UNIVERSITY OF TWENTE.

Bachelor Thesis

Point cloud segmentation via active learning in the context of railway infrastructure

Jonas **Hentschel |**

14.02.2022

Supervisor: Dr. Faizan Ahmed

Critical Observer: Dr. Marcus Gerhold

University of Twente

Drienerlolaan 5

7522 NB Enschede, Netherlands

Acknowledgment

First and foremost I would like to thank my supervisor and critical observer, Dr. Faizan Ahmed and Dr. Marcus Gerhold. Through their support and feedback I was able to complete my research successfully. Their guidance helped steer my research in the right direction.

I would also like to thank everyone working at the AMI research group that provided me with technical and academic support throughout the research of this project. Their help in organizing computation environments, the preparation of data and constant feedback was crucial to the success of this research. In addition to this I would like to express my gratitude towards Strukton Rail that provided this opportunity to work on this challenge.

Lastly I would like to thank my family and friends that provided me with full support throughout the lengthy process of this project.

Abstract

This thesis presents insights into how the active learning framework can be effectively implemented into the point cloud semantic segmentation in the railway environment. The active learning framework sets out to drastically reduce the amount of labeled data needed to train an efficient segmentation model. It achieves this by intelligently selecting new batches of data for future training through metrics that tell the algorithm how useful this data would be for the model.

The overarching goal of this research is to determine how active learning compares to fully supervised learning. This will be determined by comparing their precision when running inference. In addition to this another goal is to explore the computation time required for active learning compared to fully supervised learning.

The main findings in this paper conclude that active learning can provide an alternative to fully supervised learning. We were able to train models with up to 76% (86% for fully supervised) accuracy with using only 16% labeled data. Furthermore by reducing the amount of labeled data that is fed into the training process it is also possible to reduce computation times for the training process by up to 43%.

Table of Contents

Table of Figures

1 Introduction

Trains are some of the most important transportation devices for millions of people, as well as some of the major drivers of global trade. Keeping the railways on which, those trains run, in working condition can be a very challenging and time-consuming task. In Europe alone there are over 200,000 km of railway that needs to be checked for maintenance.

Railway company Strukton has begun making advances in using Point Clouds to generate a 3D representation of many different parts of the railway like Catenary arches or Signal lights. This is done to check these components more easily and effectively for maintenance. They have tasked the Saxion University's Ambient Intelligence Group (AMI) to develop a deep learning model that is able to detect key railway infrastructure from large sets of point cloud scans. These scans are produced by LiDAR scanners mounted on trains and therefore are vast and require filtration to find the required components. The deep learning model for this job requires a lot of data to be trained accurately. The project described in this paper explores options to reduce the amount of data needed for accurate training through an active learning framework.

1.1 Preliminaries

The main data type that will be used in this project is point cloud data. This data is in its most basic form a set of x, y, and z coordinates. It is used to represent real-world objects in 3D space. Most often point cloud data is recorded by LiDAR scanners. LiDAR stands for laser imaging, detection, and ranging and is produced by a sensor sending out light beams in all sorts of directions and measuring the time to receive the reflected light beam in order to give an estimate of the reflective surface. This is also how the majority of the data used in this project was captured.
The concept that is used to run object detection is called deep learning. This is a category of machine learning that is based on artificial neural networks that imitate the human neurological system. Most commonly, a deep learning model is comprised of a number of different layers that are connected through weights. In each layer the raw input data passes through edits and transforms the data closer to a desired output

1.2 Research Objectives

The main focus of this research is the investigation of the active learning paradigm and how it can be implemented into point clouds. As traditional active learning models are mostly focused on a 2D image environment, it is a new challenge to fit this form of semi supervised learning into the point cloud environment. In this research an existing active learning framework for point cloud [18] will be used as a basis and adjusted throughout the entire process to fit a different dataset. We will investigate the adaptability and performance of this adjusted framework and discuss how it fits into the data Strukton needs.

As fully supervised training methods are currently used by the AMI research group, We will also investigate how this active learning framework performs compared to a fully supervised trained model. These goals result in the following main research question:

How can the active learning paradigm be effectively implemented into point cloud and how does it perform compared to fully supervised training?

In addition, and as a refinement to the main research question, two sub questions emerged during my research. One of those is related to color influence on the active learning cycle, which will be explained in more detail in later chapters. In addition to the color, the main argument for active learning is needing less data. This added additional sub questions to my research:

How do RGB values influence the active learning process?

How does the initial dataset influence the precision of the model?

2 State of the Art

2.1 Introduction

To train an AI (Artificial Intelligence) that can run segmentation, a dataset providing information on what the railroad components look like is needed so the AI can learn how to detect those effectively and reliably. This dataset is usually labeled by a human. This process is very labor intensive as the amount of inspection that needs to be done can be quite massive. Therefore, it is necessary to research and develop solutions that can optimize this data labelling process.

Thus, the goal of this chapter is to provide an overview of the state of the art in active learning for segmentation and object detection and explore possibilities supported by research. The first part of this chapter will focus on an introduction to methods and models used for basic object detection and segmentation. The second part of the chapter will surround the data labelling process and briefly explore the topic of unsupervised learning. This will include current solutions and models for self-learning AIs that can label and learn from data on their own. In the third part of the chapter, a short overview of the state of active learning will be given. And in the final part of the chapter, a conclusion will be given about active learning and how it positions itself in the world of AI-driven computer vision.

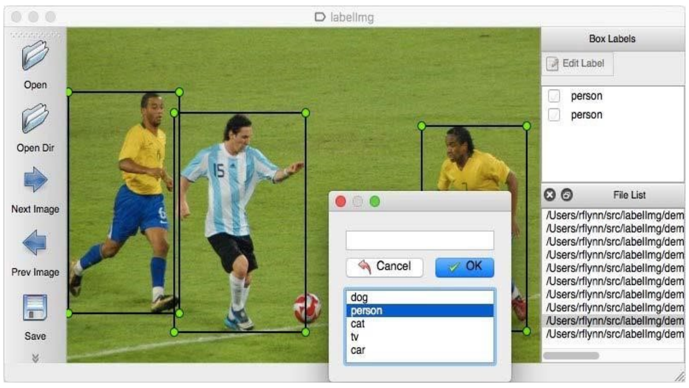
2.2 Computer Vision and Object Detection

Computer vision is about teaching a computer to recognize patterns. For this, Convolutional Neural Networks or short CNNs are most commonly used [15]. CNNs are a mimic of neural networks just like in our brains that are rearranged by a training process to be able to recognize the patterns we want them to recognize. O’Shea et al. [9] describe the structure of a CNN as being comprised of three types of layers, which include convolutional layers, pooling layers and fully connected layers. Yamashita et al. [12] have also described those three layers in more detail. First of all, the convolutional layer is described as the most fundamental of layers in a CNN as it processes the input fed into the network, most commonly images, and computes this input and sends it to the next layer. The computation done in more detail is a repeating process of representing each pixel of the input in an array of numbers, called feature maps. They are then tweaked through a filter each convolution to improve the structure of the network to recognize what pattern it should recognize. A CNN usually has multiple convolutional layers, and each is handling a different filter, these tell each layer what pattern to look for. Some of these filters might be used to recognize edges in an image while others might be used to identify texture and brightness levels. Brownlee [2] describes these layers in their research in a similar way they adds to the research of Yamashita et. al. [12], by introducing flattening and densing processes which are added to improve the structure of the output that the previous layers produce. Albawi et. al. [1] describe the pooling layer in more detail. Pooling layers are used to reduce the size of feature maps to make the network more efficient. Pooling layers take the output of multiple nodes in the convolutional layer and average these out to combine them. These layers are usually placed in between convolutional layers. Liu et. al. [8] also describe the pooling layer as an essential step within CNN networks that combine several pixels within a feature map to a combined pixel with added weight and bias.

The learning process is usually done in a fully supervised way. This means that a premade set of data is manually labeled, which means that a human shows the computer exactly what needs to be detected. The size of this dataset can substantially based on the application. This data is then fed into the CNN and through all the convolutional layers, where it is then output into a weighted model. Alzubaidi et. al. [15] describe recurrent neural networks (RNNs), a different approach to neural networks as previously mentioned research. RNNs are often used in speech-processing applications. This is due to their semantic nature which lets them understand context better, which is necessary for speech and text usage. For image processing however, this context awareness is not crucial. Lao et. al [18] identify that these RNNs have a major problem of poor scalability due to their semantic and parallel nature. Scaling these networks results in an exponential increase in computing power and training time. This adds to RNNs not being feasible for object detection with a multitude of labels. As a large amount of data is needed for this task.

2.3 Data labelling and unsupervised learning

For every Neural network training, data is necessary. Therefore, the way we label data is an integral part of getting accurate results out of the models.
The data labelling process for a network designed to be used for simple image object detection is very straightforward. Usually, software like LabelImg (seen in Figure 1) is used to draw bounding boxes around the objects that need to be detected [13].



/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem
/Users/rflynn/src/labelImg/dem

Figure 1 Screenshot of LabelIMG [Source: https://github.com/heartexlabs/labelImg]

On a more technical level, Guo et. al. [5] describe this as a set of coordinates that are noted and related to a label, like “human” or “cat”, which is then connected with the corresponding image. Both of these are then fed into and read by the neural network.

This data labelling process is quite labor intensive, as for a normal dataset hundreds if not thousands of images have to be labeled, which is a very numbing and error-prone task for a human. This is proven by Anish et. al. [16] who have identified errors of at least 3.3% within the 10 most commonly used computer vision datasets. They also showed that these mislabeled datasets have a measurable effect on model performance. This is backed by Yang et. al. [17] who also concluded that mislabeled data has a negative impact on model performance. To solve this issue, unsupervised learning comes into play. When using clustering, which is one form of unsupervised learning for a Neural Network, no specific labelled data is provided, which means that the Neural Network is just fed the raw data input. No desired output is specified, and the Neural Network is just trying to recognize patterns and structures on its own [7][4]. The network sorts the data into patterns and structures it recognizes to be similar in order to provide a set of detected classes.

2.4 The Active Learning paradigm

Active learning slots right in the middle between supervised and unsupervised learning. Active learning uses only a small set of hand-labelled data. An initial model is then trained on this small batch of data. Roy et. al. [11] use this initial model to label the next set of data for training. Haussmann et. al. [6] showed that active learning can not only drastically reduce the time and effort spent manually labeling a dataset but also improve accuracy. Their active learning model was able to achieve a 3x improvement in accuracy over their manually labeled dataset. This is backed by Choi et al. [3] as their active learning models also showed improved accuracy with less label budget.

Active learning works by probabilistic modelling of the output of the network. In conventional neural networks that do object detection, the bounding box that marks the detection object is represented by the x and y coordinates of the center of the bounding box as well as the width and height of the box. In active learning, these values are replaced by the mean, the variance and the weight [3]. These are then used to find the best data for the next revolutions of active learning processes.
Yu et al. [14] use a consistency-based approach to active learning, meaning that for each data point a reference prediction is compared with their corresponding predictions and analyzed if they are consistent with each other to evaluate the informativeness of these data points for further training.

2.5 Conclusion and Discussion of Literature

The goal of this chapter is to provide insight into the active learning paradigm within machine learning. To fully understand this matter first, a look at the general idea of object detection and machine learning is necessary. The research here showed that CNNs are trained to recognize patterns and structures within sets of data and to output them in a weighted model at the end, which can be used to perform object detection. The majority of research for object detection describes a similar structure of their used CNNs consisting of three layers: convolutional, pooling and fully connected [9][12][1][5][8].

Data labeling is a consistent factor through all research regarding machine learning. The research discussed in the second part of this chapter [13][7][4] showed that the data labelling process, or more specifically the acquisition of a sufficient amount of labelled data, often acted as a bottleneck for many machine learning paradigms. This is why unsupervised learning methods are proposed and researched in the papers found. The active learning paradigm is a compromise between manual data labeling processes and completely unsupervised learning. The research for active learning [11][6][3] showed that the amount of labeled data needed can be drastically reduced by using active learning methods.

Research on the topic of active learning was limited by its novelty as this paradigm has not been explored until the last few years. Additionally, machine learning methods are quite diverse. CNN's, which this literature review focused on are not the only form of machine learning algorithms [1]. This variety of base algorithms means that subsequent forms of unsupervised and active learning do not work on the same ground basis, making it hard to compare different approaches. The type of data that needs to be recognized by the algorithms might also change from images to 3D models to text and much more. Even though the approaches might be similar the output and precisions provided by the research might differ drastically between those different data inputs.

