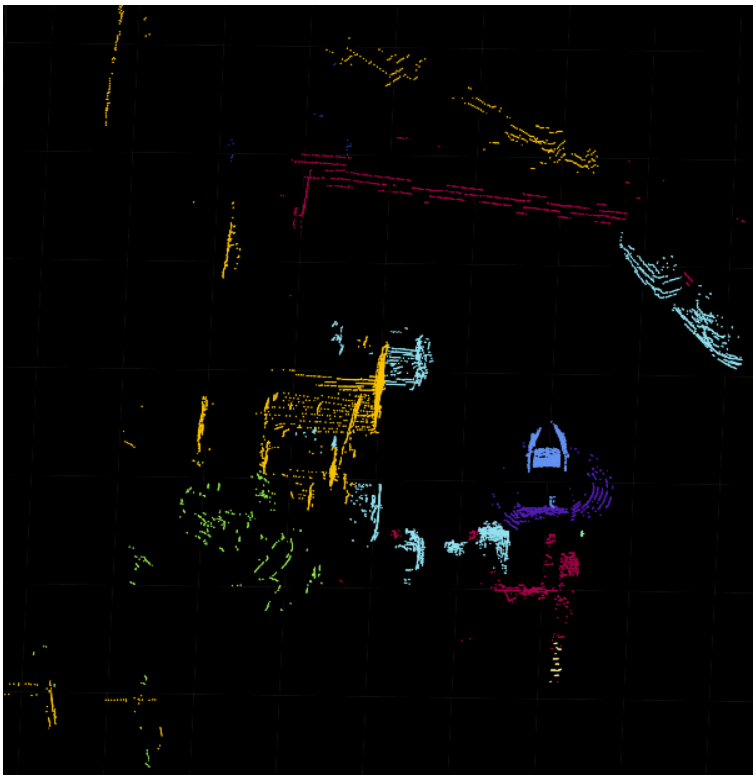
CHALMERS
UNIVERSITY OF TECHNOLOGY



Semantic Segmentation in Marine Environment

Using 2D spherical projection and convolutional neural networks

Master's thesis in
Communication Engineering & Systems, Control and Mechatronics

Emma Dahlin
Hanna Jonsson

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022
www.chalmers.se

Master's thesis 2022:19

Semantic Segmentation in Marine Environment

Using 2D spherical projection and convolutional neural networks

Emma Dahlin
Hanna Jonsson

CHALMERS

Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
Gothenburg, Sweden 2022

Semantic Segmentation in Marine Environments
Using 2D spherical projection and convolutional neural networks

Emma Dahlin
Hanna Jonsson

Supervisor: Jonatan Nordh, CPAC Systems AB
Examiner: Peter Forsberg, Department of Mechanics and Maritime Sciences

Master's Thesis 2022:19
Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Ground truth of a LiDAR point cloud in a harbour

Typeset in LaTeX
Gothenburg, Sweden 2022

Semantic Segmentation in Marine Environment
Using 2D spherical projection and convolutional neural networks
Emma Dahlin
Hanna Jonsson
Department of Mechanics and Maritime Sciences
Chalmers University of Technology

Abstract

The marine environment is constantly changing and therefore it can be difficult to control a vessel under these conditions. In this thesis, a method is proposed to interpret the surroundings to aid easy and safe travels on water. This is done through semantic segmentation by transforming 3D point clouds to 2D images, using a projection-based method. The transformation enables training with convolutional neural networks to achieve a fast and high performance network.

The above method is successfully implemented in the marine environment and the results show that fewer classes are preferable to reach a high accuracy of the model. The features from the environment was unbalanced, which was compensated for by implementing a loss function that weighted the underrepresented classes higher. The model increased in performance for the minority classes. Furthermore, the real-time semantic segmentation was slower compared to the sensors update-time but there are possibilities to reduce the prediction time in future work. Precipitation was hard to detect due to low amounts of annotated data but the other surroundings could be detected in harsh weather conditions either way. The results show promising outcome for future implementation.

Keywords: Machine learning, Semantic segmentation, CNN, Marine environment, LiDAR, U-Net

Acknowledgements

We would like to take the opportunity to thank CPAC Systems AB for making this thesis possible by providing us with resources and ideas for the project. It would not have been possible without all the support we have gotten from our supervisor, Jonatan Nordh, who we want to give many thanks to. Furthermore, we would like to express our gratitude to our examiner, Peter Forsberg, for all the guidance and advice during the spring. Additional thanks to the warm welcome and inclusive environment at CPAC Systems AB and all the people there.

Emma Dahlin & Hanna Jonsson, Gothenburg, June 2022

Thesis supervisor: Jonatan Nordh, CPAC Systems AB
Thesis examiner: Peter Forsberg, Department of Mechanics and Maritime Sciences

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AD	Autonomous Driving
ADAS	Advanced Driver Assistance Systems
BEV	Bird's Eye View
CAD	Computer-aided design
CE	Cross Entropy
CNN	Convolutional Neural Networks
FL	Focal Loss
FoV	Field of View
GNSS	Global Navigation Satellite Systems
GPU	Graphics Processing Unit
LiDAR	Light Detection And Ranging
RAM	Random Access Memory
ReLU	Rectified Linear Unit

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

Indices

i	Indices for class in loss function

Parameters

C	Number of classes
γ	Hyperparameter for Focal loss
T_{L2}^{L1}	Transformation matrix between LiDAR 2 & 1
T_{L3}^{L1}	Transformation matrix between LiDAR 3 & 1
T_{L4}^{L1}	Transformation matrix between LiDAR 4 & 1
T_{L5}^{L1}	Transformation matrix between LiDAR 5 & 1
T_{L1}^{Base}	Transformation matrix between LiDAR 1 & boat base
FoV_{up}	Field of View up
FoV_{down}	Field of View down
FoV	Entire Field of View
H	Height of image
W	Width of image

Variables

TP	True positive in confusion matrix
TN	True negative in confusion matrix
FP	False positive in confusion matrix

FN	False negative in confusion matrix
y_i	Ground truth in loss function
p_i	Prediction in loss function
(x, y, z)	Cartesian coordinates in point cloud
(R, ϕ, θ)	Spherical coordinates in point cloud
X_{proj}	Projected x coordinate
Y_{proj}	Projected y coordinate

Contents

List of Figures

List of Tables

1
Introduction

In this section the background and purpose of the master thesis work is described. The background presents the current problem and the importance of finding a solution to it. The purpose mentions why this master thesis is done and which outcome the work strives for.

1.1 Background

During year 2020 there were 311 incidents and accidents reported in professional shipping on Swedish waters. In addition to this, 29 persons passed away in private boating [1]. This is only statistics from Swedish waters and the number of accidents that has been reported. These numbers should be as close to zero as possible and a step to reduce this is to implement an accurate surrounding view system. For a human it can be hard to detect and interpret the surroundings of a vessel, depending on the size of the vessel, which is an issue that can be avoided with such a system. Furthermore, it can be a step in developing autonomous help systems. When an autonomous system is implemented, it needs to be capable of making accurate and precise decisions as well as interpreting its surroundings correctly. Today semantic segmentation is applied in vehicles on land to identify the surroundings and classify obstacles, like pedestrians, other cars etc. This allows development of applications such as parking assistance and warning systems where identification of the surrounding is a crucial asset.

Systems, like the ones mentioned above, can be implemented to be able to more safely navigate the vessel, especially in confined spaces. Furthermore, the weather can complicate navigation with rain, snow, wind, darkness etc. Therefore, it is of great importance to know the vessels surroundings to avoid collisions and other dangerous situations. Similar to ground vehicles, other types of vessels can now be equipped with a multitude of sensors, such as LiDARs and cameras. However, the data from the sensors still needs to be interpreted to locate obstacles or to implement autonomous applications that can assist the captain. For ground vehicles, where the use of sensors is more widespread and the technique more mature, semantic segmentation is widely used to classify the surroundings. One drawback with this technique is the heavy computational demand required. A master thesis work was carried out to investigate the use of lightweight CNNs (Convolutional neural networks) for fast and accurate segmentation [2]. However, the circumstances in marine environments differ from those on ground in substantial ways [3]. Today

there are limited amount of data sets from marine environments that are labeled and ready to train on a CNN. The complex environments, lack of features and the limited amount of data are something that needs to be handled to achieve better systems for identifying the surroundings on sea.

1.1.1 Complex Marine Environment

The previously mentioned complex marine environment makes it difficult to perform semantic segmentation due to several factors. Harsh weather conditions often create low-quality data to use for training, which is a problem since learning models need a lot of high-quality data to perform well. Weather is an issue in non-marine environments as well, however there has not been as much data collected in marine environments compared to traffic environments. This causes the existing data sets to often have data or label starvation. Furthermore, water spray from waves and other movements in the water constitutes a class that has a lot of different shapes, making it harder to use for training. In comparison to semantic segmentation on land the marine environment generally provides fewer features. Roads and traffic most often have a lot of other vehicles, road signs, buildings and vegetation close by. However, when driving a boat there is usually only water in proximity, most often it is only in harbors or other confined spaces that there are noticeable features.

1.2 Purpose

The purpose of this master thesis is to implement and adapt fast semantic segmentation on solely sparse LiDAR data in marine environments. To enable a larger area of use the thesis will focus on using data with a lower resolution while still achieving a fast and sufficient precision in real-time scenarios. The sensors used to collect data are five LiDARs placed on a boat.

1.2.1 Objectives

The master thesis aims to answer the following questions.

- How could limited access to good training data and harsh weather conditions in marine environments be handled and compensated for?
- Are data from LiDAR sensors with low resolution enough to provide high performance for segmentation in marine environments?
- How can the training and the network be designed to reach an accurate and precise segmentation with low resolution data in marine environment?
- How can the training and the network be designed to achieve a fast segmentation that can be used in real-time with low resolution data in marine environment?

1.3 Limitations

The scope of this project is to investigate low resolution LiDAR data for semantic segmentation, hence high-resolution data will not be used. The data will be preprocessed from a 3D point cloud to a 2D spherical view, like a panorama image and therefore projections in birds eye view will not be used. Since there is real data available no simulated data will be gathered and used. This is due to lack of findings of simulation tools for marine environment. Moreover, according to a previous master thesis work [2] a network trained on only real world data performs better than one trained on simulated data. The thesis evaluated a network that had been pretrained on artificial data and tested with real-world data which showed that simulated data can not replace training with real-world data. Furthermore, this project will be using only LiDAR sensors since one of the objectives is to investigate if only LiDAR data is enough to do semantic segmentation on. This type of sensor can also provide quality point clouds even in challenging weather compared to cameras.

2
Theory

The techniques used in the thesis work are presented in this chapter, such as LiDAR, semantic segmentation, preprocessing and training of a network.

2.1 LiDAR

Light Detection and Ranging (LiDAR) is a core technique when designing Advanced Driver Assistance Systems (ADAS) and developing Autonomous Driving (AD) vehicles. The principle of a LiDAR is built on light being reflected on different objects. The signals sent are several infrared beams or laser light that the sensor is emitting, which enables detection of the surroundings. Here, a method called Time-of-Flight (TOF) is used to calculate the distance to a detected object by using the time it takes for a light particle to travel back and forth [4]. TOF is a proven and reliable technique used for fast detection of objects [5]. As an example, a high-resolution LiDAR can generate 1 300 000 points per second [6]. All the points recorded by the sensor can be illustrated as a point cloud in 3D vision which contains the parameters of position (x, y, z) and the intensity of the light returned.

Today, different LiDARs can cover a horizontal field of view (FoV) of up to 360° and a vertical FoV of up to 90° [7]. An illustration of the FoV can be seen in Figure 2.1.

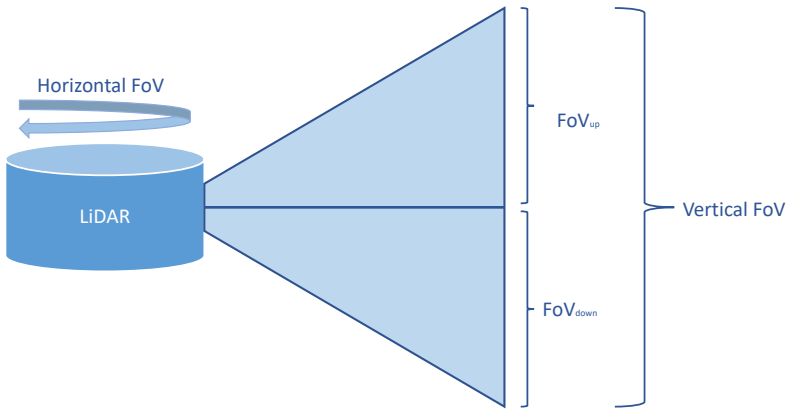


Figure 2.1: A LiDAR sensor where its Field of View (FoV) is showed.

There are mainly two types of LiDARs. One kind has internal rotating parts and gives a horizontal perception of 360°, here called mechanical LiDAR. The other one is a static (solid-state) type that does not have any rotating parts and hence can normally only deliver a perception of the horizontal plane of a maximum of 120°. Mechanical ones are more popular when used in designs of AD vehicles since they can scan the whole horizontal area of the vehicle's surroundings. A downside with a LiDAR with moving parts is an increased possibility of mechanical faults, which could make it less reliable and not as robust as the static one. Mechanical sensors are also more expensive and have an inferior resolution [6].

Camera sensors could be a complement for LiDARs since they provide pictures of colour and textures which can be used to perceive the surroundings, for example traffic signs and lights in ADAS. However, it is more difficult to measure the distance to objects in images compared to point clouds since there is no depth information [4]. Furthermore, it is difficult to gather high-resolution images that are possible to be used for training in the marine environment due to the complexity of the surroundings [3]. LiDARs are more suitable under hard conditions such as darkness, fog, rain and splashes from the surrounding water. Even if the beams from a LiDAR are affected by rain and water drops the sensor still manages to positively detect the other surroundings.

2.2 Semantic Segmentation

In this work, semantic segmentation is used to classify each pixel in an image. In this process, each pixel gets assigned to a class, which means that the pixel gets a label that is connected to a class. The classes are often visualised in different colours. If there are several instances of the same class semantic segmentation does not differentiate between the instances. For example, if there are several boats to classify in an image all the pixels of the boats will get the same label [8].

For LiDAR data each 3D point in a point cloud will be classified. There are mainly two different methods for this, the projection-based method and the point-based method [9], [10] that are explained more in Sections 2.3.1 and 2.3.2.

