Master thesis

Semi-supervised point cloud segmentation on railway data

by Bram Dekker

Supervisor: dr. F. Ahmed
Committee: dr. N. Strisciuglio, B. Ton MSc, dr. J. Linssen

Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS)
University of Twente
Drienerlolaan 5, 7522 NB Enschede, Netherlands
August 30, 2023

Abstract

Monitoring digital twins of the railway infrastructure is safer, less erroneous and faster compared to physical inspection. Digital representations of the railway environment are often obtained via sensors that produce unlabelled point clouds. The point clouds need to be semantically labelled to generate digital twins. Manual labelling requires substantial effort. Deep learning has shown great potential in semantic segmentation tasks using supervised learning. However, there are no publicly available railway datasets. This study thus explores semi-supervised learning for point cloud semantic segmentation. Specifically, two approaches are implemented. In the active learning approach, an algorithm is developed to select the most informative data. The SO-Net segmentation model is trained on only a small portion of the most informative data. For the generative few-shot learning approach, synthetic data is created based on a small labelled dataset. A PointNet++ model is then trained on this synthetic dataset. Both approaches show promising results. The active learning approach achieves over 95% of the performance compared to the fully supervised method using 37.5% less labelled data. The performance of the few-shot learning approach is equivalent to the state-of-the-art while training on only synthetic data. A major challenge for point cloud semantic segmentation in the context of railways is the inherent class imbalance in the data. Further research towards techniques that address class imbalance could improve the performance of the models.

Keywords: deep learning, semi-supervised learning, point cloud, semantic segmentation, railway

Acknowledgements

Firstly, I would like to thank my supervisor, dr. F. Ahmed, for all the support and guidance during the entire length of this project. Furthermore, I would like to thank B. Ton MSc for the technical support on the topic. Through their feedback, I was able to complete this research successfully.

Secondly, I want to thank everyone working at the Ambient Intelligence group from Saxion University of Applied Sciences. Their work environment provided me with everything I needed in order to execute this project. A special thanks to everyone involved in the DTSpoor project. The weekly meetings gave structure to my schedule which aided me in accomplishing this work.

Lastly, I would like to thank my family and friends for their unconditional support during the entire length of my studies and for making my study period much more enjoyable.

Contents

Chapter 1

Introduction

The train is a commonly used means of transport for both people and freight. It accounted for about 5.5% of the total passenger transport and 16% of the freight transport in the EU in 2022 [21]. To keep the trains running safely and punctually, the rail environment should be regularly monitored for damages and wear to fix defects quickly and perform maintenance timely. The railway is often monitored manually by railway employees which is unsafe, error-prone and time-consuming [34, 107]. Unsafe because the employees are exposed to electrically charged parts of the wiring system and moving rail vehicles. Error-prone because it is human to make mistakes. Time-consuming because of the physical travelling to the location of inspection. Automating the monitoring process means employees are less exposed to dangers in the railway environment. And since computers are efficient at repetitive tasks [61] there will be fewer errors on top of a faster execution time.

To automate the monitoring of railway tracks, digital twins could be generated. A digital twin is best described as "the effortless integration of data between a physical and virtual machine in either direction" [37]. This virtual model of the railway could then be analysed by employees via a computer or, ideally, by the computer itself. Digital analysis ensures there is less manual inspection needed. Additionally, the remaining manual maintenance and repairs can be thoroughly prepared beforehand saving time and reducing error probability and safety risks.

To create digital twins, an efficient method to digitise the railway environment must be developed. Due to the rapid technological advancements of recent years, several sensors have been developed that can scan surroundings quickly and precisely. Most of these sensors output a 3D point cloud. A point cloud is a set of points in a 3D space.

The initial step to produce a digital twin from an unlabelled point cloud is to semantically segment the point cloud. Semantic segmentation is assigning every point to a certain, user-defined class with a semantic meaning. From the segmented point cloud, particular objects can be located and extracted to be reconstructed with CAD models. A point cloud typically consists of millions of points, so manual semantic segmentation is infeasible. Recent advances in machine learning have shown great potential in automating tasks like semantic segmentation. The problem is that most methods rely on labelled data to train the models and there are currently no publicly available datasets of labelled point clouds of railway environments.

Because of the high cost of labelling point cloud data and the lack of publicly available labelled railway datasets, using supervised learning to train machine learning models for point cloud semantic segmentation is not a suitable solution. Semi-supervised learning means only a part of the training data is labelled while the remaining part is unlabelled. Semi-supervised methods have already shown to achieve state-of-the-art results in different

machine learning tasks like image segmentation [6, 93, 137], image classification [132, 139, 144] and 3D object detection [136, 141]. These results motivate the research to extend semi-supervised learning to the task of point cloud semantic segmentation.

In this work, two semi-supervised approaches are implemented for the point cloud semantic segmentation task on railway data. One approach combines over-segmentation and active learning and the other is a form of generative few-shot learning. The contribution of this work is applying active learning and generative few-shot learning to point cloud semantic segmentation of railway scenes thereby advancing the research of scene understanding in the context of railways. By training the models on a publicly available dataset, this work establishes a new benchmark for semi-supervised learning in the railway domain. Furthermore, this study determines the applicability of existing techniques to the railway domain and identifies domain-specific challenges.

1.1 Research question

The main research question this research attempts to answer is:

- **How can semi-supervised learning be applied effectively to the semantic segmentation of point cloud scenes of the railway environment?**

Two solutions employing different semi-supervised paradigms are implemented to address this question. The semantic segmentation performance is evaluated on a publicly available point cloud railway dataset. The effectiveness is measured by comparing the performance of the semi-supervised methods with fully supervised methods.

1.2 Limitations

This work focuses specifically on railway data from the Netherlands. However, the railway infrastructure differs considerably per country [113]. Care must therefore be taken with generalising the results gained in this study.

In addition, the semi-supervised learning here consists of exactly two techniques: active learning and generative few-shot learning. There are other semi-supervised techniques like self-supervised learning or embedded learning. Applying these techniques to the problem of point cloud segmentation of railway data could lead to different results compared with this study. Hence, this work is unable to provide a definitive conclusion about the performance of semi-supervised learning in general on point cloud semantic segmentation on railway data.

1.3 Structure

The remainder of this work is organised as follows: in Chapter 2 background information and relevant studies are described. Chapter 3 presents the methodology followed in this study. In Chapter 4 the results are listed. The results are discussed in Chapter 5, where also a conclusion and recommendations for future work are given.

Chapter 2

Background and related work

2.1 Semantic segmentation on images

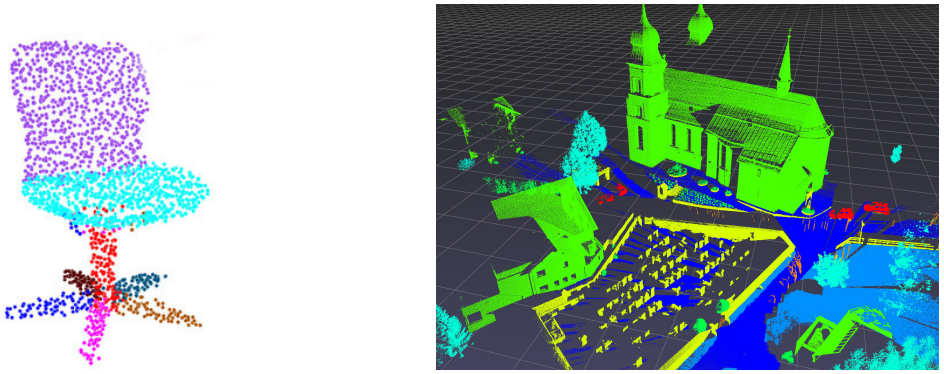
Image semantic segmentation is the task of assigning each pixel in an image with a semantic label. The task is predominantly solved using supervised learning because of the large number of available labelled datasets [86]. The vast majority of models for image semantic segmentation are based on Convolutional Neural Networks (CNN) [65], more specifically Fully Convolutional Networks (FCN) [80]. FCNs take advantage of the well-known image classification models (VGG [112], GoogleNet [115], ResNet [44], AlexNet [60]) but replace the last fully connected layers with convolutional layers. Current state-of-the-art models like SeeThroughNet [43] and InternImage [124] achieve high performance, reaching a mean intersection over union (mIoU) of over 85% on the CityScapes dataset [23].

2.2 Point cloud data

To capture the inherent 3D structure of the railway infrastructure, the data could be stored as point clouds instead of images. A point cloud is a set of points in 3D space where each point is characterised by x-, y- and z-coordinates. Optionally each point has extra attributes like intensity and colour depending on the type of equipment used to gather the data [15].

Point clouds can be obtained from different sources like laser scanners, images and videos. Typically the sensors in laser scanners use LiDAR technique to determine the distances between objects and surfaces and the scanner [41]. Laser scanners have the advantage of producing high-quality output no matter the light conditions as opposed to digital cameras [29]. On the other hand, cameras can supplement the points with colour information. Due to recent developments, LiDAR technology has become more popular, accessible and widely used [129]. Consequently, more and more point cloud data is gathered and made available as public datasets.

Point cloud datasets can be generally divided into two types. Datasets where the point clouds represent a single object like a chair or car (ModelNet [128], ShapeNet [13]) and datasets where the point cloud captures an entire scene (Semantic3D [42], S3DIS [5], SemanticKITTI [7]). The former type of dataset is used frequently for shape classification and part segmentation. The latter is used more for semantic- and instance segmentation. See Figure 2.1 for examples from both types of datasets. These types of datasets are so different from each other that point cloud segmentation models are usually optimised for one of them.



(a) A chair from the ShapeNet dataset. (b) A town scene from the Semantic3D dataset.

Figure 2.1: Examples from an object point cloud dataset (a) and from a scene point cloud dataset (b).

The datasets containing entire scenes can be further divided into indoor and outdoor datasets. Indoor scenes are usually smaller and generally obtained via static scanners like Kinect or from multiple images [77]. Outdoor scenes on the other hand could consist of very large areas and are usually obtained with mobile laser scanners (MLS) or aerial laser scanners (ALS) [30]. Indoor scenes typically have a higher point density and less noise, which ease extracting relevant information from them for machine learning models.

2.3 Point cloud semantic segmentation

In recent years point cloud semantic segmentation has been researched extensively because of its applications in fields such as autonomous driving and robotics [25, 131]. Before machine learning became popular, structural methods were used for segmentation tasks. Structural methods leverage the inherent geometrical properties in point cloud data to segment them. Nowadays, machine learning and in particular deep learning methods are being researched in the literature.

Deep learning methods for point cloud semantic segmentation can be grouped into several categories based on the type of input data: **end-to-end** methods take a raw point cloud as input, **indirect** methods a product derived from a point cloud and **hybrid** methods take combined data from multiple different sources (e.g. point clouds + images) as input [41].

Before the revolutionary PointNet model [99], indirect methods were primarily utilised for point cloud segmentation [142]. Indirect methods transform the point cloud so that convolution operations can be exploited. Projection-based methods project a point cloud to a representation in 2D space and usually process them with 2D convolutional models. Discretisation-based methods transform the point cloud into a 3D regular structure like a voxel cloud, octree or kd-tree. The major disadvantage of indirect methods is that inevitably information is lost during the transformation.

The PointNet model [99] meant a significant breakthrough for deep learning on point cloud data since it was the first to take raw point clouds as input. PointNet captures the global structure of point clouds as a feature vector. The authors built upon PointNet to design the PointNet++ model [100] which also captures local structures. These methods proved inspirational for future research and the underlying principle was used as a basis for numerous other papers. Today, a legion of end-to-end models exists ranging from multi-

layer perceptrons (MLP) [48, 98, 148], pointwise convolutions [71, 85, 145], recurrent networks [51, 135, 149] to graph based models [64, 117, 126] among others.

Hybrid methods can draw upon more information than end-to-end methods. Unfortunately, there is seldom additional data available. And if it is available, a way has to be found to integrate the different data inputs. Therefore, the resulting hybrid models are generally larger than non-hybrid models and suffer from long run-times [52] and/or a vast number of trainable parameters [27].

The performance of current state-of-the-art point cloud segmentation models is not yet up to par with the state-of-the-art in image semantic segmentation. The 2DPASS model [133] currently achieves the best performance on the SemanticKITTI dataset with a mIoU of 72.9[1] and on Semantic3D ConvPoint [9] achieves 77.7 mIoU[2]. A reason for the performance difference is that point cloud data exhibits certain characteristics that make it more difficult to work with than images.

2.4 Challenges of point cloud data

At first sight, it may seem straightforward to transfer the knowledge of image semantic segmentation to point cloud semantic segmentation. Unfortunately, point cloud semantic segmentation proves to be considerably more difficult because of the nature of point cloud data [41].

Point clouds are irregular, unstructured and unordered, unlike 2D images, and are thus a challenging data type to work with [8]. The following is a list of the most significant challenging characteristics that are inherent to point cloud data. Sensor type, environment, weather conditions and sensing distance influence the degree to which point clouds suffer from these characteristics [72]:

- **Irregularity**: point clouds usually have non-uniform distributed point density.
- **Unstructured**: point clouds are not placed on a regular grid. Each point is scanned independently and its distance to neighbouring points is not fixed.
- **Unordered**: a point cloud is a set of points, the order in which the points are stored does not change the representation.
- **Size**: point clouds often contain millions of points taking up large chunks of memory and thus it is time-consuming to process and analyse them.
- **Noisy**: point clouds can contain noise in the data produced for example by errors of the scanner or moving objects [92].
- **(Partial) Occlusion**: point clouds suffer from (partial) occlusion of objects since other objects may block them [40].

An extra challenge for railway scenes is the large variance in object sizes (a top bar can be well over 20 metres long, while an insulator typically is around 30 centimetres [119]) and frequency of objects in the dataset resulting in a significant class imbalance in the dataset.

There are currently no publicly available large-scale, labelled datasets of point clouds of the railway environment. The datasets that are publicly available contain a limited number

[1] `http://semantic-kitti.org/tasks.html#semseg`
[2] `http://semantic3d.net/view_results.php?chl=1`

of samples [118] or the annotations encompass only a few categories (cable + rail track) [24]. However, point cloud segmentation of railway environments is actively researched usually constituting a case study with custom data [39, 63, 66, 73].

2.5 Semi-supervised learning

With the lack of abundant available labelled data and the high cost of manual labelling, supervised learning is not a feasible solution for point cloud segmentation on railway data. Semi-supervised learning is learning with a training dataset consisting of both labelled and unlabelled data where typically the labelled dataset is smaller [121]. Semi-supervised learning can be categorised into different paradigms based on their underlying concepts. With self-supervised learning, the model is pre-trained with a pretext task to learn one part of the input from another part of the input whereby the labels are auto-generated [58]. Examples of pretext tasks are next-word prediction or rotation prediction of images. The knowledge gained from this pre-training can then be used in downstream tasks like segmentation. Active learning and few-shot learning will be explained briefly in the next sections.

Active learning

Active learning is a subfield of machine learning consisting of methods that can query data samples from a pool of unlabelled data to be labelled by an oracle (often a human annotator) [109]. The fundamental belief behind the active learning concept is that a model could potentially reach a higher level of accuracy while using a smaller number of training samples if it were allowed to choose the data it wants to learn from [20]. The model can employ different query scenarios. The most prevalent are stream-based selective sampling, pool-based sampling and membership query synthesis [108]. **Stream-based selective sampling** draws one unlabelled sample at a time and the model must decide whether to query the sample or not. In the **pool-based sampling** scenario, the model attempts to evaluate the entire dataset before it selects the best query or set of queries according to some informativeness measure. Lastly, with **membership query synthesis** the model is allowed to generate its own synthetic queries to be sent to the oracle.

The query strategy evaluates the informativeness of unlabelled samples. Many different strategies are developed in the literature and they can be broadly classified into the following categories [108]:

- **Uncertainty sampling**: Query the samples whose label the model is most uncertain about.
- **Query-by-committee (QBC)**: Query the samples for which the committee (set of models) most disagree.
- **Expected model change**: Query the samples that result in the greatest change of the model if their labels are known.
- **Variance reduction**: Query the samples that minimise the variance of the future error of the model.
- **Estimated error reduction**: Query the samples that minimise the expected future error of the model.

- **Density-weighted methods**: Query the samples that are most informative and are representative of other samples.

Both uncertainty sampling and QBC are likely to query outliers since they are often the most uncertain samples. Querying outliers will not result in more knowledge of the underlying data distribution for the model. The main drawback of the expected model change, variance reduction and estimated error reduction is their computational cost since they calculate features over the whole model instead of over individual samples.

Few-shot learning

Few-shot learning (FSL) is a machine learning paradigm where the model output is based on only a few labelled training samples. Stricter variants of FSL are one-shot learning and zero-shot learning, where the model respectively uses only a single example or even no examples at all. FSL techniques achieve good performance by making use of prior knowledge about the data, model and/or optimisation strategy [125]. There are plenty of different techniques to exploit this prior knowledge [95]. Generative FSL techniques generate more labelled training samples with prior knowledge about the data for example by manual augmentation [76, 111], learned augmentation [46, 62, 78, 143] or using a Generative Adversarial Network [38] (GAN) [17, 84, 87]. Metric learning or embedded learning embeds each sample in a lower dimension such that similar samples are close together while dissimilar samples are far apart as calculated by a distance function [122, 147]. With multitask learning a few related tasks are learned simultaneously exploiting task-generic and task-specific information [10, 50]. Another approach is to enhance the models to better suit them for FSL tasks. Memory-augmented neural networks [106] and memory-matching networks [12] contain memory components in the model while meta networks [88] and CSN [89] use techniques to rapidly adapt parameters. Yet another method is to modify the optimisation strategy to adapt models for FSL [36, 114].