An interesting future research direction could be to analyze what facets of data make it valuable for active learning processes. This might help with selecting which dataset might be best suited for the initial base model. Another valuable research direction could be to analyze the optimal size of the base dataset for different approaches of active learning as this provides insight into model performances and efficiency.

3. Methodology

In this chapter, the tools and processes related to active learning in the point cloud space are explained in detail. To fully understand how active learning can be applied to reduce the data needed for deep learning models, we have to first look at the general approach to deep learning and how it works. The process of active learning for point cloud data can be divided into two separate stages. The first stage consists of data preparation and processing [18]. In this stage the raw data recorded by the LiDAR sensors is analyzed in a variety of preprocessing steps. This is done in order to fit the raw data into a format that is suitable for active learning processes. The second stage of the active learning process is the actual training process. This usually consists of training an initial deep learning model and using this as a baseline to run multiple iterations of active learning processes.

3.1 Deep learning training process

Deep learning was introduced in Chapters 1 and 2 already. But it is necessary to look more into the training and learning process of deep learning models. Deep learning models are based on artificial neural networks or specifically convolutional neural networks (CNNs). These neural networks imitate the human neuron system in the form that they are made up of layers of nodes that are connected via weights [5], a visualization of this can be seen in Figure 2.

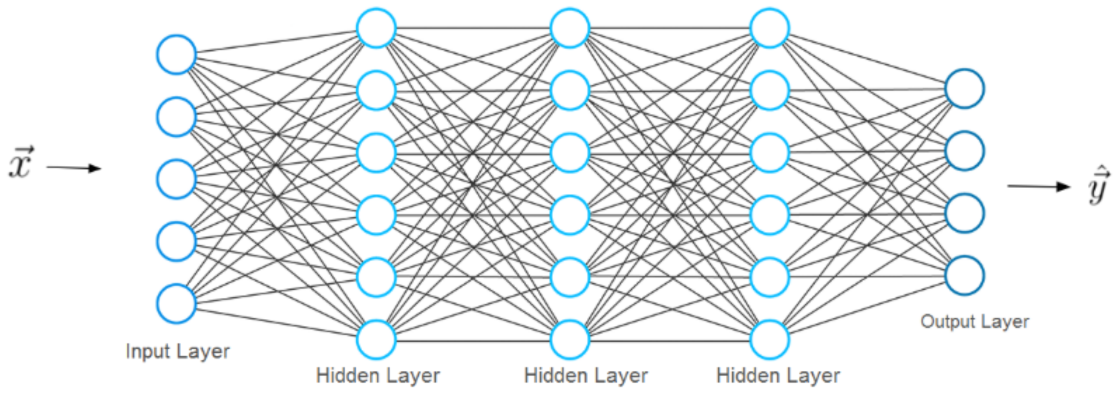


Figure 2 Structure of CNN

The input data fed into a CNN is passed through each layer of neurons and modified with each pass through a node. Modern CNNs can consist of millions of these nodes. The value that is passed to a node is multiplied by the weight connecting these nodes. The training process, therefore, changes the way the nodes are connected as well as the weight values that connect those nodes. Typically training processes consist of multiple iterations. In each iteration, the network and its weight are adjusted accordingly to generate the desired output. The performance of such models can be measured by using the model to make predictions on data that was not used in the training process of the model and is therefore unknown to it. These predictions are then compared with a ground truth of the dataset (typically referred to as a validation set). From this comparison, you can then calculate a mean intersection over union (MIoU) which indicates how much of the unknown data was correctly predicted by the model. This MIoU will also be the main metric used to judge the performance of a given model in this research.

3.2 Point cloud semantic segmentation

Point cloud segmentation is one case in which deep learning can be applied. Point clouds are comprised of a set of points that make up a 3D model. The goal of semantic segmentation is to have the AI label each point to a class. Initially, training, data that was manually labeled by humans, is used to train the AI, so it understands how to separate objects. Then data unknown to the AI is passed through and it attempts to label each object, an example of a segmented point cloud can be seen in Figure 3. In my project, this means labeling the different track components.

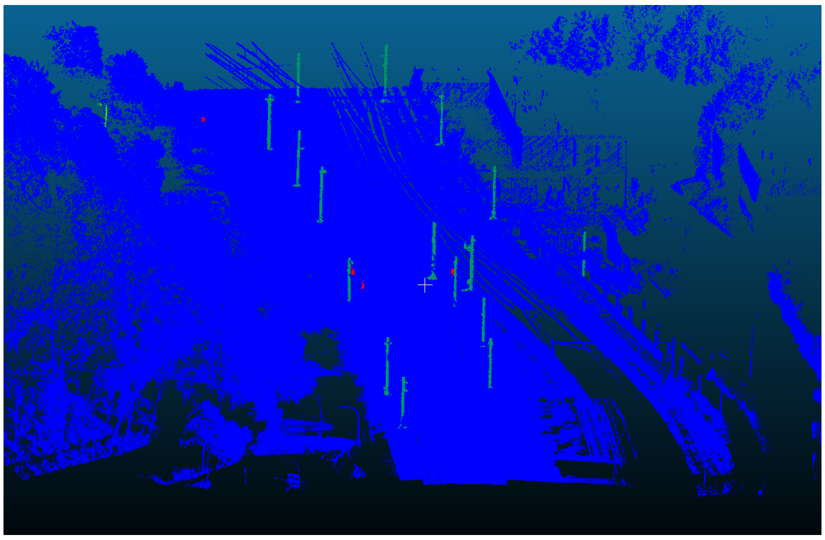


Figure 3 Example of segmented point cloud scan

This segmentation is done by giving each pixel a score for each label [20]. These scores represent the confidence of the AI, how sure it is that a pixel belongs to the given class. This score is based on a multitude of different components, which are described in later chapters. These segmentations work by using the context in which each pixel is in to evaluate its score.

3.3 Active learning

The active learning framework, visualized in Figure 4, uses a scene-based approach [18]. In traditional fully supervised training, a whole dataset is taken for training. The amount of labeled data needed for an accurate training result is enormous. That is why active learning aims to reduce the amount of data needed. Active learning utilizes an initial model that was trained on a small amount of labeled data. This initial model is then utilized to evaluate the non-labeled data and learn from it. In more detail, this means that in each iteration the model selects a new batch of data that it will train on until it provides the required accuracy [3].

When considering a whole dataset scan, it is clear that not all labeled regions contribute effectively to the training process. That is why the active learning framework subdivides the data into regions that can then be evaluated individually based on how much new information they provide to improve the performance of the model. To evaluate the information that each region provides to the AI, three steps are done in each active learning iteration: SoftMax entropy, color discontinuity, and structural complexity.

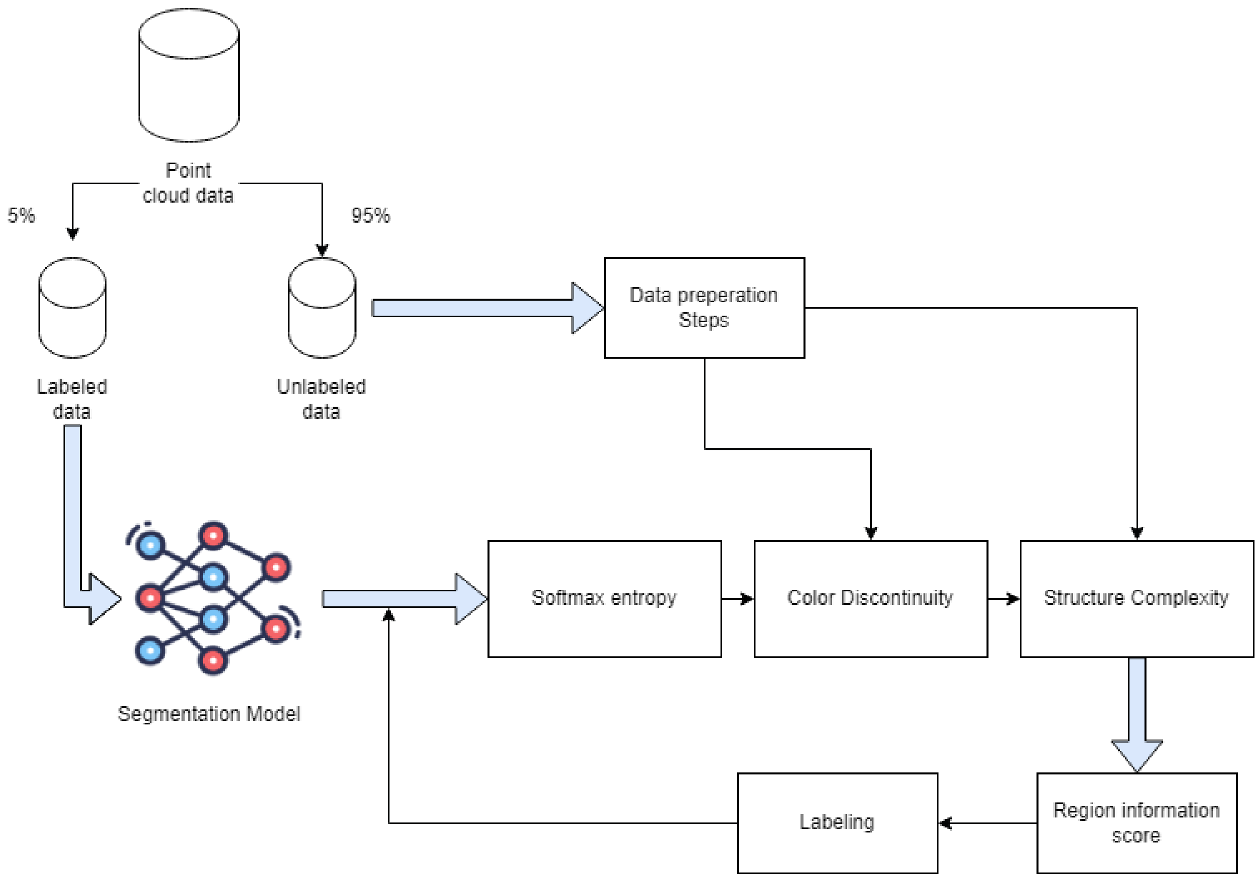


Figure 4 Active learning pipeline

The description in the following subsections is based on [18].

3.3.1 SoftMax Entropy

The SoftMax Entropy [18] uses the model trained in the last active learning to obtain the class probability of all regions. The probabilities are then normalized and analyzed based on their uncertainty.

Class	Probability
Catenary arch	0.92
Signal	0.05
Relay Cabinet	0.02

Class	Probability
Catenary arch	0.45
Signal	0.31
Relay Cabinet	0.24

Lower uncertainty Higher uncertainty

Figure 5 Example of SoftMax Entropy

In the tables above (Figure 5) a clear example of how this uncertainty is analyzed can be seen. Essentially, it shows the distance between the probabilities of different predictions. The bigger the gap is, the lower the uncertainty. If the probabilities are closely grouped together, the AI is more uncertain of the data. This in turn makes it more useful for the improvement of the segmentation model.

3.3.2 Color Discontinuity

Color Discontinuity is a useful tool to analyze differences and information scores in active learning. It utilizes RGB values, which are color values, assigned to each point cloud coordinate, thus giving us a colored 3D representation (Figure 6) of the environment.

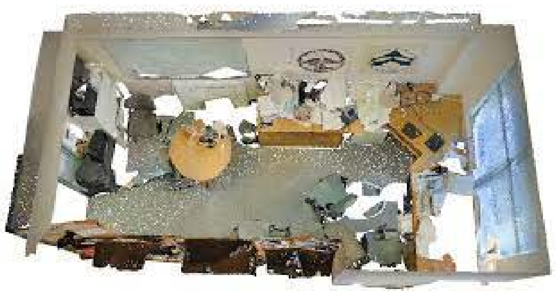


Figure 6 Point cloud scan with RGB values [20]

Each point’s color intensity value (see section 4.3) is compared to its nearest neighbors’ values, this way a color difference score is calculated. The color difference score for all points in a region are then averaged and a color discontinuity score for all regions is calculated.

3.3.3 Structural complexity

Structural complexity is another indicator that is taken into account for point cloud active learning. For each point cloud the surface variation computed in the data preparation steps (see section 4.3) is utilized. These values give us valuable information about structural and geometric complexity, as sharp edges or corners are good indicators for different objects. Especially for railway infrastructure, the surface structure of the different classes varies drastically. An example of this is shown in Figure 7.