2.3 Preprocessing of data

In machine learning, the input data is of big importance to get a good performance of a model. There are several different ways to preprocess the data depending on the data format and what kind of input is needed for the network. The data is often divided into three different sets; training, validation and test. They all need to contain a mixture of data and the training data need to be shuffled on its own to decrease the risk of overfitting. Overfitting is when the network learns specific patterns for some data but performs poorly on new data [11]. If the training data is not shuffled the network will learn on frames that are almost identical to each other at the beginning of training. Therefore, as the training progresses and new data

are introduced the network will not generalise well. To avoid this phenomenon a validation loss can be examined, when its value deviates from the training loss and increases rapidly in value the model has been overfitted [11].

As previously mentioned, segmentation of 3D clouds can be done in mainly two different ways. These are the projection-based method and the point-based method. The projection-based method requires a preprocessing of the point cloud while the point-based method directly uses the 3D point cloud.

2.3.1 Projection-Based

The Projection-based method for a 3D point cloud involves a transformation of the 3D points into a 2D projection. The purpose of the method is that it enables the usage of the widely tried techniques of CNNs, which require images as input [10]. The most common branches of the method are either projection by a spherical view or a bird's eye view (BEV).

The spherical view projection technique uses the knowledge of the depth to project the point cloud to a panorama 2D image. The projection is done by working with the azimuth and zenith angles. With this technique, semantic segmentation can be performed with a low computational cost [10]. Depending on how the point cloud looks the 2D projection images can have a strange spatial distribution compared to normal images. The data can be very sparse in some locations and extremely concentrated in others [12]. Furthermore, objects that are hidden behind other objects from the point cloud centre will not be seen as distinctly on the projections.

BEV projects the 3D data to a 2D grid from above to the horizontal plane. For this it is a necessity that all the objects that are meant to be detected are located on the same plane [13]. Hence, this technique does not use depth information since it is not needed for the provided purpose of this method [10].

2.3.2 Point-Based

The point-based method uses point clouds as data input to the network, which means that no preprocessing of the data is performed [10]. Therefore, the data contains all the information from the beginning. The method is good for smaller point clouds since they have less data to handle. However, for larger clouds it will bring a lot of computations and will result in longer processing time, hence higher computational cost. The point-based method is suitable for indoor scenes since the 3D point clouds are usually in much smaller sizes and ranges in these areas [12].

2.3.3 Supervised learning

The training in machine learning can either be supervised or unsupervised. With supervised learning, the input data has labels that are also known as the ground truth. During training, the output prediction from the network is compared to the

ground truth. The comparison is done through a loss function and the weights in the network are updated to minimise it. Supervised learning is often used for segmentation and to be able to perform it the data need to have ground truth values. If it does not exist it needs to be created during preprocessing.

2.4 Training

In this section the network structure of a U-Net, multi-class accuracy and two different loss functions are explained. The loss functions are Cross-Entropy (CE) loss and Focal loss.

2.4.1 U-Net

For classification tasks, CNNs are often used to get one label per image [14]. However, in semantic segmentation each pixel or point need to have a label which makes for a slightly different network structure than for normal image classification.

The U-Net is a CNN that is used for semantic segmentation on images and was first developed for biomedical image segmentation [14]. The network consists of convolution layers, max pooling layers and ReLu activation layers. An overview of U-Net can be seen in Figure 2.2.

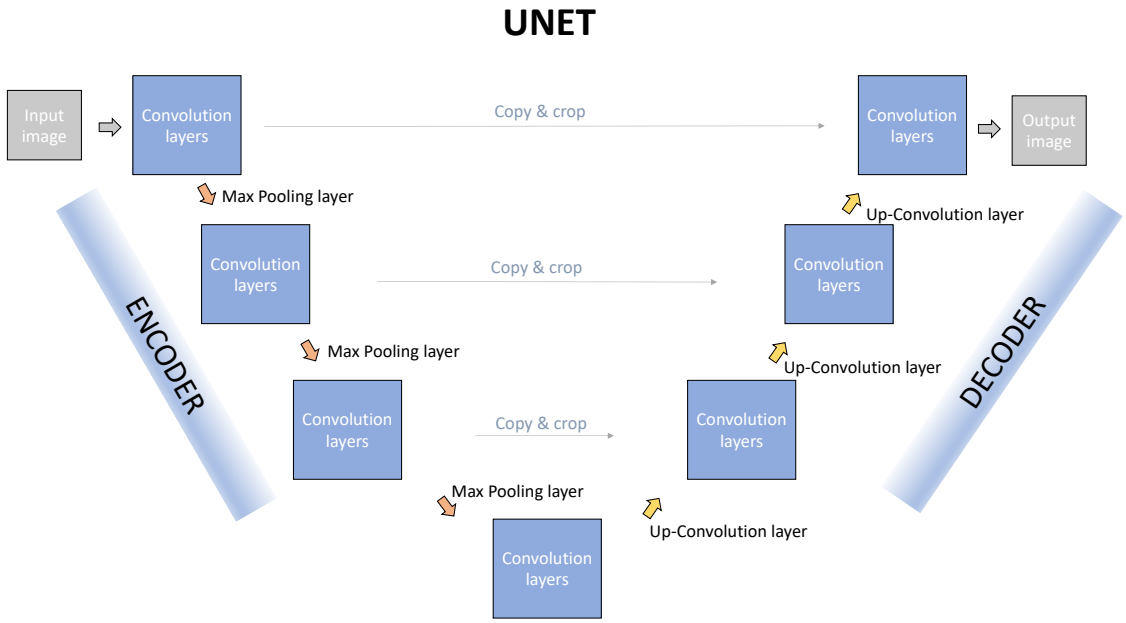


Figure 2.2: Overview of U-Net with blocks of CNNs and the steps in between on the encoder and decoder side.

The name comes from its U-shape where an encoder forms the left side of the U, and a decoder forms the right side. The encoder has several blocks that each consist of convolutions with Rectified Linear Unit (ReLU) activation layers, followed by

a max pooling layer to downsample. The decoder upsamples with similar blocks but instead of max pooling layers there are convolution layers that upsamples the image, called up-convolution. To not lose information in the encoder the feature map from every block is copied and cropped with the corresponding feature map in the decoder [14].

2.4.2 Accuracy

For a machine learning problem accuracy is a common validation method to evaluate the model performance. Accuracy is a measurement of how good the model is at making the correct predictions. It is defined as,

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ amount\ of\ predicted\ samples} \tag{2.1}$$

For binary classification problems, the accuracy can be written as,

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.2}$$

where TP - true positive, TN - true negative, FP - false positive and FN - false negative. In Figure 2.3 an illustration can be seen of the binary classification problem. The matrix is called a confusion matrix, which is widely used for calculation of accuracy [15].

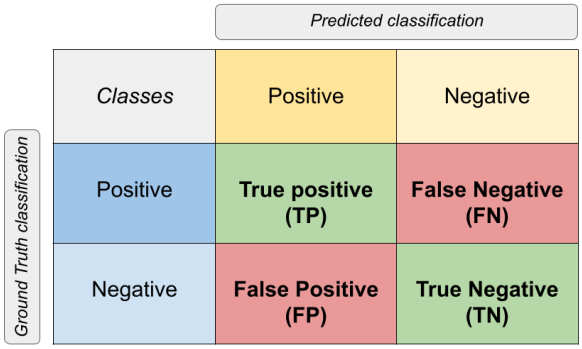


Figure 2.3: Confusion Matrix for binary classification problem.

When described as a binary problem where there are positives and negatives the TP are the ones that have been labelled as positive and have a ground truth value that is positive as well. TN are the ones that are labelled as negative and have negative ground truth. FP and FN have the same principle, but the label and the ground truth does not match [15].

The Confusion Matrix can easily be expanded and used for multi-class problems, as shown in Figure 2.4. Instead of using a positive and a negative case, as in the binary matrix, each row and column represents a class. Therefore, the TP, TN, FP and FN labels cannot be applied in the same way. Instead, the diagonal represents all the correct classified labels, and the rest are incorrect classifications. To calculate the

accuracy of a multi-class problem the diagonal is summarised and divided by the sum of the matrix. In the example from figure 2.4 with four classes the accuracy would be calculated as: $acc = \frac{4+8+12+9}{55}$.

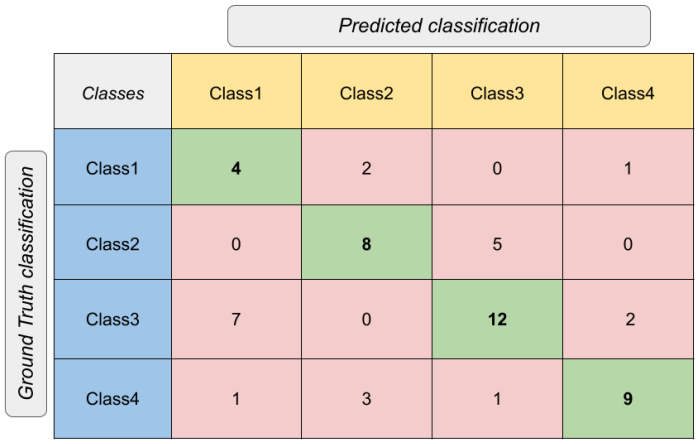


Figure 2.4: Example over Confusion Matrix for multiple classes.

2.4.3 Cross entropy loss for multi-class classification

The Cross Entropy loss function (CE) is often used in multi-class classification and the aim is to minimise the loss. The loss is calculated using Equation (2.3) on each pixel where the sum is adding the loss of the different classes to each other [16].

$$CE = -\sum_{i}^{C} y_i log(p_i) \tag{2.3}$$

y_i is the ground truth and has one probability matrix for each class. In each one of these matrices every pixel has either a value of 1, if it is the true class, or 0. p_i is the prediction that will be compared to the ground truth. It has the same size as the ground truth but instead the probabilities depend on how certain the prediction is on what class it is. In other words, each pixel has values on every matrix that ranges from 0 to 1.

2.4.4 Focal loss

Focal loss is based on the Cross Entropy loss but is adapted to manage class imbalance specifically [17]. A modulating factor is added to the CE-loss to weigh down the classes that are easier to predict and make the classes that are harder to predict more important. The Focal loss term added is $(1 - p_i)^\gamma$ and the function with the modulating factor can be seen in Equation (2.4). $\gamma \geq 0$ is a parameter that can be changed in value depending on how important the features that are less represented should get. Research has been made and a study has shown that a value of $\gamma = 2$ gives the most promising results with an imbalanced data set [17].

$$FL = -\sum_{i}^{C} (1 - p_i)^\gamma y_i log(p_i) \tag{2.4}$$

3
Method

In this chapter the method carried out is presented. This includes different data processing, collection of additional data, building network structures, network training and evaluation. An overview of the working process and design of the method can be seen in Figure 3.1, where a block schedule is presented with the different steps.

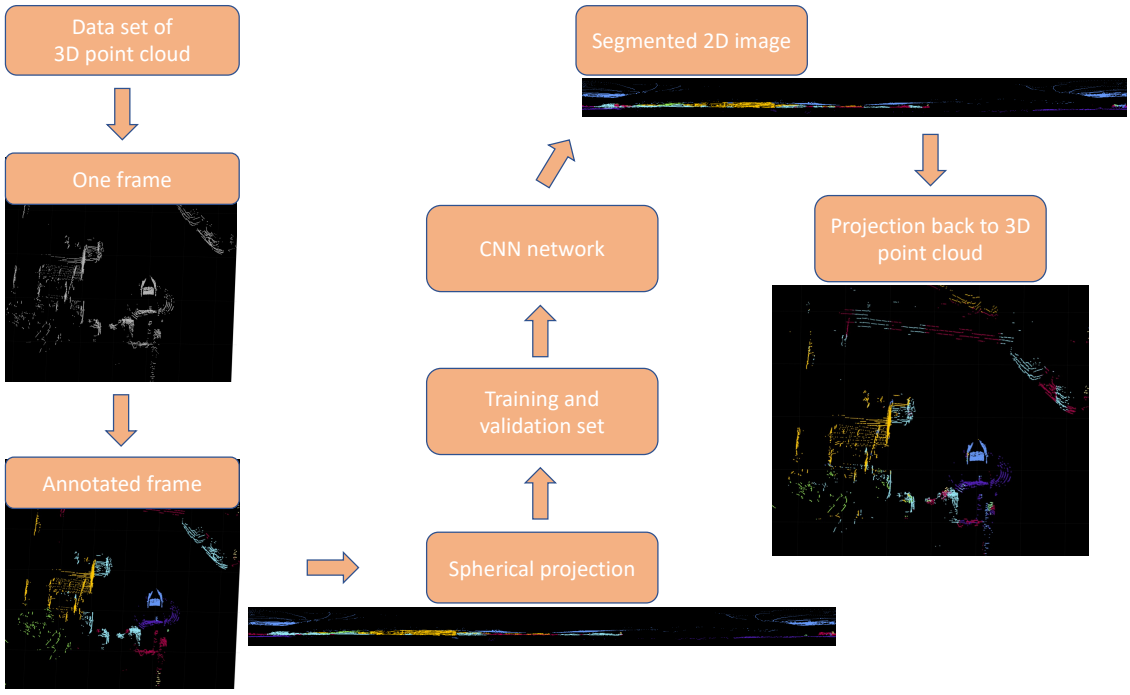


Figure 3.1: Blockschedule representing an overview of the executed method. The data set consists of frames which are annotated in the next step. The 3D clouds are then projected into 2D images. The 2D images compose the training, validation and test set which are used in the CNN network. From the trained network a segmented 2D image is produced that is projected back to a 3D cloud.

3.1 Data preprocessing

Previous and newly acquired data were used. The collected data were from a boat equipped with five LiDAR sensors and gave a 360° view of the boat's surroundings. The placement of the LiDARs and the GNSS sensor can be seen in Figure 3.2.

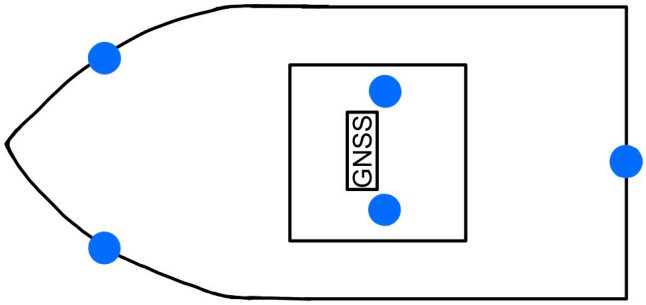


Figure 3.2: The blue dots represent the five LiDARs placements on the boat. The GNSS receiver is also marked.

To view the data from the same coordinate system, transformations were needed between the LiDARs. The data were collected as time series similar to videos, which creates several frames for each recording. The update rates of the LiDARs were set to 10 Hz, meaning 10 frames per second. Recordings were cut to a maximum of 400 frames to easier handle the data and make sure that longer recordings would not take over the training. Each frame was annotated and then projected onto a 2D image before the data set could be sent to the training algorithm.