2.6 Related work

In the literature, several studies apply semi-supervised learning techniques to tasks with point cloud data. The work of [127] employs active learning with diversity-aware selection, while [110] utilises superpoints as query units in the active learning loops. In [120] generative few-shot learning on railway crossings is researched. In [90] point cloud data is generated from virtual railway scenes in a game engine to perform landmark detection. However, none of these studies specifically focus on segmenting the railway infrastructure.

The student-teacher model in [53] is trained using self-supervised learning. Embedded learning is used in [32] to segment railway data into six classes. These works focus specifically on segmenting the railway infrastructure. The novelty in our work lies in applying active learning and generative few-shot learning to segment point clouds of the railway infrastructure. To the best of our knowledge, these semi-supervised techniques have not been applied to the segmentation task on point cloud data of the railway.

Chapter 3

Methodology

Two different approaches are taken to the task of point cloud semantic segmentation. Both employ semi-supervised learning to minimise the manual labelling effort. One approach leverages the active learning paradigm while the other is a form of generative few-shot learning.

An overview of the active learning approach can be seen in Figure 3.1. First, the catenary arches are split up into boxes (1). Then, superpoints are generated by applying an over-segmentation method to the boxes (2). Finally, the active learning phase is started by training a segmentation model on a small portion of labelled data (3). With the trained model, the most informative superpoints are selected (4), labelled and added to the training set (5). When the annotation budget for the round is reached, the model is trained on the new training set before the next active round starts.

The main advantage of this approach is a reduction in labelling effort by both using superpoints as query units as well as labelling only the most informative ones. Per-point labelling will still result in millions of queries despite only a small percentage of the full dataset being labelled. A disadvantage is that the over-segmentation method is not perfect and will produce superpoints that overlap object boundaries. These superpoints will cause incorrect labels when annotated. Moreover, the method relies on the correctness of the oracle, but this can not be guaranteed since, in practice, the oracle is often human. Another drawback is that the training loop is interactive and thus more time-consuming than non-interactive training loops.

A schematic overview of the generative few-shot learning approach is visible in Figure 3.2. The first step is to extract individual objects from the labelled dataset (1). Synthetic arches are created following object placement rules (2). The synthetic data is used to train a model using supervised learning (3).

The primary benefit of the few-shot learning approach is that supervised learning can be exploited. A serious drawback is that the generated data is specific to the used dataset and domain knowledge is required to construct the object placement rules.

These two approaches were chosen because of their different benefits and because they were feasible to implement within the time constraint. Active learning has proven to achieve good performance for point cloud segmentation on shapes [110] and indoor scenes [127], so it is reasonable to extend it to railway data. Generative few-shot learning also shows promising results for point cloud segmentation on both outdoor scenes [68] and level crossings [120].

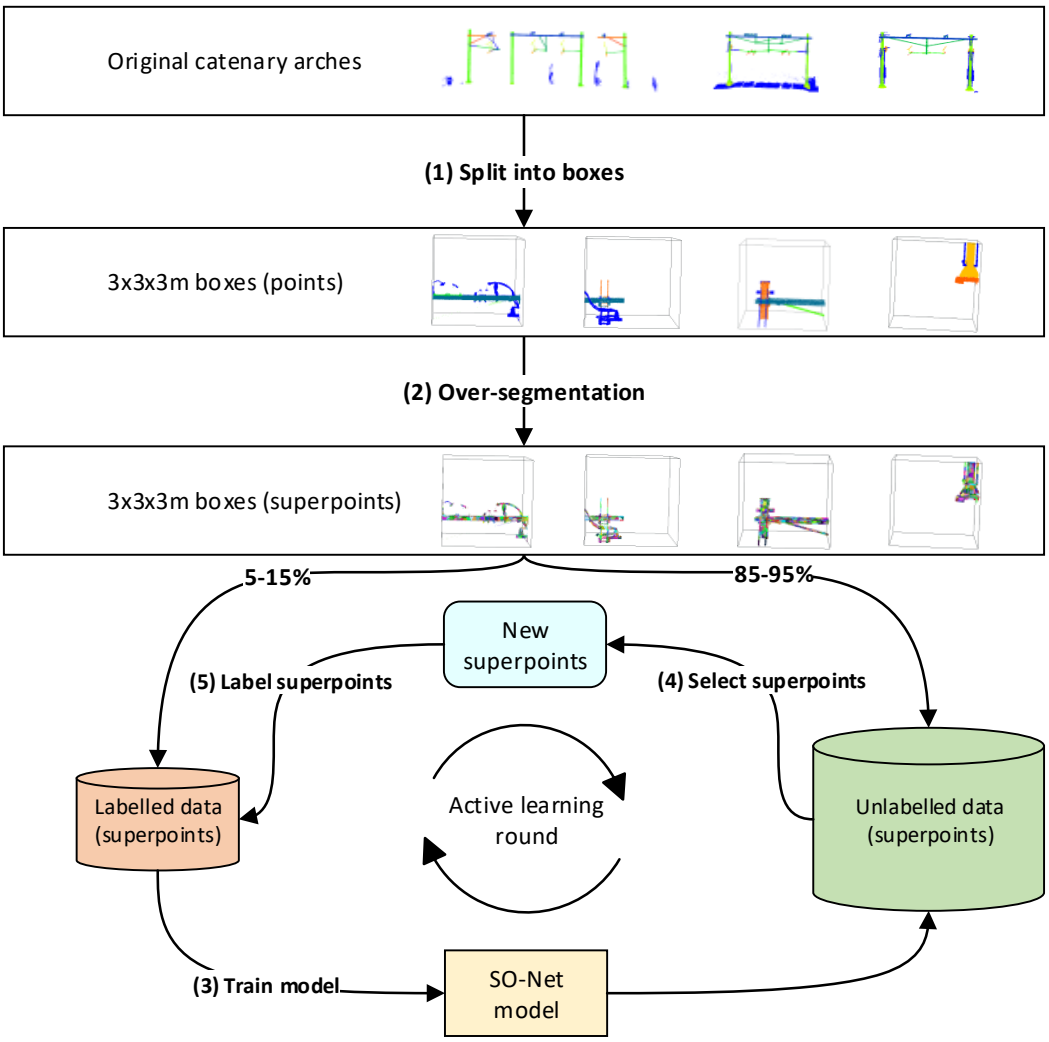


Figure 3.1: A schematic overview of the active learning approach. First, the original arches are split into boxes (1). Superpoints are generated by applying an over-segmentation method on the boxes (2). After the SO-Net segmentation model is trained on a small portion of labelled data (3), the active learning rounds begin where repeatedly superpoints are selected (4), labelled (5) and added to the training data.

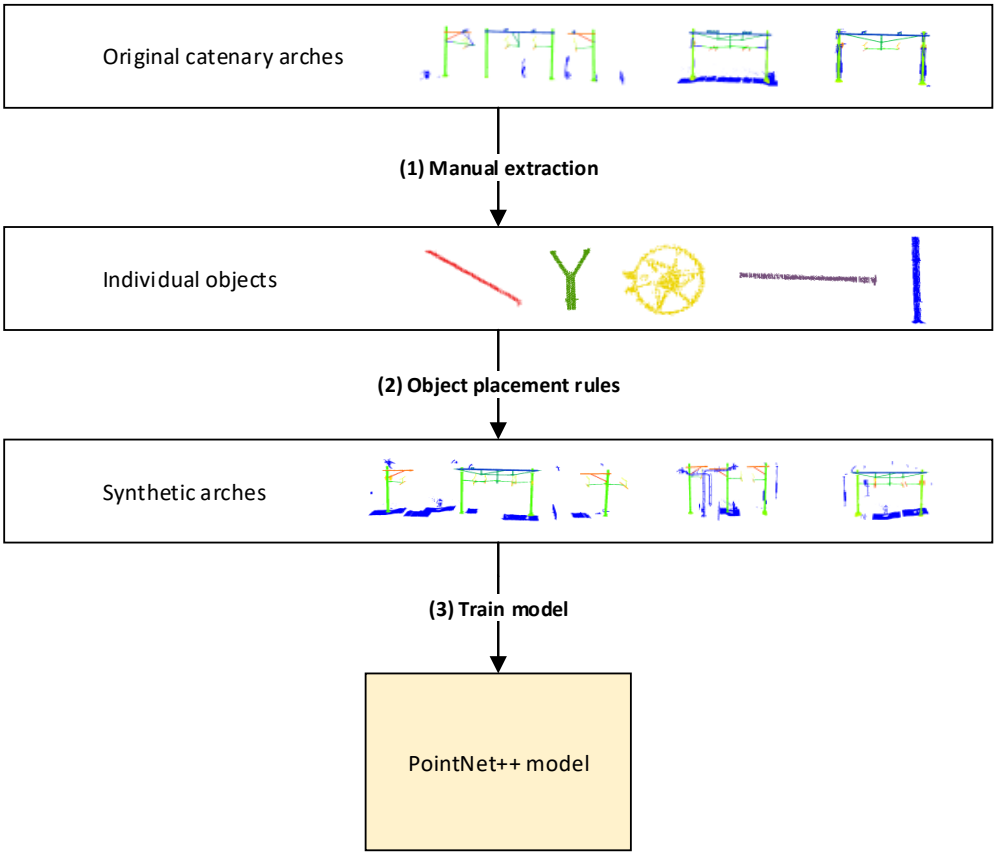


Figure 3.2: A schematic overview of the generative few-shot learning approach. The first step of the generative FSL approach is to manually extract individual objects from the original catenary arches (1). Then, following object placement rules, a labelled synthetic dataset is created (2). Finally, a PointNet++ model is trained on the synthetic data using supervised learning (3).

3.1 Dataset

The dataset used in this study contains catenary arches from railways in the Netherlands [118]. This dataset contains 15 high-resolution point clouds of catenary arches. The number of points per arch ranges from around 1.6 million to 11 million. The dataset is labelled into 14 different classes. An example of an arch is presented in Figure 3.3. The advantage of using this fully labelled dataset is that the implemented semi-supervised methods can be compared to supervised methods.

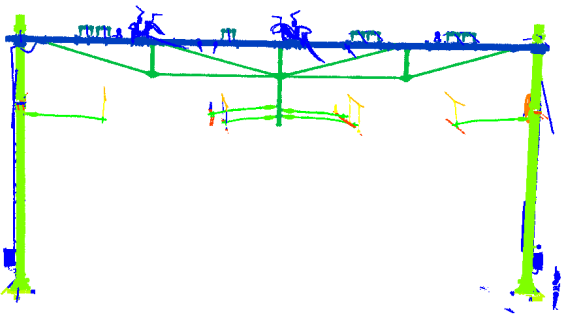


Figure 3.3: Example of an original, unprocessed arch from the catenary dataset [118] visualised by CloudCompare [19]. Different colours indicate different labels.

3.1.1 Preprocessing

Removing duplicate points

It was found that the dataset contained 33.039.193 duplicate points. Two points are considered duplicates if their x-, y-, and z-coordinate are identical. The coordinates in the dataset have a precision of five decimals. The duplicated points are probably due to the fact that the original laser scanner had a higher precision, but the coordinates got rounded down to five decimal places. The duplicated points were removed via CloudCompare software [19].

Transforming labels

In the original dataset, the labels range from 0 to 16, where 0 is the unlabelled category. Label 12 is not present in the dataset. Labels 11 and 13 originally represented tension rods and tension rod foundations respectively, but are regarded as unlabelled objects in this study. Consequently, their labels are set to 0 following the approach in [119] to ensure a fair comparison with their results. The remaining 14 labels are transformed to a consecutive range from 0 to 13 to ease processing later on. The final label scheme is listed in Table 3.1.

3.2 Active learning approach

The idea behind the active learning approach is to select only the most informative data samples for labelling. This way the manual labelling effort is reduced while the model is still able to achieve relatively high performance. The manual labelling effort is further decreased by selecting superpoints instead of points.

Label	Class	Label	Class
0	Unlabelled	7	Pole foundation
1	Top bar	8	Dropper
2	Messenger wire support	9	Stitch wire
3	Drop post	10	Wheel tension device
4	Steady arm	11	Contact wire
5	Insulator	12	Top tie
6	Pole	13	Bracket

Table 3.1: The 14 label ids and their corresponding object class as used in this study.

3.2.1 Dividing into boxes

The samples in the catenary arches dataset [118] are not perpendicular to the x- and y-axes and are also not centred around the origin (0,0,0). As a first preprocessing step, all arches are rotated along the z-axis, so that the poles are aligned with the x-axis and afterwards translated to the origin. The pole alignment algorithm is based on the fact that the poles are (approximately) on a straight line (see Algorithm 1). The arches are translated to the origin with the formula in (3.1) separately for every dimension (x,y,z), $\hat{p}$ is the translated point and p is the original point.

$$\hat{p} = p - d_{min} - (\frac{1}{2} \cdot (d_{max} - d_{min})), \text{where } d \text{ is the dimension (x,y,z)} \tag{3.1}$$

Algorithm 1 Aligning the poles of point clouds to x-axis

Input: *pole_points* = List of all pole points
Output: Rotation angle in radians over the z-axis to align poles with x-axis

$avg\_x \leftarrow$ average x-coordinate of *pole_points*
positive_points $\leftarrow$ all pole points with x-coordinate $> avg\_x$
negative_points $\leftarrow$ all pole points with x-coordinate $< avg\_x$

$pos\_avg\_x \leftarrow$ average x-coordinate of *positive_points*
$neg\_avg\_x \leftarrow$ average x-coordinate of *negative_points*
$pos\_avg\_y \leftarrow$ average y-coordinate of *positive_points*
$neg\_avg\_y \leftarrow$ average y-coordinate of *negative_points*

$x\_len \leftarrow \text{abs}(pos\_avg\_x - neg\_avg\_x)$
$y\_len \leftarrow \text{abs}(pos\_avg\_y - neg\_avg\_y)$

return $-tan^{-1}\ \frac{y\_len}{x\_len}$

Next, surface normals for each point in the point clouds are calculated with Point Data Abstraction Library (PDAL) [22] and stored in a compressed .LAZ file. The normals, together with the coordinates, are input to the segmentation model. PDAL estimates the normals based on the k nearest neighbours for each point. The k parameter was set to 25.

The average size of arches in the original dataset is 19.5x4.5x9.2m. The catenary arches are split up into boxes of 3x3x3 metres for several reasons: to get more data samples, to better fit the SO-Net model and to lower memory utilisation per batch. The specific

	Number of points	**Point density** (p/m^3)	**Volume** (m^3)
Minimum	529	44	0
Maximum	531.598	702.161	27
Average	42.990	40.556	5

Table 3.2: Statistics for the number of points, point density and volume of the bounding box of the points inside the box. Note: A volume of zero is due to all points lying on a plane.

size is a trade-off between the number of data samples and preserving spatial properties of the objects. Smaller boxes result in more objects being split between multiple boxes compromising the spatial structure of the objects. The left-bottom corners of the boxes are determined by applying the following formulas to all three dimensions (x,y,z), where d_{size} is the size of the boxes along dimension d:

$$start = d_{min} - (\frac{1}{2} \cdot remainder), \text{where } remainder = d_{size} - (length \bmod d_{size})$$

$$stop = start + num\_boxes \times d_{size}, \text{where } num\_boxes = \left\lceil \frac{length}{d_{size}} \right\rceil$$

By using the remainder, the boxes are equally distributed at the edges of the point cloud as can be seen in Figure 3.4 (c) and (d). If the start position is instead chosen to be simply the minimum of the dimension, only the boxes at the other end will exceed the bounding box of the arch and contain empty space as can be seen in Figure 3.4 (a) and (b).

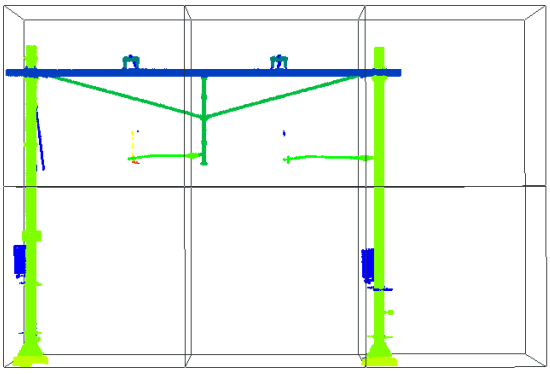
A downside of splitting the arches into boxes is that some spatial information is lost because objects are divided between boxes. The loss of information could harm the performance of both the over-segmentation and the semantic segmentation task. On the other hand, the processing of these smaller point clouds is more efficient and less information is lost at down-sampling.

Additionally, the .LAZ files are reformatted to .XYZ files containing only coordinates and labels where each line represents a point. This is necessary since the C++ implementations of the over-segmentation methods do not have the functionality to read .LAZ files.

