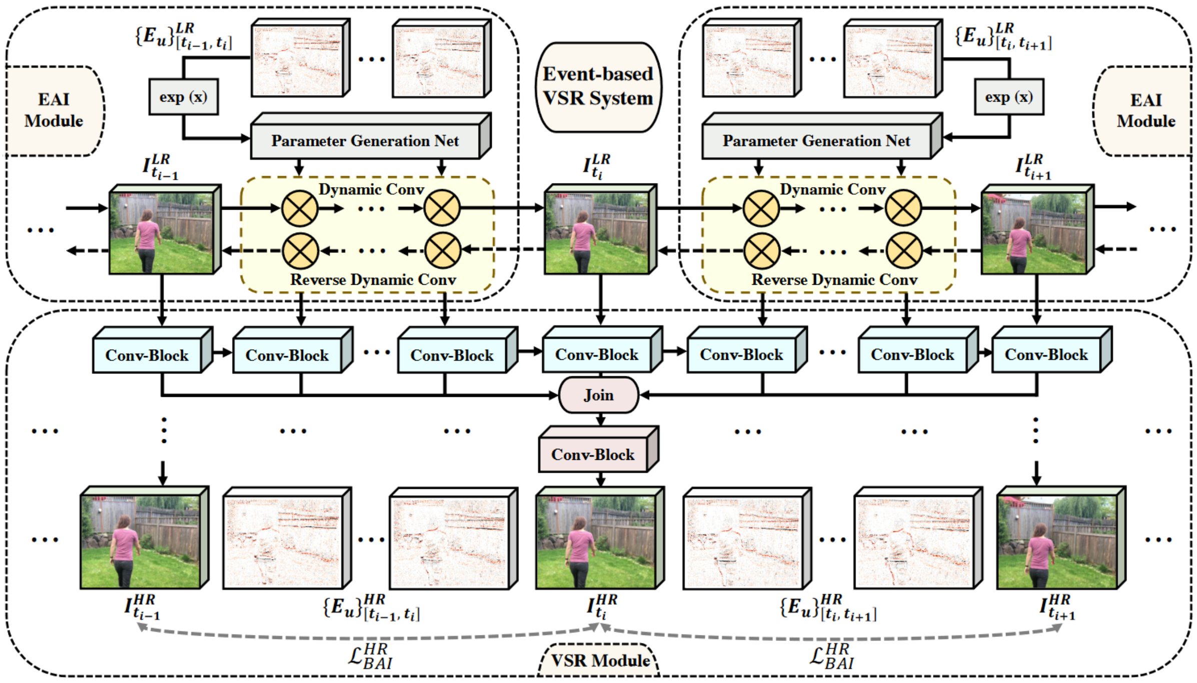


Figure 2.12: An overview of the framework of E-VSR which uses both colored frames (I^{LR}_{t-1}, I^{LR}_t and I^{LR}_{t+1}) and events to super-resolve the colored frame in a recurrent fashion [11]

Spiking Neural Network

All the reviewed super-resolution methods for events inflict a conversion from event-space to frames and thereby lose some temporal information as the exact timestamp of the events is neglected in the frame. This makes sense because it enables the use of frame-based SR methods from which performance is proven extensively. There is however also a special class of neural network that is designed to cope with the representation of events by spikes directly. Spikes is a representation of data that is biologically inspired on the human nervous system. These networks are called Spiking Neural Networks (SNN) [32] [24] and is only recently been successfully implemented. A major challenge is to train an SNN as backpropagation is not applicable. Backpropagation is not applicable as the spiking signals are not derivable which is key for backpropagation to work. There are methods developed to transform a trained conventional DNN counterpart of the SNN. This training fashion is used in [23] to find the parameters of their proposed model which does classification on the neuromorphic MNIST (N-MNIST) dataset. There is also a temporal credit assignment policy designed as an alternative for backpropagation and is used in [35] to train their proposed network which was able to do action classification on a hand gesture dataset with an accuracy of 96.59% for 10 category classification and 90.28% for 11 category classification. Another challenge is the lack of mature software toolboxes that includes functions to create an SNN. This makes it currently challenging to create SNNs and in return prevent more difficult computer vision tasks -like image segmentation or super-resolution- to be performed as they need far more complex architectures. Inherently, it is not proven to work in the first place for more complex computer vision tasks.