3.1.1 Transformations between coordinate systems

The data were collected with five different LiDARs, referred to as L1, L2, L3, L4 and L5. Hence, it consisted of five different point clouds, one for each sensor. For further processing of the data, the point clouds needed to be transformed to the same coordinate system since the recordings were made for each LiDAR's reference point. To transform the point clouds the transformation matrices between the LiDARs needed to be produced. These were computed with quaternions and translation vectors. Prior to this work the quaternions and vectors were calculated through calibration. A scan from each LiDAR was needed at the same time in an environment with a lot of features. These clouds needed to overlap over mutual features in order for a program to approximate the sensors' positions and directions. When this was made the point clouds were placed in relation to a CAD model of the own boat which then could be calibrated properly. The calibration is done through an algorithm, Iterative closest point, which fine-tunes the point clouds until they match each other and the clouds from L1-L5 have the same measurement values [18]. When the fine-tuning was done the translation vectors and quaternions for each sensor

could be used to produce the transformation matrices between each LiDAR. The transformation matrix can be expressed as

$$T_m = \begin{bmatrix} R_1^0 & O_1^0 \\ 0_3^T & 1 \end{bmatrix}$$

where R is a 3x3 rotation matrix and O the translation vector [19]. The transformations are visualised in Figure 3.3 as yellow arrows while the computed matrices can be found in Appendix A.

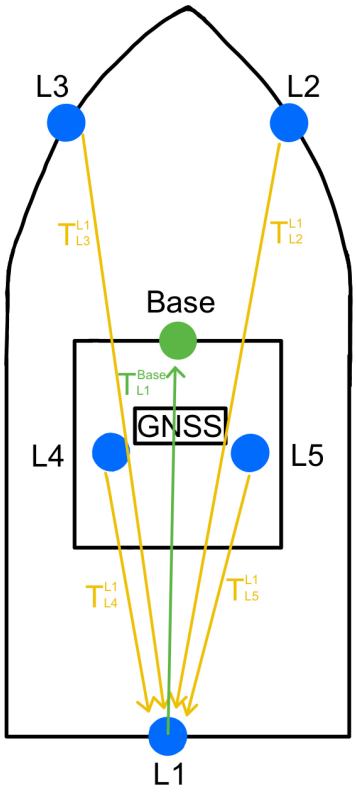


Figure 3.3: Overview of the boat's LiDAR positions and the tranformations between them. These can be seen with yellow arrows between L1 and the other LiDARs, while the green arrow represents the transformation between L1 and the base.

After applying each transformation to its corresponding point cloud, the clouds were merged together. Afterwards, one last transformation was performed on the point cloud to the base of the boat. It was done to easier visualise each frame and to make the labelling smoother in the annotation tool. The transformation from L1 to the base is shown as a green arrow in Figure 3.3 and the matrix can be seen in Appendix A as T_{L1}^{Base}.

The transformation with T_{L1}^{Base} should make it possible for the following method, in this chapter, to be used regardless of what kind of LiDARs are used. A requirement is that the point cloud covers an angle of 360° around the vessel and that the LiDARs have roughly the same installation as those that the model was trained for. The transformation matrices to L1 need to be adapted to make sure that the coordinate system matches the system of the training data.

3.1.2 Annotation

The collected data were not annotated from the beginning. Therefore, different annotation tools were investigated to label the point clouds in the data set. The SemanticKITTI data set consists of point clouds that have been collected by a car, and a point cloud labeller has been created for that data set [20].

This tool made it possible to label multiple frames at the same time. The program stacked the frames on each other and thus created a map of point clouds. However, only 50 frames at a time could be labelled. More frames became difficult to handle, partly due to the changes in the marine environment (the water's constant movements) and the addition of more points that overlapped which made it hard to distinguish between different objects. Furthermore, the LiDARs recorded parts of the own boat, which covers other objects in frames where the boat was in motion during annotation. There was also an issue with the transformation matrices between each frame since it is not certain that all of the LiDARs and the GNSS sensor record a frame at the exact same time, leading to timing errors.

Labels were created for the marine data set where the annotations were made with 14 classes. However, all the classes were most likely not necessary for marine conditions. To evaluate the performance of the number of classes, three constellations of classes were made. These three sets can be seen in Table 3.1.

Set	Number of classes
Large	14
Medium	8
Small	3

Table 3.1: The size of the three sets used.

The class colours for the Large set, which the annotations were made with, can be seen in Figure 3.4.

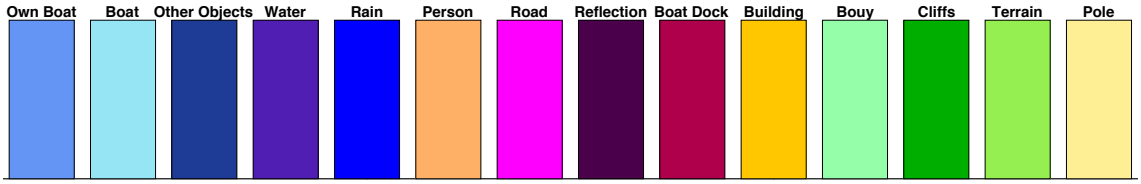


Figure 3.4: Class colours for Large set.

For the Medium set the classes on land were merged to form the class 'Ashore', and 'Pole' was added to 'Boat dock'. The reduced set can be seen in Figure 3.5. There is also the Small set where there are only three classes; 'Own Boat', 'Water' and 'Other', which is visualised in Figure 3.6.

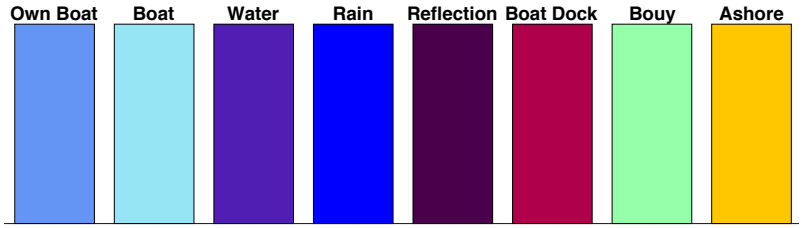


Figure 3.5: Class colours for Medium set.

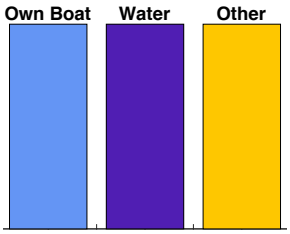


Figure 3.6: Class colours for Small set.

3.1.3 2D projection

The point clouds are built with Cartesian Coordinates (x, y, z) which makes it possible to view all the data points in 3D. These clouds take a lot of memory space and can be very time-consuming when the purpose is to train a network to recognize different objects. The frames in the data set were of sizes around 70 000 to 150 000 points, which could make it difficult to use a point-based method since it could make it hard to reach efficient processing, especially in real-time applications [12]. Hence, a projection-based method was used to reduce the time-consuming part, increase efficiency and enable the use of CNN.

The 3D point clouds were transformed into 2D spherical projections that created 2D panorama images that enabled higher computational efficiency during training. An example of such an image can be seen in Figure 3.7. The projection calculations will be described further down in this section.

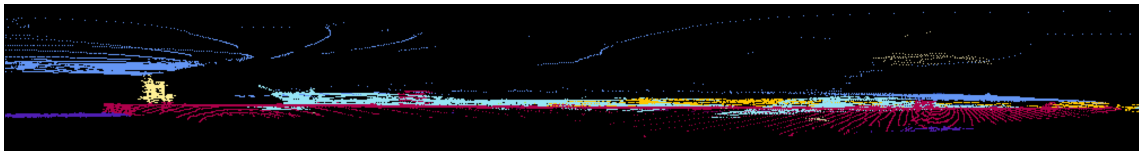
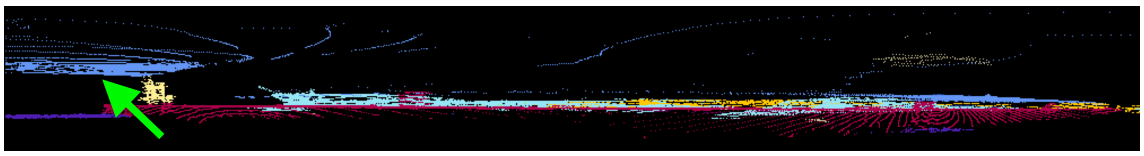
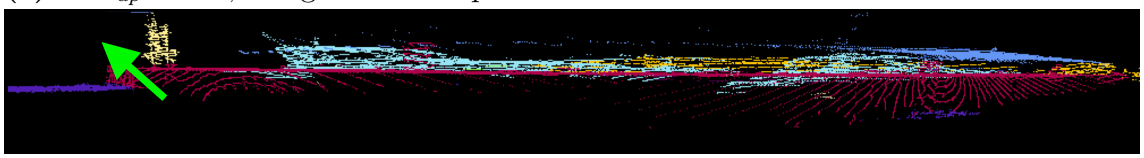


Figure 3.7: Left half of a 2D spherical projection image of labelled 3D point cloud. The original size can be found in Appendix B in Figure B.1.

The size of the 2D images needed to be optimised, which means that as many points as possible needed to fit in the images while minimising the number of empty pixels (unlabelled with depth 0). This optimisation was partly done by choosing the FoV, which is usually a set value for a LiDAR. However, since the point clouds were merged from several LiDARs with different positions and directions their FoV did not describe the point clouds' FoV. Therefore, the optimal value for the FoV was decided by testing different values. FoV_{up} was set to 90° since smaller values caused points to disappear and larger values were not necessary due to a horizontal FoV of 360°. In Figure 3.8 one image can be seen where $FoV_{up} = 90°$ and one with $FoV_{up} = 30°$. The green arrows highlight that the roof of the boat (blue colour) has been cut out with the smaller FoV, which means that data points have disappeared.



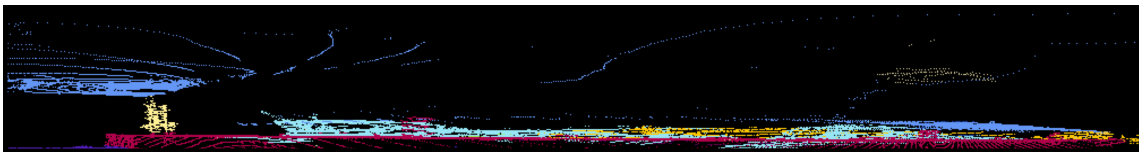
(a) $FoV_{up} = 90°$, the green arrow points at the roof of the boat.



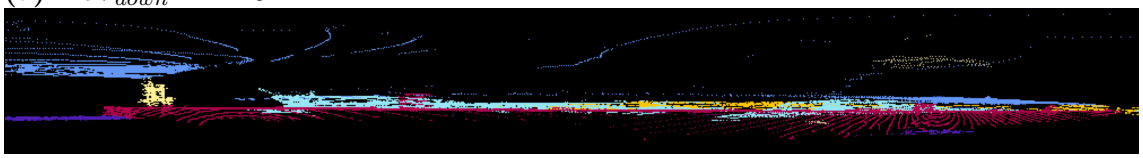
(b) $FoV_{up} = 30°$, the green arrow points at the area where the roof of the boat has disappeared.

Figure 3.8: Left half of two 2D images with different values of FoV_{up}. FoV_{down} is on both images set to −40°. The original size can be found in Appendix B in Figure B.2.

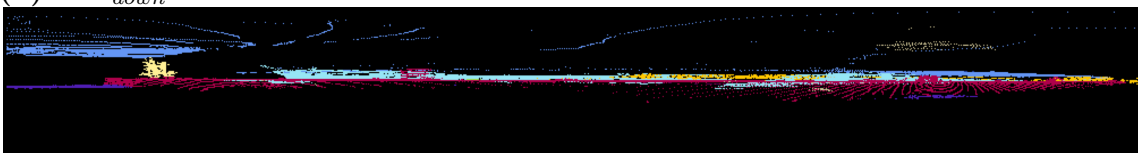
FoV_{down} was more challenging to decide in the term of minimising the number of empty pixels (black space) while not removing any points. After testing different values it was decided that $FoV_{down} = -40°$ was the most optimal. Some of the testings can be seen in Figure 3.9, where three different angle values of FoV_{down} are shown.



(a) $FoV_{down} = -10°$.



(b) $FoV_{down} = -40°$.



(c) $FoV_{down} = -90°$.

Figure 3.9: Left half of three 2D images with different values of FoV_{down}. FoV_{up} is on both images set to 90°. The original size can be found in Appendix B in Figure B.3.

In Figure 3.9a an angle of $-10°$ is shown which clearly shows that many of the lower data points were missed. While in Figure 3.9c, where the angle value was set to $-90°$, more black space can be seen in the lower part of the image than in Figure 3.9b where the angle value is $-40°$. More angles were tried closer to $-40°$ to estimate the value as thoroughly as possible. Some of the points were quite small and were difficult to view in the upper and lower parts of the panorama images when the points were more spread out and not located close to one another.

To achieve the 2D images a spherical transformation was needed to be made on the 3D point clouds. This means that the Cartesian coordinates were transformed to spherical coordinates (R, ϕ, θ) which were used to create an image. To project the 3D points to a 2D image the depth (R) was calculated first.

$$R = \sqrt{x^2 + y^2 + z^2} \tag{3.1}$$

Furthermore, this was used together with the 3D point coordinates to receive the values for the angles,

$$\phi = -\arctan2\left(\frac{y}{x}\right) \tag{3.2}$$

$$\theta = \arcsin\left(\frac{z}{R}\right) \tag{3.3}$$

These angles made it possible to calculate the projection (X_{proj}, Y_{proj}). An illustration of the spherical projection can be seen in Figure 3.10.

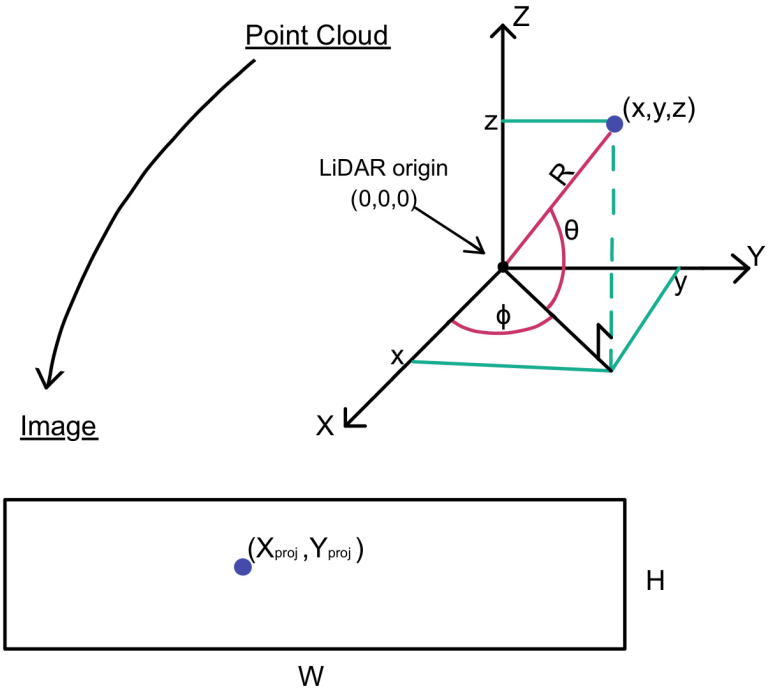


Figure 3.10: Coordinate system over one point in the point cloud and its projection to a 2D image with size H x W.