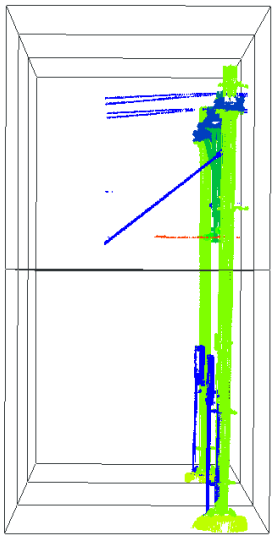
Dataset statistics

The 3x3x3m boxes dataset consists of 22.354.689 points distributed over 520 boxes. Boxes containing less than 512 points are excluded because it is deemed that they contain insufficient points to represent a meaningful representation of railway objects. Table 3.2 shows statistics about the number of points, point density and volume per box. Here, volume is the volume of the bounding box of the points and is calculated with (3.2). Table 3.3 shows the percentage of points for every object class as well as the percentage of boxes containing points of the class. There is a large class imbalance.

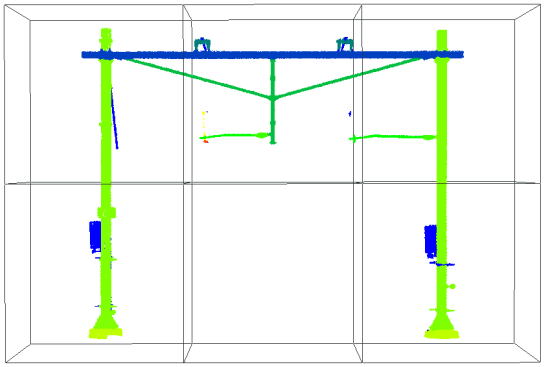
$$volume = (x\_max - x\_min) \times (y\_max - y\_min) \times (z\_max - z\_min) \tag{3.2}$$



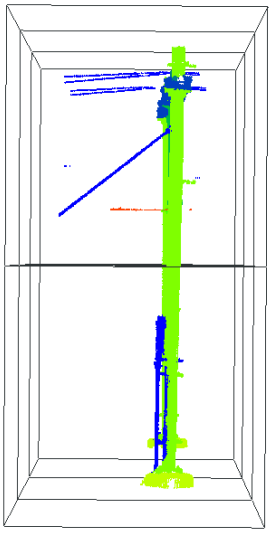
(a) Front-view of the division in boxes without taking remainder into account.



(b) Side-view of the division in boxes without taking remainder into account.



(c) Front-view of the division in boxes when taking remainder into account.



(d) Side-view of the division in boxes when taking remainder into account.

Figure 3.4: Boxes calculated without taking remainder into account (a)-(b) and with taking the remainder into account (c)-(d).

Class	% points	% box	Class	% points	% box
Pole	40.45	44.62	Catenary wire	1.53	19.81
Unlabelled	27.28	79.91	Messenger wire support	1.51	16.54
Top bar	13.98	23.46	Bracket	0.91	3.85
Drop post	4.34	24.04	Insulator	0.80	14.81
Pole foundation	4.19	15.0	Stitch wire	0.41	16.54
Top tie	2.35	5.19	Dropper	0.24	16.25
Steady arm	1.79	20.77	Wheel tension device	0.23	1.73

Table 3.3: The percentage of points per class in the 3x3x3 boxes dataset sorted by descending percentage of points.

3.2.2 Over-segmentation

Unsupervised over-segmentation is commonly used as a preprocessing step for tasks like clustering and semantic segmentation [94]. The output of over-segmentation are regions that are perceptually similar called superpixels (2D) [102] or supervoxels/superpoints (3D) [131]. The similarity is usually based on a combination of spatial, colour and geometric features. An important characteristic of the resulting superpoints is that object boundaries should be preserved because the error of non-preserved object boundaries will propagate through to the segmentation. Additionally, it is also convenient if the superpoints have a regular shape for further processing [130]. In this research, superpoints are used as the units queried to the oracle in the active learning phase.

The performance of over-segmentation methods can be measured by different metrics. The three most common metrics in the literature are boundary recall, under-segmentation error and global consistency error. Originally, they were developed to measure the performance of 2D over-segmentation but they can be easily extended to the 3D case.

Boundary recall [91] is defined as the fraction of ground truth edges that fall within a certain distance d of at least one superpoint boundary point. A boundary point is a point for which any of its k nearest neighbours has a different label than itself ($k = 50$ in this work). The definition of boundary recall as used in this research is stated in (3.3). Here, TP is the number of boundary points in the ground truth point cloud for which exists a superpoint boundary point in range d while FN is the number of boundary points in the ground truth point cloud for which does not exist a superpoint boundary point in range d. In this work, d is set to 0.03m.

$$Boundary\ recall = \frac{TP}{TP + FN} \tag{3.3}$$

Under-segmentation error [67] measures to what extent superpoints overflow the ground truth segment borders. It is based on the observation that a ground truth segment divides a superpoint into an *out* and *in* part. The implementation of under-segmentation error as used here is defined in (3.4), where GT is the set of all ground truth segments, s_{GT} is a ground-truth segment and sp is a superpoint.

$$Under\text{-}segmentation\ error = \sum_{s_{GT} \in GT} \frac{\sum_{sp:sp \cap s_{GT} \neq \emptyset} |sp_{out}|}{|s_{GT}|} \tag{3.4}$$

Global Consistency Error (GCE) [83] is a metric that simultaneously evaluates over-segmentation error and under-segmentation error based on the intersection of superpoints and the ground truth segments. GCE is regularised to range [0, 1], where

Method	Boundary recall	Under-segmentation error	GCE
BPSS	0.922	**0.099**	0.095
VCCS	0.924	1.158	**0.070**
VCCS-kNN	0.962	0.210	0.109
GPU-SS	**0.994**	0.239	0.217

Table 3.4: The performance of the different over-segmentation methods using a seed resolution of 0.15m and a voxel resolution of 0.01m on the subset of catenary arches. The best scores are indicated by **bold** text.

0 indicates no error and 1 indicates the worst possible segmentation. The definition used in this study can be found in (3.5), where $P_{ij} = (1 - \frac{|s_{GTi} \cap sp_j|}{|s_{GTi}|}) \times |s_{GTi} \cap sp_j|$ and $Q_{ij} = (1 - \frac{|sp_i \cap s_{GTj}|}{|sp_i|}) \times |sp_i \cap s_{GTj}|$. M is the number of ground truth segments, N is the number of superpoints, s_{GTi} is ground truth segment i and sp_i is superpoint i.

$$GCE = \frac{1}{\sum_i^M \sum_j^N |s_{GTi} \cap s_j|} min(\sum_i^M \sum_j^N P_{ij}, \sum_i^N \sum_j^M Q_{ij}) \tag{3.5}$$

Besides these metrics, the number of generated superpoints is also an important characteristic. The number of superpoints should be significantly less than the number of points in the dataset. In the worst case, the over-segmentation could lead to every superpoint containing exactly one point. This will be a perfect over-segmentation according to the metrics discussed above, but neglects the goal of over-segmentation namely to reduce the number of units to work with. In practice, the best over-segmentation generates as large as possible superpoints while still attaining high values on the metrics.

The different over-segmentation methods tested in this study are Voxel Cloud Connectivity Segmentation (VCCS) [94], Boundary Preserving Supervoxel Segmentation (BPSS) [75] and GPU Supervoxel Segmentation (GPU-SS) [31] as well as a variant of VCCS, called VCCS-kNN. VCCS-kNN works on points instead of voxels. Existing C++ implementations[1,2] of these methods were applied to the boxes dataset. To quantitatively measure the performance the three metrics as described above were implemented.

A subset of the arches dataset is used to speed up the testing. The subset consists of all boxes from arches 01_01, 02_02, 03_01, 03_03 and 04_04 totalling 4.737.663 points contained in 174 boxes. These arches are quite different from each other so the performance on this subset will be a good indicator for the whole dataset.

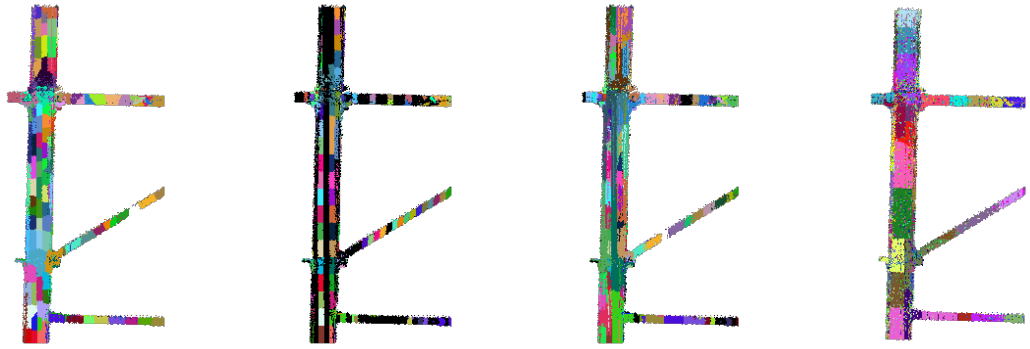
All methods have a seed resolution which determines the size of the (initial) superpoints. Additionally, the methods using voxels instead of points (VCCS and GPU-SS) also have a voxel resolution which determines the size of the voxels. The seed resolution was set to 0.15m and the voxel resolution to 0.01m, as determined by exploratory research. The results for all methods can be seen in Table 3.4. The number of superpoints generated varied per box and method, but is approximately 250 for this configuration. A visualisation of the results is presented in Figure 3.5.

The BPSS method is selected to generate superpoints for the active learning phase as it has good metric scores and relatively regular shaped superpoints (see Figure 3.5 (a)).

To get superpoints for all boxes, the BPSS method was executed on the full dataset of 520 boxes. The over-segmentation performance of the BPSS method is listed in Table 3.5.

[1]BPSS, VCCS, VCCS-kNN: `https://github.com/yblin/Supervoxel-for-3D-point-clouds/tree/master`

[2]GPU-SS: `https://github.com/dongxiao0401/GPUSupervoxelForPointCloud/tree/main`



(a) BPSS. (b) VCCS (c) VCCS-kNN. (d) GPU-SS.

Figure 3.5: The generated regions from the different over-segmentation methods for the same input data using a seed resolution of 0.15m and a voxel resolution of 0.01m.

Boundary recall	**Under-segmentation error**	**GCE**
0.920	0.117	0.109

Table 3.5: Over-segmentation performance of the BPSS method on the full 3x3x3 boxes dataset.

The total number of superpoints generated is 83.503. Compared to the 22.360.815 total points, this is a reduction of more than 99% in terms of units. The distribution of superpoint sizes can be viewed in Figure 3.6, the average size is approximately 268 points. There is a substantial number of superpoints containing no more than six points, probably due to noise and/or outliers.

The computed superpoint label for each point is added to the .XYZ file and via a Python script added to the .LAZ file of the boxes. The attributes of the .LAZ files of the 3x3x3 boxes are shown in Table 3.6.

3.2.3 Active learning loop

The semantic segmentation is performed by the SO-Net model [69] since it achieved good performance on point cloud segmentation. The encoder of this network consists of a self-organising map (SOM) [59], fully connected layers and max pooling operations. The produced global feature is fed into the segmentation network along with the normalised point features and SOM node features to produce segmentation scores for all input points.

Before the active learning rounds are executed, the segmentation model is pre-trained on a small portion of the data to get a somewhat meaningful segmentation result that the first query selection can be based on. The small portion of initial supervised training data has a significant influence on the performance of the active learning rounds [49, 138]. To

Name	**Description**
xyz	Array (n_points, 3) with x-, y- and z-coordinates of all points
classification	Array (n_points) with the ground-truth label as int for all points (0-13)
normals	Array (n_points, 3) with the normal vector of all points
superpoint	Array (n_points) with the superpoint label for each point

Table 3.6: The attributes stored in the .LAZ files of box data.

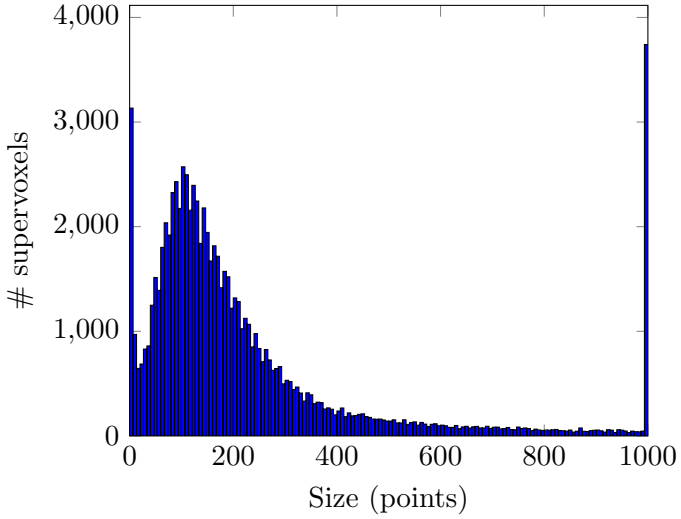


Figure 3.6: Histogram showing the size distribution of the superpoints where size is defined as the number of points they consist of. The right-most bin contains all superpoints with 994 or more points.

ensure a fair comparison between the active learning experiments, the initial training data and the validation data are fixed by randomly splitting the data with a fixed seed.

The query strategy used in this research combines uncertainty, feature diversity, class diversity as well as location diversity for selecting the most informative superpoints. Margin-based uncertainty (also called best-versus-second-best (BVSB) measure) is used as an uncertainty metric since it better captures the model confusion than entropy-based measures [56]. The uncertainty (U) for a superpoint is the sum of the margin-based uncertainty for each point divided by the number of points in the superpoint. The formula for uncertainty of a superpoint sp can be seen in (3.6), where pt_{θ_1} is the highest probability and pt_{θ_2} is the second highest probability for point pt.

$$U(sp) = \frac{1}{points\ in\ sp} \sum_{pt \in sp} 1 - (pt_{\theta_1} - pt_{\theta_2}) \tag{3.6}$$

The feature vector of a superpoint is the normalised average of all point features of the points constituting the superpoint. A point feature is a 128-dimensional vector from the second-to-last layer of the SO-Net segmentation model. The feature diversity (FD) of a superpoint sp is the minimum distance between the feature vector of sp and the feature vectors of all other unlabelled superpoints (3.7), where S is the set of unlabelled superpoints. The class diversity (CD) of a superpoint is determined by comparing its feature vector to the feature vectors of the classes. The feature vector for a class c is the normalised average of all feature vectors of superpoints that have label c. The class diversity is the minimum distance between the feature vector and class feature vectors (3.8), where C is the set of all classes.

$$FD(sp) = min(\|\vec{\boldsymbol{sp}}_{feature} - \vec{\boldsymbol{s}}_{feature}\| \text{ for } s \in S), \text{ where } \|\cdot\| \text{ is the } L^2 \text{ norm} \tag{3.7}$$

$$CD(sp) = min(\|\vec{\boldsymbol{sp}}_{feature} - \vec{\boldsymbol{c}}_{feature}\| \text{ for } c \in C), \text{ where } \|\cdot\| \text{ is the } L^2 \text{ norm} \tag{3.8}$$

The formula for the informativeness (I) of a superpoint is given in (3.9), where α, β and γ are user-defined parameters.

$$I(sp) = \alpha \cdot U(sp) + \beta \cdot FD(sp) + \gamma \cdot CD(sp) \tag{3.9}$$

The location diversity is based on the box the superpoint belongs to. To select diverse superpoints the diversity-aware selection procedure as described in [127] is implemented. A superpoint with a lower informativeness score than another superpoint belonging to the same box is penalised. This way the query selector is stimulated to select superpoints from different locations in the catenary arches. The received penalty is dependent on the decay rate and the number of superpoints from the same box with higher informativeness. The adjusted informativeness (I_a) for a superpoint sp is calculated with (3.10), where n is the number of superpoints with higher informativeness belonging to the same box as sp. The algorithm that selects the superpoints for labelling is listed in Algorithm 2.

$$I_a(sp) = I(sp) \cdot decay\_rate^n, \text{ where } 0 < decay\_rate < 1 \tag{3.10}$$

Algorithm 2 Selecting the most informative superpoints for an active learning round

Input: *superpoints* = List of all superpoints,
budget = the maximum number of points that may be selected each round
Output: List of superpoints selected for labelling

```
sort superpoints on adjusted informativeness in descending order

selected_superpoints = empty list
index = 0

while budget > 0 do
    current superpoint = superpoints[index]

    if budget ≥ number of points in current superpoint then
        Add current superpoint to selected_superpoints
    else if budget < 50 then
        break
    end if

    index++
end while

return selected_superpoints
```

It is important to note that in our case, each superpoint is annotated by giving all points the same label, contrary to [127], where points in a single superpoint can be assigned to different objects. This difference in annotation method influences the query strategy: while intra-variety is important, in our case inter-variety between superpoints holds greater significance.

Each active learning round has an annotation budget. The model queries as many superpoints as possible to the oracle until the annotation budget is reached. The newly annotated superpoints are added to the training set. The model parameters can then be

reset for retraining or preserved for fine-tuning. In this study, the model is fine-tuned after every active learning round since it is less time-consuming.

To compare the effectiveness of the specialised query selection algorithm, it is compared against a random query selection strategy. This strategy involves selecting superpoints at random for labelling and serves as a baseline for the performance of active learning methods.

The active learning phase can be terminated once the model reaches the desired performance. However, the model might need a significant amount of labelled data to attain this performance. For this reason, the number of active learning rounds for the experiments is fixed at seven.

3.3 Few-shot learning approach

The type of few-shot learning used in this approach is generative FSL [95]. Synthetic, labelled data is created based on the small portion of labelled data that is available. This way the training set is expanded and the segmentation can be executed on state-of-the-art supervised point cloud semantic segmentation models.