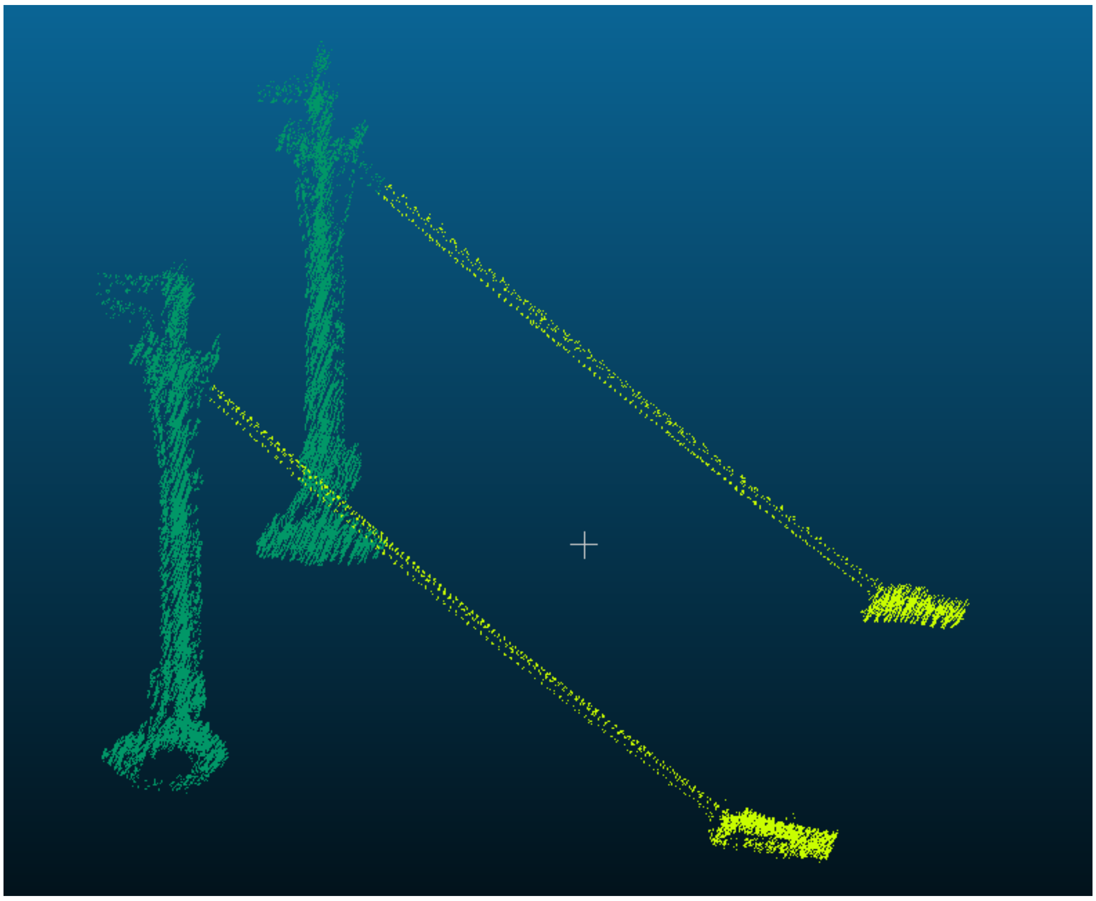


Figure 7 Catenary arch pole and Tension rod and foundation

These three indicators: Softmax entropy, color discontinuity and structural complexity, are then used to create a list of the most useful regions, which are then used for active learning selections. These scores are calculated iteratively using the newest model trained in each active learning cycle, and the region list is updated accordingly. After each iteration, a checkpoint for the model is saved, from which the best checkpoint can be loaded to evaluate the model. In order to evaluate the model, a small validation set of the unlabelled data is reserved and inference is run on it.

4 Specification

4.1 Business Requirements

The overarching goal of this project is to explore a method of reducing the amount of labelled data needed in order to train a precise semantic segmentation model for the detection of railroad infrastructure. This ties into a larger project of Strukton that aims to create digital twins of real-world railway infrastructure environments. This task was handed to AMI research group at Saxion University.

In Figure 8 below you can see the overall architecture proposed by the AMI research group that aims to solve the problem of digital twinning. The active learning research described in this thesis is highlighted in red.

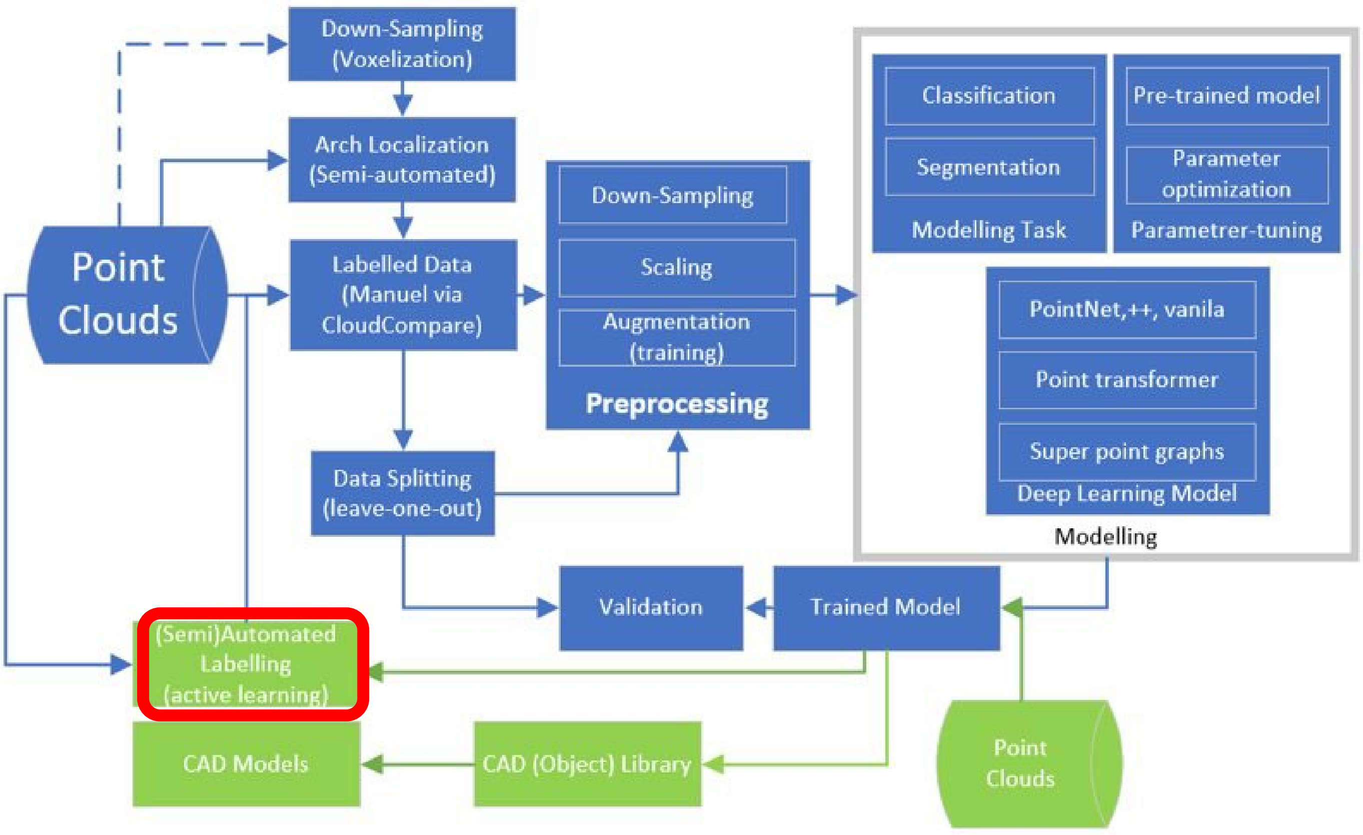


Figure 8 AMI Project Pipeline

To judge the results of this project, we had to set goals or requirements that this project aims to meet. Firstly, the project should develop and implement a pipeline that introduces active learning into the architecture created by previous research. This pipeline needs to be adaptable to changing datasets. Secondly this pipeline should produce enough results/measurements in order to judge the performance of active learning compared to fully supervised learning. Thirdly the project should highlight necessary steps that need to be taken to implement and improve the pipeline to be used by Strukton rail in the future.

4.1.1 Resources

As deep learning applications are notoriously computing power and memory demanding, we were provided with a Linux environment on university servers for training and testing purposes. Through this we were able to utilize powerful NVIDIA GPUs that are required for these types of applications.

Strukton provided datasets of the railway environment which were previously analysed and manually labelled by the researchers. These datasets are the key datapoints which were utilized throughout the course of this paper. Additionally, to the Strukton railway data, we utilized a publicly available dataset named S3DIS [19], which is a dataset made available by Armeni et al. from Stanford University. This dataset contains various indoor point cloud scans with rgb data included.

4.2 Data understanding

The dataset provided by Strukton is a LiDAR scan of roughly 1km of railway environment. These are stored in las files which is an efficient format for saving LiDAR scanned point clouds. In Figure 9, one of these files can be seen visualized. This visualization also shows the labelled components. This labelling process was done by the AMI researchers manually. Here, each color represents a different label.

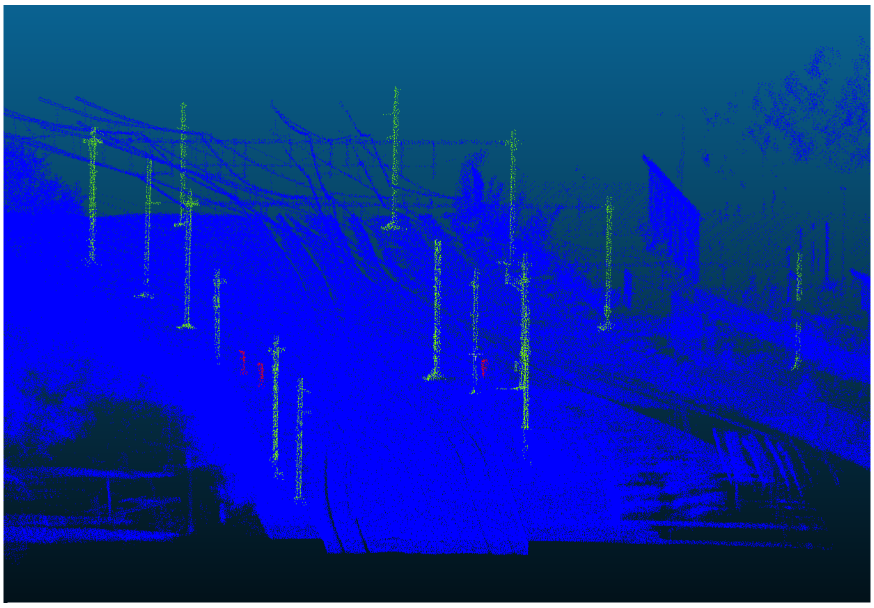


Figure 9 Example of labeled point cloud scan

The dataset includes 7 different labels and components (Figure 10) that need to be labelled and detected by the active learning algorithm, one of which being the background/unlabelled points. It is important to note that all these components are not represented equally within the dataset, due to some of the classes being found more frequently alongside a train track. The reduced sample size for some classes could potentially result in varying accuracy.

ID	Label
00	Background
01	Catenary arch poles
02	Street light
03	Tension rod
04	Signal
05	Relay cabinet
06	Signs

Figure 10 Table of labels

The entire dataset consists of 21 las files (Figure 11), each of different sizes. In total the number of points accumulates to about 206 million. Even though this data requires a large amount of physical disk space (10.07GB), it only covers about 1km of railway. Regardless of the large data size, some objects are only represented 3-5 times within the entire dataset.