The projections were calculated by [21],

$$X_{proj} = \left(\frac{1}{2}\left(\frac{\phi}{\pi} + 1\right)\right) \cdot W, \qquad \phi \in [-\pi, \pi] \tag{3.4}$$

$$Y_{proj} = \left(1 - \left(\frac{\theta + |FoV_{down}|}{FoV}\right)\right) \cdot H, \qquad \theta \in [FoV_{down}, FoV_{up}] \tag{3.5}$$

where ϕ and θ can be seen in Equations (3.2) and (3.3). $FoV = |FoV_{down}| + |FoV_{up}|$ which all were translated to radians. The width and height of the image were set to $W = 2048$ and $H = 128$. This image size was chosen because a bigger size would take up a lot of memory and a smaller size made images with too low resolution. Furthermore, the smaller size also meant that the number of max pooling layers needed to be reduced.

In Figure 3.11 an example of one projected point can be seen where the calculation steps are explained in further detail. The 3D points were translated to angles where the LiDAR position was in the centre of the 2D plane. Hence, the origin was translated to the upper corner of the image plane since it is preferred for computer vision tasks to have it in one of the corners [21]. After the 3D point has been translated the projected image needs to be adjusted to the LiDAR system. This was done through normalisation and scaling to keep as many points as possible in the projection when one dimension is ignored. The normalisation and scaling were made with the help of the vertical FoV and the image size.

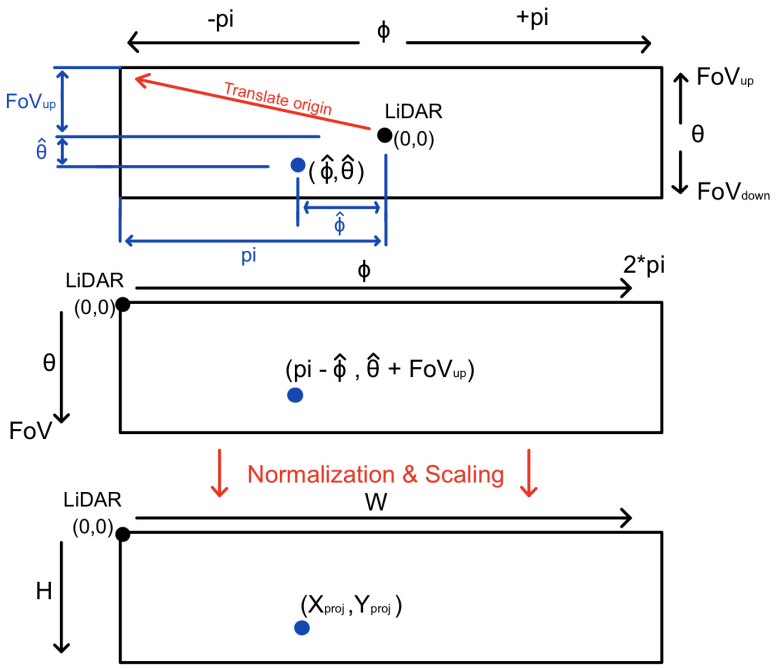
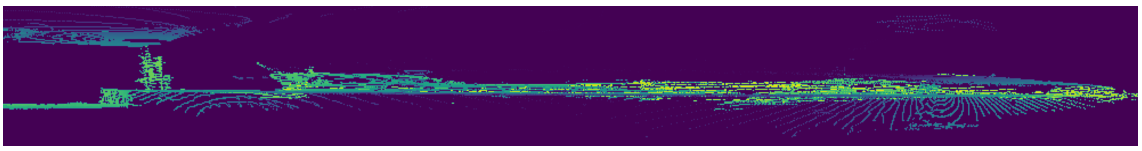


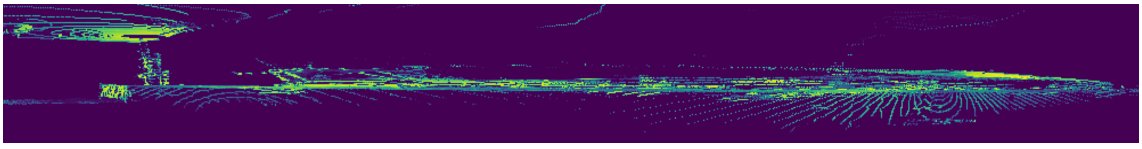
Figure 3.11: The calculation steps of projection for one point in the 2D image from the 3D point cloud.

3.1.4 Images used in training

With the 2D projection a depth image and an intensity image were created for each frame. In the same way as RGB images have three channels the depth and intensity images were merged to create 2D images with two channels. These were then used as input to the network. They can be viewed separately in a half format in Figure 3.12. A third 2D image was also created and used as ground truth in the training loop, which can be seen in Figure 3.9b.



(a) Left half of a depth image, where the blue is closer to the sensor and yellow further away.



(b) Left half of an intensity image, where the yellow parts have higher intensity values and darker blue parts lower intensity values.

Figure 3.12: Left half of depth and intensity 2D images that are used as input to the network architecture.

3.1.5 Imbalanced data set

After labelling the data and projecting the clouds onto 2D images it became clear that the data set suffers from highly imbalanced classes. In Figure 3.13 the 'Unlabelled' class can clearly be seen with a major difference compared to the other classes. The 'Unlabelled' class represents the black pixels in the 2D projections, which are areas where there are no points to project. Since the areas where there are no points to project are unimportant for the segmentation of the 3D cloud this class was ignored during the training.

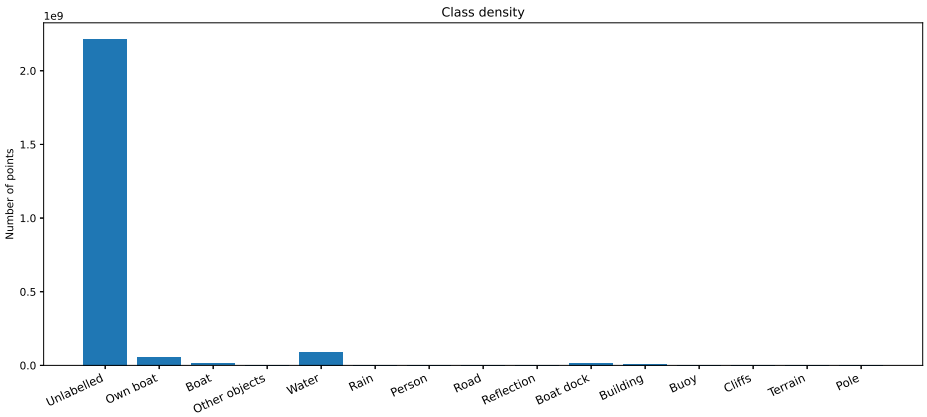


Figure 3.13: Class density for 16 classes.

By ignoring the 'Unlabelled' class the other classes can be seen more clearly, as shown in Figure 3.14. As can be seen in the graph there are still some classes that have a large advantage compared to other classes. The LiDARs detect most of the own boat and water near the boat which makes the densities for these classes larger.

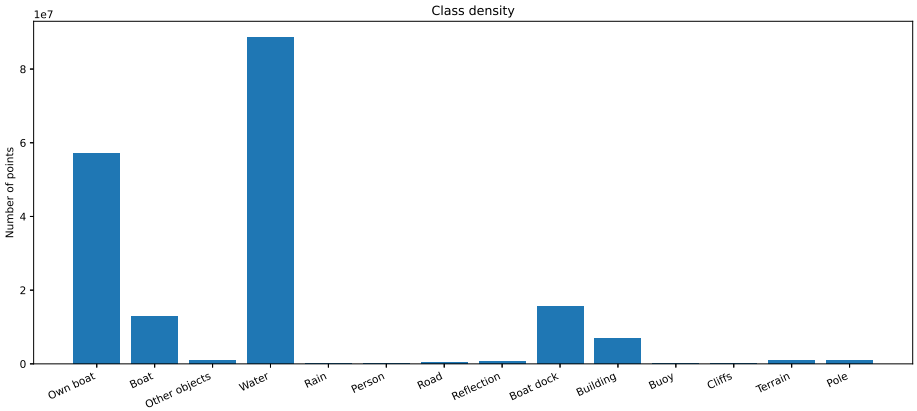


Figure 3.14: Class density when ignoring class 'Unlabelled'.

It was decided to try and reduce the number of classes to get a better distribution between them, as mentioned briefly in Section 3.1.2. Figure 3.15 visualises the class densities after merging some of them together. The classes 'Road', 'Other objects',

'Person', 'Building', 'Cliffs' and 'Terrain' were merged to 'Ashore'. The class 'Pole' was added to 'Boat dock' since the poles in question were placed on the docks.

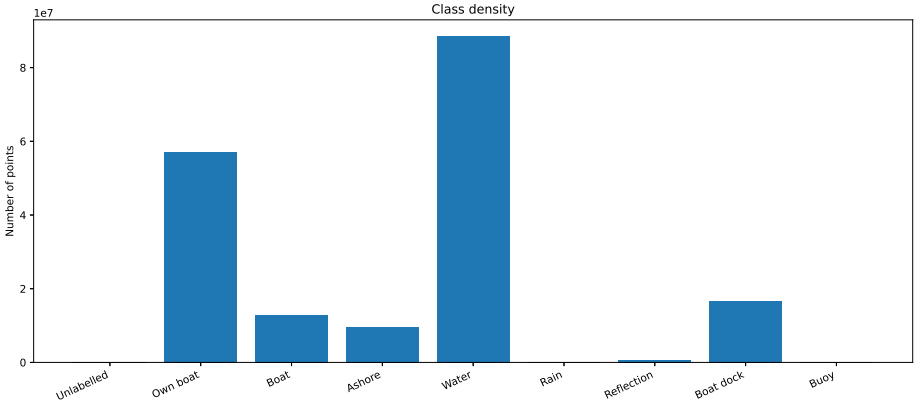


Figure 3.15: Class density with eight classes.

To merge classes even further only 3 classes were kept, as seen in Figure 3.16. Here the classes are 'Own boat', 'Water' and 'Other'. The class 'Rain' was added to water and every other class, except 'Own boat' and 'Water', were merged to 'Other'. This was done to see how well the model can predict the difference between where the boat can and cannot travel.

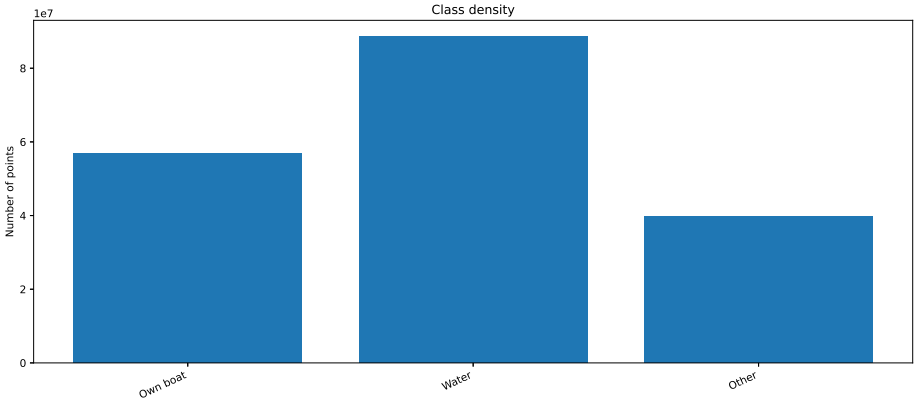


Figure 3.16: Class density with three classes.

3.2 Data Collection

During labelling and analysing the data that was previously collected it became clear that the data sets were quite similar to each other. Most of the recordings were in two distinct parts of the same harbour, something that would make the model recognize those features well but have a harder time with features in other harbours. The recordings that were not in that harbour were of high-speed movement on open

water without any other land masses.

To ensure that the data set would achieve more diversity, new data was collected. New recordings were made in another harbour that had docks and land of other structures as well as vessels of varied sizes, such as large fishing boats. Furthermore, recordings were also made when moving and being near different objects, for example buoys, moving vessels and cliffs.

3.3 U-Net

It was decided to use the U-Net architecture because it is a CNN that can be used for multi-class segmentation and has proven to be reliable. Since the training data have two channels, a depth channel and an intensity channel, the network's input size was the size of the image × 2. The encoder's channels, used as the number of channels for the convolution layers, were 2, 64, 128, 256 and 512, where each downsampling doubled the number of channels. The channels for the decoder were similar but reversed and the last block ended with as many channels as there were classes, to get one probability map for each class. An overview of the used network can be seen in Figure 3.17.

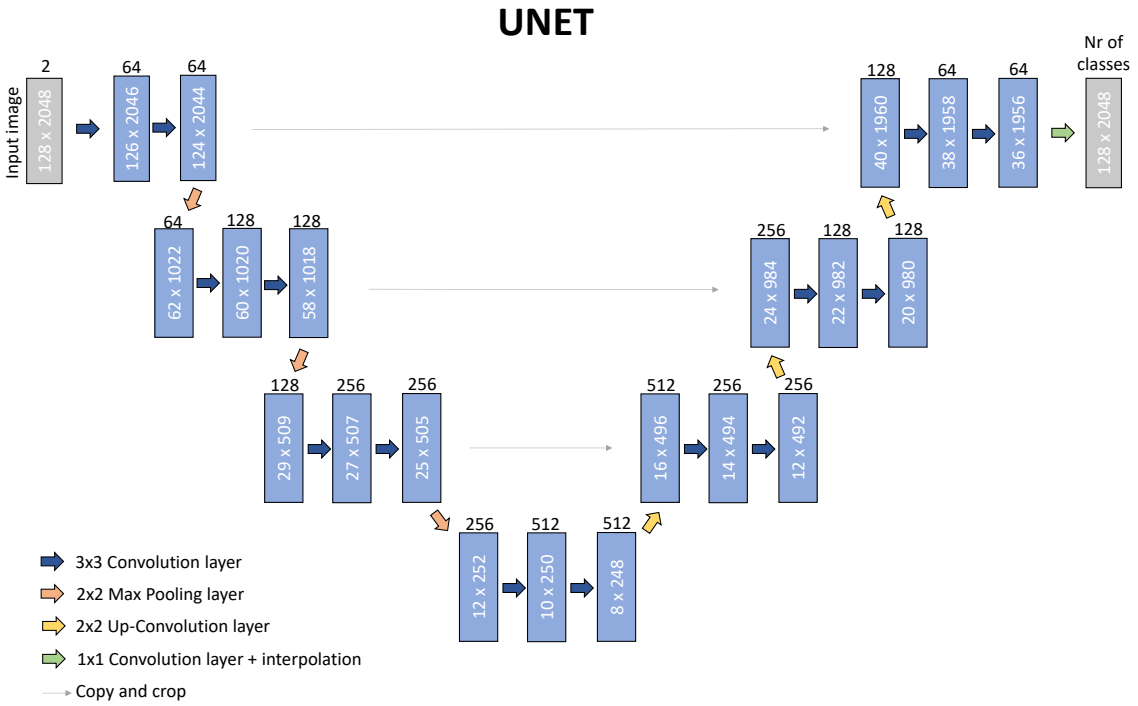


Figure 3.17: The implemented U-Net structure where the encoder can be seen on the left side and decoder on the right side.