3.3.1 Generating synthetic data

There are several strategies for generating synthetic data (see Section 2.5). Applying manual rules to generate the synthetic data requires domain knowledge and produces rules tailored for one specific dataset. Moreover, the generated rules probably will not cover all possible variations [125]. Learning augmentations via machine learning models suffer less from these characteristics. However, implementing these models is not straightforward.

In this study, the labelled training set is expanded by following object placement rules. This is inspired by [120], where the objects that make up a level crossing are assembled to generate the synthetic data. The objects are sampled from real point cloud data instead of from 3D object models. Using point cloud samples ensures a better resemblance to real point cloud data. For example, [82] used volumetric point clouds derived from 3D models while real point cloud data is not volumetric. Using object placement rules was deemed appropriate for the catenary arches dataset since it exhibits many geometrical properties that can be captured in rules. For example, the top bar is always perpendicular to the poles and stitch wires are always directly above catenary wires.

The first step was to split up every arch into 14 point clouds containing points belonging to the same object class. These point clouds were then manually split into individual objects. The individual objects were translated to the origin and oriented similarly. Table 3.7 lists the number of individual objects obtained per class.

The unlabelled category was split into three different object categories (ground, pole-like and wire-like objects) to construct different object placement rules per category. Furthermore, only whole unlabelled objects are extracted and not individual noisy points. Although some extracted objects were incomplete (see Figure 3.7). This can be due to occlusion, too low precision of the scanner or imprecise labelling. These partial objects were left in the object pool to increase robustness and simulate real-world data.

The object placement rules are listed in Table A.1 in Appendix A. These rules are translated into a Python script to generate the new scenes. For all objects, randomness is added to their final positions, ranging from a few centimetres to a maximum of two metres. This randomness is added separately per dimension. The amount of randomness depends on the dimension and type of object. Unlabelled objects, poles and top bars have

(a) Catenary wire. (b) Pole foundation.

Figure 3.7: Some extracted point cloud samples from real point clouds are incomplete due to occlusion, too low precision or imprecise labelling.

Label	Category	Count	Label	Category	Count
0	Unlabelled (ground)	37	6	Pole	40
0	Unlabelled (pole-like)	34	7	Pole foundation	39
0	Unlabelled (wire-like)	29	8	Dropper	54
1	Top bar	14	9	Stitch wire	40
2	Messenger wire support	62	10	Wheel tension device	8
3	Drop post	16	11	Catenary wire	63
4	Steady arm	64	12	Top tie	8
5	Insulator	64	13	Bracket	8

Table 3.7: The number of extracted, individual objects from the original arches per class. Note: The unlabelled category is split up into three separate categories.

relatively large randomness. Smaller objects and objects that need to be placed a certain way (e.g. bracket should be between pole and top tie) have relatively small randomness.

A dataset containing 750 synthetic catenary arches totalling 533.416.024 points was created. Statistics about the number of points, point density and volume are listed in Table 3.8. The point density of the synthetic- and original arches are comparable, though, the synthetic arches are smaller in terms of both volume and number of points. Table 3.9 shows the distribution of points across each object class along with the percentage of arches containing points of that class. The presence of classes in the arches is more uniform in the synthetic data than in the original data. Only the messenger wire support, and to a lesser extent top bar and drop post, are present in fewer arches. The distribution of points is similar and thus the synthetic dataset has a large class imbalance as well. Although the wheel tension device and unlabelled classes have significantly more points in the synthetic dataset, the pole foundation and messenger wire support contain notably fewer points. Figure 3.8 shows examples of the generated synthetic arches.

The synthetic dataset is not split up into boxes like in the active learning approach because boxes lose spatial information about objects. Moreover, as many synthetic scenes as needed can be generated with the object placement rules, so an insufficient number of training samples is not an issue.

3.3.2 Class imbalance

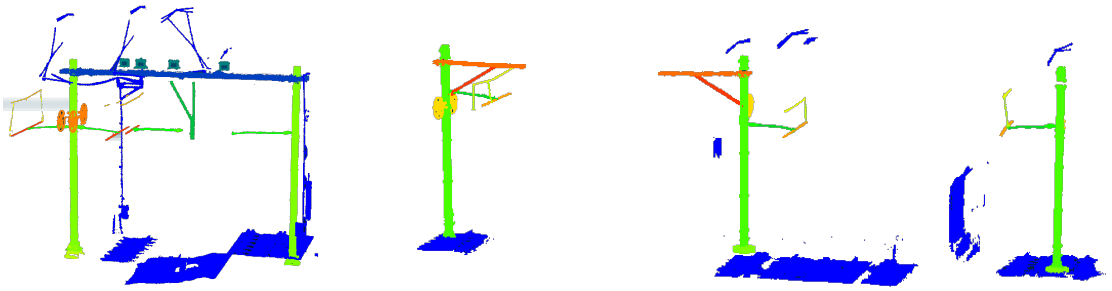
As can be seen in Table 3.9, the dataset is highly imbalanced. The unlabelled category consists of 45% of the total number of points, while five classes consist of less than 1% each. It is more challenging for the model to learn about and correctly predict the under-represented classes since it has less data to learn from and the learning process is dominated

	Number of points	**Point density** (p/m^3)	**Volume** (m^3)
Minimum	190.217 (*283.172*)	605 (*604*)	97 (*310*)
Maximum	1.836.916 (*3.979.413*)	4921 (*4368*)	1137 (*2049*)
Average	711.221 (*1.490.721*)	2210 (*1957*)	358 (*855*)

Table 3.8: Statistics for the number of points, point density and volume for the 750 synthetic catenary arches dataset. (*In brackets are the statistics for the original catenary arches dataset (after alignment and deleting duplicate points)*).

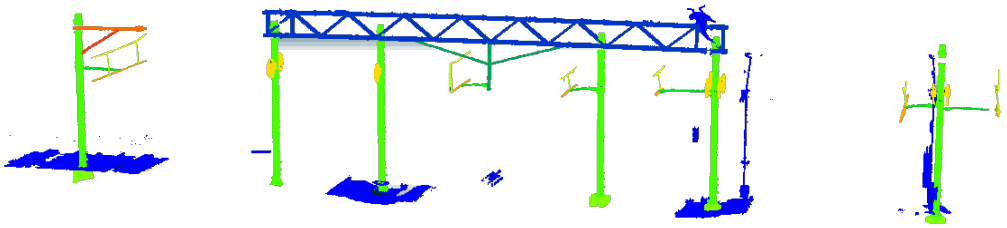
Class	**Points** (%)	***Points*** (%)	**Arch** (%)	***Arch*** (%)
Unlabelled	45.04	*27.30*	100	*100*
Pole	28.03	*40.45*	100	*100*
Top bar	10.14	*13.97*	79.47	*93.33*
Drop post	3.24	*4.34*	75.87	*93.33*
Wheel tension device	3.05	*0.23*	95.07	*20.0*
Top tie	2.96	*2.35*	69.07	*26.67*
Pole foundation	2.28	*4.19*	94.4	*100*
Steady arm	1.37	*1.79*	99.73	*100*
Catenary wire	1.00	*1.53*	99.33	*100*
Bracket	0.95	*0.91*	65.87	*26.67*
Messenger wire support	0.71	*1.51*	59.33	*93.33*
Insulator	0.60	*0.80*	99.33	*100*
Stitch wire	0.44	*0.41*	98.27	*80.0*
Dropper	0.19	*0.24*	97.2	*86.67*

Table 3.9: The percentage of points per class with respect to the total number of points and the percentage of arches in which the class is present for the 750 synthetic catenary arches dataset. The italicised columns contain the statistics for the original catenary arches dataset (after alignment and deleting duplicate points)).



(a) Arch with two poles.

(b) Arch with three poles and without top bar.



(c) Arch with six poles.

Figure 3.8: Examples of the generated, synthetic catenary arches. Different colours indicate different labels.

by the majority classes [11]. In the literature, multiple strategies are developed for dealing with the class imbalance problem. In this study, the focus is on adjusting the loss function to improve the performance of the minority classes.

Traditionally, class imbalance can be handled with sampling strategies [55]. The most straightforward approach involves removing samples from the majority classes and/or duplicating samples from the minority classes. This method does not work in our case since the samples themselves contain all classes. This technique could be applied when generating the synthetic dataset or in the down-sampling step, but it is out-of-scope for this research.

Adjusted loss functions

The loss function calculates the loss for every sample in the input. For each batch, the model uses an optimizer together with backpropagation to adjust its parameters (weights + biases) to minimise the loss. The minimisation process involves calculating the gradient of the loss function. So the loss function influences the updating of the model parameters and thus the learning process. In a non-weighted loss function, the loss for each point in a sample contributes an equal amount to the overall loss. When the points in the input are highly imbalanced, the loss depends primarily on the loss of the majority classes. Thus the model will update its parameters to reduce the loss of the majority classes [3] and does not learn much about the minority classes.

One technique to combat this problem is to adjust the loss function so that each class equally contributes to the loss, despite having a different number of points in the input [123].

A weighted loss function employs different weights for each class [47]. For example, the inverse of the frequency of occurrence of the class in the whole dataset or batch could

be used as a weight factor to increase the influence of the minority classes [26].

The focal loss function adds a factor to the loss function that reduces the relative loss for well-classified samples, focusing more on the misclassified samples [74]. Originally, the focal loss function is designed for binary classification but it can be easily extended to multiclass classification. Focal loss (FL) for multiclass classification for a point p is defined in (3.11), where pt_{p_c} is the probability that pt belongs to class c, α_c is a balancing factor for class c and γ_c is the focusing parameter for class c which should be greater or equal to zero [14]. The term t_c is 1 if c is the ground truth class, 0 otherwise. Here, log is the natural logarithm.

$$\mathbf{FL}(pt) = -\sum_{c \in C} t_c \; \alpha_c \; (1 - pt_{p_c})^{\gamma_c} \; \log(pt_{p_c}), \text{where } C \text{ is the set of all classes} \tag{3.11}$$

The IoU loss is a loss function based on the intersection over union metric [150]. This loss was originally developed for object detection, but it can essentially be applied to all tasks that measure performance with the intersection over union. IoU loss (IoUL) for semantic segmentation on a point cloud pc can be defined as in (3.12), where $mIoU$ is the mean IoU of all classes (defined in (4.3)). Samples that have a large IoU contribute less to the loss while samples with a low IoU contribute more. IoU loss is defined on point clouds and not on individual points as opposed to focal loss.

$$\mathbf{IoUL}(pc) = 1 - mIoU(pc) \tag{3.12}$$

The IoU loss function can be combined with class weights to obtain a weighted IoU loss function. It is defined in (3.13), where w_c is the weight for class c and IoU_c is the IoU for class c (defined in (4.2)) and C is the set of all classes.

$$\mathbf{IoUL}_{weighted}(pc) = \frac{1}{|C|} \sum_{c \in C} w_c \; (1 - IoU_c(pc)) \tag{3.13}$$

In this study, the loss is reduced to a scalar value for each batch by summing all losses and dividing by the number of elements in the batch.

Chapter 4

Results

4.1 Evaluation metrics

To measure the performance of semantic segmentation, two metrics are used: accuracy and intersection over union. Accuracy is defined as the ratio between correctly classified points and the total number of points [35] (see (4.1)). Intersection over union (IoU) (also known as the Jaccard index) measures the overlap between the predicted- and ground truth regions [18]. The IoU for a point cloud *pc* is calculated per class (4.2). Besides, the mean (mIoU) (4.3), median (4.4) and interquartile range (IQR) (4.5) of the IoUs are calculated. The interquartile range measures the variability in the IoUs. The mean and median are both an indication of the overall performance though the median is more robust to outliers than the mean [104]. In (4.4) *ious* is a sorted list of the IoUs per class and n is the number of classes. In (4.5) $ious_{largest}$ is the sorted list of $\lfloor \frac{n}{2} \rfloor$ largest values of *ious* and $ious_{smallest}$ is the sorted list of $\lfloor \frac{n}{2} \rfloor$ smallest values of *ious*. Accuracy is not robust against class imbalance since it is biased towards the majority class [2], thus in this work IoU is chosen as the primary metric to measure the performance of the segmentation.

$$Accuracy(pc) = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

$$IoU_c(pc) = \frac{TP_c}{TP_c + FN_c + FP_c} \tag{4.2}$$

$$mIoU(pc) = \frac{1}{|C|} \sum_{c \in C} IoU_c(pc), \text{ where } C \text{ is the set of all classes} \tag{4.3}$$

$$median\ IoU(pc) = \begin{cases} \frac{ious[\frac{n}{2}-1]+ious[\frac{n}{2}]}{2} & \text{if } n = even, \\ ious[\frac{n+1}{2} - 1] & \text{if } n = odd. \end{cases} \tag{4.4}$$

$$IQR(pc) = median\ IoU(ious_{largest}) - median\ IoU(ious_{smallest}) \tag{4.5}$$

4.2 Experimental setup

The experiments are executed on a server with an Intel processor containing 56 cores and a clock speed of 2.0 GHz and an NVIDIA A10 GPU with 24 GB of memory. The active learning experiment is implemented with PyTorch 1.13.1 [96], while the few-shot learning experiment makes use of TensorFlow version 2.11.1 [1]. This decision was made to profit from existing implementations in the respective frameworks.

Parameter	R5%	R10%	R15%	I5%	I10%	I15%	S
Unlabelled	75.20	73.47	74.04	72.87	74.31	75.45	74.65
Top bar	59.15	64.84	62.83	64.70	66.82	67.19	72.66
Messenger wire support	59.05	63.52	64.38	65.44	71.55	70.41	60.55
Drop post	62.75	69.91	69.84	67.08	68.11	72.36	60.72
Steady arm	59.30	63.56	67.19	63.94	66.72	64.10	65.15
Insulator	48.05	53.42	55.11	48.03	54.99	48.94	49.53
Pole	80.18	80.89	78.84	80.07	80.50	80.01	77.39
Pole foundation	60.40	59.00	61.50	55.27	60.17	51.27	55.27
Dropper	61.38	63.64	64.52	64.20	56.89	62.88	53.21
Stitch wire	51.17	48.46	50.55	52.84	54.47	51.25	51.97
Wheel tension device	0.00	0.00	0.00	0.00	0.00	0.00	34.76
Catenary wire	65.36	62.19	65.45	64.99	62.59	69.27	76.81
Top tie	10.05	14.67	12.67	12.41	12.45	14.18	20.39
Bracket	14.40	0.92	0.29	10.29	2.99	17.66	25.19
mIoU	50.46	51.32	51.94	51.58	52.33	53.23	55.59
Median IoU	59.23	62.86	63.61	64.07	61.38	63.49	57.91
IQR	14.70	16.38	16.64	17.41	13.64	21.47	23.13
Mean Accuracy	82.34	83.01	82.94	82.52	83.37	83.74	82.54

Table 4.1: The results for the different active learning experiments. **R** = random query selector, **I** = informativeness query selector and **S** = supervised.

4.3 Active learning

For the active learning experiments, the boxes are normalised to a range of [-1, 1] with the formula in (4.6) before being fed into the model. This is required since the self-organizing map (SOM) of the SO-Net model expects normalised data. The boxes are also randomly down-sampled to 1024 points. A SOM with 64 nodes is trained offline for 80 epochs for every input point cloud. Afterwards, random Gaussian noise is added to all points, surface normals and SOM nodes. The used Gaussian distribution has a zero mean, a standard deviation of 2cm and it is truncated at ±5cm.

$$Normalised\ point = (point - box\_center)\ /\ (\frac{1}{2} \cdot box\_size) \tag{4.6}$$

In each active learning round, the most informative superpoints are labelled and added to the training set. The labelling is done by assigning all points in a superpoint to the majority class within the superpoint. If there are multiple class labels with the maximum count in the superpoint, one is chosen at random.

4.3.1 Experiments

Fully supervised baseline

As a baseline to compare the active learning method with, the SO-Net segmentation model is trained in a supervised manner on 80% of the 3x3x3m boxes data and validated on the remaining 20%. The results are averaged over ten runs to reduce the randomness of the results. The results are listed in column **S** of Table 4.1. The hyperparameter configuration for this experiment was determined via exploratory research and is given in Table 4.2.

Parameter	Value	Parameter	Value
Epochs	100	Learning rate	0.001
Batch size	32	Activation	ReLU
Dropout	0.5	Batch normalisation	Yes
Input points	1024	Surface normals	True
Feature vector size	256	Optimizer	Adam
Loss	Cross-entropy		

Table 4.2: Hyperparameter configuration for the fully supervised experiment on 3x3x3m boxes dataset with the SO-Net model.

Parameter	Value	Parameter	Value
Initial epochs	30	Annotation budget	5% of points (of all data)
Active epochs	15	Active learning rounds	7

Table 4.3: Hyperparameter configuration for the active learning experiments on 3x3x3m boxes dataset with the SO-Net model.