File	Num. of Points	File	Num. of Points
poly_00.las	23,024,515	poly_14.las	6,943,666
poly_01.las	16,513,911	poly_15.las	7,308,539
poly_02.las	16,892,955	poly_16.las	6,733,068
poly_03.las	15,756,931	poly_17.las	6,766,072
poly_04.las	12,545,790	poly_18.las	6,704,505
poly_06.las	15,978,495	poly_19.las	6,457,386
poly_07.las	8,385,019	poly_20.las	6,564,614
poly_10.las	8,167,105	poly_21.las	6,419,163
poly_11.las	7,059,190	poly_22.las	6,696,304
poly_12.las	6,672,750	poly_25.las	7,648,357
poly_13.las	6,793,630	TOTAL	206,031,965

Figure 11 Point cloud sizes

4.2.1 Data conversion

The data format required by the active learning pipeline differs from the data format that the original dataset is in. As mentioned above, the dataset provided by Strukton is stored in a .las format, this is not the format the active learning pipeline requires. Therefore, a conversion algorithm was made. The desired data format can be seen in Figure 12 where poly_00.txt contains all the x, y and z coordinates in the datafile and the folder annotations contains a txt file with all the coordinates belonging to a specific class.

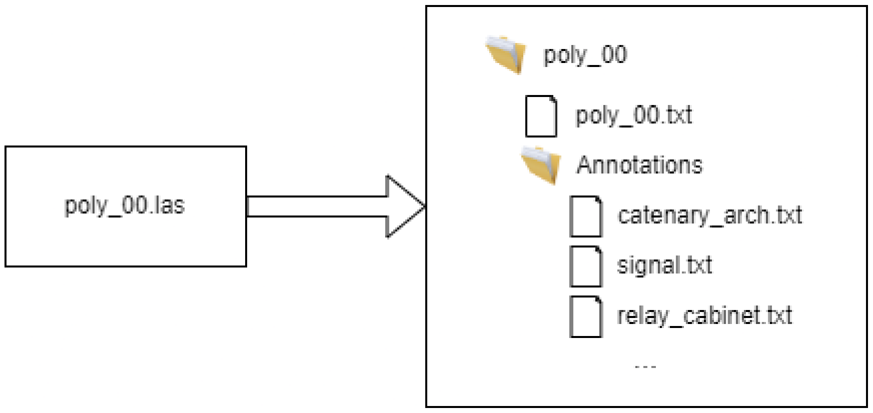


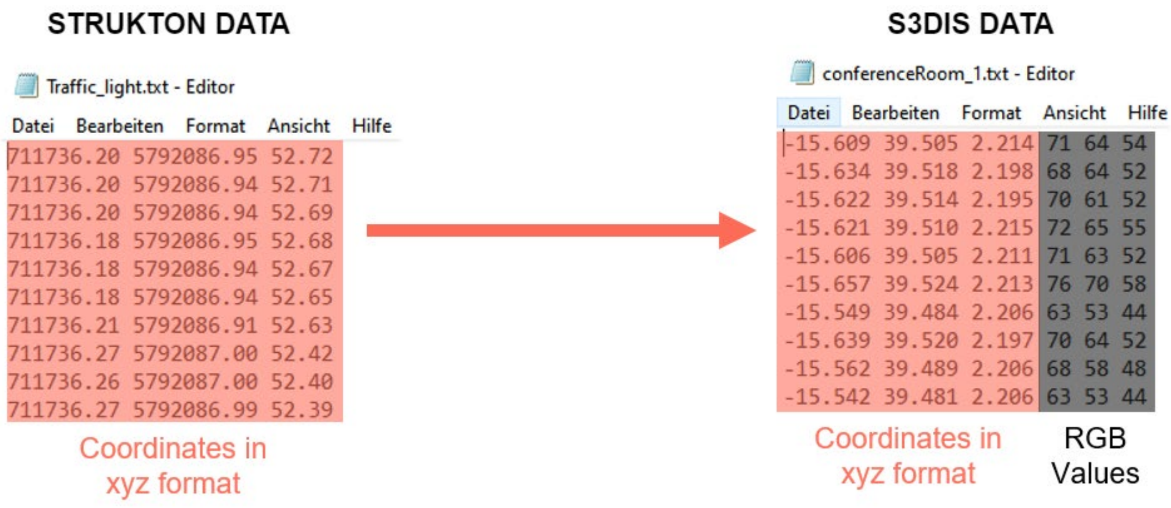
Figure 12 File structure

This was done utilizing the python library laspy which lets you load in and manipulate .las files within a python environment. We extracted the coordinates of each class by using the label ids. After extracting the data, numpy, which is a python library to manipulate and calculate arrays and other forms of numerical data, was used to write each set of coordinates to a .txt file.

A major disadvantage of utilizing this file structure is the physical storage requirements of each format. The filetype of .las is a very efficient format in terms of physical storage. The whole dataset of about 206 million points requires 10.07GBs of storage. The .txt format is inefficient and requires a substantial amount of more physical storage than the .las format. Additionally, the conversion script duplicates many of the x,y and z data points, as some are found in the poly_00.txt file as well as the annotations text files. Given these two disadvantages, the entire dataset requires 29.2GBs after the conversion step. This might pose a problem for Strukton rails operation, especially when considering scalability.

4.2.2 RGB values

A major problem of the dataset provided by Strukton is that it is missing RGB values. This means that no color information is available for the active learning algorithm. As mentioned in section 3. color variation is one of the factors that the active learning algorithm utilizes to make a judgment on the information value of each region. Nonetheless, it is only one of a multitude of factors that the active learning algorithm takes into account. Without the RGB values, however, the active learning model is not able to run without running into errors. This is due to the model structure and the format of its layers. To run the active learning model without RGB values, a total overhaul of the entire model and data readout process would be required, which is out of the scope of this project. To solve this issue, we attached uniform RGB values to each coordinate, effectively creating artificial RGB data. In Figure 13, an example of the Strukton data structure and the data structure of the S3DIS [19] dataset, which includes RGB values can be seen:


STRUKTON DATA
Traffic_light.txt - Editor
Datei Bearbeiten Format Ansicht Hilfe
Coordinates in xyz format
S3DIS DATA
conferenceRoom_1.txt - Editor
Datei Bearbeiten Format Ansicht Hilfe
Coordinates in xyz format
RGB Values

```
711736.20 5792086.95 52.72
711736.20 5792086.94 52.71
711736.20 5792086.94 52.69
711736.18 5792086.95 52.68
711736.18 5792086.94 52.67
711736.18 5792086.94 52.65
711736.21 5792086.91 52.63
711736.27 5792087.00 52.42
711736.26 5792087.00 52.40
711736.27 5792086.99 52.39
```

```
-15.609 39.505 2.214 71 64 54
-15.634 39.518 2.198 68 64 52
-15.622 39.514 2.195 70 61 52
-15.621 39.510 2.215 72 65 55
-15.606 39.505 2.211 71 63 52
-15.657 39.524 2.213 76 70 58
-15.549 39.484 2.206 63 53 44
-15.639 39.520 2.197 70 64 52
-15.562 39.489 2.206 68 58 48
-15.542 39.481 2.206 63 53 44
```

Figure 13 Example of rgb values in point cloud data

In the evaluation section of this paper, we will analyse and compare the effects color values have on the performance of the active learning algorithm. This is done by utilizing the S3DIS [19] dataset and running it through the active learning pipeline, once with proper RGB values and once with all RGB values set to a uniform value; practically eliminating the influence of the rgb values on the training process.

4.3 Active learning pipeline

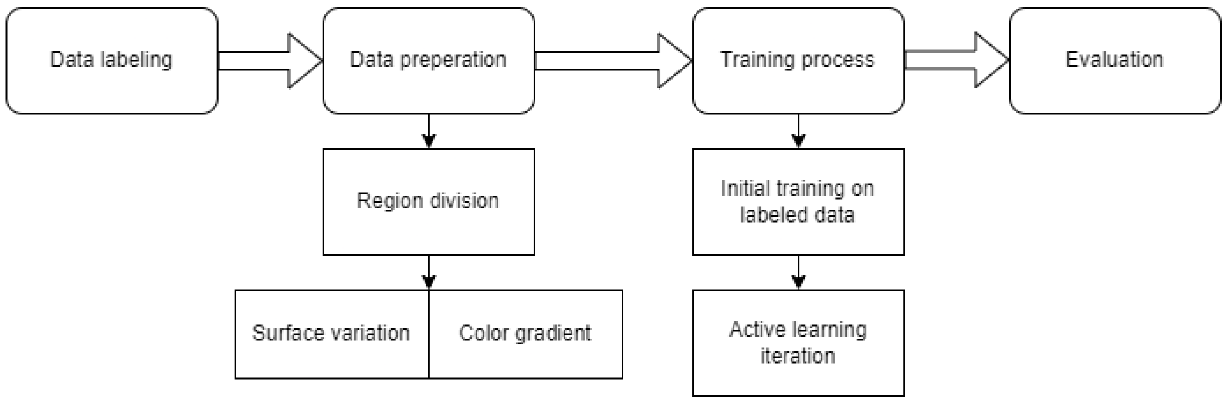


Figure 14 Basic active learning pipeline

The planned pipeline for active learning (Figure 14) was implemented into the Linux environment. This pipeline includes the data labelling step which is done by the researchers manually using the open-source tool cloud compare. These labelled data files are then stored in .las files for further usage. In the data preparation step, we first complete a region division step. The division subdivides each point cloud scan into smaller subregions which can then later be evaluated in the training process step. This subdivision is done using the VCCS algorithm developed by Papon et al. [21]. After subdividing the point cloud scenes, two different analyses are run to get more information that is later required by the active learning algorithm.

First surface variation analysis, which uses a combination of edge extraction [22] and feature extraction [23]. This step looks at a small amount of neighbours for each point and calculates how significant their shift in position and angle is in relation to the origin point. By doing this we can assign points close to edges a higher value, thus indicating a different object / classification.

The second data preparation step is the analysis of color discontinuity, this step could not be completed for the strukton dataset due to its missing RGB values. The effects of this will be analyzed in a later chapter. This analysis works similar to the structural complexity step. This means that we look at the neighbours of points and calculate a score based on the difference in color that each of these points have towards the origin point.

After completing these two data preparation steps we load the data into the active learning algorithm. First the algorithm loads a pre-set amount of labelled data and trains its initial model. After this step is complete, the first active learning iteration takes place and the algorithm calculates the information score for each region, based on the values described earlier. It then selects the best region to learn from and runs its training process again. This is done throughout multiple iterations.

After finishing this training process we ran simple inference with the best checkpoint model. During this step we also calculate the MIoU for each class.

5 Realization and Evaluation

In this chapter, the active learning pipeline will be implemented, and its results analyzed. First, we will describe the process of implementing the data preparation algorithms as well as the training and testing algorithms into the environment. Technical limitations concerning hardware and software will also be discussed during this chapter. Additionally, the results and their collection process will be described.

5.1 Active learning pipeline implementation

The implementation of the active learning framework required a Linux environment as many of the dependencies were only available on Linux. Utilizing the jupyter environment, all necessary python libraries were installed using anaconda. This part ran into very little issues as the installation process for python packages using conda runs mostly automatic. A first limitation arose when trying to install the Linux packages that were needed. These packages were Sparsehash, which is an efficient HashMap package and PCL (Point Cloud Library). As we was using a jupyter environment running on university servers, we had no root access, which means we had no administrative rights in the system. Python packages are usually installed using the" apt-install" command. This, however, requires sudo privileges. This issue was solved by building and installing the required packages from source. This added onto the time needed to set up the environment but proved to be an effective solution in the end.

After finishing the environment setup, the data preparation steps were taken. This included the region division, surface variation and color gradient. The region division step is done using a C++ program included in the active learning framework. This step required the C++ program to be compiled and built from its source code. This ran without issues.

Next the surface variation and color gradient analyses were run. These are simple python scripts that needed to be pointed to the dataset and executed. This step concluded the data preparation step. The next step was the training process which consisted of the initial training iteration on a limited amount of labeled data and the active learning iterations. These required an extensive amount of parameters to be tuned. The values of these parameters were chosen to provide the required results but also to be feasible in terms of computing power and training time. The parameters required a lot of tuning as a big limitation here was memory usage. First tests often ended in out of memory errors. To resolve this issue, a lot of tuning of the parameters was required for the active learning cycle to run properly. This tuning resulted into the following parameters being used.

Active learning runs

Model	Spvcnn
Training_epoch	20
Finetune_epoch	10
Train_batch_size	4
Val_batch_size	10
Distributed_training	false
Max_active_iterations	7
Initial_data	5%,9%,12%, 16%

Supervised runs

Model	Spvcnn
Training_epoch	80
Train_batch_size	4
Val_batch_size	10
Distributed_training	false

Figure 15 Active learning parameters

After a successful training process, the training log and checkpoints were saved to be used later on for validation. The inference validation step also calculates and logs MIoU on a per class basis.

