CHALMERS
UNIVERSITY OF TECHNOLOGY

Detection and classification of marine vehicles

Master's thesis in Computer Science and Engineering

ATHANASIOS ROFALIS

Department of Mechanics and maritime sciences

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

Master's thesis 2021

Detection and classification of marine vehicles

ATHANASIOS ROFALIS

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mechanics and maritime sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
Gothenburg, Sweden 2021

Detection and classification of marine vehicles
ATHANASIOS ROFALIS

Supervisor: Ola Benderius, Department of Mechanics and maritime sciences
Examiner: Ola Benderius, Department of Mechanics and maritime sciences

Master's Thesis 2021:82
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: Detection and classification of marine vehicles.

Typeset in LaTeX
Printed by Chalmers Chalmers Digitaltryck
Gothenburg, Sweden 2021

Detection and classification of marine vehicles
ATHANASIOS ROFALIS
Department of mechanics and maritime sciences
Chalmers University of Technology

Abstract

One of the most common tasks within the computer vision field is the detection and classification of different objects. This thesis aims to deliver a software that can be deployed into real world scenarios and mange to detect and classify marine vehicles accurately. Using one of the pre-defined deep neural network models *You look only once (YOLO)*, we managed to achieve a high performance for the detection and classification task. The training of the model took place using a specific dataset of grayscale images, which led to a model that can classify the objects with an accuracy of 68% and predict the relevant position with mean average precision (mAP) of 0.77. Moreover, the model tested into different weather conditions and achieved an accyracy of 0.85% and mAP of 0.068. In general, the YOLO model seems to be a robust detector that can be trained and deployed for detecting efficiently objects with high performance.

Keywords: Classification, detection, deep learning, computer vision, YOLO

Acknowledgements

First of all I would like to thank everyone who supported me during my master's thesis and gave me the motivation to continue. Secondly, I feel that I need to express my deepest acknowledgement to my supervisor at Chalmers university Ola Benderius, who guided me during these months and supported me whenever I felt overwhelmed. Moreover, it is important to thank everyone worked on the Revere's autonomous ships project and managed to collect the data that was used for this thesis.

Athanasios Rofalis, Gothenburg, May 2021

Contents

List of Figures

List of Tables

1

Introduction

Self-driving vehicles are considered the future while the traditional driving style largely might be eliminated. During the last decades many companies have included in their massive production vehicles advanced driver-assistance systems, like the change line system or the emergency stop. Moreover, the Tesla corporation was the first company who made the next step and made the autonomous driving technology accessible to everyone by producing a car capable of autonomous driving in certain driving scenarios.. The software of the car is a complex achievement which using cameras and other sensors is able to identify objects while calculating the trajectory that the car needs to follow in order to protect the passengers. Therefore, the detection and classification of different objects is necessary. That task can be handled by the combination of automotive engineer and computer vision.

1.1 Motivation

Modern computer vision techniques are able to solve complex tasks in a short period of time and also yield results that classic methods were not able to achieve. During the machine learning era, algorithms have been implemented to simplify the computer vision task of a system which nowadays does not require expensive hardware and complex software. Managing to establish a model that is stable and accurate the system might be able to prevent accidents while evolving the driving experience. There are machine learning algorithms that are able to solve the classification and detection task accurately, however it is important to ensure that it performs at its highest potential by evaluating the model after its training. A worth mentioning problem concerning machine learning is that it does not detect and classify objects that it has not seen before. That issue is called interpolation and extrapolation problem and it is one of the drawbacks of machine learning. This means that the model needs to be executed in scenarios that was trained for. For instance, a traffic is an example of a mathematically complex scenario which by nature is unpredictable, therefore for an algorithm that is not trained based on that situation might perform poorly and could cause an accident by not avoiding other objects. That problem is considered a reason machine learning has not been deployed for autonomous driving vehicles yet.

1.2 Goal

Part of the excessive research on the autonomous driving field is the vehicle laboratory Revere at Chalmers university of technology in which theoretical and current algorithms are implemented and deployed into autonomous vehicles. This thesis aims to establish a software that will be able to solve the object classification and detection task specifically for an autonomous boat. The software will be consisting of a machine learning model trained for detecting marine vehicles using monochrome cameras placed on the boat.

1.3 Research questions

This thesis aims to presents some results regarding a detection and classification task. A method that can be implemented in order to present some results is by deploying deep learning models using annotated data, so that the model becomes as accurate and robust as it can be. The data, which is monochrome images, is collected in a certain weather condition, therefore images with weather diversity, for instance blurry images due to fog or rain, could affect the model's performance and robustness. Thus, we need to ensure that the model is trained properly in order to minimize potential false detections while maximizing its accuracy into different weather conditions. Furthermore, since the model will be used in autonomous vehicles it is essential to make sure that the algorithm can be structured in a way that can be executed on the vehicle's processor while it is still possible to quickly update the software. Therefore, we introduce the *continuous integration and continuous deployment (CI/CD)* pipeline which is a way of automatically testing and deploying an algorithm.

Considering the aforementioned, there are three research questions that this thesis will try to give an answer to. Those research questions are:

- How much annotation data is needed to achieve high performance, with at least 90% correct classifications and an acceptable level of false detections, in detection and classification of at least five different ship classes?
- How well does the network behave in different weather conditions given that it was trained only on data with no rain, snow, fog, or wind?
- How can the algorithms be structured based on the microservice architecture and CI/CD pipelines?

1.4 Limitations

Considering the research questions mentioned above, we will try to answer them specifically for the autonomous ship and deploy the model into real case scenarios. The algorithm was not tested in different weather conditions or during night time. The machine learning model in this project was constructed specifically for the boat at Chalmers Revere lab and it used real case scenarios from this boat through the

learning process. Specifically, the data was gathered at Göta älv by a monochrome camera installed on the boat used for the project.