Random query selector

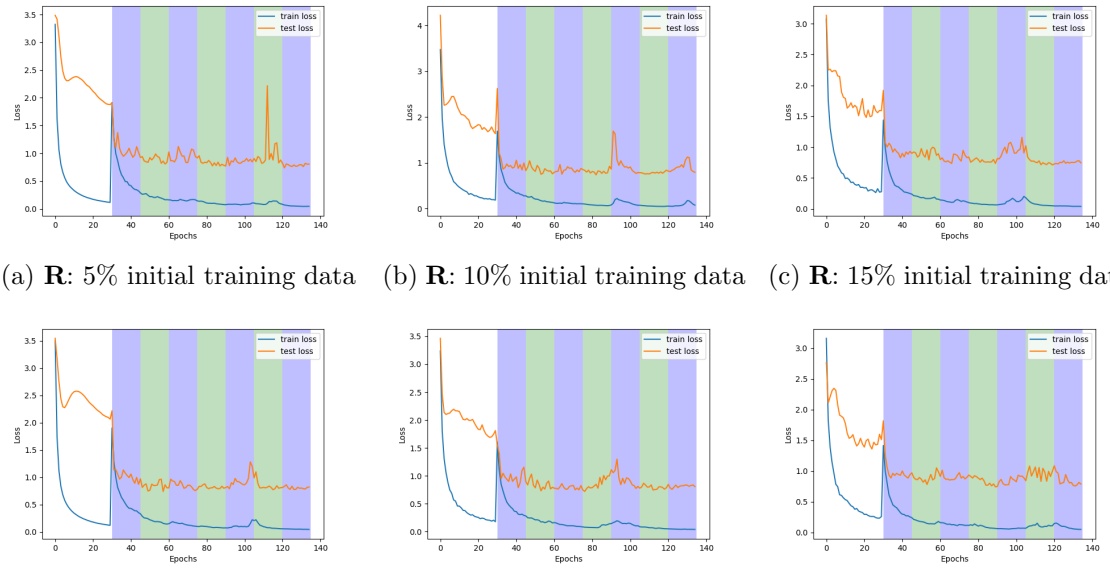
Superpoints are selected for annotation at random in the random query selector experiment. This experiment acts as a baseline to compare the informativeness query selector. The hyperparameter settings for the active learning experiments as determined by exploratory research are listed in Table 4.3. The query selection is implemented so that when the budget is below 50 points, the procedure is halted. This prevents the algorithm from running a long time searching for small superpoints. The hyperparameters in Table 4.2 are still valid, except for the epochs parameter. With this configuration, the model is trained for 135 epochs on 40-50% of the total data in the final epochs depending on the amount of initial training data.

Experiments with 5%, 10% and 15% initial training data were conducted. The results presented here are averages over three runs. Only three runs were carried out for each experiment because of time constraints: one run took about three hours to complete. The results for the experiments with 5%, 10% and 15% initial data can be seen in columns **R5%**, **R10%** and **R15%** of Table 4.1 respectively.

It is expected that the model's performance will increase when the percentage of initial training data increases because there is more data to learn from. Table 4.1 confirms this hypothesis, both the mIoU and the median IoU increase as the amount of initial data increases. The median IoU is significantly higher for all three experiments. This is due to several classes with particularly low performance dragging down the mean IoU (wheel tension device, top tie and bracket). Especially the wheel tension device class, as it scores 0 for all active learning experiments. This is because there are no points from the wheel tension device class in the validation set. Without ground truth points, the intersection and thus the IoU are zero.

The experiment with 15% initial data achieves 93.48% of the performance of the supervised baseline in terms of mIoU. The performance suffers from the wheel tension device class. It is important to note however that the model is trained for 35 more epochs in this experiment than in the supervised experiment. But

From Figure 4.1 (a)-(c) it can be seen that there is an increase in training loss when the first active learning round begins. This may be attributed to the model overfitting on the initial training data. It can also be seen that the more initial training data, the smaller



(a) **R**: 5% initial training data (b) **R**: 10% initial training data (c) **R**: 15% initial training data

(d) **I**: 5% initial training data (e) **I**: 10% initial training data (f) **I**: 15% initial training data

Figure 4.1: The training and test loss for the active learning method with random query selection (**R**) and informativeness query selection (**I**). The blue and green colours indicate active learning rounds.

the increase in training loss and the bigger the increase in test loss at the first epoch of the first active learning round. The model overfits more when the amount of initial training data is smaller, thus the bigger increase in training loss. The increase in test loss is a consequence of the new training data modifying the model parameters considerably at the first epoch causing more inaccurate predictions. The increasing test loss when more initial data is used, is probably a consequence of the test loss being lower after the initial training stage for larger percentages of initial training data.

Informativeness query selector

In the informative query selector experiment, the query selection algorithm described in Section 3.2.3 is employed. The query selection is based on informativeness instead of randomness. It is thus expected to achieve a higher performance than the experiment with random query selection. Since superpoints of minority classes are considered more informative, the class imbalance should be handled automatically by this query selection algorithm. Again, experiments with 5%, 10% and 15% initial training data were conducted and the results are averages over three runs. The hyperparameters are the same as the active learning experiment with a random query selector described in the previous section. Equivalent to the random query selector experiment, the query selection is stopped when the budget is below 50 points.

The results in columns **S5%**, **S10%** and **S15%** in Table 4.1 seem to confirm the hypothesis that selecting queries based on informativeness results in higher performance than random queries. There is a noteworthy difference between mIoU and median IoU. The mIoU slightly increases between random and informativeness query selection (5%: +1.12, 10%: +1.01, 15%: +1.29). The median IoU for the informativeness experiment is however slightly lower (5%: +4.84, 10%: −1.48, 15%: −0.12), except for the experiment with 5% initial data where the informativeness experiment scores almost five percentage points

higher. This median IoU is likely an outlier since it is the highest out of all experiments.

From this experiment, it is not clear if the informativeness query selector handles the class imbalance better than the random query selector. Looking at the four least present classes (disregarding wheel tension device) two have a higher IoU (stitch wire: +2.79, bracket: +5.11) and two a lower (dropper: −1.86, insulator: −1.54). Also, the IQR is not significantly different from the random experiment, except for the I15% experiment where it is considerably higher.

Interestingly, both the bracket (+5.11 IoU) and top tie (+0.55 IoU) classes are better recognised by the model trained using the informativeness selector. These classes are present in the least number of boxes (after the wheel tension device). This indicates that the informativeness selector is more likely to pick data that it has not seen often.

In Figure 4.1 (d)-(f) the training and test losses are plotted for the experiments, they are comparable to the plots of the random experiments although the test loss spikes are less extreme.

The experiment with 15% initial data achieves 95.75% of the performance of the supervised baseline in terms of mIoU. It must be taken into account that the model is trained for 135 epochs instead of 100 as in the supervised experiment. On the other hand, the wheel tension device class drags down the mIoU of the active learning experiment considerably.

The results from Table 4.1 provide evidence that the informative query selector achieves better performance than the random query selector in terms of mIoU. This claim is confirmed by the fact that the informativeness experiment also achieves higher accuracy than the random experiments.

4.4 Few-shot learning

4.4.1 Data preprocessing

The catenary arches are voxelised with the voxel centroid nearest neighbour method. The voxel centroid nearest neighbour method divides the point cloud into voxels where each populated voxel is represented by the centroid of its points (with two points, the voxel is represented as the point closest to the voxel centre). Afterwards, the arches are downsampled to 131.072 points via random sampling to reduce their memory footprint. Both procedures are implemented with PDAL [22] filters. The synthetic arches are normalised by applying a scaling factor of 13.5m (half of the maximum dimension in the real catenary arches dataset). Lastly, there are some random augmentations applied to each input separately: random rotation between −180 and +180 degrees around the z-axis, random translation of the point coordinates between −1m and +1m in all directions and random noise. The random noise is selected from a truncated normal distribution with a mean of zero and a standard deviation of 2cm and is truncated at ±5cm. This preprocessing is identical to the preprocessing employed in [119] so that the results in this experiment can be compared fairly to their results.

4.4.2 Experiments

The generative few-shot learning experiments are conducted on the modified PointNet++ [100] model as described in [119]. The difference with the original PointNet++ model is a third set abstraction level to enhance the segmentation of smaller objects. The hyperparameters used for the experiments are found through exploratory research and can be seen in Table 4.5. The training set is the full synthetic dataset consisting of 750 catenary

Parameter	NW	BW	GW	FL	IL	[119]†
Unlabelled	73.80	74.30	73.76	73.28	26.85	69
Top bar	82.48	80.60	86.80	85.71	0.00	80
Messenger wire support	33.53	35.27	42.40	45.01	0.00	69
Drop post	91.28	84.11	91.80	90.14	0.00	81
Steady arm	56.27	59.44	60.84	53.04	0.00	58
Insulator	54.69	49.33	56.38	57.71	0.00	48
Pole	86.77	85.37	85.73	85.20	0.00	83
Pole foundation	55.23	63.15	54.55	40.92	0.00	67
Dropper	57.16	44.07	60.17	53.90	0.00	51
Stitch wire	65.89	40.89	65.95	69.43	0.00	71
Wheel tension device	85.41	79.51	85.26	80.93	0.00	70
Catenary wire	58.65	77.20	61.32	57.92	0.00	69
Top tie	62.92	70.39	76.94	70.35	0.00	83
Bracket	83.51	74.99	90.57	83.85	0.00	88
mIoU	67.69	65.62	70.89	67.67	1.92	71
Median IoU	64.41	72.35	69.86	69.89	0.00	69.5
IQR	27.24	30.18	25.56	29.95	0.00	14
Mean Accuracy	87.94	87.22	88.66	87.87	26.85	-

Table 4.4: The results for the different few-shot learning experiments. **NW** = non-weighted cross-entropy loss, **BW** = batch-weighted cross-entropy loss, **GW** = global-weighted cross-entropy loss, **FL** = focal loss, **IL** = IoU loss. †Results as reported in [119] using a non-weighted loss function.

arches. After training the model is validated on the real catenary arches dataset consisting of 15 arches.

The experiments were repeated three times to decrease the randomness of the results. Averaging over three runs is a trade-off between reducing randomness and saving time since a single run took approximately eight hours.

Non-weighted loss

The non-weighted loss experiment calculates the loss with the non-weighted cross-entropy loss function. The results are listed under column **NW** in Table 4.4 and the loss plot can be seen in Figure 4.2 (a). From Table 4.4 we can see that the mIoU differs largely per class. Drop post has the highest with 91.28 while messenger wire support only has an mIoU of 33.53. This difference is likely due to class imbalance in the training data. This also shows in the performance of the majority and minority classes. The four classes with the most points have an average IoU of 83.58 while the four classes with the fewest points have an average of 52.82.

In [119] the same modified PointNet++ model is trained on the original catenary dataset and validated using leave-one-out cross-validation. The batch size is four and the model is trained for 400 epochs with a learning rate of 0.01 followed by 200 epochs with a learning rate of 0.001. The results of this experiment are listed in the right-most column (**[119]**) in Table 4.4. The training set in [119] consists of 14 arches, compared to the 750 arches used in our experiment. The number of arches that the model sees during training is comparable in both experiments, 8400 in [119] vs 7500 in our experiment. The model in our experiment reaches 95.34% of the performance achieved in [119] in terms of mIoU.

Parameter	Value	Parameter	Value
Epochs	100	Batch size	6
Input points	131.072	Learning rate	0.01
Batch normalisation	Yes	Optimizer	Adam
Loss	Non-weighted cross-entropy		

Table 4.5: Hyperparameter configuration for the few-shot learning experiments on the synthetic catenary arches dataset.

The performance difference possibly arises from the nature of the training data. In our experiment, the model is trained on synthetic data while in [119] the model is trained on real-world data. The model in [119] is thus trained on data that better resembles the validation data which likely causes higher performance.

The IoU of the wheel tension device (+15.41) and drop post (+10.28) classes is notably better than in the experiment of [119], while the messenger wire support (−35.47), top tie (−20.08), pole foundation (−11.77) and catenary wire (−10.35) score significantly lower. It is plausible that this is due to the different class distributions in the training data of the experiments. Remarkably, the drop post class has a lower point percentage (−1.10) and arch percentage (−17.46) than in the original dataset and the top tie class has both a higher point percentage (+0.61) and arch percentage (+42.40) (see Table 3.9). In the synthetic dataset, the boundary between drop post and top bar is more apparent because there exists some space between them instead of being directly connected. The model might be able to better learn the distinction between drop post and top bar, hence the increase in performance. Because the top tie is present in relatively many training samples, the model might learn that it is likely a top tie is present in a sample. Then at validation, the model also predicts many points as top tie. However, the validation set contains relatively few top ties resulting in a high false positive count.

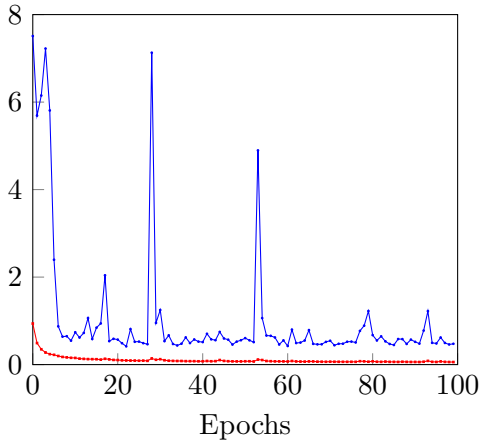
Weighted loss

In this study, the weighted loss function weighs the labels by the inverse of their frequency. The inverse frequency for a class c is calculated by the formula in (4.7). The weights are the normalised inverse frequencies as calculated by the formula in (4.8), where C is the set of all classes.

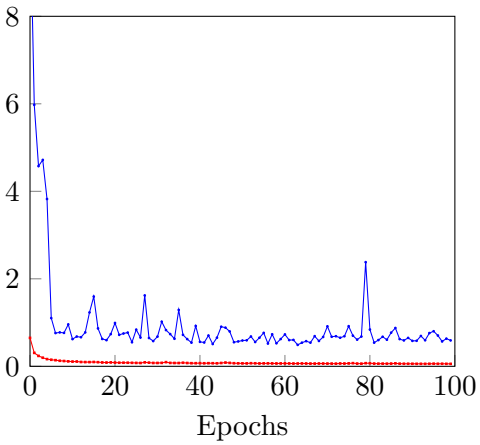
$$IF_c = \frac{total\ points}{points\ of\ class\ c} \tag{4.7}$$

$$W_c = |C| \cdot \frac{IF_c}{\sum_{cl \in C} IF_{cl}} \tag{4.8}$$

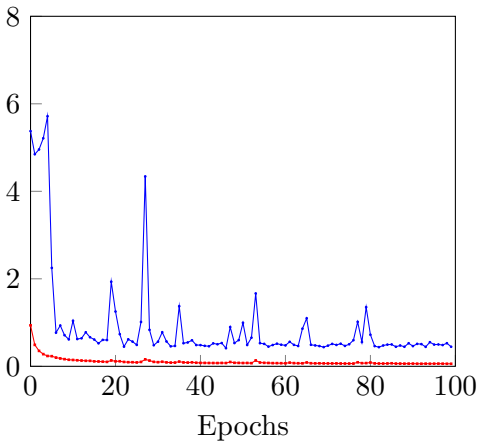
The loss function is weighted with both batch- and global weights. Batch weights are calculated with the inverse frequency per batch and global weights with the inverse frequency in the entire training set. Calculating the inverse frequencies every batch is more computationally heavy than calculating it once and reusing them every batch. On the other hand, each batch contributes an equal amount to the overall loss of the epoch when using batch weights. With global weights, this is not necessarily the case. A batch with relatively many points from minority classes contributes more than a batch with relatively few points from minority classes. Larger batch sizes mitigate this effect since the class distribution in the batches better resembles the distribution of the entire dataset.



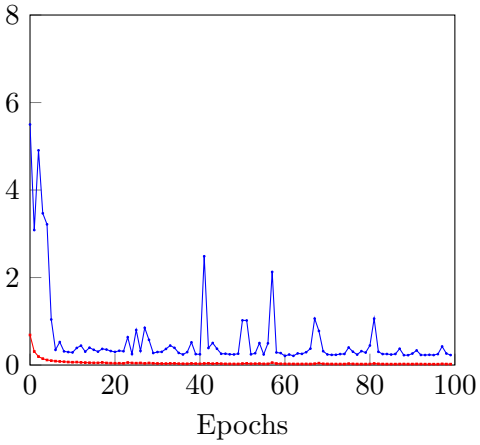
(a) Non-weighted loss



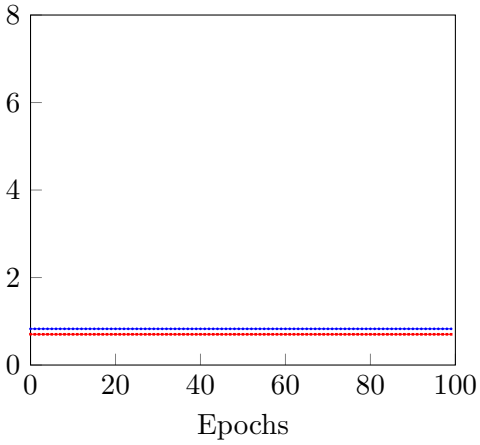
(b) Batch-weighted loss



(c) Global-weighted loss



(d) Focal loss



(e) IoU loss

Figure 4.2: The training (red) and validation (blue) loss for the few-shot learning experiment with different loss functions. The experiments are performed with the modified PointNet++ model trained on the 750 synthetic arches dataset.

The results of the experiment with a weighted loss function are listed in Table 4.4 in the **BW** and **GW** columns for the batch-weighted and global-weighted loss respectively. The loss plots can be seen in Figure 4.2 (b) and (c).

Because of the proportional weights per class, it is expected that the minority classes will perform better than in the experiment with the non-weighted loss function.