From this, we conclude that SNNs are currently not far enough developed to be used to realize super-resolution of events. But looking at data representation we see similarities between events and pointclouds from which we will review upsampling algorithms in the next section.

2.2.4. Point Cloud Super Resolution

A point cloud is a representation in two or more dimensions of data in which coherence between the data points is not explicitly present, it is comparable with a scattered plot. This makes it interesting to study how architectures that are designed to cope with point cloud representation are able to do so as it could be of inspiration for processing events. One example of a sensor that outputs this kind of representation is a LiDAR. This sensor consists of multiple laser pointers which are spinning around an axle while each laser measures distances. The measured data is represented by the physical euclidean dimensions (x, y, z) of each measured point. With the LiDAR -or other point cloud representation- the data density is often sparse which results in a likewise urge for super resolving the LR pointcloud by predicting an HR representation.

The first proposed network that was able to super resolve (or upsample) is Point Upsampling-Net (PU-Net) [39] in 2018. This network is designed to upsample point cloud representatives of an item which tend to have more coherence with respect to events as it resembles one object. The data is fed to the network as an unordered stacked list of size $(N, 3)$ of the N points, each with 3 euclidean dimensions. Then the proposed model searches for geometric patterns of the presented object which are used to find its the surface on which points are generated by a Poisson disk sampling. Results of this network are shown in Figure 2.13. Using the same geometric features as a principal to determine the surface of the object, [38] proposes a multi-step progressive upsampling (MPU) network one year later in 2019. MPU has an even better edge awareness which enhances the performance to retain details in the upsampled pointcloud.

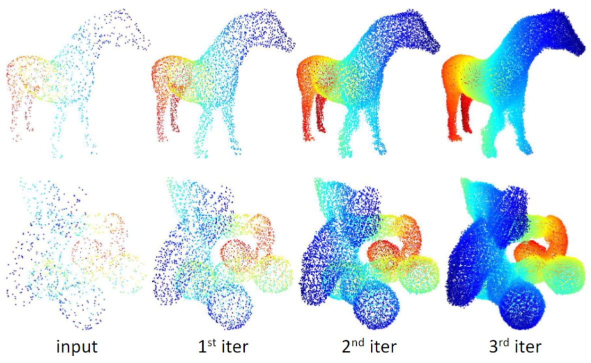


Figure 2.13: Results of super-resolving pointclouds each iteration with scaling factor 2 using PU-Net [39]

Other methodologies have been invented like Point Upsampling Generative adversarial Network (PU-GAN) [15]. A GAN is a special type of neural network and consists of two separate networks -hence adversarial-. One of them is the generator which -in this case- predicts the HR pointcloud given an LR pointcloud. The second network is called the discriminator which has the goal to determine if it is fed the output of the generator or the ground truth. Depending on whether the generator is making a valid verdict on its decision, the parameters of either the generator or discriminator. In this way, over time both the generator as well as the discriminator are converging towards an optimization in which the generator is keep generating more similar representatives of the ground truth, and the discriminator is getting better in distinguishing generated input versus a ground truth.

In PU-GAN, the generator consists of three components that successfully extract features, expand features and generate the HR pointcloud. The discriminator has a more basic architecture but with a self-attention layer to enhance the feature learning. An overview of the network can be seen in Figure 2.15.