1.5 Outline

For this thesis project a software was developed, for deploying the machine learning model, running alongside the OpenDlv software, that was implemented by the researchers at Revere lab, which is responsible for processing and transferring the data to the next microservice. The main part of the software is a robust machine learning model that was trained, deployed and tested for a maximized performance. It was also trained for a fast and accurate detection and classification of marine vehicles.

2
Background

The idea of autonomous ships was presented in the early of 1970s by Rolf Schonknecht who claimed that one day ships will not be controlled by a captain onboard, instead they will be given directions from someone on land. As a first step in this direction, autopilots on boats were being installed during the last decades of the 20^{th} century. Excessive research then started off in 2011 with a project at *Korea research institute of ship and ocean engineering* (KRISO). A year later (2012), the European Union launched a similar project. These projects fueled this field of research, and governments, such as Norway, as well as different corporations started to invest in similar projects, Lloyd's register and Rolls–Royce to mention a few. Nowadays, there are many active on the field of autonomous ships with promising results. For instance, Rolls–Royce marine has announced in 2018 that they have an autonomous ship ready to be deployed at sea by 2025. Likewise, the Japanese consortium have made public that they will also present an autonomous ship that same year [4].

The purpose of this thesis project is, as mentioned in previous sections, to detect and classify marine vehicles using installed cameras on boats. However, part of the work has also been to take a closer look at what has been studied and tested recently. There are many publications regarding object detection combining computer vision and machine learning from the foregoing years. The next section describes a few of the methods used in recent studies.

2.1 Object detection using tubelets

The research paper 'object detection from video tubelets with convolutional neural networks' has investigated a way of making object predictions on videos. This specific inquiry was based on the 'large scale visual recognition challenge 2015 (ILSVRC2015)' and the ImageNet VID dataset. According to the authors, the proposed method can solve the task of object detection accurately and can be deployed on videos.

The model that they used was a deep neural network but the predictions of the potential object's position differ from the anchor box technique which is widely used. The first part of the algorithm was something the authors called a 'spatio-temporal tubelet proposal'. This approach means that a line parses the image and tries to identify objects that might appear. A score-based algorithm was implemented in order to predict the position of the objects. Consequently, they deployed the GoogLeNet

and AlexNet models for calculating probabilities for each object while deleting proposals with small probability. However, due to a high variance, they included two additional steps into the model to minimize overfitting and achieve a higher performance. Following that, they deployed max-pooling and reinstated the predicted bounding boxes during the tubelet procedure. Lastly, a four layer 1-D CNN, called 'temporal convolutional network (TNP)', was trained and yield the decision if the predicted bounding box overlaps the ground truth above 50% [5].

2.2 Sea surface analysis for ship detection

A different approach was published in 2018, treating ship detection by analysing the sea surface. The data used for the model was high quality images which were captured by a satellite.

The most important part of this model is how it analyses the sea surface. Due to the fact that images are taken by a satellite, different weather conditions might affect the outcome of the images. For instance, fog and clouds can give blurry images, and wind naturally creates waves which can make the images distorted. Therefore, they created a mathematical equation based on the number of pixels. The model that they implemented and trained was a support vector machine (SVM) model able to classify shape features. The proposed method called SDSSA, was able to remove objects smaller than fifty pixels including the false detection using the compactness and length-width ratio of the object. However, a key point of the method occurs before the removal of false predictions, when the model creates a list of potential ship candidates using a mathematical equation based on the pixels and the area of an object. After completing these three steps the remaining candidates are the detected objects that the algorithm outputs [6].

2.3 Dataset for ship detections

During 2018 a dataset was published and it consists of 31,455 images of six different classes of marine vehicles. The name of the dataset is SeaShip and compared to other datasets, like VOC2007 or CIFAR-10 the size of that particular dataset is larger while the resolution of the images is also higher, specifically 1920x1080. The images were extracted from surveillance videos from cameras on different locations in order to have a diversity of the background and the weather conditions.

Although, the main purpose of the SeaShip paper was to present the dataset and the annotation procedure, the authors went a step further providing also some baselines and comparisons of three 2018 different state of the art models. These models were the faster region based convolutional neural network (faster R-CNN), the you only look once (YOLO) and the single shot multibox detector (SSD). The metric that was used to evaluate the models' performance were the mean average precision (mAP) and the frames per second (FPS). The model with highest mAP was the faster R-CNN with almost 0.930 and the model with the highest FPS was the YOLO with

91 [7].

2.4 Adversarial convolutional network

The region based convolutional neural network (RCNN), is a widely used model for detection and classifictation of objects. Although, it is a robust model with average performance, there have been many researches trying to improve it. The authors proposed two methods in order to optimize the performance of the Fast-RCNN (FRCNN), which is a faster version of the RCNN model. Specifically, like data augmentation techniques, these methods are able to change the images slightly in order to train the FRCNN model into more difficult images and consequently improve its performance.

These models are the Adversarial Spatial Dropout Network (ASDN) and the Adversarial Spatial Transformer Network (ASTN). The ASDN is a model which removes parts of the images and feed them as input to the FRCNN model which predicts the position and class of a potential object. The ASTN model reforms the image, by changing the position of some blocks of the images, and again feed to FRCNN for classification and detection. After training the models using these techniques the authors then tested them on the VOC2007 dataset and compared them to the standard FRCNN model using the mean average precision (mAP) metric. Specifically, the ASTN model achieved 58.1 and the ASDN 58.5 and combing them into one model gave 58.9, while the FRCNN managed to score 55.4 [8].

2.5 Integration and delivery

Modern software development is not only focused on implementing robust techniques, but also aims to support the development and deployment itself. Therefore, continuous integration and delivery techniques have recently been explored. A combination of software developement and IT operations is the *development and operations (DevOps)* and is a method for shortening the development's cycle and delivering software of higher quality.