The mIoU for the batch-weighted loss function is lower than for the experiment with a non-weighted loss function while the median IoU is higher. Considering the six classes consisting of 1% of the points or less, only catenary wire (+18.55) and messenger wire support (+1.74) show a performance gain. Stitch wire (−25.0), dropper (−13.09), bracket (−8.52) and insulator (−5.36) all experience a performance drop. A possible explanation could be that the weights are relatively unstable. It could be that there are batches where the point distribution is even more imbalanced than in the full dataset. The weights for the minority classes in these batches are so substantial that they disrupt the learning process for these classes, causing a worse IoU compared with the non-weighted loss function.

For the global-weighted loss function, both the mIoU and median IoU are higher than the non-weighted loss function. The minority classes have also improved (messenger wire support (+8.87), bracket (+7.06), dropper (+3.01), catenary wire (+2.67), insulator (+1.69), stitch wire (+0.06)). This indicates that a global-weighted loss function does indeed handle class imbalance better than a non-weighted loss function.

There is a significant difference between the performance of the batch-weighted and the global-weighted loss. Particularly the messenger wire support and bracket classes increase considerably. Those classes are present in relatively few arches (59.33% and 65.87% respectively). This suggests that a global-weighted loss function is better at handling classes that scantly appear in the training samples.

Another interesting result is that the validation loss for the weighted loss functions (see Figure 4.2 (b) and (c)) fluctuates less than the loss from the experiment with a non-weighted loss function (see Figure 4.2 (a)). This is an indication of a more stable training process and leads to fewer variations in the results of different training runs.

Focal loss

An existing implementation of the focal loss function[1] is used with the focusing parameter $\gamma_c = 2$ for all classes and the balancing factor $\alpha_c = 1$ for all classes. The results are listed under column **FL** in Table 4.4. The loss plot can be seen in Figure 4.2 (d).

Since the focal loss function focuses more on the misclassified samples than the well-classified ones, it is expected that the focal loss will emphasise the minority classes because they are relatively frequently misclassified. It is clear from the results that the mIoU is similar to the non-weighted experiment and the median IoU is higher. Out of the six minority classes, only the dropper (−3.26) class sees a significant decrease in performance. The other minority classes have similar (bracket (+0.34), catenary wire (−0.73)) or better performance (messenger wire support (+11.48), stitch wire (+3.54), insulator (+3.02)). This indicates that focal loss is indeed able to implicitly address class imbalance to a certain extent by focusing on misclassified samples.

Compared with the global-weighted loss, the mIoU for the focal loss experiment is lower and the median IoU is similar. Focusing on the six minority classes, it can be seen that three classes have a higher IoU for focal loss (stitch wire (+3.48), messenger wire support (+2.61), insulator (+1.33)) and three have a higher IoU for global-weighted loss (bracket (+6.72), dropper (+6.27), catenary wire (+3.40)). The increase is more significant

[1] `https://github.com/artemmavrin/focal-loss/tree/master`

for the global-weighted loss and since it also has a higher mIoU, it can be concluded the global-weighted loss function results in better performance than the focal loss function.

In Figure 4.2 it can be seen that the loss plot of the focal loss functions displays less extreme peaks than the non-weighted loss plot. This indicates a more stable training process when a focal loss function is employed. The focal loss and both weighted loss plots are comparable.

IoU losses

The loss function must be differentiable in order to calculate its gradients which are used to update the model parameters. The original IoU metric is not differential, hence a differential approximation of IoU is used in the IoU loss function. The approximation of mIoU for a point cloud pc is calculated by (4.9), where $\vec{\boldsymbol{T}}_{pt}$ is the ground truth label of point pt as one-hot vector and $\vec{\boldsymbol{P}}_{pt}$ the predicted labels for point pt as probabilities [101]. The $\cdot$ symbol indicates the dot product operation.

$$mIoU'(pc) = \frac{\sum_{pt \in pc} \vec{\boldsymbol{T}}_{pt} \cdot \vec{\boldsymbol{P}}_{pt}}{\sum_{pt \in pc} \vec{\boldsymbol{T}}_{pt} + \vec{\boldsymbol{P}}_{pt} - (\vec{\boldsymbol{T}}_{pt} \cdot \vec{\boldsymbol{P}}_{pt})} \tag{4.9}$$

The weighted IoU loss function adds a weighting term to the approximation of the mIoU in (4.9). The weighting term is a vector $\vec{\boldsymbol{w}}$ with the weights for class i at index i. The formula for the weighted mIoU for a point cloud pc is stated in (4.10), where $\vec{\boldsymbol{T}}_{pt}$ is the ground truth label of point pt as one-hot vector and $\vec{\boldsymbol{P}}_{pt}$ the predicted labels for point pt as probabilities. The $\cdot$ symbol indicates the dot product operation.

$$mIoU'_{weighted}(pc) = \frac{\sum_{pt \in pc} \vec{\boldsymbol{T}}_{pt} \cdot \vec{\boldsymbol{P}}_{pt} \cdot \vec{\boldsymbol{w}}}{\sum_{pt \in pc} \vec{\boldsymbol{T}}_{pt} + \vec{\boldsymbol{P}}_{pt} - (\vec{\boldsymbol{T}}_{pt} \cdot \vec{\boldsymbol{P}}_{pt} \cdot \vec{\boldsymbol{w}})} \tag{4.10}$$

Training the model with the IoU loss function resulted in poor performance. The model did not learn patterns in the data but simply always predicted the class with the highest point percentage in the training data. The model converged to this configuration after about ten steps in the first epoch. Thus both the training and test loss are constant during the entire training process, as can be seen in Figure 4.2 (e).

The unlabelled class is the class with the highest point percentage in the synthetic arches dataset (see Table 3.9). Hence, the model in this experiment always predicts a point to be of the unlabelled class. The results are identical for both the non-weighted and weighted IoU loss. Therefore, the results are listed in a single column (**IL**) in Table 4.4. The percentage of unlabelled points in the original catenary arches dataset is 27.30%. The IoU of the unlabelled class is slightly different due to the down-sampling of the arches.

4.5 Summary of the results

The results of the active learning experiment show that the informativeness query selector algorithm achieves higher IoUs than the random query selector algorithm. However, it does not necessarily handle the class imbalance better. There does seem to be a significant increase for the classes that are present in a limited number of input samples. Compared with the supervised baseline, the active learning experiments perform reasonably well. The informativeness experiment with 15% initial data achieves 95.75% of the performance of the supervised baseline in terms of mIoU.

The global-weighted loss function achieves the highest mIoU compared to the other loss functions. The weights help to address the class imbalance in the dataset. The focal loss is also able to address the class imbalance but to a lesser extent. The performance is equivalent compared with the experiment conducted in [119]. The synthetic dataset created in this work is thus a good representation of the real-world data and the model is able to learn useful patterns from the synthetic data that are also present in real-world data.

The IoU is generally higher for the few-shot learning experiment (global-weighted: 70.89 mIoU) than for the active learning experiment (**I15%**: 53.23 mIoU). The difference is likely due to the nature and amount of the training data. The training data in the active learning experiment consists of 520 boxes of each 1024 points, while the model in the few-shot learning approach is trained on 750 full arches consisting of 131.072 points. Furthermore, it is more difficult for the model in the boxes experiment to identify patterns in the data since the spatial relations in the arches are not preserved in the boxes.

Both approaches demonstrate performance that is on par with their supervised counterparts. Choosing the best approach depends on the use case. The main drawback of the generative few-shot learning approach in this study is that the generated synthetic dataset is very specific to the original catenary arches dataset. The active learning approach on the other hand is generalisable to other tasks taking point cloud data as input: the active learning loops and informativeness query selector work on any point cloud data. With the active learning approach, the manual labelling effort is reduced considerably. The training process is however interactive and thus more time-consuming than the few-shot learning approach. The main advantage of the few-shot learning approach is that it can take advantage of thoroughly researched supervised learning techniques.

Chapter 5

Discussion and conclusion

5.1 Discussion

The dataset used in this study is gathered from one specific part of the Netherlands. The railway environment varies in different parts of the world (different terrain, signs, poles for example) [113]. Moreover, the dataset is focused on catenary arches. Novel train transportation techniques like magnetic levitation do not use arches. Hence, care must be taken with generalising the results of this study to other railway datasets.

The hardware used in this study poses a substantial limitation. Out-of-memory errors were frequently encountered throughout the training process. Down-sampling the input data and small batch sizes are necessary to prevent memory errors. However, both measures limit the performance of the segmentation models. The results of the experiments will likely improve if they are conducted on hardware that allows larger batch sizes and less down-sampling.

The down-sampling applied to the point clouds in this study is random down-sampling. The class distribution in the point clouds remains similar after the down-sampling step. A solution that potentially better handles class imbalance is to down-sample the point clouds based on the point distribution. Points from majority classes have a higher probability of being removed than points from minority classes. This way the down-sampling step creates a more uniform class distribution in the points clouds thereby reducing the class imbalance in the dataset. A disadvantage of this technique is that it can only be applied to labelled data, which is not always easy to obtain.

The active learning approach in this work is not yet mature enough to be used on large unlabelled datasets. The oracle is fully automatic, which works in this case because a labelled dataset is used. A mechanism that allows for smooth manual labelling by human annotators should be developed to make the active learning approach feasible in practice.

Since the oracle is automatic, it is also difficult to measure the labelling effort per active learning round. In this study, a percentage of points is used as the annotation budget, but in practice, the number of clicks or the spent time will be more relevant budget units.

The option to run the active learning approach on the created synthetic dataset is not explored in this work since the synthetic data is already labelled. Here, it is easier to train on all the labelled data instead of selecting the most informative samples.

The generative few-shot learning approach creates new data by following object placement rules. The main drawback is that this synthetic data is too specific to be used on other railway datasets. To get a synthetic dataset that is more general, extra objects and extra object placement rules are needed. The object placement rules should then take more different objects into account which will result in more complex rules. Creating

these rules will be a difficult and time-consuming task.

The synthetic data obviously should have a resemblance to the original data. However, too much resemblance will likely lead to overfitting, especially if the initial dataset is small. This overfitting will lead to a good performance on the original dataset but does not allow for any generalisation to other datasets. On the other hand, if the synthetic data does not resemble the original data enough, the model will not be able to learn patterns in the original data. The synthetic data samples must be exactly in the middle of too much and too little resemblance. The synthetic dataset produced in this work likely exhibits too strong a resemblance to the original dataset since the object placement rules are derived from the original data and the randomness introduced is limited.

In this work, the whole labelled catenary arches dataset is used to create the synthetic data and thus in this particular case, the manual labelling effort is not reduced. On the other hand, out of the 15 arches 750 new arches are created. So, from the total of 765 arches less than 2% is manually labelled. This approach thus shows that it is feasible to generate synthetic data out of a small set of manually labelled data, hence reducing the manual labelling effort.

The semi-supervised learning techniques applied in this study, seem to be suitable techniques to solve the point cloud semantic segmentation problem for railway data. The downside is that implementing it is considerably more difficult than supervised learning. The advantage is that the manual labelling effort is reduced considerably. Since only two semi-supervised paradigms are implemented and tested, a general conclusion about semi-supervised cannot be drawn.

Deep learning has gained a lot of attention recently (think of ChatGPT) and is used regularly as a solution to complex problems. Hence, it is reasonable to apply deep learning to point cloud semantic segmentation. Nevertheless, deep learning might not always be the most appropriate solution. Point clouds are large data structures that take up significant amounts of memory. This limits the model- and batch size. Furthermore, deep learning takes advantage of patterns and relations found in the data. Because of the diversity in railway infrastructure, railway data possibly does not contain enough patterns and relations that the model can leverage for its predictions. These arguments raise the question of whether deep learning is the best solution for point cloud semantic segmentation of railway scenes.

Auto-encoder active learning approach

Initially, the idea behind the active learning approach was to first cluster the points in the point cloud and then use these clusters as query units. The clustering is based on the feature vectors learned from the auto-encoder described in [69]. At first, the auto-encoder was trained with unsupervised learning on the catenary arches. However many out-of-memory errors were encountered since the arches took up lots of memory, even after downsampling. After hyperparameter tuning, the reconstruction results were not sufficient (see Figure 5.1). It was then decided to split up the arches into boxes. As can be seen in Figure 5.2, the results were much better.

Using the trained encoder from the auto-encoder, an initial segmentation was obtained from the segmentation network developed in [69]. Without using any labelled data the performance was not sufficient. Using 10% labelled data, the performance increased substantially. Now the segments could be used for active learning on the same segmentation network.

However, the active learning rounds were not implemented for this method. The reason is that it seems redundant to use a segmentation model to get an initial segmentation

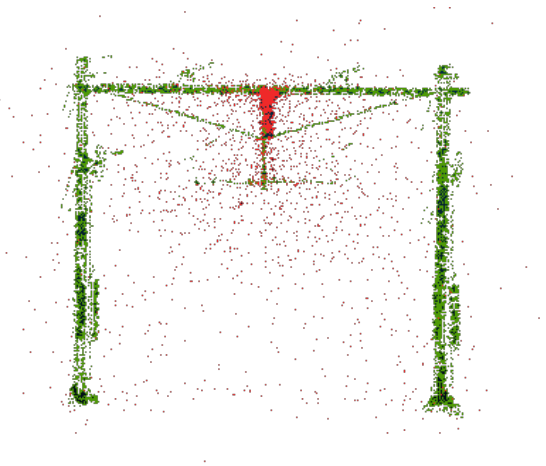


Figure 5.1: The reconstruction generated by the SO-Net auto-encoder of a catenary arch (red) and the original catenary arch (green). A cluster is formed at the top-centre of the arch.

(a) Side view.

(b) Top view.

Figure 5.2: The reconstruction generated by the SO-Net auto-encoder (red) and the original input data (green) when using 3x3x3m boxes as input.

which serves as input for the same segmentation model. If the initial segmentation model is not performing well, the second segmentation model will probably also not perform well. On the other hand, if the initial segmentation is good, then the end goal of semantic segmentation is already reached by this intermediate step. Moreover, the segmentation model predicts the class labels for each point while clustering just groups similar points together without knowledge about object categories. Grouping similar points can be done with simpler tasks than semantic segmentation.

Traditional clustering algorithms base the clustering on some feature set of the units. The learned feature vectors from the auto-encoder representing the boxes could be used with traditional clustering algorithms. However, since the goal is to cluster points and not boxes, clustering did not seem appropriate in this setup.

At this time, the over-segmentation technique was brought up and considered a better fit to generate the units for active learning.

5.2 Conclusion

Point cloud semantic segmentation on railway data using supervised learning requires a considerable amount of manual labelling effort because no labelled railway datasets are available. Semi-supervised learning can be exploited to reduce the labelling effort. In this work, two approaches following different semi-supervised paradigms are implemented. This way it is attempted to get more insight into how to effectively apply semi-supervised learning to the task of semantic segmentation of large-scale point clouds of the railway environment.

Both active learning and generative few-shot learning can be effectively applied to the semantic segmentation of large point cloud scenes of the railway environment. The active learning approach with the informativeness query selector achieved over 95% performance of the supervised baseline while using 37.5% less labelled data. The generative few-shot learning approach with a global-weighted loss function achieved equivalent performance as in [119] while training on only synthetic data.

Choosing which approach to use depends on the situation. If the data can be recreated with relatively simple rules and generalisability it is not a critical requirement, generative few-shot learning seems the best option. However, the active learning approach is more general and can be applied to different point cloud datasets.

A major challenge encountered in this work for point cloud segmentation in the domain of railway environments is the class imbalance in the data. The objects that need to be recognised differ largely in size and thus also in the number of points in the dataset. Developing better techniques to handle class imbalance will improve both approaches and is thus an interesting topic for future research.

5.3 Future work

The models in this work suffer from class imbalance in the dataset even when measures like adjusted loss functions are implemented. Future work could look into other or new class imbalance techniques to improve the performance of models on class-imbalanced datasets. An appropriate option is to extend networks with a memory module to alleviate the problem of forgetting patterns from minority classes. The work in [45] could be taken as a baseline to implement memory modules for segmentation models.

Recently, the explainability of artificial intelligence models has been broadly studied [28, 33]. Explainability helps researchers understand the models better and so could improve upon the shortcomings made visible through model understanding. Applying explainability frameworks like SHAP [81] or LIME [103] on point cloud models could give insight into further improvements of the models.

Another direction to explore is data fusion. In this work, the input data consists of only point clouds with x-, y- and z-coordinates. If images are available, it could be helpful to use them in combination with the point clouds [25]. Images provide extra information like colour. Colour is an important factor in determining object boundaries [57, 140]. Adding colour information can improve the over-segmentation as well as the semantic segmentation performance.

Furthermore, other forms of semi-supervised learning could be explored. For instance, self-supervised learning where a feature representation is learned through a pretext task which can be based for example on contrast learning [70] or reconstruction of incomplete point clouds [151]. Another interesting approach is to combine active learning with region-growing methods. Here, the oracle selects seed points that initialise the region-growing

method. Region-growing algorithms have shown promising results on railway data [4, 152]. But generally, seed selection is the bottleneck so selecting them through an oracle could alleviate this problem.

Another approach quite common in literature is to focus on one specific object category in the dataset like (catenary) wires [16, 73] or rail tracks [105, 134]. These methods are usually based on the geometric properties of the object categories. These structural methods could be leveraged by detecting objects from sparse classes followed by a deep learning model to detect the general objects. For example, first detecting the wires, removing them from the point cloud and feeding the remaining points to a deep learning model.