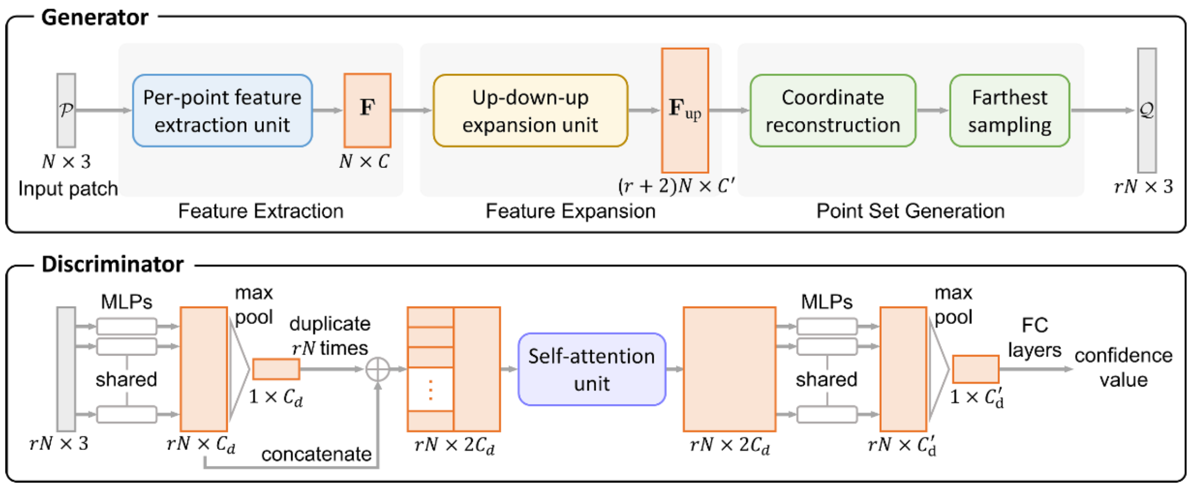


Figure 2.14: An overview of the framework and its different modules in the Generator and Discriminator of PU-GAN which super-resolves pointcloud using a GAN network [15]

Graph Convolutional Network (GCN) is a type of CNN. Whereas CNNs are made for 2D array convolutions, GCNs can be used on graph-structured data directly. In GCNs a specified amount of neighbour nodes are used in each convolution. The points in a pointcloud are also not aligned in a 2d array but each point does have neighbours at a certain distance. Neighbouring points have embedded information about the coherence of the pointcloud and thus can be exploited to find hidden features in order to make predictions about the SR of the pointcloud. Pointcloud Upsampling - Graph Convolutional Network (PU-GCN) [26] is a network that makes use of GCN to upsample a pointcloud. Its performance is compared in Table 2.1 to the previously discussed networks whilst it has the least amount of parameters and computation time. The used metrics are Chamfer distance (CD), Hausdorff distance (HD), and point-to-surface distance (P2F) [33]. In addition, their proposed algorithm is made to upsample noisy and complex data and thus have some degree of resemblance with the coherence of events.

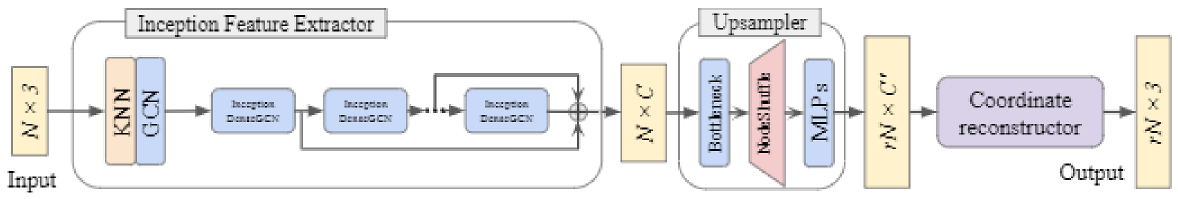


Figure 2.15: An overview of the framework of PU-GCN which super-resolves point clouds using a graph convolutional network architecture [26]

Table 2.1: Performance comparison of the previous discussed pointcloud upsampling networks with different metrics Chamfer distance (CD), Hausdorff distance (HD), and point-to-surface distance (P2F)

Network	**CD** ↓ 10^{-3}	**HD** ↓ 10^{-3}	**P2F** ↓ 10^{-3}	Param. Kb	Time ms
PU-Net [39]	0.556	4.750	4.678	814.3	10.04
MPU [38]	0.298	4.700	2.855	76.2	10.86
PU-GAN [15]	0.280	4.640	**2.330**	684.2	14.28
PU-GCN [26]	**0.258**	**1.885**	2.721	**76.0**	**8.83**

The latest (2021) proposed pointcloud upsampling network is proposed by [37]. Meta-PU makes likewise use of GCNs as in PU-GCN, but while PU-GCN is trained for a fixed scale factor, the scale factor can be adjusted in Meta-PU. The ability to adjust the scale factor is preferable in practical cases for example in LiDAR data when sparsity is divergent depending on the distance of the detected object to the LiDAR sensor. The variable scaling factor is made possible by implementing a meta-learning block which dynamically changes the graph convolutional weights depending on the desired scale factor.