To get one feature map with class predictions a softmax function was added along the channel dimension to get values between 0 and 1 on the probabilities. An argmax function was applied along the same dimension to receive the index of the highest

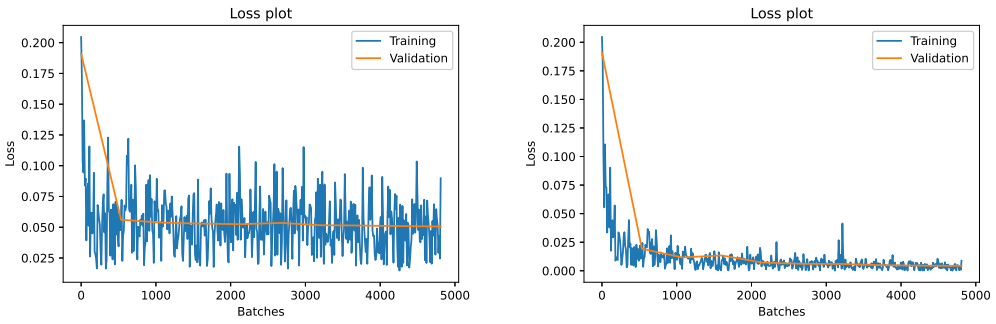
probability for each pixel. The index corresponded to a label that was connected to a class, thereby giving each pixel a predicted class.

3.4 Training and evaluation of network

The training and evaluation were done on a computer with 32 GB RAM and an Nvidia Quadro RTX 4000 GPU [22]. During training, the optimizer used was the Adam optimizer. It was used because it is an adaptive optimizer that automatically tunes the learning rate value during training [23]. Furthermore, the data set consisted of almost 10 000 frames with around 70 000 to 150 000 points per frame. During each epoch, there were several validations performed with a validation data set (20% of data set), which was in order to gather more results from the validation set. The test data set was two recordings that made up approximately 10% of the data, which were used for evaluation of the model.

3.4.1 Learning rate

A few different values were tested for the learning rate to get a good result on the model. At first, a learning rate of 0.001 was tried and the loss can be seen in Figure 3.18a. The loss was decreasing fast in the beginning and thereafter converged around 0.05, hence the learning rate was a bit too high. This caused the optimizer to not find the minimum point due to too drastic updates. With a lower learning rate, of 0.0001, the loss managed to decrease in a better way while not fluctuating as much, which can be seen in Figure 3.18b. It was decided to not use an even lower learning rate since that could cause the loss to not decrease as fast, thereby making the training longer.



(a) Loss with learning rate 0.001. **(b)** Loss with learning rate 0.0001.

Figure 3.18: Graph over loss with two different learning rates over the training time.

3.4.2 Accuracy

To analyse the model performance, accuracy was calculated for all classes. This was done by using the Confusion Matrix. The accuracy for all classes was calculated

as shown in the example related to Figure 2.4. The accuracies for each class were calculated with,

$$acc_{class1} = \frac{Correct\ classified\ labels\ for\ class1}{The\ sum\ of\ the\ ground\ truth\ classification\ for\ class1} \tag{3.6}$$

Using the scenario shown in Figure 2.4, it would mean that the accuracy for class1 would be $acc_{class1} = \frac{4}{4+2+1} = 0.57$. This calculation has been done for each of the classes during the validation step.

3.4.3 Loss function

In the 2D projection images there was a clear problem with class imbalance as well as for the 3D point clouds. The unlabelled areas in the images were a lot larger than the labelled areas. The LiDARs managed to detect more points closer to themselves, which created more points in the classes 'Own boat' and 'Water'. To create training that compensated for this the loss function was of considerable importance. At first CE loss was used as it works well for image segmentation. The class 'Unlabelled' was ignored in the loss function, as mentioned in Section 3.1.5. Even by ignoring the background class, the data set was still rather imbalanced and other loss functions needed to be tried.

Focal loss, as introduced in Section 2.4.4, is particularly good at handling class imbalances. It is based on CE and the background class 'Unlabelled' was ignored in the same way as with CE loss. The modulating term $(1 - p_i)^\gamma$ was multiplied with the CE and the value γ was set to 2 to handle the class imbalance in the best conceivable way [17].

3.4.4 2D projection to 3D cloud

After training and evaluation were finished the 2D images needed to be transformed back to the 3D space again to visualise the data in a 3D LiDAR point cloud. The transformation was performed by backwards computation with spherical coordinates. The angles could therefore be calculated by using Equation (3.4) and (3.5). Since the depth was known, the 3D coordinates could be calculated by,

$$x = R\cos(\theta)\cos(-\phi) \tag{3.7}$$
$$y = R\cos(\theta)\sin(-\phi) \tag{3.8}$$
$$z = R\sin(\theta) \tag{3.9}$$

where R is the depth, $\phi \in [-\pi, \pi]$ and $\theta \in [FoV_{down}, FoV_{up}]$.

4

Results

In this section the results are presented with validation accuracy graphs from training and accuracies on test data. The section is divided into the three different class sets, execution time and the models' predictions in heavy rain. All models were trained for three epochs, with a learning rate of 0.0001.

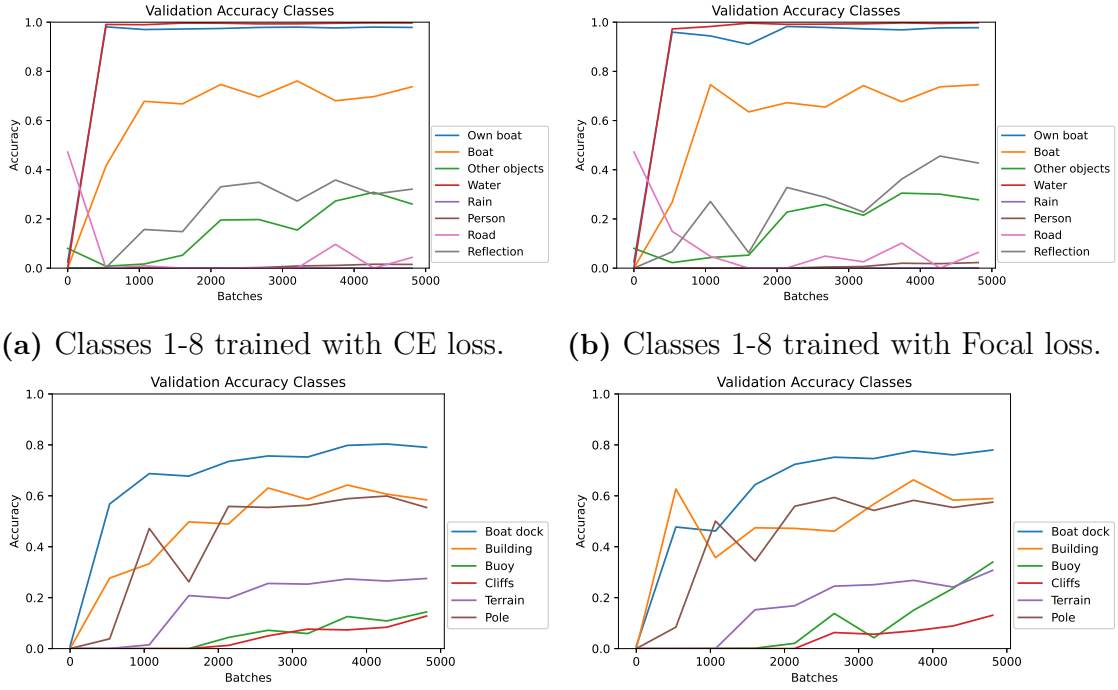
4.1 Large set

The validation accuracy and loss from the last validation, for the large set of classes, can be seen in Table 4.1. The accuracy reached the same value for both of the loss functions. However, the loss is slightly lower for Focal loss.

Loss function	Accuracy	Loss
CE	95.2%	0.010
Focal	95.2%	0.004

Table 4.1: The last validation accuracy and loss, for each of the loss functions. With the large set of classes.

Figure 4.1 visualises graphs with the validation accuracy from training on the large set of classes. The graphs are divided depending on the class and which loss function was used during training.



(a) Classes 1-8 trained with CE loss.

(b) Classes 1-8 trained with Focal loss.

(c) Classes 9-14 trained with CE loss.

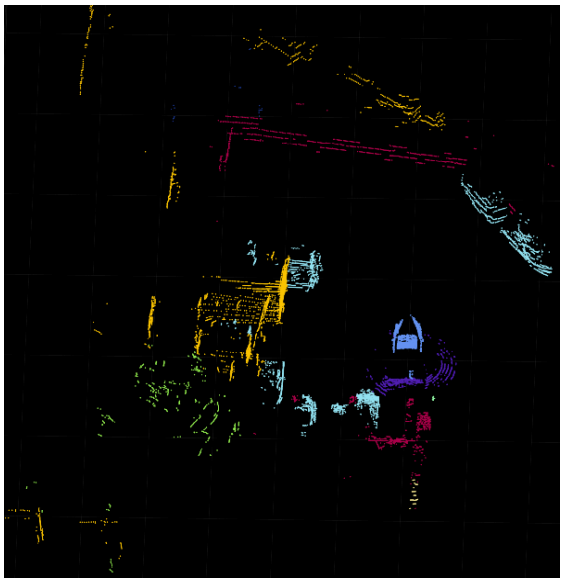
(d) Classes 9-14 trained with Focal loss.

Figure 4.1: The validation accuracy for each class, using the large set of classes. Figures (a) and (c) are with CE loss, while figures (b) and (d) are with Focal loss.

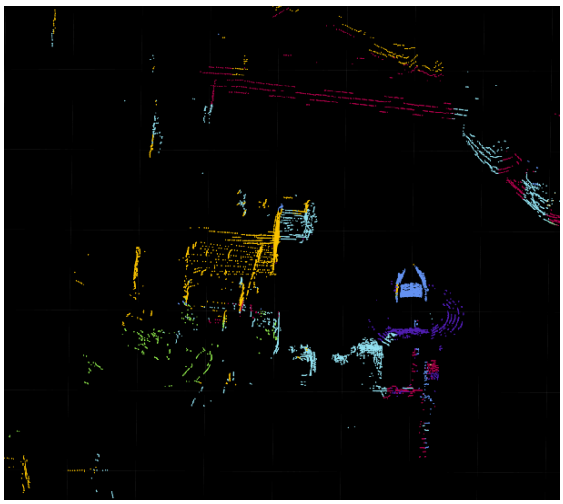
For both of the loss functions the classes 'Own boat' and 'Water' reached a validation accuracy higher than 95%, as can be seen in Figures 4.1a and 4.1b. Other classes that had a decent accuracy were the classes 'Boat', 'Boat dock', 'Building' and 'Pole', ranging between 55% and 80%. There was not a big difference in the validation accuracy depending on the loss function for either of those classes. However, for classes that have a smaller density the validation accuracy increased with Focal loss compared to CE loss. The class 'Buoy' is such a class, as can be seen in Figures 4.1c and 4.1d. The validation accuracy with CE loss reached a value of around 15% while the model with Focal loss finished at around 40%.

4.1.1 Segmentation on test data

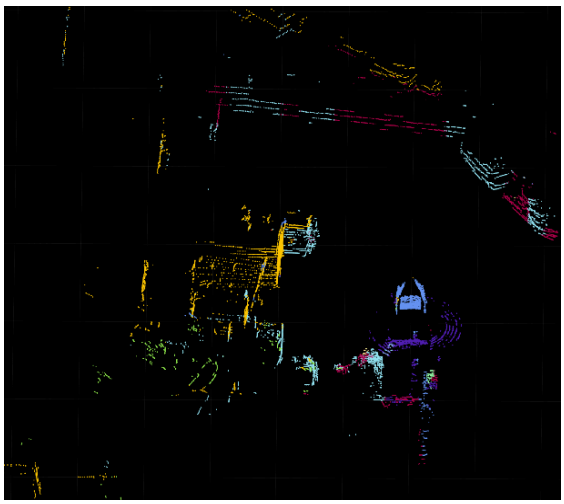
To visualise the performance, in more than validation accuracy and loss, test data were used for predictions. The 3D point clouds for ground truth and predictions of one test frame are visualised in Figure 4.2. Both of the models managed to predict the majority of the points with the right class. However, there are some points that were falsely predicted where the objects that are behind or in front of other objects get mixed up.



(a) Ground truth.



(b) Segmentation from the model trained with CE loss.



(c) Segmentation from the model trained with Focal loss.

Figure 4.2: Ground truth and segmented 3D point clouds for the large set of classes.

Furthermore, to calculate average test accuracy for the classes two recordings from different locations were used as test data. The segmented 3D point cloud from Figure 4.2 is from a recording that is referred to as 'Still', as the boat stays in the same position. The second recording was made while the boat was moving towards a different harbour, referred to as 'Moving'. The mean accuracy from all frames per recording can be seen in Table 4.2. If a class is not present it means that neither of the recordings have any objects of that class and a '-' means that the class is absent from one of the recordings. Clearly, classes with higher class density get higher accuracy, such as 'Water' and 'Own boat'. The two models mostly have similar accuracy for the different classes. It is only for a few classes, for example 'Pole' and 'Boat dock', that the model with CE loss performs better.