5.2 Evaluation

5.2.1 Initial training iteration

To analyze the behavior of the model in the initial training iteration, we can look at the MIoU logged for each epoch (Figure 16). These initial training steps were done in a fully supervised manner, so all the data was labelled and no active learning had begun yet.

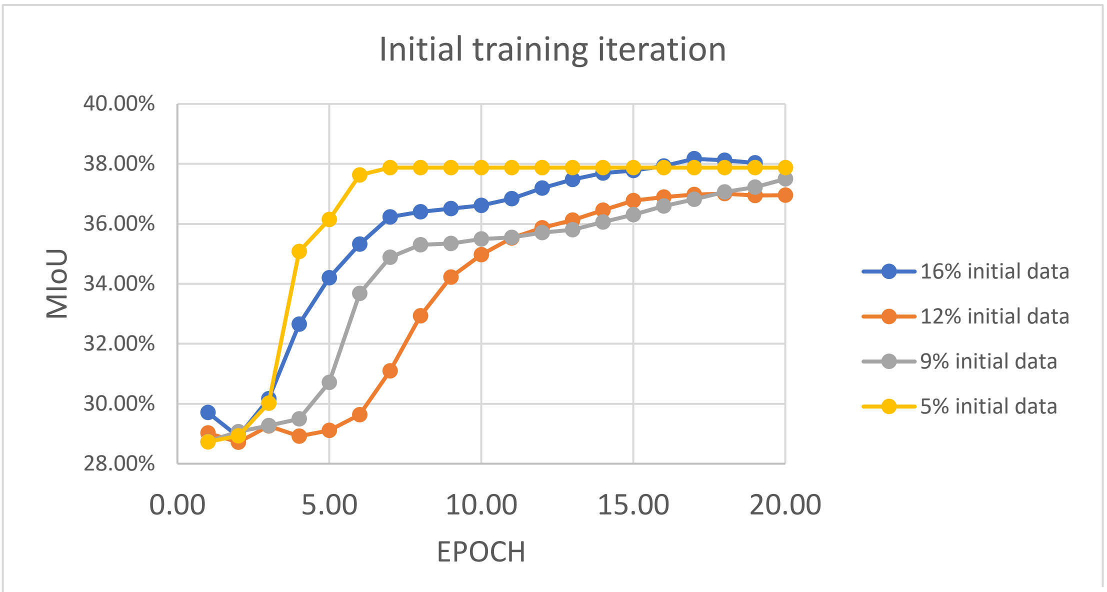


Figure 16 Initial training iteration performance

Looking at the data we can see that all runs, regardless of how much data was used, end up at around the same accuracy of 37%-38%. Additionally, it is clear that the amount of data used does not directly correspond to the final accuracy. A difference between the different runs can be seen when looking at how the training process behaved throughout the epochs.

The run with 5% initial data shows a steep learning curve within the first 8 epochs and then flattens out. This is contrasted by the other runs of 9%,12%and 16% which show a more steady learning curve. This might be caused by the type of data that is loaded in the beginning. As we needed to make sure that each run, regardless of amount of data, had some labeled data for all classes, the 5% run only had a very limited amount of data for each class especially the less represented ones, like relay cabinets and tension rods. In the first few epochs it has seen all of the classes once and does not find any new valuable data to improve on in later epochs. For the other runs with more labeled data available, the data is more spread out and therefore takes more epochs to learn.

5.2.2 Active learning and fully supervised learning

Analyzing the performance of active learning against fully supervised learning, it is best to do this on a per class basis. In figure 17 a comparison between each training run per initially used data can be seen:

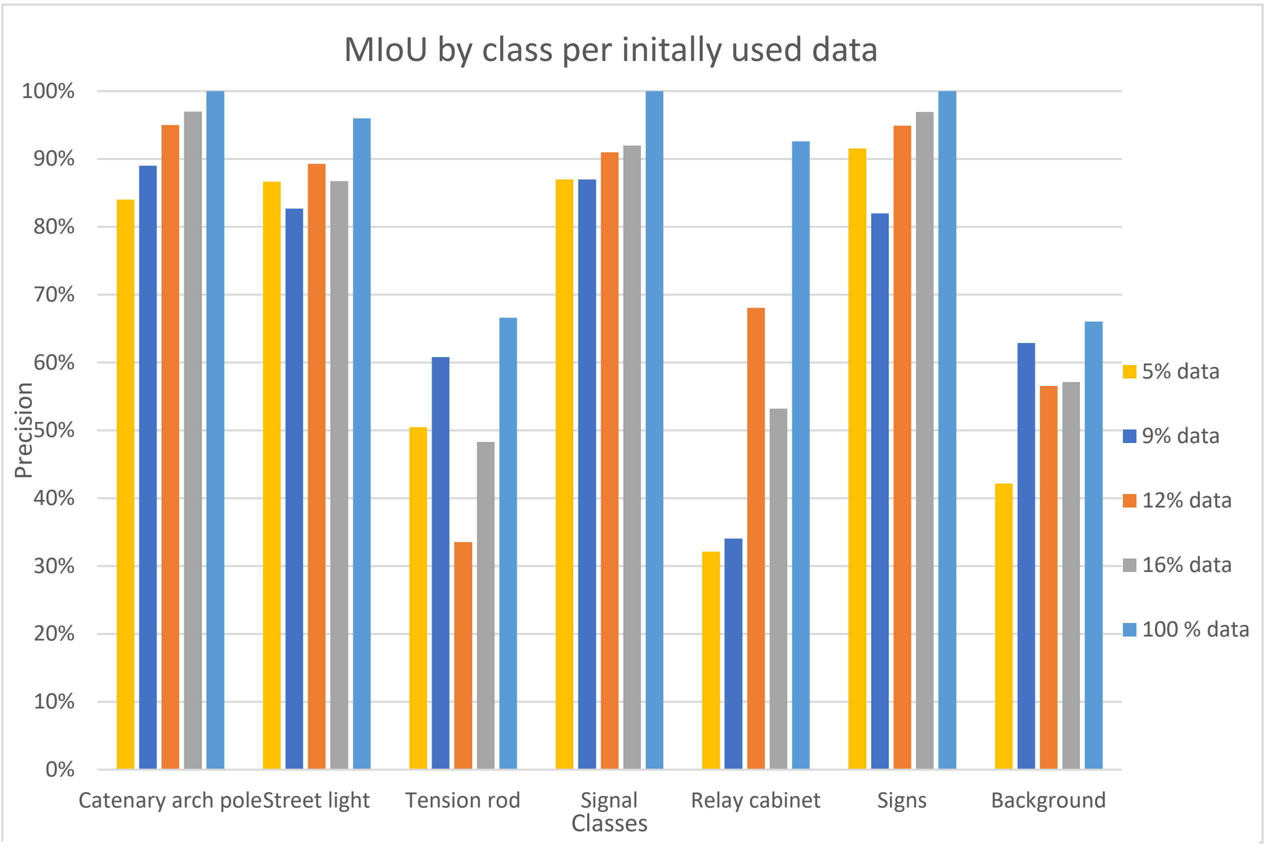


Figure 17 MIoU by class

First of all, it is clear some classes seem to be consistently less well detected than others. This is mainly due to classes being less represented in the data. Relay cabinets and tension rods are not as often found in the data as the other objects.

Looking at the difference between fully supervised (100% data) and active learning (5%, 9%, 12%, 16%), we can see that fully supervised training tends to outperform all active learning runs, regardless of class. The mean accuracy for fully supervised training was 89%, when the 16% active learning run had a mean accuracy of 76%, which was the highest of all active learning runs.
All classes that are found often within the data like catenary arch poles, signs, street lights and signals, mostly have a pretty clear slope towards more data resulting in more accuracy. Looking at the catenary arch poles, we can obsereve a linear increase in precision with rising intial data percentages throughout all active learning runs. Fully supervised learning only proves to increase the precision by a small amount, even though it is using vastly more labelled data. This result is a clear argument to choose active learning over fully supervised learning.

Looking at the classes that have an overall lower accuracy, we can see that the results are more inconclusive and tend to fluctuate as there is less data available for these classes. Training a deep learning model is a variable process and results can change each training cycle slightly. Therefore, these results might be explained by the active learning algorithm not selecting the right data. Even though this selection process is not random as explained in chapter 3, there is still a chance it might miss more interesting and useful data points. This means it is possible that in some runs it just did not select the right data that included these classes. To improve upon this, it is necessary to include data of all classes more evenly throughout the dataset.

5.2.3 Influence of initial data on training process

To find out how the amount of initial data influences the training process, the training process over the entire duration is shown in Figure 17.

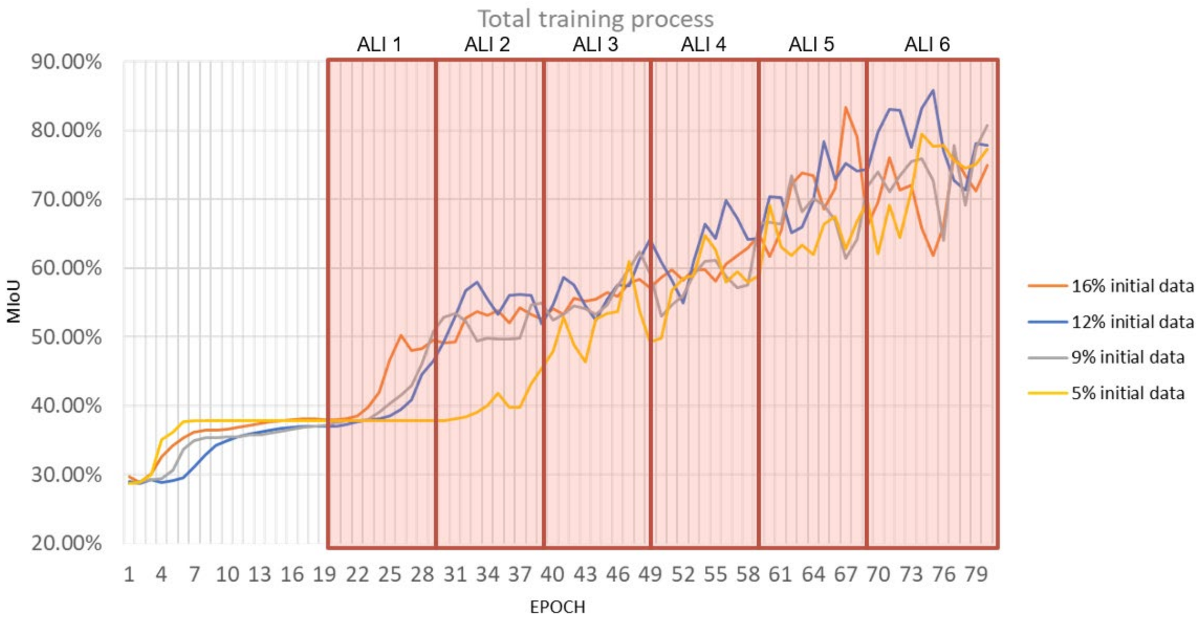


Figure 17 Active learning training process

Looking at the graph we can see that all runs share a similar initial training iteration. After the initial training they all show a somewhat linear increase in precision. The differences between each run can mainly be seen in their variance. The runs with less initial data tend to have a more unstable training process and their precision tends to fluctuate more. To better understand this, we have calculated the variance of each set of MIoU data using the following formula where x is the mean average of the set and n is the sample size:

$$\frac{\sum (x-\bar{x})^2}{(n-1)}$$

The results of this calculation support the observations made on the graph. Each run had a respective variance of 0.0174 (5% data), 0.0122 (9% data), 0.0169(12% data) and 0.0114 (16% data). The model trained on 5% initial data has a smaller variety of classes and data available. It therefore might missselect the most useful data points in comparison to the 16% initial model which has more available information to base its selection upon.

5.2.4 Computation time

Another interesting metric we set out to investigate was how active learning and its initial dataset influenced the computing time of the whole training process. In earlier sections we mentioned that a benefit of active learning is that is has a more efficient computing time.