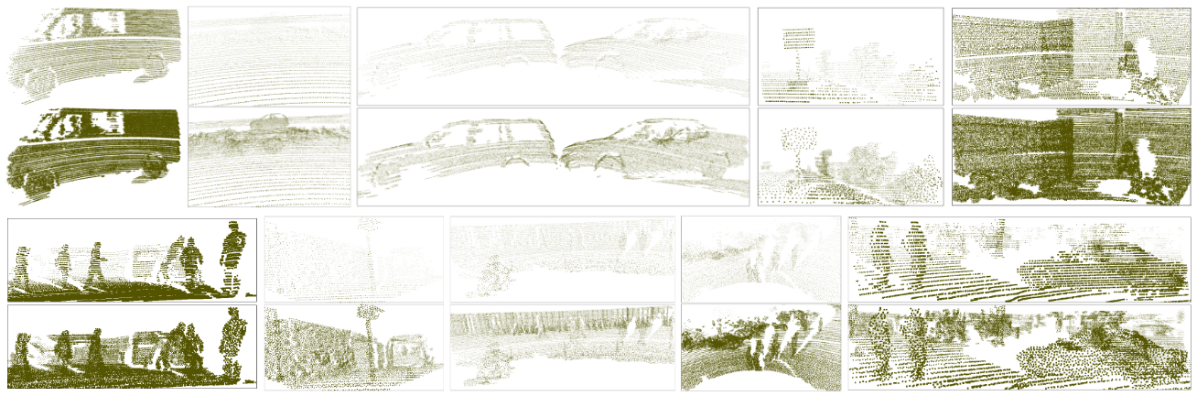


Figure 2.16: Results of Meta-PU on unseen sparse and non-uniform LiDAR data. Every first row is depict LR input data and every second row its corresponding SR. Meta-PU can predict both isolated objects (depicted on the 2nd row) as well as scenery (depicted on the 4th row)[37]

3 Discussion

Super-resolution is a problem in computer vision on which research is done extensively. Literature shows that convolutional neural networks form the backbone of both image as frame-based video resolution. Additionally, state-of-the-art frame-based video super-resolution makes use of consecutive frames I_{t-1} and I_{t+1}. Those frames embed hidden information which is relevant to uncovering 'unseen' information about details. Recurrent networks prevent frames from being used multiple times in the calculation of their neighbouring super-resolved frames which is preferable to keep reducing the computational complexity.

After reviewing the proposed super-resolution methods, it is concluded that (except for the non-learning-based method [14]) the same methodologies as reviewed in frame-based super-resolution are applied. To do so, event-based data are converted to a frame-based representation to make the algorithms fit. Conversion is done by discretizing events into frame-like representations. Discretization has inherently the following disadvantages: (i) temporal information is lost, and (ii) the transformation from events to frames counters the sparsity of the data which was one of the key properties of event-based vision. Except for [5] -from which it is unclear how they are managed-, all proposed models return a super-resolved frame instead of events. Having a super-resolved event frame is interesting for human interpretability, but there is no necessity for machine interpretability. From this we argue that it should be omitted to transform the input to frames if there is no need for human interpretability.

To maintain the temporal information of the events whilst cherishing the sparsity of event-based vision, we should seek techniques that enable event representation as directly input. One class of networks that cope with event representations are spiking neural networks. These networks enable the processing of bio-inspired spiking signals (similar to the event representation). Unfortunately, research towards spiking neural networks is still in a fairly early stage and thus is not far enough developed to implement them in complex tasks like super resolving. It is however most likely to be a crucial class of neural network for processing event-based vision in the future.

After comparing data representations, it was concluded that events and pointclouds are similarly structured. This lets us get inspiration from pointcloud upsampling algorithms. We reviewed two different state-of-the-art methodologies. Namely, one which is suited to upsample pointclouds representations of self-contained surface of single items (e.g. chairs, cars, and coffee cups). They realize this by finding the surface of the items and generating points on that surface, this is not possible in the way how events are observed as they do not form a self-contained surface nor represent a single item and are in practice quite noisy and have much less coherence with nearby points. The second method is by the use of graph convolution networks which is a special type of convolutional network, but instead of convolving a filter of a 2-dimensional array it directly convolves points with a specified amount of nearest neighbours. This method is used in algorithms to super-resolve LiDAR generated pointclouds that are quite noisy and share a certain level of similarities with events.