Yet another technique that could be investigated is segmentation refinement. Refinement techniques try to improve the output of segmentation models. The most prevalent refinement techniques are conditional random fields (CRF) [97, 116] and Markov random fields (MRF) [54, 79]. Other techniques like the attention-based score refinement (ASR) module [146] could also be explored.

Lastly, the oracle employed in the active learning rounds in this study is not a human but an automaton. This is possible because the catenary arches dataset is labelled. In practice, the labelled data is usually not available and an actual unlabelled pool must be used. To improve the quality and reduce the effort of labelling, an efficient labelling process for human annotators must be developed. This was deemed out-of-scope for this study but is important to make the active learning approach practical.

Bibliography

[1] M. Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. URL: `https://www.tensorflow.org/`.

[2] S. M. Abd Elrahman and A. Abraham. "A Review of Class Imbalance Problem". *Journal of Network and Innovative Computing* 1.2013 (2013), pp. 332–340.

[3] R. Anand, K. G. Mehrotra, C. K. Mohan, and S. Ranka. "An Improved Algorithm for Neural Network Classification of Imbalanced Training Sets". *IEEE Transactions on Neural Networks* 4.6 (1993), pp. 962–969.

[4] M. Arastounia. "An Enhanced Algorithm for Concurrent Recognition of Rail Tracks and Power Cables from Terrestrial and Airborne Lidar Point Clouds". *Infrastructures* 2.2 (2017), p. 8.

[5] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. "3D Semantic Parsing of Large-Scale Indoor Spaces". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 1534–1543.

[6] W. Bai, O. Oktay, M. Sinclair, H. Suzuki, M. Rajchl, G. Tarroni, B. Glocker, A. King, P. M. Matthews, and D. Rueckert. "Semi-Supervised Learning for Network-Based Cardiac MR Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2017: 20th International Conference, Quebec City, QC, Canada, September 11-13, 2017, Proceedings, Part II 20*. Springer. 2017, pp. 253–260.

[7] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9297–9307.

[8] S. A. Bello, S. Yu, C. Wang, J. M. Adam, and J. Li. "Deep Learning on 3D Point Clouds". *Remote Sensing* 12.11 (2020), p. 1729.

[9] A. Boulch. "ConvPoint: Continuous Convolutions for Point Cloud Processing". *Computers & Graphics* 88 (2020), pp. 24–34.

[10] Q. Bouniot, I. Redko, R. Audigier, A. Loesch, and A. Habrard. "Improving Few-Shot Learning Through Multi-Task Representation Learning Theory". In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XX*. Springer. 2022, pp. 435–452.

[11] M. Buda, A. Maki, and M. A. Mazurowski. "A Systematic Study of the Class Imbalance Problem in Convolutional Neural Networks". *Neural Networks* 106 (2018), pp. 249–259.

[12] Q. Cai, Y. Pan, T. Yao, C. Yan, and T. Mei. "Memory Matching Networks for One-Shot Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4080–4088.

[13] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. *ShapeNet: An Information-Rich 3D Model Repository*. 2015. arXiv: 1512.03012 [cs.GR].

[14] J. Chang, X. Zhang, M. Ye, D. Huang, P. Wang, and C. Yao. "Brain Tumor Segmentation Based on 3D U-Net with Multi-Class Focal Loss". In: *2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. IEEE. 2018, pp. 1–5.

[15] E. Che, J. Jung, and M. J. Olsen. "Object Recognition, Segmentation, and Classification of Mobile Laser Scanning Point Clouds: A State of the Art Review". *Sensors* 19.4 (2019), p. 810.

[16] X. Chen, Z. Chen, G. Liu, K. Chen, L. Wang, W. Xiang, and R. Zhang. "Railway Overhead Contact System Point Cloud Classification". *Sensors* 21.15 (2021), p. 4961.

[17] X. Chen, Y. Li, L. Yao, E. Adeli, Y. Zhang, and X. Wang. "Generative Adversarial U-Net for Domain-Free Few-Shot Medical Diagnosis". *Pattern Recognition Letters* 157 (2022), pp. 112–118.

[18] Y.-J. Cho. *Weighted Intersection over Union (wIoU): A New Evaluation Metric for Image Segmentation*. 2023. arXiv: 2107.09858 [cs.CV].

[19] CloudCompare. *CloudCompare - Open Source project. Open-source point cloud editing software*. Version 2.12.4. 2023. URL: https://www.cloudcompare.org/ (visited on 01/30/2023).

[20] D. A. Cohn, Z. Ghahramani, and M. I. Jordan. "Active Learning with Statistical Models". *Journal of Artificial Intelligence Research* 4 (1996), pp. 129–145.

[21] E. Commission, D.-G. for Mobility, and Transport. *EU Transport in Figures: Statistical Pocketbook 2022*. Publications Office of the European Union, 2022.

[22] P. Contributors. *PDAL Point Data Abstraction Library*. Aug. 2022. URL: https://doi.org/10.5281/zenodo.2616780.

[23] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 3213–3223.

[24] M. Cserep. *Hungarian MLS Point Clouds of Railroad Environment and Annotated Ground Truth Data*. 2022. URL: https://data.mendeley.com/datasets/ccxpzhx9dj/1.

[25] Y. Cui, R. Chen, W. Chu, L. Chen, D. Tian, Y. Li, and D. Cao. "Deep Learning for Image and Point Cloud Fusion in Autonomous Driving: A Review". *IEEE Transactions on Intelligent Transportation Systems* 23.2 (2021), pp. 722–739.

[26] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie. "Class-Balanced Loss Based on Effective Number of Samples". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9268–9277.

[27] A. Dai and M. NieSSner. "3DMV: Joint 3D-Multi-View Prediction for 3D Semantic Scene Segmentation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 452–468.

[28] P. Dardouillet, A. Benoit, E. Amri, P. Bolon, D. Dubucq, and A. Crédoz. "Explainability of Image Semantic Segmentation Through SHAP Values". In: *Proceedings of the International Conference on Pattern Recognition, Computer Vision, and Image Processing. ICPR 2022 International Workshops and Challenges*. 2022, pp. 188–202.

[29] M. Dassot, T. Constant, and M. Fournier. "The Use of Terrestrial LiDAR Technology in Forest Science: Application Fields, Benefits and Challenges". *Annals of Forest Science* 68 (2011), pp. 959–974.

[30] A. Diab, R. Kashef, and A. Shaker. "Deep Learning for LiDAR Point Cloud Classification in Remote Sensing". *Sensors* 22.20 (2022), p. 7868.

[31] X. Dong, Y. Xiao, Z. Chen, J. Yao, and X. Guo. "GPU-Based Supervoxel Segmentation for 3D Point Clouds". *Computer Aided Geometric Design* 93 (2022), p. 102080.

[32] A. R. Fayjie and P. Vandewalle. "Few-Shot Learning on Point Clouds for Railroad Segmentation". *Electronic Imaging* 35.17 (2023), pp. 100-1–100-1. URL: `https://library.imaging.org/ei/articles/35/17/3DIA-100`.

[33] T. Fel, L. Hervier, D. Vigouroux, A. Poche, J. Plakoo, R. Cadene, M. Chalvidal, J. Colin, T. Boissin, L. Bethune, A. Picard, C. Nicodeme, L. Gardes, G. Flandin, and T. Serre. *Xplique: A Deep Learning Explainability Toolbox*. 2022. arXiv: `2206.04394 [cs.LG]`.

[34] H. Feng, Z. Jiang, F. Xie, P. Yang, J. Shi, and L. Chen. "Automatic Fastener Classification and Defect Detection in Vision-Based Railway Inspection Systems". *IEEE Transactions on Instrumentation and Measurement* 63.4 (2013), pp. 877–888.

[35] E. Fernandez-Moral, R. Martins, D. Wolf, and P. Rives. "A New Metric for Evaluating Semantic Segmentation: Leveraging Global and Contour Accuracy". In: *IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2018, pp. 1051–1056.

[36] C. Finn, P. Abbeel, and S. Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. PMLR. 2017, pp. 1126–1135.

[37] A. Fuller, Z. Fan, C. Day, and C. Barlow. "Digital Twin: Enabling Technologies, Challenges and Open Research". *IEEE Access* 8 (2020), pp. 108952–108971.

[38] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative Adversarial Networks". *Communications of the ACM* 63.11 (2020), pp. 139–144.

[39] J. Grandío, B. Riveiro, M. Soilán, and P. Arias. "Point Cloud Semantic Segmentation of Complex Railway Environments Using Deep Learning". *Automation in Construction* 141 (2022), p. 104425.

[40] Y. Guo, F. Sohel, M. Bennamoun, J. Wan, and M. Lu. "A Novel Local Surface Feature for 3D Object Recognition Under Clutter and Occlusion". *Information Sciences* 293 (2015), pp. 196–213.

[41] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun. "Deep Learning for 3D Point Clouds: A Survey". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.12 (2020), pp. 4338–4364.

[42] T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler, and M. Pollefeys. *Semantic3D.net: A new Large-scale Point Cloud Classification Benchmark*. 2017. arXiv: `1704.03847 [cs.CV]`.

[43] D. Han, J. Yoo, and D. Oh. "SeeThroughNet: Resurrection of Auxiliary Loss by Preserving Class Probability Information". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 4463–4472.

[44] K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[45] T. He, D. Gong, Z. Tian, and C. Shen. "Learning and Memorizing Representative Prototypes for 3D Point Cloud Semantic and Instance Segmentation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2020, pp. 564–580.

[46] A. Heidari, J. McGrath, I. F. Ilyas, and T. Rekatsinas. "HoloDetect: Few-Shot Learning for Error Detection". In: *Proceedings of the 2019 International Conference on Management of Data*. 2019, pp. 829–846.

[47] Y. Ho and S. Wookey. "The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling". *IEEE Access* 8 (2019), pp. 4806–4813.

[48] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham. "RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11108–11117.

[49] R. Hu, B. Mac Namee, and S. J. Delany. "Off to a Good Start: Using Clustering to Select the Initial Training Set in Active Learning". In: *Proceedings of the 23rd International Florida Artificial Intelligence Research Society Conference (FLAIRS)*. Technological University Dublin, 2010, pp. 26–31.

[50] Z. Hu, X. Li, C. Tu, Z. Liu, and M. Sun. "Few-Shot Charge Prediction with Discriminative Legal Attributes". In: *Proceedings of the 27th International Conference on Computational Linguistics*. 2018, pp. 487–498.

[51] Q. Huang, W. Wang, and U. Neumann. "Recurrent Slice Networks for 3D Segmentation of Point Clouds". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2626–2635.

[52] M. Jaritz, J. Gu, and H. Su. "Multi-View PointNet for 3D Scene Understanding". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019, pp. 3995–4003.

[53] A. Ji, Y. Zhou, L. Zhang, R. L. Tiong, and X. Xue. "Semi-Supervised Learning-Based Point Cloud Network for Segmentation of 3D Tunnel Scenes". *Automation in Construction* 146 (2023), p. 104668. ISSN: 0926-5805. URL: `https://www.sciencedirect.com/science/article/pii/S0926580522005386`.

[54] V. Jiménez, J. Godoy, A. Artuñedo, and J. Villagra. "Ground Segmentation Algorithm for Sloped Terrain and Sparse LiDAR Point Cloud". *IEEE Access* 9 (2021), pp. 132914–132927.

[55] J. M. Johnson and T. M. Khoshgoftaar. "Survey on Deep Learning with Class Imbalance". *Journal of Big Data* 6.1 (2019), pp. 1–54.

[56] A. J. Joshi, F. Porikli, and N. Papanikolopoulos. "Multi-Class Active Learning for Image Classification". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2009, pp. 2372–2379.

[57] F. S. Khan, R. M. Anwer, J. Van De Weijer, A. D. Bagdanov, M. Vanrell, and A. M. Lopez. "Color Attributes for Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, pp. 3306–3313.

[58] V. Kocaman, O. M. Shir, T. Bäck, and A. N. Belbachir. "Saliency Can Be All You Need in Contrastive Self-supervised Learning". In: *Advances in Visual Computing: 17th International Symposium, ISVC 2022, San Diego, CA, USA, October 3–5, 2022, Proceedings, Part II*. Springer. 2022, pp. 119–140.

[59] T. Kohonen. "The Self-Organizing Map". *Proceedings of the IEEE* 78.9 (1990), pp. 1464–1480.

[60] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". *Communications of the ACM* 60.6 (2017), pp. 84–90.

[61] A. Kumar and P. Sinha. "Human Error Control in Railways". *Jordan Journal of Mechanical and Industrial Engineering* 2.4 (2008).

[62] R. Kwitt, S. Hegenbart, and M. Niethammer. "One-Shot Learning of Scene Locations via Feature Trajectory Transfer". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 78–86.

[63] D. Lamas, M. Soilán, J. Grandío, and B. Riveiro. "Automatic Point Cloud Semantic Segmentation of Complex Railway Environments". *Remote Sensing* 13.12 (2021), p. 2332.

[64] L. Landrieu and M. Simonovsky. "Large-Scale Point Cloud Semantic Segmentation with Superpoint Graphs". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4558–4567.

[65] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. "Handwritten Digit Recognition with a Back-Propagation Network". In: *Proceedings of the 4th International Conference on Neural Information Processing System (NIPS)*. Vol. 2. 1989.

[66] J. S. Lee, J. Park, and Y.-M. Ryu. "Semantic Segmentation of Bridge Components Based on Hierarchical Point Cloud Model". *Automation in Construction* 130 (2021), p. 103847.

[67] A. Levinshtein, A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson, and K. Siddiqi. "TurboPixels: Fast Superpixels using Geometric Flows". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.12 (2009), pp. 2290–2297.

[68] G. Li, G. Kang, X. Wang, Y. Wei, and Y. Yang. "Adversarially Masking Synthetic To Mimic Real: Adaptive Noise Injection for Point Cloud Segmentation Adaptation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 20464–20474.

[69] J. Li, B. M. Chen, and G. H. Lee. "SO-Net: Self-Organizing Network for Point Cloud Analysis". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 9397–9406.

[70] X. Li, L. Zhang, and Z. Zhu. "SnapshotNet: Self-Supervised Feature Learning for Point Cloud Data Segmentation using Minimal Labeled Data". *Computer Vision and Image Understanding* 216 (2022), p. 103339.

[71] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. "PointCNN: Convolution on X-Transformed Points". *Proceedings of the 32nd International Conference on Neural Information Processing System (NeurIPS)* 31 (2018).

[72] Y. Li, L. Ma, Z. Zhong, F. Liu, M. A. Chapman, D. Cao, and J. Li. "Deep Learning for LiDAR Point Clouds in Autonomous Driving: A Review". *IEEE Transactions on Neural Networks and Learning Systems* 32.8 (2020), pp. 3412–3432.

[73] S. Lin, C. Xu, L. Chen, S. Li, and X. Tu. "LiDAR Point Cloud Recognition of Overhead Catenary System with Deep Learning". *Sensors* 20.8 (2020), p. 2212.

[74] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. "Focal Loss for Dense Object Detection". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2980–2988.

[75] Y. Lin, C. Wang, D. Zhai, W. Li, and J. Li. "Toward Better Boundary Preserved Supervoxel Segmentation for 3D Point Clouds". *ISPRS Journal of Photogrammetry and Remote Sensing* 143 (2018), pp. 39–47.

[76] J. Liu, F. Chao, and C.-M. Lin. *Task Augmentation by Rotating for Meta-Learning*. 2020. arXiv: `2003.00804 [cs.CV]`.

[77] W. Liu, J. Sun, W. Li, T. Hu, and P. Wang. "Deep Learning on Point Clouds and its Application: A Survey". *Sensors* 19.19 (2019), p. 4188.

[78] Y. Liu, J. Lee, M. Park, S. Kim, E. Yang, S. J. Hwang, and Y. Yang. *Learning to Propagate Labels: Transductive Propagation Network for Few-shot Learning*. 2019. arXiv: `1805.10002 [cs.LG]`.

[79] D. Lodi Rizzini, F. Oleari, A. Atti, J. Aleotti, and S. Caselli. "Unsupervised Range Image Segmentation and Object Recognition using Feature Proximity and Markov Random Field". In: *Intelligent Autonomous Systems 13: Proceedings of the 13th International Conference IAS-13*. Springer. 2016, pp. 807–820.

[80] J. Long, E. Shelhamer, and T. Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3431–3440.

[81] S. M. Lundberg and S.-I. Lee. "A Unified Approach to Interpreting Model Predictions". In: *Proceedings of the 31st International Conference on Neural Information Processing System (NIPS)*. Vol. 30. 2017, pp. 4768–4777.

[82] J. W. Ma, T. Czerniawski, and F. Leite. "Semantic Segmentation of Point Clouds of Building Interiors with Deep Learning: Augmenting Training Datasets with Synthetic BIM-Based Point Clouds". *Automation in Construction* 113 (2020), p. 103144.

[83] D. Martin, C. Fowlkes, D. Tal, and J. Malik. "A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics". In: *Proceedings of the 8th IEEE International Conference on Computer Vision*. Vol. 2. IEEE. 2001, pp. 416–423.

[84] A. Mehrotra and A. Dukkipati. *Generative Adversarial Residual Pairwise Networks for One Shot Learning*. 2017. arXiv: `1703.08033 [cs.CV]`.

[85] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss. "RangeNet++: Fast and Accurate LiDAR Semantic Segmentation". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 4213–4220.