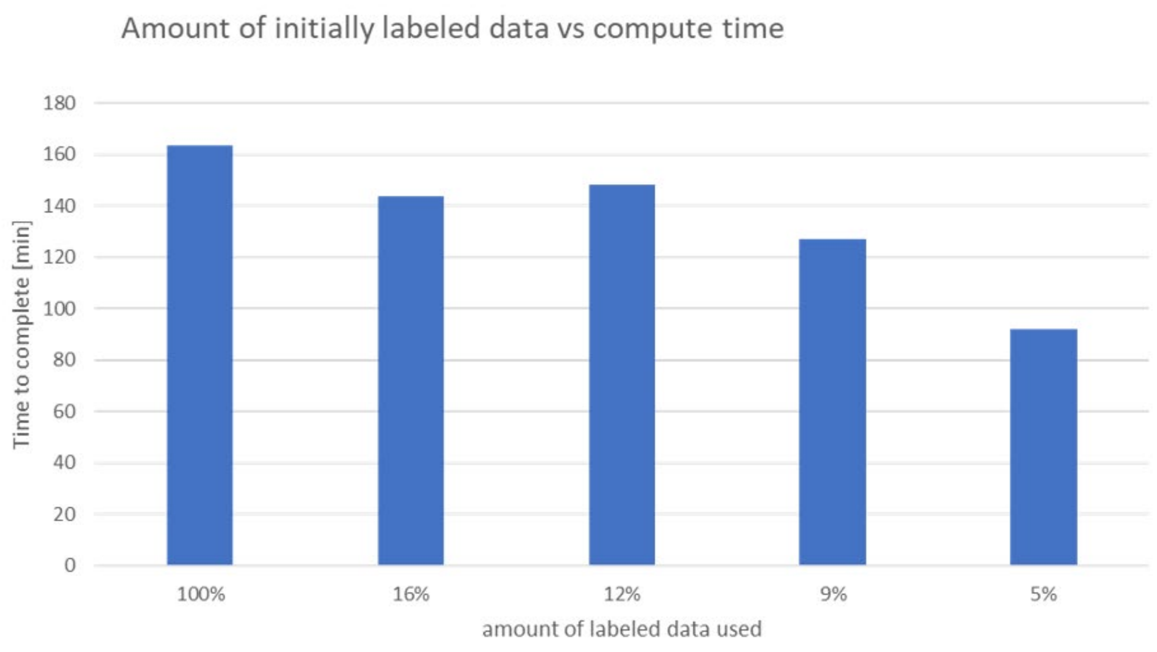


Figure 18 Computation time

Looking at the data in Figure 18, we can confirm that active learning does in fact improve upon computation time. These runs were all done on the same dataset and same hardware and software environment, ensuring comparability. The time save when using only 5% initial data in comparison with fully supervised learning is 43.7%.

5.2.5 Effect of color on active learning

As mentioned in section 4.2.2 the dataset provided by Strukton did not have any RGB values. This means in order to analyze the effect that color has on the precision of the active learning model, the open-source dataset S3DIS [19] needed to be used. To compare the effect of color on the precision, we ran one run with proper color values and one with uniform color values, effectively negating the influence of color.

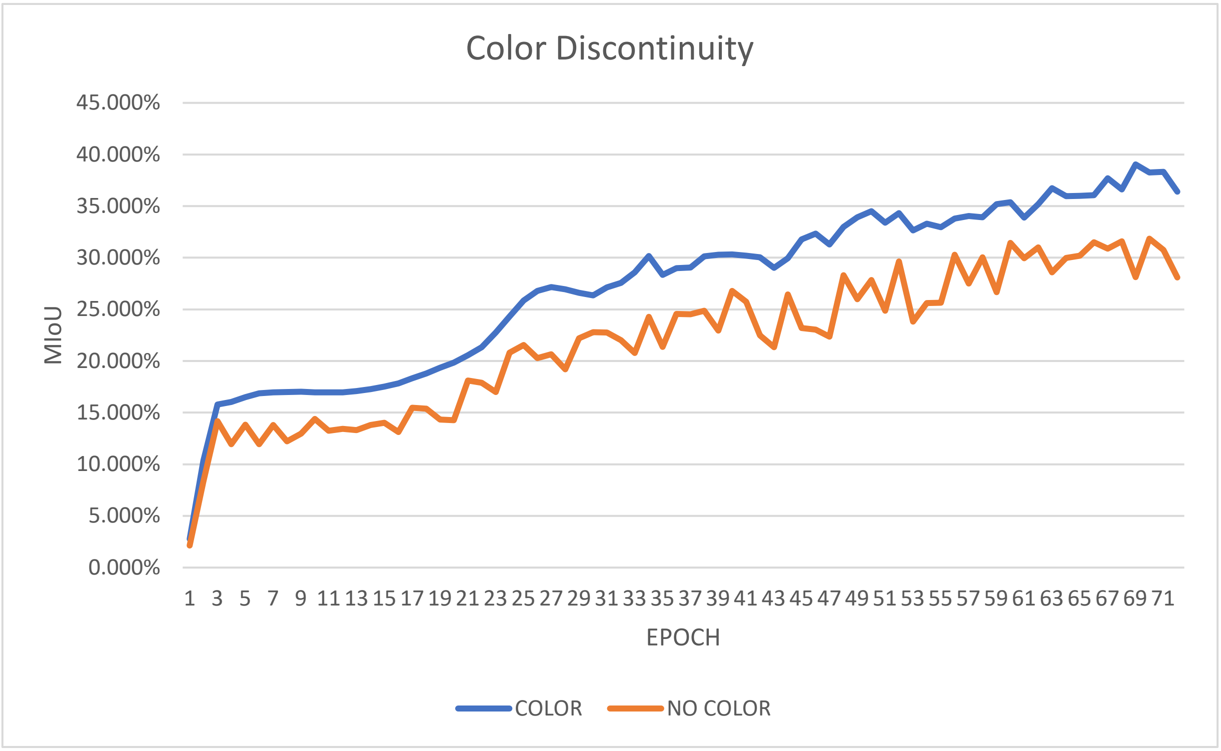


Figure 19 Effect of color on active learning

In Figure 19, the precision throughout the entire training process can be seen. We can observe that color values do increase the precision of the active learning model, as it has more information to take into account. Color is a great tool to tell objects of different classes apart. The highest precision achieved by the color run was 39.042% and the highest achieved by the no color run was 31.850%, which results in an improvement of 22.58% when using color for active learning. It is important to note that the S3DIS [19] is a vastly more complex dataset then the Strukton dataset, thus explaining the precision difference depicted in Figure 17 and Figure 19. It is important to note that color values drastically increase the physical storage space required for the dataset.

6 Conclusion and Future Work

In this section we will discuss the outcomes of the experiments conducted in relation to the research questions formulated earlier. In addition to this, we will discuss future research opportunities in the field of active learning in the point cloud environment and give suggestions to the AMI research group on how to best proceed with this research.

6.1 Conclusions

To answer the main research question, we can look at the pipeline described in section 3.3. We have created a pipeline that can take a dataset with any combination of classes and train a segmentation model on it. This process however does not come without hindrances. A big limitation of this method is still the extensive amount of environment setup required. The active learning framework [18] needs a very specific environment to run properly and can not be deployed on any system easily. Additionally, even though we was able to achieve good precisions with only 5% labeled data, the entire dataset still has to go through a large amount of preparation to be suitable for training.

Another goal of the main research question was to investigate the performance of the model compared to fully supervised training. The results presented in section 5.2.2 showed that fully supervised learning is the best way to train a segmentation model, when the main priority is precision. However, even with only 5% data we achieved great precision across most classes. This 95% decrease in labeled data needed is a huge time save for companies and researchers in this field.

We also set out to investigate how the initial dataset influenced the precision of the model. Our research shows that more data tends to lead to a higher overall precision and a more stable training process. In addition to this, it also influences computation time. Using less data results in a shorter computation time to train the entire model.

Lastly the effect of color on the active learning process was analyzed. Here my research shows that including color values in the point cloud data adds about 22% precision to the model. This is a significant increase in precision. However, including color values also increases the physical storage required.

6.2 Future work

To improve our understanding of active learning in the point cloud environment it is necessary to run more tests on different datasets. It is especially interesting to investigate the quality of the data that goes into the initial model. In addition to this, more tests need to be run to improve the reliability of my results. To properly assess if active learning is the best way to improve upon the need for less labeled data, other methods of semi-supervised and unsupervised learning need to be investigated and compared to active learning.

To properly adapt this in a commercial environment, the compatibility and adaptability of this framework needs to be improved upon as well. The main problems are the data preparation steps and the very specific structure the dataset needs to be in. To resolve this issue, the data readout process needs to be overhauled and investigated on how it can be more effective with more standard file types. Another interesting research opportunity would be the investigation on how RGB values can be substituted. Point clouds typically possess other values like intensity values that might also be of use when running active learning algorithms.

7 References

[1]S. Albawi and T. A. Mohammed, "Understanding of a Convolutional Neural Network," in *International Conference on Engineering and Technology (ICET)*, 2017.

[2]J. Brownlee, Deep Learning for Computer Vision, Machine Learning Mastery, 2019.

[3]J. Choi1, I. Elezi, H.-J. Lee, C. Farabet and J. M. Alvarez, "Active Learning for Deep Object Detection via Probabilistic Modeling," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 10264-10273*, 2021.

[4]S. Dridi, Unsupervised learning a systematic literature review*,* University at Buffalo, the State University of New York, 2021.

[5]T. Guo, J. Dong, H. Li and Y. Gao, "Simple Convolutional Neural Network on Image Classification," in IEEE 2nd International Conference on Big Data Analysis (ICBDA), 2017, pp. 721-724, doi:, 2017.

[6]E. Haussmann, M. Fenzi, K. Chitta, J. Ivanecky, H. Xu, D. Roy, A. Mittel, N. Koumchatzky, C. Farabet and J. M. Alvarez, Scalable Active Learning for Object Detection, IEEE Intelligent Vehicles Symposium (IV), 2020, pp. 1430-1435, 2020.

[7]M. Khanam and T. Mahboob, "A Survey on Unsupervised Machine Learning Algorithms for Automation, Classification and Maintenance," *International Journal of Computer Applications Volume ,* vol. 119, no. 13, p. 975 – 8887, 2015.

[8]T. Liu, S. Fang, Y. Zhao, P. Wang and J. Zhang, "Implementation of Training Convolutional Neural Networks," University of Chinese Academy of Science, 2015.

[9]K. O'Shea and R. Nash, *An Introduction to Convolutional Neural Networks,* ArXiv abs/1511.08458: n. pag., 2015.

[10] J. Redmon, S. Divvala, R. Girshick and A. Fahradi, "You Only Look Once: Unified, Real-Time Object Detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779-788*, 2016.

[11] S. Roy, A. Unmesh and V. P. Namboodiri, *Deep active learning for object detection,* Department of Computer Science and Engineering, Indian Institute of Technology, 2018.

[12] R. Yamashita, M. Nishio, R. K. G. Do and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights Imaging ,* vol. 9, p. 611–629, 2018.

[13] C.-W. Yu, Y.-L. Chen, K.-F. Lee, C.-H. Chen and C.-Y. Hsiao, "Efficient Intelligent Automatic Image Annotation Method based on Machine Learning Techniques," in *IEEE International Conference on Consumer Electronics*, 2019.

[14] W. Yu, S. Zhu, T. Yang and C. Chen, "Consistency-based Active Learning for Object Detection," in *EEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, pp. 39513960*, 2022.

[15] A. L, Z. J and H. A.J, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *J Big Data,* vol. 8, no. 53, 2021.

[16] N. C. G, A. Anish and M. Jonas, Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks, arXiv, 2021 .

[17] Y. Yang and A. Whinston, "Identifying mislabeled images in supervised," in Lecture Notes in Networks and Systems, Austin, Springer International Publishing, 2021, pp. 266--282.

[18] Tsung-Han Wu, Yueh-Cheng Liu, Yu-Kai Huang, Hsin-Ying Lee, Hung-Ting Su, Ping-Chia Huang, Winston H. Hsu, ReDAL: Region-based and Diversity-aware Active Learning for Point Cloud Semantic Segmentation, Cornell Univsetiy, 2021

[19] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, Silvio Savarese, 3D Semantic Parsing of Large-Scale Indoor Spaces, in CVRP, 2016

[20] Francis Engelmann, Theodora Kontogianni, Jonas Schult, and Bastian Leibe, Know What Your Neighbors Do: 3D Semantic Segmentation of Point Clouds, RWTH Aachen University, 2017

[21] Jeremie Papon, Alexey Abramov, Markus Schoeler, Florentin Worgotter, Voxel Cloud Connectivity Segmentation - Supervoxels for Point Clouds, Bernstein Center for Computational Neuroscience (BCCN), 2013

[22] Dena Bazazian, Josep R Casas, and Javier Ruiz-Hidalgo. Fast and robust edge extraction in unorganized point clouds. In 2015 international conference on digital image computing: techniques and applications (DICTA), pages 1–8. IEEE, 2015.