4 Research Questions

A lot of different computer vision architectures are developed and the combinations of these lead to even a more vast variety of advanced networks. These are however optimized for frame-based representation which results in needed conversions of events to make it fit the frame-based algorithm. But we should cherish the advantage of data sparsity and embedded temporal information of the event representation as it is one of its key features and can be used as an extra feature to enhance super-resolution of events in both power efficiency as well as quality. Conversion to intensity- or reconstructed images should thus be prevented as much as possible. Of course, this is necessary in order for humans to visualize the data, but when this may not necessary this conversion could be skipped. So to assure the data sparsity and to make optimal use of temporal information, we should look for techniques that are able to process the data in event-space. Research towards super-resolution in event-space is a novel area in which no results are made nor solutions proposed.

Looking at other data representations, it can be concluded that a pointcloud representation has similarities with events. Reviewed upsampling algorithms for pointclouds by the use of a graph convolutional network seems to have potential for super-resolution applications of events. There could be thought of a two-step framework in which (i) an eventstream is transformed to a pointcloud representation by neglecting the polarity which is upsampled by a pointcloud upsampling framework, and (ii) the polarities of the upsampled points are predicted by either a K-nearest neighbor or a trainable classifier/segmentation algorithm.

This raises the following research question: **To what extent can event-based vision be super-resolved in event-space?**

With sub-questions:

1. To what extent can naive algorithm be used to super-resolve events or would it suffice to have a learning-based algorithm?
2. To what extent is it power and computational efficient to realize super resolution in event-space instead of super-revolve event-frames?
3. Do events converted to frame-based video result in a lower mean squared error with respect to a ground truth when the events are super-resolved in eventspace rather then after the conversion?

5 Hypothesis

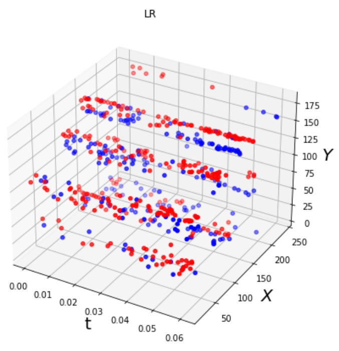
We see in the literature currently no proposed solutions for super resolving event-based vision in event-space. All the solutions in the topic of super-resolution applied to event-based vision involve a discretization somewhere in the process. There is, however, a special class of neural network that is suited for event-based vision representation, but its development still needs major progress before complex computer-vision tasks like super-resolution can be performed. Also, it is unsure if it will work ever.

Fortunately, we see potential in applying graph convolutional networks on event-based vision as was done in [26] and [37]. With these types of networks, it is not necessary to discretize events and it is already proven to super-resolve noise sparse pointclouds. Although pointcloud (x, y, z) representation is the same as event representation (t, x, y, p), we still see potential to overcome this. One way is to neglect the polarity initially and thus convert the events to a pointcloud as both have now 3 euclidean dimensions. The polarity could be predicted after it is super-resolved, this converts it back to events. A visualization these conversion steps are shown in Figure 5.1.

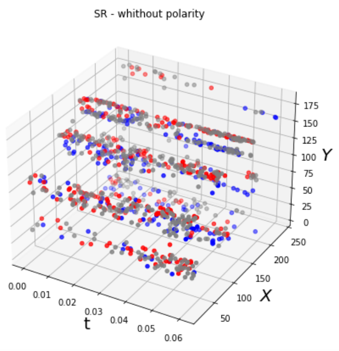
Retrieving the polarity using a naive k-NN algorithm is shortly tested, which gave a 93% accuracy without optimizing. Some more advance (learnable) binary classification algorithms can be investigated if optimization of the k-NN algorithm does not improve accuracy. Another possibility is to split the events based on their polarity into a set of positive events and a set of negative events. In this way, you can super-resolve the point clouds apart from each other and concatenate in the end without losing information about the polarity. It is, however, expected that splitting by polarity prevents coherence to be used by the network as a hidden feature in its prediction and thus worsens performance.