Class	Still		Moving	
	CE loss	Focal loss	CE loss	Focal loss
All	88.3%	86.0%	96.4%	95.5%
Own boat	97.2%	97.2%	99.0%	98.8%
Boat	82.9%	77.2%	75.7%	73.0%
Water	99.0%	99.6%	99.5%	99.6%
Reflection	4.5%	3.7%	-	-
Boat dock	61.4%	41.5%	88.7%	78.0%
Building	89.1%	90.6%	-	-
Buoy	2.2%	1.9%	19.6%	19.5%
Terrain	56.6%	62.4%	-	-
Pole	40.2%	29.4%	-	-

Table 4.2: Mean test accuracy for CE and Focal loss with the large set of classes, over two recordings.

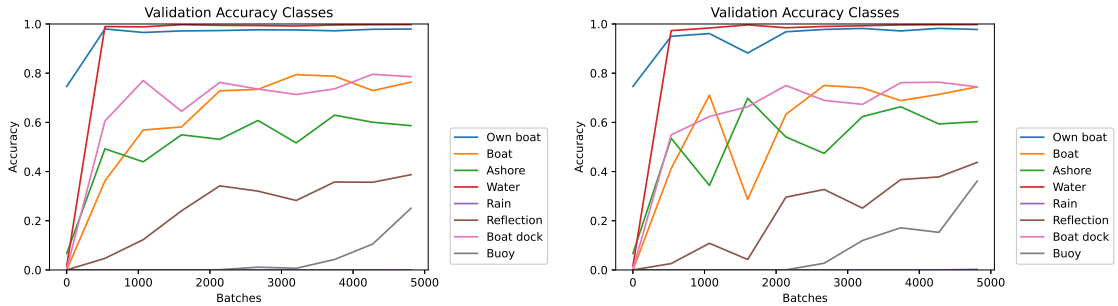
4.2 Medium set

The medium set contains, as mentioned, eight classes. The last values of the validation accuracy and loss can be seen in Table 4.3. Both the accuracy and loss reached similar values as for the case with the large set of classes.

Loss function	Accuracy	Loss
CE	95.6%	0.009
Focal	95.3%	0.003

Table 4.3: The last validation accuracy and loss, for each of the loss functions. With the medium set of classes.

Similarly to the case with the large set of classes, the validation accuracy for the medium set did not differ much when comparing the two loss functions. The accuracy per class can be seen in Figure 4.3.



(a) The medium set of classes trained with CE loss.

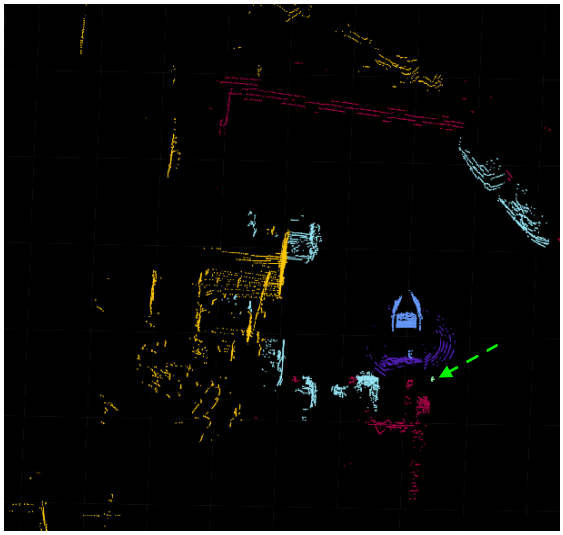
(b) The medium set of classes trained with Focal loss.

Figure 4.3: The validation accuracy for each class, using the medium set of classes. Figures (a) is with CE loss and Figure (b) is with Focal loss.

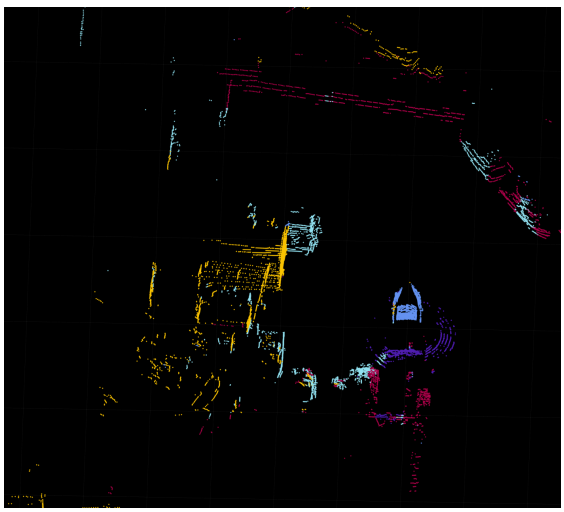
As can be seen in the graphs in Figure 4.3 the classes 'Own boat' and 'Water' had a high validation accuracy, over 95%. The new class 'Ashore' finished with an accuracy of around 60% for both of the loss functions. The class that had the largest change in regard to the loss function was 'Buoy', similarly to the previous case the model with Focal loss had a higher accuracy compared to CE loss.

4.2.1 Segmentation on test data

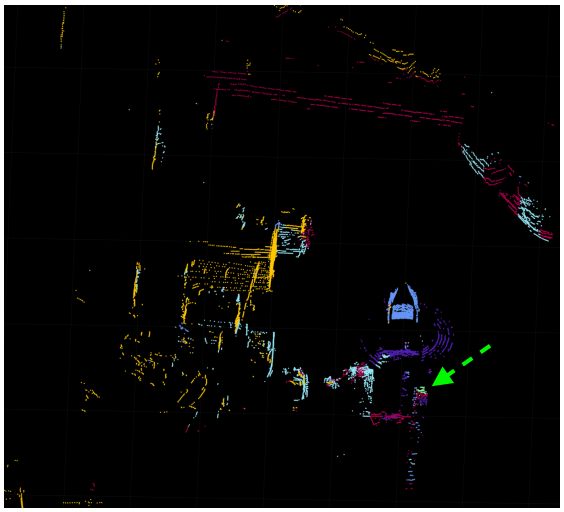
In Figure 4.4 the ground truth and segmented clouds for one frame can be seen. Since, with the medium set, the amount of classes on land has been reduced it is easier for the model to predict the points on land. Instead of having several classes with features on land it is only the yellow 'Ashore' class.



(a) Ground truth.



(b) Segmentation from the model trained with CE loss.



(c) Segmentation from the model trained with Focal loss.

Figure 4.4: Ground truth and segmented 3D point clouds for the medium set of classes with green arrows marked at class 'Buoy'.

Comparing Figures 4.4b and 4.4c it is clear that the loss functions did not make much difference. However, behind the boat a buoy can be seen in the ground truth and the model with Focal loss has tried to predict that class nearby, marked by a green arrow. The model with CE loss predicts everything behind the boat as either 'Water' or 'Boat dock'.

As for the large set, the mean accuracies for the two recordings were calculated for the medium set, shown in Table 4.4. The recordings clearly had fewer classes that were not present at all compared to the large set of classes, since most of them were merged with other classes.

Class	Still		Moving	
	CE loss	Focal loss	CE loss	Focal loss
All	89.1%	87.8%	96.2%	96.0%
Own boat	96.9%	97.2%	99.0%	99.2%
Boat	78.8%	75.7%	74.0%	74.4%
Ashore	86.7%	90.0%	-	-
Water	98.6%	98.4%	99.6%	99.7%
Reflection	0.4%	1.2%	-	-
Boat dock	77.3%	62.1%	89.1%	84.3%
Buoy	2.5%	2.7%	12.4%	28.4%

Table 4.4: Mean test accuracy for CE and Focal loss with medium set of classes, over two recordings.

The 'Ashore' class was present in the Still recording and reaches a high accuracy at 86.7% for CE loss and 90% for Focal loss. However, the class was not present in the second recording. A noticeable difference when comparing the loss functions is the class 'Buoy' in the Moving recording. It had a higher accuracy with Focal loss, but overall the class had a low accuracy compared to other classes. It should also be noted that the recording Moving has a higher accuracy for all of the classes compared to the Still recording.

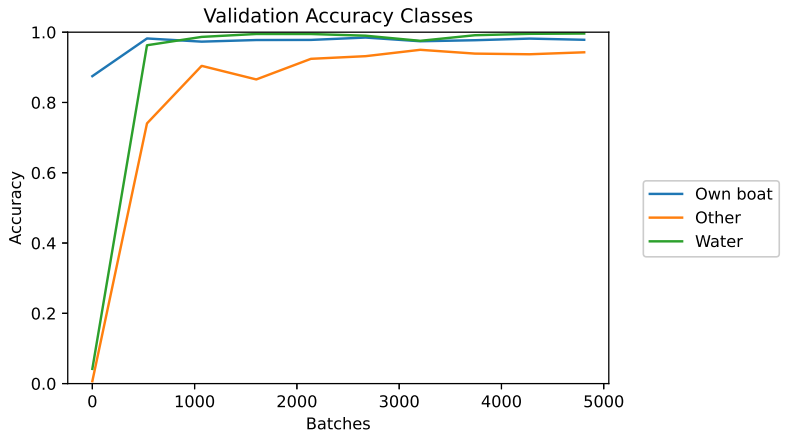
4.3 Small set

The small set of classes only had the classes 'Own boat', 'Water' and 'Other'. It is clear when looking at the values in Table 4.5 that the last validation accuracy is very high.

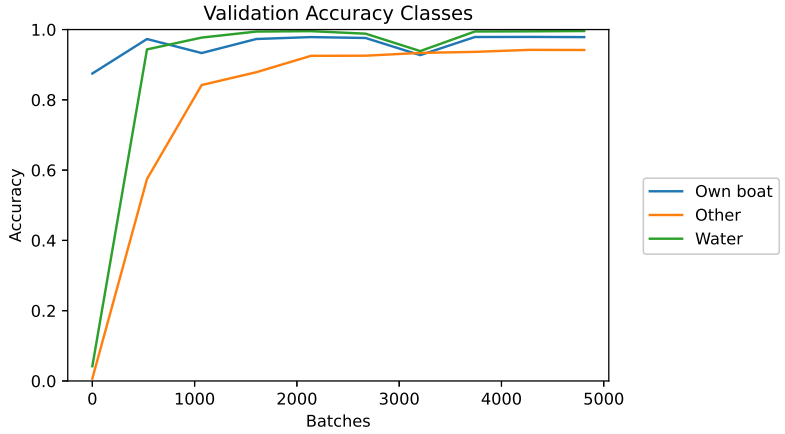
Loss function	Accuracy	Loss
CE	98.8%	0.002
Focal	98.7%	0.001

Table 4.5: The last validation accuracy and loss, for each of the loss functions. With the small set of classes.

In Figure 4.5 the validation accuracies for the three classes can be seen. Clearly, all of the classes reached very high accuracies at around 95% or higher. The different loss functions did not make a significant difference in this case.



(a) The medium set of classes trained with CE loss.

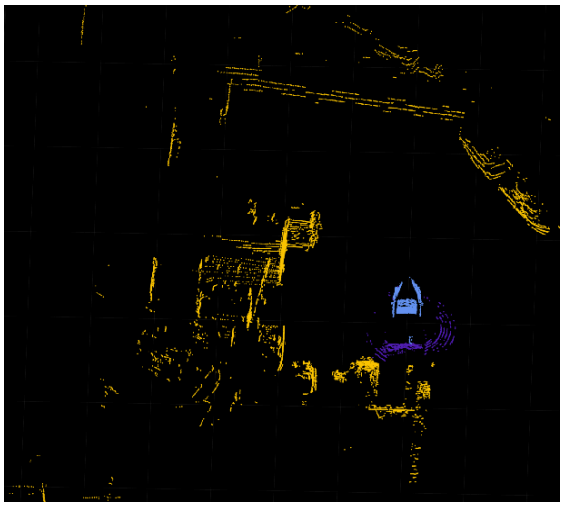


(b) The medium set of classes trained with Focal loss.

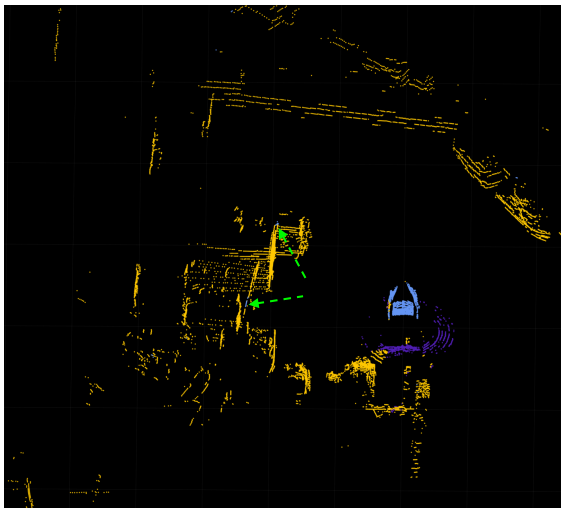
Figure 4.5: The validation accuracy for each class, using the small set of classes. Figures (a) is with CE loss and Figure (b) is with Focal loss.

4.3.1 Segmentation on test data

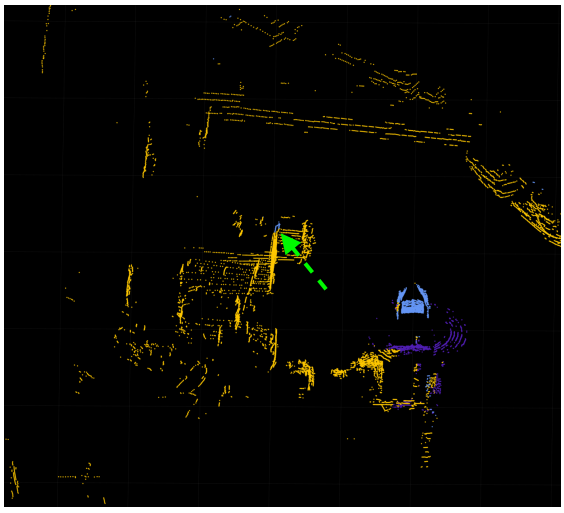
The segmented clouds in Figure 4.6 show how well the model managed to predict when using the small set of classes. There are some points that are falsely predicted, mostly behind the boat where the model mixed the classes 'Water' and 'Other'. However, there are a few points up to the left of the boat where the model predicted 'Other' as 'Own boat'. This is marked with green arrows in the images.



(a) Ground truth.



(b) Segmentation from the model trained with CE loss.



(c) Segmentation from the model trained with Focal loss.

Figure 4.6: Ground truth and segmented 3D point clouds for the small set of classes. The green arrows indicate the points where the model classifies 'Other' as 'Own Boat'.

As for the other cases, the mean test accuracies were calculated for the case with the small set of classes, shown in Table 4.6. It is clear that all of the classes have high accuracy, both for Focal and CE loss. The class 'Other' has a slightly lower accuracy than the others, except for CE loss in the Still recording.