[23] Mark Pauly, Richard Keiser, and Markus Gross. Multi-scale feature extraction on point-sampled surfaces. In Computer graphics forum, volume 22, pages 281–289. Wiley Online Library, 2003.

8 Appendicies

DATA

RUN 1 (16% DATA)

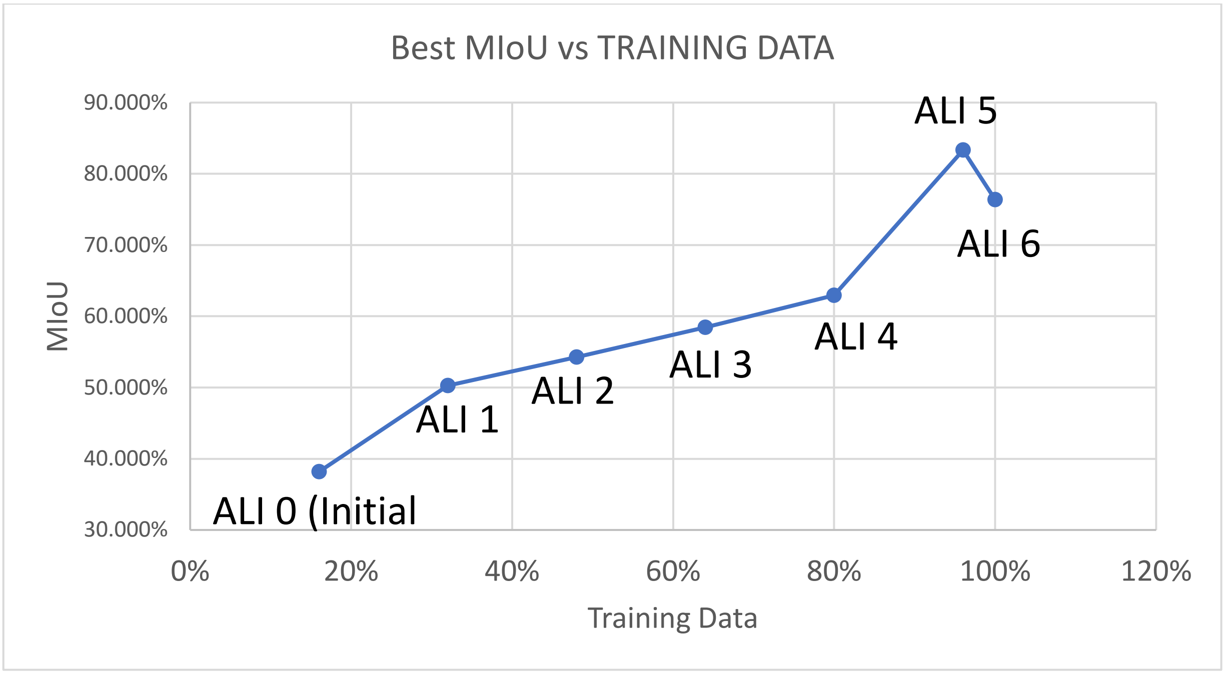


Figure 20 Active learning iterations 16% Data

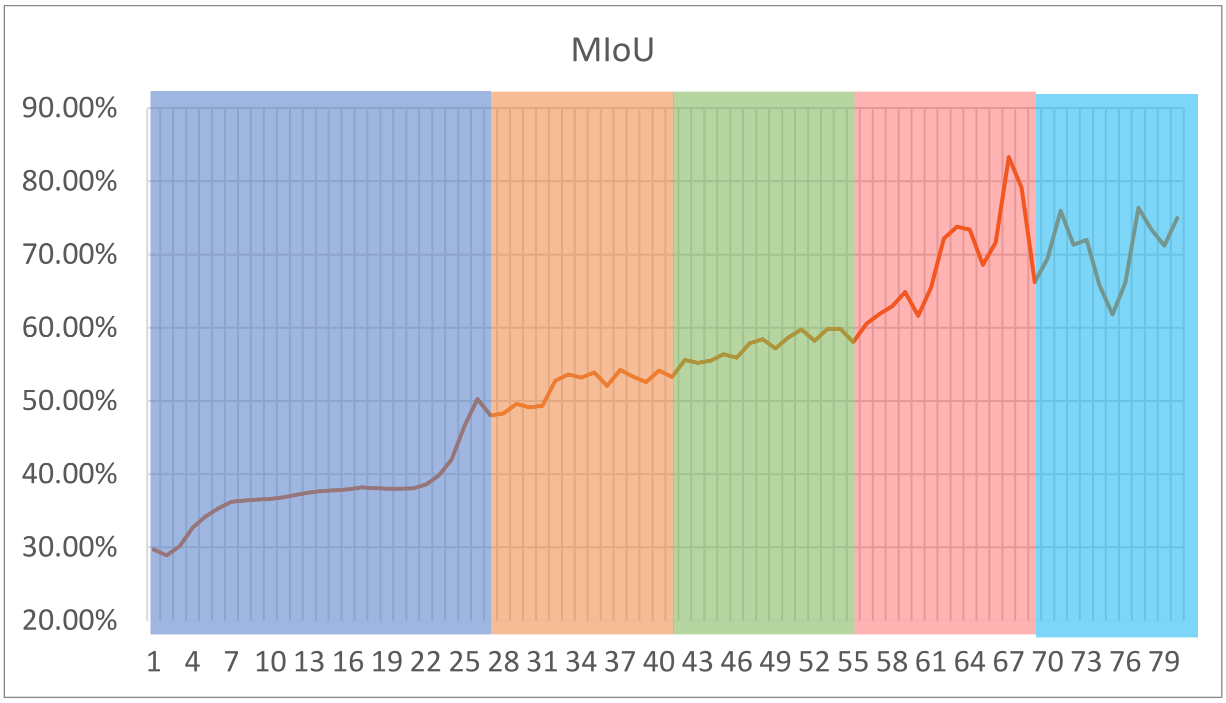


Figure 21 Active learning 16% Data

RUN 2 (12% DATA)

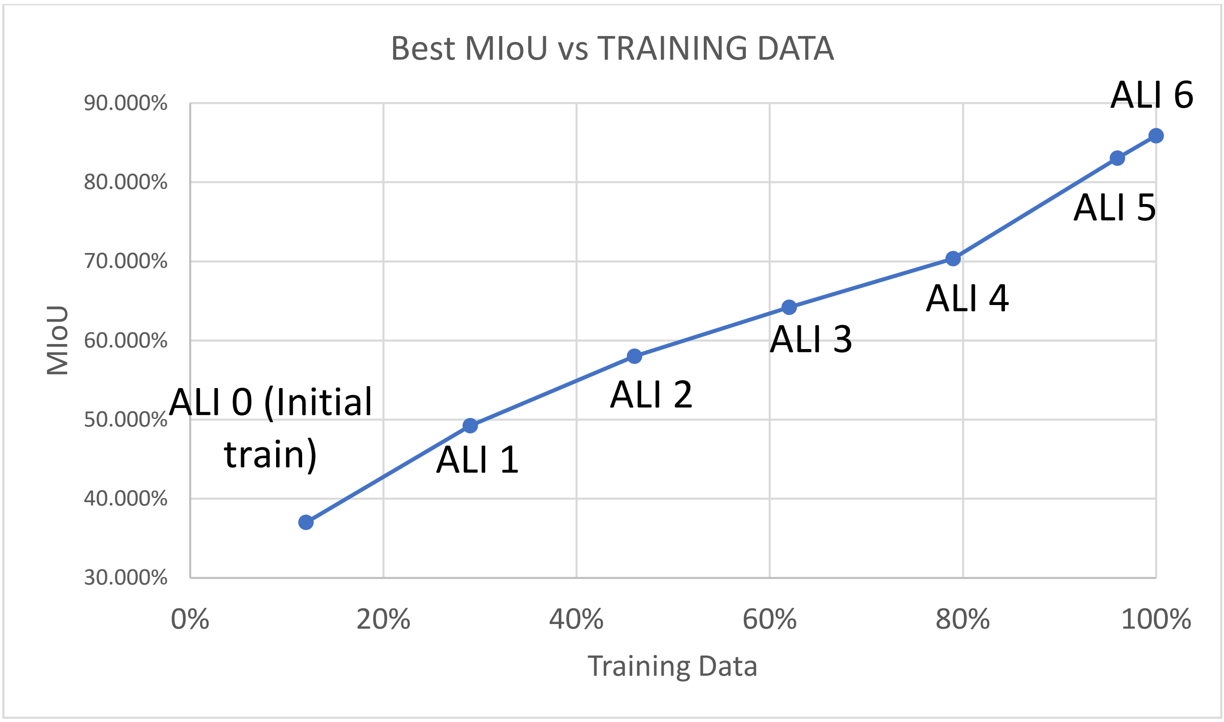


Figure 22 Active learning iterations 12% Data

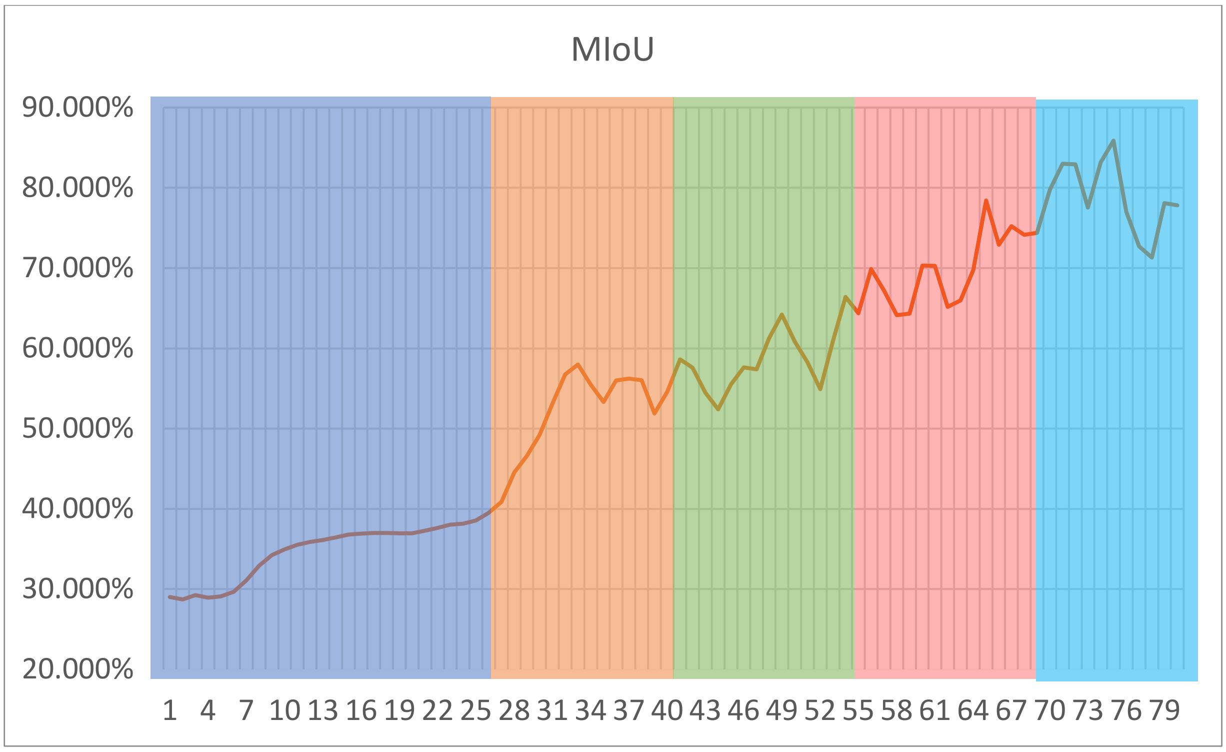


Figure 23 Active learning 12% Data

RUN 3 (9% DATA)