Power and computational complexity are almost entirely induced by the given amount of input arguments and the amount of made computations to each input parameter. Keeping the events in sparse representation results in a lower amount of input arguments than when events are converted to event-frames. The reduction of input parameters is not explicit as it is depending on the number of observed events in an arbitrary time. The difference in the number of parameters between proposed event super-resolution networks and a network that is able to super-resolved events in event-space needs investigation. From the reviewed papers it was not clear how many parameters event super-resolution networks have. But looking at the difference between pointcloud networks in Table 2.1 it is clear that graph convolution network-based models have 10 times fewer parameters than their competitors. Although pointcloud super-resolving networks are not comparable with event-based super-resolution networks, it still gives insight that graph convolution-based networks do not require the most amount of parameters. Reasons behind this can be explained due to the fact that graph convolution networks have shared parameters like a convolutional neural network, which is a type of network that we see being used in state-of-the-art frame-based super-resolution networks.

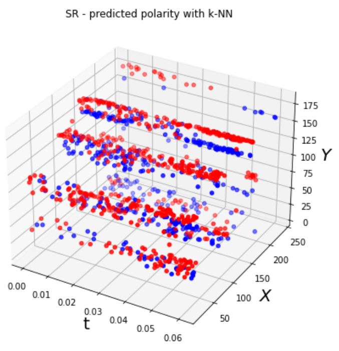
We see potential in performing super-resolution directly on events as the data representation is sparse and includes temporal information, unlike discretized conversions that are currently used. This makes it a data representation that is potentially less power-demanding to super-resolve while including a more complete set of features makes it feasible to super-resolve more accurately compared to discretized events. It will be interesting to see if a model can be found that maximizes the potential of event-based data representation to super-resolve more accurately while being less power demanding than event-frame-based super-resolution counterparts.



(a) LR



(b) SR without polarity



(c) SR with predicted polarity

Figure 5.1: A two-stage scheme results of super resolving a set and retrieving its polarity back in the second stage

6
Methodology

Our research differs from available solutions to super-resolve events in both methodology and output. The proposed working principle has similar input and output as the EventZoom model [5], but with a different methodology. Namely, EventZoom makes use of convolutional neural networks while we see potential to use graph convolutional networks like done in [26] and [37] instead. Because of this, it is no longer necessary to discretize events which in it self is unique, and therefore an interesting research topic. EventZoom, however, is a network that makes an interesting option to use as a benchmark. All other reviewed models do not have solely eventstream as input and output not directly usable as a benchmark.

At the beginning of the research, an investigation will be performed on ’simpler’ naive algorithms to realize super-resolution of events. The found algorithm will then be used as a benchmark, next to EventZoom, to test more advanced algorithms later on in the research. This initial investigation also contributes to even more insight into event-based vision which will be of use when implementing more advanced algorithms like the graph convolutional network.

During the research, various validations will be done at different phases of the research using datasets ranging from simple to complex. In this way, more simple models can be found in an early stage which will be upgraded into more powerful and complex models within each phase. The majority of neuromorphic datasets can be categorized into (i) artificial conversion of existing frame-based data set, (ii) conversion of an image datasets by filming a static image using an event camera [13], or (iii) actual neuromorphic captures. The first dataset which will be used to validate our model is N-MNIST [22] followed by N-CIFAR10 [13], and finally, N-Caltech101 [22] these are all conversions of an image data set. Depending on how well our proposed model converges towards a solution, a validation can be done on an actual event-based vision.

Answers to our research question will be found by a final evaluation to verify and validate our hypothesis. This evaluation will consists of three separate tests, (i) in event space by a combination of a pointcloud similarity metric like Chamfer distance or Hausdorff distance and a precision metric for the binary classification of the polarities (if applicable). (ii) Computation time will be compared with other models and our found naive algorithm as a benchmark. (iii) Frame-based conversion will be compared by performing super-resolution in event space and then converting events to frames using E2VID to review models that first convert to event frames and then super-resolve.

7 Important Dates

Out of the office due to vacation from 27.07.2022 - 09.08.2022

- Literature research – 07.02.2022 - 15.04.2022
- MSc Thesis – 19.04.2022 - 02.09.2022
  - Midterm Review – 01.07.2022
  - Submitting the draft version of thesis – 22.07.2022
  - Green light – 05.08.2022
  - Thesis deadline – 02.09.2022
- Presentation and defense – 07.10.2022

References

[1] Jose Caballero et al. *Real-Time Video Super-Resolution with Spatio-Temporal Networks and Motion Compensation*. Tech. rep.