The DevOps process can be divided into five phases: 1) continuous planning, 2) integration, 3) deployment, 4) testing, and 5) monitoring. The planning includes continuous communication with the customer adapting at the same time the software. One important part of the whole procedure is the deployment in which different hardware systems are used to test the integrated software. During the last two phases the software goes through testing by the developers in order to detect possible bugs that could affect the efficiency of the software. Moreover, every step in the DevOps process is executed completely automatically, even though one step relies on the previous one. The advantages of applying such a method are that the development's cycle can be shortened and also there is no need to run the software on different expensive hardware systems, since those have been replaced by the cloud computing systems [9].

3
Methods

The purpose of this thesis is to establish a software that can classify and detect marine vehicles. However, the procedure of software development needs to be divided into three parts, annotation, training and execution. The theoretical aspect of each of these steps can be grounded without any input of the previous one. However, during the implementation of the software every step relies on the previous one for achieving the highest potential performance of the model.

3.1 Annotation

In supervised learning the dataset, which is used to train the machine learning model and accomplish its task, is a fundamental part of the training procedure. In general, data annotation is considered to be an underlying part of machine learning, since it is the way of establishing the ground truth for training and testing. However, in order to maximize the performance of the model the dataset needs to fulfill two requirements. Firstly, it needs to be balanced, and secondly, properly annotated.

A balanced dataset is consisting of approximately the same number of data for the different classification classes. For instance, if the model tries to solve a binary classification of images then we need to make sure that the dataset has an equal number of images for each of the two classes. Since the main task of a machine learning model after the training process is to achieve maximum performance, imbalanced datasets are affecting negatively its goal and conclude to a model that makes predictions based on a non-uniform distribution [10]. However, some techniques can be deployed and secure a balanced dataset. Two of the most common techniques are the *simple random sampling (SRS)* and the *trial-and-error methods*. According to the SRS technique, the sampling of the dataset is done randomly following the uniform distribution. In the trial-and-error methods, the SRS method is performed multiple times and afterwards the results are averaged [11]. To create and establish a dataset that will be well-formed and able to be deployed to train this project's model efficiently, we are going to use the SRS technique managing to establish a balanced dataset.

Data annotation is the procedure through which the data is composed into input-output format, which is going to be used for the training of the model. In classification and object detection problems, the dataset is important to have the format of a vector `object-class,x ,y, w, h` where object-class is the type of the object.

This (x, y) is the starting point of the bounding box followed by the w, h, which are the width and height respectively, used to calculate the other three coordinates of the remaining corners of the box.

The data was captured by a monochrome camera installed on the boat, and was stored into .raw format files with resolution 3208 × 2200 and bit depth 10. However, the .raw images were scaled down and stored into .png files with resolution 1920x1200 and bit depth of 16 bits. Specifically, a microservice was implemented using the OpenCv framework to access the .raw images and then produce the required .png images. Using that open-source software *labelImg* the images properly annotated into `object-class,x ,y, w, h` vector, which is the annotation format that the thesis model supports.

3.2 Training

The training of the model is the part where a deep neural network model learns through experience how to solve specific tasks, which in our case is to detect and classify marine vehicles. Although the training is a fundamental part of the whole process, the architecture of the neural network is not the same for each task. There are many algorithms and models that are able to solve the detection and classification task efficiently, like the *region based CNN (R-CNN)* [12] or the *single shot multiBox detector (SSD)* [13]. However, the *you only look once (YOLO)* is a model that outperforms most of the detectors and this is the reason why it was decided to use this model. The YOLO model is already trained into a specific dataset, but since we need it to be applicable to our marine vehicles dataset, the training will be based on the dataset we have collected. Despite the fact that the architecture of the model remains the same, it is essential to change some of its parameters. As it is presented in Figure 3.1, the process is split into two parts with the first being the model's selection and dataset's establishment. After forming the dataset, we need to annotate it and use data augmentation techniques, like blurring or changing the brightness of the images, to extend its size. Afterwards, we train the selected model into the dataset multiple times, while fine-tuning its parameters for maximum performance. Whenever the model reaches the highest of its performance, we export and deploy it.

The YOLO model is using the bounding boxes technique over the image. Initially, the model splits the image into SxS grid cells. For each cell the algorithm makes predictions about possible bounding boxes while computing probabilities, named as confidence scores. These scores represent the likelihood of each potential box to have been placed correctly and contain an object. During the testing process, the model multiplies these probabilities with the confidence score for each of the classes, and selects the highest one [14].

However, the developers of YOLO wanted to improve it, consequently they created an updated version that is faster and more accurate than the previous one. Therefore, they presented the YOLOv2 by making changes compared to the first version.

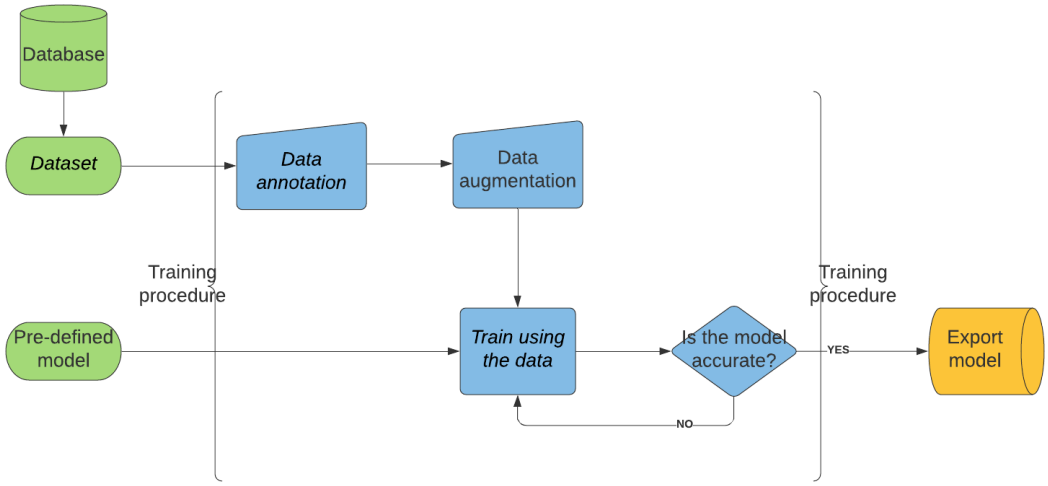


Figure 3.1: The learning procedure describing the steps of for the training process.

First of all, they doubled the input size of the images and instead of 224x224, which is the size of YOLOv1, they used 448x448 to train the classifier on the ImageNet dataset. Also, they changed the way of making predictions of the bounding boxes and instead they used the anchor box technique. According to that method, the model can generate bounding boxes longer than the grid cells and also link multiple objects to them. Moreover, they added batch normalization to every convolutional layer of the model which lead to improvement of the performance. Furthermore, the YOLOv2 is able to detect more than 20 classes which is the limit of classes that YOLOv1 can detect. While there is great demand for fast detectors, like in autonomous driving concepts, YOLO's authors implemented an improved model that was both robust and fast. Thus, instead of establishing a deeper model they reduced the number of layers. The YOLOv1 consists of 24 convolutional layers, while the YOLOv2 has 5 less, id est 19 layers, and the name of the model's architecture is Darknet-19 as it is shown in table 3.1. That way, they managed to establish a faster version of YOLO while the robustness remains untouched [15]. In this thesis the second version of YOLO is going to be used since it combines the robustness and the speed that a detector and classifier needs. Furthermore, because the model's architecture is not large, iT could be faster during the training procedure especially when the training process is performed on a single GPU.

Additionally, the YOLO developers presented the YOLOv3 model which is a larger deep neural network model which gains in performance. The model's architecture is called Darknet-53 and consists of 53 convolutional layers. The model is able to detect smaller objects compared to the objects that YOLOv1 and YOLOv2 can detect. An important improvement is that YOLOv3 is able to make multi labels predtictions meaning that an object can be classified into its class but also it will be categorized into a broader class that it belongs [16]. However, the researchers continued their work and presented the YOLOv4 which is the latest version of YOLO. According to their results, the YOLOv4 model has higher performance regarding the detection

and localization part of the task but also is faster which means that it has higher FPS rate and allows the model to make predictions accurately in shorter period of time [17].

Darknet-19			
Type	Filters	Size/Stride	Output
Convolutional	32	3x3	224×224
Maxpool		2×2/2	112x112
Convolutional	64	3x3	112x112
Maxpool		2×2/2	56x56
Convolutional	128	3x3	56x56
Convolutional	64	1x1	56x56
Convolutional	128	3x3	56x56
Maxpool		2×2/2	28x28
Convolutional	256	3x3	28x28
Convolutional	128	1x1	28x28
Convolutional	256	3x3	28x28
Maxpool		2×2/2	14x14
Convolutional	512	3x3	14x14
Convolutional	256	1x1	14x14
Convolutional	512	3x3	14x14
Convolutional	256	1x1	14x14
Convolutional	512	3x3	14x14
Maxpool		2×2/2	7x7
Convolutional	1024	3x3	7x7
Convolutional	512	1x1	7x7
Convolutional	1024	3x3	7x7
Convolutional	512	1x1	7x7
Convolutional	1024	3x3	7x7
Convolutional	1000	1x1	7x7
Avgpool		Global	1000
Softmax			

Table 3.1: The YOLOv2 model architecture.

During the learning procedure, the model needs to be evaluated multiple times after each training in order to have an insight on the performance of the model. For this project the evaluation is divided into two parts. Firstly, we need to evaluate how the classifier performs and secondly, how the detector does. Thus, we need to use three different metrics, percentage accuracy of correctly classified objects, percentage of accuracy of missclassified objects, and the mean average precision (mAP) respectively. Since the dataset is annotated, we know in advance the optimal result that the model needs to predict. However, in order to compute these metrics we need to define the precision and recall. The mathematical expression of precision is

$$P = \frac{TP}{TP + FP}$$

and recall is

$$R = \frac{TP}{TP + FN}$$

where TP = True Positive (predicted as positive and it was correct), TN = True Negative (predicted as negative and it was correct) FP = False Positive (predicted as positive but it was wrong) and FN = False Negative (Predicted as negative but it was wrong). Therefore, the accuracy of the correctly classified objects is:

$$A_c = \frac{TN + TP}{TN + TP + FN + FP}$$

and the accuracy of the missclassified objects is:

$$A_m = \frac{FN + FP}{TN + TP + FN + FP}$$

However, in object detection tasks the mAP is determined by the intersection over union (IoU) which is the

$$IoU = \frac{\text{AoO}}{\text{AoU}}$$

, where AoO is the area of overlap which is the percentage of the predicting box that overlaps the ground truth and AoU is the area of union which is the union of the predicted and ground truth boxes. Setting up a threshold we calculate the IoU for the predicted bounding box. If the IoU is bigger than the threshold then we classify the prediction as TP otherwise as FP. Repeating this process for every class in the dataset we compute the average precision which is the area under the precision–recall curve. For the mAP we just compute the average of the AP of each class in our dataset.

However, in order to report any results for the second research question we need to handle the images in a specific way. As it was aforementioned, the data collection will not be conducted on different weather conditions. Therefore, the images will not have a weather diversity, but it is important to establish a method for evaluating the trained model on different weather conditions. Accordingly, a script has been implemented in order to create images with different light exposure, snow, rain or fog conditions. Since these images are not part of the regular evaluation of the model, we are going to use the same metrics, namely accuracy for classification and mAP for detection. It is important to say that even though we are making changes to the images, the ground truth remains the same, because the images are not going to be rotated at all. Thus, the annotation made in the previous step has not changed and those files can be used as a way of evaluating the model. The script that was used to create the diverse images can be found on the Appendix 1.