[86] Y. Mo, Y. Wu, X. Yang, F. Liu, and Y. Liao. "Review the State-of-the-Art Technologies of Semantic Segmentation Based on Deep Learning". *Neurocomputing* 493 (2022), pp. 626–646.

[87] A. K. Mondal, J. Dolz, and C. Desrosiers. *Few-shot 3D Multi-Modal Medical Image Segmentation using Generative Adversarial Learning*. 2018. arXiv: `1810.12241 [cs.CV]`.

[88] T. Munkhdalai and H. Yu. "Meta Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. PMLR. 2017, pp. 2554–2563.

[89] T. Munkhdalai, X. Yuan, S. Mehri, and A. Trischler. "Rapid Adaptation with Conditionally Shifted Neurons". In: *Proceedings of the 35th International Conference on Machine Learning*. PMLR. 2018, pp. 3664–3673.

[90] M. Neri and F. Battisti. "3D Object Detection on Synthetic Point Clouds for Railway Applications". In: *10th European Workshop on Visual Information Processing (EUVIP)*. IEEE. 2022, pp. 1–6.

[91] P. Neubert and P. Protzel. "Superpixel Benchmark and Comparison". In: *Proceedings of the Forum Bildverarbeitung*. Vol. 6. 2012, pp. 1–12.

[92] A. Nurunnabi, G. West, and D. Belton. "Outlier Detection and Robust Normal-Curvature Estimation in Mobile Laser Scanning 3D Point Cloud Data". *Pattern Recognition* 48.4 (2015), pp. 1404–1419.

[93] G. Papandreou, L.-C. Chen, K. P. Murphy, and A. L. Yuille. "Weakly- and Semi-Supervised Learning of a Deep Convolutional Network for Semantic Image Segmentation". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1742–1750.

[94] J. Papon, A. Abramov, M. Schoeler, and F. Worgotter. "Voxel Cloud Connectivity Segmentation - Supervoxels for Point Clouds". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 2027–2034.

[95] A. Parnami and M. Lee. *Learning from Few Examples: A Summary of Approaches to Few-Shot Learning*. 2022. arXiv: `2203.04291 [cs.LG]`.

[96] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Proceedings of the 32nd International Conference on Neural Information Processing System (NeurIPS)*. Vol. 32. 2019, pp. 7994–8005.

[97] Q.-H. Pham, T. Nguyen, B.-S. Hua, G. Roig, and S.-K. Yeung. "JSIS3D: Joint Semantic-Instance Segmentation of 3D Point Clouds with Multi-Task Pointwise Networks and Multi-Value Conditional Random Fields". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8827–8836.

[98] C. R. Qi, O. Litany, K. He, and L. J. Guibas. "Deep Hough Voting for 3D Object Detection in Point Clouds". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9277–9286.

[99] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 652–660.

[100] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". In: *Proceedings of the 31st International Conference on Neural Information Processing System (NIPS)*. Vol. 30. 2017.

[101] M. A. Rahman and Y. Wang. "Optimizing Intersection-over-Union in Deep Neural Networks for Image Segmentation". In: *International Symposium on Visual Computing*. Springer. 2016, pp. 234–244.

[102] X. Ren and J. Malik. "Learning a Classification Model for Segmentation". In: *Proceedings of the 9th IEEE International Conference on Computer Vision*. IEEE. 2003, pp. 10–17.

[103] M. T. Ribeiro, S. Singh, and C. Guestrin. ""Why should I trust you?": Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 1135–1144.

[104] P. J. Rousseeuw and M. Hubert. "Robust Statistics for Outlier Detection". *WIREs Data Mining and Knowledge Discovery* 1.1 (2011), pp. 73–79.

[105] S. Sahebdivani, H. Arefi, and M. Maboudi. "Rail Track Detection and Projection-Based 3D Modeling from UAV Point Cloud". *Sensors* 20.18 (2020), p. 5220.

[106] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. "Meta-Learning with Memory-Augmented Neural Networks". In: *Proceedings of the 33rd International Conference on Machine Learning*. PMLR. 2016, pp. 1842–1850.

[107] B. W. Schlake, C. P. Barkan, and J. R. Edwards. "Train Delay and Economic Impact of In-Service Failures of Railroad Rolling Stock". *Transportation Research Record* 2261.1 (2011), pp. 124–133.

[108] B. Settles. *Active Learning Literature Survey*. 2009. URL: `http://digital.library.wisc.edu/1793/60660`.

[109] B. Settles. "From Theories to Queries: Active Learning in Practice". In: *Active Learning and Experimental Design Workshop in Conjunction with AISTATS 2010*. Vol. 16. PMLR. 2011, pp. 1–18.

[110] X. Shi, X. Xu, K. Chen, L. Cai, C. S. Foo, and K. Jia. *Label-Efficient Point Cloud Semantic Segmentation: An Active Learning Approach*. 2021. arXiv: `2101.06931 [cs.CV]`.

[111] P. Shyam, S. Gupta, and A. Dukkipati. "Attentive Recurrent Comparators". In: *Proceedings of the 34th International Conference on Machine Learning*. PMLR. 2017, pp. 3173–3181.

[112] K. Simonyan and A. Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: `1409.1556 [cs.CV]`.

[113] A. Steimel. "Under Europe's Incompatible Catenary Voltages A Review of Multi-System Traction Technology". *2012 Electrical Systems for Aircraft, Railway and Ship Propulsion* (2012), pp. 1–8.

[114] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele. "Meta-Transfer Learning for Few-Shot Learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 403–412.

[115] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. "Going Deeper with Convolutions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1–9.

[116] L. Tchapmi, C. Choy, I. Armeni, J. Gwak, and S. Savarese. "SEGCloud: Semantic Segmentation of 3D Point Clouds". In: *2017 International Conference on 3D Vision (3DV)*. IEEE. 2017, pp. 537–547.

[117] G. Te, W. Hu, A. Zheng, and Z. Guo. "RGCNN: Regularized Graph CNN for Point Cloud Segmentation". In: *Proceedings of the 26th ACM International Conference on Multimedia*. 2018, pp. 746–754.

[118] B. Ton. *Labelled High Resolution Point Cloud Dataset of 15 Catenary Arches in the Netherlands*. 2022. URL: `https://data.4tu.nl/articles/dataset/Labelled_high_resolution_point_cloud_dataset_of_15_catenary_arches_in_the_Netherlands/17048816/1`.

[119] B. Ton, F. Ahmed, and J. Linssen. "Semantic Segmentation of Terrestrial Laser Scans of Railway Catenary Arches: A Use Case Perspective". *Sensors* 23.1 (2022), p. 222.

[120] G. Uggla and M. Horemuz. "Towards Synthesized Training Data for Semantic Segmentation of Mobile Laser Scanning Point Clouds: Generating Level Crossings from Real and Synthetic Point Cloud Samples". *Automation in Construction* 130 (2021), p. 103839.

[121] J. E. Van Engelen and H. H. Hoos. "A Survey on Semi-Supervised Learning". *Machine learning* 109.2 (2020), pp. 373–440.

[122] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. "Matching Networks for One Shot Learning". In: *Proceedings of the 30th International Conference on Neural Information Processing System (NIPS)*. 2016, pp. 3637–3645.

[123] S. Wang, W. Liu, J. Wu, L. Cao, Q. Meng, and P. J. Kennedy. "Training Deep Neural Networks on Imbalanced Data Sets". In: *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2016, pp. 4368–4374.

[124] W. Wang, J. Dai, Z. Chen, Z. Huang, Z. Li, X. Zhu, X. Hu, T. Lu, L. Lu, H. Li, et al. "InternImage: Exploring Large-Scale Vision Foundation Models with Deformable Convolutions". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 14408–14419.

[125] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni. "Generalizing from a Few Examples: A Survey on Few-Shot Learning". *ACM Computing Surveys* 53.3 (2020), pp. 1–34.

[126] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. "Dynamic Graph CNN for Learning on Point Clouds". *ACM Transactions On Graphics (TOG)* 38.5 (2019), pp. 1–12.

[127] T.-H. Wu, Y.-C. Liu, Y.-K. Huang, H.-Y. Lee, H.-T. Su, P.-C. Huang, and W. H. Hsu. "ReDAL: Region-Based and Diversity-Aware Active Learning for Point Cloud Semantic Segmentation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 15510–15519.

[128] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. "3D ShapeNets: A Deep Representation for Volumetric Shapes". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1912–1920.

[129] S. Xia, D. Chen, R. Wang, J. Li, and X. Zhang. "Geometric Primitives in LiDAR Point Clouds: A Review". *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 13 (2020), pp. 685–707.

[130] Y. Xiao, Z. Chen, Z. Lin, J. Cao, Y. J. Zhang, Y. Lin, and C. Wang. "Merge-Swap Optimization Framework for Supervoxel Generation from Three-Dimensional Point Clouds". *Remote Sensing* 12.3 (2020), p. 473.

[131] Y. Xie, J. Tian, and X. X. Zhu. "Linking Points with Labels in 3D: A Review of Point Cloud Semantic Segmentation". *IEEE Geoscience and Remote Sensing Magazine* 8.4 (2020), pp. 38–59.

[132] I. Z. Yalniz, H. Jégou, K. Chen, M. Paluri, and D. Mahajan. *Billion-Scale Semi-Supervised Learning for Image Classification*. 2019. arXiv: `1905.00546 [cs.CV]`.

[133] X. Yan, J. Gao, C. Zheng, C. Zheng, R. Zhang, S. Cui, and Z. Li. "2DPASS: 2D Priors Assisted Semantic Segmentation on LiDAR Point Clouds". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer. 2022, pp. 677–695.

[134] B. Yang and L. Fang. "Automated Extraction of 3D Railway Tracks from Mobile Laser Scanning Point Clouds". *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 7.12 (2014), pp. 4750–4761.

[135] X. Ye, J. Li, H. Huang, L. Du, and X. Zhang. "3D Recurrent Neural Networks with Context Fusion for Point Cloud Semantic Segmentation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 403–417.

[136] J. Yin, J. Fang, D. Zhou, L. Zhang, C.-Z. Xu, J. Shen, and W. Wang. "Semi-Supervised 3D Object Detection with Proficient Teachers". In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXVIII*. Springer. 2022, pp. 727–743.

[137] C. You, Y. Zhou, R. Zhao, L. Staib, and J. S. Duncan. "SimCVD: Simple Contrastive Voxel-Wise Representation Distillation for Semi-Supervised Medical Image Segmentation". *IEEE Transactions on Medical Imaging* 41.9 (2022), pp. 2228–2237.

[138] W. Yuan, Y. Han, D. Guan, S. Lee, and Y.-K. Lee. "Initial Training Data Selection for Active Learning". In: *Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication*. 2011, pp. 1–7.

[139] X. Zhai, A. Oliver, A. Kolesnikov, and L. Beyer. "S4L: Self-Supervised Semi-Supervised Learning". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 1476–1485.

[140] Q. Zhan, Y. Liang, and Y. Xiao. "Color-Based Segmentation of Point Clouds". *Laser Scanning* 38.3 (2009), pp. 155–161.

[141] J. Zhang, H. Liu, and J. Lu. "A Semi-Supervised 3D Object Detection Method for Autonomous Driving". *Displays* 71 (2022), p. 102117.

[142] J. Zhang, X. Zhao, Z. Chen, and Z. Lu. "A Review of Deep Learning-Based Semantic Segmentation for Point Cloud". *IEEE Access* 7 (2019), pp. 179118–179133.

[143] R. Zhang, S. Yang, Q. Zhang, L. Xu, Y. He, and F. Zhang. "Graph-Based Few-Shot Learning with Transformed Feature Propagation and Optimal Class Allocation". *Neurocomputing* 470 (2022), pp. 247–256.

[144] W. Zhang, L. Zhu, J. Hallinan, S. Zhang, A. Makmur, Q. Cai, and B. C. Ooi. "BoostMIS: Boosting Medical Image Semi-Supervised Learning with Adaptive Pseudo Labeling and Informative Active Annotation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 20666–20676.

[145] Z. Zhang, B.-S. Hua, and S.-K. Yeung. "ShellNet: Efficient Point Cloud Convolutional Neural Networks Using Concentric Shells Statistics". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 1607–1616.

[146] C. Zhao, W. Zhou, L. Lu, and Q. Zhao. "Pooling Scores of Neighboring Points for Improved 3D Point Cloud Segmentation". In: *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2019, pp. 1475–1479.

[147] F. Zhao, J. Zhao, S. Yan, and J. Feng. "Dynamic Conditional Networks for Few-Shot Learning". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 19–35.

[148] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun. "Point Transformer". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 16259–16268.

[149] Z. Zhao, M. Liu, and K. Ramani. *DAR-Net: Dynamic Aggregation Network for Semantic Scene Segmentation*. 2019. arXiv: `1907.12022 [cs.CV]`.

[150] D. Zhou, J. Fang, X. Song, C. Guan, J. Yin, Y. Dai, and R. Yang. "IoU Loss for 2D/3D Object Detection". In: *2019 International Conference on 3D Vision (3DV)*. IEEE. 2019, pp. 85–94.

[151] J. Zhou, X. Wen, B. Ma, Y.-S. Liu, Y. Gao, Y. Fang, and Z. Han. *3D-OAE: Occlusion Auto-Encoders for Self-Supervised Learning on Point Clouds*. 2022. arXiv: `2203.14084 [cs.CV]`.

[152] R. Zou, X. Fan, C. Qian, W. Ye, P. Zhao, J. Tang, and H. Liu. "An Efficient and Accurate Method for Different Configurations Railway Extraction Based on Mobile Laser Scanning". *Remote Sensing* 11.24 (2019), p. 2929.

Appendix A

Object placement rules

Name	**Amount [probabilities]**	**Orientation**	**Placement**
Top bar	0-1 [0.2, 0.8]	Like x-axis	Center at x=0 and y=0, z=bit lower than max pole height
Pole without top bar	2-4 [0.6, 0.3, 0.1]	Like z-axis	Symmetrical (-x, x), y=0, z=0
Pole with small top bar†	2-4 [0.4, 0.2, 0.4]	Like z-axis	Symmetrical (-x, x), y=0, z=0
Pole with large top bar†	2-6 [0.2, 0.1, 0.4, 0.1, 0.2]	Like z-axis	Symmetrical (-x, x), y=0, z=0
Pole foundation	0-1 [0.3, 0.7] on every pole	Like pole	z = pole z - $\frac{1}{2}$ pole height, x/y same as pole
Drop post	0-1 [0.05, 0.95] on top bar	Like top bar	Center at x=0 and y=0, z=top bar z - $\frac{1}{2}$ top bar height
Steady arm at drop post	0-2 [0.1, 0.3, 0.6] at every pole/drop post	Perpendicular to drop post and aligned with top bar	z = bottom drop post, x/y same as pole
Steady arm at pole	0-2 [0.1, 0.8, 0.1] at every pole/drop post	Perpendicular to drop pole and aligned with top bar	z = $\frac{2}{3}$ pole height, x/y same as pole
Insulator	0-1 [0.1, 0.9] at every steady arm	Same as steady arm	At side where pole/drop post is, z/y same as steady arm

Messenger wire support on small top bar	0-4 [0.25, 0.125, 0.25, 0.125, 0.25]	Same as top bar	x = random on top bar, z/y same as top bar
Messenger wire support on large top bar	0-6 [0.2, 0.07, 0.2, 0.07, 0.2, 0.06, 0.2]	Same as top bar	x = random on top bar, z/y same as top bar
Wheel tension device	0-4 [0.3, 0.2, 0.3, 0.1, 0.1] on every pole	Same as pole	On front or back of post, z= $\frac{2}{3}$ drop post height, x=same as pole, y=pole y $\pm\frac{1}{2}$ pole width
Catenary wire	0-1 [0.1, 0.9] on every steady arm	Perpendicular to steady arm/topbar and aligned with z=0 plane	x = steady arm x $\pm\frac{1}{2}$ steady arm length, z/y same as steady arm
Stitch wire	0-1 [0.1, 0.9] on every catenary arch	Same as catenary wire	z=catenary wire z + 1m, x/y same as catenary wire
Dropper	0-2 [0.1, 0.7, 0.2] on every catenary wire	On plane through catenary wire and stitch wire	z = $\frac{1}{2}$(catenary wire z + stitch wire x), x/y same as catenary wire
Top tie	0-1 [0.1, 0.9] at every standalone pole‡	Perpendicular to pole and aligned with x-axis	On line y=0, z=bit lower than max pole height
Bracket	0-1 [0.1, 0.9] at every top tie	Same as pole, angle is same as in original data	So that both points are touching top tie and pole
Unlabelled (ground)	0-8 [0.05, 0.05, 0.1, 0.2, 0.2, 0.2, 0.1, 0.05, 0.05]	Along z=0 plane	At random positions, z=bottom of pole
Unlabelled (pole-like)	0-5 [0.1, 0.2, 0.2, 0.2, 0.2, 0.1]	Like z-axis	Near the outer poles
Unlabelled (wire-like)	0-6 [0.1, 0.1, 0.2, 0.2, 0.2, 0.1, 0.1]	Perpendicular to top bar and along plane z=0	z= 0.5m above top bar, x random

Table A.1: The object placement rules. †A small top bar is defined as a top bar with a width of less than 20m. A large top bar has a width greater or equal to 20m. ‡A standalone pole is defined as a pole without a top bar attached to it.