[2] Qiqin Dai et al. "Dictionary-based multiple frame video super-resolution". In: *Proceedings - International Conference on Image Processing, ICIP*. Vol. 2015-December. IEEE Computer Society, Dec. 2015, pp. 83–87. ISBN: 9781479983391. DOI: 10.1109/ICIP.2015.7350764.

[3] Shengyang Dai et al. "SoftCuts: A soft edge smoothness prior for color image super-resolution". In: *IEEE Transactions on Image Processing* 18.5 (2009), pp. 969–981. ISSN: 10577149. DOI: 10.1109/TIP.2009.2012908.

[4] Chao Dong et al. *LNCS 8692 - Learning a Deep Convolutional Network for Image Super-Resolution*. Tech. rep. 2014. URL: http://mmlab.ie.cuhk.edu.hk/projects/SRCNN.html..

[5] Peiqi Duan et al. *EventZoom: Learning to Denoise and Super Resolve Neuromorphic Events*. Tech. rep. URL: https://sites.google.com/view/EventZoom.

[6] Guillermo Gallego et al. "Event-based Vision: A Survey". In: (Apr. 2019). DOI: 10.1109/TPAMI.2020.3008413. URL: http://arxiv.org/abs/1904.08405%20http://dx.doi.org/10.1109/TPAMI.2020.3008413.

[7] Daniel Gehrig et al. "End-to-end learning of representations for asynchronous event-based data". In: *Proceedings of the IEEE International Conference on Computer Vision*. Vol. 2019-October. Institute of Electrical and Electronics Engineers Inc., Oct. 2019, pp. 5632–5642. ISBN: 9781728148038. DOI: 10.1109/ICCV.2019.00573.

[8] Jin Han et al. *EvIntSR-Net: Event Guided Multiple Latent Frames Reconstruction and Super-resolution*. Tech. rep.

[9] Yuhuang Hu, Shih-Chii Liu, and Tobi Delbruck. *v2e: From Video Frames to Realistic DVS Events*. Tech. rep. URL: https://github.com/SensorsINI/v2e.

[10] IEEE Staff and IEEE Staff. *2008 IEEE Conference on Computer Vision and Pattern Recognition*. ISBN: 9781424422432.

[11] Yongcheng Jing et al. *Turning Frequency to Resolution: Video Super-resolution via Event Cameras*. Tech. rep. URL: https://osf.io/6c3d9/,.

[12] Christian Ledig et al. "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network". In: (Sept. 2016). URL: http://arxiv.org/abs/1609.04802.

[13] Hongmin Li et al. "CIFAR10-DVS: An event-stream dataset for object classification". In: *Frontiers in Neuroscience* 11.MAY (May 2017). ISSN: 1662453X. DOI: 10.3389/fnins.2017.00309.

[14] Hongmin Li et al. *Super-resolution of spatiotemporal event-streamimagecaptured bytheasynchronous temporal contrast vision sensor*. Tech. rep.

[15] Ruihui Li et al. "PU-GAN: a Point Cloud Upsampling Adversarial Network". In: (July 2019). URL: http://arxiv.org/abs/1907.10844.

[16] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. "A 128 $\times$ 128 120 dB 15 μs latency asynchronous temporal contrast vision sensor". In: *IEEE Journal of Solid-State Circuits* 43.2 (Feb. 2008), pp. 566–576. ISSN: 00189200. DOI: 10.1109/JSSC.2007.914337.

[17] Warren S Mcculloch and Walter Pitts. *A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY* n*. Tech. rep. 2. 1990, pp. 99–115.

[18] Carver A Mead and M A Mahowald. *A Silicon Model of Early Visual Processing*. Tech. rep. 1988, pp. 91–97.

[19] S I Mohammad Mostafavi GIST et al. *Learning to Super Resolve Intensity Images from Events https://github.com/gistvision/e2sri*. Tech. rep. URL: https://github.com/gistvision/e2sri.

[20] Elias Mueggler et al. “The Event-Camera Dataset and Simulator: Event-based Data for Pose Estimation, Visual Odometry, and SLAM”. In: (Oct. 2016). DOI: 10.1177/0278364917691115. URL: http://arxiv.org/abs/1610.08336%20http://dx.doi.org/10.1177/0278364917691115 .

[21] Anh Nguyen et al. “Real-Time 6DOF Pose Relocalization for Event Cameras with Stacked Spatial LSTM Networks”. In: (Aug. 2017). URL: http://arxiv.org/abs/1708.09011.