Class	Still		Moving	
	CE loss	Focal loss	CE loss	Focal loss
All	98.2%	96.9%	99.1%	99.0%
Own boat	97.0%	97.1%	98.8%	99.0%
Other	99.0%	96.6%	98.2%	97.5%
Water	98.0%	98.1%	99.6%	99.4%

Table 4.6: Mean test accuracy for CE and Focal loss with the small set, over two recordings.

4.4 Segmentation time

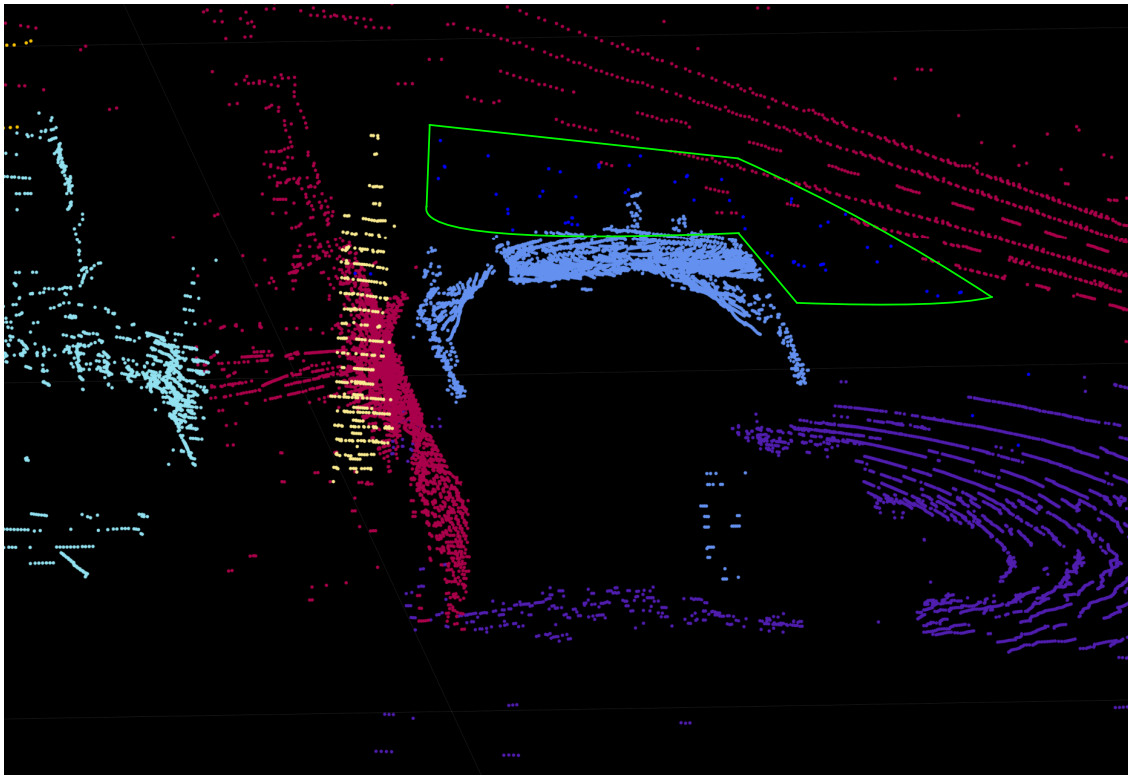
The mean segmentation time was computed by measuring the time it takes to project a point cloud to 2D, make a prediction on the projection and then transform the prediction back to a 3D point cloud. The segmentation was done using Python and a recording with 400 frames. The mean time for each case with Focal loss is shown in Table 4.7. Furthermore, to investigate the time the model takes to make a prediction compared to the 2D projections, the mean prediction time was also computed, which can be seen in Table 4.7. When comparing the columns in the table, the prediction time constitutes the majority of the mean segmentation time.

Case	Mean segmentation time [s]	Mean prediction time [s]
Large	0.281	0.260
Medium	0.292	0.272
Small	0.278	0.258

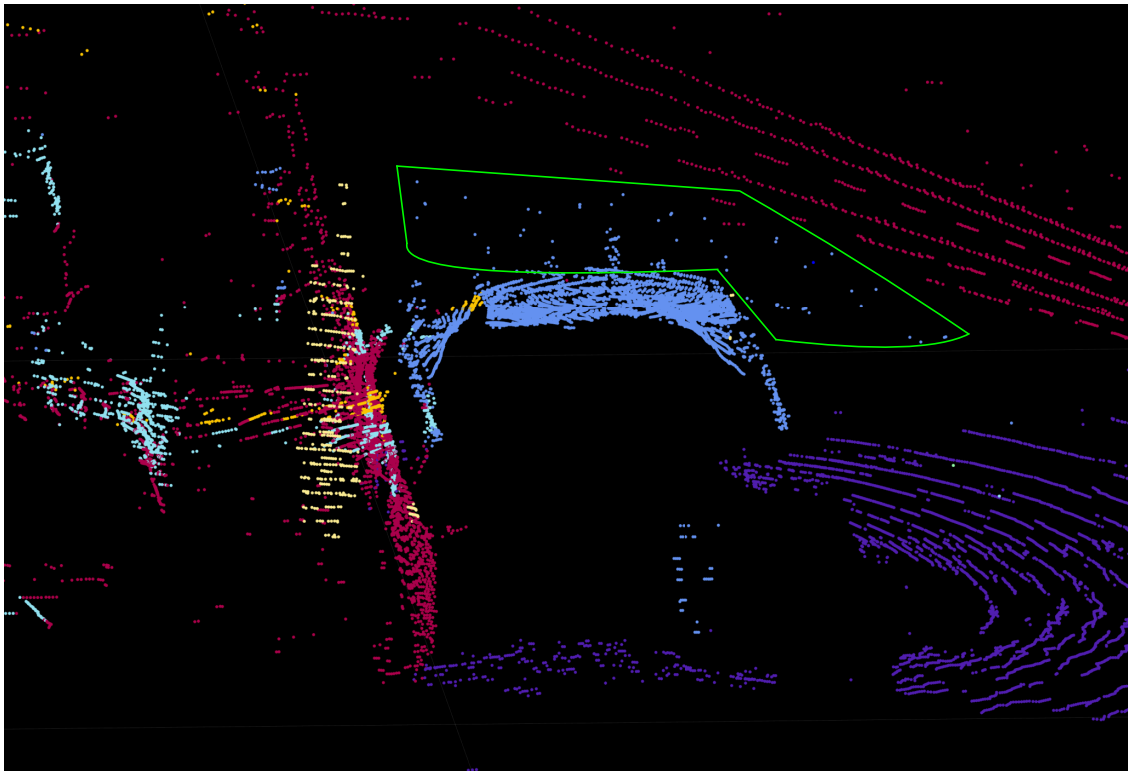
Table 4.7: Mean segmentation time and mean prediction time in seconds.

4.5 Segmentation in heavy rain

The model was trained on a couple of recordings with rain, where the rain was mostly seen as scattered dots around the LiDARs' positions. In Figure 4.7 the ground truth and prediction of a frame with rain can be seen where the rain is inside the lines of green colour. The rain is barely visible as dots around the roof of the boat. In the ground truth image they have a dark blue colour, whereas in the segmented image they have the same colour as the own boat.



(a) Ground truth point cloud.



(b) Segmented point cloud.

Figure 4.7: Ground truth and segmented 3D point cloud of a frame with rain. The rain can be noticed inside the lines with green colour.

Clearly, the LiDARs manage to detect features through the rain. In the figures, the dock can be seen as red, a pole as yellow and another boat as light blue. In Figure 4.7b the model clearly manages to predict these objects, with a few errors to the left in the 3D cloud.

5
Discussion

The following chapter is divided into a comparison between the different cases and loss functions, a discussion of the input data and an evaluation of real-time segmentation.

5.1 Comparison loss function and class sets

In the following section, the loss and accuracy values as well as the predicted point clouds in the result section is discussed.

5.1.1 Large set

For the large set of classes, it is clear from Table 4.1 that there was no difference in the last validation accuracy. The models reached a high accuracy at 95.2%. The model with Focal loss reached a slightly lower loss than CE, but both of the models reached a low loss. However, since the data set was imbalanced it was of higher interest to analyse the accuracy per class, which was shown in Figure 4.1. The model was very good at predicting the classes 'Water' and 'Own boat', which is reasonable since those classes have the highest density. The predictions were also high for the classes 'Boat', 'Boat dock', 'Building' and 'Pole', which have accuracies that range between 60% and 80%. The Focal loss performs very similarly for those classes, but for the classes that had a lower density a difference can be noticed.

When comparing the class accuracy graphs for CE and Focal loss in Figure 4.1, it could be seen that the accuracy for 'Buoy' was around 40% with Focal loss. However, it was at 15% with CE loss. Clearly, the Focal loss made the model predict the low-density classes more accurately, which was the purpose behind the implementation of the Focal loss function. The same behaviour can be seen with the class 'Reflection' that also had a higher validation accuracy with Focal loss.

In Figure 4.2, three images of the same frame could be seen, predictions with each of the two models and the ground truth. When comparing the predictions to the ground truth, the models occasionally had challenges when objects were behind other objects. From Figure 4.2b and 4.2c this can be seen at the boat dock behind the water and the own boat. The labels got mixed up and the boat dock was predicted as 'Boat dock', 'Own boat' and 'Water'. Since the clouds were projected to 2D images, objects that were behind other objects were projected at the same

place in the image. If a point was placed on the exact same pixel as another, the point that was closest to the centre of the boat was chosen. However, as seen in the 2D projections, objects are often a bit 'see-through' and other objects will be seen in-between gaps, which was where the LiDARs did not detect any points. In the prediction this causes the model to mix the classes when objects are behind other objects. A technique to avoid this might be to have a loss function that punishes the model for this behaviour or weigh the depth parameter higher in training.

When comparing the loss functions in the segmented predictions there is not a big difference. However, it can be noted that Focal loss predicted more different classes behind the own boat that resembled the ground truth.

The test accuracies, shown in Table 4.2, have a slightly different result compared to the validation accuracies for the different classes. Classes with higher densities have high accuracy, as in the validation accuracy. However, for some classes that were harder to predict, e.g. 'Pole' for the 'Still' case, the CE loss had a higher accuracy than Focal loss. Overall it can be noted that it is mostly the class density, both for the training data and the test data, that makes a big difference in the accuracy. For example, the class 'Buoy' appears more in 'Moving' than it does in 'Still', and it can be noted that the accuracy was around 19.5% for 'Moving' and around 2% for 'Still'.

5.1.2 Medium set

As with the large case of classes, the last validation accuracies for both of the models were around the same value, as shown in Table 4.3. The accuracies were high and the loss values low, which indicates a high performance on the model. In Figure 4.3 the accuracy graphs for the classes showed that both models perform similarly as with the large set for certain classes. 'Own boat', 'Water', 'Boat' and 'Boat dock' have around the same accuracies as they did in the previous case. The 'Ashore' class performed with an accuracy of around 60% for both loss functions, making it easier to predict than some of the low-density classes, such as 'Road'. However, it has roughly the same accuracy as the class 'Building' which is one of the classes that were a part of 'Ashore'.

The Focal loss still performed better with the classes that were harder to predict, which could be seen for the class 'Buoy' that has a validation accuracy of around 25% for CE loss and 36% for Focal loss. However, other than 'Reflection', which was slightly better with Focal loss, there was not much difference in terms of the class accuracies.

For the medium set, there was still a slight problem of objects being behind other objects. It can be seen in Figure 4.4, which shows the ground truth and the predicted point cloud of each model. The models did not manage to predict correctly behind the boat, which was a problem with the large set. The classes present behind the boat were predicted but not in the correct places at all times. Something that

could depend on the number of classes and how close they are to each other, making the 2D projections also have objects remarkably close to or on each other.

The test accuracies were very high for the medium set, as shown in Table 4.4. It was only for the classes 'Buoy' and 'Reflection' that the percentage was under 60%. Focal loss was better at predicting 'Buoy' than CE loss when the class had more features, as it did in the 'Moving' recording. Otherwise, the different loss functions do not make much of a difference, the accuracies were usually similar.

5.1.3 Small set

For the small set, the models reached a better validation accuracy and loss, as shown in Table 4.5, compared to the two other sets of classes. This could be because the model was particularly good at predicting 'Water' and 'Own boat'. Making the third class, 'Other', easy to predict as well. However, there was not much difference between the loss functions other than that the model with Focal loss had a slightly lower loss.

In the graphs with accuracy per class, Figure 4.5, the 'Water' and the 'Own boat' still have high accuracy between 95% and 100%. 'Other' has remarkably high accuracy as well, that converged around 94% for both CE and Focal loss, which made the difference between the loss functions unnoticeable. Since the class imbalance was smaller than in the previous cases, the changes made to CE loss to create Focal loss made less difference in the training.

In the segmented point clouds, Figure 4.6, the potential usage of segmentation with three classes can be seen. For example, if the classes 'Water' and 'Own boat' were filtered out the rest of the point cloud could be used to determine areas that the vessel should avoid. However, for functions such as docking help it might be beneficial if the class 'Boat dock' would be segmented.

The test accuracies in Table 4.6 clearly show how well the model performs, with all of the accuracies being over 96%. 'Other' often has slightly lower accuracy, which could be because of the small class imbalance that still exists or how many different shapes the class consist of compared to the other two classes. The change in loss function did not make much difference on the test data set either, CE loss performed slightly better than Focal.

5.2 Input data

A segmentation model with high performance was possible to reach with a system solely based on input data from LiDARs. However, it is not perfect since colours and detailed shapes were hard to capture. For instance, it is impossible to see the difference between red and green buoys, which are used as navigation marks. Cameras could, in bright weather conditions, capture the difference in images. The same applies to detailed shapes that would either need a LiDAR with higher resolution or a camera to be captured.

Other aspects to take into consideration when analysing the input data are the diversity and imbalance in the data set as well as the resolution of the recordings. Furthermore, it is possible that errors might have occurred during annotation. Considering how difficult some shapes were to classify it is likely that points were wrongly classified in the ground truth. However, even if this happened the results still showed that the input data were good enough to use for segmentation. When using CNN it often handles these kinds of deviations which could also be an indication that the input data hold the requirements needed for the task.

5.2.1 Diversity of data

The data used were collected from two different harbours and on open water, which might have caused the data to not be as diverse as would have been preferred. The model was tested on data recordings that it had not been previously trained on at all. However, it had still been trained on other recordings at the same locations, only from different angles and positions. Therefore, it is unclear how the model would be able to handle predictions in completely new locations or if the model would need to have training data from more harbours to better predict different shapes of the classes.