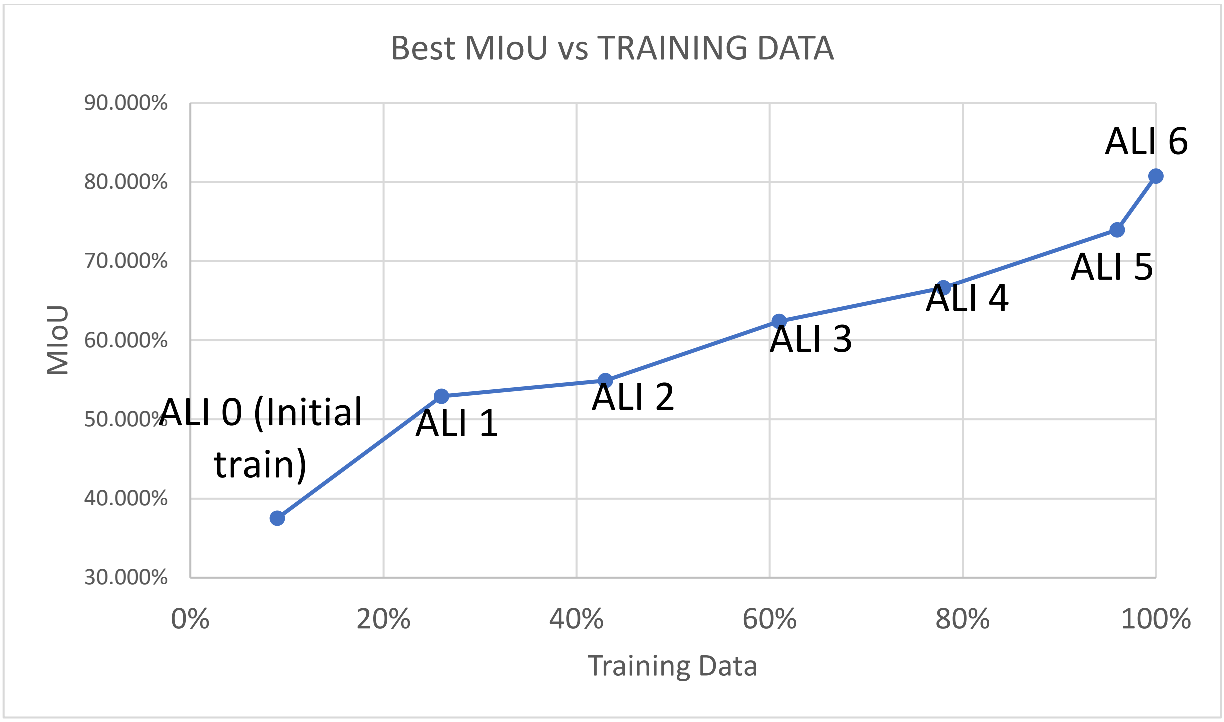


Figure 24 Active learning iterations 9% Data

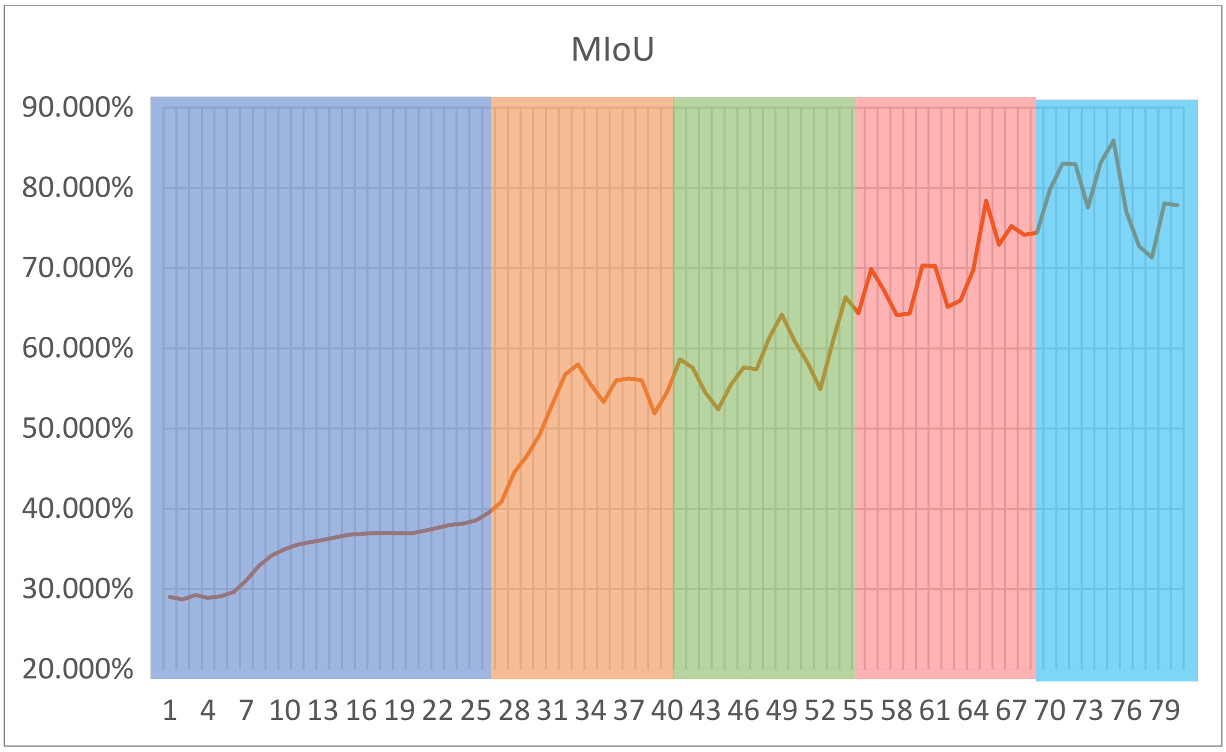


Figure 25 Active learning 9% Data

RUN 4 (5% DATA)

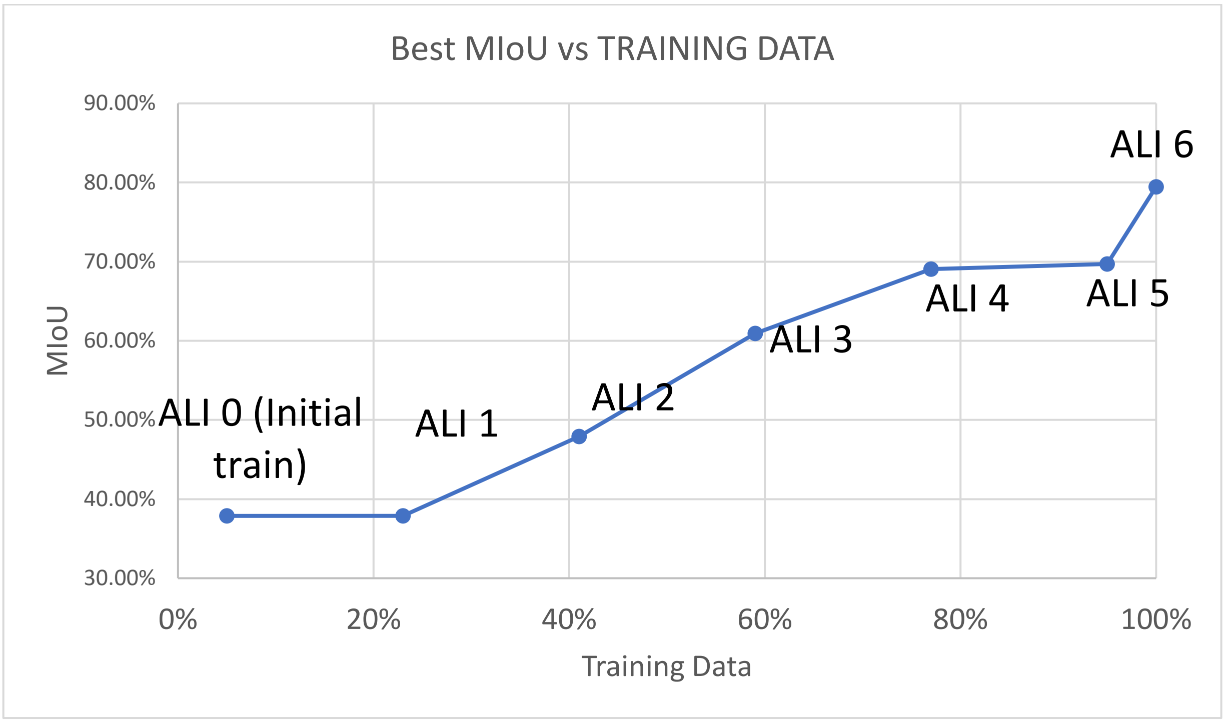


Figure 26 Active learning iterations 5% Data

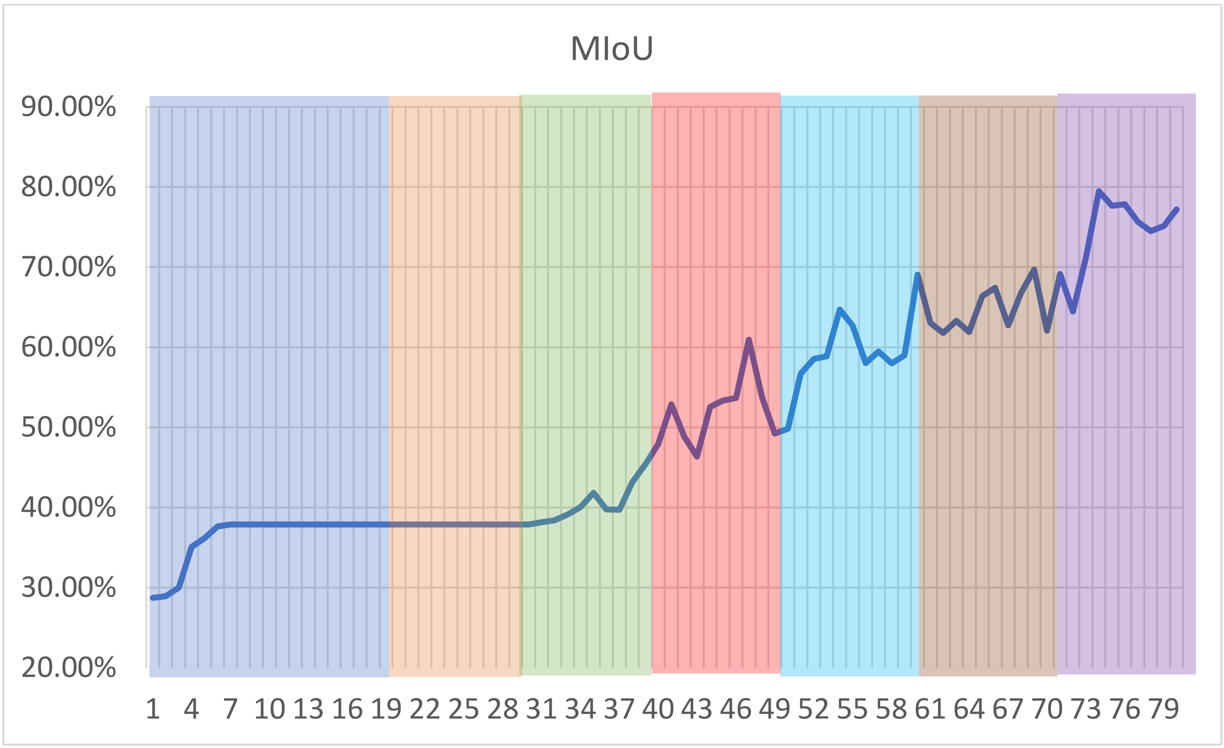


Figure 27 Active learning 5% Data

Code Snippets

Conversion Script:

```
# Import the necessary modules
import numpy as np
import laspy
import os

folder_path = "data"


for file_name in os.listdir(folder_path):
    if file_name.endswith(".las"):
        # Open the .las file and read its contents
        las_file = laspy.read(os.path.join(folder_path, file_name))
        point_cloud = las_file.points

        x = point_cloud.x
        y = point_cloud.y
        z = point_cloud.z

        all_coords = np.column_stack([x, y, z]).astype(float)

        rgb_all = np.ones([len(x), 3], dtype=float)

        all_coords = np.concatenate([all_coords, rgb_all], axis=1)

        root, ext = os.path.splitext(file_name)

        os.mkdir(root)

        np.savetxt(os.path.join(root, "{}.txt".format(root)), all_coords)

        # Extract the labeled coordinates from the point cloud data
        labeled_coordinates = point_cloud[point_cloud["label"] > 0]

        # Extract the unique labels from the labeled coordinates
        labels = np.unique(labeled_coordinates["label"])
```