3.3 Model's execution

At this point, two of the main parts of the project have been successfully implemented and tested in order to make sure that the software is ready for the deploying

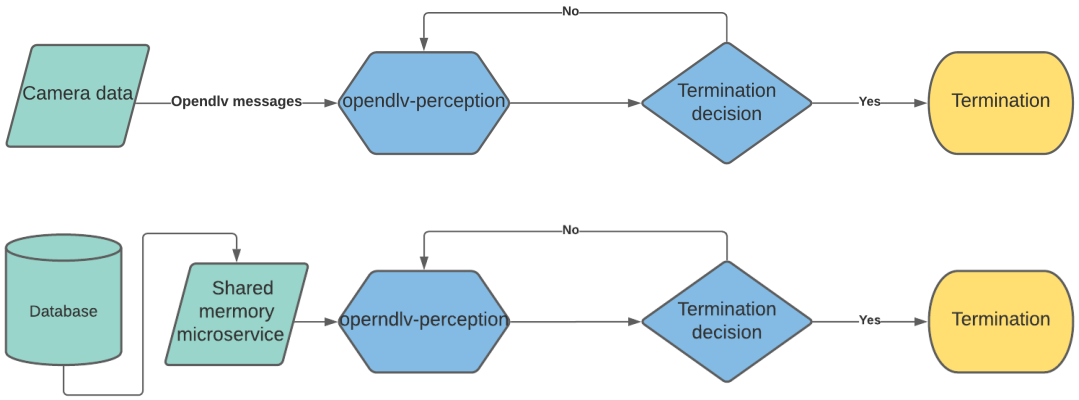


Figure 3.2: The model's deployment receiving data either directly through camera or extracted from a database.

procedure. According to this step, we need to make sure that the implemented software is applicable to the autonomous boat and there are no conflicts between and the software and the hardware. Hence, we use the software development technique CI/CD, which allows us to test if the software is applicable to the platform without actually deploying it.

For the CI/CD part of the process we use a docker file that allows us to integrate the software into a specific operating system with specific requirements. In our project, since the machine learning model requires GPU for fast execution, we use as a builder for the docker container the Linux Ubuntu distribution, specifically the 20.04 version while enabling the software responsible for executing the model using Nvidia Cuda. Since this docker image is a fresh operating system that lives into a docker container, we need to install the necessary packages needed for our software. Afterwards, using a cmake file we build the software into the container. The code is compiled and tested for any kind of conflicts, like non-installed packages. In this project there are two major programs that are important for this part and those are the Libcluon and the program that runs the model for detection and classification. The fist one is the software that is used for the communication between components in a distributed software system and the other one is the program that it is able to execute the YOLO model into the frames shared through the cluon software.

As it is shown in the Figure 3.2, the process of the model's deployment can be performed using two different methods. The first one is to deploy the model into direct data captured by the boat's sensors and the second one is through the shared memory of a system using images extracted from a database. In the first case the camera captures frames while the OpenDlv software handles the data and transfer it into the opendlv-perception microservice that executes the YOLO model which returns the potential detected objects. In the deployment through the shared memory the stored images are parsed into the opendlv-perception microservice which executed the YOLO model while making predictions for potential marine objects and their

relevant position. In both cases the procedure continues until the user stops the data input from camera or database to the microservice respectively.

The evaluation of the model into new data is a complex task because it requires the annotation of the new images. Similarly to the Zooniverse website [18], a web site can be implemented in order to extract the files from the server and using an online service, annotate the images and save the annotation files. Using the shared memory technique the model can be trained into the new data and also be evaluated.

4
Results

This thesis aims to give a comprehensive answer to the research questions specified in the introduction section. After the training of the model and the multiple evaluations of the model, we managed to reach to the following results.

The first research question is about evaluating the trained model and give a comprehensive answer about the number of images we need in order to achieve the potential highest classification and detection of the different classes. The dataset that we designed and annotated was consisting of 1,000 images. The dataset was split into two sets, the training and the test set which were used to train and evaluate the model respectively. At the Table 4.1 are presented the performance results of the model after the training.

IoU threshold = 0.25			
	A_c	A_m	mAP
Training Set	68%	31.86%	0.75
Test Set	68%	31.87%	0.77
IoU threshold = 0.50			
	A_c	A_m	mAP
Training Set	64%	35.5%	0.63
Test Set	68.1%	31.87%	0.68

Table 4.1: YOLO evaluation into the training and test set using two different thresholds for the IoU.

The second research question is about evaluating the model into different weather conditions. Since the model was trained into a dataset with no weather diversity, the model was evaluated using both the training and the test sets in order to get a better insight on how the model actually performs. At the Table 4.2, the results of the evaluating procedure are presented and it is clear that the model achieved lower scores compared to the original sets without weather conditions diversity.

The last question is about an optimal design of the algorithm giving the opportunity to an external user to utilize the microservices and the model that we trained. A specially designed webservice platform will host the implemented microservices and through that site external users will have access to the collected data. As it is shown

IoU threshold = 0.25			
	A_c	A_m	mAP
Training Set	18.4%	81.59%	0.1506
Test Set	0.85%	91.48%	0.1106
IoU threshold = 0.50			
	A_c	A_m	mAP
Training Set	16.2%	83.75%	0.107
Test Set	0.7%	92.90%	0.068

Table 4.2: YOLO evaluation into the training and test set on different weather conditions using two different thresholds for the IoU.

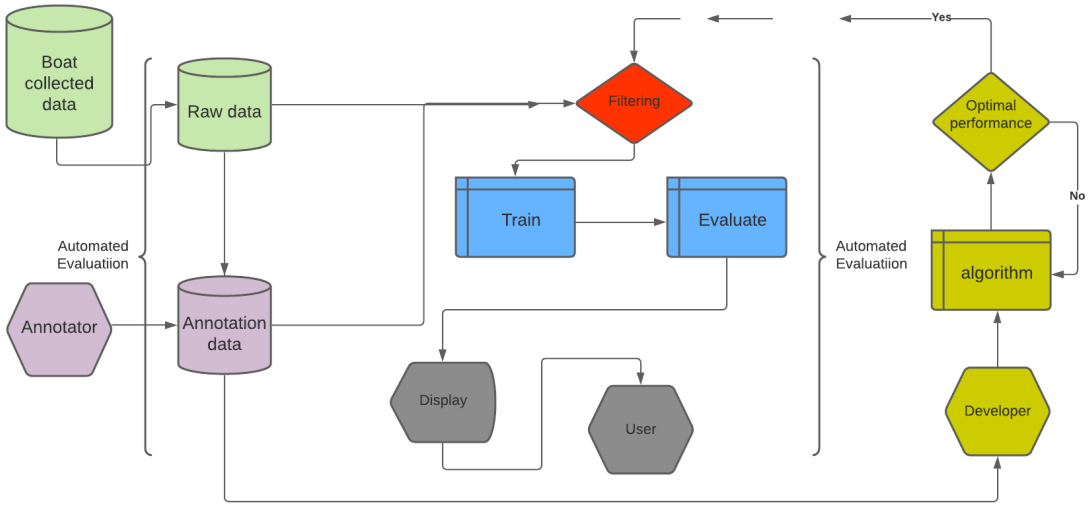


Figure 4.1: Automated evaluation process including the three different type of users and the interaction with the web-service.

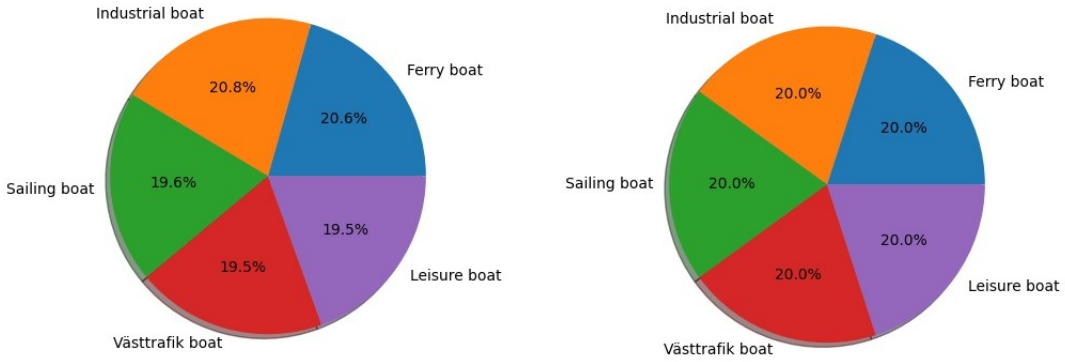
in Figure 4.1 initially the data are stored into a database. From that database we extract data which will be used for running the model. If we have the data annotations needed as ground truth then we can run the microservice instantly and report the results to the user. However, if the annotations are not available, we need to follow a different approach. In that case, an external researcher needs to annotate manually data that will be stored and used for the training and evaluation. But, we need to make sure that the annotations are correct which is accomplished through the filtering procedure. For instance, we do not gather data that have been annotated once but we select data that are roughly similarly annotated by more than annotators. The next step is to train the model using the configuration, that the developer has decided as the optimal one, evaluate the model and just display the results to the external user alongside with the results of the other algorithms that will be executed using the same data.

5
Discussion

The results indicate that the model is able to solve the task of classification and detection of marine vehicles. The first two questions are relevant to a machine learning model and specifically about its performance. Those questions are about the evaluation of the classification and object detection of marine vehicles based on data collected for Chalmers Revere's ongoing project related to autonomous ships. Additionally, there is a question regarding a way of establishing a web service that users will have access to the collected data and either use the algorithms and get some results or develop one and deploy it. More specifically regarding the first two questions, the model managed to achieve an accuracy of 68% for the classification part on the test set and a mAP of 0.77 while on the training set the accuracy was 68% and mAP of 0.75. Concerning the last question we gave the theoretical baseline for developing the required website and how the external users and researchers could use the platform.

The dataset used for the training of the model consists of 1,050 grayscale images with resolution of 1920x1080 and bit depth 10-bit. The dataset is split into two sets: 1) the trainig set that contains 1,000 images and the annotation files and 2) the test set which encloses 50 images with the annotated files which is around 5% of the initial dataset. The training set is used to train the model and the test set is for evaluating its performance. Both of the datasets are balanced, since they contain approximately the same number of objects for each of the five classes as it is shown in Figure 5.1. Moreover, the sets were established in a way that there are no common images. That way the evaluation of the model is more accurate and we can ensure that the testing of the model is performed on images that the model has no access before.

The training of the model is a procedure that requires many repetitions while changing and adapting the parameters of the model in order to get its highest performance. However, we managed to train a model that is able to solve our task efficiently and make accurate predictions. Moreover, the model was initially trained for 1,500 epochs with input size of the images to be 512x512 and managed to achieve accuracy of correct classification 54.65% and mAP of 0.576 on the test set. Furthermore, the training continued for additional 1,000 epochs with input size of 608x608. It is worth mentioning that the model was trained on a singe GPU, specifically NVIDIA GeForce RTX2060, for approximately 15 hours. For the evaluation part of the performance the configuration changed and used the input size of 896x896 which allowed the model to achieve the presented results. The input size was increased



(a) Training set **(b)** Test set

Figure 5.1: Percentage of objects' numbers for each class.

following the instructions of YOLO's authors mentioning that it is important to increase the input size of the model after its training. Additionally, the model was trained for another 500 epochs with the input size of 640x640 but it did not manage to outperform the performance of the previous configuration. However, even though the training of the model is considered to be short, YOLO managed to achieve remarkable performance. Specifically, with the IoU threshold set to 0.25 the model managed to classify correctly 68% of the objects and have a mean average precision of 0.77 meaning that it detected the objects but it was not able to create the bounding boxes as the ground truth of the annotation did.

On the other hand, the YOLO model did not manage to have a robust performance when it was deployed into images on different weather conditions. That outcome is rational considering that the model was not trained into images with weather diversity. Since the model is a deep neural network and needs data to be trained efficiently, if the dataset consisted of images into different weather conditions, it could be possible to have a better performance. That is an issue that any machine learning algorithm needs to face. The extrapolation and interpolation issue which means that an algorithm is trying to solve a task that is harder than the one that was trained for. In our case interpolation is the detection and classification of marine vehicles into images with certain weather conditions, the task that the model is trained to solve. The extrapolation is the harder version of the task which is the detection and classification of marine vehicles into different weather conditions. Since the deep neural network model was not trained to solve the task into images with weather diversity, it performs poorly. The model tries to somehow distinct the weather conditions from the images and make accurate predictions about the marine vehicles appeared. That task is not easy to be solved by the model since it was not trained on it,a fact that proves that the extrapolation problem remains which is one of the drawbacks of machine learning.

5.1 Ethical perspective and data management

Nowadays data is considered to be the new oil. Data is collected, stored and handled in order to extract information which trespasses in some cases the user's privacy right. Therefore, it is important to secure that data has some security protocols and no unauthorized users have access to it, but also it does not contain sensitive personal information. In May 2018 EU has enacted the General Data Protection Regulation (GDPR) to establish a legal background for data protection. Moreover, the GDPR sets some boundaries on the collection, storage, and process of data and specifically regulates that data must not contain information that could connect it with the user that has been collected from [19]. The data that this thesis processes is images collected by a marine vehicle and it might not contain sensitive information, such as names, national ID numbers. However, the images of other boats can enclose information that could eventually be personal. Consider an image of a boat. Directly from the image, someone could extract the boat's name and registration number, information that is sufficient enough to search the owner of the ship through Sjöfartsverket, the Swedish Maritime Administration.

Following the GDPR legislation, we need to secure that there are no violations. As mentioned above about the design of the web service, there are three types of users: the developer, the annotator, and the user. It is important to make sure that each and every one of them complies with the GDPR and personal information laws. However, the external user is not able to download and process the data because he uses the web service waiting for the display of the results of the algorithms and has no direct access to the data. Following that the annotator has no access to the data since the annotation will be done online and no data will be downloaded. On the other hand, the developer has access to the data, can download and process it because the algorithm that is going to be implemented needs to be tested locally for maximum performance. In that case, the GDPR law takes immediate effect. Before even downloading any data the developers need to sign a contract stating that there will not be any data leaks and no data will be stored after the implementation of the algorithm.

6

Conclusion

The task of detection and classification of objects is considered to be a traditional task in the field of computer vision. Due to the excessive research on deep learning, there are many models that can be trained and deployed for this particular task. This thesis aims to detect and classify marine vehicles using data collected by monochrome camera on a boat which is part of the Chalmers Revere's ongoing project on autonomous ships. Moreover, the training of the model is not the only task that the thesis addresses. Additionally, the thesis proposes a system that integrates the model and training into an end-to-end data process.

Following the presented method, the predefined deep learning model, YOLO, managed to classify the required objects with accuracy of 68% and detect the correct bounding boxes in a percentage of 77%. However, the model was not able to achieve the goal of the correct classification of at least 90% and have on the same time a decent percentage of incorrect classifications. Additionally, a software for deploying the model into real world scenarios was implemented providing a tool to test the model and its performance into real time detections. Moreover, the thesis presented the theoretical baseline for implementing a public web service in which users and researchers will have access to the model and data while testing several implemented algorithms into the same data.

The dataset that was used for the training of the model consists of 1000 images, a number that could be increased in a future work since the is adequate size of data that images can be extracted from. Also, the training of the model is considered short due to the fact that it was trained on a singe GPU, specifically NVIDIA GeForce RTX2060, for 15 hours. However, with a larger dataset and a longer training the model could be more robust and accurate while achieving higher performance compared to the presented one. Additionally, the model could be able to reach the goal of at least 90% accuracy on the classification task while minimizing the wrong classification percentage. Additionally, the model could be able to achieve higher mAP percentage regarding the object's detection which is related to the prediction of the object's relative position. Correspondingly, the model's performance on the dataset consisting of images with weather diversity is considered poor. That comes to prove the already mentioned issue of machine learning models about interpolation and extrapolation, since both the accuracy of the classification and the mAP percentage are low, which means that the model is not able to solve its task whenever the are weather conditions different than the ones trained on.

According to this thesis, a baseline has been established for the detection and classification of marine vehicles related to the Chalmers Revere's project on autonomous ships. Handling data captured by a monochrome camera a deep neural network model has been trained, providing some results on the classification and detection of marine vehicles task. Moreover, if future work conducts following the proposed recommendations the model could become more accurate and robust. In conclusion, this current work hands over the process that can be followed in order to establish a dataset using the captured images, train the model and eventually deploy it using a software into microsevice architecture.

Bibliography

[1] Schmidhuber, Jürgen. “Deep Learning in Neural Networks: An Overview.” Neural Networks 61 (2015): 85–117. Crossref. Web.

[2] Shahin, Mojtaba, Muhammad Ali Babar, and Liming Zhu. "Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices." IEEE Access 5 (2017): 3909-3943.

[3] Wang Qi, Fu Li and Liu Zhenzhong, "Review on camera calibration," 2010 Chinese Control and Decision Conference, Xuzhou, 2010, pp. 3354-3358, doi: 10.1109/CCDC.2010.5498574.

[4] R. (2018, August 31).Timeline–Development of Autonomous Ships (1970s – 2018). Infomaritime.Eu.http://infomaritime.eu/index.php/2018/06/08/timeline-development-of-autonomous-ships/

[5] Kai Kang, Wanli Ouyang, Hongsheng Li, Xiaogang Wang; Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 817-825

[6] G. Yang, B. Li, S. Ji, F. Gao and Q. Xu, "Ship Detection From Optical Satellite Images Based on Sea Surface Analysis," in IEEE Geoscience and Remote Sensing Letters, vol. 11, no. 3, pp. 641-645, March 2014, doi: 10.1109/LGRS.2013.2273552.

[7] Z. Shao, W. Wu, Z. Wang, W. Du and C. Li, "SeaShips: A Large-Scale Precisely Annotated Dataset for Ship Detection," in IEEE Transactions on Multimedia, vol. 20, no. 10, pp. 2593-2604, Oct. 2018, doi: 10.1109/TMM.2018.2865686.

[8] Xiaolong Wang, Abhinav Shrivastava, Abhinav Gupta; Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2606-2615

[9] Virmani, Manish. "Understanding DevOps & bridging the gap from continuous integration to continuous delivery." Fifth international conference on the innovative computing technology (intech 2015). IEEE, 2015.

[10] Provost, Foster. "Machine learning from imbalanced data sets 101." Proceedings of the AAAI'2000 workshop on imbalanced data sets. Vol. 68. No. 2000. AAAI Press, 2000.

[11] Reitermanova, Z. (2010). Data splitting. In WDS (Vol. 10, pp. 31-36)

[12] Girshick, R. (2015). Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).

[13] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). Ssd: Single shot multibox detector. In European conference on computer vision (pp. 21-37). Springer, Cham.

[14] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
[15] Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017
[16] Redmon, J., Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.
[17] Bochkovskiy, A., Wang, C. Y., Liao, H. Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934
[18] Zooniverse. 12 Dec. 2009, www.zooniverse.org/
[19] N. Gruschka, V. Mavroeidis, K. Vishi and M. Jensen, "Privacy Issues and Data Protection in Big Data: A Case Study Analysis under GDPR," 2018 IEEE International Conference on Big Data (Big Data), 2018, pp. 5027-5033, doi: 10.1109/BigData.2018.8622621.

A

Appendix 1

The script that it was used to reform the images by changing the brightness, blurring them or adding snow or rain drops.

```
import cv2
import numpy as np
import glob
import random
import os
import argparse

from numpy.lib.type_check import imag


ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image",required = True, help = "path to images")
args = vars(ap.parse_args())

inputFolder = os.path.sep.join([args["image"]])
folderlen = len(inputFolder)

for img in glob.glob(inputFolder + "/*.png"):

    image = cv2.imread(img, cv2.IMREAD_GRAYSCALE)
    randomWeather = random.choice(["Brighter","Darker","Rainy",
                    "Snowy","Fogy"])

    if randomWeather == "Brighter" :

        randomBrightness = random.randint(10,80)
        bright = np.ones(image.shape, dtype = "uint8")*randomBrightness
        brightIncrease = cv2.add(image,bright)
        brightIncrease = ((brightIncrease + 1 ) * 256 ) -1
        cv2.imwrite(inputFolder + img[folderlen:],brightIncrease)

    elif randomWeather == "Darker" :

        randomDarkness = random.randint(10,80)
```

```
        bright = np.ones(image.shape, dtype = "uint8") * randomDarkness
        brightDecrease = cv2.subtract(image,bright)
        brightDecrease = ((brightDecrease + 1 ) * 256 ) -1
        cv2.imwrite(inputFolder + img[folderlen:],brightDecrease)

    elif randomWeather == "Rainy":

        rain = []
        rain_drops = random.randint(1000,2000)
        random_number = random.randint(-10,10)
        for elem in range(rain_drops):
            if random_number < 0 :
                x = random.randint(random_number,image.shape[1])
            else:
                x = random.randint(random_number,image.shape[1]
                                   - random_number)
            rain.append((x,random.randint(0,image.shape[0]- 5)))
        for drop in rain:
            start_point = (drop[0],drop[1])
            end_point = (drop[0] + random_number,drop[1] + 5)
            color = (175,195,204)
            cv2.line(image,start_point,end_point,color,5)
        image = cv2.blur(image,(7,7))
        rainy_image = cv2.add(image,70)
        rainy_image = ((rainy_image + 1 ) * 256 ) -1
        cv2.imwrite(inputFolder + img[folderlen:],rainy_image)

    elif randomWeather == "Snowy":

        rain = []
        rain_drops = random.randint(1000,2000)
        random_number = random.randint(-2,2)
        for elem in range(rain_drops):
            if random_number < 0 :
                x = random.randint(random_number,image.shape[1])
            else:
                x = random.randint(random_number,image.shape[1]
                                   - random_number)
            rain.append((x,random.randint(0,image.shape[0]- 2)))
        for drop in rain:
            start_point = (drop[0],drop[1])
            end_point = (drop[0] + random_number,drop[1] + 2)
            color = (255,250,250)
            cv2.line(image,start_point,end_point,color,2)
        image = cv2.blur(image,(5,5))
        snowy_image = cv2.add(image,70)
```

```
        snowy_image = ((snowy_image + 1 ) * 256 ) -1
        cv2.imwrite(inputFolder + img[folderlen:],snowy_image)

    elif randomWeather == "Fogy":

        foggy_image= cv2.blur(image,(10,10))
        foggy_image = ((foggy_image + 1 ) * 256 ) -1
        cv2.imwrite(inputFolder + img[folderlen:],foggy_image)
```

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

CHALMERS
UNIVERSITY OF TECHNOLOGY