Furthermore, there is an issue with an imbalance in the data set. To avoid the issue some of the classes were merged into different constellations, as previously mentioned. This worked very well but if certain classes are needed in the prediction it is not the most desirable approach. Instead, more data on these particular classes would need to be collected or fabricated to train better on them. However, by collecting more data other large classes, like 'Own boat' and 'Water', would also be collected, which would not solve the imbalance issue. A solution could be to specify exactly what the segmentation should be used for and then create classes for that specific case. It should help in keeping the number of classes as low as possible while still having the important classes present. Another approach might be to remove the points that belong to the own boat prior to training and segmentation. Thus, 'Water' would be the class with the largest density and the imbalance would be less noticeable. The removal of these points could be performed with a distance filter since the points often are at a certain distance from the LiDARs.

The marine environment and weather issues are one of the bigger problems with segmentation and one of the reasons why cameras are not used. The LiDARs manage to record the surroundings very well, even through heavy rain. However, rain was one of the classes that the model had difficulties predicting. It could be because of the few points detected and the low amount of frames with rain in the data set. There are a couple of recordings with rain that the model was trained on but without success. As could be seen in Figure 4.7, the rain was labelled as 'Own boat' in the segmentation. The reason for this could be the proximity to the points that belong to the 'Own boat' and the significant difference in the number of points between 'Own boat' and 'Rain'. If a larger percentage of the data set would be of recordings with rain the model might perform better with the 'Rain' class. Additionally, it might be beneficial to add points that imitate rain and by that have more of the 'Rain' class without adding to the other classes. However, even if the models did not manage to predict rain they managed to correctly predict other features in the frames with rain.

5.2.2 Resolution

In the set-up for this thesis, five LiDARs were used to capture the surroundings around the boat. These mechanical LiDARs have a low resolution which made the price lower and therefore enabled the possibility to have several of them. From the results, it is possible to say that the low-resolution LiDARs provide sufficient resolution to produce accurate results that can be used in real-life scenarios. This is partly because the total point cloud is the sum of all points from each of the five LiDARs. If only one of these sensors would have been present the result would not perform as accurate since the measure points would decrease considerably. However, five low-resolution LiDARs gave a better view of the surroundings near the boat than what one high-resolution LiDAR would have if it would be placed on the roof.

When segmenting only three classes the model gave high accuracy and performed on a high level with the LiDARs used. The accuracy for each class for both CE loss and Focal loss reaches above 94%, as could be seen in Figure 4.5. For more classes used with the model, the resolution was high enough to provide high performance. Why the accuracy is lower for some classes probably depends more on the fact that these classes contained fewer points than the classes with higher accuracy. LiDARs with higher resolution could provide more data points on the objects that were located further away, but are probably not worth the higher price and computational power requirements (due to higher memory demand).

5.3 Real-time segmentation

The purpose of the segmentation is to perform it in real-time scenarios on a vessel and therefore it is of considerable importance how fast the segmentation is. Table 4.7 in Section 4.4 shows the averages of the segmentation and prediction times depending on the number of classes. The segmentation times are around 0.3 seconds and do not differ much depending on the cases. The different amount of classes

only make a difference towards the end of the model where the number of channels corresponds to the number of classes, which is why it does not affect the prediction time noticeably. As mentioned in Section 4.4 the largest part of the segmentation time is the prediction time. Therefore, it is mainly the model that would need to be sped up to achieve a faster segmentation time. Furthermore, since the evaluation was run on a computer with a GPU and a large RAM memory the segmentation would most likely be even slower on a smaller computer, which would be the case for real-time segmentation. There are a few different methods to make the segmentation faster, it could be to migrate the model to C++ since it is known for being faster than Python. There are also a few changes that can be made to the network structure to make the predictions faster.

The LiDARs record a frame with an update rate of 10 Hz, as mentioned in Section 3.1. In other words, every 0.1 seconds there is a new frame to be segmented and even with the fast computer, the segmentation was not fast enough. This means that the segmentation would be delayed, or frames need to be skipped, which in turn means that information would be lost. Furthermore, the times presented do not include the plotting of the segmented 3D cloud, which will also take time if that is a wanted feature.

6
Conclusion

The purpose of this project was to investigate the possibilities to implement and adapt fast semantic segmentation on solely sparse LiDAR data in marine environments. This was done through an approach where 3D clouds were projected to a 2D image that a CNN model was trained on. To evaluate and get a satisfactory performance, two different loss functions and different constellations of the number of classes were tried.

Semantic segmentation can be done in a marine environment, only using data from sparse LiDAR sensors. Even if there were some errors in the predictions, it could be seen in the loss and accuracy, especially for certain classes, that the segmentation was highly accurate. The number of classes affects the segmentation in terms of accuracy and the best result was achieved with only 3 classes. However, if specific classes are wanted for a certain use the accuracy was sufficient for classes such as 'Boat dock' and 'Boat'. For classes that were harder to predict the model with Focal loss worked better when segmenting 14 or 8 classes, but it made no difference for only 3 classes.

Even through heavy rain the LiDAR point clouds can be segmented. However, the model failed at predicting the detected rain drops as 'Rain' and instead set them as 'Own boat'. Something that was not surprising since the rain is mostly detected remarkably close to the sensors and therefore close to the own boat. As for other features in frames with rain, the LiDARs manage to detect the surrounding features and the model to predict them with sufficient accuracy.

6.1 Future work

At the moment segmentation is too slow as it is significantly slower than the update rate of the LiDARs. Since it is done in Python a solution could be to migrate to C++ and otherwise the network would need to be changed to make faster predictions.

To make the segmentation more generic a good start would be to collect more data. The data would need to be from other locations than where the current data was collected since more diversity of the features is wanted. Furthermore, it would be interesting to focus the training more on weather conditions and how they affect the segmentation. Something that could be done by recording and labelling more of the conditions, as well as creating fake data that imitates them.

Bibliography

[1] Transportstyrelsen, "Säkerhetsöversikt Sjöfart 2020", Tech. Rep., 2021.

[2] J. Lindgren and F. Odehnal, "Segmented Classification of Traffic Environments Using RGB-D Data", Department of mechanics and maritime sciences, Gothenburg, Tech. Rep., 2020.

[3] N. Wang, Y. Wang, and M. J. Er, "Review on deep learning techniques for marine object recognition: Architectures and algorithms", *Control Engineering Practice*, vol. 118, p. 104 458, Jan. 2022, ISSN: 09670661. DOI: 10.1016/j.conengprac.2020.104458.

[4] D. J. Yeong, G. Velasco-Hernandez, J. Barry, and J. Walsh, "Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review", *Sensors*, vol. 21, no. 6, p. 2140, Mar. 2021, ISSN: 1424-8220. DOI: 10.3390/s21062140.

[5] Frost & Sullivan, "LiDAR: Driving the Future of Autonomous Navigation", Tech. Rep., 2016.

[6] *The Automotive LiDAR Market*, 2018. [Online]. Available: http://www.woodsidecap.com/wp-content/uploads/2018/04/Yole_WCP-LiDAR-Report_April-2018-FINAL.pdf.

[7] Robosense, *LiDAR*. [Online]. Available: https://www.robosense.ai/en/product.

[8] Anil Chandra Naidu Matcha, *A 2021 guide to Semantic Segmentation*, 2021. [Online]. Available: https://nanonets.com/blog/semantic-image-segmentation-2020/.

[9] X. Xie, L. Bai, and X. Huang, "Real-Time LiDAR Point Cloud Semantic Segmentation for Autonomous Driving", *Electronics*, vol. 11, no. 1, p. 11, Dec. 2021, ISSN: 2079-9292. DOI: 10.3390/electronics11010011.

[10] M. Ibrahim, N. Akhtar, K. Ullah, and A. Mian, "Exploiting Structured CNNs for Semantic Segmentation of Unstructured Point Clouds from LiDAR Sensor", *Remote Sensing*, vol. 13, no. 18, p. 3621, Sep. 2021, ISSN: 2072-4292. DOI: 10.3390/rs13183621.

[11] Bert Carremans, "Handling overfitting in deep learning models", *Towards Data Science*, Aug. 2018. [Online]. Available: https://towardsdatascience.com/handling-overfitting-in-deep-learning-models-c760ee047c6e.

[12] C. Xu, B. Wu, Z. Wang, *et al.*, "SqueezeSegV3: Spatially-Adaptive Convolution for Efficient Point-Cloud Segmentation", Apr. 2020.

[13] B. Yang, W. Luo, and R. Urtasun, "PIXOR: Real-time 3D Object Detection from Point Clouds", Feb. 2019.

[14] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation", May 2015. DOI: 10.48550/arxiv.1505.04597.

[15] M. Grandini, E. Bagli, and G. Visani, "Metrics for Multi-Class Classification: an Overview", Aug. 2020.

[16] Inara Koppert-Anisimova, "Cross-Entropy Loss in ML", *unpackAI*, 2021. [Online]. Available: https://medium.com/unpackai/cross-entropy-loss-in-ml-d9f22fc11fe0.

[17] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection", Aug. 2017.

[18] J. Zhang, Y. Yao, and B. Deng, "Fast and Robust Iterative Closest Point", Jul. 2020. DOI: 10.1109/TPAMI.2021.3054619. [Online]. Available: https://arxiv.org/abs/2007.07627.

[19] D. Paloschi, S. Korganbayev, K. Bronnikov, A. Wolf, A. Dostovalov, and P. Saccomandi, "Transformation matrices for 3D shape sensing with polyimide-coated multicore optical fiber", in *2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT*, IEEE, Jun. 2020, pp. 250–254, ISBN: 978-1-7281-4892-2. DOI: 10.1109/MetroInd4.0IoT48571.2020.9138221.

[20] J. Behley, M. Garbade, A. Milioto, *et al.*, "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences", Apr. 2019.

[21] Anirudh Topiwala, "Spherical Projection for Point Clouds", *Towards data science*, 2020.

[22] NVIDIA, *NVIDIA QUADRO RTX 4000*, Jun. 2019. [Online]. Available: https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/quadro-product-literature/quadro-rtx-4000-datasheet.pdf.

[23] D. Giordano, "7 tips to choose the best optimizer", *Towards Data Science*, Jul. 2020. [Online]. Available: https://towardsdatascience.com/7-tips-to-choose-the-best-optimizer-47bb9c1219e.

A

Appendix

The translation matrix for each translation between each LiDAR and to the base can be seen below. Where T_{L2}^{L1} are the transformation matrix from L2 to L1.

$$T_{L2}^{L1} = \begin{bmatrix} -0.5041 & -0.0227 & 0.8633 & -0.6712 \\ 0.8349 & 0.2427 & 0.4940 & 2.2907 \\ -0.2207 & 0.9698 & -0.1034 & -10.9597 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{L3}^{L1} = \begin{bmatrix} -0.5712 & 0.0612 & 0.8185 & -0.6143 \\ -0.7851 & 0.2503 & -0.5666 & -1.4116 \\ -0.2395 & -0.9662 & -0.0950 & -10.7367 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{L4}^{L1} = \begin{bmatrix} 0.2300 & -0.0689 & -0.9707 & -1.9929 \\ -0.9729 & 0.0067 & -0.2310 & -0.9582 \\ 0.0224 & 0.9976 & -0.0655 & -3.9522 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{L5}^{L1} = \begin{bmatrix} 0.1802 & 0.0313 & -0.9831 & -1.9920 \\ 0.9836 & -0.0062 & 0.1801 & 1.8895 \\ -0.0005 & -0.9995 & -0.0319 & -4.0052 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{L1}^{Base} = \begin{bmatrix} 0.0008 & -0.0002 & -1.0000 & -7.1700 \\ -0.0010 & -1.0000 & 0.0002 & 0.4800 \\ -1.0000 & 0.0010 & -0.0008 & 0.3960 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

B

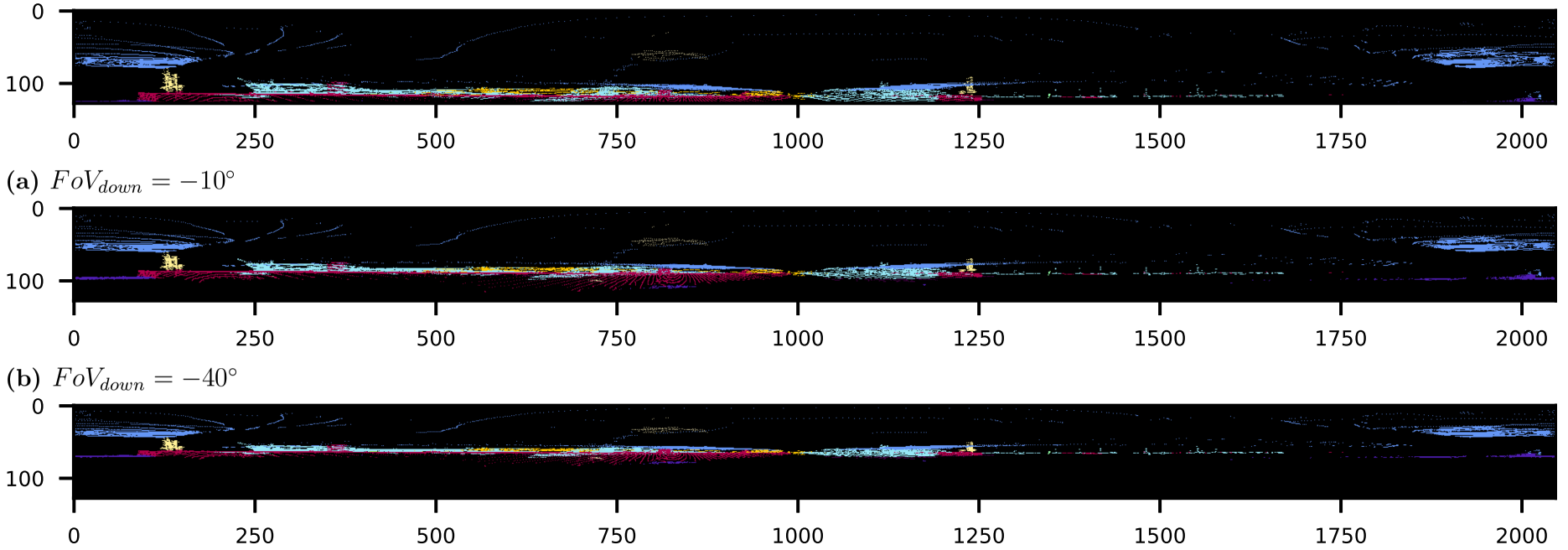
Appendix

Figure B.1: 2D spherical projection image of labeled 3D point cloud

(a) $FoV_{up} = 90°$

(b) $FoV_{up} = 30°$

Figure B.2: Two 2D images with different values of FoV_{up}. FoV_{down} is on both images set to $-40°$



(a) $FoV_{down} = -10°$

(b) $FoV_{down} = -40°$

(c) $FoV_{down} = -90°$

Figure B.3: Three 2D images with different values of FoV_{down}. FoV_{up} is on both images set to 90°

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS