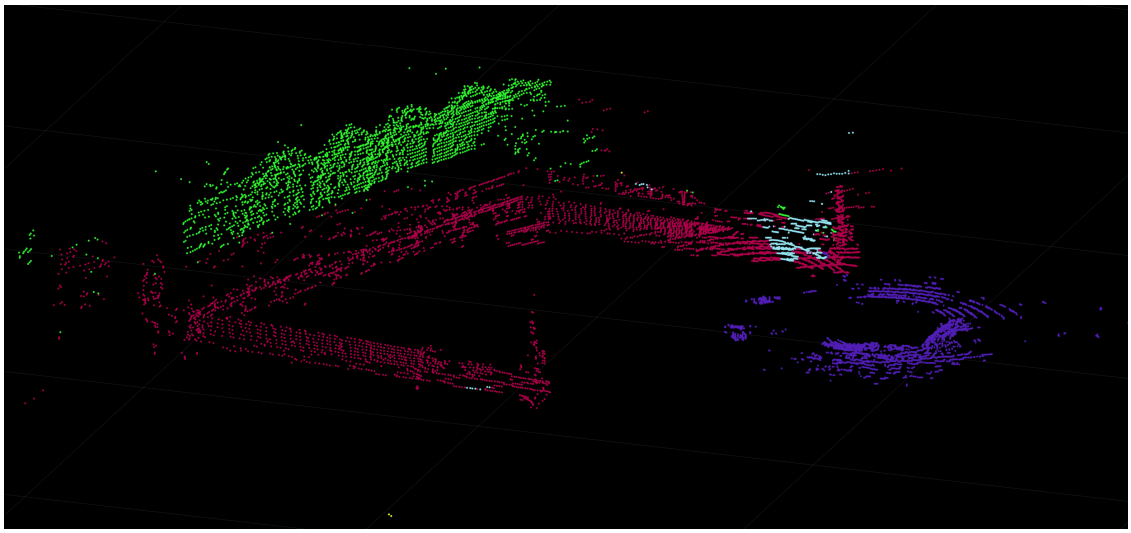
CHALMERS
UNIVERSITY OF TECHNOLOGY



LiDAR-Based Semantic Segmentation for Marine Surroundings

Optimization strategies for segmentation classification in a marine environment

Master's thesis in Systems, Control and Mechatronics

HANNA JONSSON
DANIEL SCHOLTZ

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

LiDAR-Based Semantic Segmentation for Marine Surroundings

Optimization strategies for segmentation classification in a marine environment

Hanna Jonsson
Daniel Scholtz

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
Gothenburg, Sweden 2023

LiDAR-Based Semantic Segmentation for Marine Surroundings
Optimization strategies for segmentation classification in a marine environment

Hanna Jonsson
Daniel Scholtz

Supervisors: Jonatan Bergenwall & Hanna Jonsson, CPAC Systems AB
Examiner: Peter Forsberg, Department of Mechanics and Maritime Sciences

Master's Thesis 2023
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: Predicted LiDAR point cloud in a fishing boat harbour

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2023

LiDAR-Based Semantic Segmentation for Marine Surroundings
Optimization strategies for segmentation classification in a marine environment
Hanna Jonsson
Daniel Scholtz
Department of Mechanics and Maritime Sciences
Chalmers University of Technology

Abstract

In this thesis, methods for optimize an existing Convolutional Neural Network model for semantic segmentation are proposed. This is done through examining the size of the network, loss functions, dataset and how it can be preprocessed in different ways. The investigation show that preprocessing the data do not improve the model and that cross entropy loss is the best loss function when the dataset is highly imbalanced. The results from this project together with suggestions for future work shows bright results for future implementation.

Keywords: Semantic Segmentation, LiDAR, CNN, U-net, Optimization

Acknowledgements

We would like to express our thanks to CPAC Systems AB for providing us with the opportunity to conduct our master's thesis research within their organization. A special thanks to Jonatan Bergenwall and Hanna Jonsson, for their continuous support, guidance, and encouragement. It would not have been possible to compleat this project without their mentorship and willingness to share their expertise. Furthermore, we would like to express our deepest appreciation to our examiner, Peter Forsberg, for his guidance and support. Finally, we are grateful to the entire staff at CPAC Systems AB for creating a conducive and inspiring work environment.

Hanna Jonsson & Daniel Scholtz, Gothenburg, June 2023

Thesis advisors: Jonatan Bergenwall & Hanna Jonsson, 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:

ADAS	Advanced Driver Assistance
AI	Artificial Intelligence
CDF	Cumulative distribution function
CE	Cross Entropy
CMYK	Cyan-Magenta-Yellow-Key
CNN	Convolutional Neural Network
FoV	Field of view
HSL/HSV	Hue-Saturation-Lightness/Value
LiDAR	Light Detection and Ranging
ML	Machine Learning
NIST	National Institute of Standards and Technology
RGB	Red, green, blue
WCE	Weighted Cross Entropy

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

R_{ref}	Reference range
N	Number of classes
γ	Hyperparameter for Focal loss

Variables

α	Angle of incidence
(x, y, z)	Cartesian coordinates in point cloud
I	Intensity value in point cloud
I_c	Corrected intensity value
R	Range detected from the LiDAR
T	Atmospheric transmission
y_i	Ground truth in loss function
$\hat{y}_i$	Prediction in loss function

Contents

List of Figures

List of Tables

1
Introduction

In this section, the background, purpose, and limitations of this master thesis work are described.

1.1 Background

Every year, autonomous vehicles are becoming increasingly popular. On road vehicles lead the way in the autonomous sector, using software technologies such as object detection, neural networks, artificial intelligence (AI), and hardware technologies including cameras and Light Detection and Ranging (LiDAR) [1][2]. However, while the on road sector pushes forward, the marine sector falls behind. The challenges and goals of autonomous vehicles are the same in the marine environment but with different requirements, such as slower speeds, more space, and fewer objects in motion.

One step in developing autonomous systems for vessels is the implementation of accurate surrounding view systems. Today, semantic segmentation is a commonly used method for self driving cars to get an overview of their surroundings by classifying each pixel in an image as a predefined class [3]. Convolutional Neural Network (CNN) based methods have been proven to work successfully when applied in semantic segmentation [4]. The evolution of CNNs accelerates in the past ten years, leading to improvements in the complexity and reliability of the methods for designing and enhancing the performance of CNN models [5]. Even though the complexity of CNNs increases over the years, the best way to improve the accuracy of a model is with a large amount of training data. Other ways to improve the CNN's ability to accurately make predictions are data preprocessing, using advanced loss functions, and adjusting the network structure [6].

1.1.1 Previous work

This master's thesis continues a previous work on semantic segmentation in the marine environment. The purpose of this previous work was to "implement and adapt fast semantic segmentation on solely sparse LiDAR data in marine environments" [7]. Previous work involved a pipeline with five main steps: recording data from LiDAR sensors on a boat, preprocessing the data, manually annotating using a point cloud labeler tool originally developed for the SemanticKITTI dataset (a car collected point cloud dataset) [8], projection of 3D point clouds onto 2D images,

creation of a CNN-compatible dataset, training of the CNN, and prediction and visualization of results. The data from the LiDARs are recorded in a format called rosbag. In the preprocessing step, the rosbags are unpacked using MATLAB. The CNN model is implemented using PyTorch lightning which has the advantage of using a prewritten training loop once a model is created. The model used is called U-Net. Finally, the results are visualized using MATLAB.

Key takeaways from the project include the viability of using solely LiDAR sensors for semantic segmentation. The CNN model achieves an accuracy between 96% to 99% for the three classes; 'Own boat', 'Water', and 'Other'. However, the accuracy quickly reduces when more classes are added. Additionally, the network is too slow for real time usage. Suggestions for future work include modifying the network to improve prediction speed for real time performance, developing suitable methods for handling imbalanced datasets, investigating how different weather conditions impact the results, and determining the optimal loss function to use.

1.2 Purpose

The purpose of this master's thesis is to improve the current data processing chain (hereafter called pipeline) and optimize the CNN model [7] for fast semantic segmentation on LiDAR data in marine environments. A part of the purpose is also to be able to run the model in real time on an embedded device. To achieve this, this thesis will focus on making the model more accurate, and faster, and also explore which machine learning (ML) software that is most suitable for embedded devices.

1.2.1 Objectives

To reach this master's thesis purpose the following research questions will be investigated:

- Which learning strategy, loss function, and data preprocessing steps are most suitable for semantic segmentation in the marine environment?
- Is it possible to use low resolution data with high resolution data, or do they need to be trained separately?
- How does the accuracy of the model change when different data resolutions are used or when a combination of resolutions is employed?
- What kind of data must be collected in order to perform highly accurate and precise semantic segmentation in the marine environment?
- Which classes are necessary for semantic segmentation in the marine environment?
- How could weather drawbacks such as rain and snow be handled?
- What are the requirements and key components for a complex algorithm to be able to run on an embedded device platform?

1.3 Limitations

In this master thesis, several limitations must be acknowledged. Previous work [9] on using artificial data for training has indicated that it does not improve real world performance, therefore it will not be utilized. In the previous year's thesis work [7], experiments were conducted by training the network on three different variations of the same dataset: a large set, a medium set, and a small set. The large set consisted of 15 different classes, the medium set had eight classes, and the small set had three classes. The small and medium sets showed promising results; however, the large set with numerous classes posed challenges. Due to limited data availability for many of these classes, the network was unable to accurately detect them. Therefore, a reduced number of classes will be used, with a focus on classes related to water. All land objects will be treated as a single class.

Deep learning has been highly effective in solving 2D vision problems, including image recognition, object detection, and image segmentation. Only recently deep learning has moved to 3D point cloud segmentation. Therefore, early adopters are using deep learning techniques suitable for 2D models on 3D problems. A LiDAR point cloud contains 4 parameters per point: x, y, z, and intensity, whereas an image normally consists of three parameters: red, green, and blue (RGB). Typically, a modified version of a CNN is used to extract information from an image, capturing semantic information through techniques such as downsampling, pooling, bridging, and others. However, this deep learning theory does not necessarily translate to the 3D case, simply because the input data is very different[10]. To address this challenge, researchers have taken two approaches: modifying the input data to resemble the image format or deploying new deep learning techniques specifically designed for 3D data. The latter approach is more complex, and network structures like RandLA-Net have emerged, demonstrating efficiency and speed comparable to the projection methods used in earlier models. However, due to the complexities associated with these new methods, they will not be used in this thesis. Instead, the focus will be on utilizing the 2D projection method and optimizing the network structure and existing methods to improve performance.

The number of data points available for training is one of many factors that significantly impact the accuracy of the network. Common datasets in the domain of LiDAR segmentation vary in size but some of the popular ones are *Semantic3D*, *Oakland*, *iQmulus* and *Paris- Lille-3D* where each respective set contain 4 billion, 1.6 million, 300 million and 143.1 million data points, respectively [11]. However, the dataset used in this thesis consists of fewer than 100 thousand data points, which is significantly smaller. This limited dataset size is likely to have a negative impact on the network's ability to learn and adapt to new, unseen data. The decision to use a smaller dataset was influenced by the location of the testing boat, which was situated far away from other harbors, as well as the constraints on recording new data due to time limitations. While this decision was made, it is important to note that the lack of diversity in the dataset could present a problem. Since most of the recordings were obtained from similar environments, the model may be trained on

homogeneous data, which can result in inaccurate predictions when applied to new, unseen data.

2
Theory

The theory for the techniques used in this thesis work is described in this chapter.

2.1 Semantic segmentation

Semantic Segmentation is a computer vision technique employed to assign a predefined class label to each pixel in an image. Its purpose is to identify and differentiate different objects or regions within the image accurately. In the marine environment, semantic segmentation algorithms can be utilized to distinguish floating objects, noise, and land from images. This approach enables a more accurate and detailed analysis of the images [12].

Feature based segmentation methods aim to classify individual pixels separately, often assuming that each pixel is independent, not taking into account contextual relations between data points [4]. To encode spatial relationships between data points, graphical models such as Markov random fields and conditional random fields can be used. These methods improve the computational speed of the classifier because each pixel only depends on its subsequent neighbor.

2.2 LiDAR

LiDAR is an optical technology built on a light from a laser that is sent from a transmitter and then reflected on objects to measure a distance. The distance is measured by measuring the time lapse between the transmitter sending a light pulse until the detection of the reflected light pulse coming back [13]. LiDAR systems can also be used to provide 3D models as point clouds with all the recorded points represented with the parameters of position and intensity of the returned light (x, y, z, I). According to [13], LiDAR is a key method for autonomous vehicles and Advanced Driver Assistance (ADAS).

There are many different kinds of LiDARs on the market, with new models constantly being developed. Today's LiDARs largely have either a wavelength of 905 nanometers or 1550 nanometers [14]. The different wavelengths in LiDAR sensors affect the performance and characteristics of the system. Generally, a longer wavelength will penetrate atmospheric conditions better, allowing for longer-range detection. The wavelength also affects the spatial resolution, shorter wavelengths can provide higher resolution and finer detail in the resulting 3D point cloud but it

is also more respectable to noise.

The different LiDARs can be divided into two groups, mechanical LiDAR, also known as spinning LiDAR, and solid-state LiDAR. The mechanical LiDAR uses a prism or a spinning mirror to collect data over a horizontal field of view (FoV) up to 360 degrees and a vertical FoV up to 90 degrees [15]. Solid-state LiDARs, on the other hand, use electronic components to direct the laser beam and collect data. The solid state LiDARs can be divided into three types: flash-based LiDAR, a microelectromechanical system based LiDAR, and optical phased array based LiDAR [16]. The horizontal FoV of these LiDARs can range from around 20 degrees to 120 degrees, while the maximum vertical FoV for a solid-state LiDAR is 90 degrees.

For this master thesis, six LiDARs are used; three mechanical RS-Bpearl, two mechanical RS-Helios, and one solid-state RS-LiDAR-M1. The RS-Bpearl has a horizontal FoV of 360 degrees and a vertical FoV of 90 degrees. The measurement range is 30m@10% NIST [17]. The RS-Helios has a horizontal FoV of 360 degrees, and a vertical FoV from +15 degrees to -55 degrees (70 degrees in total). The measurement range is 70m@10% NIST[18]. The solid-state RS-LiDAR-M1 has a horizontal FoV of 120 degrees and a vertical of 25 degrees. It has a measurement range of 150m@10% NIST[19].

2.3 Preprocessing of data

Data preprocessing is a crucial step in ML that involves preparing and transforming raw data into a format that is suitable for analysis [20]. This step is important since it can greatly affect the performance of a ML model. By preprocessing the data, noise in the data can be removed, and make the patterns more apparent. This can lead to better model performance, improved accuracy, and faster convergence. Additionally, data preprocessing can help to reduce overfitting and increase the generalization of the model to new data.

2.3.1 Intensity measurement influencers in LiDAR data

To make accurate predictions on LiDAR data, it is important to understand its behavior and how to preprocess it efficiently. There are four main categories that influence intensity measurements: surface characteristics, data acquisition geometry, instrumental effects, and environmental factors [21].

Surface characteristics, refer to how the laser reflects off different materials with varying intensity values. For example, a car may have a higher intensity value than a road because it is made of aluminum compared to concrete or tarmac. The second category, data acquisition geometry, refers to the effect of the position, field of view, and orientation of the LiDAR scanner. Two factors will have an impact on the reflectivity value; range and angle of incidence. Since the laser from the emitter is traveling through a medium (air), the intensity of the beam will degrade over distance because the strength of the pulse diminishes. The angle of incidence is

the angle between the laser and the surface plane the laser is hitting. If the plane is perpendicular to the laser the maximum intensity of a given material and range will be reflected. If the angle is greater than 90 degrees the intensity values will degrade. Therefore, objects that are curved will have an intensity gradient across their surface, making detection more complex with tools such as object detection algorithms. The third category is instrumental effects, which have to do with how the manufacturer builds their product. Different instruments have different instrument-specific parameters, which in turn affect intensity. As a result, different LiDARs might not have exactly the same intensity values for the same object. The fourth and last category is the environmental effect. When a laser is traveling through the air, it hits small particles which causes laser scattering, effectively lowering the intensity value. Such particles are dust, aerosol scattering, and air clusters. Water molecules in the air absorb light energy and will contribute to energy loss in the laser. Consequently, highly humid areas cause the intensity to degrade faster over distance.

To prevent the above mentioned intensity influencers a variety of techniques can be used. Either by using theoretical correction methods or empirical correction methods where data is analyzed and correction models are defined[21].

The most common theoretical correction models compensate for degeneration in intensity over range and angle of incidence. One such model can be seen in equation 2.1, where R is the range detected from the LiDAR and R_{ref} is a user defined reference range designed to normalize the equation. α is the angle of incidence that has to be known or estimated with methods such as plane fitting.

$$I_c = I \frac{R^2}{R_{ref}^2} \frac{1}{cos(\alpha)} \tag{2.1}$$

The advantage of using this equation is that it is independent of material.

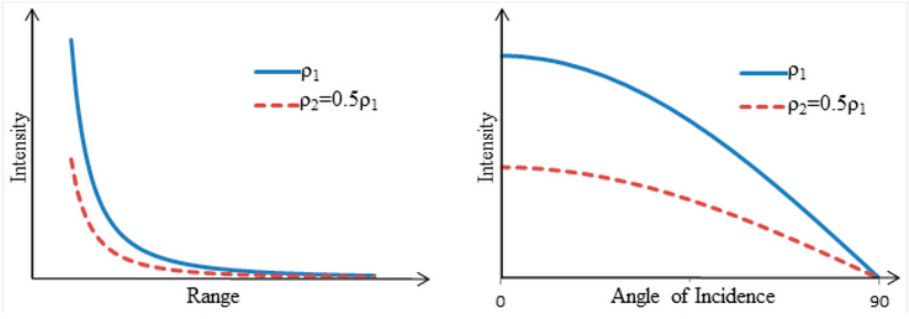


Figure 2.1: Relationship of intensity with respect to range (left) and angle of incidence (right) [21].

Limitation for using theoretical models is that objects which have a smaller surface area than the laser beam, for instance, wires will return with higher than normal intensity. Therefore, equation 2.1 is suitable when applying intensity correction on larger objects. Some LiDAR hardware is fitted with an intensity reduction filter on

objects which are very close to the LiDAR (10m usually) which also has to be taken into consideration when using intensity correction models [21].

Empirical methods estimate correction parameters and models by analyzing several frames where the same object appears but at varying distances. The intensity values for the object are observed and correction parameters or models can be derived [21].

Models for estimating the effect of atmospheric scattering have also been investigated [22]. One method to correct the intensity is to use MODTRAN, which is a method to estimate the atmospheric transmission T. Then the corrected intensity is $I_c = I\frac{1}{T^2}$.

2.3.2 Normalization

When working with 3D data, normalization is usually a preprocessing step worth considering. A primary benefit of normalization is the use of a common boundary to enclose all features, which enables the preservation of information while ensuring consistency across all features. When the feature values are within a boundary range, the gradient descent flow is smoother, and it helps the descent method avoid biases towards outliers in the data [23]. Several approaches exist to normalize a 3D point cloud, with scaling being a popular option that constrains all values within the range of negative one to one. Scaling is achieved by dividing all points in the point cloud with a selected maximum value.

2.3.3 Histogram equalization

Histogram equalization is a widely used technique in digital image processing for enhancing the contrast of images [24]. The objective of this technique is to increase the global contrast of an image by redistributing the intensities in a way that the full range of intensities is used. The process involves computing a histogram of the input image and then manipulating the image's pixel values to produce an output image that has a more uniform histogram.

The basic idea behind histogram equalization is to transform the pixel values such that the cumulative distribution function (CDF) of the image is made as linear as possible. This is achieved by computing the CDF of the input image, which gives the probability of each intensity value occurring in the image. After, the pixel values are transformed by mapping each input intensity value to its corresponding output intensity value using the normalized CDF, which can be seen in Figure 2.2.

The result of histogram equalization is an image that has a more uniform distribution of pixel intensities, and consequently, higher contrast. The method is particularly useful when the pixel intensity values are predominantly concentrated in a narrow range. By spreading out the intensities to cover the entire range of values, usually 0 to 255, histogram equalization can produce images with more details and be easier to analyze.

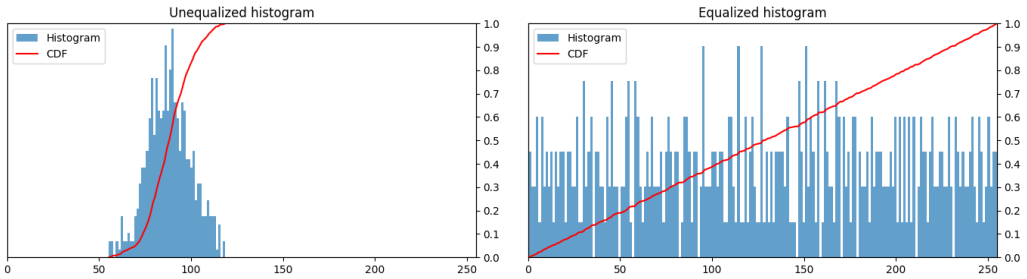


Figure 2.2: Comparison of histogram equalization where the left images display an unequalized histogram and the right image displays an equalized histogram.

2.3.4 Image resolution

The resolution of images used for semantic segmentation can have a significant impact on the performance of the segmentation model since segmentation involves assigning a label to each pixel in an image. Therefore, it is important that the image contains fine details and boundaries of objects within the image. If the image resolution is too low, important details may be lost, leading to inaccurate segmentation. However, if the image resolution is too high, the model may struggle to process the large amount of information contained within the image. This can lead to slower inference times and increased memory usage, making it difficult to deploy the model on resource-limited devices or in real-time applications [25].

2.3.5 Colour profiles effect on feature extraction

Training on images often requires working with different color models, including Red-Green-Blue (RGB), Cyan-Magenta-Yellow-Key (CMYK), and Hue-Saturation-Lightness/Value (HSL/HSV). However, when working with LiDAR data that has been projected into 2D images, the color model differs from that of a typical image-trained convolutional neural network (CNN), as 2D LiDAR data images usually only have two channels: one for depth and one for intensity. Normalizing the color models is a common practice that can increase the robustness and accuracy of image-based classification models, such as VGG16 and ImageNet [26].

There are different perspectives on what is most important for a model to perform well. Some argue that a model's ability to learn features, such as shapes and forms, is paramount, while others suggest that color plays a significant role in performance. Ultimately, the significance of color in a model's performance depends on the task at hand. For instance, color is more significant when classifying road signs, while predicting environmental objects like trees may require a combination of features and color, and modern cars may require feature-based predictions only [27].

Previous research [26] has examined the impact of color on model accuracy and found that manipulating the color profile in various configurations resulted in a degradation of accuracy. However, some color manipulations had minimal impact, with accuracy falling only a few percentage points below the baseline RGB color gamut. Removing

one channel, such as depth or intensity information, may potentially double the prediction speed, with a small and manageable loss in accuracy.

2.4 CNN architecture for segmentation

One of the most popular CNN architectures to use in semantic segmentation is U-Net, which was originally developed for biomedical image segmentation [28]. The U-Net architecture consists of a contracting path (encoder) and an expanding path (decoder). The contracting path has a series of convolutional layers and max pooling operations that reduce the spatial resolution of the feature maps. This part is a traditional CNN that captures the context (semantic) information of the image. The expanding path contains a series of upsampling layers that upsample the image using transposed convolution which halves the number of feature channels. It is then concatenated with the saved feature maps from the contracting path, i.e the "copy & crop" operation in Figure 2.3, and passed through two convolutional layers with small kernel size to increase the spatial resolution of the feature map. The output layer is a convolutional layer that produces a map with the same number of channels as the number of classes in the problem.

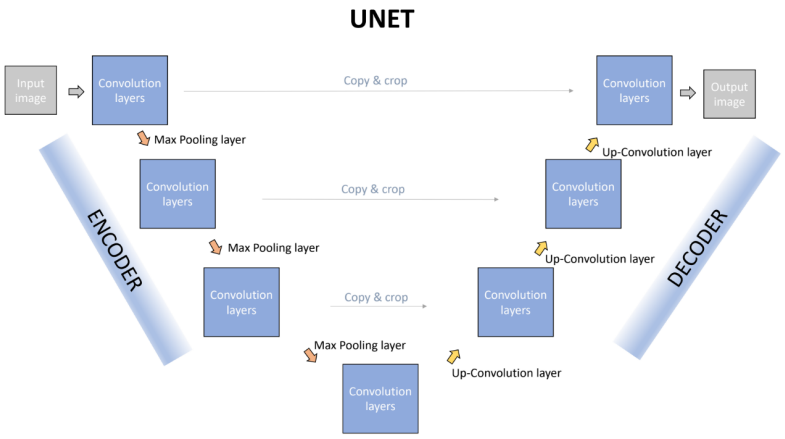


Figure 2.3: CNN model: Overview of U-Net [7].

The number of feature maps that the network produces in each convolutional layer corresponds to the number of channels in the U-Net architecture. Each feature map represents particular features from the images [29]. When the number of channels in the network is increased, the network becomes more complex and is able to learn more abstract and higher-level features from the input images. However, increasing the complexity of the network could result in overfitting if the model learns to fit noise and very specific features in the data instead of the underlying pattern [30]. If the dataset is small or homogeneous, a complex model with many trainable parameters tends to perform worse than smaller models with less trainable parameters, on the other hand, if the dataset is larger and not so homogeneous a more complex network might be needed to receive good results.

The number of trainable parameters also has an impact on the training time and data capacity [31]. A network with more trainable parameters holds more information and therefore usually takes longer to train and demands more data capacity, such as data storage and computational resources. However, it is important to note that the relationship between the number of trainable parameters and the training time and data capacity is not linear. In some cases, networks with more parameters can require less training time or data capacity compared to a smaller network, particularly if the larger network is more efficient or better optimized.

The original U-Net uses convolutional layers with a kernel size 3x3 and since it does not use padding, the produced output is smaller than the provided input, and as a result of this information loss could occur [32]. The reason for this was never clarified from the original errors, but according to Minh Tran, one reason could be that the authors did not want to introduce segmentation errors at the image margin. If U-Net is to be used without information loss one could either run it multiple times on overlapping different parts of the images to get full segmented images in the end, or use padding to the convolutional layers.

2.5 Loss function and imbalanced data

One important part of a CNN is the loss function. Each loss function has its use case and which loss function to choose depends on the attributes of the dataset and the task of the neural network. For example, if the task is to predict an output value corresponding to an input value, then the regression loss function is normally used. If the task is to identify a predetermined set of categories (classes) with a probability, which is the task in this thesis, then the classification loss function is often used [33].

A common challenge with classification problems is dealing with imbalanced data [34]. Imbalanced data refers to a dataset where the distribution of classes is not equal. This means that one or more classes may have many more samples compared to other classes in the set. One of the more effective ways of dealing with an imbalanced dataset is with a loss function which takes the imbalanced frequency of each class into account [33].

The commonly used classification loss functions for image segmentation are cross entropy (CE) loss, weighted cross entropy (WCE) loss, focal loss, and dice loss. There are variations for each of these loss functions, which again depend on the dataset and the specific task of the network [35].

CE loss, (2.2), is the most common loss function when dealing with a multiclass or multilabel classification problem. If the task is to identify one class per pixel, i.e. multiclass classification, a softmax activation function is used before the loss is calculated. If the task is to identify multiple labels (classes) per pixel, then a Sigmoid activation function is applied on each predicted class from the network [36].

$$CE = -\sum_{i=1}^{N} y_i log(\hat{y}_i) \tag{2.2}$$

Equation (2.2) shows the standard multiclass CE loss function where N is the number of classes, y_i is the ground truth, and $\hat{y}_i$ is the prediction of the model. One problem with CE loss is that the loss function can produce bad results when the data is highly imbalanced. This is because the majority classes, such as background data, will be prioritized, which can result in poor performance for the minority classes. A workaround to this problem is to use WCE loss [37] which is shown in equation (2.3).

$$WCE = -\sum_{i=1}^{N} w_i y_i log(\hat{y}_i) \tag{2.3}$$

Here w_i is the weights of the classes. To determine the weights, the most rudimentary approach is to use the inverse proportionality of their respective frequency. That is, the majority class will have a smaller weight and the minority class will have a larger. To magnify the effect of this method, square weights or cubed weights can be used when the dataset is heavily imbalanced [37]. A more nuanced method to solve the imbalanced dataset problem is focal loss. Focal loss is an attempt to change the loss function over the training period by gradually degrading the importance of classes the CNN already is good at predicting.

$$\textit{Binary Focal Loss} = -\alpha_i (1 - \hat{y}_i)^{\gamma} log(\hat{y}_i) \tag{2.4}$$

$$\textit{Focal Loss} = -\sum_{i=1}^{N} (1 - \hat{y}_i)^{\gamma} y_i log(\hat{y}_i) \tag{2.5}$$

Equation (2.4) is the original authors [37] binary focal loss function and equation 2.5 is an extension to a multiclass variant [38]. The important part of equation (2.4) is $\alpha_i(1-\hat{y}_i)^{\gamma}$. Here α_i is the weight of class i similar to w_i in equation 2.3, and the focal factor term is $(1-\hat{y}_i)^{\gamma}$ where γ is a tunable hyperparameter. The reasoning behind this loss function is the following: if the model is good at predicting a specific class the predictions ($\hat{y}_i$) for this class will be closer to one. The focal factor will then be close to zero which, as can be seen in the equation 2.5, will cause the whole loss function to be close to zero for this specific class. Hence, classes in the image that the network already is good at predicting will not influence the loss for the whole image, rather the classes that the network is bad at predicting will. Therefore, this will in theory focus the loss function on hard to predict classes such as the minority classes.

$$\textit{Weighted Focal Loss} = -\sum_{i=1}^{N} \alpha_i (1 - \hat{y}_i)^{\gamma} y_i log(\hat{y}_i) \tag{2.6}$$

2.6 Evaluation

During training a deep machine model can be overfitted. This means that the model knows the training data well but cannot generalize it to other data. Therefore the whole data set is split up into three separate datasets; training, validation, and test set.

When evaluating the performance of a deep machine model, benchmarks for open datasets primarily prioritized accuracy over runtime performance [39]. To evaluate the accuracy several methods can be used. If the training data is highly balanced a fairly good metric is accuracy. The accuracy measure how well the model makes correct predictions. The accuracy is defined as

$$\text{Accuracy} = \frac{\text{Number of correct prediction}}{\text{Total number of samples}} \tag{2.7}$$

However, for imbalanced datasets, the accuracy score is not as good, and therefore other methods are preferred [40]. F1-score, precision, and recall (per-class accuracy) all derive from the confusion matrix. Figure 2.4 shows an example of a confusion matrix. On the x-axis, the prediction from the model is shown. The y-axis shows the true value.

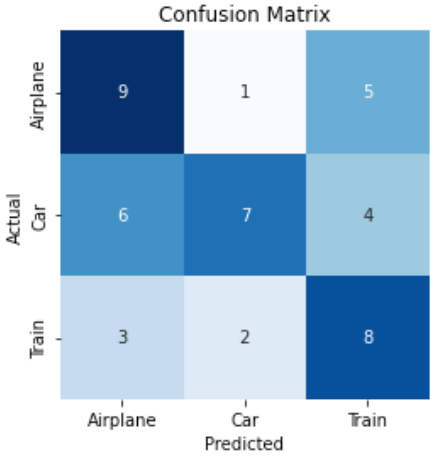


Figure 2.4: Example confusion matrix [41]

Precision and recall can then be extracted from the matrix. The precision metric is interpreted as how many of the predicted positives are actually true positives. In Figure 2.4, the columns represent the precision values for each class. The precision score for airplane would be calculated as $\frac{9}{9+6+3} = 0.5$. The recall score is then represented in the rows of the confusion matrix. Airplanes recall score is then $\frac{9}{9+1+5} = 0.6$. The two metrics have different use cases. For example, in biomedical classification tasks, one would want to optimize for precision rather than recall. The reason is that if the model detects something, it is more important that it classifies it correctly rather than just finding something and then classifying it incorrectly. Whereas, for self driving cars it can be more important that the car does not miss

a detection rather than classifying it correctly. For example, if the car classifies a person as a cyclist rather than not classifying it at all, that is a preferred behavior.

F1-score is the harmonic mean of the precision and recall which can be seen in Equation 2.8. One property of harmonic mean is that it is weighted, therefore if either the precision or recall is low, the whole f1-score suffers greatly[40].

$$F1 = \frac{2 \times recall \times precision}{recall + precision} \tag{2.8}$$

The precision, recall, and f1-score are best evaluated per class to get the best overview of the performance of the deep machine model.

3
Method

This chapter describes the method that was carried out, including collection and annotation of data, 3D to 2D projection, different preprocessing methods of the data, training, testing, and evaluation of semantic segmentation.

3.1 Collection of data

In this work, both previously and newly collected data gathered through the deployment of LiDAR sensors on a boat, were used. The LiDAR sensors provide complete 360° coverage of the boat's surroundings, allowing for comprehensive data collection. As shown in Figure 3.1, the LiDAR sensors are strategically positioned on the boat, with three low-resolution Bpearl LiDARS denoted by red cylinders, two low-resolution Helios LiDARS by orange cylinders, and a high-resolution M1 LiDAR denoted by a blue cylinder.

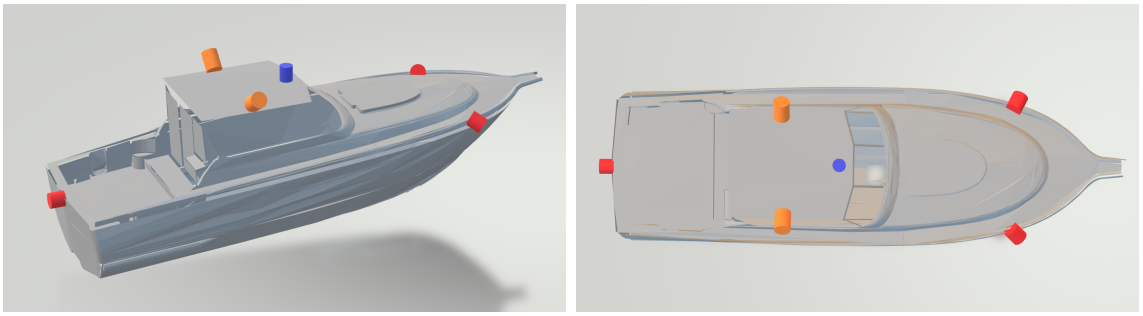


Figure 3.1: Placement of LiDAR sensors on the boat. Low-resolution Bpearl LiDARs are denoted by red cylinders, low-resolution Helios LiDARs by orange cylinders, and a high resolution M1 LiDAR is denoted by a blue cylinder.

The previously collected data were obtained by the five low-resolution LiDARs while the newly collected data also included the high-resolution M1 LiDAR. In each recording, every LiDAR created its point cloud. For easier handling, all of the different point clouds in a recording were merged and transformed into a joint coordinate system. To make data handling easier and prevent long recordings from taking up the training, the recordings were limited to a maximum of 400 frames.

After collecting the data, it was decided to use a filter to remove all points in the point clouds that belonged to the boat on which the data was collected. This was

done since the boat always has the same position relative to the point cloud origin, and there is no need to predict these points.

3.1.1 Data annotation

To enable segmentation, all collected data needed to be annotated. Here it was decided to only annotate for those classes which had direct contact with water. As such, the following classes were identified as necessary for annotation: *Other Boat, Water, Boat Dock, Ashore, Other object in water*. Figure 3.2 below illustrates the color scheme utilized for annotating with these five classes.

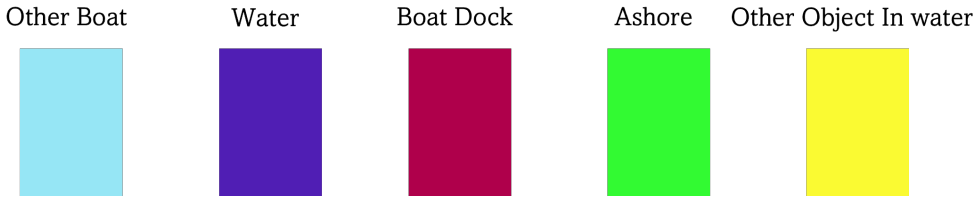


Figure 3.2: Class colours used

3.1.2 3D to 2D projection

The 3D point cloud was transformed into a 2D image, using spherical view projection technique [7]. To assign each point to a pixel in the image the projected x- and y-coordinates were rounded to the nearest integer pixel index. If two or more points were projected onto the same pixel index, only the point that was closest to the LiDAR (i.e., the point with the smallest depth value) was kept. The depth value represents the distance of the point from the LiDAR and is calculated as the Euclidean norm of the point's x-, y-, and z-coordinates. This ensures that the visible points in the image are those that are closest to the LiDAR, and any points that are occluded by other points are not visible.

To optimize the network's performance in terms of both accuracy and speed, it was necessary to minimize the complexity of the 2D images. To achieve this objective, a method was devised to determine the optimal FoV for the images. This involved removing pixels that did not contain informative content (pixels with depth and intensity equal to zero), without compromising the integrity of critical features. The selection of the most suitable FoV was accomplished through systematic testing of various values. One critical scenario was that a too narrow FoV could result in information loss when the boat was close to objects. Therefore, the chosen frame for study was a frame captured when the boat was docked at a pier, as shown in Figure 3.3. An FoV from -20° to 30° was ultimately selected for optimal informative content.

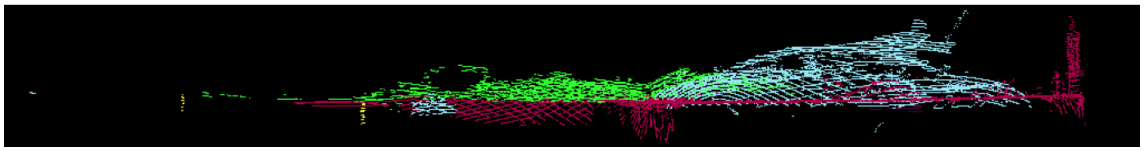
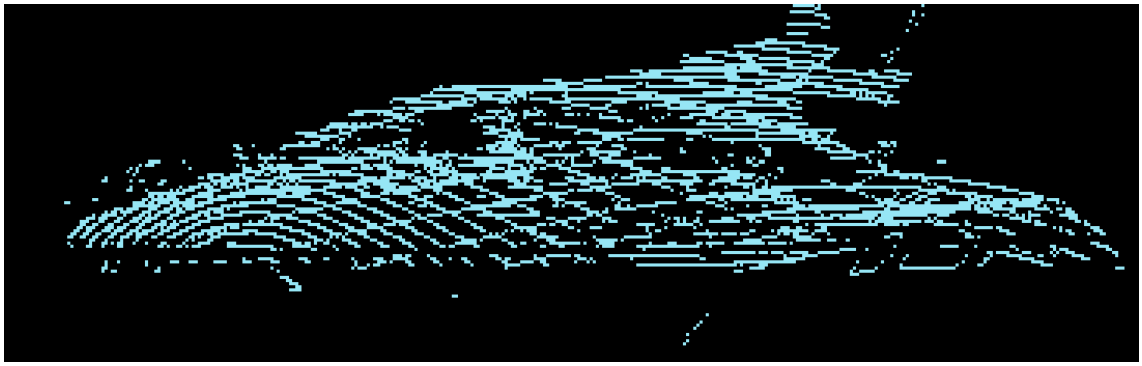


Figure 3.3: 2D spherical projection image of annotated point cloud data with FoV from -20 to 30.

The next step involved the selection of appropriate width and height dimensions for the images. Here, a concern was that when changing the image size, it could result in points from the point clouds ending up in the same pixel and therefore only the point closest to the LiDARs being used. This was investigated and the result showed that changing the image size did not affect the information in the image significantly. The appropriate width and height were achieved through an investigation of different dimensions, with a focus on identifying the image size that maximized the recognition of boat-like shapes. Figure 3.4 illustrates the results of this process.

(a) Width=2048 and Height=128



(b) Width=1024 and Height=128

(c) Width=1024 and Height=64

Figure 3.4: Visualization of an other boat with different width and height.

Upon inspection of the different values, it was observed that an image resolution of 2048x128 (Figure 3.4a) resulted in a boat shape that was somewhat challenging to

discern. By halving the width to 1024, a clear boat shape was obtained, as depicted in Figure 3.4b. An even lower resolution of 1024x64 was also tested (Figure 3.4c) but yielded poor results in terms of boat detection when visually examining the image. Consequently, a resolution of 1024x128 was deemed optimal for the purposes of this Master's thesis.

The 2D projection images used for training were created using two channels: one depth channel and one intensity channel. Figure 3.5 displays the two different channels. The top image shows the depth channel, where darker blue color corresponds to smaller values, i.e. closer to the LiDARs, and brighter blue and green represent greater values, indicating greater distance from the LiDARs. For the intensity image, brighter colors correspond to high intensity, whereas dark blue indicates low intensity values.

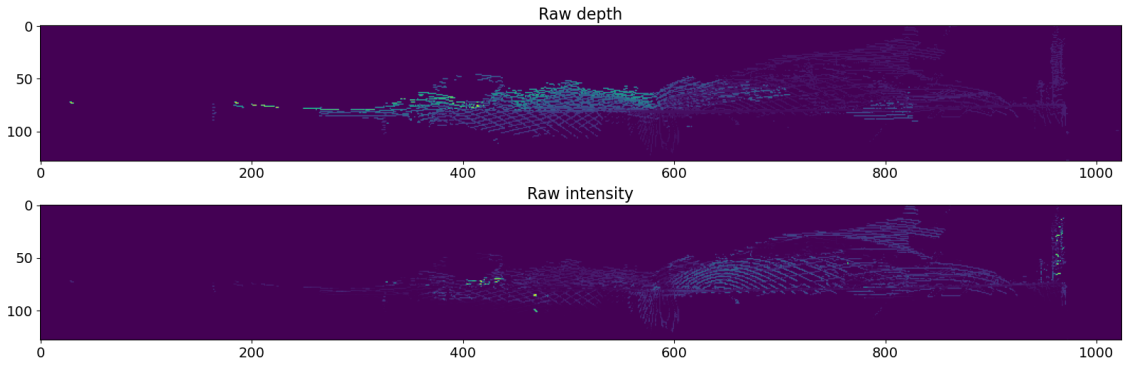


Figure 3.5: Depth and intensity channel from the raw data.

3.1.3 Dataset

The dataset used was a collection of 11,149 frames from several recordings. The dataset was split into training, validation, and test sets in the following percentage: 80 (training), 10 (validation), and 10 (test). Since each frame in a recording was similar to the previous and next frame a test set was created which consisted of separate recordings to simulate an unseen environment during benchmarks. 8314 frames were low resolution frames from previous recordings when the high resolution LiDARs were not mounted on the boat. The last 3000 frames were a mix of high resolution and low resolution data in each frame, since the Bpearl, Helios and M1 Lidars were all collecting data.

3.2 Data preprocessing

Different data preprocessing techniques were tested on the entire dataset, i.e. both the low resolution and high resolution data. As mentioned in section 2.3.1 there are several influences that can alter the intensity coming back to the sensor. The method used as a correction for these influencers was both a theoretical approach combined with an empirical study of the data to find tuning parameters. Furthermore, nor-

malization of the 3D LiDAR point cloud was also tested to help the gradient descent method as discussed in section 2.3.2.

3.2.1 Normalization

As mentioned before, each LiDAR frame has two channels, depth, and intensity. Both channels were normalized in different ways. The depth channel was normalized by scaling the numbers between -1 and 1 as depicted in section 2.3.2. The M1 LiDAR has the longest range of 150 meters. An investigation was conducted into how the distribution of the maximum range varies across all frames. The greatest distance was recorded at 149.32 meters in the test set. Based on the results, it was determined that normalization through scaling with a maximum value of 150 meters was appropriate.

Histogram equalization on the intensity was done by computing the CDF and changing the distribution of the intensity values as mentioned in chapter 2.3.3. The results can be seen in Figure 3.6.

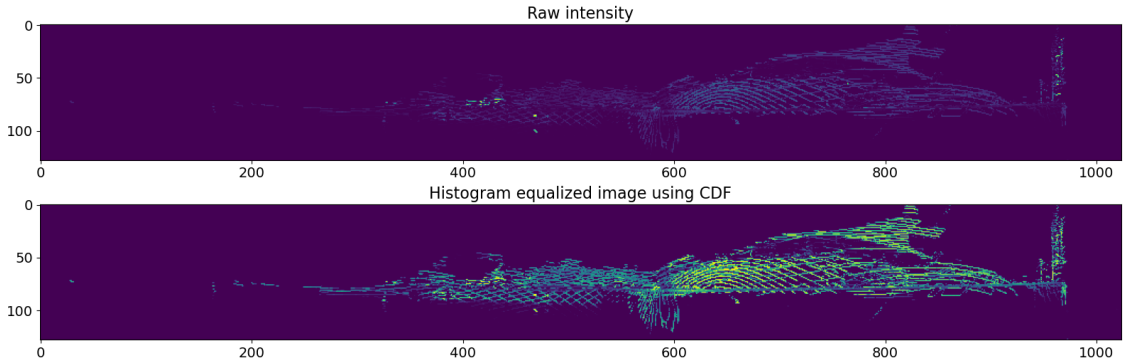


Figure 3.6: Raw and history equalized intensity.

3.2.2 Intensity and depth affection

In Section 2.3.1, a theoretical correction model was presented to compensate for the degeneration of intensity over range and angle of incidence. Here, it was decided to utilize a simplified version of Equation 2.1 by omitting the angle computation, due to the complexity of a plane-fitting algorithm.

$$I_c = I \frac{R^2}{R_{ref}^2} \tag{3.1}$$

Equation 3.1 was employed to use the depth information in order to perform intensity compensation. The raw intensity data showed a similar behavior in that the intensity degrades following an inverse logarithmic function over distance. The first subplot in Figure 3.7 shows the behavior of the intensity distribution and corresponding CDF of the raw data, while the rest of the subplots show the distribution and CDF for different cases where the data has been preprocessed.

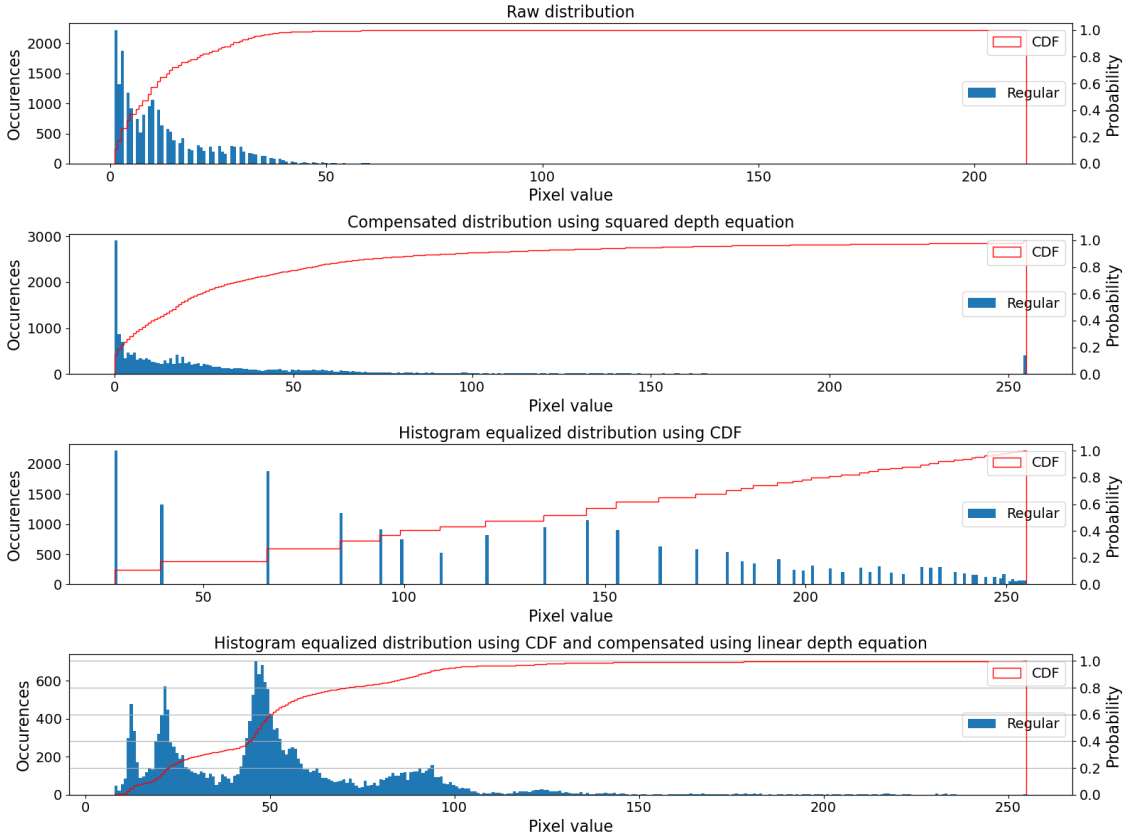


Figure 3.7: LiDAR intensity distribution for one image with different correction techniques.

An investigation was made on how the distribution of the intensity turned out after the histogram equalization was applied. This result can be seen in the third subplot in Figure 3.7. The figure shows that due to the high amount of intensity being in the lower range in the raw data the histogram equalization compensated data is skewed towards the maximum value of 255. This can be confirmed by looking at the actual images in Figure 3.8 where the bright yellow parts are intensity values closer to 255. Hence, a decision was made to apply the depth compensation on the histogram equalized intensities as well. Since the consequence of using histogram equalization is a broader distribution of the data, a linear depth equalization equation was applied compared to the squared in equation 3.1. Equation 3.2 is the linear correction model and the result of the depth correction can be seen in the fourth subplot in Figure 3.8.

$$I_c = \frac{R}{R_{ref}} \tag{3.2}$$

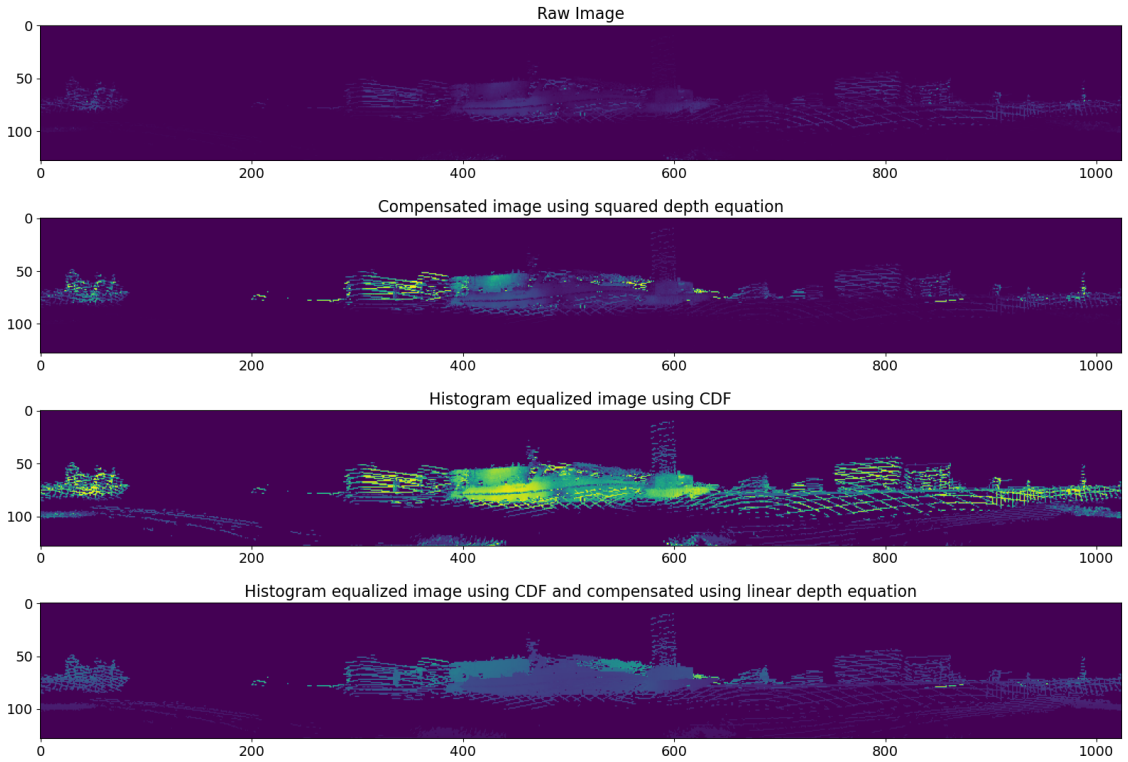


Figure 3.8: Intensity for one image with different correction techniques.

This had two desirable affects on the image. In the center there is a boat (dark blue) and behind is a building (blue/green). The first three subplots in Figure 3.8 show a gradient over the hull of the boat, whereas the fourth image shows a more consistent intensity value across the hull. Secondly, there is a clear distinction in the image of object boundaries and shapes which is desirable according to section 2.3.5.

3.3 Software and pipeline optimization

Since one of the objectives of this thesis was to run the pretrained model on an embedded device, the software for creating the model needed to be optimized for such hardware. As mentioned in section 1.1.1, the previous CNN model was written in PyTorch Lightning. While PyTorch does support running on embedded devices, PyTorch Lightning does not and both PyTorch and PyTorch Lightning are primarily used by and targeted towards researchers and prototyping. Therefore, a decision was made to change the framework to Tensorflow and use their mobile specific and embedded device oriented Tensorflow Light software. As a result, the architecture, loss functions, and data preprocessing methods were rewritten in Tensorflow and the final pre-trained CNN model was saved in the .tflite format. The .tflite CNN models could then be exported to the embedded device and tested on recorded Rosbags on the hardware that was used on the boat.

One major bottleneck in the previous pipeline was in the process of training the

network. Previous work used .CSV format to store the images. While this is convenient, in terms of being able to investigate the data directly, it is not optimized for Python in terms of speed. Since the dataset consisted of several GB worth of data the process of loading the .CSV files on the fly during training was very slow. Therefore, a decision was made to migrate to the .py format which is a binary format optimized for Python. As a result of this, the training time was reduced by more than ten times which enabled faster prototyping and testing.

3.4 Testing and evaluation

In order to systematically examine the impact of various preprocessing steps, network structures, and loss functions on the outcome of semantic segmentation, a series of tests were designed. These tests were designed to operate in an elimination fashion, where the most favorable outcome from each test was carried over to the next subsequent test.

The training and evaluation were performed on a computer with an Intel(R) Core(TM) i7-9800X CPU @ 3.80GHz, 32 GB RAM, and an Nvidia Quadro RTX 4000 GPU. During training the learning rate was set to 0.0001, this value was chosen since it previously had shown good results [7]. All models were trained for 10 epochs. The prediction time was done with a tflight model on the CPU and the model was set to compute on 16 threads. The CPU prediction time was chosen over GPU prediction time due to the fact that a GPU in a PC built for training deep machine models is not normally used in embedded applications.

3.4.1 Network structure

It was decided to test five different network architectures and compare which one gave the best results. All of the networks were based on U-Net and for these tests, CE loss was used. The input size to the network was the image size (128x1024) x 2 since the image had two channels. One problem with U-Net is that the 3x3 convolutional layers may lead to information loss when processing large input images since they produce outputs that are smaller than the inputs it was decided to compare two identical network structures, one with padding and one without.

Previous work found that a U-Net architecture with encoder channels as 64, 128, 256, and 512 used for the convolutional channels and in a reversed order for the up-convolutional layers performed well, and thus this structure was kept as one of the networks, containing 7.7 million trainable parameters. An overview of this structure is shown in Figure 3.9.

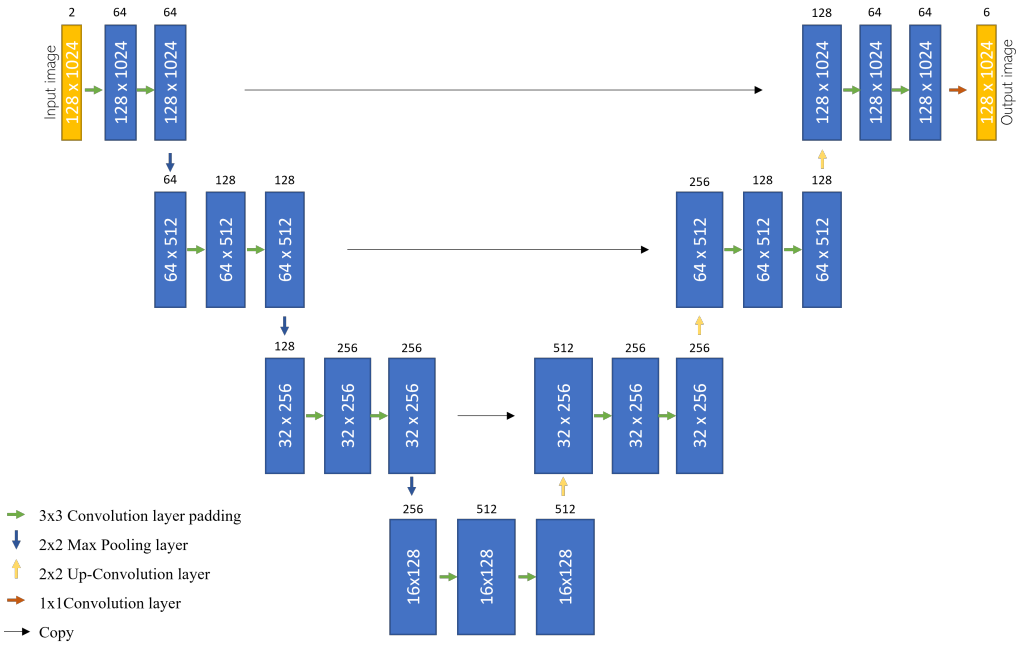


Figure 3.9: An overview of one of the tested U-Net structure.

However, since the training dataset was homogeneous, reducing the number of trainable parameters was desirable. This was accomplished in two different ways, by reducing the number of channels in the convolutional layers and by increasing the depth of the network and eliminating the last convolutional layers. A summary of the different network structures that were tested is shown in Table 3.1.

Table 3.1: Network structures

Test	Encoder channels	Decoder Channels	# trainable parameters
N:a	[2 64 128 256 512]	[512 256 128 64 6]	7 697 094
N:b	[2 32 64 128 256]	[256 128 64 32 6]	1 925 478
N:c	[2 64 128 256]	[256 128 64 6]	1 862 598
N:d	[2 32 64 128]	[128 64 32 6]	466 406
N:e	[2 16 32 64 128]	[128 64 32 16 6]	481 974

3.4.2 Loss function

The network structure that was proven to perform best was then used during the testing of three different loss functions. It was decided to test and evaluate CE loss, WCE loss, and weighted focal loss. The weights used for the two weighted loss functions were determined by the inverse proportionality of their respective frequency. After the loss functions had been tested, the loss function that provided the best results was kept for the rest of the testing.

3.4.3 Normalization, depth, and intensity affection

After determining the most appropriate loss function and the number of epochs, a series of tests was conducted to examine the impact of normalizing and depth compensation of the LiDAR data. These tests were a different combination of normalization and depth compensation. The following tests were done:

Test P:a	Raw data
Test P:b	Normalized depth
Test P:c	Histogram equalized intensity
Test P:d	Normalized depth, and histogram equalized intensity
Test P:e	Normalized depth, histogram equalized intensity, and linear depth correction
Test P:f	Normalized depth, and squared depth correction

To be able to run the pretrained model on an embedded device only one channel could be used. Therefore, the impact of the model when only using one channel was tried. This was evaluated by taking away one channel at a time and evaluating the results.

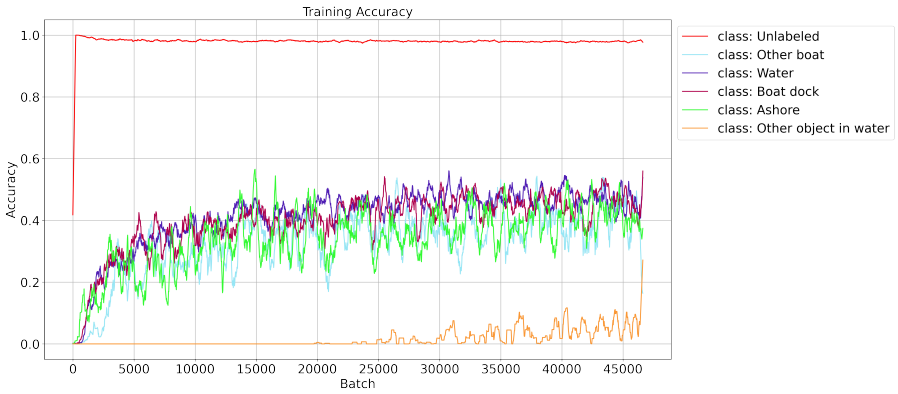
4

Results

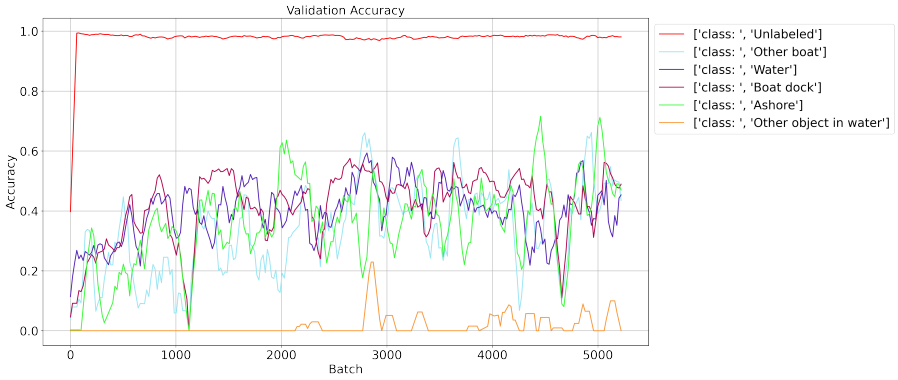
In this section the result of the different tests are presented. All models were trained with a learning rate of 0.0001 and the training went on for 10 epochs.

4.1 Network structure

The training accuracy and validation accuracy of the model without padding are presented in Figure 4.1, while the corresponding accuracy figures for the padded model are illustrated in Figure 4.2. A comparison of these results revealed that utilizing padding improved the performance of the model compared to the scenario where padding was not used, leading to the elimination of the model without padding.

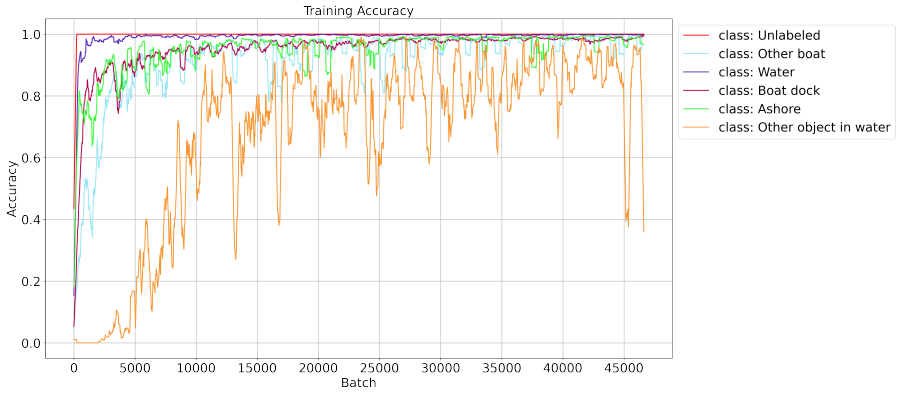


(a) Training accuracy

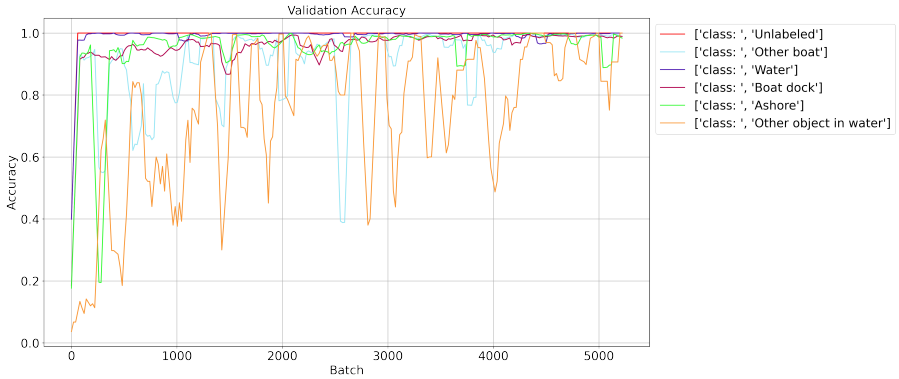


(b) Validation accuracy

Figure 4.1: The training and validation accuracy for each class during training on the network without padding.



(a) Training accuracy



(b) Validation accuracy

Figure 4.2: The training and validation accuracy for each class during training on the network with padding.

4.1.1 Comparison between different network structure

Table 4.1 presents the per class accuracy, and Table 4.2 shows the mean prediction time of the test set for different network structure tests. The network structures can be found in Table 3.1.

Table 4.1: Class accuracy for each network structure test

Class	Test				
	N:a	N:b	N:c	N:d	N:e
Unlabeled	100%	100%	100%	100%	100%
Other Boat	74.22%	84.51%	74.11%	75.79%	77.52%
Water	99.72%	99.62%	99.91%	99.76%	99.76%
Boat dock	73.20%	75.31%	71.16%	66.10%	63.90%
Ashore	76.92%	78.70%	77.00%	80.19%	78.66%
Other object in water	41.37%	28.99%	35.12%	38.18%	50.83%
Mean class accuracy	**77.57%**	**77.86%**	**76.22%**	**76.67%**	**78.37%**

Table 4.2: Mean prediction time on the CPU for each test, operating on 16 threads

Test	N:a	N:b	N:c	N:d	N:e
Mean prediction time	0.9004s	0.2878s	0.6664s	0.2130s	0.1239s

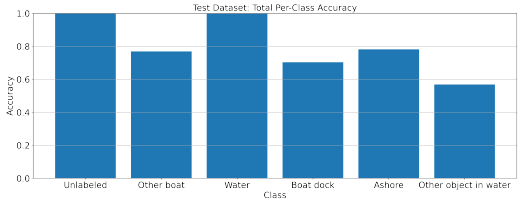
For all network structures, the accuracy for 'Unlabeled' reached 100% and over 99% for 'Water'. There was not a significant difference in the accuracy for 'Other boat', 'Boat dock', and 'Ashore' between the network structures. However, for the class 'Other object in water' the accuracy ranged between 28% to 51 %. The total accuracy for each test was almost the same with only a small difference. The highest total accuracy was 78.37% for test N:e, which also had the fastest prediction time, as shown in Table 4.2. Test N:e with encoder channels 2-16-32-64-128 and decoder channels 128-64-32-16-6 was selected for further testing.

4.2 Loss function

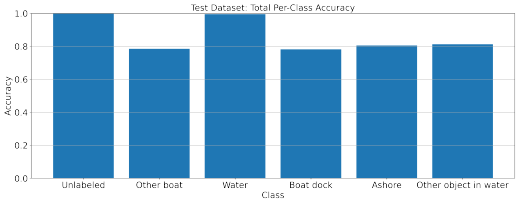
The accuracy for each class is depicted in Figure 4.3, with each loss function represented in the subfigures. The weights utilized for WCE and weighted Focal loss can are shown in Table 4.3.

Table 4.3: Weights used in WCE loss and weighted Focal loss

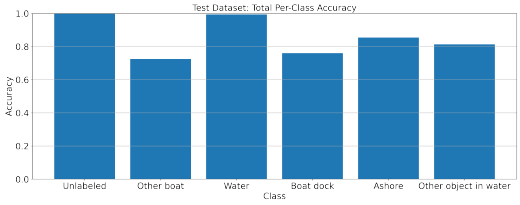
Unlabeled	Other boat	Water	Boat dock	Ashore	Other object in water
0.1829	18.66	2.965	12.40	16.33	1805.2



(a) CE loss



(b) WCE loss

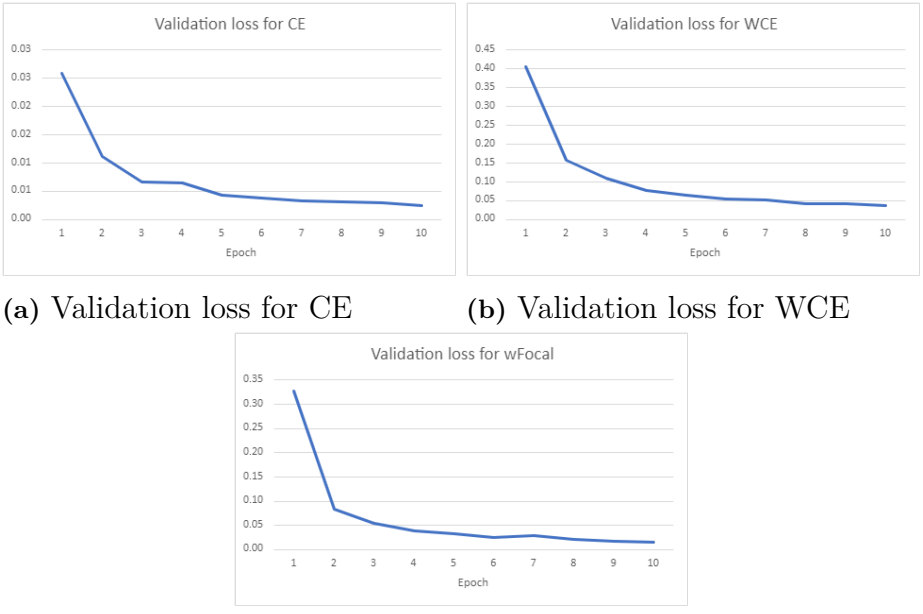


(c) Weighted Focal loss

Figure 4.3: Accuracy for each class for the three different losses.

Network e from Table 3.1 was used in the following tests. For all three loss functions, the two classes 'Unlabeled' and 'Water' achieved a test accuracy of nearly 100%. There was not a significant difference in the accuracy for the classes 'Other Boat', 'Boat Dock', and 'Ashore' across the different losses. However, for the two

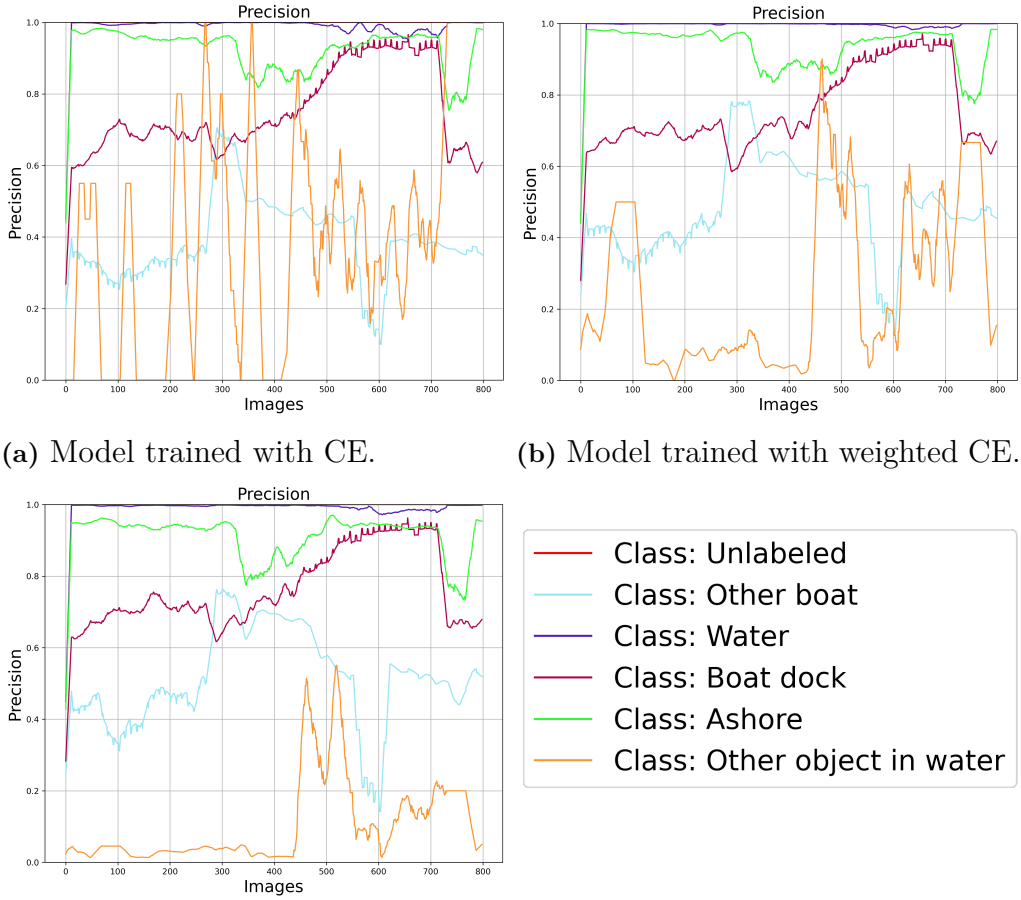
weighted loss functions, WCE and weighted Focal, the accuracy for 'Other object in water' exceeded 80%, while the accuracy with CE loss was 58%. The highest overall accuracy was observed with weighted CE loss.



(a) Validation loss for CE **(b)** Validation loss for WCE

(c) Validation loss for wFocal

Figure 4.4: Validation loss over 10 epochs for three loss functions



(a) Model trained with CE.

(b) Model trained with weighted CE.

(c) Model trained with weighted Focal.

(d) Legends for the three graphs a-c.

Figure 4.5: Precision score for each class when the model has been trained with different loss functions.

Figure 4.5 highlights the convergence speed of each loss function. The values on the y-axis vary considerably. This variation is caused by the substantial impact of class weights on the sum of the loss function. The WCE loss was more stable than the CE loss and the weighted focal loss was converging faster than the WCE loss. The results align well with the theory presented in section 2.5 where WCE loss should perform better with an imbalanced dataset and focal loss should have a faster converging time than WCE. However, WCE was still the better option of the three due to the higher overall accuracy score in Figure 4.3b.

As stated in Section 2.6, the precision score measures the proportion of correctly predicted class instances for a given class out of all the positive predictions for that class. This is illustrated well in Figure 4.5c. Even though the class accuracy for 'Other object in water' in Figure 4.3c was high, the precision was low. The result of this was that the network sporadically guessed the location of 'Other object in water,' correctly classifying the true positive about 80% of the time but with a high false positive rate.

There were no significant differences between the results when using different loss functions. Therefore, it was decided to use WCE loss for further testing since it provided a good middle ground between accuracy, convergence speed, and precision.

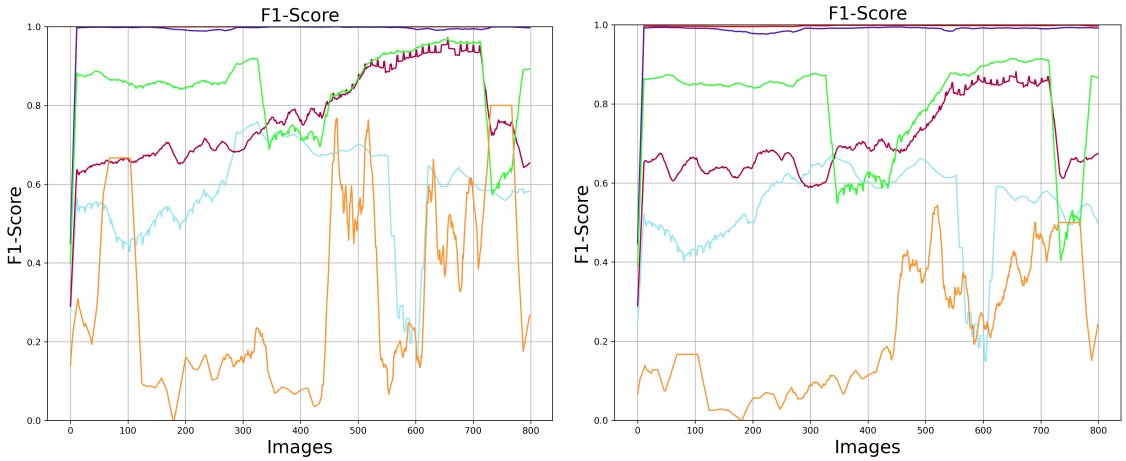
4.3 Normalization, depth, and intensity affection

Table 4.4 displays the per class accuracy for each test, along with the mean accuracy of each test. Each test (P:a-P:f) represents a different preprocessing method as described in Section 3.4.3.

Table 4.4: Per class accuracy for each test (a-f)

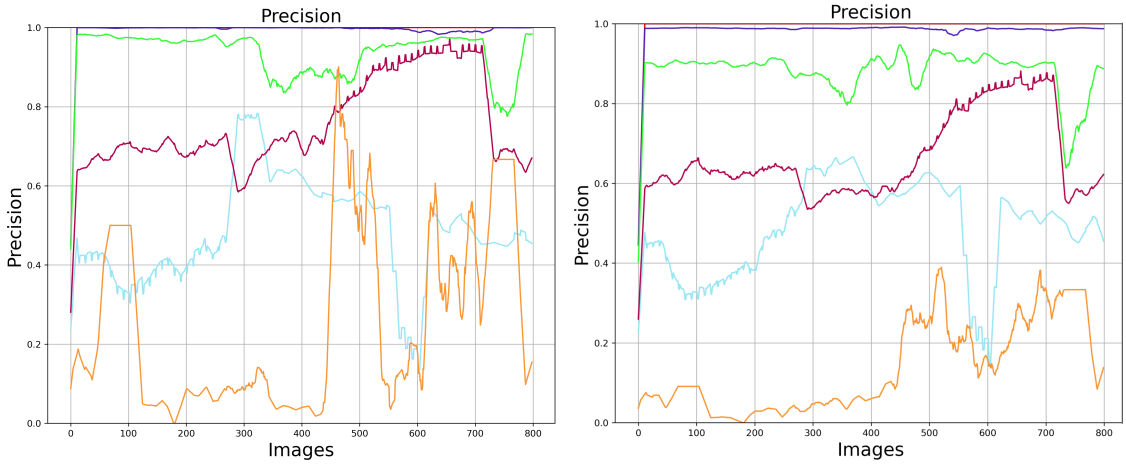
Class	Test					
	P:a	P:b	P:c	P:d	P:e	P:f
Unlabeled	99.98%	99.91%	99.98%	99.93%	99.90%	99.54%
Other Boat	78.59%	78.56%	77.53%	70.87%	73.63%	67.32%
Water	99.64%	99.39%	99.06%	99.29%	98.85%	99.01%
Boat Dock	78.53%	65.31%	71.08%	59.87%	64.17%	75.95%
Ashore	81.03%	79.03%	79.16%	75.92%	77.98%	80.99%
Other object in water	81.57%	79.57%	80.21%	83.51%	73.76%	91.38%
Mean class accuracy	**86.56%**	**83.63%**	**84.50%**	**81.57%**	**81.38%**	**85.70%**

The classes 'Unlabeled' and 'Water' consistently performed well regardless of the data preprocessing. All other classes achieved decent accuracy after preprocessing. The mean accuracy ranged between 81% and 86% for all tests. The two tests that achieved the highest accuracy were test P:a, which used raw and unprocessed data, and test P:f, which involved normalizing the depth and applying squared depth correction. Figure 4.7 displays the F1-score and the precision for these two tests with the highest overall accuracy. The F1-score and precision for the remaining tests are presented in Appendix A.2

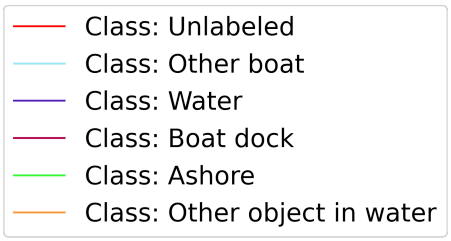


(a) F1-score for test P:a (raw data) **(b)** F1-score for test P:f (squared)

Figure 4.6: F1-score for the two tests with highest total accuracy.



(a) Precision for test P:a (raw data) **(b)** Precision for test P:f (squared)



(c) Legend for above graphs.

Figure 4.7: Precision for the two tests with highest total accuracy.

The F1-score and precision for each class reached similar values for both tests. The raw data had slightly higher values for some of the classes and therefore the raw data was used for the following tests.

4.3.1 One channel

The following tests were performed on raw data with weighted CE loss. Table 4.5 shows the per class accuracy for the "one channel" test. When only the depth channel was used, the accuracy reached a higher value for all classes except 'Other Boat' compared to when only the intensity channel was used. As a result of this, the mean accuracy received the highest value when only the depth channel was used.

Table 4.5: Per class accuracy when only one channel has been used.

Class	Only depth channel	Only intensity channel
Unlabeled	99.92%	99.84%
Other Boat	68.63%	75.32%
Water	99.40%	99.48%
Boat Dock	82.62%	60.35%
Ashore	79.81%	72.44%
Other object in water	79.55%	75.89%
Mean accuracy	**84.99%**	**80.55%**

Figure 4.8 and Figure 4.9 visualize the F1-score and precision, respectively, for the case where only the depth channel was used.

Figure 4.8: F1-Score when only depth channel was used.

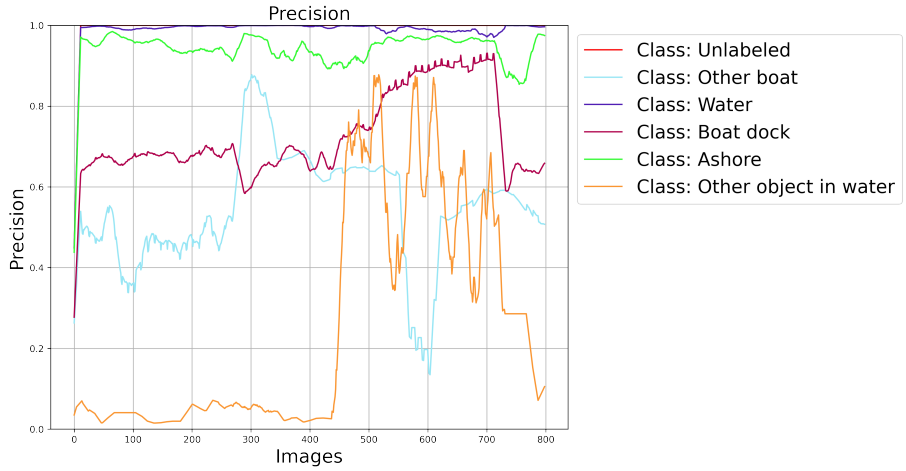


Figure 4.9: Precision when only depth channel was used.

4.4 Other observations

Here, observations of the results which were deemed important and could contribute to the overall understanding of the behavior of the model are gathered.

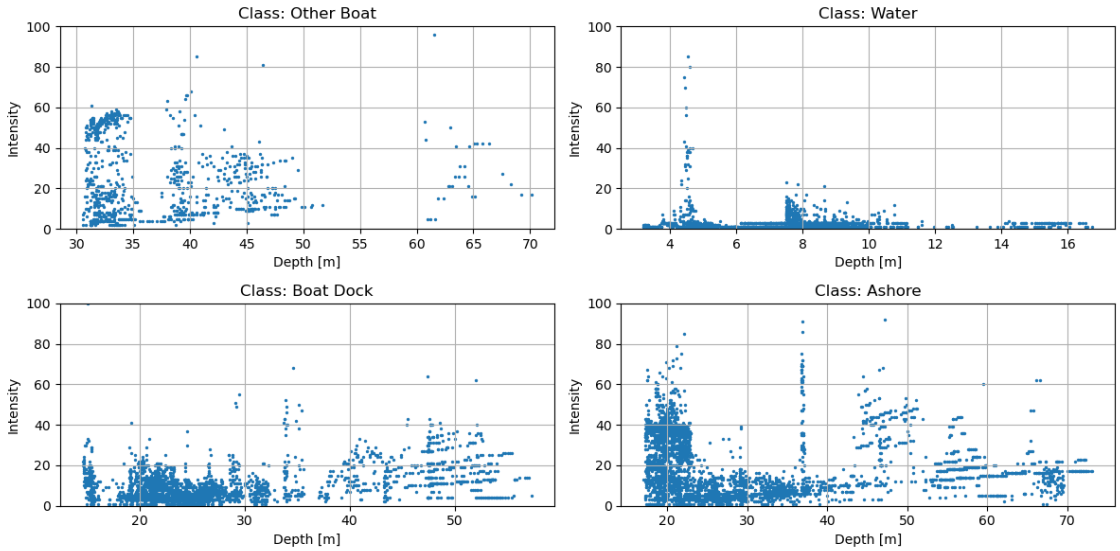


Figure 4.10: Correlation between depth and intensity for one image with high resolution LiDAR (M1).

What can be observed in Figure 4.10 is that there are some differences in intensity per class. There are also indications that the intensity drops in the 'Water' class and a little in the 'Ashore' class but it is not as prevalent in the other two.

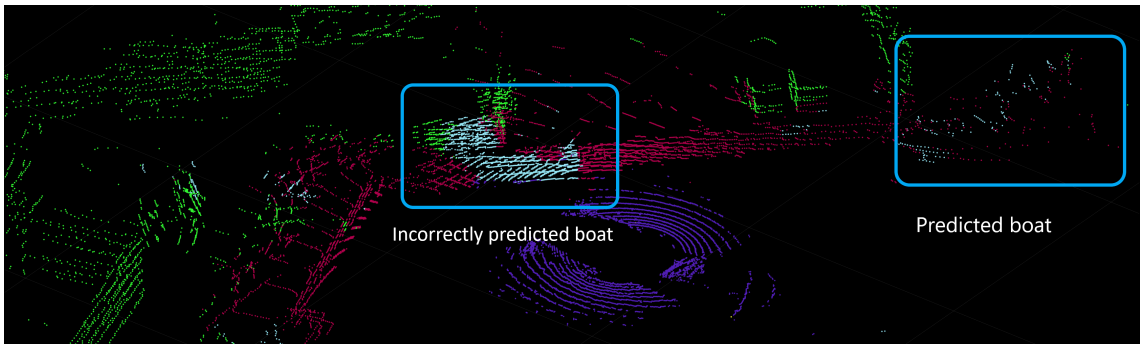


Figure 4.11: Prediction where the class 'Other boat' is confused with the classes 'Boat dock' and 'Ashore'.

Figure 4.11 highlights how the model struggles with boat predictions. In the image, the model confuses the classes 'Boat dock' and 'Ashore' with the 'Other boat'. This was a common behavior of the model. This image was part of the training or validation set.

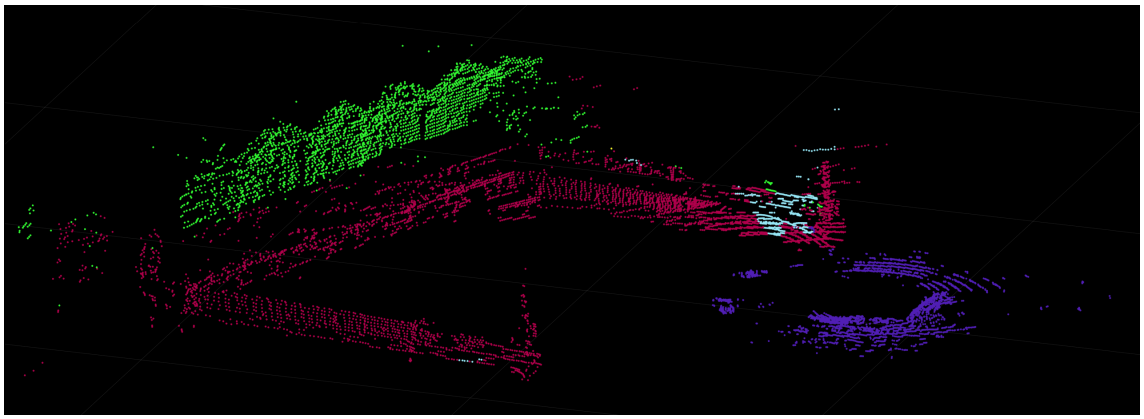


Figure 4.12: Prediction where the class 'Other boat' is confused with the edge of the boat dock.

Figure 4.12 shows a common faulty prediction, namely that the model often thinks there should be a boat at the edge of the boat dock. This image is part of the training or validation set.