[22] Garrick Orchard et al. “Converting static image datasets to spiking neuromorphic datasets using saccades”. In: *Frontiers in Neuroscience* 9.NOV (2015). ISSN: 1662453X. DOI: 10.3389/fnins.2015.00437.

[23] Alberto Patiño-Saucedo et al. “Event-driven implementation of deep spiking convolutional neural networks for supervised classification using the SpiNNaker neuromorphic platform”. In: *Neural Networks* 121 (Jan. 2020), pp. 319–328. ISSN: 18792782. DOI: 10.1016/j.neunet.2019.09.008.

[24] Michael Pfeiffer and Thomas Pfeil. *Deep Learning With Spiking Neurons: Opportunities and Challenges*. Oct. 2018. DOI: 10.3389/fnins.2018.00774.

[25] Christoph Posch et al. “Retinomorphic event-based vision sensors: Bioinspired cameras with spiking output”. In: *Proceedings of the IEEE* 102.10 (Oct. 2014), pp. 1470–1484. ISSN: 15582256. DOI: 10.1109/JPROC.2014.2346153.

[26] Guocheng Qian et al. “PU-GCN: Point Cloud Upsampling using Graph Convolutional Networks”. In: (Nov. 2019). URL: http://arxiv.org/abs/1912.03264.

[27] Henri Rebecq et al. *Events-to-Video: Bringing Modern Computer Vision to Event Cameras*. Tech. rep. URL: https://youtu.be/IdYrC4cUO0I.

[28] Mehdi S M Sajjadi, Raviteja Vemulapalli, and Matthew Brown. *Frame-Recurrent Video Super-Resolution*. Tech. rep.

[29] Catherine D. Schuman et al. “A Survey of Neuromorphic Computing and Neural Networks in Hardware”. In: (May 2017). URL: http://arxiv.org/abs/1705.06963.

[30] Eli Shechtman, Yaron Caspi, and Michal Irani. *Space-Time Super-Resolution*. Tech. rep.

[31] Emma Strubell, Ananya Ganesh, and Andrew McCallum. “Energy and Policy Considerations for Deep Learning in NLP”. In: (June 2019). URL: http://arxiv.org/abs/1906.02243.

[32] Amirhossein Tavanaei et al. “Deep Learning in Spiking Neural Networks”. In: (Apr. 2018). DOI: 10.1016/j.neunet.2018.12.002. URL: http://arxiv.org/abs/1804.08150%20http://dx.doi.org/10.1016/j.neunet.2018.12.002.

[33] Dahlia Urbach, Yizhak Ben-Shabat, and Michael Lindenbaum. *DPDist : Comparing Point Clouds Using Deep Point Cloud Distance*. Tech. rep. URL: https://github.com/dahliau/DPDist.

[34] Bishan Wang et al. *Event Enhanced High-Quality Image Recovery*. Tech. rep. URL: https://github.com/Shi.

[35] Yannan Xing, Gaetano Di Caterina, and John Soraghan. “A New Spiking Convolutional Recurrent Neural Network (SCRNN) With Applications to Event-Based Hand Gesture Recognition”. In: *Frontiers in Neuroscience* 14 (Nov. 2020). ISSN: 1662453X. DOI: 10.3389/fnins.2020.590164.

[36] Wenming Yang et al. “Deep Learning for Single Image Super-Resolution: A Brief Review”. In: (Aug. 2018). DOI: 10.1109/TMM.2019.2919431. URL: http://arxiv.org/abs/1808.03344%20http://dx.doi.org/10.1109/TMM.2019.2919431.

[37] Shuquan Ye et al. “Meta-PU: An Arbitrary-Scale Upsampling Network for Point Cloud”. In: *IEEE Transactions on Visualization and Computer Graphics* (2021). ISSN: 19410506. DOI: 10.1109/TVCG.2021.3058311.

[38] Wang Yifan et al. “Patch-based Progressive 3D Point Set Upsampling”. In: (Nov. 2018). URL: http://arxiv.org/abs/1811.11286.

[39] Lequan Yu et al. *PU-Net: Point Cloud Upsampling Network*. Tech. rep.

[40] Alex Zihao Zhu. *Unsupervised Event-based Optical Flow using Motion Compensation*. Tech. rep.