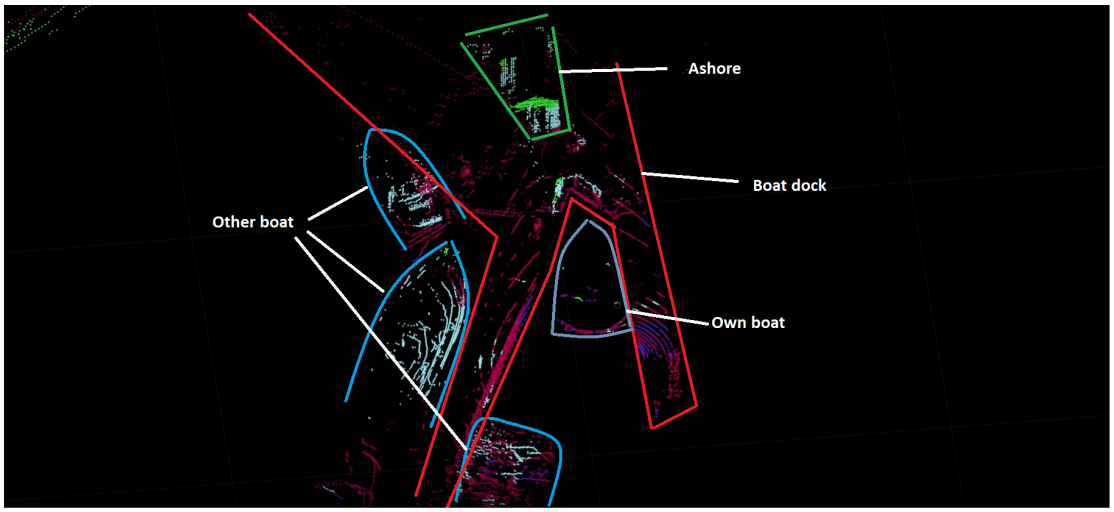


Figure 4.13: Prediction of completely unseen data. The drawn lines show the boundary of different classes, and the dots are the predictions.

Figure 4.13 shows a prediction on unseen data. The prediction showed that the model had a hard time discerning the edges of objects. The 'Other boats' were partly correctly predicted, but there were still a lot of dots that the model confused. Moreover, the boat was located between two boat docks. The boat docks were therefore where the class 'Water' usually was. Consequently, the model got confused with the boat dock that was nearby and the class 'Water' as can be seen in Figure 4.13.

5

Discussion

5.1 Network structure

The purpose of testing different network structures was twofold. Could the number of parameters be reduced while maintaining an acceptable accuracy, and could the network architecture be structured such that the prediction time was minimized? Adding padding results in a noticeable increase in both training and validation accuracy as shown in Figure 4.2a and 4.2b. This is likely due to the fact that there is information of significance along the corners and sides of the 2D images and the center cropping that no padding introduces will remove this information.

To reduce the number of parameters and, consequently, the prediction time, one can either decrease the number of channels per encoder/decoder layer or reduce the depth of the network, i.e., the number of encoder/decoder layers. The network structure b and c in table 3.1 shows a comparison of how the depth of the network affects the accuracy and prediction time. Both networks have a similar mean class accuracy of 77.86 and 76.22 percent, but the prediction time is twice as high on test N:b with a prediction time of 0.2878s compared to 0.6664s on test N:c. There is a similar jump in prediction speed between test N:d and N:e, where test N:d has a prediction time of 0.2130s compared to 0.1239s on test N:e. This is while the number of parameters is similar between N:b-N:c and N:d-N:e. One difference between the network structures is the depth. But test N:c has fewer layers and a slower prediction speed. This is also true on tests N:d and N:e, where N:e has more layers but a faster prediction speed. Across all tests, the prediction speed correlates well between the number of channels of the image size, i.e. input channels, and the number of channels of the first encoder. Test N:a and N:c have the slowest prediction time and largest jump between the number of input channels and first encoder channels. The smallest jump is between 2 and 16, which also shows the fastest prediction time. The overall accuracy of all tests is similar but the networks with more layers and therefore more copy operations of different resolution maps have a small advantage. Therefore, a good final network structure is a network with three copy operations, i.e., four layers, where the difference between the number of input channels and the first encoder channels is small. In this case, 16 encoder channels appear to work well.

5.2 Loss function

With such an imbalanced dataset, the goal of weighted cross entropy loss and weighted focal loss is to emphasize the importance of the minority classes in order to compensate for this imbalance. The difference in accuracy between CE loss and WCE in Figure 4.3b indicates an improvement in overall class accuracy. However, when examining the precision of the minority classes such as 'other object in water' and 'other boat', it becomes evident that the network is likely making many guesses. Especially, the weighted focal loss in Figure A.1 shows that the class precision of 'other object in water' in the first half of the test set is close to zero, while the class accuracy in Figure A.1a is near 100%. This indicates that the network is adapting to the weights assigned to each class in the loss function. Since the class, 'other object in water', consists of the least amount of samples in the training set, it has the highest weight (see Table 4.3). If the penalty for misclassifying 'other objects in water' is too high, it becomes preferable to make numerous guesses for this class, since the penalty for misclassifying other objects is much lower.

The converging speed of the three loss functions can be seen in Figure 4.4. The theory discussed in section 2.5 regarding focal loss suggests that the loss function should have a faster convergence and reach lower loss values compared to CE and WCE. Both of these aspects are observed to be true, but the overall accuracy results are still worse, as depicted in Figure 4.3c, indicating a potential case of overfitting. It should be noted that the test set used was not part of the original data set; instead, it comprised two completely separate recordings. Consequently, the test set will most likely show worse accuracy score if the model is overfitted or if the model has trouble generalizing to new unseen data. It could be a combination of the two and the reason could be caused by how the dataset is constructed, in both size and similarity between frames. This will be discussed further in section 5.4.

5.3 Data preprocessing methods

In this section, two data preprocessing methods are discussed. The intensity compensation methods are highlighted and a discussion is carried out about how the model performs when either depth or intensity channel is used as input data.

5.3.1 Intensity compensation methods

For the different data preprocessing methods that are being tested, using raw data yields the best result in mean accuracy and the highest accuracy score for all classes except 'Other Object in Water'. This indicates that preprocessing the data before training makes it harder for the model to find distinct patterns and therefore it results in worse prediction. The F1-score and precision for the raw data also show that the raw data yielded the best results.

The statement that raw data yields better results than preprocessed data when training a CNN is fairly unusual. In general, preprocessing data is considered an

essential step in preparing data for model training, especially when working with CNNs. So, why did the raw data yield the best result in this work? One reason could be that the raw data contains critical information that is important for the model's learning process, that is lost when preprocessing the data.

To try and maximize the information that is contained in the LiDAR data, different preprocessing methods are utilized where the depth information is used to alter the intensity value of each pixel. The reason behind it stems from the fact that materials have certain characteristics such as reflectivity and albedo. So if a material such as wood has an intensity value of 30 at a range of 10m and an intensity value of 5 at 50m, then it could be advantageous to compensate for the loss of intensity, due to range, with the depth value in order to try and shorten the spectrum of intensity values that a given class or material can reflect back. This could, in theory, help the network not only classify objects based on shape but also on intensity, which is conceptually discussed in section 2.3.5. One result that points towards this is Table 4.5, where the class 'Other boat' achieves a higher accuracy when the intensity channel is used, as opposed to the depth channel. Since the albedo of a boat typically is higher, the reflectivity coming back from boat objects could give the model a hint that it is a boat it is detecting. Nevertheless, the results point to the fact that using any of the three compensation methods does not result in an increase in overall accuracy as Table 4.4 shows.

If both the theory in section 2.3.1, which discusses the degradation of intensity over distance, and the theory in section 2.3.5, which explores the influence of colors on models and their ability to identify objects, suggest that these factors can be helpful, why do they not yield successful results in this case? The depth compensation methods use equation 3.1 and 3.2. Both equations incorporate the tuning parameter R_{ref}, which should be adjusted based on testing of the LiDAR equipment. However, in this project, the selection of this parameter is solely based on visual inspection of a few example images. Additionally, it is important to note that each LiDAR sensor is calibrated differently, depending on the brand and methodology employed, which introduces another potential source of error.

The distribution of the relation between depth and intensity is important when analyzing the effects of intensity compensation. Figure 4.10 shows that the intensities for class 'Water' follows a degrading curve where the intensities get smaller further away. This is likely due to that the angle of reflection gets larger when the distance to the LiDAR increases and therefore less reflection comes back. Above 16-17 meters the LiDAR beam likely experiences a total reflection on the water and therefore no reflection comes back beyond this range. Moreover, the classes 'Other boat', 'Boat dock', and 'Ashore' in Figure 4.10 do not appear to show that the intensities degrade over distance. Interestingly, this contradicts what was previously discussed and the reason could be that the classes are not materially bound, i.e. there is more than one material per class. This might explain why the results in Figure 4.5 seem to indicate that the network almost reached the same level of accuracy with only the depth channel as it did with both channels. Consequently, the intensity could be

negatively contributing to the prediction of the intensity alludes to one material, and therefore one specific class, when in fact it could be another material but still the same class.

Lastly, the model does not seem to care if the images are visually more distinct. The theory was that if humans can easily discern between classes such as 'Boat' and a 'Boat dock', the network should also benefit from this. One of the correction methods used was histogram equalization, which results in a visually more distinct image. However, the results in Table 4.4 do not seem to indicate that something more visually appealing for humans to analyze, necessarily translates to better performance for a CNN model.

5.3.2 Intensity and depth channel

One analysis conducted in this thesis aimed to evaluate the performance of the depth and intensity channels in a single-channel scenario. The obtained results, as shown in Table 4.5, reveal distinct patterns between the two channels in terms of per class accuracy. It is observed that, except for the 'Other boat' class, the depth channel demonstrates superior accuracy performance, achieving a value of 68.63%. In contrast, the intensity channel achieves an accuracy of 75.32% for the same class.

An interesting observation emerges when examining the relationship between the accuracy of 'Other boat' and 'Boat Dock'. Table 4.4 shows that if 'Boat Dock' reaches high accuracy, 'Other Boat' reaches slightly lower accuracy. In other words, it appears that the model either learns to predict 'Other Boat' or 'Boat Dock' well and has a harder time predicting both of these classes. This could be a result of 'Boat Dock' and 'Other Boat' often having the same depth and similar shapes in the 2D images. During training, the model encounters a randomized sequence of images, and depending on the order in which it encounters these classes, it may initially learn to predict one class more effectively, consequently misclassifying instances of the other class. One example of this misclassifying where the boat dock is incorrectly predicted as a boat is shown in Figure 4.11.

The better performance of the model when only using the depth channel, compared to when using the intensity channel, indicates that the model places greater importance on the depth information rather than the intensity values when making predictions. Compared to the case when both the depth and intensity channels were used, the difference in results was not that distinct, which strengthens the theory of depth being more important than intensity.

5.4 Dataset quality and overfitting

Since there is evidence pointing towards overfitting, it is important to discuss the quality of the dataset. As mentioned in section 3.1, the dataset consists of a set of recordings. Each recording consists of 400 or fewer frames recorded at a speed of

10 Hz, which means 10 frames per second. Hence, the resulting dataset contains consecutive frames that are similar to one another. This is equivalent to training on images that largely look the same, which can negatively affect the network and lead to overfitting.

To address overfitting caused by the similarity of frames, a potential solution could be to decrease the LiDAR data capture frequency. This approach would introduce a greater temporal spacing between frames, allowing for increased boat movement and enhanced dataset diversity. By exposing the model to a wider range of scenarios, the risk of overfitting to specific frames is reduced, leading to improved performance.

Another issue with the dataset is its high imbalance. This is a consequence of, that the environment in which the recording were made, is homogeneous. For example, many of the recordings have boats close to the boat docks. As a result, the network tends to establish a strong association between boats and boat docks, as discussed previously. Although it is reasonable to expect boats to be located near boat docks in certain instances, it is crucial to acquire data that encompasses diverse boat locations.

The association between 'Boat dock' and 'Other boat' can be seen in Figures 4.11 and 4.12. There are several issues that these figures highlight. The first issue is that the model incorrectly places the class 'Other boat' randomly on the class 'Boat dock'. This mostly occurs when there is another object on top of the boat dock, which makes it appear as if the boat dock is not flat. Moreover, the model struggles with details, particularly when predicting 'Other boat', as can be seen in Figure 4.11. The model thinks there is a boat in the top right corner, but many of the dots close to the boat are still predicted as 'Boat dock'. Both of these issues are most likely caused by too few training examples of the 'Other boat' class. The lack of detail could also be the result of too few learning parameters in the model. Figure 4.13 shows a completely new scene, where the model predicts unseen data. This highlights the previous suspicions that the model is overfitted. The class 'Boat dock' is misclassified as 'Water' because the water is usually located close to the boat. The 'Ashore' class is classified as 'Boat dock', 'Other boat', and 'Ashore'. This may indicate that the model has not learned to associate shapes with the classes but rather where the object is located in the vertical space of the image. The model clearly shows signs of overfitting in that it has problems transferring its knowledge to new unseen data.

As mentioned earlier, the model encounters significant challenges predicting 'Other object in water' and mostly guesses. This issue could be a result of the imbalanced dataset, which contains very few instances of 'Other object in water'. To address this problem, more data on this class would need to be collected. However, it is important to acknowledge that the marine environment has a lower proportion of objects in the water compared to the other classes. Therefore, when gathering more data that contains objects in the water, an increase in data from other classes would also be expected. One potential solution is to plan and stage the collection of data

specifically targeting the 'Other objects in water' class. For instance, the data collection process could be organized in a scene with plenty of buoys in the water, ensuring a more balanced representation of this class within the dataset.

One additional factor that has the potential to impact the quality of the dataset is the possibility of annotation errors. The labeling of each data point was done by hand, and therefore, there is a possibility that some of the points were wrongly labeled, resulting in errors in the ground truth. This could lead to the model predicting a label correctly compared to the reality but getting a wrong prediction compared to the ground truth.

5.5 Objectives that were not investigated

Two objectives of this thesis were to investigate "*how the accuracy of the model change when different data resolutions are used or when a combination of resolutions is employed*", and "*how weather drawbacks such as rain and snow could be handled*". However, these objectives were not investigated in this work due to several factors. The main reason is that there was not enough collected data for this purpose. The collected data did not include fog, rain, and snow, making it unworkable to investigate the impact of weather drawbacks. Additionally, to compare different resolutions, it would be necessary to have annotated recordings gathered exclusively by high-resolution LiDARs and comparable recordings gathered exclusively by low-resolution LiDARs. Unfortunately, there was not enough time to collect and annotate this specific type of data, leading to the decision not to investigate this objective.

6
Conclusion

This thesis's purpose is to improve an existing data processing chain and optimize the CNN for fast semantic segmentation on LiDAR data in the marine environment. Through comprehensive analysis and discussion, valuable insights have been gained into the most suitable learning strategy, loss function, and data preprocessing steps for achieving accurate and precise segmentation results.

The addition of padding in the U-Net architecture of the network significantly improves both the training and validation accuracy. Analysis of network structures reveals that networks with fewer parameters tend to have faster prediction time. Additionally, it is clear that the prediction time is significantly affected by the difference in the number of encoder channels between the input image and the first convolutional layer. A larger difference leads to a slower prediction time. By carefully considering the trade off between the number of parameters, accuracy, and prediction time, it is possible to optimize the network structure for efficient and effective image processing tasks.

When dealing with imbalanced datasets, WCE loss is the most effective technique for addressing the class. However, the precision of minority classes reveals that the network tends to make numerous guesses for classes with fewer instances in the dataset. The difference in performance between the three different loss functions was not very significant.

Preprocessing the data using intensity compensation methods did not yield improved results compared to using raw data. The depth channel was found to be more informative than the intensity channel, with the model placing greater importance on depth information during predictions.

It is possible to use low and high resolution data together during training. Even if there were no investigations for training the model on sparsely high or low resolution data, the result when using both resolutions yielded high accuracy.

One requirement to run a CNN on an embedded device is to use TensorFlow, specifically the export format TensorFlow light. To reach a low prediction time on the embedded device, some key network design decisions can be identified. The number of parameters affects prediction time and the most crucial design decision, as mentioned before, is the difference in the number of encoder channels between the input image and the first convolutional layer. By minimizing the number of parameters

and this difference, a network can be constructed that is able to run in real time on an embedded device.

6.1 Future work

In future work, it would be beneficial to collect a more diverse dataset with a greater range of object locations and additional instances of the minority classes. Exploring different data resolutions and addressing whether drawbacks could also provide valuable insights into improving the model's performance. Additionally, further investigation into optimizing the network structure and loss functions could lead to enhanced accuracy and prediction time.

Bibliography

[1] W. Ning, W. Yuanyuan, and E. Meng Joo, "Review on deep learning techniques for marine object recognition: Architectures and algorithms," *Control Engineering Partice*, vol. 118, 2022. DOI: https://doi.org/10.1016/j.conengprac.2020.104458.

[2] W. Jon, "Autonomous ships timeline – comparing rolls-royce, kongsberg, yara and more," *The AI research and Advisory Company*, 2019.

[3] W. Shi, Y. Zhang, Q. Yu, H. Zhao, and X. Gao, "Ssl: Self-supervised learning for semi-supervised point cloud segmentation," *arXiv preprint arXiv:2207.12939*, 2022.

[4] L. Deng, M. Yang, Y. Qian, C. Wang, and B. Wang, "Cnn based semantic segmentation for urban traffic scenes using fisheye camera," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 231–236. DOI: 10.1109/IVS.2017.7995725.

[5] A. Gupta, "Evolution of convolutional neural network architectures," *Medium*, 2020.

[6] P. Dipti, "Improving performance of convolutional neural network," *Medium*, 2018.

[7] D. Emma and J. Hanna, "Segmented classification in marine environment," M.S. thesis, Chalmers University of Technology, 2022.

[8] Z. Liu, Y. Zhang, J. Cao, J. Cheng, and F. Xu, "Point cloud model compression using graph convolutional networks," *arXiv preprint arXiv:1904.01416*, 2019.

[9] L. Johannes and O. Florian, "Segmented classification of traffic environments using rgb-d data," M.S. thesis, Chalmers University of Technology, 2020.

[10] Q. Hu, B. Yang, L. Xie, *et al.*, "Randla-net: Efficient semantic segmentation of large-scale point clouds," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[11] Y. Wu, Y. Tian, Q. Yang, T. Liu, and Y. Fu, "Understanding few-shot learning: A comprehensive survey," *arXiv preprint arXiv:2005.09830*, 2020. arXiv: 2005.09830 [cs.LG]. [Online]. Available: https://arxiv.org/pdf/2005.09830.pdf.

[12] K. Kim and J. Kim, "Semantic segmentation of marine radar images using convolutional neural networks," in *OCEANS 2019 - Marseille*, 2019, pp. 1–6. DOI: 10.1109/OCEANSE.2019.8867504.

[13] Synopsys. "What is lidar?" (2023), [Online]. Available: https://www.synopsys.com/glossary/what-is-lidar.html (visited on 01/24/2023).

[14] V. Lidar. "Guide to lidar wavelengths." (Aug. 2021), [Online]. Available: https://velodynelidar.com/blog/guide-to-lidar-wavelengths/.

[15] LeddarTech, "How leddartech is driving forward the autonomous vehicle lidar solutions of tomorrow," 2019. [Online]. Available: https://leddarsensor.com/app/uploads/2019/04/Article-How-LeddarTech-is-Driving-Forward-the-Autonomous-Vehicle-LiDAR-Solutions-of-Tomorrow.pdf.

[16] J. Gao, H. Zheng, Y. Zhan, and Z. Chen, "Miniaturized lidar sensors for autonomous driving: A review," *Laser & Photonics Reviews*, vol. 15, no. 4, 2021. DOI: 10.1002/lpor.202100511. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/lpor.202100511. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/lpor.202100511.

[17] Robosense. "Rs-bpearl product specification." (), [Online]. Available: https://www.robosense.ai/en/rslidar/RS-Bpearl (visited on 01/27/2023).

[18] Robosense. "Rs-helios product specification." (), [Online]. Available: https://www.robosense.ai/en/rslidar/RS-Helios (visited on 01/27/2023).

[19] Robosense. "Rs-lidar-m1 product specification." (), [Online]. Available: https://www.robosense.ai/en/RS-LiDAR-M1 (visited on 01/27/2023).

[20] I. C. Guide, *Data preprocessing: Definition, techniques, benefits & examples*, https://ca.indeed.com/career-advice/career-development/data-preprocessing, Accessed on: May 2, 2023, 2023.

[21] A. G. Kashani, M. J. Olsen, C. E. Parrish, and N. Wilson, "A review of lidar radiometric processing: From ad hoc intensity correction to rigorous radiometric calibration," *Sensors*, vol. 15, no. 11, pp. 28 099–28 128, 2015, ISSN: 1424-8220. DOI: 10.3390/s151128099. [Online]. Available: https://www.mdpi.com/1424-8220/15/11/28099.

[22] A. Vain, S. Kaasalainen, U. Pyysalo, A. Krooks, and P. Litkey, "Use of naturally available reference targets to calibrate airborne laser scanning intensity data," *Sensors*, vol. 9, no. 4, pp. 2780–2796, 2009, ISSN: 1424-8220. DOI: 10.3390/s90402780. [Online]. Available: https://www.mdpi.com/1424-8220/9/4/2780.

[23] K. Argueta. “Normalizing feature scaling point clouds for machine learning.” (2021),
[Online]. Available: https://medium.com/@kidargueta/normalizing-feature-scaling-point-clouds-for-machine-learning-8138c6e69f5 (visited on 04/03/2023).

[24] A. Chauhan. “Histogram equalization.” (2021),
[Online]. Available: https://towardsdatascience.com/histogram-equalization-5d1013626e64 (visited on 04/26/2023).

[25] C. F. Sabottke and B. M. Spieler, “The effect of image resolution on deep learning in radiography,”
Radiology: Artificial Intelligence, vol. 2, no. 1, e190015, 2020.
DOI: 10.1148/ryai.2019190015. [Online]. Available:
https://doi.org/10.1148/ryai.2019190015.

[26] K. De and M. Pedersen,
“Impact of colour on robustness of deep neural networks,”
in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, Oct. 2021, pp. 21–30.

[27] R. Laguna, R. J. Barrientos, L. F. Blázquez, and L. J. de Miguel, “Traffic sign recognition application based on image processing techniques,”
IFAC Proceedings Volumes, vol. 47, pp. 104–109, 2014.

[28] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, vol. 9351, pp. 234–241, 2015.
DOI: 10.1007/978-3-319-24574-4_28.

[29] deeplizard, *Introduction to tensorflow with python example*, Nov. 2018.
[Online]. Available: https://deeplizard.com/learn/video/k6ZF1TSniYk (visited on 03/22/2023).

[30] S. Hooda, “Model complexity & overfitting in machine learning,”
Data Analytics, May 2022. [Online]. Available:
https://vitalflux.com/model-complexity-overfitting-in-machine-learning/?utm_content=cmp-true (visited on 03/22/2023).

[31] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, “Learning hierarchical features for scene labeling,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1915–1929, 2013.
DOI: 10.1109/TPAMI.2012.231. [Online]. Available:
https://arxiv.org/abs/1312.5851.

[32] H. Lamba, “Understanding u-net,” *Towards Data Science*, Jul. 2018.
[Online]. Available:
https://towardsdatascience.com/understanding-u-net-61276b10f360 (visited on 03/03/2023).

[33] A. Kumar. "Loss functions and their use in neural networks." (2021), [Online]. Available: https://towardsdatascience.com/loss-functions-and-their-use-in-neural-networks-a470e703f1e9 (visited on 02/01/2023).

[34] TurinTech. "What is imbalanced data and how to handle it?" (), [Online]. Available: https://www.turintech.ai/what-is-imbalanced-data-and-how-to-handle-it/ (visited on 01/30/2023).

[35] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[36] N. Threads. "Categorical cross entropy loss: The most important loss function." (2021), [Online]. Available: https://neuralthreads.medium.com/categorical-cross-entropy-loss-the-most-important-loss-function-d3792151d05b (visited on 02/01/2023).

[37] J. Gao, L. Li, X. Liu, and M. Zhou, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1708.02002, 2017. arXiv: 1708.02002. [Online]. Available: http://arxiv.org/abs/1708.02002.

[38] W. L. et al., *IOP Science*, vol. 428, no. 1, p. 012 043, 2021. DOI: 10.1088/1757-899X/428/1/012043. [Online]. Available: https://iopscience.iop.org/article/10.1088/1757-899X/428/1/012043/pdf.

[39] J. Almeida and I.-M. Sintorn, "Benchmarking deep learning frameworks for 3d segmentation on open datasets," in *17th IEEE International Symposium on Biomedical Imaging (ISBI)*, 2020. [Online]. Available: https://research.chalmers.se/publication/526057.

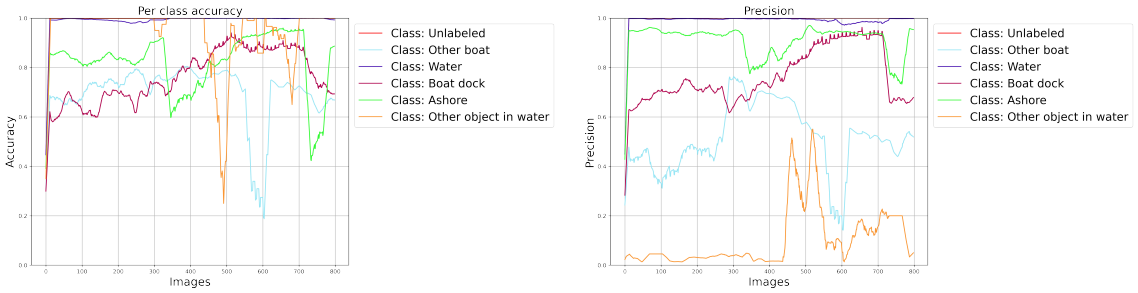
[40] M. Kukar and S. Hore, "Comprehensive guide on multiclass classification metrics," *Towards Data Science*, Aug. 2021. [Online]. Available: https://towardsdatascience.com/comprehensive-guide-on-multiclass-classification-metrics-af94cfb83fbd.

[41] R. Aranha, "Multi-class model evaluation with confusion matrix and classification report," *Towards Data Science*, Feb. 2020. [Online]. Available: https://towardsai.net/p/l/multi-class-model-evaluation-with-confusion-matrix-and-classification-report.

A

Appendix

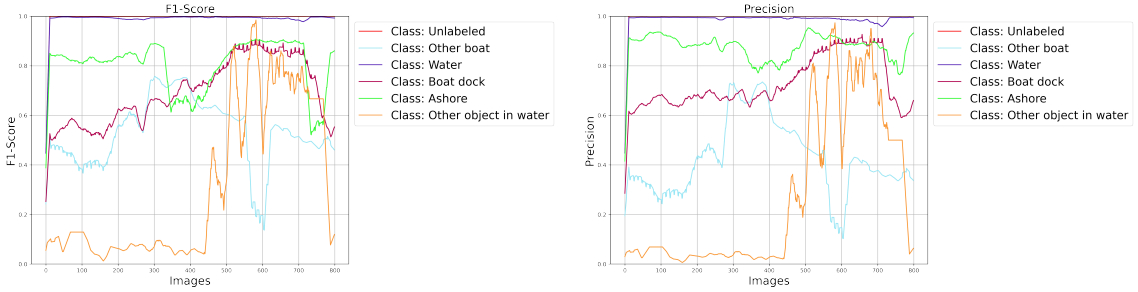
A.1 Compairtion perclass accuracy /precision



(a) Class accuracy (recall) on test set **(b)** Class precision on test set

Figure A.1: Comparison of accuracy and precision for each class with Weighted Focal loss

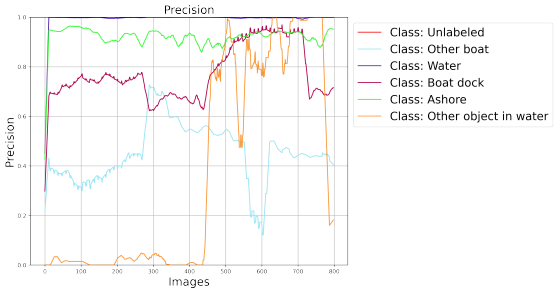
A.2 F1-score and precision



(a) F1-score test b **(b)** Precision test b

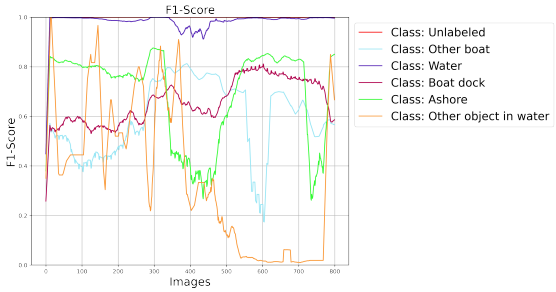
Figure A.2: F1-score and precision for test b

(a) F1-score test c



(b) Precision test c

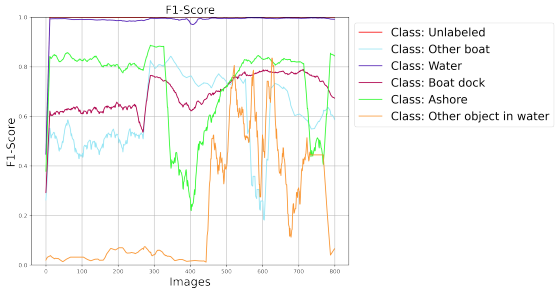
Figure A.3: F1-score and precision for test c



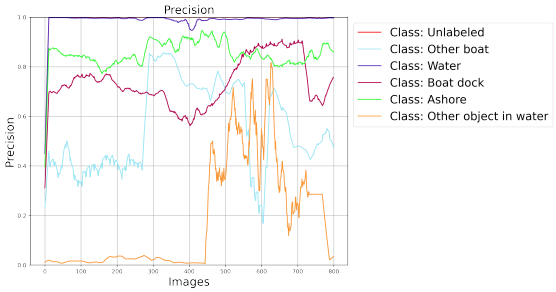
(a) F1-score test d

(b) Precision test d

Figure A.4: F1-score and precision for test d



(a) F1-score test e



(b) Precision test e

Figure A.5: F1-score and precision for test e

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY