Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Kristoffer Landsnes

Domain Adaptation for Detection of Maritime Vessels in Images

A Comparative Study on the Effects of Targeted Detection Pre-Training Using Real-World Data

Master's thesis in Cybernetics and Robotics
Supervisor: Edmund Brekke
Co-supervisor: Øystein Kaarstad Helgesen
July 2021

Kristoffer Landsnes

Domain Adaptation for Detection of Maritime Vessels in Images

A Comparative Study on the Effects of Targeted Detection Pre-Training Using Real-World Data

Master's thesis in Cybernetics and Robotics
Supervisor: Edmund Brekke
Co-supervisor: Øystein Kaarstad Helgesen
July 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Abstract

Object detection is imperative for situational awareness in autonomous systems, promoting safe and controlled autonomous navigation. Maritime camera-based object detectors, though being one of the key-systems for providing rich object structure-information, are often based on incomplete and small-scale datasets for training and evaluation.

In this thesis, we explore the effects of pre-training and fine-tuning object detectors for maritime vessel detection; referred to as targeted detection pre-training. Existing annotated maritime data is acquired, resulting in three experimental datasets of optical images for detection pre-training. The largest of which, comprises a total of 17,871 images with 95,398 labeled maritime vessels. In a real-world setting, domain adaptive fine-tuning is executed on a manually labeled target domain dataset representing the operational area of the autonomous ferry milliAmpere.

The state-of-the-art EfficientDet-D3 detector is selected in accordance with inference time requirements from the sensor rig of milliAmpere. Fine-tuning into the target domain is executed for full fine-tuning (FF), frozen backbone (FB) and fine-tuning of the EfficientDet-D3 prediction heads only (HO).

Based on reported COCO AP metrics on the target domain test set and several case-study scenarios, we highlight our main findings. 1) Targeted detection pre-trained models consistently converge faster and to higher performance scores than all baselines, even for fewer fine-tuned epochs. 2) Targeted detection pre-trained models are more robust, mitigating false-negative predictions in challenging scenarios while producing tighter and more confident predicted bounding boxes. 3) More freezing is inferior to full fine-tuning when the pre-training and target tasks and labels are the same.

Targeted detection pre-training is found highly beneficial for improving maritime vessel detection in the target domain, encouraging the adoption of this scheme for faster stream-lined and more robust detector development on small-scale maritime target datasets.

Sammendrag

Deteksjon av objekter er imperativt for situasjonsforståelse i autonome systemer og bidrar til trygg og kontrollert autonom navigasjon. I maritime miljøer er kamerabaserte detektorer et av de viktigste systemene som gir tilgang på detaljert strukturell informasjon om objekter, men ofte er slike detektorer basert på ufullstendige og småskala datasett for trening og evaluering.

I denne avhandlingen undersøker vi effekten av pre-trening og finjustering av objekt-detektorer for deteksjon av maritime fartøy; også kalt målrettet deteksjons pre-trening. Eksisterende annotert maritim data er ervervet og benyttet til å designe tre eksperimentelle datasett med optiske bilder for deteksjons pre-trening. Det største datasettet består av totalt 17,871 bilder med 95,398 annoterte maritime fartøy. Domeneadaptiv finjustering utføres på et eget-annotert måldomene datasett, som representerer operasjonsområdet til den autonome fergen milliAmpere.

State-of-the-art detektoren, EfficientDet-D3, er valgt i samsvar med krav til deteksjonstid fra sensorriggen til milliAmpere. Finjustering inn i måldomenet utføres for full finjustering (FF), fryst konvolusjonsnettverk (FB) og finjustering av EfficientDet-D3 prediksjonshodene (HO).

Basert på COCO AP ytelseskriterium på måldomenets testsett og flere case-studie scenarioer, fremhever vi hovedfunnene våre. 1) Målrettede deteksjons pre-trente modeller konvergerer konsekvent raskere og til høyere ytelse enn alle basislinje (baseline) modeller, selv for færre finjusterte epoker. 2) Målrettede deteksjons pre-trente modeller er mer robuste og avverger falske-negative prediksjoner i utfordrende scenarier, mens de produserer mer presise bounding-bokser med høyere trygghet. 3) Mer fryste parametere gir dårligere ytelse enn for full finjustering når pre-trening oppgaven og måldomenets oppgave samt annoterings klassene er de samme.

Målrettet deteksjonstrening er funnet meget gunstig for å forbedre deteksjon av maritime fartøy i måldomenet og motiverer videre bruk av denne teknikken for raskere og mer robust detektorutvikling på småskala maritime måldatasett.

Preface

This thesis marks the end of my Master of Science (MSc) degree in Cybernetics and Robotics at the Norwegian University of Science and Technology (NTNU). The thesis is written in the spring semester of 2021 under the guidance of Edmund Brekke and co-supervisor Øystein K. Helgesen. Parts of the thesis are motivated or adapted from the author's unpublished specialization project [56] conducted in the fall of 2020, as follows:

- *Parts of chapter 2.*
- *Parts of chapter 3.*
- *Parts of section 4.1 - 4.2.*

I would like to thank Edmund Brekke and Øystein K. Helgesen for support and invaluable feedback on drafts of this thesis, provided even during weekends and holidays. The experiments and results in this thesis would not have been possible without the annotated Hurtigruten dataset, provided by DNV (Det Norske Veritas), together with instructions from Kristian B. Karolius. Øystein K. Helgesen and Erik Wilthil further provided kayak videos recorded in experiments on milliAmpere, used in this thesis. I would also like to thank Andreas T. Henriksen who has been an important sparring-partner and challenged my ideas and experiment choices.

Lastly, this thesis would not have been possible without the unconditional support of my parents.

Trondheim, July 2021

Kristoffer Landsnes

ACRONYMS

ANN Artificial Neural Network. iii, 6–8

ASV Autonomous Surface Vessels. iii, 1, 103

BiFPN Bi-directional Feature Pyramid Network. iii, 14, 61

CNN Convolutional Neural Network. iii, 6–11, 13, 14, 28, 37, 100

COLREGs Convention on the International Regulations for Preventing Collisions at Sea. iii, 1

DA Domain Adaptation. iii, 2, 23, 24, 55, 85, 87, 102

DCNN Deep Convolutional Neural Network. iii, 2, 21, 22, 24–26, 30, 87, 100, 101

DNN Deep Neural Network. iii, 6, 25, 26

EO Electro-optical. iii, 3, 50, 51

FLOPS Floating Point Operations. iii, 14

FPN Feature Pyramid Network. iii, 9, 11, 13, 14

FPS Frames Per Second. iii, 3, 12, 13, 45, 50, 52, 58, 87, 89

IOU Intersection over Union. iii, 10, 13, 16, 18, 27–29, 47, 61, 62, 65, 72, 76

LIME Local Interpretable Model-Agnostic Explanations. iii, 101, 104

mAP Mean Average Precision. iii, 2, 10, 12–15, 18, 20, 27–30, 57, 58

MMD Mixed Maritime Dataset. iii, 56, 64, 74–77, 82, 87

MSE Mean Squared Error. iii, 6

NIR Near-Infrared. iii, 2, 43

NMD Nordic Maritime Dataset. iii, 55, 56, 64, 70, 72, 74–77, 82, 85, 87, 88, 94, 100

NMS Non-Maximum Suppression. iii, 10, 12, 13, 62, 87

RoI Region-of-interest. iii, 9–11

RPN Region Proposal Network. iii, 9–11

SGD Stochastic Gradient Descent. iii, 58, 67

SHAP SHapley Additive exPlanations. iii, 101, 104

SMD Singapore Maritime Dataset. iii, 2, 4, 5, 20, 21, 43–50, 55, 56, 64–70, 74–77, 82, 85, 87, 100, 102

SVM Support Vector Machine. iii, 10, 100

VIS Visual-Optical. iii, 2, 32, 43–45

XAI Explainable Artificial Intelligence. iii, 101, 104

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Autonomous navigation has in recent years become an increasingly hot topic of research. The potential advantages are many-fold and expand beyond efficient route planning, reduced labor-cost and increased traffic safety. Autonomous cars are often featured in the news with impressive results from companies such as Tesla, Uber and Google. On the other hand, Autonomous Surface Vessels (ASV) engender a research branch with just as large implications. ASVs are considered one of the eight most important future maritime technologies [93].

The future, on the other hand, may be closer than expected. The Yara Birkeland project, a collaboration between Kongsberg and Yara, is in the final stage of launching their zero-emission autonomous container ship, projected to perform fully-autonomously by 2022 [9]. An autonomous navigation system for ferry traversing and docking provided by Kongsberg was used to autonomously conduct the world's first adaptive ferry transit with passengers in 2020 [7]. The Mayflower Autonomous Ship, is projected to cross the Atlantic ocean fully-autonomously during 2021 [8].

An autonomous navigation system inherently depends on a complex pipeline of environmental perception, situational-awareness and robust algorithms for collision-avoidance. The first step in such a pipeline is perceiving and locating surrounding objects, or *detecting* objects.

The detection of objects is imperative seen in relation to the Convention on the International Regulations for Preventing Collisions at Sea (COLREGs) [48], which is a set of rules regulating several aspects of maritime navigation, including collision avoidance. Collision scenarios often occur in close proximity navigation, such as crossing situations and while overtaking another vessel (rule 13-15).

Active non-visual sensors, such as lidar and radar, provide precise positional information of the surrounding environment and vessels. Standalone, radar and lidar struggle to perform accurate obstacle detection and are often used in fused sensor systems.

Helgesen [42] implemented a sensor fusion system of active and passive sensors. Helgesen found that fusing lidar and radar measurements improved tracking accuracy, while the addition of infrared camera measurements further increased tracking robustness. Turøy [105] developed a COLREGs collision compliant system based on collision avoidance (COLAV) with AIS transmitted signals. Turøy further implemented an IPDA lidar tracking system, verifying robust obstacle avoidance for the COLAV method.

Smaller boats often do not have AIS transmitters and while possible to detect by non-visual sensors, either standalone or fused, object detection in optical images have proven to be a valuable addition by providing accurate localization and object structure-information [52] extendable for integration into camera tracking and obstacle avoidance systems. For instance, by detecting objects such as sea kayaks and boats without radar reflector or AIS transmitters [15].

In line with recent advances in deep learning, Deep Convolutional Neural Network (DCNN) methods are considered state-of-the-art in object detection [51]. Deploying deep learning based object detectors for maritime object detection is not a new phenomena.

The Mask R-CNN [40] architecture was deployed in a study comparing detection performance of an unmanned bridge with a human navigator for ship navigation [15]. Blanke et al. found the Mask R-CNN to detect objects faster than its human counterpart, but pointed out the need for more training data to improve the detector's performance in certain situations, making it more suitable as an extra fifth sense for the human navigator. Nita and Vandewal [72] similarly present generalization issues for the Mask R-CNN when training on a small custom dataset.

Tangstad [102] utilized Faster R-CNN [86] to detect maritime objects for collision avoidance. Similar to other research [67], Tangstad scraped maritime images from large universal detection datasets [26]. Grini [35] trained the YOLOv3 [82] and SSD [63] one-stage detectors for boat and building detection on a manually collected and annotated dataset of 1,916 images, partially from local maritime waters in Trondheim, made available for this thesis. The detectors demonstrate robustness on evaluation, though struggling with overfitting behaviour due to the small data foundation.

The Singapore Maritime Dataset (SMD) [79] is, to the best of our knowledge, the largest publicly available maritime dataset with instance labels for different vessel types, spanning over 30,000 labeled frames from the Visual-Optical (VIS) and Near-Infrared (NIR) spectra. The SMD has been treated as to represent a maritime benchmark dataset [70], with a proposed dataset split and benchmark results from the Faster R-CNN and a pseudo-mask supervised Mask R-CNN.

The computer vision community evolves in an extremely rapid pace. One of the corner-stones in this evolution is based on *transfer learning* and particularly *pre-training*. Pre-training on a source task has proven to transfer useful features for a target task [32]. In object detection, this discovery has been widely accepted by using large-scale classification datasets, such as ImageNet [26], for pre-training the feature extracting modules of object detection architectures to learn more low-level generic and transferable features [111]. A subsequent stage of fine-tuning the object detector adapts the more domain and task-specific features.

Novel work in the field adopts large-scale classification pre-training; such as YOLO based detectors [82] [83] [84] and region based detectors [34] [33] [86] [40]. Kornblith et al. [54] demonstrate that ImageNet pre-trained features are less generic than previously thought. He et al. [41] further challenge the concept of large-scale classification pre-training for detection, by showing that training an object detection architecture from scratch, with a modified training schedule, achieves similar detection performance to models pre-trained on ImageNet.

Larger available detection datasets, such as the partially-labeled OpenImages [55] of 1.9 million images and Objects365 [91] consisting of more than 600,000 images, have inspired more in-depth research on the effects of *detection pre-training*. Pre-training and fine-tuning for the same task is intuitively logical, as to update the relevant parameters for solving the target task during pre-training. For the detection task, this is referred to as *targeted detection pre-training*.

Li et al. [60] find that targeted detection pre-training produces superior target domain detection performance compared to classification pre-training. It provides faster convergence to higher detection scores, measured in Mean Average Precision (mAP), and improved fine-localization capabilities, better capturing the spatial entirety of detected objects. Similar results are presented by Zhong et al. [113], additionally pointing out that the whole detection network's feature representations are adapted more towards detection. Shao et al. [91] similarly report an overall higher achieved mAP from detection pre-training. Generally, [60] [113] [91] all use rather large detection pre-training datasets based on OpenImages, Objects365 and bounding box labeled ImageNet. Nevertheless, targeted detection pre-training always outperforms from scratch training and classification pre-training.

In this thesis, we present the first experiments with *targeted detection pre-training* in maritime environments, designed from theory in Domain Adaptation (DA). Moreover, to the best of our knowledge, we report detection results for the largest annotated maritime vessel dataset of optical images to date, spanning a total of 17,871 images with 95,398 labeled vessels.

1.1 Problem formulation

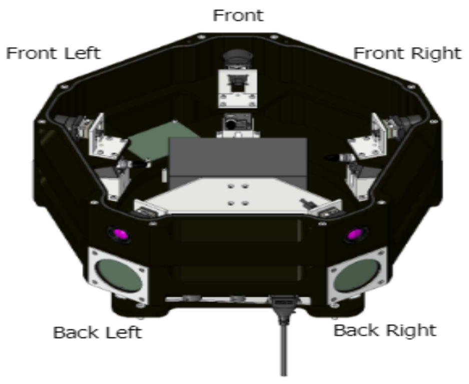
The general problem of interest in this thesis is the detection of maritime vessels in optical camera images. The images are colourized in three colour channels following the RGB (Red-Green-Blue) colour model. By *detection*, we mean the localization and classification of objects in the scene. The algorithm for detecting the objects, or the *detector*, locates the object by the pixel positions of a rectangle enclosing the object in the image, referred to as a *bounding box*.

The detector is designed in accordance with a real-world application. In particular, the detector is intended for future deployment and integration with the Electro-optical (EO) cameras of the sensor rig of the autonomous ferry milliAmpere.

The sensor rig integrates video streams of five EO cameras as presented in figure 1.1a. Each camera is of the type BlackFly S GigE with 2448 × 2048 resolution and frame-rate of 22 Frames Per Second (FPS). An illustration of the integrated cameras on the sensor rig of milliAmpere is presented in figure 1.1b. Due to bandwidth restrictions, each camera records images of 5 FPS with 1224 × 1024 resolution. As such, the designed detector must at least be capable of processing 25 FPS, considering a sequential feeding of one image at a time from each camera.

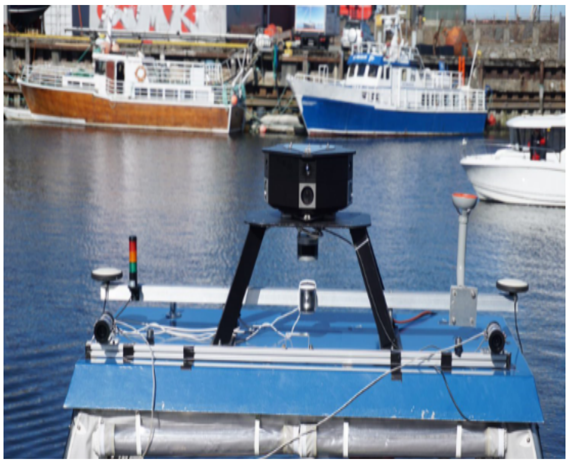
Based on the described detection setting of milliAmpere, two more observations are important. Firstly, the mounted EO cameras of the sensor rig are rather close to the ocean surface. Secondly, milliAmpere operates in the area around Ravnkloa, the harbour orifice of Trondheimsfjorden and Nidelven. The first point defines the expected object viewpoint, which is rather close to the ocean surface. The second point formulates the environment and situations to expect on deployment. Together, these two points determine the requirements for the *target domain*.

A dataset representing the target domain must be designed for generalization during training of the detector and in order to design test-scenarios in the wild. The images composing the dataset should be of an as large diversity as possible, due the dynamical nature of maritime environments. Optimally, including images of different weather conditions, lens water droplet occlusion, lens flare from the sun, waves and still ocean surface, to mention some. Specific scenario testing and a video inference test are imperative to ensure reliability and robustness of the designed detector.



(a) milliAmpere electro-optical camera setup

Source: [47]



(b) milliAmpere complete sensor rig

Figure 1.1: EO camera station milliAmpere. EO camera model-type: BFS_PGE_50S5C-C.

So far, the real-world requirements for the detector design are presented.

However, this thesis also serves as a continuation from the author's specialization project [56]. In the specialization project, techniques for improving object detection in maritime environments without manually labeling more data were created, resulting in a weakly-supervised instance segmentation scheme. Briefly, pixel-level mask annotations were automatically created from two segmentation algorithms to supervise a Mask R-CNN [40].

In the writing of the specialization project, the author made several other observations regarding maritime object detection research. Most important, a recurring observation is the use of particularly small and custom annotated datasets for supervising object detectors. This is in contrary to the computer vision research field, where dataset size is tightly linked with the fast evolution in the field, often spanning millions of images.

The background for this observation is firstly that many maritime datasets are not publicly available [70]. Moreover, maritime object detectors are often designed for a specific target domain, from which data must be sampled. Nevertheless, there exist large maritime datasets publicly available, the most noteworthy large with bounding box ground truths is likely the SMD [79].

The following question has therefore largely guided the research of this thesis:

- "*How can existing bounding box annotated maritime datasets be exploited to improve the detection of maritime vessels in the target domain defined by milliAmpere?*"

With "improve detection", we mean to improve performance metric scores, but also possibly ameliorate performance more visible in scenario testing, as for instance better detection in challenging scenarios and more consistent detections over time. The guiding question may permit us to shed light on smarter methods for improving target domain performance than manually collecting and labeling large amounts of target domain data, which is inherently laborious.

1.2 Contributions

The main contributions from this thesis may be summarized as follows.

- Presenting an overview of several state-of-the art object detectors.
- A literature survey on research in maritime object detection and environments.
- A literature survey on two topics in transfer learning. 1) Fine-tuning strategies and feature transferability, 2) Recently published research in pre-training.
- A mapping and detailed overview of several maritime datasets. Additionally, an in-detail walk through of the generation of the video-based SMD [79].
- The generation and labeling procedure of a custom target domain dataset for milliAmpere of 1,061 images.
- The design of three experimental maritime bounding-box annotated datasets. Including; to the best of our knowledge, the largest recorded maritime vessel detection dataset of optical RGB images to date.
- Implementation details for the EfficientDet-D3 architecture, together with three custom fine-tuning settings.
- To the best of our knowledge; The first formulation and conducted experiments of *targeted detection pre-training* in maritime environments.
- Carefully designed experiments with concise baselines, resulting in 21 trained and evaluated models on the target domain, including a class-aware target domain ablation study.
- Several case-studies and a video inference test from the target domain.

1.3 Report outline

- Chapter 2 presents theory related to the object detection task, as well as some of the state-of-the art architectures.
- Chapter 3 conducts a literature survey on maritime object detection and transfer learning techniques in computer vision.
- Chapter 4 presents all datasets explored in this thesis, including the SMD, the Hurtigruten dataset; a private dataset provided by DNV for this thesis, and a custom designed target domain dataset.
- Chapter 5 presents all aspects of the experiment design. Including; the targeted detection pre-training formulation, all experimental datasets, the selected object detection architecture and corresponding hyperparameters.
- Chapter 6 covers the execution and details of all conducted experiments. Firstly, presenting the experiments to obtain detection pre-trained weights, before executing the targeted detection pre-trained experiments, fine-tuning into the target domain.
- Chapter 7 presents the results obtained when evaluating all models on the target domain test set, as well as multiple case-studies to 1) Assess the effects of targeted detection pre-training and detector robustness in the target domain 2) Execute a video inference test in the wild.
- Chapter 8 provides a further discussion of the results and potential error sources.
- Chapter 9 concludes the report with recommendations for future work.

CHAPTER 2 OBJECT DETECTION

This chapter treats the theoretical foundation in object detection, covering supervised learning in the evolution from neural networks to state-of-the-art object detectors. Additionally, several object detection performance metrics are presented.

2.1 Artificial neural networks

An Artificial Neural Network (ANN) is inspired by the neurons activated by electric impulses in the human-brain.

ANNs consist of connected nodes in a network structure, with a corresponding weight on the connecting edge between nodes. The input nodes in the *input layer*, simply propagate their input values by their weighted edges. The following layer is normally a *hidden-layer*. If the network has more than one hidden-layer, it is called a Deep Neural Network (DNN).

All nodes in a hidden layer are modelled as an approximation to a biological neuron, which mathematically is achieved by a non-linear *activation-function*. One common non-linear activation function is the sigmoid function, $\sigma(x) = \frac{1}{1+e^x}$. The input to each node in the hidden layer is a weighted sum of propagated input values, either from the input layer or a previous hidden layer, and an additional bias from the *bias-nodes* in the network. The non-linear activation function outputs the mapped value [11]. The final output from the ANN is obtained from the output node(s) in the *output-layer*. Figure 2.1 illustrates an ANN architecture with one hidden layer.

The above-described network has information flowing from the input nodes to the output nodes, often called a *feedforward neural network*. Such networks are suited for supervised classification. The weights in the network are learned during a *training-procedure*, supervised by ground-truth labels. The bias nodes shifts the activation function while the non-linear activation functions enables the network to find non-linear patterns in the inputted data. The network's class predictions are encapsulated in a loss-function, for instance Mean Squared Error (MSE), to quantify its classification error from the ground truth. The learned weights are as such updated by the back-propagation algorithm until training is completed.

Overfitting A more general concept of training neural networks, which includes the training of ANNs and CNNs, is overfitting. Conceptually, if the loss-function of the neural network keeps decreasing one might expect the classification performance to increase. However, the network might become to specialized or *overfitted* to the data seen during training.

One way to counter this is by dividing the dataset into separate split sets designated for training and testing. The network is trained on the training set and tested on the test set. Thus, enabling

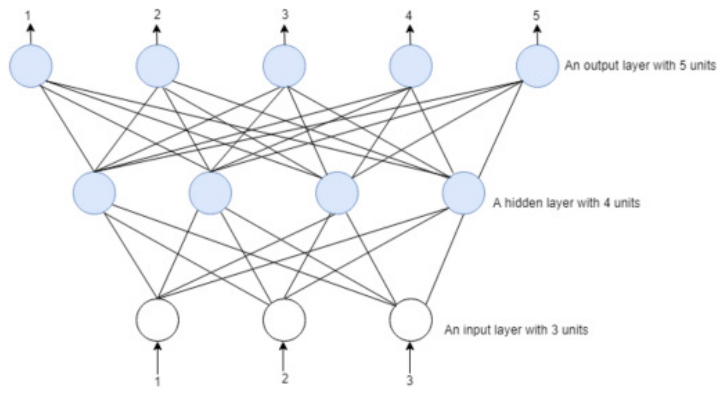

1
2
3
4
5
An output layer with 5 units
A hidden layer with 4 units
An input layer with 3 units
1
2
3

Figure 2.1: Two-layer feedforward ANN

Source: [11]

a monitoring of the training and testing error separately. If the train-error keeps decreasing while the test-error increases, it is a sign of overfitting. Additionally, the test set can be split into a validation set and strict test set, where the validation set can be used for hyper-parameter tuning and model-selection. The strict test set serves as a test of the predictive capacity on unseen samples in the wild.

2.2 Convolutional neural networks

For image classification, Convolutional Neural Network (CNN)s are more suitable to use than ANNs. Namely, because they take advantage of the spatial or grid-like structure inherent for the pixels in images. A CNN normally consists of three types of layers which stacked together form a CNN architecture.

Convolutional layers slides a learnable kernel over the pixels in the inputted image horizontally and vertically. This is similar to a *sliding window*. The resulting output has a decreased dimension and is often called a *feature-map*. That is, a lower-level resolution representation, containing the kernel's response from sliding over the pixels. The convolutional layer horizontally slides over all pixels, if the *stride* is set to zero. By increasing the stride, the kernel will correspondingly skip pixels horizontally, reducing the output dimension.

Pooling layers downsamples the feature map inputted, decreasing the resolution and easing the amount of network parameters needed, while extracting distinct features. Max-pooling, outputs the highest value of each grid-cell in the feature map, where the grid-size depends on the max-pooling kernel size. This layer is not learnable.

The convolutional and pooling layers, are often referred to as the *feature extraction* part of the CNN. Several stages of convolutional and pooling layers reduces the outputted feature map dimensions, yielding high-level feature maps inputted to the fully connected layers for classification.

Fully connected layers denote the final stage of a CNN architecture, returning the class probabilities. A feedforward neural network is found to perform well as the final classification layer [30]. The final output neurons' activation function predicts the probability of a sample belonging to each class. One such function is the *softmax* function.

An example of a CNN architecture is illustrated in figure 2.2.

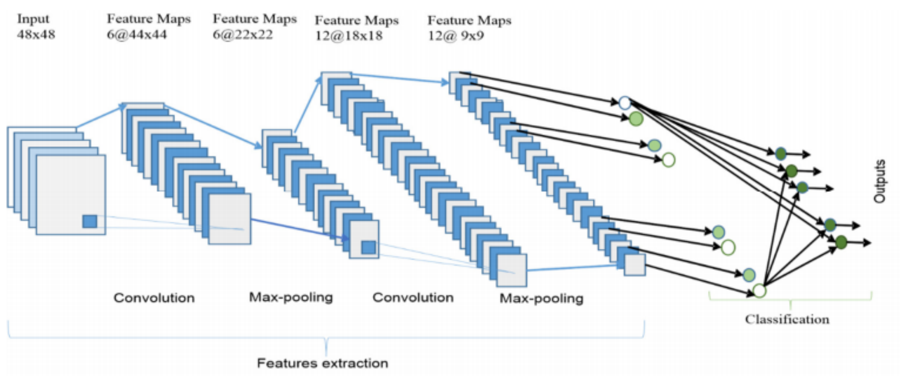


Figure 2.2: CNN layers and architecture

Source: [13]

Regularization Another important concept in training ANNs and CNNs is regularization. Regularization techniques are designed to mitigate overfitting by imposing the learning of a less complex model. Methods such as dropout, which randomly drops the weight update of certain nodes during training, or batch normalization [49], which introduces a layer for normalizing each batch of training samples to mitigate changing data distribution between layers, are commonly adopted [101]. The filters and max pooling layers in a CNN also serve as a regularization measure by restricting the amount of learnable parameters.

2.3 Object detection

Before passing on to the main part of this chapter, we make a short review of the four most common tasks in computer vision, presented in figure 2.3.

Image classification, is related to predicting the presence of objects belonging to a certain class which occur in the scene. *Object detection* extends the image classification task, by requiring both to classify and spatially locate all instances in the scene. Predictions for these two tasks are often accompanied by a confidence score, indicating the level of certainty in the observed class in the image or in a specific predicted spatial location.

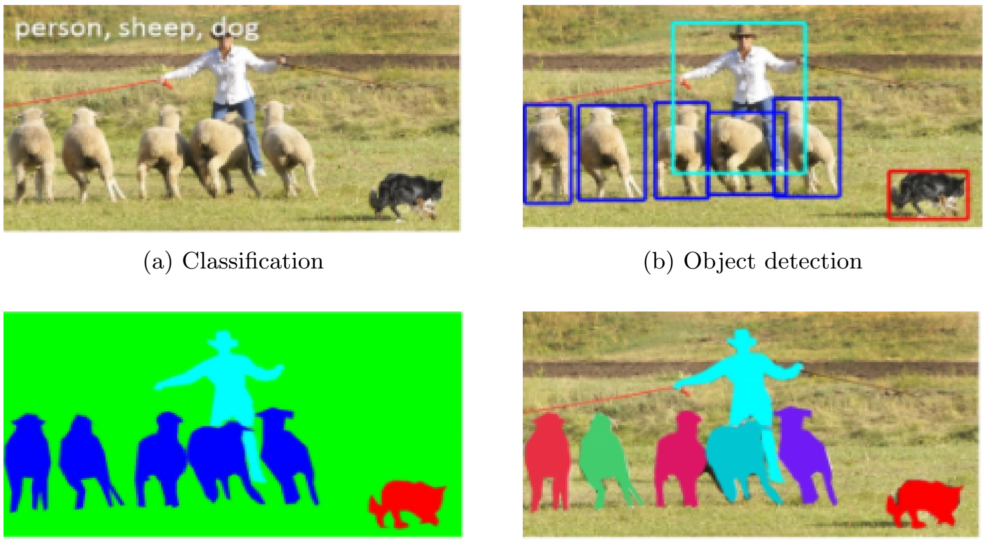
Segmentation, on the other hand, demands pixel-level accuracy when classifying and localizing objects. We separate between *semantic* and *instance* segmentation. Semantic segmentation aims to localize all pixels belonging to a class. As observed in figure 2.3c, where all pixels belonging to the sheep class are marked blue. Instance segmentation further separates between instances of each class, treating each instance as a unique instance of the class.

For the remaining part of this chapter, object detection is the main focus.

2.3.1 Concepts in object detection

Modern object detection architectures are composed of modules, often with specific terminology in the object detection research environment. Here, we clarify some of the most important concepts and building-blocks.

Backbone A backbone simply refers to the feature extraction network used in an object detection architecture. This is the same as a CNN without its outputting fully-connected layer. As it is often preferable to reduce training time for deep-learning based object detectors, it is common practice to use backbones with *pre-trained* weights. Conceptually, this practice is categorized as transfer learning, treated in section 3.2. In essence, the backbone is trained for classification on a large-scale dataset, such as ImageNet [26], before fine-tuning the object detection architecture on a different dataset. A famous backbone is ResNet [39], introduced in 2015. ResNet managed



(a) Classification

(b) Object detection

(c) Semantic segmentation

(d) Instance segmentation

Figure 2.3: Main tasks in computer vision. Figure reused with permission from publisher.

Source: [61]

to train a deep architecture for improved performance on previous state-of-the-art, utilizing skip connections to tackle the previous vanishing-gradient problem.

Feature Pyramid Network (FPN) FPN [62] (2016) was introduced to better detect objects of different scales. Instead of utilizing feature maps of a fixed resolution for detection and classification, the FPN creates a pyramid of feature maps with different resolutions. The down-sampling is based on a standard CNN in a bottom-up pathway. The more semantic rich low-resolution layers are combined with higher resolution layers by lateral skip connections and a top-down pathway to retrieve feature maps of high semantic value at different scales. Combining this with a Faster R-CNN architecture outperformed all previous models on the COCO-detection challenge [62]. The RoI pooling layer of the Faster R-CNN thus extracts features from the FPN at different levels according to the scale. The FPN with the lateral connections is illustrated in figure 2.4.

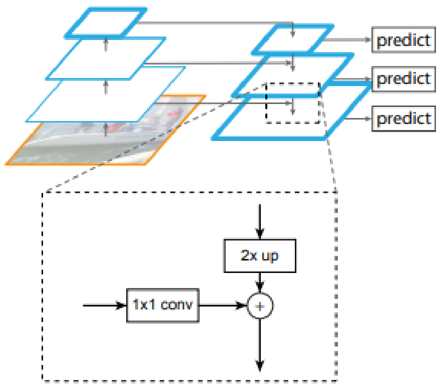


Figure 2.4: FPN building block with lateral connection and top-down pathway. ©[2016] IEEE.

Source: [62]

Region Proposal Network (RPN) RPN is one of the main components in two-stage object detector architectures, explained below. Essentially, the RPN uses something called *anchor boxes* for proposing regions in the feature map containing objects. The anchor boxes are rectangles of fixed size and aspect ratio, which are used for each sliding-window location over the feature map to capture objects of various shape and size. Each proposed region is assigned an objectness score for the probability of containing an object of a class. Typically, the RPN produces a lot of overlapping proposals, which can be countered by a following stage of Non-Maximum Suppression (NMS). NMS removes proposed regions with a higher overlap than a predefined IOU threshold based on the regions' objectness score and is commonly adopted to filter out multiple bounding box predictions and retain the best ones [83]. The remaining proposed regions are thereafter used for classifying and detecting objects. RPNs were first introduced in the Faster R-CNN architecture [86] (2015).

One-stage and two-stage detectors Modern object-detector architectures are divided into two groups. Object detectors such as the Faster R-CNN and Mask R-CNN depend on region proposal methods, such as RPN, for pinpointing the region of interest in an image. The region proposals are thereafter processed for for classification and bounding box regression later in the detection pipeline. Such methods are referred to as *two-stage detectors* and are highly accurate, but due to the intermediary stage of region proposals, slower than *one-stage detectors*.

One-stage detectors learns class probabilities and bounding boxes as a regression problem over the entire image. By avoiding the intermediary stage of region proposals, one-stage detectors obtain a higher inference speed [51], by trading-off a lower accuracy. As follows, such methods are considered more suitable for real-time object-detection. Two such methods are YOLO [83] and SSD [63].

2.4 Object detection architectures

As the computer vision community is rapidly evolving, a review of the state-of-the-art object detectors, together with their historical predecessors, is considered necessary. Both one-stage and two-stage detectors are presented.

2.4.1 R-CNN family

Today's two-stage detectors are largely based on the evolution of the R-CNN family of detectors. The R-CNN architecture [34] was introduced in 2013, shortly followed by the Fast R-CNN [33] in 2015 and Faster R-CNN [86] later in 2015. The last model extending the R-CNN family is the Mask R-CNN [40] from 2017.

R-CNN R-CNN uses a multi-step architecture, making it subject to multi-stage training. Firstly, a selective search algorithm [106] is utilized for proposing regions in the image. The region proposals are passed through a CNN feature extractor for extracting the feature maps. Further, each feature map is passed to a Support Vector Machine (SVM), classifying the objects in the region. There is one SVM specialized and trained for each class in the dataset. As a last step, a bounding-box regressor is added for localization of the objects in the proposed regions. The mAP on the VOC 2012 challenge was improved by over 30% compared to previous best results. Indicating the significance and importance of this method. However, the R-CNN requires around 47s per image on test inference and requires separate training of the feature extractor(pre-training and fine-tuning), SVMs and bounding box regressors.

Fast R-CNN The forward pass of each proposed region through the feature extracting CNN in R-CNN proves to be memory consuming and inefficient. To solve this, Girshick proposed to extract all features from the image once, before extracting the relevant feature maps through a Region-of-interest (RoI) pooling layer. The RoI pooling layer extracts a fixed size feature maps for different sized regions, conserving their spatial information. Training Fast R-CNN on the Pascal VOC 2007 dataset is 9 x faster than for R-CNN, while achieving a higher mAP score. Test time inference similarly proved to be up to 213 x faster.

Faster R-CNN Faster R-CNN improves the region proposal methods of its predecessors, previously carried out by the selective search algorithm. The selective search algorithm proposes regions

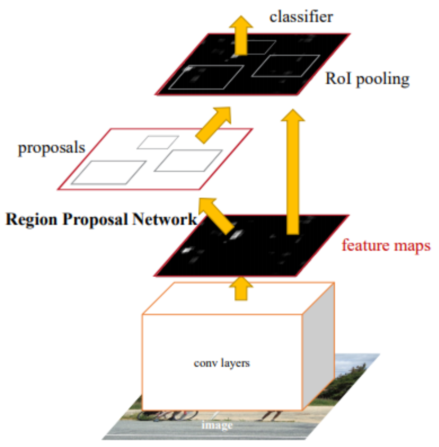


Figure 2.5: Faster R-CNN architecture. Figure reused with permission from publisher.

Source: [86]

based on a hierarchical grouping of sub-regions from different cues, such as colour and texture. Inherently the algorithm is slow in its iterative process of merging sub-regions. Thus, Ren et al. propose to use a RPN, described in section 2.3.1. The RPN enables more robust object-detection for objects of different sizes due to the anchor boxes. The same RoI pooling layer is utilized. The performance improvements are clear, improving Fast R-CNN's reported mAP on Pascal VOC 2007 by 3% with a 10 x faster inference time, reporting 0.5 fps. The Faster R-CNN architecture is visualized in figure 2.5.

Mask R-CNN Instance segmentation has proved to be a valuable addition for a multi-task loss in the object detection pipeline [38]. Exploiting semantic features in an object detection architecture is further found to improve the detection performance [32] [19]. This observations makes the Mask R-CNN [40] a very interesting architecture.

Mask R-CNN has been a popular choice of architecture for the object detection and instance segmentation task in recent years. Since Mask R-CNN predicts the localization of instances in the bounding box and pixel-wise mask format, it is required to be supervised with both bounding box and mask annotations.

The network is a further development of Faster R-CNN, improved and changed for three different parts. Firstly, a ResNet-FPN architecture is used as backbone, for efficient multi-scale detection and feature map extraction, particularly improving small object detection. Secondly, the RoI pooling layer is replaced with a new pooling layer named RoIAlign. RoI pooling utilizes two-steps of quantization for extracting the pooled feature map, causing misalignment from the pooling layer and the extracted feature map. RoIAlign is replacing the two-stages of quantization with bilinear interpolation and pooling to retrieve a better aligned feature map. Lastly, a fully convolutional mask head branch is added for predicting the pixel-wise mask for each RoI in parallel with the bounding box recognition head. He et al. found it important to predict the masks for each class independently, without competition amongst classes. The total loss-function of the architecture consist of the classification, bounding box regression and segmentation mask loss in a multi-task loss function; $\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{box} + \mathcal{L}_{mask}$. Figure 2.6 illustrates the final Mask R-CNN architecture.

Mask R-CNN outperforms all previous contestants in the COCO 2016 detection challenge, even though the reported inference speed is 5 fps, proving to be slower than the preceding Faster R-CNN. An interesting result is the Mask R-CNN's sensitivity to the mask branch for object detection. The architecture reportedly performs better object detection with the mask branch, than without. This is stated to be solely because of the multi-task loss that includes the mask-loss. In other words, the isolated instance segmentation improves object detection for the architecture. Similar observations for a general CNN architecture is found in [19].

This can be linked to several benefits of pixel-wise annotation:

- Bounding box annotations tend to include background and features outside the object.
- Mask annotations gives pixel-level accuracy.
- Masks can benefit from richer and more structured representations

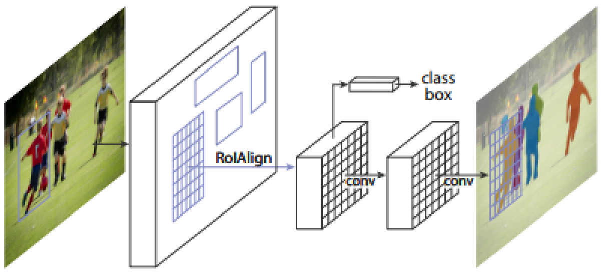


Figure 2.6: Mask R-CNN architecture. ©[2017] IEEE.

Source: [40]

Mask R-CNN extensions There exists several versions and extensions of Mask R-CNN. Tensormask and BMask R-CNN are two such methods.

Tensormask [20] examines mask predictions in dense regular grids through a dense window sliding method, making it a one-stage method. As it does not produce better results than Mask R-CNN for instance segmentation, rather a different methodology, it will not be discussed further. It has further been pointed out that Mask R-CNN does not utilize the shape features of the object instances it aims to segment. [22] addresses this by leveraging boundary information in a new mask-head parallelly learning mask and object boundary, in an architecture named BMask R-CNN. BMask R-CNN outperforms Mask R-CNN on the COCO dataset, and particularly for metrics demanding fine localization(AP_{75}).

2.4.2 You Only Look Once (YOLO)

The development of one-stage object detection architectures is to a large extent based on the ground-breaking work represented by the YOLO-family of detectors. The YOLO detectors have evolved in six stages from the original paper [83] published in 2016, with additional instance segmentation versions such as YOLACT [17] existing.

For this review, the initial concepts in YOLOv1 [83] are presented with a brief summary of improvements leading to one of the state-of-the-art detectors today, scaled-YOLOv4. YOLO-V5 is omitted as it is not peer-reviewed at the time of writing this thesis.

YOLOv1 YOLOv1 [83] (2016) is inspired by convolutional networks such as GoogleNet [98]. YOLOv1 phrases the object detection problem as a regression problem to predict bounding boxes and their confidence, together with corresponding class probabilities, from an image in one evaluation. Hence, the catchy name “You only look once”.

Each inputted image of 448×448 resolution is divided into grid cells, $S \times S$, with $S = 7$, where one grid cell can predict $B = 2$ bounding boxes. By using NMS for conserving the predictions of highest confidence, the process from input to prediction stage is presented in figure 2.7. In 2016, YOLOv1 reported the first real-time object detection architecture with lower but comparable mAP scores to the current state-of-the-art Faster R-CNN, with an astounding inference time of 45 FPS.

YOLOv2 - YOLOv3 YOLOv2 and v3 successively improve some of the drawbacks from YOLOv1. Namely, that the maximum detectable objects per image is 49, upper-bounded by the grid cell size. Moreover, the two possible bounding boxes ($B = 2$) to predict per grid cell, which can only belong to one class, diminishes the detection of multiple small and clustered objects.

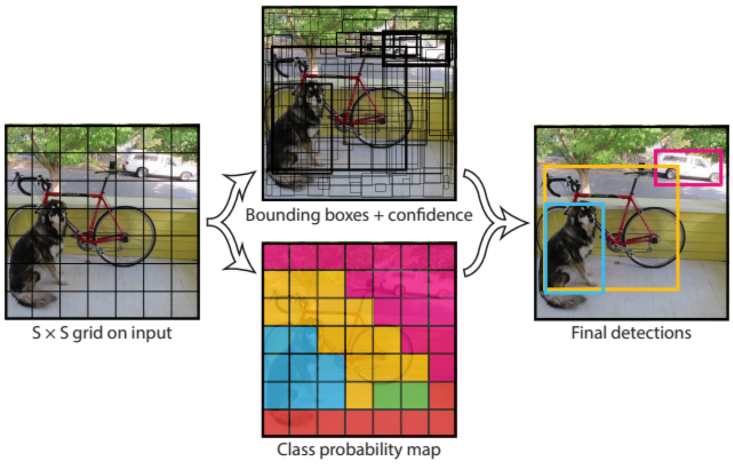


Figure 2.7: YOLO model overview as illustrated in Figure 2 of [83]. ©[2016] IEEE.

Source: [83]

In YOLOv2 [84] (2016), three significant changes are introduced to mitigate the localization errors and additionally reported low recall of YOLOv1. Batch normalization [49] is added as a regularization tool, improving initial mAP with 2%. The Darknet-19 convolutional base replaces the previously used customized GoogleNet, requiring close to 3 billion less operations in a forward pass, while maintaining accuracy. Darknet-19 is additionally in pre-training fine-tuned for 448×448 resolution image classification, allowing filter adjustment and increasing mAP with close to 4%. Anchor boxes are adapted for predicting $k = 5$ bounding boxes per grid cell, k found from a clustering algorithm on the bounding boxes in Pascal VOC 2007 [29], improving recall significantly. Yolov2 outperforms Faster R-CNN on the mAP metric on Pascal VOC 2007, while running inference at 40 FPS, 5-8 $\times$ faster than Faster R-CNN.

In YOLOv3 [82] (2018), several small adjustments are adapted to speed up YOLOv2. YOLOv3 includes the prediction of an objectness score to predicted bounding boxes, with the addition of multi-label class predictions, allowing an object to belong to multiple classes. Multi-scale bounding box prediction for three scales is adopted, with feature extraction similar to FPN [62]. DarkNet-19 is redesigned with the addition of shortcut connections and more convolutional layers into a more accurate feature extractor named DarkNet-53. DarkNet-53 provides more fine-grained information, particularly useful for small object detection. Overall, YOLOv3 is a fast detector, comparable to state-of-the-art detectors for lower IOU thresholds, though struggling for higher IOU thresholds and medium and large size object detection.

YOLOv4 YOLOv4 [16] (2020) reported state-of-the-art results on COCO of 43.5 mAP with inference speed of 65 FPS. YOLOv4 leverage several new concepts to obtain this high score and is designed with the goal of training efficiently on one GPU. A new backbone, CSPDarknet53 is utilized to mitigate the vanishing gradient problem and motivate more robust feature propagation. Feature extraction over different scales in a pyramid approach, is performed with PANet. Also, various different data augmentation techniques are adopted to improve performance. Particularly the new mosaic augmentation, which encourages the detection of small objects by tiling and merging together images. Self-adversarial training, different activation functions, a new NMS technique and cross mini-batch normalization include some of the concepts tested in the development of YOLOv4. The reader is referred to [16] for more details.

Scaled-YOLOv4 [108] (2021) is one of the state-of-the-art object detectors at the writing of this thesis. Scaled-YOLOv4 is based on a similar scaling methodology to EfficientDet, explained in section 2.4.3. Three factors are scaled; image size resolution, the amount of layers and the channels in the backbone, while focusing on high inference speed and accuracy. Dissimilar to EfficientDet, Wang et al. initially conduct depth-scaling before making other scaling adjustment according to real-time inference time requirements. New concepts in designing and scaling CNNs [107] are used to optimize the computation load of different backbones to achieve this.

Upon its publishing, the scaled-YOLOv4 large model, achieved the highest reported COCO mAP of 56.0. For different scaled-YOLOv4 models of similar accuracy to EfficientDets, the scaled-YOLOv4 versions are significantly faster.

2.4.3 EfficientDet

The one-stage EfficientDet [101] family of detectors is tightly connected to the backbones with very similar name, EfficientNet [99]. In line with reason trends in the computer vision community, these architectures address the issue of designing feature extractors and detectors capable of achieving the same or better accuracy of predecessors, but with fewer network parameters and Floating Point Operations (FLOPS).

EfficientNet CNNs or *ConvNets* typically trade-off their accuracy by the available computational resources and constraints posed on inference time. The EfficientNet [99] (2019) family of feature extractors are designed as to provide different backbones with high classification accuracy, fewer network parameters and implicitly faster inference time.

The recent development of ConvNets results in more accurate but significantly larger architectures. Comparing GoogLeNet [98] with the more recent and accurate GPipe giant neural network AmoebaNet [46], the network parameters have increased almost by two orders of magnitude. Moreover, common approaches for increasing classification accuracy in ConvNets are based on increasing depth [98] [39], width [100] or resolution [46]. While each of these scaling factors have their benefits, the combined effect of scaling all factors simultaneously is not extensively researched.

Tan and Le point out that increasing one dimension, such as resolution, also intuitively requires more depth, to adapt the receptive fields, and more width, to capture fine-detailed features of the increased amount of pixels. A neural-architecture search [100] is performed to establish an baseline architecture, EfficientNet-B0, weighting all these three scaling factors. By simply changing a compound scaling coefficient ϕ, the user can select a more accurate and parameter-heavy architecture, ranging from B0-B7.

The largest scaled architecture, EfficientNet-B7, achieves state-of-the-art top-1 accuracy of 84.3% on ImageNet, with 8.4 x times fewer parameters and 6.1 x faster inference time to previous best performing CNN, proving the usefulness of compound-scaling. Additionally, from class activation maps, compound scaling is found to activate features more explainable of complete object regions, than for single-factor scaling.

EfficientDet The EfficientDet [101] (2020) family of detectors further integrates the same compound-scaling methodology of EfficientNet, to design a set of efficient and highly accurate object detection architectures. In the design process, Tan et al. contribute with two progressively new concepts.

Firstly, previous methods in multi-scale feature fusion, based on the original FPN architecture [62], are extended. In particular, repeated blocks of bidirectional pathways form a weighted Bi-directional Feature Pyramid Network (BiFPN). Only input nodes of multi-level edges are kept in the input to each block, with an additional edge connection from the input nodes to the output nodes. More interestingly, all node edges contain weights to also learn the feature-fusion from different feature levels (P3-P7), implemented in a modified softmax function during training.

Secondly, the EfficientNet backbone is integrated together with the BiFPN, forming a complete object detection architecture compound-scalable over the EfficientNet backbone, weighted BiFPN and a box and classification network. Similar to EfficientNets the EfficientDets range from D0-D7, with an additional D7x model using a larger backbone and also the P8 feature level. D0 has an image input size of 512×512, while D7 is inputted images of 1536×1536 resolution.

Overall, the EfficientDet detectors prove to be highly efficient in terms of model parameters while providing a much smaller amount of FLOPS compared to previous state of the art detectors. The EfficientDet-D7 architecture achieves 55.1 mAP on COCO test-dev, with up to 9x times fewer model parametes and 42x fewer FLOPs than competing state-of-the-art detectors on publishing in

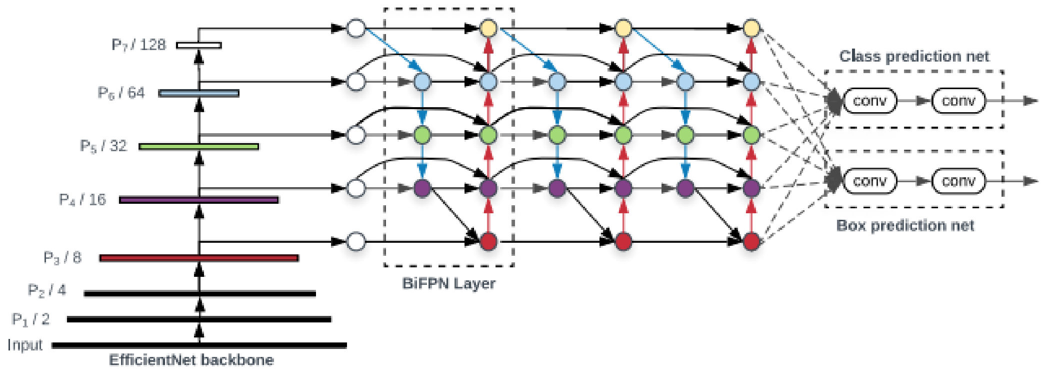


Figure 2.8: EfficientDet architecture. ©[2020] IEEE.

Source: [101]

2020. The complete architecture is displayed in figure 2.8

2.4.4 CenterNet

CenterNet [28] (2019) introduces a conceptually new manner of performing object detection. Upon its publishing in 2019, CenterNet exceptionally outperformed all comparable one-stage detectors with over 4.9 percentage points on the COCO test-dev, which is considerable in a research field moving as fast as computer vision.

The CenterNet architecture can be seen as an extension of previous work in paired keypoint object detection. In 2018, the CornerNet [58] architecture was published as a remedy to former object detection architectures' dependency on anchor boxes. Instead of using a fixed amount of anchor boxes guided by the regression process over ground truth objects, an object is detected and represented as a pair of keypoints. The top-left and bottom-right corners are detected in separate modules with a corresponding class heatmap and an embedding distance vector deciding if the keypoints constitute an object or not. As Duan et al. point out, the corner pooling layer utilized in CornerNet detects corner features consistently by trading-off a lower degree of visual context understanding of the objects.

Duan et al. extend the CornerNet architecture by detecting keypoint triplets, to better perceive the center part of proposed objects and include more global context understanding of the visual object information. Intuitively, a high class-prediction for paired keypoints to contain a class, should likewise be true for the center point of the object. Two new two-directional pooling layers are also introduced to better generate more characteristic features in proposals and the internal perception of these features for the central parts of the object.

CenterNet2 [114] (2021) was published at the writing of this thesis, achieving an mAP of 56.4 on COCO test-dev, outperforming both EfficientDets and YOLOv4-CSP. Due to time-constraints, CenterNet2 is unfortunately not treated in detail.

2.5 Performance metrics

In object detection, another layer of complexity is added in assessing model performance compared to the traditional machine learning classification task. Since the objective is not only classifying an object in the scene but also localizing it, the performance metric needs to somehow measure the localizing predictive capacity of an object detection model.

To define a performance metric for the classification and localization of an object thus requires; a clearly defined localization format predicted from the detection-model, a strict classification of the prediction's validity, and a methodology for assessing the prediction's performance.

In recent years, the format for localizing objects is almost exclusively recognized as a rectangle enclosing the predicted localization of the object, referred to as a *bounding box*. To quantitatively say something about this bounding box prediction, we need to compare it with the *ground truth*. The ground truth is essentially the supervised in supervised-learning; it is the true *label* which we compare the predicted label with. For the object detection task, the label is the bounding box. Thus, we have two set of bounding boxes: the inferred bounding boxes from the object detector, and the true bounding boxes, or the ground truths.

Starting from here, how is it possible to infer knowledge about the performance of a predicted bounding box compared to the corresponding ground truth bounding box?

An intuitive method would be to compare how much the inferred bounding box overlaps with the true bounding box. If there is no overlap, inherently this prediction would be a false observation of an object. Contrarily, if it completely overlaps, it must be a true observation of the object. This concept is often referred to as **Intersection over Union (IOU)** in object detection and is important to give a measure of the predicted bounding box's overlap to the ground truth, and separate between true and false observations of objects.

$$IOU = \frac{A_t \cap A_p}{A_t \cup A_p} \tag{2.1}$$

A_t, A_p denotes the true and predicted bounding box area. IOU therefore returns a number in range from zero to one.

However, to infer more information about the object detector's detection performance, a more direct classification is desirable. If an IOU threshold is set, where all IOU values below and above are treated separately, four different detection classifications are possible.

- *True-positive (TP)*: The predicted bounding box is above the IOU threshold and correctly predicted the localization of an object.
- *False-positive (FP)*: The predicted bounding box is below the IOU threshold and falsely predicted the localization of an object.
- *False-negative (FN)*: It is not predicted a bounding box, while there exists a ground truth object in the scene. Missed detection.
- *True-negative (TN)*: It is not predicted a bounding box and there is no ground truth object in the scene.

Intuitively, TNs are normally diregarded as they do not contribute to information about the detector's performance. Some visualized examples of the different detection classifications are shown in figure 2.9.

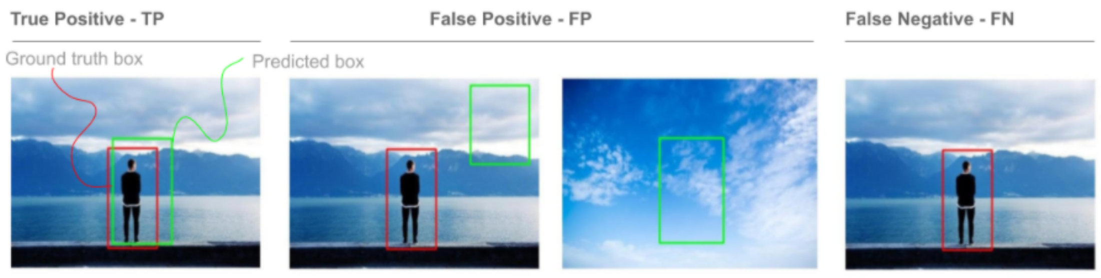


Figure 2.9: True positive, false positive and false negative examples. Red is ground truth, green is predicted bounding box. Courtesy of M. E. Aidouni.

Source: [12]

2.5.1 Precision-recall

Precision and recall are two metrics defined from the different classifications of detections in list 2.5. They are defined as follows:

$$precision = \frac{TP}{TP + FP} = \frac{TP}{\sum_i DO_i} \tag{2.2}$$

DO_i represents a detected objected. Thus the denominator represents the number of detected objects returned from the detector.

$$recall = \frac{TP}{TP + FN} = \frac{TP}{\sum_i GT_i} \tag{2.3}$$

GT_i represents each ground truth object. So, the denominator represents the number of ground truth objects.

It can be observed that the precision and recall are interlinked. Ideally, a precision and recall of one would represent a perfect detector, where all predicted bounding boxes match the ground truth boxes and no ground truth boxes are overseen. It is however more common to find a trade-off between the two metrics.

On the one hand, having a high precision with low recall implies a detector very accurately localizing its detected objects, though often missing objects in the scene. On the other hand, having a high recall with low precision would oppositely imply a detector rarely missing objects but also rarely detecting and localizing them in a precise manner, with substantial amount of FP predictions.

Inherently, the precision and recall scores are dependent on the detector's *confidence threshold*. That is, the model's confidence in observing an object of a class. Only predictions with a confidence score above the confidence threshold are taken into consideration in calculating the precision and recall.

2.5.2 Precision-recall curve

As discussed in section 2.5.1, different confidence scores from the object detection model will give different precision and recall. The *precision-recall curve*, summarizes the precision and recall pairs for different confidence score thresholds.

The best precision-recall curve is by definition the curve consecutively trading-off high precision values for the same recall values. In other words, the curve centered towards the upper-right corner. For instance, the blue curve is considered better than the green curve in figure 2.10.

2.5.3 Average precision

The precision-recall curves are less quantitative when comparing several models' performance. The curves can intersect each other, be noisy and often tend to have a saw-tooth shape. Deciding on the best curve is therefore not necessarily conclusive.

Another popular metric using the precision-recall curve to output only a single number is the *average precision* or *AP*. AP averages the precision correspondingly for all recall values, and can be seen as the area under the precision-recall curve.

Because of the often saw-tooth behaviour of the precision-recall curve, the calculation normally requires an interpolation step, previously done in an 11-point interpolation for each 0.1 recall value, for instance in the Pascal VOC 2008 competition [29]. The green area in figure 2.10 presents an example of calculated AP based on more recently developed method; sampling the recall value in rectangular blocks, every time the maximum precision drops, which provides a more accurate approximation.

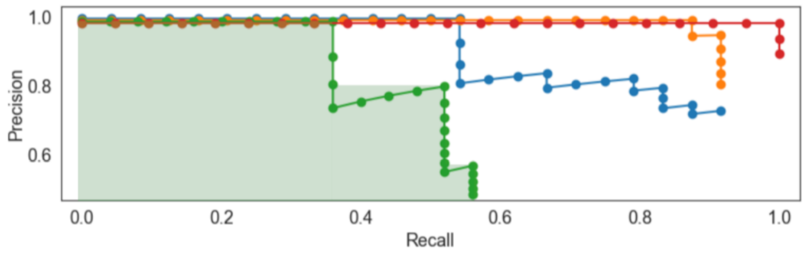


Figure 2.10: Precision-recall curves. The AP for the green curve is denoted by the green area under the curve.

Source: [43]

2.5.4 Mean average precision

The mean average precision, or mAP, is simply the mean of the AP of all classes. Inherently the metric is informative for the overall detection performance over all classes in the dataset. For class-imbalanced datasets the metric is useful, not weighting biases due to the restricted area size of one under the curve. The metric is commonly used for different IOU thresholds T, denoted as mAP@[T]. The main metric in the COCO detection challenge [61] is the mAP@[0.5:0.05:0.95], averaging the mAP score over IOU thresholds from 0.5 to 0.95 with a 0.05 increment. Thus providing a measure which penalizes poor localization over a representative interval of thresholds. As the reader will observe, the conducted experiments in this project thesis are mainly evaluated by the COCO metrics as explained in section 5.5.

2.5.5 Average recall

Similar to recall as for precision, there exists an average recall, or AR, metric. The metric is calculated over several IOU thresholds, typically starting from 0.5 since detection performance is proved to correlate strongly with AR for recall values over 0.5 [44]. AR is a useful metric for measuring high recall and fine object localization.

CHAPTER 3 RELATED WORK

The literature survey in this chapter aims to explore two different topics.

Firstly, it is considered important to understand the typical situations encountered in a maritime environment, further motivating an exploration of optical image maritime datasets. In addition, maritime object detection research is presented.

Secondly, transfer learning is commonly adopted in computer vision. We motivate a review of the theoretical background and research related to object detection.

3.1 Maritime environment

It is important to understand the domain where we want to deploy an object detector. The object detection task in a maritime environment is a computer vision problem phrased in challenging conditions. A thorough analysis of the difficulties of video object detection in maritime environments is discussed in [78]. The paper discusses the challenges of separating between foreground objects and background in the presence of the dynamical nature of the ocean, the movements of the optical sensor and challenging detection conditions (fog, rain, substantial wakes, high correlated nature of waves and more).

For the detection task in still-images, some of these challenges would be less present, as for instance spatio-temporal correlation of background between frames. Nevertheless, the environment naturally poses challenging conditions for an object detector to perform robustly. Classical computer vision methods typically aim to perform a background modelling or horizon-detection for object detection [78][112]. Comparisons of background subtraction methods in video [80] has shown how the methods potentially struggle to generalize to new data and precisely model the foreground - background separation. Even though object detectors can perform well to high-quality tailored datasets, their robustness to new real-world images can not be guaranteed. This can be linked to the lack of an established benchmark dataset for computer vision in maritime environments.

Some challenges to be tackled in a maritime environment are:

- *Clustered objects*: A grouping of vessels, typically in the horizon (background) or tight groupings of small vessels close to the camera.
- *Blurred images*: Weather conditions such as rain or fog, causing undesired blur in the image.
- *Lens occlusions*: Water droplets on the lense, occluding objects in the image.

- *Lens flare*: Scattered light in the lens, caused by strong reflections in very bright conditions. Typically caused by reflection from the waves or low-hanging sun conditions.
- *Viewpoint changes*: Over-represented point-of-view of vessels in the dataset. E.g front-side of boat might occur less than its side.

3.1.1 Maritime detection

Due to the complexity of maritime environments, deep learning methods achieve state-of-the-art results for the object detection task [70]. Moosbauer et al. tackled the object detection task in maritime environments by training the Faster R-CNN and Mask R-CNN architectures on the bounding box annotated SMD dataset [79]. Other work also utilize the R-CNN family of detectors [67] [102]. Tangstad trained a Faster R-CNN architecture for ship detection in a collision avoidance setting. One example of work using multiple types of deep-learning based detectors is Grini [35]. Grini trained two different one-stage detectors on boat and building detection in maritime environments. With exception of [70], the other presented papers depend on either scraping boat images from large universal datasets or custom designing datasets, resulting in generally small-scaled datasets.

Hammedi et al. [37] further explore maritime object detection in channel and fluvial environments. Five one-stage detector architectures as well as the two-stage Faster R-CNN are benchmarked on a custom labeled dataset of 2,488 images, labeled for various object types such as vessels, riverside, and persons. Faster R-CNN reports the best performance on the dataset, though not achieving real-time performance or generalization to the riverside class, which is the majority class of the dataset. Even though their results provide a solid foundation for comparison of different detection architectures, the dataset is inherently class imbalanced and the reported results do not provide a thorough comparison among the architectures.

Mask R-CNN [40] was also deployed in a study comparing detection performance of an unmanned bridge with a human navigator for ship navigation [15]. Blanke et al. found the Mask R-CNN to detect and classify objects 24s faster than its human counterpart on average, quantified from recorded eye fixation of an object lasting over 100 ms, by using a specialized eye-tracking software. Figure 3.1 demonstrates one such example. The utilized dataset of 517 images includes two classes; *buoy* and a generic ship class. Even with train time augmentation, the dataset is considered very small for fully fine-tuning the Mask R-CNN, possibly causing some overfitting behaviour explanatory of the high reported mAP scores.

For increased classification performance and detection of small objects, proven to be particularly difficult, Blanke et al. pinpoint the need for more training data to improve the detector's performance in certain situations. In conclusion, Blanke et al. propose the detection system as an additional fifth sense for a human navigator, and in particular highlights the importance of such an electronic outlook detection system for objects such as kayaks and sea boats, which do not have radar reflectors or AIS transmitters on board.

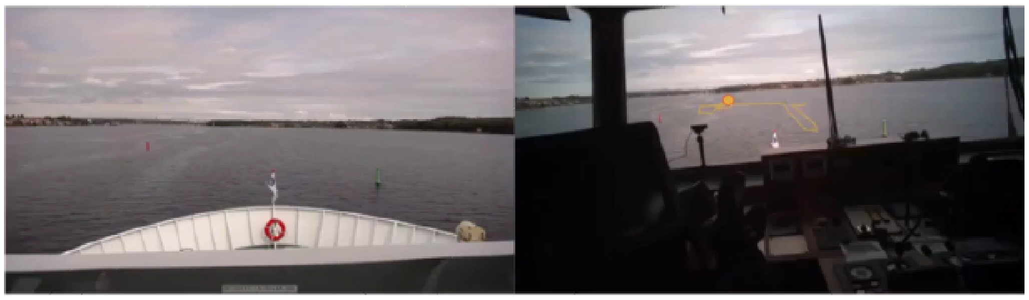


Figure 3.1: Human navigator visualized eye tracking of objects [15]. Right frame marks fixated object in yellow with red boundary.

Source: [15]

As specified in [70], due to the lack of a common maritime benchmark dataset, benchmarking performance of different research is difficult. Based on this research, we motivate a more thorough review into maritime environment datasets.

3.1.2 Maritime environment datasets

Large datasets are considered important in enabling DCNNs to generalize well to a domain. However, large datasets for object detection and segmentation requires large amount of effort to annotate. Particularly, annotated datasets of maritime vessels are rare and hard to find. Unlike facial detection or pedestrian detection, maritime vessel detection has no large public available benchmark dataset. Extensive annotated marine datasets, such as VesselID-539 [27], proves to be a significant contribution for future research. However, since the dataset mostly contains images of reoccurring large cruise and container-ships we look further for a more diverse dataset. The Marvel dataset [36], used in maritime vessel recognition [97] comprises of 2 million boat images of collected on the web. Indeed the Marvel dataset creates a good benchmark dataset for maritime vessel classification and recognition. However, it lacks bounding box annotations and is not very class diverse. Online databases of ship images exist [4] for web-scraping. This goes beyond the scope of this project.

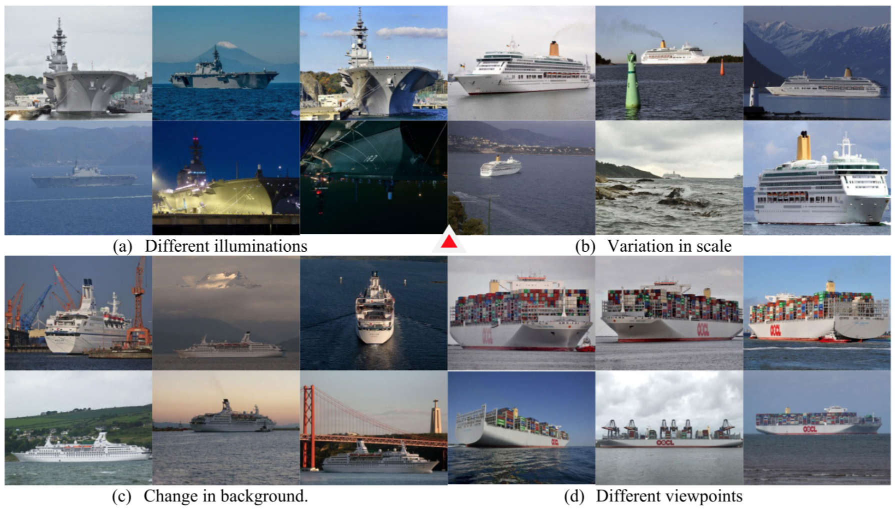


Figure 3.2: Example image VesselID-539. ©[2020] IEEE.

Source: [27]

The most established benchmark dataset in maritime environments, also containing bounding box labels is, to the knowledge of the author, the SMD [79]. The SMD spans over 30,000 thousand bounding box labeled frames collected from Singapore maritime waters. This dataset has additionally been treated in an attempt to establish a maritime benchmark dataset, with a distinct train, validation and test set by [70]. The SMD is treated in detail in section 4.4.

3.2 Transfer learning

Transfer learning is a field within machine learning. Transfer learning includes a variety of techniques for reusing previously learned knowledge and apply it in solving a new problem. Generally, such techniques are indispensable for machine learning problems where the data might be biased or scarcely available, motivating the reuse of similar datasets with different labels, distribution, or prediction task.

To illustrate, one might imagine a classifier intended to learn and predict different vehicle classes.

Unfortunately, suitable training data is hard to come by and it is preferable to avoid the manual sampling and annotation of images. A classifier already trained to recognize generic vehicles may be adapted to reuse knowledge and successfully generalize to a small amount of images of different vehicle classes.

In this section a theoretical review of transfer learning is conducted, before presenting related work in transfer learning for DCNNs and object detectors.

3.2.1 Transfer learning background

In a machine learning problem, training is performed on the training set to learn a predictive function, which is presumably capable of generalizing to new samples introduced in the test set. However, if either one of the datasets is biased or unrepresentative of the underlying dataset distribution, prediction performance suffer and generalization to new samples in the test set is questionable.

In object detection, the prediction task is the classification and localization of objects. There are several scenarios where training to test set dissimilarity may occur. For instance, the training set could include a large weight of small boat types while the test set includes mostly large ships. Moreover, specific background and weather conditions may be overrepresented in one of split sets. As follows, the *learning task* of predicting boats is the same, but the training and test sets are not representative of one another. What is more, if the test set includes more fine-grained boat classes, while the train set treats all boats in an agnostic manner (one generic class), the labels are additionally no longer similar.

Transfer learning essentially addresses all of the described problems. The training and testing set is used to illustrate the transfer learning formulation, however we underline that the following definitions are far more general. The reader is encouraged to read the remaining part of this section in parallel with the notations defined in table 3.1. All notations are defined in accordance with one of the corner-stone surveys in transfer learning [74].

Let each dataset in the above-described setting be representative of a *domain*, $\mathcal{D}$. The training set constitutes the *source domain*, $\mathcal{D}_S$, while the test set constitutes the *target* domain, $\mathcal{D}_T$. Each domain is composed of two different constituents; a feature space $\mathcal{X}$ and a marginal distribution $P(X)$ over the feature space. A general domain is therefore defined as $\mathcal{D} = \{\mathcal{X}, P(X)\}$.

In order to finalize the necessary terminology for the machine learning model, the well-known prediction function and labels must be defined. In the context of transfer learning, these definitions relates to the *learning task*. A task $\mathcal{T}$ thus includes a label space $\mathcal{Y}$ and a prediction function $f(\cdot)$ such that $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$. The prediction function is learned from the data samples with the corresponding labels, which pairwise are represented as $\{x_i, y_i\}$ for $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. The predicted label for a newly introduced data sample x is thus $f(x)$, which also can be phrased as a conditional distribution over the data sample in a discriminative learning setting, $P(y|x)$. Where the conditional distribution represents the decision boundary between datapoints in the feature space.

Finally, the source and target domain can generally be defined as follows.
$\mathcal{D}_{\mathcal{S}} = \{(x_{S_1}, y_{S_1}), \cdots, (x_{S_{n_S}}, y_{S_{n_S}})\}$ where $x_{S_i} \in \mathcal{X}_S$ and $y_{S_i} \in \mathcal{Y}_S$. Similarly, the target domain is defined as $\mathcal{D}_{\mathcal{T}} = \{(x_{T_1}, y_{T_1}), \cdots, (x_{T_{n_T}}, y_{T_{n_T}})\}$ where $x_{T_i} \in \mathcal{X}_T$ and $y_{T_i} \in \mathcal{Y}_T$. Typically, the amount of samples in the target domain, n_T is substantially smaller than the source domain samples n_S. Transfer learning is defined as follows in [74].

Definition 3.2.1 (Transfer learning). "Given a source domain $\mathcal{D}_S$ and learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and learning task $\mathcal{T}_T$, transfer learning aims to help improve the learning of the target predictive function $f_T(\cdot)$ in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$, or $\mathcal{T}_S \neq \mathcal{T}_T$." [74]

Based on the transfer learning in definition 3.2.1, there are generally four possible transfer learning settings.

Notation	Description	Notation	Description
$\mathcal{X}$	Input feature space	$P(X)$	Marginal distribution
$\mathcal{Y}$	Label space	$P(Y\|X)$	Conditional distribution
$\mathcal{T}$	Learning task	$\mathcal{D}_S$	Source domain
Subscript S, T	Source, Target	$\mathcal{D}_T$	Target domain

Table 3.1: Transfer learning notation.

1) $\mathcal{D}_S = \mathcal{D}_T$ and $\mathcal{T}_S = \mathcal{T}_T$
2) $\mathcal{D}_S \neq \mathcal{D}_T$ and $\mathcal{T}_S = \mathcal{T}_T$
3) $\mathcal{D}_S = \mathcal{D}_T$ and $\mathcal{T}_S \neq \mathcal{T}_T$
4) $\mathcal{D}_S \neq \mathcal{D}_T$ and $\mathcal{T}_S \neq \mathcal{T}_T$

The first case defines a *traditional machine learning* problem. In other words, the source and target domains are equal and the learning task is the same. However, the following cases define various possible transfer learning settings. Since a domain is defined as $\mathcal{D} = \{\mathcal{X}, P(X)\}$, either the feature space $\mathcal{X}$ or the marginal distribution $P(X)$ are unequal when $\mathcal{D}_S \neq \mathcal{D}_T$. Similarly for the learning task, $\mathcal{T} = \{\mathcal{Y}, P(Y|X)\}$, either the label space, $\mathcal{Y}$, or the conditional distribution, $P(Y|X)$, are different when $\mathcal{T}_S \neq \mathcal{T}_T$.

Case 2 - 4 each define a transfer learning setting in the literature. Pan and Yang, use data availability in source and target domains as a defining factor for each transfer learning setting. There seems to be disagreement in the literature of the importance of data availability in each domain [109]. For simplicity, the remaining part of this section do not take data availability into account.

Case 2 defines a transfer learning setting thoroughly explored in the literature [109], namely *Domain adaptation* (DA), which is in-detail covered in section 3.2.2.

Case 3 is defined as *inductive transfer learning*, while case 4 is defined as *unsupervised transfer learning* by [74]. Both of these cases experience a change in the learning task and have stricter requirements for successful transfer learning. As these cases are less relevant for this thesis, they are not treated in detail.

3.2.2 Domain adaptation

DA is one of the special cases of transductive transfer learning [74]. Formally, defined as follows:

Definition 3.2.2 (Transductive transfer learning). "Given a source domain $\mathcal{D}_S$ and learning task $\mathcal{T}_S$, a target domain $\mathcal{D}_T$ and learning task $\mathcal{T}_T$, transductive transfer learning aims to improve the learning of the target predictive function $f_T(\cdot)$ in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ and $\mathcal{T}_S = \mathcal{T}_T$."[74]

An unequal source and target domain implies either that $\mathcal{X}_S \neq \mathcal{X}_T$ or that $P(X_S) \neq P(X_T)$. DA is essentially defined by a changing marginal distribution between the source and target domain; $P(X_S) \neq P(X_T)$. Thus, strategies in DA aim to overcome the distribution differences in the source and target domain to enable generalization to the target domain predictive function $f_T(\cdot)$. To further shed light on what a changing marginal distribution signifies, some common *domain shifts*, also referred to as *data shifts*, must be explained.

Assuming a source domain's marginal distribution $P(X_S)$ is unequal to the target domain marginal distribution $P(X_T)$, the conditional distributions of the learning tasks may still be equal, ensuring a similar learning task. In other words, $P(Y|X_S) = P(Y|X_T)$ such that $\mathcal{T}_S = \mathcal{T}_T$, where we have assumed the same label space $\mathcal{Y}$ for both domains. This is referred to as a *prior shift* [109].

However, this is rarely the case. As follows, DA typically needs to handle the case where $\mathcal{T}_S \approx \mathcal{T}_T$ due to a *covariate shift* in the source and target domain. A covariate shift implies $P(Y|X_S) \approx P(Y|X_T)$ or $P(Y|X_S) \neq P(Y|X_T)$, often occurring as a result of a *sample-selection bias*. A sample-selection bias is the result of collecting data from various data sources, missing data samples or by introducing other biases in the dataset creation, for instance favoring the representation of certain samples based on subjectivity. As such, the source domain may be tailored to fit a particular distribution and do not well represent generalization for samples in the target domain.

As pointed out by Shimodaira [94], a machine learning model trained on a source domain does not perform optimally on a target domain with a different marginal distribution. The following section presents different manners of conducting transfer learning and DA for DCNNs and object detectors.

3.2.3 Fine-tuning & pre-training

In this section, novel and recently researched methods in transfer learning for DCNNs and object detection architectures are presented.

Firstly, it is of interest to assess transfer learning strategies conducted by altering the training procedure of the architecture in question. Usually, such methods differ by the amount of parameters which are subject to re-training and those who are *frozen*, or not subject to training. We refer to this topic as *fine-tuning strategies*. Such methods are characterized as *supervised DA* methods.

The second topic to investigate is *pre-training*. Pre-training is commonly adopted before fine-tuning on smaller-scale target data. Our objective is to summarize common pre-training paradigms on the classification and object detection task.

Fine-tuning strategies

Fine-tuning and pre-training are inherently linked. DCNNs and object detector backbones are typically pre-trained on large-scale classification datasets. The pre-trained weights are thereafter used for weight initialization, followed by another subsequent training cycle which *fine-tunes* into the target domain, represented by a small-scale dataset.

Depending on the source and target task similarity, as well as the label space, it might be necessary to entirely replace the top layers of the pre-trained DCNN. We refer to such operations as *off-the-shelf methods*, which essentially replaces some of the layers (shelves) before fine-tuning the resulting architecture.

Fine-tuning strategies differ in the amount of frozen parameters and layers in the DCNN. For instance, in object detection architectures it is possible to freeze the entire convolutional base, or backbone, and only fine-tune the classification and bounding box heads. Additionally it is possible to adapt fine-tuning into different modules of an object detection architecture.

In this section we mainly separate between two popular strategies.

- *Full fine-tuning*: All layers and parameters are subject to update during training.
- *Gradual fine-tuning*: The lower level layers are frozen while the deeper layers unfreeze sequentially from the deepest to more shallow layers.

Fine-tuning background Fine-tuning is based on the transferability of learned features in DCNNs. An architecture pre-trained on a large-scale dataset has already learned more generic lower-level features, which have been found to be representative for many datasets and tasks [111]. The pre-trained architecture is thereafter subject to fine-tuning on a new dataset. The fine-tuning is essentially necessary for adapting the deeper layers, which captures more specific and complex object features relevant for the task and the new domain.

Yosinski et al. [111] highlight the importance of the source and target task similarity when fine-tuning a DCNN. If the source and target task are very distant, reusing pre-trained features are significantly less efficient, though still better than random initialization in some cases. Moreover, initializing from pre-trained features may provide a boost in performance which lasts even after fine-tuning into the new domain. On the other hand, techniques which do not train all layers may risk to deteriorate performance due to the co-adapted interactions between neurons. In conclusion, the findings of [111] are also very relevant for object detection architectures, which inherently depend on the feature extraction of the DCNN backbone module.

Chu et al. [23] conduct extensive experiments for assessing the suitability of different fine-tuning settings for transferring features in DCNNs. Based on their selected six target datasets, various dataset splits and different freezing and fine-tuning settings, a total of 138 experiments are conducted. The overall results from the conducted experiments provide two main findings.

Firstly, it is found that keeping as many layers as possible from a pre-trained model is the most beneficial. In other words, randomly initializing lower level layers may upper-bound model performance, compared to fully fine-tuning copied layers.

Secondly, fully fine-tuning of the copied layers is better than freezing and particularly when more target data is available. The more distant the source and target domain, the more inferior is freezing layers. Moreover, freezing may be beneficial for smaller target domain datasets, if the source and target domain distance is small. In their experiments, the domain distance is measured amongst other as the cosine distance of the source and target datasets' fully connected layers' mean response. In conclusion, a fully fine-tuning scheme is the simplest and most reliable fine-tuning option.

Ouyang et al. [73] provide more insight into the effects of the target domain dataset class-distribution when fine-tuning object detection architectures. With class-distribution, what is meant is the number of samples of each class for the dataset. A commonly encountered problem when collecting and designing target domain datasets is collecting enough samples of each desired class. This can be caused by a natural bias of instances in the foreground, e.g there are more pedestrians than stop signs in the street, or the lack of time to generate a diverse and large enough target domain dataset. Often this problem is handled by performing agnostic object detection [50][70] or merging classes into a superclass [73].

Ouyang et al. explore this phenomena further for datasets with long-tail class distributions, in essence where the majority of the samples are represented by a few classes. In their experiments, training and test sets are created from the large scale ImageNet detection dataset, consisting of 200 object classes. GoogLeNet [98] is adapted for object detection and different fine-tuning settings. Class-imbalanced datasets oversample features from certain object classes, which hinders satisfactory learned feature representation of the underrepresented classes. As follows, Ouyang et al. find that detection accuracy improves when reducing training dataset class instances by as much as 40%, if the class-samples are more evenly distributed.

Furthermore, Ouyang et al. experiment with backbone freezing of selected layers in GoogLeNet [98]. Overall, fine-tuning on the detection dataset is consistently best when all layers are subject to training. The performance likewise decrease with the amount of frozen layers, even though Ouyang et al. point out that it is possible to achieve only slightly worse performance than fully fine-tuning when freezing some of the lower level layers, which remarkably reduces the amount of parameters to train.

Before passing on to the less explored gradual fine-tuning setting, we note that the undoubtedly most common and well-documented fine-tuning strategy is *full fine-tuning.*

Gradual tuning

Montone et al. [68] present one of the novel methods for fine-tuning DNNs while addressing the problematic of catastrophic forgetting. Namely, that during a full fine-tuning training scheme, all parameters are trainable and thusly, the model performs substantially worse on the initial source task. This may or may not be problematic depending on the use-case. Nevertheless, Montone et al. design a fine-tuning scheme that trades-off performance and catastrophic forgetting, retaining performance on the initial task as well as similar performance to full fine-tuning on the target task.

One of the main arguments behind the gradual tuning is based on the following observation. The loss function and its internal neuron weights, which are the basis for training a DNN, heavily depend on interconnected weights through the neurons on different layers in the architecture. As such, the evaluated loss function gradient with respect to one of the weight vectors, inherently is also a function which depends on the manner the weight vector is connected to other neurons in the neural architecture. A full-fine tuning scheme may overwrite a good feature for solving the task, based on its connection to upper layers, which a gradual feature update starting from the upper levels would avoid.

In their experiments, Montone et al., design two different feedforward neural networks with two and three fully connected hidden layers, trained and evaluated on eight variations of the MNIST dataset for two different tasks A and B. In the gradual fine-tuning scheme, the output layer is initially unfrozen before subsequently unfreezing each hidden layer when the networks performance stops increasing. The full-fine tuning scheme simply keeps all layers trainable.

For almost all of the eight dataset experiment settings, both in single and multi task experiments, the catastrophic forgetting was smaller. The performance of the new task, or the target task, is either similar or better which motivates using the gradual tuning scheme also for improved performance at a new task. Even though their results are interesting, the transferability to DCNNs and to more challenging domains is not directly clear. The synthetic modifications of the MNIST dataset are most likely less challenging than in transfer learning settings for RGB images from more different domains.

Gradual fine-tuning has been explored together with synthetic data. Reiersen [85] fine-tunes a detector in a few-shot learning setting to improve target domain performance of real-world images from pre-training on synthetic data. The reported results overall indicate a fully fine-tuning scheme is superior to a gradual fine-tuning of the convolutional base. Henriksen [43] addresses a similar setting for different object classes and a larger real-world image dataset. Henriksen additionally implements a gradual fine-tuning of the entire Mask R-CNN architecture, from the detection and mask heads to the convolutional base. His results indicate better adaptation of domain invariant features from gradual fine-tuning than fully fine-tuning, with marginally improved performance scores.

Pre-training

The most common transfer learning approach for DCNNs and object detectors, is based on the paradigm of large-scale pre-training followed by a subsequent stage of fine-tuning. In particular, from the early developments in deep learning based object detectors, such as R-CNN [34] and YOLO [83], classification pre-training on ImageNet [26] for the feature extracting backbone was established as a common practice. The same practice has been adopted for instance segmentation architectures such as Mask R-CNN [40]. To benchmark performance between architectures, fine-tuning is executed on datasets as COCO [61] or previously Pascal VOC [29].

For the remaining part of this section, the ImageNet classification dataset [26] is referred to as ImageNet-CLS, while ImageNet-LOC denotes ImageNet-CLS with additional bounding box labels.

Image classification pre-training Large-scale image classification pre-training of the feature-extracting backbone in object detection architectures is common practice.

He et al. [41] question the importance of large-scale classification pre-training. The Mask R-CNN architecture is experimented with for different variations of backbones, both with ImageNet-CLS pre-trained weights and randomly initialized backbone weights. Different variations of batch normalization [49] are proposed to mitigate normalization issues for small batch sizes when training from scratch. Model performance is benchmarked by fine-tuning and evaluating on the COCO dataset with the COCO metrics [1] for the detection task. As such, the effect of fine-tuning with and without ImageNet-CLS pre-trained weights is isolated.

Overall, He et al. find that ImageNet-CLS pre-trained weights have a clear effect on the convergence speed, when assessing the mAP validation curve with regard to training iterations, as presented in figure 3.3. On the other hand, the randomly initialized counterpart achieves comparable performance, if trained similar in length to the total ImageNet-CLS pre-training and fine-tuning training. The longer training scheme adapts the learning of more generic low/mid level features typically learned during ImageNet pre-training. Similar observations are done for larger and wider architectural modifications of Mask R-CNN.

Moreover, these observations hold even when using as little as 10% of COCO training data (10k Images), after adapting hyperparameters and training schedule. In addition, there is little performance gains from ImageNet-CLS classification pre-training on the target task when well-localized spacial context is considered. Actually, higher IOU threshold AP metrics improve when the backbone weights are randomly initialized. Indicating, that classification pre-training does not transfer directly or benefits fine-localization of objects.

The results of He et al. question the formerly accepted ImageNet-CLS pre-training, indicating this paradigm has rather been a work-around for the lack of target domain data. Moreover, when the source pre-training and target task are significantly different, acquiring more target data may prove more useful. However, it should also be mentioned that ImageNet-CLS pre-training has not shown to deteriorate target domain task performance. Since the convergence speed is significantly faster with this method, it is still a convenient option for fine-tuning into a target domain dataset if the user does not have access to more data and/or GPU resources. It is not further discussed if the results for smaller-scale target datasets dissimilar to COCO, would report similar results.

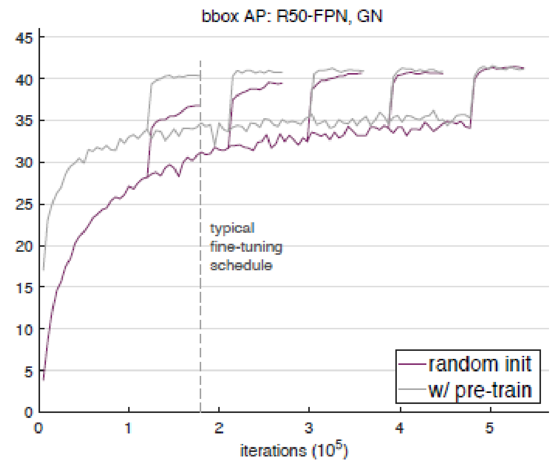


Figure 3.3: Validation AP curves for Imagenet-CLS pre-trained and randomly initialized models. ©[2020] IEEE.

Source: [41]

Even though He et al. quantitatively analyse the mAP metric, other potential effects of training from random initialization and skipping ImageNet-CLS pre-training are not discussed.

This topic has further been explored by Shinya et al. [95]. By analysing the eigenspectrum of the feature maps of different layers of the Faster R-CNN and Mask R-CNN architecture, Shinya et al. highlight important observations regarding the training and generalization to COCO, with and without ImageNet-CLS pre-training.

Firstly, it is found that models which obtain the same accuracy with and without ImageNet-CLS pre-training behave differently. Based on the eigenspectrum analysis, which essentially covers which feature map is responsible for what information at a point in time, there are indications that ImageNet-CLS pre-training features are easily forgotten during fine-tuning. Moreover, based on an intrinsic architecture algorithm which proposes the channel width of an object detector based on its eigenspectra, it is found that ImageNet-CLS pre-trained models generate a more narrow eigenspectrum than the models trained from scratch. The ImageNet-CLS pre-trained models do have an increase in parameters which are not reflected by improved performance on the mAP metric on COCO evaluation.

The usefulness of ImageNet-CLS pre-training is not disregarded, even though Shinya et al. point out the need for better methods for compressing and reusing the most task-relevant features learned in pre-training. In particular, this is an issue since the classification and detection task are very different and information is rapidly lost during fine-tuning.

Object detection pre-training In this section, it is desired to explore the effects of object detection pre-training and its effect when fine-tuning into a target domain for the detection task. We refer to this general concept as *targeted pre-training*. That is, pre-training on the same task which is later desired for fine-tuning. Which in this case is *targeted detection pre-training*. As this topic has become popular in very recent times, we present several newly published research papers.

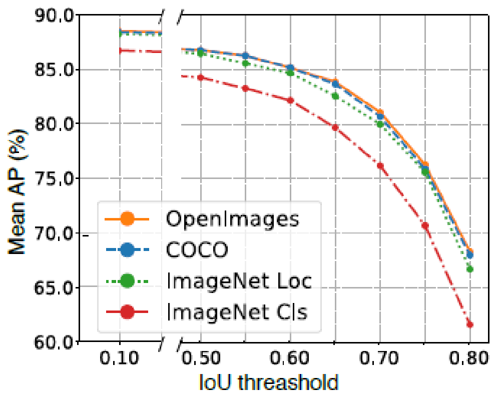
Recent work [60] has more extensively explored the effects of classification and detection pre-training before fine-tuning into a smaller target dataset for three different tasks in computer vision; image classification, object detection and semantic segmentation. Li et al. present some of the novel work on the effects of classification and detection pre-training. The motivation behind comparing these pre-training schemes are two-fold.

Firstly, pre-training on the object detection task has not been as extensively researched as image classification pre-training [34] [41]. Nevertheless, detection pre-training on COCO [61] is still one of the most commonly applied transfer learning settings for training and testing object detectors on small-scale target datasets in most practical applications. One reason that this type of pre-training has not been researched as extensively, is the lack of large-scale object detection training datasets comparable in size to image classification datasets such as ImageNet-CLS. To further shed light on the effect of large-scale detection pre-training, Li et al. utilize the recently published OpenImages [55], which consists of 1.9 million images with 15.4 million bounding box annotated instances.

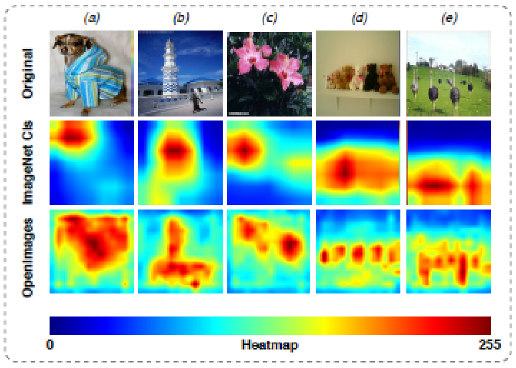
Secondly, detection pre-training both learns the classification and localization of objects. Thus, a network which performs object detection should implicitly also learn richer object features. Li et al. argue that a detection network's access to orthogonal semantic information such as the spatial context of the object encourages the learning of such features. Previous work [31] indicates that CNNs possess a certain level of bias towards texture features, while other work [88] indicates shape features are implicitly learned. Clearly, a more thorough review of the effects of detection pre-training is a very interesting research contribution. Li et al. also apply feature activation maps for visualizing which parts of the object that has the highest activation when detected.

Li et al. conduct their pre-training experiments in the following manner. The Sniper [96] detector architecture is utilized. All detection pre-training experiments are firstly pre-trained on the ImageNet-CLS before subsequently fine-tuned on one of the three different datasets; OpenImages, ImageNet-LOC and COCO. Image classification pre-training is executed on ImageNet-CLS only. For the object-detection task, the Pascal VOC [29] dataset represents the target dataset. For image classification and semantic segmentation, various target datasets are used, which we do not detail report here.

The reported mAP scores for the detection task with different IOU thresholds are presented in figure 3.4a. The most inferior model is the model pre-trained on the ImageNet-CLS. Furthermore, the models pre-trained on object detection datasets prove to generally improve for all IOU thresholds, but particularly for the higher IOU thresholds. This indicates that detection pre-training favours the recognition of spatial object features and mostly benefits detections where fine-localization of objects are important. This observation is supported by the activation map presented in figure 3.4b. The first row of activation maps are pre-trained on ImageNet-CLS while the second row is pre-trained on OpenImages. The OpenImages pre-trained model demonstrates higher neuronal activation for the spatial entirety of the object, while the ImageNet-CLS pre-trained model favours more discriminative object features. Similar observations are made for the semantic segmentation task. Contrarily, object detection pre-training is not found to benefit fine-tuning for the image classification task, where recognizing an object from a subset of features is beneficial.



(a) mAP for different IOU thresholds



(b) Backbone activation for different pre-training, Conv5

Figure 3.4: Object detection pre-training performance results and visualized backbone activations [60]. Courtesy of Hengduo Li.

Source: [60]

Based on the datasets used for object detection pre-training, Li et al. claim that the amount of samples in the pre-training dataset does not affect the detection fine-tuning results notably. The crucial factor is whether the datasets contains approximately above one million instances or not. Furthermore, larger pre-training datasets implicitly perform better for detecting occluded objects.

Shao et al. present several interesting pre-training experiments with the Objects365 dataset [91]. Objects365 consists of over 600,000 training images with over 10 million bounding box labels, and is a useful addition to the partially labeled large-scale detection dataset OpenImages [55]. Additionally, the Objects365 is more diverse than both COCO and OpenImages, with significantly more variation of different object categories per image.

The FPN and RetinaNet detectors are fine-tuned on COCO with pre-trained weights from different combinations of ImageNet-CLS, OpenImages and Objects365. Figure 3.5 presents the mAP scores on COCO for the different pre-training combinations. Overall, training from scratch on COCO converges to the same score as for ImageNet-CLS pre-training. Training from scratch on Objects365 or subsequent detection pre-training on Objects365 after ImageNet-CLS initial pre-training, is largely superior to training from scratch on COCO or with only ImageNet-CLS pre-training. Both in terms of convergence speed and final mAP scores.

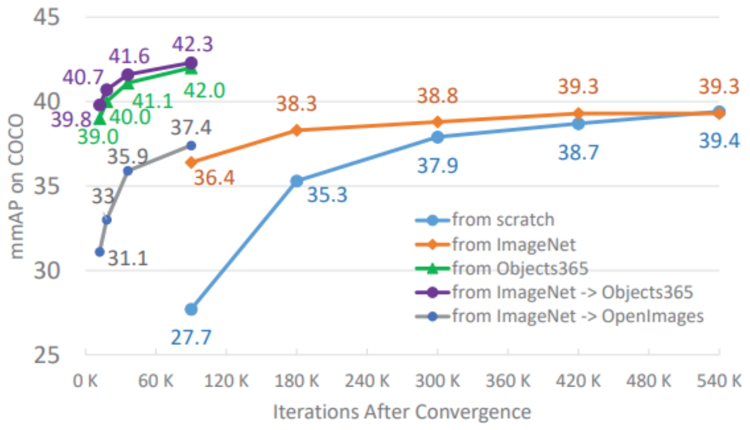


Figure 3.5: Results for fine-tuning on COCO for different pre-training tasks and settings [91]. ©[2019] IEEE.

Source: [91]

To elaborate, after 90,000 fine-tuning iterations, the models pre-trained on Objects365 report 5.6 and 5.9 percentage-point gain on the mAP metric compared to ImageNet-CLS pre-training. Moreover, the Objects365 pre-trained models fine-tuned for 90,000 iterations, outperform the ImageNet-CLS pre-trained models fine-tuned for 540,000 iterations, with 2.7 percentage-points. The findings of Shao et al. indicate that targeted pre-training for detection is highly beneficial for fully annotated datasets. The improved localization ability from the detection pre-training and the diversity of object categories in Objects365 is explanatory for much of the performance gain. In addition, the pre-training of the detector heads, which only occur during detection pre-training, is in the ablation studies found to be a very important part of the improved localization ability.

Weakly-supervised object detection pre-training A more experimental approach which is significantly cheaper for obtaining detection labels to use in pre-training, is designed by Zhong et al. [113]. In their experimental setup, a weakly supervised pre-training algorithm based on class-activation maps is utilized for generating bounding box labels for two versions of the ImageNet-CLS dataset. The pseudo-labeled datasets are subsequently used for pre-training a Faster R-CNN, before fine-tuning it on the *downstream task*, which is represented by the Pascal VOC and COCO dataset in two separate learning settings for object detection.

Compared to the ImageNet-CLS classification pre-trained baselines, the weakly supervised detection pre-trained model consistently performs better on all mAP and AP metrics for both of the learning settings for both versions of ImageNet-CLS. In addition, the detection pre-training regime is found to be stronger either when the pre-training dataset is larger or when the target domain dataset represented by the downstream task, is smaller.

Similar to previous findings in detection pre-training [60], the convergence speed is significantly faster from detection pre-training than classification pre-training, giving a large initial boost on reported mAP. Zhong et al. also find that detection pre-training is a strong cue for adapting the whole network's feature representation more towards detection, and not only the backbone weights as previously thought. This is also logical, considering that in classification pre-training only the backbone is trained, while the remaining parts of an object detector is randomly initialized.

3.3 Summary

Maritime environments prove to pose challenging conditions for detection and the lack of available large-scale annotated datasets do not ease the task. Traditional object detection methods typically use horizon-detection and local background subtraction in maritime environments, often suffering from the dynamical nature of the environment. State-of-the-art object detection models are deep learning based DCNNs. Maritime object detection research are often based on two-stage detectors, such as Faster R-CNN or Mask-RCNN, subject to fine-tuning on custom small-scale target domain datasets [35] [15] [37].

Transfer learning is one of the corner-stones in the evolution of deep learning. Object detection architectures adopt the pre-training & fine-tuning paradigm. Pre-training the backbone on a large-scale dataset is used to learn more general transferable features, while fine-tuning adapts more domain-specific features for solving the fine-tuning task [111]. When fine-tuning for the detection task, detection pre-training has proven more beneficial than classification pre-training of the backbone, providing faster convergence and overall better reported performance scores [60] [91]. This scheme, called targeted detection pre-training, is mostly researched for large-scale pre-training datasets ($>$ 600,000 images).

The transferability of different fine-tuning strategies depend on the source to target domain distance and the target domain dataset size [111]. Full fine-tuning is often superior to fine-tuning strategies utilizing more freezing [23] [73], particularly for dissimilar domains. Gradual fine-tuning of layers in a DCNN or modules in an object detection architecture may prove beneficial [43], however subject to specific hyperparameter considerations of the gradual fine-tuning scheme.

CHAPTER 4 DATASETS

In this chapter, we provide an overview over all the datasets explored in this thesis.

4.1 Grini dataset

In his master's thesis, Grini [35] designed several datasets, based on optical images from Norwegian maritime environments, including Trondheimsfjorden. Grini annotated a total of 1,916 images, later divided into separate datasets.

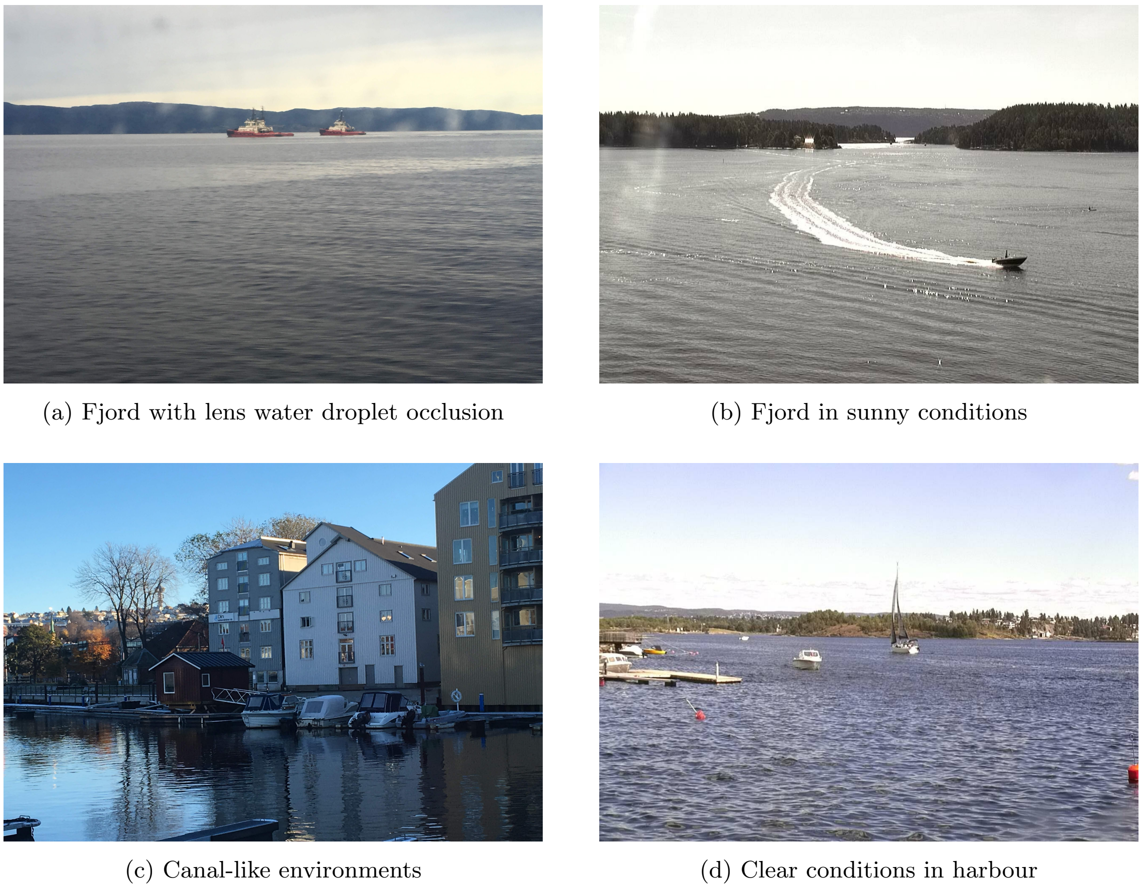
All data is made available for this thesis. The delivered data by Grini is delivered in used train and test splits with slightly different folder names for the datasets than reported in his thesis. We do our best efforts for matching folder names consistently. Table 4.1 presents the amount of annotated images and instances per dataset, after filtering out images without existing annotation files and vice versa.

Dataset summary			
Dataset	**Annotated images**	**Boats**	**Buildings**
Trondheimsfjorden	337	482	-
BoatsFar	1214	3413	-
BoatsClose	35	90	2
MooredBoats	107	300	263
Total	1693	4285	265

Table 4.1: Labeled images and instances in Grini dataset.

As a first observation, the boat annotations labeled by Grini are class agnostic with regard to different boat-types. That is, the boat annotations are solely labelled *boat* without separating between different boat-types. As such, it is empirically observed a variety of different boat types, kayaks and sailboats with, and without sail, in the generic boat class. The Trondheimsfjorden and MooredBoats are the only datasets which contain images from Trondheimsfjorden and the Nidelven river.

Some example images for different environments and conditions are shown in figure 4.1. The sample images nicely display the diversity of the Grini dataset.


(a) Fjord with lens water droplet occlusion (b) Fjord in sunny conditions

(c) Canal-like environments (d) Clear conditions in harbour

Figure 4.1: Example images from Grini dataset for different conditions

4.2 Brekke & Lopez dataset

The Brekke & Lopez dataset was collected by Edmund Brekke and Michael Ernesto Lopez in 2019. The annotated dataset consists of images in the VIS and infrared (IR) spectra, captured on board the Norwegian ferry Hurtigruten, mostly in open-sea conditions and on docking, as well as more city-like environments from canals in Amsterdam. We refer to the sub datasets as Hurtigruten$_{BL}$ and Amsterdam$_{BL}$.

The two sub datasets provide two different sets of class labels, one fine-detailed set, consisting of 39 classes, and a more simplified version. The majority of the 39 classes account for very few annotated objects, thus the simplified labels are adopted. The simplified labels consist of nine unique classes, as presented in table 4.2.

Class-labels
Airplane
Barge
Building
Helicopter
Kayak
Motorboat
Motorboat with priority
Sailboat with sails down
Sailboat with sails up

Table 4.2: Simple class-labels in Brekke & Lopez dataset

Since only the visual spectrum images are of interest for this thesis, the IR images are disregarded in our exploration of the two sub datasets. An overview of the annotated images is presented in table 4.3. The Hurtigruten$_{BL}$ dataset accounts for the large majority of the annotated images.

Annotated images			
	Hurtigruten$_{BL}$	**Amsterdam$_{BL}$**	**Total count**
Annotations	713	114	827

Table 4.3: Overview visual spectrum images Brekke & Lopez dataset

Moreover, we present an overview of the simplified class labels for both of the sub datasets in table 4.4. As a first observation, the Brekke & Lopez dataset is strongly class-imbalanced and consists of several classes which are not boat-types. In particular, the *Airplane*, *Catamaran*, *Helicopter* and *Kayak* classes have very few instance samples.

The non boat-type classes include different types of flying vehicles, such as *Airplane* and *Helicopter*. In addition, a large amount of buildings are labeled, typically appearing close to harbours and on docking.

The boat-classes are overrepresented by motorboat samples, labeled according to their location in the scene. In total, there are 1,518 motorboat samples. The sailboats instances, separated by the presence of raised sails, are not noteworthy. Lastly, the *Kayak* class is practically not represented in the dataset with a total of eight instances.

In conclusion, the Brekke & Lopez dataset is most suitable to treat in a class-agnostic manner with regard to boat-types. Some samples from the respective sub-datasets are visualized in figure 4.2.

Class-labels summary			
Class	**Hurtigruten$_{BL}$**	**Amsterdam$_{BL}$**	**Total count**
Airplane	3	0	3
Barge	35	62	97
Building	1,386	1,047	2,433
Catamaran	0	2	2
Helicopter	1	0	1
Kayak	6	2	8
Motorboat	429	440	869
Motorboat with priority	604	45	649
Sailboat with sails down	44	14	58
Sailboat with sails up	17	2	19
Class total	2,525	1,614	4,139

Table 4.4: Class-labels visual spectrum images dataset Brekke & Lopez dataset

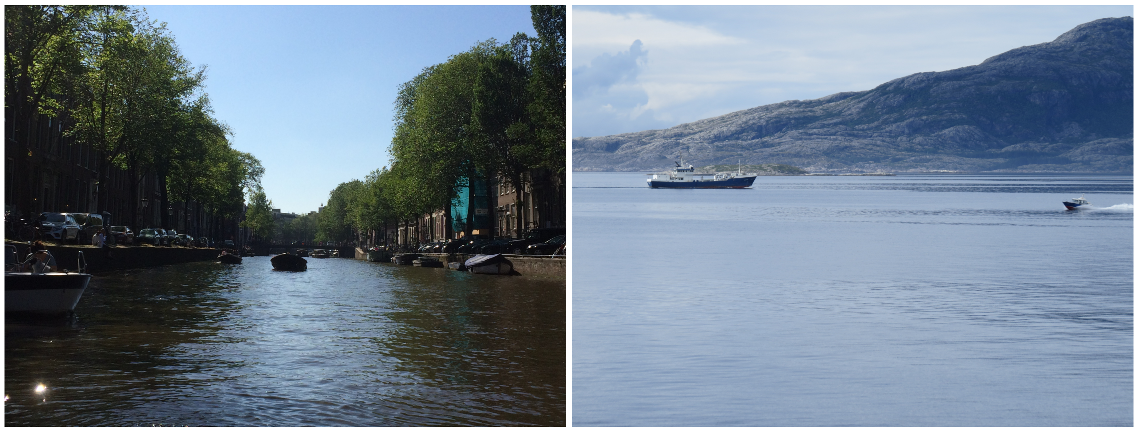

(a) Sample image Amsterdam$_{BL}$ (b) Sample image Hurtigruten$_{BL}$

Figure 4.2: Sample images Brekke & Lopez dataset

4.3 Hurtigruten dataset

The Hurtigruten dataset is a set of frames captured from recorded videos of the cruise ship Hurtigruten's journey along the coast-line of Norway. The videos are online available as a part of a Norwegian television-show called "Hurtigruten minute by minute" [6]. The dataset is labeled with bounding-box annotations and provided for this master thesis with the courtesy of DNV. This particular dataset is composed of videos captured on the journey from Bergen to Kirkenes. All annotation files are provided in the Pascal VOC XML format [29].

We summarize some initial observations when exploring the dataset.

- *Dataset partition*: The dataset is divided into 32 distinct episodes. Each episode contains images from the starting harbour and the journey towards the destination harbour.
- *Class-awareness*: The dataset is labelled with a total of 27 different maritime object classes.
- *Diversity*: The videos are recorded over a continuous time-period and in different geographical locations, providing many different maritime detection conditions.

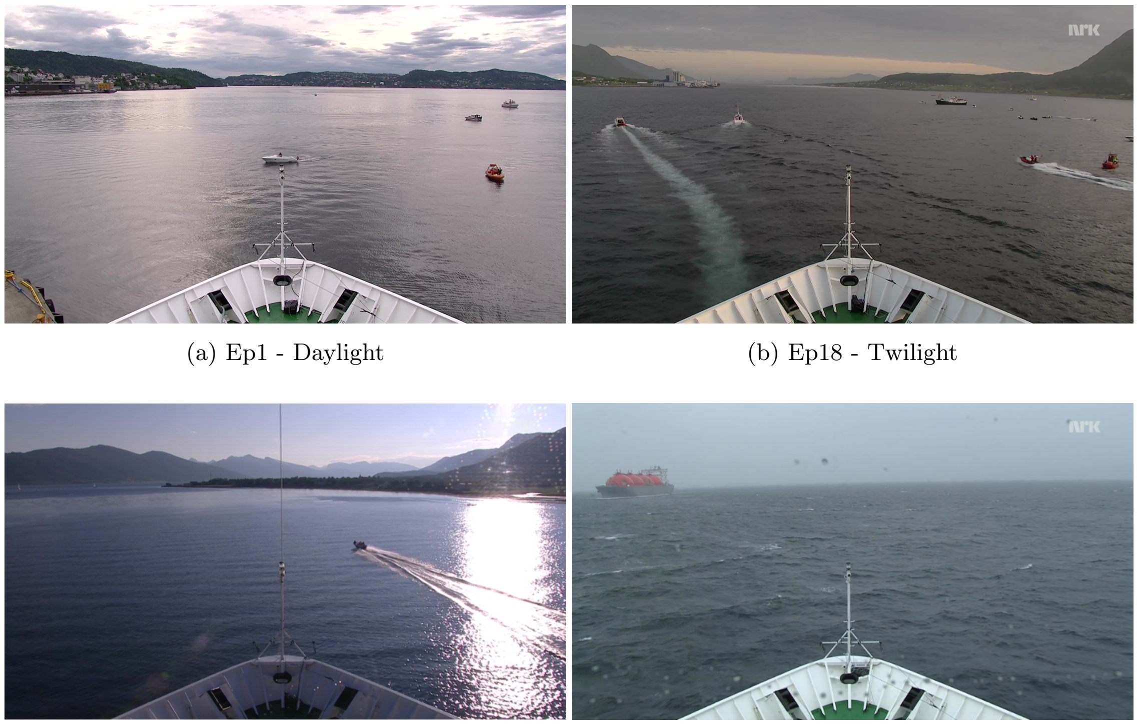
Based on these initial observations, we highlight some benefits in utilizing this dataset for training and testing an object detector. Each episode represents one journey, from dock to dock. Therefore, each episode mostly contains unique instances, which is convenient for designing a unique and clearly defined dataset split, without reoccurring and resembling instances in the training and testing set. As such, mitigating sample leakage[1].

The large amount of maritime object classes in the dataset might prove suitable for class-aware object detection, even though the class-balance must be further explored. The episodes, which originates from different geographical locations and points-in-time, provides a diverse dataset. In total, the episodes represent a wide range of different backgrounds, object viewpoints, weather and lighting conditions, which is promising for designing a robust object detector based on the dataset. To illustrate, four sample images are presented in figure 4.3.

The following exploration of the dataset is divided into three separate parts.

1) Section 4.3.1 undergoes a more in-depth exploration and quality-checking of the Hurtigruten dataset.

[1]Images containing strongly resembling instance viewpoint and scenes in training set and the validation-test set, potentially causing overfitting.



(a) Ep1 - Daylight

(b) Ep18 - Twilight

(c) Ep13 - Sunny with lens flare

(d) Ep29 - Rainy with water droplet lens occlusion

Figure 4.3: Sample images Hurtigruten dataset.

2) Section 4.3.2 explains considerations and actions performed in the post-processing of the quality-checked Hurtigruten dataset.

3) Lastly, section 4.3.3 analyses dataset statistics and presents a complete mapping of the post-processed Hurtigruten dataset as well as a proposed dataset split.

4.3.1 Dataset exploration

Following the first initial exploration, it is deemed important to quantify and quality-check the Hurtigruten dataset in detail. Since the frames are already captured and labeled from videos, an initial assumption is that every annotation file contains the presence of an object and is not empty. However, this assumption will also be tested to ensure the dataset is properly quality-checked. As such, the steps for quality-checking the dataset are sequentially presented.

Duplicate file naming The first observation concerns the naming of the image and annotation files. Several image files, with their corresponding annotation files, have what appears to be duplicates in the same folder. That is, there exists an image “image1.jpg” and an image called “image1 (2).jpg”, indicating that the image has been saved two times and a copy has been added to the folder with the automatically assigned sub string *(2)* from the operating system. The same is observed for the xml annotation files. 206 such image files are found in Episode 15 and 24 in Episode 18. During inspection, the double-named images, seem to be unique in almost all cases. One exception in the episode 15 folder is manually handled and deleted, however these cases are corner-cases. Based on this observation, it is preferable to keep all the images and annotations with existing double names. To avoid future confusion with the *(2)* extension, we rename all the double-named images and their corresponding annotations through a script. Inherently, this renaming has no practical implications except mitigating possible file name confusion. Our renaming simply replaces the substring *(2)* with *_2*, indicating that the filenames are intentionally named and not double-saved.

Image file types The delivered dataset consists of a mix of PNG and JPG images. Indeed, this observation is curious considering the images are saved from similar video streams. However, we find it necessary to decide on one image format to mitigate artifact bias from different image formats. Firstly, it is observed that the definitive majority of the images were already saved in JPG formats(approximately 72%). Furthermore, as the JPG image format has a lossy compression process, there is no way to restore previously known information. The decision is therefore to save all images in the JPG format, and hence re-save the PNG images to JPG. Inherently, the PNG image format would be preferable as to not introduce artifacts into the scene. However, considering that all other datasets treated in this chapter are also saved in JPG format, it is natural to strive for the same image format for the Hurtigruten dataset.

Image and annotation file count A last step in quality-checking the dataset is to ensure that all images and annotation files have a unique one-to-one matching. In other words, there are no images without an annotations file and there are no annotation files without an image. This analysis is performed on the dataset after removing the duplicate in episode 15. In particular, it is found some discrepancies in 8 of the 32 episodes as presented in table 4.5. It can be observed that depending on the episode, there exist both images without annotation files and annotation files without corresponding images.

All the episodes in table 4.5 are filtered through a script, leaving only unique one-to-one matched images and annotations remaining.

Annotated file discrepancies	
Episode	**Count discrepancy**
Ep1	-1
Ep3	-3
Ep8	-2
Ep10b	7
Ep12	-2
Ep14	-1
Ep18	3
Ep23	1

Table 4.5: Image and annotation discrepancies per episode Hurtigruten dataset. Negative numbers indicate the presence of more annotation files than images. Positive numbers indicate the presence of fewer annotation files than images.

Empty annotation files As mentioned introductory, a reasonable assumption is that every image with an annotation file, also contains annotated objects. That is, there are no annotation files which are empty. Nevertheless, it is considered necessary to check if this assumption is true. Indeed, it turns out that there are 16 images with empty annotation files. The images are spread out in different folders as presented in table 4.6. All the annotation files and corresponding images are deleted, as they represent only a marginal part of the total dataset, and as relabeling frames is time-consuming and avoided in this context.

Empty annotation files	
Episode	**Count**
Ep2	1
Ep3	1
Ep5	1
Ep7	1
Ep10b	2
Ep15	6
Ep17	2
Ep18	2

Table 4.6: Empty annotation file count per episode Hurtigruten dataset

Dataset summary After performing all the previous data cleaning steps, we shortly summarize some of the main characteristics of the Hurtigruten dataset in table 4.7. The dataset consists of a total of 4,700 images, all in 1920 × 1088 resolution. There is a total of 27 different classes and a total of 18,167 labeled instances. A detailed class summary is omitted here, as the later performed post-processing changes relevant dataset properties. Nevertheless, it is noted that the dataset is heavily imbalanced towards different types of pleasure crafts.

Cleaned dataset summary			
Image size	**Labeled images**	**Labeled instances**	**Object classes**
1920 × 1088	4,700	18,167	27

Table 4.7: Summary of Hurtigruten dataset after cleaning

4.3.2 Dataset post-processing

The Hurtigruten dataset, mainly due to the manner of which it has been acquired, contains certain characteristics which need to be discussed. The post-processing procedure is designed with the intention of enabling the Hurtigruten dataset for merging with other datasets, which is further discussed in chapter 5. In particular, we highlight three main observations from the dataset images which are considered in the post-processing procedure.

1) *Tv-channel watermark*: The watermark of the channel airing the tv-show, NRK, is fixed in the upper right corners of all the images as can be observed in figure 4.3b.
2) *Fixed boat hull*: The front part of the ferry's hull is present in all the images.
3) *Occurring cables*: A cable presumably connected to the mast above the camera station, occurs in some of the labeled episodes as visualized in figure 4.3c.

The three above-mentioned points do not necessarily represent any issues for using the dataset for object detection, which is the considered context. Simply due to the fact that most of the characteristics are reoccurring unlabeled fixed objects in the scene. An object detector would be guided by ground truth supervision to learn from the labeled informative regions in the scene.

On the other hand, utilizing the raw dataset in combination with other data sources might prove harmful. The Hurtigruten portion of such a combined dataset would repeatedly provide a boat-like unlabeled object (the boat hull) in the same geometrical location of the scene. The effects of combining such images with other images containing labeled objects in the same placement is uncertain. Recent research demonstrates that using uneven zero-padding on CNNs [14] causes a spatial bias for feature extraction, potentially creating blind spots and misdetection of objects in the scene. In this case, where the boat hull is not zero-valued pixels, but rather contains similar features to desirable detectable objects, it is likely to deteriorate detection performance while causing a bias towards the lower part of the scene. Therefore, the boat hull is considered vital to include in the post-processing procedure.

The same issue is not as present for the tv-channel watermark due to its small area and placement in the very upper-right corner of the scene, where it is reasonable to expect background pixels from images of different data sources. This assumption might not hold, depending on the camera viewpoint and object placement in images from other data sources, which might be in the upper-right corner in special cases. As the procedure of handling the watermark is found to be quite simple, this step is furhter included in the post-processing procedure.

Lastly, the occurring cable is only present in a subset of the frames in 6 of the total 32 episodes. One might regard the cable as nothing more than natural noise and is not necessary to manually handle.

Tv-channel watermark handling To remove the watermark, the intuitive approach is designing an algorithm which gradually fills the watermark with pixels of similar colour distribution to a close neighborhood of pixels which are not a part of the watermark. Instead of designing such an algorithm from scratch, existing methods are firstly assessed.

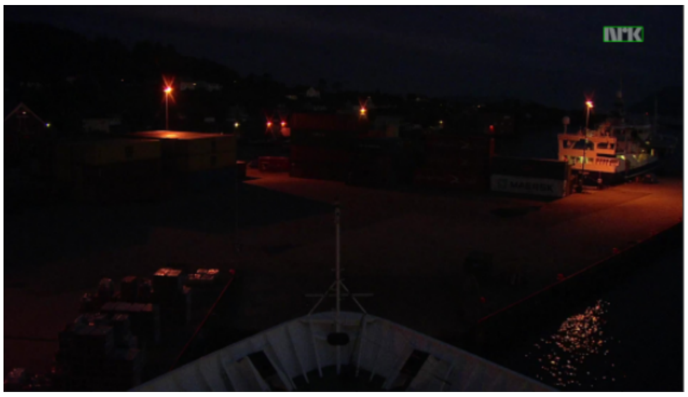
The OpenCV library [18] provides a specific method which is very similar to our specific use-case. The *inpaint()* function is used for restoring images with noise and strokes, back to an original state by filling in the bad pixels with neighbouring pixels. The method is implemented with two different algorithm choices for filling in the pixels. The first method [103], referred to as the Telea method, fills in a desired pixel region by using a weighted sum of nearby pixels by weighting more nearby pixels higher. The Fast Marching method is used to inpaint subsequent pixels in a heuristic approach.

The second method is based on principles from fluid dynamics and travels along the edge of the pixel region to fill pixels with colour while reducing the minimum variance in the region. The method is referred to as NS, indicating the use of Navier-Stokes equations.

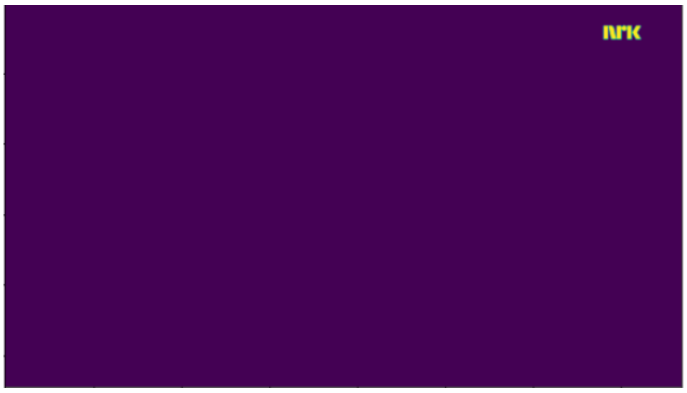
In our initial experiments, both methods are found to perform satisfactory and it is decided to use the Telea method for filling in the watermark. The process for filling in the watermark can be summarized in three separate points.

1) Locate the region of the watermark, which is the same for all images.
2) Generate a global mask of the watermark for pinpointing the pixel regions to fill in each image.
3) Fill the watermark with the chosen filling method and re-generate all images with the watermark filled.

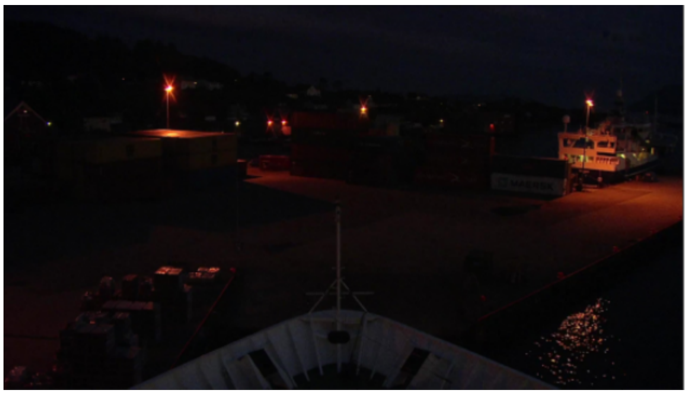
The location of the watermark is found by a trial and error approach, consisting of drawing different bounding boxes to enclose the the watermark until satisfactory boundaries are found. The bounding box with satisfactory tightness on the watermark is visualized in figure 4.4a. Secondly, one of the episode images with very dark background conditions are chosen for generating the watermark mask. This is chosen as the watermark is white, and with a dark background a simple thresholding of the image accurately yields a precise watermark mask. After generating the watermark mask locally inside the approximated bounding box, a coordinate transformation is performed to map the local mask to a global mask. The global mask, which is used for the in-filling of all images is shown in figure 4.4b. Lastly, the generated global mask is utilized for in-painting all images' watermarks with the default Telea inpainting method for the OpenCV inpaint() function, as presented in figure 4.4c.



(a) Step 1: Tv-channel watermark located with green bounding box



(b) Step 2: Mapping the thresholded local mask to a global mask



(c) Step 3: The final in-painted image without watermark

Figure 4.4: The watermark handling procedure for the Hurtigruten dataset.

Fixed boat hull handling The fixed boat hull is handled by designing an algorithm which crops each image into two separate images, while omitting any pixels covered by the boat hull. We refer to this procedure as *tiling*. Each image is tiled on the center point of the boat hull as demonstrated in figure 4.5a. Subsequently, each tiled image results in two smaller tiles with the same annotated objects.

The tiling algorithm is designed to keep all the intersecting objects with over 10% of its original area size contained in the generated tile. Figure 4.5b and figure 4.5c demonstrate one such example, where the boat caught in the middle of the tiles is partially labeled in the resulting tiles. In this manner the dataset also includes incoming and exiting vessels as part of the dataset, which encourages the detection of vessels based on a subset of their features. Such instances are considered

Post-processed dataset summary		
Labeled images	**Labeled instances**	**Object classes**
6,775	17,696	27

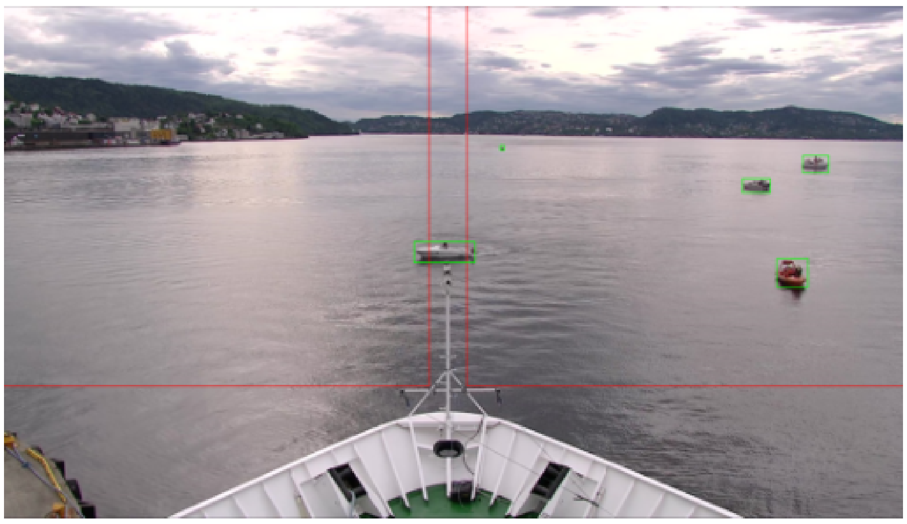
Table 4.8: Summary of the post-processed Hurtigruten dataset; after cleaning and tiling post-processing. Tiles vary in size.

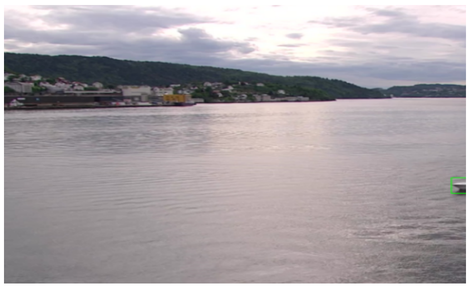
important for learning a detector to detect entering/exiting foreground objects quickly.

Furthermore, based on the tiling boundaries, objects very close to the boat hull with a steep viewpoint are naturally discarded as they exceed the tiling boundaries. In line with the problem formulation in section 1.1, this is considered beneficial. Namely, because the designed detector is intended for deployment close to the water surface. As follows, the remaining labeled instances in the tiled dataset will more closely resemble the expected viewpoint of objects in the target domain.

Table 4.8 summarizes the Hurtigruten dataset after the tiling procedure is performed. The reader might observe that the amount of images from the summarized dataset before tiling (table 4.7) is not doubled in size, as one might expect when generating two tiles per image. The explanation behind this, is that there is no guarantee that each tile produced from an image contains labeled instances. A total of 2,625 tiles with no labeled instances present were discarded in the tiling procedure. As such, the total labeled images sum to 6,775.

Moreover, the image sizes of the tiles are omitted, as they slightly differ in size. The left generated tiles are reported as 909×809 pixels resolution while the right tiles becomes slightly larger with 931×809 resolution. The tile height of 809 is equal for both tiles. In total, the tiled dataset contains 471 discarded instance labels which have been either located in between the tiling boundaries without sufficient area overlap, or placed below the tiling window.


(a) Tiling boundaries and annotated instances


(b) Left tile

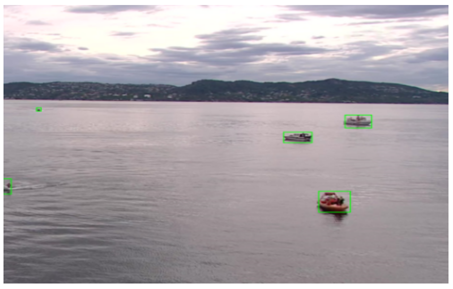

(c) Right tile

Figure 4.5: The Hurtigruten dataset tiling procedure.

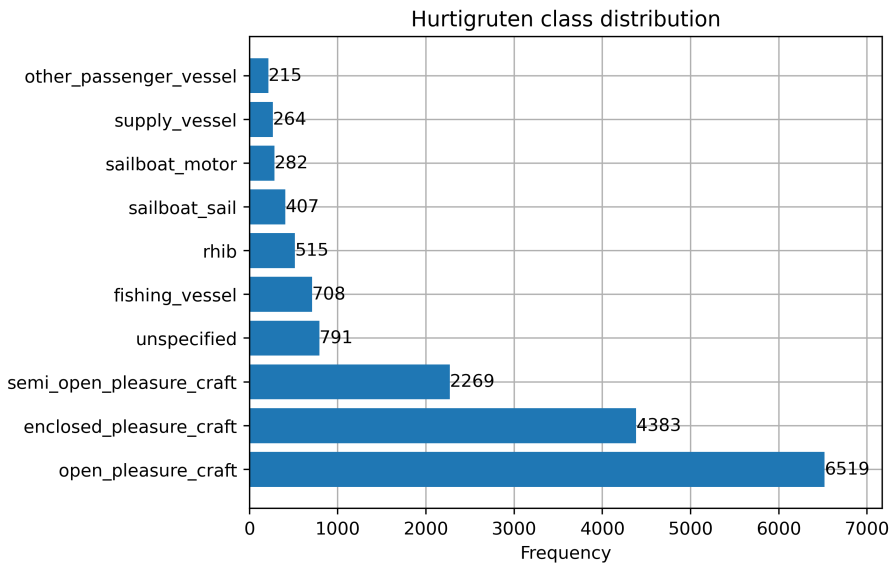


Figure 4.6: Hurtigruten dataset post-processed. Top ten classes distribution.

4.3.3 Dataset statistics

The dataset statistics in this section concerns the post-processed Hurtigruten dataset. It is prioritized to shortly summarize key-observations regarding the classes in the dataset, as well as our considerations in mapping dataset and creating a suitable dataset split.

The Hurtigruten dataset consists of 27 classes after post-processing. The top ten classes are visualized in the barplot in figure 4.6. Overall, the large majority of the classes is some type of pleasure craft. Of the total 17,696 instance labels in the dataset, a total of 13,171 belongs to the three different pleasure craft classes in the lower three bars of the plot in figure 4.6.

Inherently, the dataset is strongly class-imbalanced towards pleasure crafts. Of the remaining 17 classes, 15 classes represent different boat and vessel types with the addition of one buoy and fish-farm class. The 15 boat classes include boat types such as kayak, ferry, barge, cargo vessels and jetski. The instance labels are not specified for these classes as they represent only a small fraction of the total instance label count. Based on these observations, the Hurtigruten dataset is not considered suitable for class-aware object detection.

Episode conditions mapping To provide more insight into the dataset, a complete mapping of each episode is conducted. In mapping the dataset, quantitative measures describing the dataset diversity and conditions under which the instances have been acquired is considered the most important. Since each episode contains an amount of images in the range from ten to several hundreds, only distinct and easy recognizable measures such as weather, lighting conditions and the presence of lens occlusions are recorded. More specific measures such as object viewpoint, distribution of instance bounding box areas and class distributions per episode are considered too detailed when making an overview dataset mapping. Moreover, due to the long duration of some of the episodes, the weather and lighting conditions change accordingly, and the recorded condition is representative for the majority of the frames.

Table 4.9 presents the dataset mapping and the proposed dataset split, which is discussed below.

Hurtigruten mapping per episode

Split	Episode	Lighting	Weather	Occlusion	Images
Train	Ep1	Daylight	Cloudy	-	331
	Ep3	Daylight	Cloudy	-	100
	Ep4	Daylight	Cloudy	-	32
	Ep6	Daylight	Rainy	✓	133
	Ep7	Daylight	Cloudy	-	332
	Ep8	Dark/Twilight	Cloudy	-	286
	Ep9a	Dark/Twilight	Cloudy	-	95
	Ep9b	Daylight	Cloudy	-	90
	Ep10a	Daylight	Cloudy	-	490
	Ep10b	Daylight	Cloudy	-	749
	Ep11	Dark/Twilight	Sunny	-	302
	Ep12	Dark/Twilight	Sunny	✓	245
	Ep14	Daylight	Sunny	-	319
	Ep15	Daylight	Sunny	-	958
	Ep17	Daylight	Cloudy	-	243
	Ep18	Daylight	Sunny	-	510
	Ep20	Dark/Twilight	Haze	-	51
	Ep21	Dark/Twilight	Sunny	-	74
	Ep23	Daylight	Cloudy	-	155
	Ep24	Daylight	Cloudy	-	108
	Ep26	Daylight	Cloudy	-	100
	Ep27	Daylight	Cloudy	-	85
	Ep28	Daylight	Cloudy	-	63
	Ep32	Daylight	Sunny	-	5
Validation	Ep2	Dark/Twilight	Cloudy	-	46
	Ep5	Daylight	Cloudy	-	209
	Ep13	Daylight	Sunny	-	58
	Ep16	Daylight	Sunny	-	172
	Ep19	Dark/Twilight	Sunny	-	167
	Ep22	Daylight	Cloudy	-	127
	Ep25	Daylight	Cloudy	✓	96
	Ep29	Dark/Twilight	Rainy	✓	17
	Ep30	Dark/Twilight	Rainy	✓	16
	Ep31	Dark/Twilight	Rainy	✓	11

Table 4.9: Episode mapping Hurtigruten dataset post-processed. *Cloudy* is used to describe both clear and cloudy sky conditions. *Occlusion* refers to lens occlusions caused by flaring light or water droplets. *Images* refers uniquely to labeled images.

Proposed dataset split In addition to mapping the complete dataset, a dataset split used for experimental dataset design in chapter 5 is proposed. Since the Hurtigruten dataset is not intended for benchmarking detector performance in this thesis, but rather as an additional data contribution, a test set is not designed as to maximize the training data. However, the proposed validation set can easily be re-split by any user for creating a similarly distinct split. In designing the dataset split of the Hurtigruten dataset, it is desired to represent the mapped conditions satisfactory in both the training and validation set. The training and validation split is performed with the intention of reserving approximately 85% of the images for training and 15% for validation. Our episode choices for the dataset splitting process is summarized as follows.

Firstly, the occlusion and rain conditions are observably the rarest conditions that occur in the dataset. To avoid splitting any episodes into the train and validation set, which would break the condition of separated unique instances in the split, these episodes are examined carefully as a starting-point. Episode 6 and 29-31 are the episodes in question with rainy conditions. The intuitive split, results in keeping episode 6 for training, while reserving episode 29-31.

The reader might observe that episode 29-31 also includes lens occlusion conditions. To ensure enough lens occlusion images for the validation set, episode 25 is additionally added. By keeping episode 6 and 12 in the training set, same observable conditions are preserved for training.

The episodes captured in twilight and dark lighting conditions are overall less represented in the dataset than the daylight conditions. Thus, episode 2 and 19 is added to the validation set to further represent typical twilight sunset conditions and more cloudy sky conditions. Lastly, as the remaining episodes are captured in daylight with either cloudy or sunny weather conditions, episode 5, 13, 16, 22 and 25 is decided upon to finalize a validation set approximating 15% of the total dataset size. The final training and validation set size are summarized in table 4.10.

Hurtigruten dataset split		
Train	**Validation**	**Total**
5856 (86.4%)	919 (13.6%)	6775

Table 4.10: Hurtigruten post-processed dataset split summary.

4.4 Singapore maritime dataset

The SMD [79] is, to the knowledge of the author, the most complete online available maritime dataset with bounding box annotations. In this short introductory summary of the dataset, the reported dataset overview from the detailed analysis performed by Moosbauer et al. [70] is utilized, as presented in table 4.11. The dataset contains VIS and NIR spectrum videos in 1080 $\times$ 1920 resolution, captured from maritime waters outside Singapore. In total, the dataset consists of 81 videos, of which 63 are annotated, providing 31,653 labeled frames and 240,842 instance labels.

The visual spectrum images are separated by the manner of which they were captured. 2,400 labeled frames and 3,173 instance labels are reported to be captured *on board* a moving vessel. 17,967 labeled frames and 154,495 instance labels were captured *on shore* a fixed platform in maritime waters. Moreover, images from all spectra are labeled according to the their illumination conditions and cover; haze, twilight, and daylight.

SMD properties			
Subdataset	**Videos (annotated)**	**Labeled frames**	**Labeled instances**
NIR	30 (23)	11,286	83,174
VIS on board	11 (4)	2,400	3,173
VIS on shore	40 (36)	17,967	154,495
Total	81 (36)	31,653	240,842

Table 4.11: Summary of SMD as reported by [70] in Table 1.

Source: [70]

Moosbauer et al. thoroughly analysed the SMD, its provided label discrepancies and proposed the first official training, validation and test split of the dataset. The proposed split applied the label instances' tracking IDs to counter the appearance of the same instances in the training and test set. We refer the reader to table 2 [70] for a more in-detail overview of the episodes used in the dataset split.

Moosbauer et al. further provided several interesting observations regarding the SMD class distribution. Of the ten object classes in the dataset, which are later discussed in detail in section 4.4.3, the classes are over-represented by vessels and ships, denoted by the *vessel/ship* class. The other classes vary between different boat types and various other maritime objects encountered, such as buoys. Some sample images from the VIS spectrum are presented in figure 4.7.

The remaining parts of this section are structured in the following way.

- Section 4.4.1 presents the generation of the SMD and observations regarding the consistency

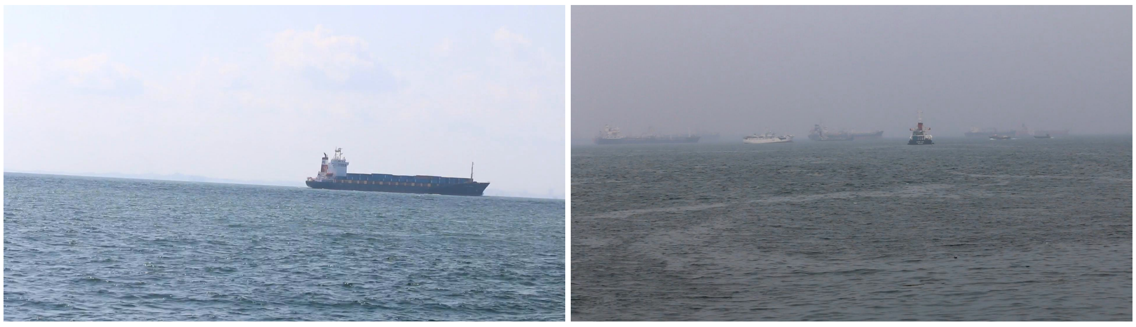

(a) Sample image on board (b) Sample image on shore

Figure 4.7: Sample images SMD VIS spectrum.

of the instance labels and classes.

- Section 4.4.2 presents post-processing considerations of the generated SMD.
- Section 4.4.3 covers some dataset statistics following our generation of the post-processed SMD

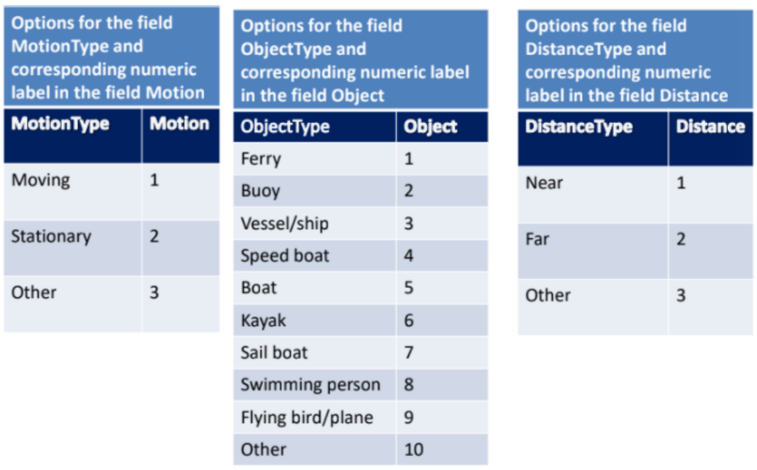
.

4.4.1 Dataset exploration

In this section we report the detail in our exploration and generation of the SMD. As an additional contribution, the dataset generation process, not particularly covered in previous work [70][79], is in detail explained. The dataset exploration further includes observations regarding the preciseness of the instance labels and discovered spatio-temporal discrepancies in the instance class assignments. As this thesis researches visual spectrum object detection, the NIR images from the SMD are neither reported nor utilized.

Dataset generation The SMD is generated from recorded videos on the AVI format, provided on the Google site of Prasad [77]. As follows, the SMD generation process involves generating frames from each provided video and matching with delivered ground truth files. The ground truths and their description files [76], hosted on the same site, include a horizon, object detection and object tracking ground truth. We refer the reader to the original paper [79] and the ground truth description file [76] for further details regarding the ground truth annotation files. All label types for the object detection ground truth file is presented in figure 4.8. As the reader might observe, the SMD is annotated with the possibility of assessing objects based on motion and distance type in addition to standard bounding boxes.

In the SMD generation process, only the object detection ground truth bounding boxes is of interest, even though the object tracking ground truth is also utilized for quality checking consistency of the class assignment for the labeled instances.



Options for the field MotionType and corresponding numeric label in the field Motion	
MotionType	Motion
Moving	1
Stationary	2
Other	3

Options for the field ObjectType and corresponding numeric label in the field Object	
ObjectType	Object
Ferry	1
Buoy	2
Vessel/ship	3
Speed boat	4
Boat	5
Kayak	6
Sail boat	7
Swimming person	8
Flying bird/plane	9
Other	10

Options for the field DistanceType and corresponding numeric label in the field Distance	
DistanceType	Distance
Near	1
Far	2
Other	3

Figure 4.8: SMD object detection ground truth description.

Source: [76]

Following the downloading of the videos and annotations, the frame generation process consists of five similar steps for the VIS on board and on shore videos.

1) *Find annotated videos*: Filter out the videos with existing object ground truth files from the ground truth folder, ObjectGT.
2) *Frame-generation*: Generate the frames from each of the videos.
3) *Frame to ground truth matching*: Match each frame with corresponding video frame annotations from ObjectGT.
4) *Annotation format*: Generate the annotations on desired labeling format. The PASCAL VOC XML format is used in our case.
5) *Clean annotations*: Remove frames without useful annotations.

Our code for the frame generation and annotation matching is based on own observations as well as one online GitHub repository [104] analysing the dataset. Since most of the scripts utilized for the frame generation process contains dependencies to several libraries and other custom frameworks developed by the author, it is deemed more important to capture the documentation process rather than displaying code source files. Most of the steps are easy to re-produce following our documentation process.

The first step covers the mapping of videos with existing ground truth files. Similar to previous work [70], it is found four on board and 36 on shore videos with existing ground truths. This filtering is simply done by matching each video prefix with the annotation prefixes in the ObjectGT folder. For instance, the on board video *MVI_0790_VIS_OB.avi* is matched with *MVI_0790_VIS_OB_ObjectGT.mat.*

The second step generates all the frames for each video. The common frame rate for all videos are found to be 30 FPS, as reported on the hosted google site. All the frames for each video is generated using the python adapted *VideoCapture*() function from the OpenCV [18] library.

In the third step, a quite time-demanding issue is encountered. Initially, each generated frame from a video is assumed to have a matching annotation from the corresponding ground truth file. Or in other words, each video should generate the same amount of frames that exist in the ground truth file. However, this is not the case for three separate videos, as presented in table 4.12. As such, the matching of frames to annotation for the three particular videos is neither one-to-one nor unique. This issue indicates that there are discrepancies in either the hosted ground truth files or the videos.

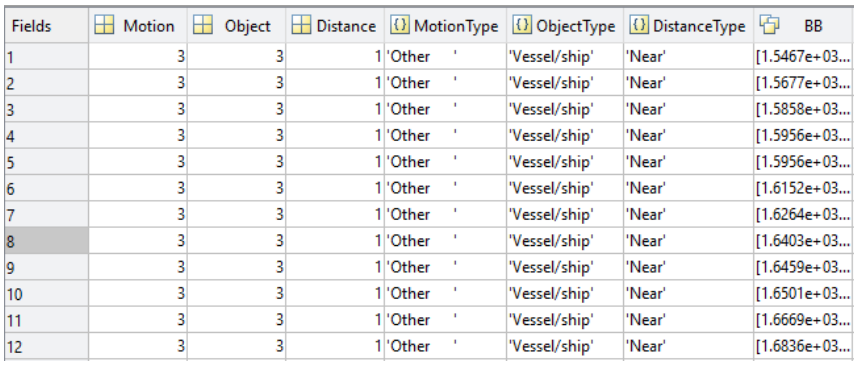
Mismatched video frames to ground truth			
Subdataset	**Video name**	**Frame count**	**Ground truth frame count**
On board	MVI_0790_VIS_OB	600	1,010
	MVI_0799_VIS_OB	600	601
On shore	MVI_1584_VIS	539	550

Table 4.12: Mismatch in amount of generated frames to amount of frames expected from ObjectGT Singapore dataset.

Initially, it was investigated if the videos were generated with an inconsistent frame-rate. However, as we found no further basis for this hypothesis, we further investigated the labeled frame count overview of each video in Table 2 of [70]. Using MVI_0790_VIS_OB as a sample video, Moosbauer et al. record 1,010 labeled frames. Thus, the issue is pinpointed to be either a discrepancy in our video generation or in the recorded labeled frames from [70].

It is found that Moosbauer et al. have reported the labeled frame overview based on the amount of existing ground-truth and not the labeled frames after matching to annotated frames. Moreover, their frame matching process is performed by generating all video frames and matching them to the annotation files corresponding to the same time-stamp[2]. As such, the amount of labeled frames are upper-bounded by the amount of generated frames per video. We follow the same methodology, implying that all the labeled frames for the videos in table 4.12 are upper-bounded by the "Frame count" column.

The fourth step involves the processing of annotation files from the MAT Matlab file format. One example of the annotations for the first 12 frames of *MVI_0790_VIS_OB_ObjectGT.mat* is presented in figure 4.9. Essentially, this processing step is a simple parsing problem, which is handled by reading each mat file through the *loadmat* function from the Scipy library. Finally, each frame is matched to its corresponding annotation and written to the Pascal VOC XML annotation format.



Fields	Motion	Object	Distance	MotionType	ObjectType	DistanceType	BB
1	3	3	1	'Other '	'Vessel/ship'	'Near'	[1.5467e+03...
2	3	3	1	'Other '	'Vessel/ship'	'Near'	[1.5677e+03...
3	3	3	1	'Other '	'Vessel/ship'	'Near'	[1.5858e+03...
4	3	3	1	'Other '	'Vessel/ship'	'Near'	[1.5956e+03...
5	3	3	1	'Other '	'Vessel/ship'	'Near'	[1.5956e+03...
6	3	3	1	'Other '	'Vessel/ship'	'Near'	[1.6152e+03...
7	3	3	1	'Other '	'Vessel/ship'	'Near'	[1.6264e+03...
8	3	3	1	'Other '	'Vessel/ship'	'Near'	[1.6403e+03...
9	3	3	1	'Other '	'Vessel/ship'	'Near'	[1.6459e+03...
10	3	3	1	'Other '	'Vessel/ship'	'Near'	[1.6501e+03...
11	3	3	1	'Other '	'Vessel/ship'	'Near'	[1.6669e+03...
12	3	3	1	'Other '	'Vessel/ship'	'Near'	[1.6836e+03...

Figure 4.9: ObjectGT annotation example MVI_0790_VIS_OB_ObjectGT.mat Singapore dataset

In the fifth and last step, some simple quality checks are performed. It is considered most important to ensure that each frame is indeed annotated with the presence of an object, as this is not checked during the frame to annotation matching of step three. Especially for the on board videos, which experience substantial camera movement due to waves and movement from the boat of which the videos are recorded, it turns out that several of the annotated frames only contains background.

Of the on shore videos, no cases of empty annotated frames are encountered. In the on board dataset, particularly the two videos mentioned in table 4.12, represent some discrepancies. To elaborate, for MVI_0790_VIS_OB, only 302 of the matched ground truth frames are annotated with the presence of an object. Similarly, for MVI_0799_VIS_OB, only 480 of the 600 matched ground truth frames contain annotated objects.

The annotated frames and the total instances of the SMD dataset, after this final step, is summarized in table 4.13. The reader can observe that particularly the on shore dataset differs from the

[2]Pointed out by Sebastian Moosbauer in personal correspondence

purely ground truth based summary by [70] presented in table 4.11, with 418 fewer labeled frames.

SMD properties			
Subdataset	**Videos (annotated)**	**Labeled frames**	**Labeled instances**
VIS on board	11 (4)	1,982	3,053
VIS on shore	40 (36)	17,967	154,494
Total	51 (40)	19,949	157,547

Table 4.13: Summary of SMD properties from our generation process.

Annotation observations To ensure that the quality of the bounding box annotations of the SMD is satisfactory, the annotation files are systematically quality checked together with visualized samples. Since the images and corresponding xml annotation files are generated by the author, file naming discrepancies and differences in image file types, as encountered in section 4.3, are easily avoidable. Overall, two observations, the latter of which already mentioned in [70], are necessary to discuss.

Firstly, a small subset of the the bounding box coordinates are observed to be negative. In best faith, all instance labels were assumed valid in the process of generation the SMD annotations. From the ground truth description file [76], all x and y coordinates are defined as positive from the upper left corner. Most of the observed negative values are observed to be barely out of range (e.g $x_{min} = -3$), while still covering the objects satisfactory. The observed negative values are likely caused by discrepancies in the labeling process or video resizing. Based on empirical observations, any bounding box exceeding the image boundary with more than 13 pixels, is discarded, while the bounding boxes below this value are reset to the image boundary value on the edge(s) exceeded. This is further discussed in section 4.4.2.

Secondly, the *bounding box tightness* of the annotated objects is inconsistent. In essence, if utilizing the same terminology as [59] which define the tightness of a bounding box; the bounding box is considered tight if it touches the enclosed object's contours on every one of the bounding box edges. This is not consistently the case for the SMD and poses a delicate problem to handle. Mainly, as there is no way to quality check all the bounding boxes' tightness quantitatively, since the bounding boxes are implicitly the only ground truth localizing the objects in the scene and a total relabeling is out of question.

Moosbauer et al. treated this problem by changing the IOU threshold on inference for their trained detectors. Even though this is a valid solution, there is no guarantee that lowering the IOU threshold will well-counter the presence of too many background or too few object foreground pixels in the ground truths during training. Another solution would be to marginally expand or contract the bounding boxes by a certain percentage, as performed in the ablation studies of [45]. However, this procedure would negatively affect the precisely annotated ground truths. Generally, the discussed issue is not feasible to properly mitigate without reannotating the entire dataset. We consider two options to counter the imprecise ground truths.

- *Training on SMD only*: Put more emphasis to metrics assessing predictions with a lower IOU threshold.
- *Training on a mix of SMD and other datasets*: Do not make any adjustments.

When training on a mix different data, we justify that not making any adjustments to the ground truths is the fairest evaluation since the imprecise ground truths are generally observed to be the exception rather than the rule.

Class observations The majority of labeled classes of the SMD vary between different sub-types of boats, as the reader may observe in the ground truth description of figure 4.8. Additionally, as underlined by Moosbauer et al. and later discussed in section 4.4.3, the SMD is strongly imbalanced towards the *ship/vessel* class. However, two other observations are prioritized to discuss here.

The boat sub-types, hereby referred to as the *boat classes*, are not mutually exclusive. For instance, there is not clear separation in the definition of a sample belonging to the *Boat* class and the *Speed boat* class. A speed boat is inherently also a boat. Additionally, it is empirically observed that several of the boat classes resemble one another or are not distinctly separated between in the labeling procedure. Figure 4.10 presents three such examples. The speed boat from figure 4.10c resembles the annotated ferries in figure 4.10b more than the speed boat in figure 4.10a. Moreover, the speed boat in figure 4.10a is more similar to typical ferry from the best of knowledge to the author. Since the SMD has been labeled by a variety of different students [77], it seems the boat classes have been defined slightly different by each individual.

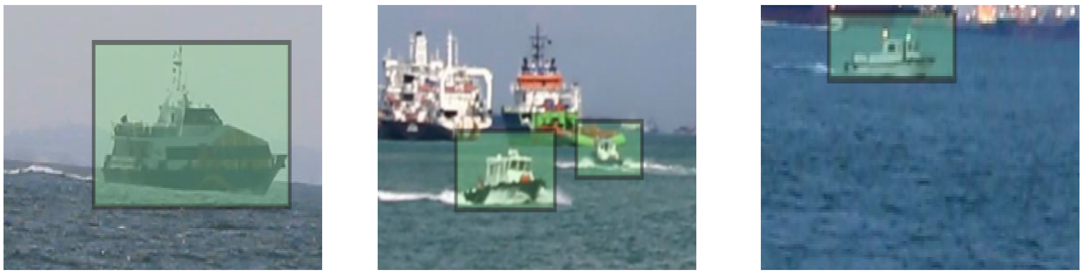
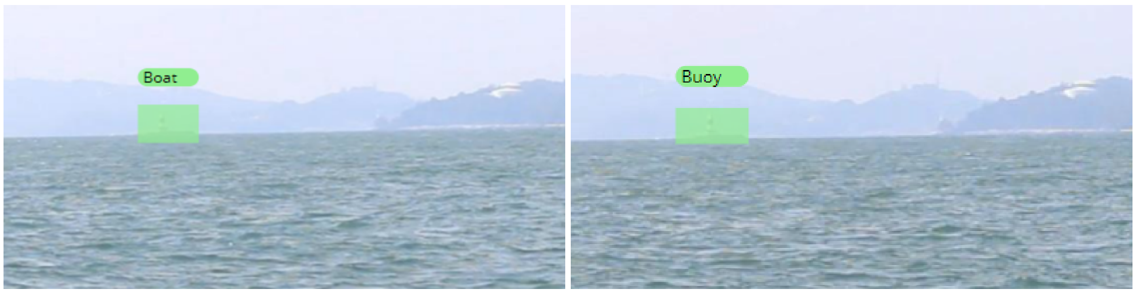

(a) MVI_0799 - speed boat (b) MVI_1474 - ferry (c) MVI_1584 - speed boat

Figure 4.10: Speed boat or ferry? Samples from SMD with specified ground truth labels.

The last point to discuss is the *Other* class. This particular class is observed to contain both boat instances and pure background pixels. As such, it is neither a mutual exclusive relationship between this class and the generic *Boat* class. The *Other* class is further discussed in the following point regarding class assignment inconsistencies.

Class assignment inconsistencies In exploring the SMD we discover some inconsistencies in the class-labeling, similar to reported by [70]. The inconsistencies are possible to quantify thanks to the tracking id labeling provided with the dataset. To clarify, each video contains an object ground-truth, which frame-wise annotate all the bounding boxes of the objects in scene, as well as their class. In addition, the tracking ground-truth frame-wise assigns an id and class-type for every object annotated in the video, over the complete duration of the video. Therefore, from the tracking ground-truth it is possible to assert that each tracked object is only assigned to one class. In other words, if a tracked object switches between several classes, which it should not, this is easy to observe.

In total, we find that 6.7% of all individual tracks in the SMD are subject to a class assignment inconsistency. In total, 27.5% of the videos contain one or more class switching tracks. However, the class assigment inconsistency is mostly observed to change between different boat classes. For instance, one object labeled *boat* is labeled *ship/vessel* at a later occurence. This observation motivates the usage of SMD in a class-agnostic setting, where such a class inconsistency would be insignificant. Some few empirical observations of class switches between *buoy* and *boat* are also observed as visualized in figure 4.11. These cases are handled in in the post-processing procedure of section 4.4.2.



(a) MVI_0790_VIS_OB frame 253. (b) MVI_0790_VIS_OB frame 259.

Figure 4.11: Class assignment inconsistency illustrated for the same object instance in MVI_0790_VIS_OB. Visualized with Remo.

4.4.2 Dataset post-processing

Based on observations from section 4.4.1 and previous experiments conducted on the SMD [70], the SMD is arguably best suited for class-agnostic object detection. As such, class-imbalance and class assignment inconsistencies are easily handled by encapsulating all boat-related classes in a generic boat class. Three adjusted steps of post-processing are performed on the raw generated SMD.

Firstly, as discussed by Moosbauer et al., the videos captured in 30 FPS generate a large quantity of practically identical frames. It is observed to particularly be the case for the on shore videos, which are recorded from a stand-still platform often with static scene and many anchored-up ships. Using all these frames represents a potential oversampling bias [71]. Therefore, similar to [70], we downsample the generated frames by a factor of two, using every other frame. However, in contrast to Moosbauer et al., we only do this for the on shore generated frames, as the on board frames do not suffer from a similar static scene and often includes objects moving in reference to the camera.

Secondly, as we in this thesis aim to perform maritime vessel detection, an agnostic relabeling of the SMD needs removal of some of the minority classes. In essence, all instance labels from the *Swimming person*, *Flying bird/plane* and *Buoy* class are deleted. Moreover, the *Other* class found to also annotate pure background pixels is handled. Empirically, the *Other* class discrepancy is found to occur in the *MVI_1584_VIS* generated frames. 270 *Other* labels from these frames are hence deleted.

The final step is to create a meaningful split of the SMD. Since this is already proposed in [70], with measures to counter the reoccurrence of the same instances in the split sets, we use this split with a few alterations. Firstly, in our experiment design later presented in chapter 5, it is not desired to benchmark performance on the SMD. Thus, a test set is not strictly necessary. To maximize training data, we merge the training and validation set of [70] to a new training set. The proposed test set is used as the new validation set. Again, we refer the reader to Table 2 [70] for a detailed episode overview. Lastly, not reported in detail by Moosbauer et al., the on shore video *MVI_0799_VIS_OB.avi* is deleted from the split. The background for this decision is varying degree of lens focus and zoom in the recorded video, causing many unusable frames and annotations[3].

4.4.3 Dataset statistics

Figure 4.12 presents the class distribution of the post-processed SMD, not separating between the split sets. The class distribution is presented as to give the reader a sense of the severe class imbalance of the SMD. As introductory mentioned, the *Vessel/ship* class represents the large majority of the dataset. Of the post-processed SMD, a total of 76.6% of the instance labels belong to this class.

[3]Pointed out by Sebastian Moosbauer in personal correspondence and observed by the author on inspection.

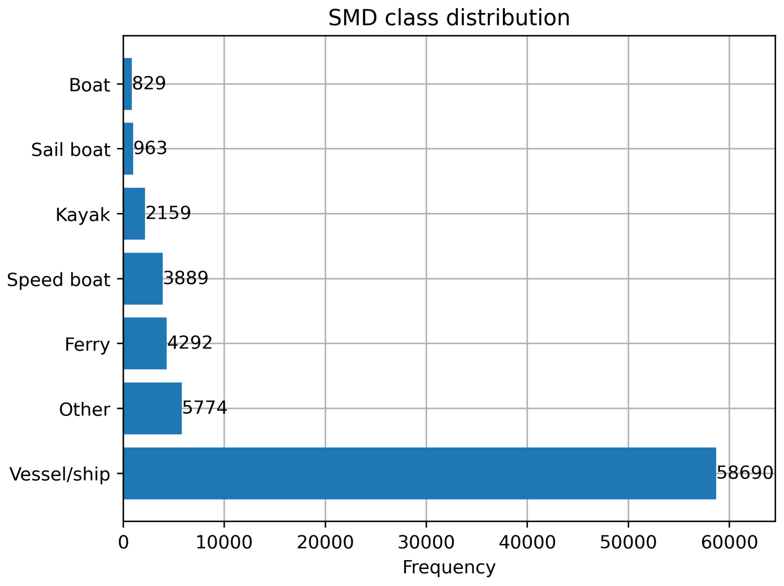


Figure 4.12: Post-processed SMD class distribution after downsampling and class removal, before agnostic relabeling.

The agnostic relabeling of the post-processed SMD is simply performed by relabeling all of the classes in figure 4.12 to a generic *Boat* class. Table 4.14 summarizes the total images in the training and validation set of the post-processed SMD.

SMD dataset split

Train	**Validation**	**Total**
7,478 (71.9%)	2,916 (28.1%)	10,394

Table 4.14: SMD post-processed dataset split summary.

4.5 Target domain dataset

In this section, the target domain dataset is created in accordance with the surrounding environment of the autonomous ferry milliAmpere.

milliAmpere is intended to operate in the crossing of the Nidelva river at the interception of the Ravnkloa dock. A custom sampled dataset from the Ravnkloa harbour and the harbour orifice towards the Trondheimsfjorden is therefore acquired by the author on the 17th of April 2021. All images and videos were sampled in accordance with other students at NTNU which sampled different types of sensor data in harbour and open-sea environments for different scenarios. All images acquired are taken on board milliAmpere, which was rigged and manoeuvred by Martin Gerhardsen and Martin Græsdal. A marker boat was rented, rigged and manoeuvred by Thomas Hellum and Kristian Auestad.

The collected images and videos are acquired by a portable mobile phone and the EO cameras installed on the sensor rig of milliAmpere. The portable mobile phone was used as an additional data source as this device gave more flexibility in viewpoint angle and enabled easier scenario creation by filming from a variety of angles. The utilized iPhone X recorded images of 2048×1536 resolution and videos of 1280×720 with 30 FPS. The EO cameras of milliAmpere recorded videos of 1224×1024 resolution with 5 FPS.

The intention behind the dataset acquisition was two-fold. Firstly, it was desired to capture more target domain images, particularly for a large variety of classes. However, since the experiments were conducted over one afternoon, the class diversity were dependent on vessel types entering and leaving the harbour that day. Secondly, is was regarded as crucial to attain a set of videos and images for final scenario testing of the designed detectors in the wild.

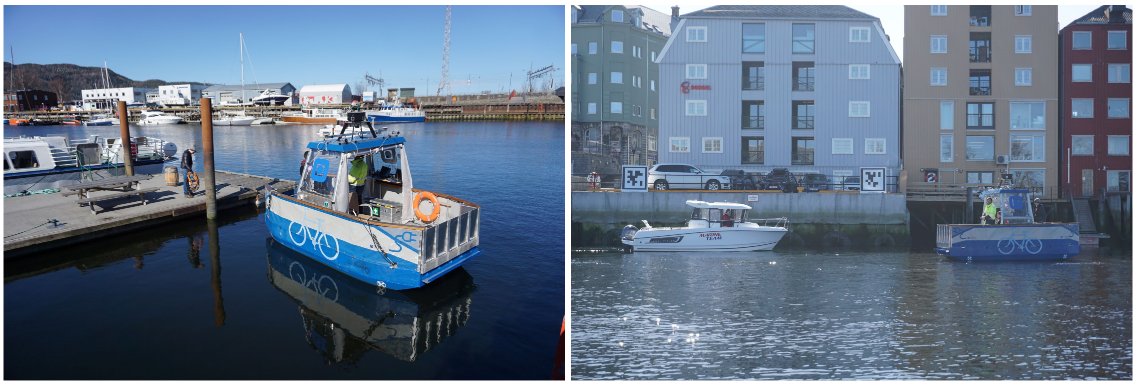

(a) milliAmpere electrical autonomous ferry (b) Experiment overview

Figure 4.13: Experiments in Ravnkloa and Trondheimsfjorden. The left-most boat in the right image is the rented marker boat.

4.5.1 Custom dataset design

Grini sampled data from the target domain in 2019, in particular the dataset named Trondheimsfjorden and MooredBoats (see table 4.1). These datasets, with boat labels only, are included in our designed target domain dataset. Simply, as to get more labeled data, while also ensuring a more diverse dataset which is less prone to sample-selection bias caused by sampling images over one experimental day only. In the splitting of the dataset, later described, it is assured that samples from Grini are represented in the training, validation and test split.

At the day of collecting data, a rather limited amount of vessel types were encountered. Overall, with the marker boat, a total of eight unique moving instances were encountered, not taking into account some samples of different docked boats. The instances vary between classes such as dinghy, ferry, and different sized motorboats. Most of these instances were captured on video and often occur alone in the foreground in the scene. In other words, many of the videos are essentially a single-object detection problem. The reason for this, is simply that there were few occasions for capturing moving multi-objects. However, with the aid of the marker boat and other more fortunate situations, several multi-object videos were acquired.

After assessing all the collected videos and frames, it was decided to add another object-type frequently faced in the target domain. Namely, samples of kayaks, which is one of vessel types of smallest object size and which often passes by in close proximity of surrounding boats, making it a challenging and important object to properly detect. To acquire such data, it was decided to re-use existing data, rather than waiting to cross kayak boats during additional experiment days.

One open-access video passing through Nideleven and into the harbour at Skansen[4], was scraped from Youtube. Three additional videos were provided by Erik Wilthil and Øystein Helgesen, acquired in challenging lighting conditions from the EO camera station at milliAmpere. A total of eight kayak videos are generated of these two unique instances, with largest possible variation in object-viewpoint, distance to object, and varying background. The two kayak instances are visualized in figure 4.14, though we note that a lot of different object-viewpoint samples are added in the final dataset.

[4]`https://youtu.be/TrpChlmCe_8`

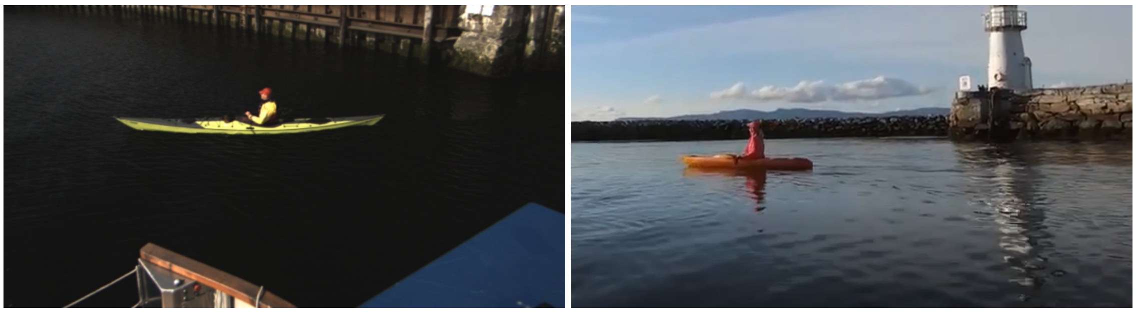

(a) Kayak sample milliAmpere EO camera station (b) Kayak sample scraped from Youtube

Figure 4.14: Kayak samples in target domain

In total, after filtering out videos considered less useful, it remains a total of 29 videos, where eight are kayak videos. Additionally, 32 still images are added containing more harbour-like conditions of docked boats and similar. Instead of using a few videos with long duration it was intentionally chosen to select many videos of short duration, as to mitigate too similar object appearance in the succeeding step of the frame generation. All videos are downsampled to 5 FPS for frame-generation, as to avoid oversampling of instances, considering the generated frames are to be merged with data from Grini. Another benefit of using this frame-rate follows in the dataset splitting.

All the generated frames are labeled using the intuitive LabelMe [89] annotation tool. In our labeling methodology the most important point has been to lable as class-aware as deemed useful as well as fitting the bounding boxes as tight as possible to the enclosed objects, touching one of the object boundaries on each side. As follows, four separate classes are separated between; *Kayak*, *Motorboat*, *Dinghy* and *Ferry*.

One example of a labeled motorboat as seen from the LabelMe user-interface is presented in figure 4.15. For kayaks or boats with a clear contour from someone manoeuvring the vessel, the driver has been included in the bounding box as we argue these features are useful for recognizing in particular kayaks and dhingys.

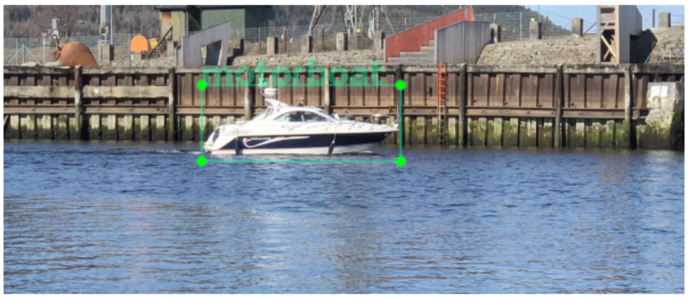


Figure 4.15: Sample image annotated from LabelMe

In a similar manner, the Trondheimsfjorden and MooredBoats dataset of Grini, previously agnostic labeled with the *Boat* label, are relabeled. As the dataset consists of a variety of different boat-types, it is found that only relabeling ferries, which there are notably many of, is the most meaningful measure as to attempt creating a more balanced class-aware target domain dataset. A total of 118 ferry instances are relabeled. Additionally, it is observed that Grini has labeled sailboats inconsistently both with the inclusion of the masts but also without. This was not found time to correct.

To summarize, table 4.15 displays all labeled images from each dataset. The accumulated images from Trondheimsfjorden and MooredBoats are denoted as *Grini target*. As the "Dinghy" class is a severe minority class, 39 dinghy samples are merged into the motorboat class of *Custom collected*.

Dataset split To design the training, validation and test split it is considered vital to avoid

Dataset	Images	Motorboat	Ferry	Kayak
Grini target	441	593	118	-
Custom collected	620	340	225	154

Table 4.15: Overview target domain dataset. *Images* refer to labeled images. Each class name refers to labeled instances.

sample leakage between the training, and validation-test set. Particularly since the target domain dataset is rather small, it would prove detrimental to have too similar images in the training and validation-test set, potentially causing a trained model to overfit.

The first measure to counter this, is to designate every video to a one of the split sets. As follows, all generated frames from a video is uniquely enclosed in one of the split sets. Nevertheless, since many of the videos capture the same object instances, it is found further necessary to manually ensure that the observed appearance between the videos in training and validation-test, are not too similar. Mainly, object appearance, distance and viewpoint are used as metrics for this procedure. One disadvantage of designating videos to each split, is that ensuring a strict percentage of frames to each split is significantly harder, considering that the videos are of different length and represents different amount of frames.

Moreover, our custom collected dataset contains more samples around Ravnkloa, which is considered closer to the target domain than many of Grini's open-sea images from Trondheimsfjorden. We therefore make sure the validation and test set contains sufficient of our collected images. Some videos are also strictly reserved to the test set as they represent particularly interesting case-study scenarios.

Grini's frames, also observed to be largely generated from videos, are manually split into a training, validation and test split with the objective of creating a 75-10-15 % split, evenly distributing the respective class instances over the split sets. The custom collect data is split by designating videos to each split set, while conserving even class distribution over the split sets and subjectively ensuring not too similar object appearance and viewpoint occurring in the training and validation-test set.

The final split is presented in table 4.16, and as the reader might observe, our collected dataset ended up with a larger validation and test set than intended, due to the video designation of each split set.

Dataset	Train	Validation	Test	Total
Grini target	336	47	58	441
Custom collected	350	113	157	620
Target domain dataset	686 (64.7%)	160 (15.1%)	215 (20.2%)	1,061

Table 4.16: Overview target domain dataset split by sub dataset. The count per set refers to labeled images.

Conclusive remarks The process of generation the target domain dataset has been both time-demanding and laborious, demonstrating for the author and hopefully for the reader, that collecting and annotating a well-representative dataset from a domain is not trivial. The author's data collection, frame generation, labeling and split set construction with the Grini dataset images, represents between one-two weeks of workload, including the planning process for the dataset collection day. Optimally, the target domain dataset would have been collected over more points in time and with a larger variety of weather conditions and unique instances, to better represent the target domain. Due to time-constraints, this was not achievable.

CHAPTER 5 EXPERIMENT DESIGN

It is desired to design our experiments as to benefit from the maritime datasets covered in chapter 4, in a manner improving detection performance in the target domain, in line with section 1.1. We adopt targeted detection pre-training [91] [60] to address this. Considerations for adapting this scheme is covered in section 5.1.

Section 5.2 present all datasets utilized for the conducted experiments in chapter 6. Including useful dataset statistics and each experimental dataset's relation to the target domain dataset.

As specified in section 1.1, the objective of this thesis is additionally to design a detector adhering to real-world deployment requirements. Section 5.3 explains in detail our procedure for choosing such a detector based on deployment requirements and available implementations/frameworks.

To the best of our ability, all details necessary for reproducing our experiments with the chosen detector implementation are presented in section 5.4. In addition to presenting baseline hyper-parameter settings, data pipeline considerations and the GPU cloud framework, three different fine-tuning strategies utilized in the experiments, are in detail covered.

Lastly, the utilized performance metrics for reporting of the experiments are presented in section 5.5.

5.1 Targeted detection pre-training

Targeted detection pre-training has proven to be beneficial for improving detection performance when fine-tuning into a target domain. Even though the performance gain is often linked to the size of the pre-training dataset [60], diversity and object variety are found to similarly be important [91].

None of the explored datasets in chapter 4 can be considered large-scale seen in relation to for instance OpenImages [55] or Objects365 [91] treated in section 3.2.3. Nevertheless, in maritime object detection, many of these datasets are undoubtedly large-scale. In order to utilize these datasets in a manner benefiting target domain performance, targeted detection pre-training is an unexplored and interesting approach.

Optionally, it would be possible to combine all the explored datasets in chapter 4, including the target domain dataset, to maximize training data and establish a large-scale maritime dataset. Such an approach is however problematic.

Firstly, the target domain dataset would be underrepresented, making model selection unreliable while likely causing unexplainable predictions on deployment in the wild. Moreover, the marginal distribution of the dataset would be strongly skewed towards the mixed source domain marginal distributions, limiting target domain performance [94]. Furthermore, the split sets would suffer from a sample-selection bias and a covariate shift. Lastly, there is no consistent way of linking transferability between each source domain and the target domain.

Targeted detection pre-training is defined according to transfer learning terminology in section 3.2, as follows for this thesis:

Definition 5.1.1 (**Targeted detection pre-training**). "Consider a maritime source domain $\mathcal{D}_{SM}$ unequal but of close proximity to a maritime target domain $\mathcal{D}_{TM}$, where $P(X_S) \neq P(X_T)$. Both domains contain objects of the same class, implying the same label space $\mathcal{Y}$. The learning tasks are the same, $\mathcal{T}_{SM} = \mathcal{T}_{TM}$, namely detecting objects. How does pre-training an object detector on $\mathcal{D}_{SM}$ improve the performance of the learned target predictive function $f_{TM}(\cdot)$ during fine-tuning on $\mathcal{D}_{TM}$?"

A source to target domain measure is not quantitatively defined, but the maritime source domains are assumed similar and close to the target domain. According to DA definitions, we additionally note that a stricter definition includes a covariate shift between $\mathcal{D}_{SM}$ and $\mathcal{D}_{TM}$, implying $\mathcal{T}_{SM} \approx \mathcal{T}_{TM}$ in definition 5.1.1.

Each experimental dataset and corresponding source domain, as well as the target domain, are covered in section 5.2. Moreover, as the target domain dataset is small-scale, additional fine-tuning options to full fine-tuning are meaningful to consider, as later presented in section 5.4.

5.2 Experimental datasets

This section covers details regarding each experimental dataset and their relation to the target domain dataset. Each experimental dataset is representative of a source domain related to the target domain.

Chapter 4 presents four maritime datasets from section 4.1 - 4.4 and the target domain dataset in section 4.5. We consider Hurtigruten$_{BL}$ (section 4.2), the Hurtigruten dataset (section 4.3), and the SMD (section 4.4) for generating experimental datasets. The Grini dataset is not considered, as this already is used to generate the target domain dataset.

Due to inherent class-imbalance for most of these datasets, all datasets are treated class-agnostic, with one generic vessel class; *Boat*. This condition also fulfills definition 5.1.1 in section 5.1, asserting similar label space for source domains and the target domain.

All of the experimental datasets only contain a training and validation split. This is done to maximize training data and as the models trained on these datasets are not intended for performance benchmarking, but model selection only. Therefore, an out-of-sample dataset is not strictly required, and is omitted.

All of the experimental datasets and the target domain dataset are presented in table 5.1, together with the respective amount of object size ratios of all the ground truth bounding boxes as defined by the COCO area classification, covered in section 5.5, in table 5.2.

Singapore Maritime Dataset (SMD) is essentially our post-processed version of the SMD in table 4.14. The SMD presents quite similar object viewpoints as the target domain dataset, even though most instances are captured in open-sea environments with a clear horizon. The majority of the instances are based on ships and vessels of different sizes and includes a total of 76,596 ground truth instances from the total 10,394 labeled images.

Nordic Maritime Dataset (NMD) consists of combined images from the post-processed Hurtigruten dataset split in table 4.10 and Hurtigruten$_{BL}$ (randomly split into train and validation) both containing images from Nordic maritime waters. The NMD is designed as to subjectively represent a source domain similar to the target domain dataset. However, the majority of the images, originating from Hurtigruten dataset, represents a quite different object viewpoint, though providing a large variety of different boat types, weather conditions and backgrounds.

All non-boat type classes are deleted from both of the datasets before agnostic relabeling of the boat classes to *Boat*, which explains some of the observed discrepancies in the image count of *Hurtigruten* in table 5.1 compared to table 4.10. The bounding box areas of the NMD are more skewed towards small instances than observed in the SMD. What is more, there are generally fewer instances per image, counting a total of 18,802 instances distributed over 7,477 labeled images.

Mixed Maritime Dataset (MMD) is designed as to maximize dataset size, while disregarding the previous finesse of clearly separating the source domains. MMD is simply generated by combining the training and validation set of SMD and NMD, thus representing a mix of the two source domains. As follows, the images contain a large variety of boat types from different maritime environments. A total of 95,398 ground truth instances are available in the 17,871 labeled images. As follows, the object size ratios of the MMD inherits its properties from the two underlying datasets.

Target domain is the custom designed target domain dataset with the following split previously presented in table 4.16. The target domain dataset has quite few instances per image, counting a total of 1,440 instances and 1,061 images. Thus, many of the images contain a single object, which is explanatory from the collected images by the author, covered in section 4.5.1. Moreover, from table 5.2 the target domain dataset is observed to to include significantly fewer small instances, and more large instances, than the experimental datasets.

Name	**Sub dataset**	**Train**	**Val**	**Test**	**Total**	**Instances**
SMD	-	7,478 (71.9%)	2,916 (28.1%)	-	10,394	76,596
NMD	*Hurtigruten*	5,845	919	-	6,775	17,667
	Hurtigruten$_{BL}$	606	107	-	713	1,135
	Total	6,451 (86.3%)	1,026 (13.7%)	-	7,477	18,802
MMD	-	13,929 (77.9%)	3942 (22.1%)	-	17,871	95,398
Target	-	686 (64.7%)	160 (15.1%)	215 (20.2%)	1,061	1,440

Table 5.1: Experimental datasets and target domain dataset. MMD is the combination of SMD and NMD. *Total* refers to total labeled images. *Instances* refers to total amount of instances. All percentages in *Train* and *Val* are calculated from *Total* on the same row.

Name	**Small**	**Medium**	**Large**
SMD	29.8%	53.0%	17.2%
NMD	48.6%	40.7%	10.7%
MMD	33.5%	50.6%	15.9%
Target	10.5%	60.2%	29.7%

Table 5.2: Experimental datasets and target domain dataset. Object size ratios of all ground truth bounding box areas, COCO classified.

5.3 Detector considerations

In this thesis, several architectures have been considered with inspection of the respective codebases. In particular, it has been prioritized to choose a state-of-the-art detector with several stable and well-tested implementations. The EfficientDet [101] architecture is finally selected for performing all experiments. We highlight some of the advantages in using this architecture.

- *State-of-the-art*: EfficientDet, even though published in 2020, still reports performance close to the most recently published architectures [108].

- *Flexible architecture*: One of the unique features of EfficientDet is its flexibility. EfficientDet is actually a family of detectors and by simply changing one parameter in the implementation, the total architecture and resulting inference time and performance metrics are altered accordingly, as presented in section 2.4.3.
- *Existing implementations*: Since the detector has existed for over a year at the writing of this thesis, there are plenty of implementations to choose between.

After deciding on the detector architecture, the next step is to decide on an implementation of the architecture to use. In choosing an implementation several criteria have been considered.

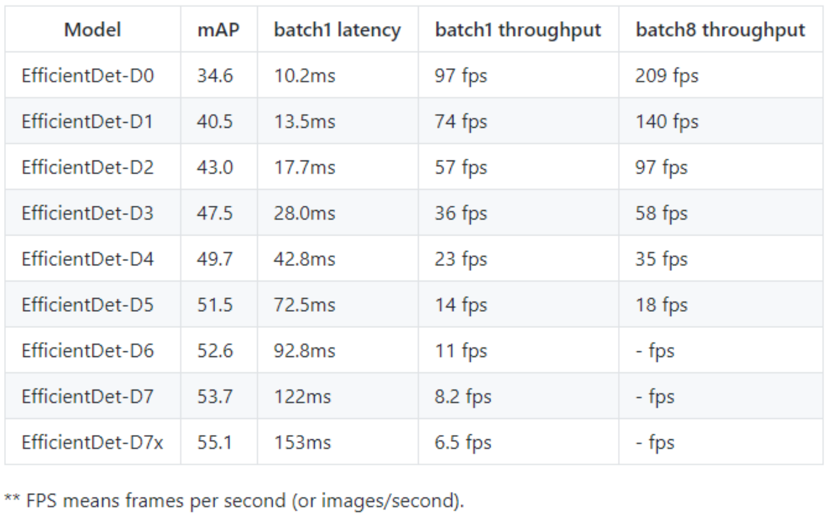
- *Reported performance*: Does the implementation report similar performance scores as the original EfficientDet paper [101].
- *GPU compatibility*: Which GPU resources are compatible with the implementation.
- *Fine-tuning strategies*: Which fine-tuning strategies are available in the implementation.
- *Community adoption*: Is the implementation well-adopted by the computer vision community on GitHub.
- *Framework preferences*: Which deep learning framework is used for implementation. We separate between PyTorch [75] and TensorFlow [10].

Two different implementations satisfying our requirements are tested. Firstly, the original TensorFlow implementation [2], which provided the reported performance scores in [101], is tested. Secondly, a PyTorch re-implementation [5] motivated by the author's preference for PyTorch, is tested. The GBL_3-720 dataset, created in the author's specialization project [56] and consisting of 2360 images of 720×720 resolution with three classes, is used for sanity-checking the implementations' performance. As reported in [56], we know the Faster R-CNN achieves an mAP of 20.2 on this dataset. Based on reported COCO scores, we expect EfficientDet-D2 (see Table 2 [101]) to perform better, or on par, with the Faster R-CNN mAP score (see Table 11 [86]).

The PyTorch re-implementation reports similar performance scores as the original paper. Moreover, the PyTorch framework accepts annotation files on the JSON format, easily convertible from the PASCAL VOC XML format. Multiple models are trained on the benchmark dataset for this implementation on Google Colaboratory with delivered GPU resources. We test the D0, D1, D2 and D3 EfficientDet models for different optimizers and learning rates. Training from the pre-trained COCO weights is very slow and converges surprisingly slow to any satisfactory performance scores. As neither D2 or D3 improves on the Faster R-CNN benchmark score and as the sample predictions are inherently untrustworthy, we refrain from using this implementation.

The TensorFlow original implementation requires the transformation of JSON annotations to TensorFlow records (TFRecords). This is trivially performed from the suggested transformation script provided in the GitHub repo [2]. Similar tests are performed for this implementation with more promising results. A must faster convergence from the COCO pre-trained weights is promising, and similarly we find the original hyperparameters [101] to outperform the Faster R-CNN benchmark score for EfficientDet-D2 and D3, which is expected. Sample predictions and loss curves are also observed to be reliable.

As follows, the TensorFlow original implementation is selected. The inference time for all EfficientDet models is presented in figure 5.1. More details regarding this implementation is treated in section 5.4.



Model	mAP	batch1 latency	batch1 throughput	batch8 throughput
EfficientDet-D0	34.6	10.2ms	97 fps	209 fps
EfficientDet-D1	40.5	13.5ms	74 fps	140 fps
EfficientDet-D2	43.0	17.7ms	57 fps	97 fps
EfficientDet-D3	47.5	28.0ms	36 fps	58 fps
EfficientDet-D4	49.7	42.8ms	23 fps	35 fps
EfficientDet-D5	51.5	72.5ms	14 fps	18 fps
EfficientDet-D6	52.6	92.8ms	11 fps	- fps
EfficientDet-D7	53.7	122ms	8.2 fps	- fps
EfficientDet-D7x	55.1	153ms	6.5 fps	- fps

** FPS means frames per second (or images/second).

Figure 5.1: Reported inference time EfficientDet models, TensorFlow original implementation. mAP is reported on COCO test-dev.

Source: [2]

5.4 EfficientDet TensorFlow configuration

In this section, all details and considerations necessary for reproducing our experiments in chapter 6, with the chosen EfficientDet implementation, are documented. More specific hyperparameter tuning is in-detail reported in the respective experiment sections of chapter 6.

Model selection *EfficientDet-D3* is selected for all experiments.

This choice is based on the inference time requirements treated initially in section 1.1. In essence, we search for an architecture which can guarantee a frame-rate of 25 FPS or higher for a batch size of one image. From the reported inference times from figure 5.1, *EfficientDet-D3* is the best option guaranteeing this.

GPU resources All experiments are trained in a Google Colaboratory pro subscription, enabling 24 hours of training on one single GPU. As the acquired GPU differs, depending on available resources from Google, we do not report any GPU specifications.

Hyperparameter baseline configuration In the initial testing of the EfficientDet TensorFlow implementation, we find that most of the original hyperparameters from [101] works well and are hence reused. The hyperparameters presented here are either changed from the original hyperparameters or presented due to their importance. We do note that exponents of base number 10 is written as e, i.e 10^{-3} = e-3.

All models are trained with Stochastic Gradient Descent (SGD) with a weight-decay of 4e-5 and a momentum of 0.9. We apply a warm up learning rate from 1e-4 linearly increased until reaching the default learning rate of 1e-3 over the first training epoch. During subsequent training the learning rate is annealed down with cosine decay annealing [65] to improve SGD performance. The SiLU(Swish-1) activation function [81] is used without exponential moving average decay, as initial tests found no benefit in this, dissimilar to the original paper [101]. Gradient clipping is further adopted to mitigate exploding gradients.

Random horizontal flipping and scale jittering are used as baseline data augmentations for each inputted image, similar to [101]. While the horizontal flipping augmentation is quite self-explanatory, scale jittering crops each image after randomly resizing it from a value in the range [0.1, 2]. For normalizing pixel values, the dataset RGB mean and standard deviation parameters are overwritten from calculations of each dataset used for training.

Hyperparameter	**EfficientDet reference**
Learning rate	learning_rate: 1e-3
Warm up learning rate	lr_warmup_init: 1e-4
Warm up epochs	lr_warmup_epoch: 1.0
Decay method	lr_decay_method: 'cosine'
Mixed precision	mixed_precision: false
Exp. moving average decay	moving_average_decay: 0
Horizontal random flipping	input_rand_hflip: true
Scale jittering min	jitter_min: 0.1
Scale jittering max	jitter_max: 2.0
RGB mean	mean_rgb:
RGB standard deviation	stddev_rgb:
Anchor box scale	anchor_scale: 4.0
Anchor box aspect ratios	aspect_ratios: [0.5, 1.0, 2.0]

Table 5.3: EfficientDet TensorFlow implementation hyperparameters specified in config file to obtain baseline hyperparameters. RGB mean and standard deviation are left empty as these hyperparameters are adjusted for each dataset.

The default anchor box parameters of [101] are utilized, with default anchor box scale of 4.0 and aspect ratios of [0.5, 1.0, 2.0].

Mixed precision is disabled during training due to the randomly allocated GPU resource of Google Colaboratory. All training is further performed with a batch size of 4 and gradient checkpointing to reduce GPU memory requirements. Gradient checkpointing essentially stores checkpoints in the computation graph of TensorFlow to calculate the gradients at backpropagation for less required GPU memory, which is found necessary for training EfficientDet-D3 on one GPU resource.

Table 5.3 presents the specified hyperparameters in the config file, which together with default settings in the config file *hparams_config.py* [2] produces the baseline hyperparameters.

Dataset pipeline In the data processing during training time, each image is resized to the target image size of EfficientDet-D3 of 896×896 by bilinear interpolation. All image resizing is performed in the training pipeline.

All datasets are locally shuffled within the training, validation and test set. All dataset annotations are thereafter converted to the TensorFlow annotation format, TFRecord, which stores sequences of images and annotations in *shards*. All shards within a split set are additionally globally shuffled during training. The amount of images and annotations within one shard depends on the available images in the split sets, but is typically between 700-1000 images.

An epoch, signifying one pass over the training set, is calculated by taking the training iteration step multiplied with the batch size, divided by the training dataset size.

EfficientDet fine-tuning strategies The chosen TensorFlow implementation of EfficientDet-D3 [2], provides hyperparameter options for altering the training strategy of the detector. The TensorBoard graph of the main building-blocks is visualized in figure 5.2. In this thesis, we separate between three different training strategies as follows:

1) *Full fine-tuning(FF)*: All parameters are subject to training.

2) *Frozen backbone(FB)*: All parameters except the backbone are trainable.

3) *Head only(HO)*: All parameters are frozen except the class/box prediction heads.

To give some more detail into these different fine-tuning strategies, we provide a more in-detail explanation of the EfficientDet architecture before specifying the detector-specific parameters used to enable each respective fine-tuning scheme.

The EfficientDet-D3 architecture is presented in figure 5.2 with its core components. All of these components are the same for all scalable versions of the EfficientDet detector family. The flow for EfficientDet-D3 of each component transpires in the following way. An image of 896×896 is inputted to the architecture. The EfficientNet-B3 backbone downsamples the image by a factor of two for each feature level, for a total of seven levels. Each level is denoted as Pi where $i \in [1, 7]$. In other words, the first feature layer, $P1$ represents a feature map downsampled by two with spatial resolution 448×448, while $P7$'s feature maps are 7×7.

As stated in the original paper [101], the BiFPN accepts features from the P3-P7 level. This stream of features are represented by the right-most arrow connecting the backbone to the BiFPN as five unique tensors in figure 5.2. In addition, the reader can observe the two resampling blocks also connecting the backbone to the BiFPN, which are used for adding another coarse representation of the smaller and more semantically rich feature maps from the backbone.

The BiFPN has a repeated block of six fpn cells for the D3 architecture with 160 input channels. These values may also be observed from Table 1 [101], which presents the amount of channels and layers for the BiFPN and box/class network in accordance with the scaling factor $\phi = 3$. Finally, the box/class prediction heads which consists of two separate heads for predicting the object class and bounding box coordinates consist of four layers each for D3.

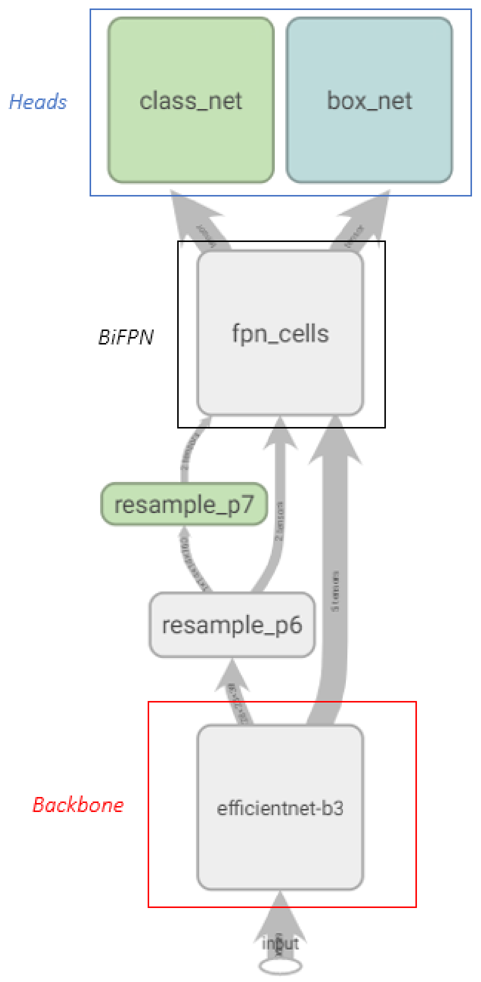


Figure 5.2: EfficientDet-D3 architecture TensorBoard graph. Annotated by author.

The config file parameters used for enabling each the fine-tuning options are specified in table 5.4. A few observations regarding the fine-tuning schemes are worth noting.

Firstly, the FF tuning scheme essentially is a traditional full fine-tuning scheme, where all parts of

Fine-tuning strategy	**EfficientDet reference**
FF	var_freeze_expr: None
FB	var_freeze_expr: '(efficientnet)'
HO	var_freeze_expr: '(efficientnet|fpn_cells|resample_p6)'

Table 5.4: EfficientDet fine-tuning strategies with config file reference.

the detector are trained and the backbone overwrites previously pre-trained features.

The FB tuning scheme, essentially conserves pre-trained features in the backbone. Since the BiFPN has weighted edges connecting the FPN cells, the multi-scale feature fusion is adapted during training. In this manner, the architecture learns the best way of combining pre-trained features for better feature fusion.

The HO tuning scheme has a very restricted learning capability. In fine-tuning with this scheme, training is significantly faster as only the box and classification network parameters are updated. From table 5.4, only the *resample_p6* is included in the HO freezing expression. The P7 resampling layer is not frozen, as to also adapt the deepest layer with most domain specific feature to the training dataset.

As a last note, we discuss a potential gradual fine-tuning scheme of the backbone and why it has not been implemented. Firstly, a gradual-fine tuning, though indicating improved performance for some architectures, has no consistent scheme for guaranteeing this. Such an implementation would be dependent on the author's decisions regarding performance stagnation for unfreezing the succeeding layer as well as corresponding learning-rates. Additionally, implementing this would represent a time-demanding altering of the training scheme to make a reliable and automatized tuning scheme. The creators of the EfficientDet architecture have discussed implementing this tuning scheme on an opened GitHub issue [3], and similar to them, we refrain from this due to a most likely marginal impact.

Experimental data augmentation In the experiments, it is also experimented with modern and more experimental strategies for designing data augmentation schemes.

Firstly, a custom data augmentation technique designed for improving fine-tuning of object detectors by exploring adversarial examples is considered, named Det-AdvProp [21]. In the EfficientDet TensorFlow implementation, the training scheme is not available. However, the pre-trained COCO weights obtained using such a scheme, reporting +1.1 COCO AP improvement for EfficientDet detectors, are tested.

Secondly, an augmentation scheme called AutoAugment [24], which learns augmentation policies from datasets is tested. As the augmentation policies are dependent on the datasets reported by the paper [24], a custom policy proven to perform good for object detection, provided in the TensorFlow EfficientDet implementation, is tested.

K-means - anchor box aspect ratios An open-source implementation based on k-means for optimizing the anchor box aspect ratio parameters of the EfficientDet detectors as to have the best IOU to each training dataset's clustered ground truth bounding boxes, is partially utilized in the conducted experiments. The implementation [57] works in a similar manner to the YOLOv2 [84] anchor box prior algorithm.

5.5 Performance metrics and terminology

In this thesis we utilize the arguably most common performance metric in the computer vision community, based on the COCO benchmark dataset [61].

The choice of reported COCO metrics depends on which aspect of model performance that is desired to measure, which will differ subject to the experiment's nature. We specify the terminology that is used for all following experiments in this project thesis which follows the terminology from [1].

- AP: Refers to the AP-defined metric in [1] for all reporting purposes. This is the same as $mAP@[0.5 : .0.05 : 0.95]$ and is the main metric in the COCO detection challenge.
- AR: The mean average recall for same IOU thresholds as AP.
- AP_{50}, AP_{75}: Refer to the mean average precision for IOU threshold 0.5 and 0.75.
- AP_s, AP_m, AP_l: Similar to the first defined AP, but for objects of small, medium or large size. The sizes are specified in table 5.5.
- AP_{class}: Reports the average precision for the class, for the same IOU thresholds as AP.

Do note: AP reports the mean average precision for all classes, which can be seen as the mean of AP_{class} for all classes.

COCO area definitions

Area classification	Area size (A)
Small	$A < 32^2$
Medium	$32^2 < A < 96^2$
Large	$A > 96^2$

Table 5.5: COCO area definitions [1]

Inference time / latency Inference time is reported for a batch size of one, as to best mimic the intended sequential image feeding in deployment on milliAmpere, as discussed in section 1.1. Moreover, inference time is reported as end-to-end latency of EfficientDet-D3. That means, the inference time includes image preprocessing, pure inference time, post-processing and NMS. This is the only realistic inference time to report, as opposed to only pure inference time, since deployment on milliAmpere would need to include all these aspects in generating a prediction.

Loss monitoring The total validation loss of EfficientDet-D3 is reported to capture overfitting behaviour and monitor the each trained model's generalization to the validation set. Also the total training loss is monitored, to observe learning improvements over time and as to be seen in relation to the validation loss.

Model weight saving Model weights are saved from observations on validation loss stagnation and validation AP curves during training time.

Confidence score threshold All predictions presented in this thesis is performed for the default classification confidence score threshold of 0.4 for the EfficientDet TensorFlow implementation.

CHAPTER 6 EXPERIMENTS

In this chapter, all experiments are conducted and reported.

6.1 Experiment overview

As a continuation from the experiment design in chapter 5, this section is dedicated to present the experiment overview and details. As presented in the experiment overview in table 6.1, the experiments are initially separated into two phases.

The first phase, referred to as the *unseen target domain* experiments, is designed for generating the targeted detection pre-trained weights for each of the experimental datasets treated in section 5.2. The EfficientDet-D3 detector, initialized from COCO pre-trained weights, is fully fine-tuned on the three experimental datasets for the class-agnostic *Boat* class. All details and considerations for training, hyperparameter tuning and model selection are covered in section 6.2.

The resulting models, or the *unseen target domain* models, are later evaluated on the target domain test set in section 7.1. The thought behind doing so, is to observe how a model trained on a maritime source domain transfers into the completely unseen target domain. As such, shedding light on the pure transferability between each of the source domains and the target domain.

The second phase, referred to as the *seen target domain* experiments, are the main experiments of this thesis. The three explored fine-tuning settings of section 5.4; full fine-tuning (FF), frozen backbone (FB) and fine-tuning the class/box prediction heads (HO) are tested for fine-tuning into the target domain. This is performed for four different pre-trained weights; baseline weights from pre-training on COCO, and the three resulting detection pre-trained weights acquired from the first phase. In total, 12 models are therefore fine-tuned on the target domain.

The models obtained from the second phase, the *seen target domain* models, are evaluated on the target domain test set in section 7.2. This experiment design provides a comparable basis between fine-tuning settings, the targeted detection pre-trained models and the baselines.

For both of the experiment phases, model training has been conducted in a sequential manner, where a model often build on hyperparameters from a previous one. The respective experiment sections therefore present partial results regarding the choice of model, as to motivate the author's reflection process in conducting the experiments.

Lastly, not covered in detail in this chapter, an ablation study on the effects of fine-tuning into the target domain when introducing class-awareness, is additionally executed and evaluated on the target domain test set in section 7.3.

	Model	Weights	Dataset	Fine-tune settings
1. Unseen target domain	D3-SMD	COCO	SMD	FF
	D3-NMD	COCO	NMD	FF
	D3-MMD	COCO	MMD	FF
2. Seen target domain	D3$_{\{\cdot\}}$-$\mathcal{D}_b$	COCO	Target	HO, FB, FF
	D3$_{\{\cdot\}}$-$\mathcal{D}_{SMD}$	COCO $\Rightarrow$ SMD	Target	HO, FB, FF
	D3$_{\{\cdot\}}$-$\mathcal{D}_{NMD}$	COCO $\Rightarrow$ NMD	Target	HO, FB, FF
	D3$_{\{\cdot\}}$-$\mathcal{D}_{MMD}$	COCO $\Rightarrow$ MMD	Target	HO, FB, FF

Table 6.1: Experiment overview. *Weights* are initialized pre-trained weights before training on the training dataset, specified in *Dataset*. *Transfer settings*: HO=Head-only, FB=Frozen backbone, FF=Full-fine tuning. Empty curly bracket subscript for phase 2 models is either HO, FB or FF, depending on the fine-tuning setting used for each model.

6.2 Unseen target domain

The first experimental phase covers the experiments for obtaining the targeted detection pre-trained weights. The EfficientDet-D3 model is fully fine-tuned on each of the experimental datasets covered in section 5.2 and evaluated on the respective validation sets.

As it is desired to fit each unseen target model to its respective experimental dataset in the best manner, specific hyperparameters and data augmentation schemes are in-detail reported. For simplicity, the hyperparameter tuning is divided into two separate parts.

Firstly, hyperparameters strictly dependent on the training dataset, such as aspect ratio and dataset mean and standard deviation adjustments, are referred to as *dataset-specific hyperparameters*. Secondly, hyperparameters related to model training, mainly treating learning rate considerations, are referred to as *model-specific hyperparameters*. This partitioning is clearly not mutually exclusive, as all hyperparameters inherently depend on the dataset. Nevertheless, it is chosen as to structure the experiments clearer.

Validation losses and sample predictions are presented where considered informative. Unless otherwise specified, all baseline settings from section 5.4 are adopted. Lastly, all model names used for the unseen target model experiments in section 6.2.1 - 6.2.3, are local in the scope of the respective section and the same model names may be reused in several of the sections. The final best reported model in each section is evaluated on the target domain test set in section 7.1.

6.2.1 Singapore maritime dataset

The initial experiments covers the models trained on the SMD dataset, treated in section 5.2. As this is the first large-scale maritime dataset used for training EfficientDet-D3, a more detailed discussion of the hyperparameter reflection process is presented. The same reflection process is treated more briefly in the following unseen target experiments.

Lastly, the AdvProp pre-trained weights covered in section 5.4 are tested, as to make a further basis of the feasibility and effect of using these pre-trained weights.

Dataset-specific hyperparameters

The first dataset-dependent hyperparameters necessary to modify, are the dataset colour distribution mean and standard deviation, denoted by the three colour channels red, green and blue as RGB. These values are used in the normalization of pixel values and are essential to adjust from the default values. The mean and standard deviation values are easily found by calculating the mean and standard deviation over all image pixels for all images and respective colour channels. The SMD specific RGB mean and standard deviation values are presented in table 6.2.

SMD RGB mean and std	
Mean	**Std**
[136.570, 167.393, 180.491]	[43.315, 36.592, 40.503]

Table 6.2: SMD RGB mean and standard deviation.

As EfficientDet is an anchor-based detector, it is very important to ensure that the default anchor boxes satisfactory cover the bounding boxes of the instances in SMD. The anchor boxes are configured by their scale and aspect ratio. Without configuring these two parameters properly, certain objects may be completely impossible to detect as their scale or bounding box aspect ratio are not covered by the EfficientDet anchor boxes.

Figure 6.1 presents the aspect ratios and bounding box areas for all instances in the training and validation set of the SMD. All instances are adjusted for the bilinear interpolation image resizing to 896×896 performed for the EfficientDet-D3 in the TensorFlow implementation. The reader should notice that bilinear interpolation, unlike other padding-based resizing methods which conserves image aspect ratios, impacts the bounding box aspect ratios and bounding box areas of images where the height and width are unequal.

The aspect ratios in figure 6.1a are presented in a barplot format to better observe the accumulated aspect ratios which are below the lowest default value (0.5) and above the largest default value (2.0). The training and validation set are observed to represent the same ratios of aspect ratios for each of the aspect ratio ranges, indicating the instances in the training set well represents the instances in the validation set.

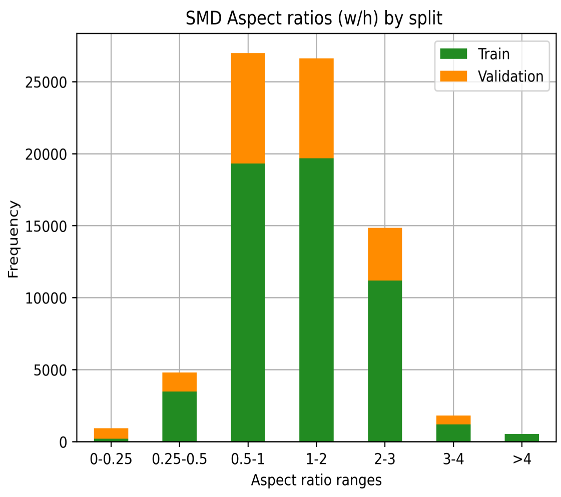
There is reason to believe a larger aspect ratio might be beneficial. In particular, from the columns larger than and including "2-3" in figure 6.1a, over 15,000 instance labels are most likely not satisfactory covered by the largest default anchor aspect ratio. Moosbauer et al. attempted to add an additional anchor aspect ratio of 3.0 to the Faster R-CNN architecture to address this observation. However, pointing out that the SMD is insufficiently large for training all network parameters with this additional anchor aspect ratio. We argue that the same problematic is likely to be present for EfficientDet-D3 with our processing of the SMD, which is also smaller in size than of Moosbauer et al.

To better visualize the anchor aspect ratios and the distribution of the ground truth bounding boxes, the ground truth bounding box aspect ratios are plotted in the format of a scatter plot in figure 6.2. It is observed that most of the instances are well-covered by the default anchor aspect ratios. However, there are several instances with aspect ratio (w/h) larger than 2.0, below the green line in figure 6.2.

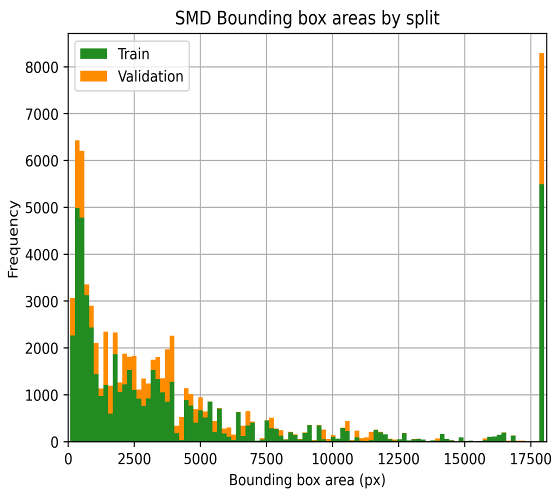
To avoid changing the default aspect anchor ratios in a manner affecting the instances already satisfactory enclosed, it is decided to attempt the k-means algorithm presented in section 5.4. The algorithm proposes the three following new aspect ratios; 1.0, 2.0 and 3.8 which for 6.53% less cases have an IOU below 50% with the ground truth bounding boxes.

The bounding box area plot in figure 6.1b is designed as a histogram plot, where very large areas ($>$ 18,000) are stacked in the right-most column of 18,000 to provide more fine-details regarding the smaller and medium sized areas. Based on the COCO area definitions, small areas are below 1,024 pixels, while medium areas are above this and below 9,216 pixels. All areas above 9,216 are defined as large instances. Also for the bounding box areas, the training and validation set are well representative for different values.

No adjustments are carried out for the default anchor box scale. Lowering the lowest anchor scale was considered, however due to the max size of objects in the dataset, we would risk not capturing the largest objects, which would not represent a good trade-off.



(a) Aspect ratio barplot distribution by split



(b) Area histogram by split

Figure 6.1: SMD bounding box aspect ratios and areas by split, after adjusting for bilinear interpolation image resizing to 896 × 896 . All areas larger than 18,000 pixels are added in the final histogram column of 18,000 in figure 6.1b. *Green* is training set samples, *orange* is validation set samples.

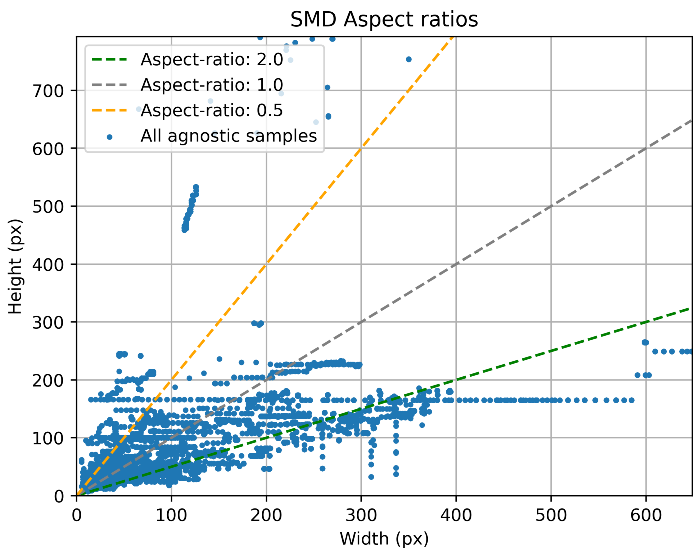


Figure 6.2: SMD ground truth bounding box aspect ratios. Scatter plot. Adjusted for bilinear interpolation resizing to 896 × 896. Aspect ratio lines are default anchor box aspect ratios.

Model-specific hyperparameters

Of the model-specific hyperparameters, only the learning rate is altered as to limit the amount of hyperparameters subject to tuning. The learning rate, regulating the step-size of the gradient in the selected SGD optimizer, has a large effect on the loss convergence and smoothness. A larger learning rate typically delivers faster model convergence on the risk of more unstable training and sub-optimal convergence. Contrarily, a smaller learning-rate generally needs more training epochs for convergence and risks getting stuck at local minimum points.

The best manner of finding a suitable learning rate is through empirical observations of the training and validation loss curves. Moreover, since the SMD is generated from 40 videos of rather similar object-viewpoint and background, it is pointed out that many of the frames are rather similar. Each batch of images may therefore, even after random shuffling, contain quite similar images. As follows, a low learning rate may be convenient for avoiding too fast initial convergence and potential overfitting.

We consider two different learning rates initially.

1) Default learning rate from table 5.3 of 1e-3.
2) A lower learning rate of 5e-4 with warm up learning rate of 1e-4.

The lower learning rate of 5e-4 is tested based on empirical observations from a couple of training runs.

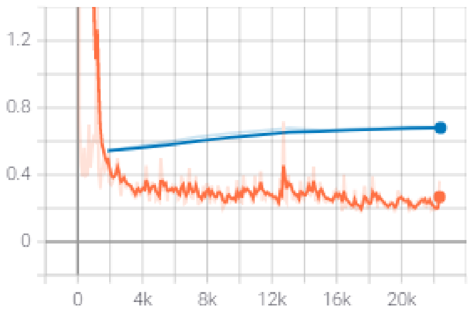
Model runs

All experiments are conducted and evaluated in a sequential manner to find the best set of combined hyperparameters. All models are trained with default batch size (4) and adjusted RGB mean and standard deviation from table 6.2. The amount of training epochs are adapted from observed validation loss stagnation, initially tested for 18 epochs for all models in this section.

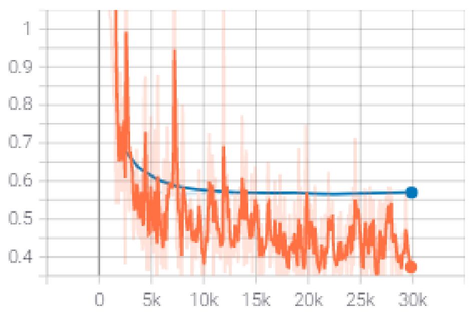
Initially, it is prioritized to find a suitable learning rate before adapting dataset dependent hyperparameters. The two models are referred to as follows:

1) D3-lr$_1$: Baseline settings and default learning rate of 1e-3.
2) D3-lr$_2$: Baseline settings with learning rate 5e-4.

Figure 6.3 presents the total training and validation loss for each of the respective models. The lower learning rate model, D3-lr$_2$, is observed to be most suitable as D3-lr$_1$ overfits the model after only a few thousand training iterations (around two epochs). As D3-lr$_1$ is undoubtedly overfitting, only D3-lr$_2$ is evaluated on the validation set as presented in table 6.3. The model weights are saved on epoch seven, or training iteration 13,076. In other words, passing over the dataset solely seven times is sufficient to obtain a well-fitted model, indicating the SMD is a rather simple dataset. Similar observations are found in [69], obtaining the best model after fine-tuning only one epoch for Faster R-CNN. The total training time is as approximately 4h30 minutes.



(a) Loss curves D3-lr_1



(b) Loss curves D3-lr_2

Figure 6.3: Loss curves D3-lr_1 and D3-lr_2 on SMD. D3-lr_1 stopped early due to overfitting. *Orange* is training loss, *blue* is validation loss. 0.6 TensorBoard smoothing.

In line with the experimental methodology, D3-lr_2 is adapted for testing the anchor box aspect ratios proposed by the k-means algorithm. We refer to this model as D3-KMM.

3) D3-KMM: D3-lr_2 hyperparameters with new anchor box aspect ratios of 1.0, 2.0 and 3.8.

The validation loss, strongly resembling the one of D3-lr_2 in figure 6.3b, is omitted. D3-KMM obtains the best reported model weights at epoch 17 and overall fails to match or outperform the evaluated validation set scores of D3-lr_2, as can be seen in table 6.3. This might be a consequence of the imprecise annotated ground truth bounding boxes of the SMD, as pointed out in section 4.4.1. Similarly, the smallest anchor box aspect ratio of 1.0 for D3-KMM, is most likely missing a lot of the smaller aspect ratio objects captured by D3-lr_2.

As introductory explained, the Det-AdvProp pre-trained COCO weights are also tested in this section. This is conducted as an initial test, both to assess the potential performance benefit of using these weights and observe practical implications during training. As such, D3-lr_2 is re-trained with the Det-AdvProp pre-trained weights, hereby referred to as D3-Adv.

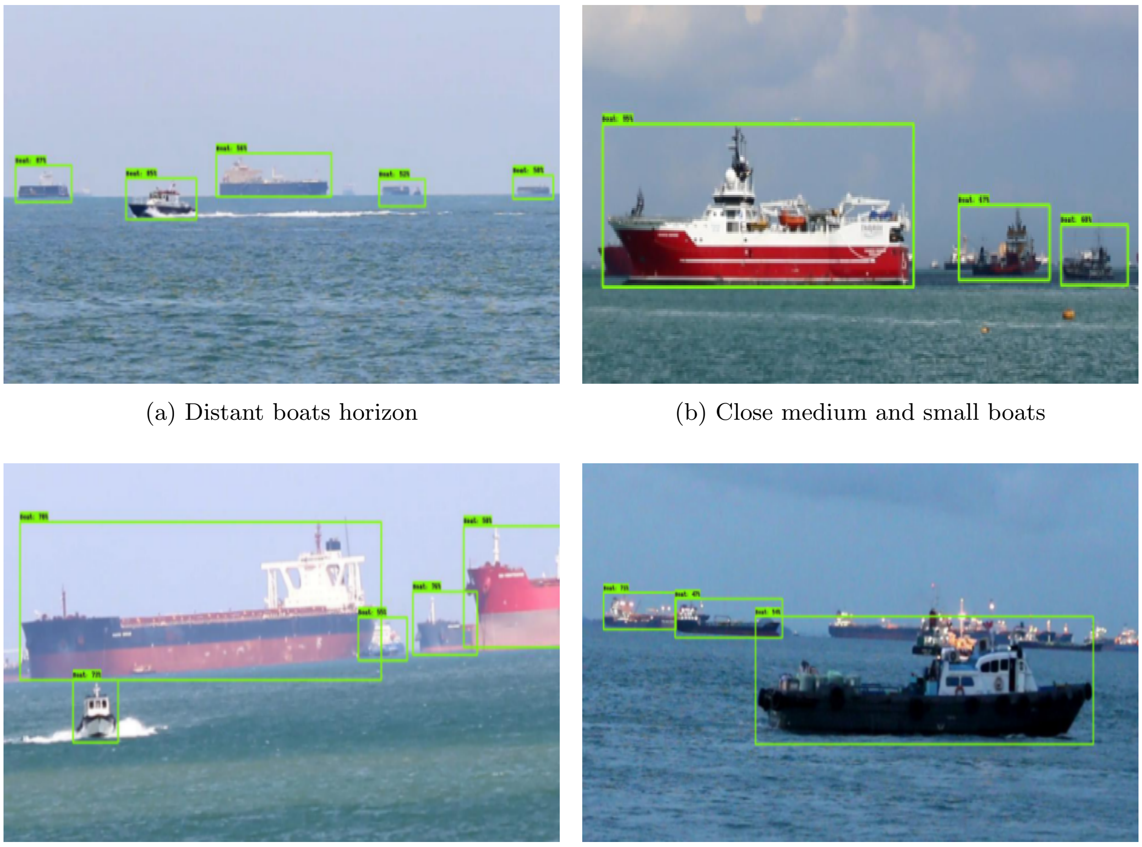
4) D3-Adv: D3-lr_2 hyperparameters with EfficientDet-D3 Det-AdvProp pre-trained weights.

A general observation from D3-Adv is the increased training time and slow validation AP convergence. The best model weights and evaluated validation AP score is found at epoch 17, after approximately 16h training, with a substantially worse performance on all reported validation scores than both D3-lr_2 and D3-KMM, as presented in table 6.3. It is possible a much longer training scheme and possibly higher learning rate could benefit better from the Det-AdvProp pre-trained weights. However, this approach is not further pursued due to time-constraints. In conclusion, the Det-AdvProp pre-trained weights are not found beneficial and are neither considered for any future experiments in this thesis.

Lastly, the reader has most likely observed the poor performance on the AP_s for all the four models. This observation indicates a decrease in anchor box scale could also be useful to test. Unfortunately, due to time-constraints, such a model was not found time to test.

	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	Epoch
D3-lr_2	**38.9**	**76.2**	**35.9**	**1.08**	**26.7**	**55.7**	7
D3-KMM	36.8	71.4	33.9	1.00	25.2	53.0	17
D3-Adv	34.1	70.1	29.2	0.80	24.2	48.0	17

Table 6.3: SMD models validation set scores. **bold** is the overall best reported score.



(a) Distant boats horizon

(b) Close medium and small boats

(c) Tank vessels close

(d) Twilight conditions

Figure 6.4: Sample predictions D3-lr$_2$ on SMD. Images cropped for visibility.

Best model

As such, the final selected model providing targeted detection pre-trained weights for the SMD and which is further evaluated on the target domain test set in section 7.1, is D3-lr$_2$, later renamed to D3-SMD in section 7.1.

Sample predictions

A few sample predictions of the best model, D3-lr$_2$, are presented in figure 6.4. D3-lr$_2$ seems to generalize well to many of the encountered situations in the SMD. All of the various distant boats are well-detected in figure 6.4a, while the same is observed for the closer boats in figure 6.4b. In figure 6.4c many of the large occluded tankers are accurately detected in addition to the un-occluded small boat and tanker in the left part of the scene. The last sample from twilight conditions in figure 6.4d, proves that in more challenging conditions D3-lr$_2$ generates some FN predictions for distant and partially occluded samples. Overall, D3-lr$_2$ performs rather well in the presented situations.

6.2.2 Nordic maritime dataset

This section treats all the trained models on NMD, presented in section 5.2. Hyperparameter selection is presented in a similar manner of the preceding section, separating between dataset and model-specific hyperparameters.

Dataset-specific hyperparameters

The first hyperparameter to adjust, is the RGB mean and standard deviation. Table 6.4 presents the calculated values for the NMD. It is observed that all the colour channel means, but particularly the green and blue, are significantly lower than for the SMD (see table 6.2). Therefore, in a general sense the NMD contains darker images than the SMD.

NMD RGB mean and std	
Mean	**Std**
[119.499, 120.302, 124.519]	[43.023, 40.710, 40.181]

Table 6.4: NMD RGB mean and standard deviation.

The bounding box aspect ratio and area plots are for the training and validation set of the NMD are presented in figure 6.5.

The bounding box aspect ratios in figure 6.5a are centered around the “1-2” column. As the large majority of the bounding boxes are inside the default anchor boxes of 0.5, 1.0 and 2.0, the default anchor box aspect ratios are considered suitable for the NMD. Moreover, the validation and training set ratio seems consistent over different aspect ratio ranges. The same is observed for the bounding box areas.

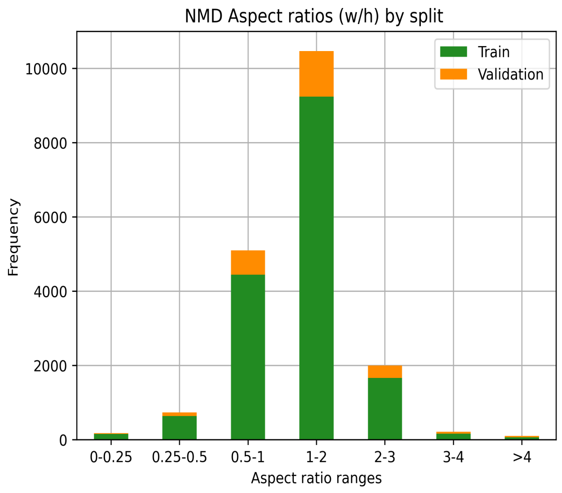
The bounding box area distribution presented in figure 6.5b, displays a very right-skewed bounding box area distribution. The right-most accumulated samples with an area over 18,000 pixels incorporates only a fraction of the total samples, dissimilar to the SMD. From the COCO area classifications (see table 5.2) only 10.7% of the bounding boxes are classified as large, while 48.6% of the bounding boxes are small. We argue that these two observations motivate a potential decrease in the anchor box scale.

The default anchor box scale is 4.0, currently limiting the smallest detectable object size to 32×32, which is the limit for satisfactory detecting COCO classified small objects. With this scale, a large part of the smallest objects are impossible to detect. By decreasing the anchor box scale to 3.0, the resulting smallest detectable object is 24×24. As follows, only approximately 0.2% of the largest objects of the NMD would be too large for detection, while capturing 15.4% more of the smallest instances. A further decrease would compromise the detection of large objects and is not pursued. The anchor scale calculations are performed by calculating the anchor scales on the EfficientNet-B3 backbone and the corresponding pyramid levels for each anchor scale value.

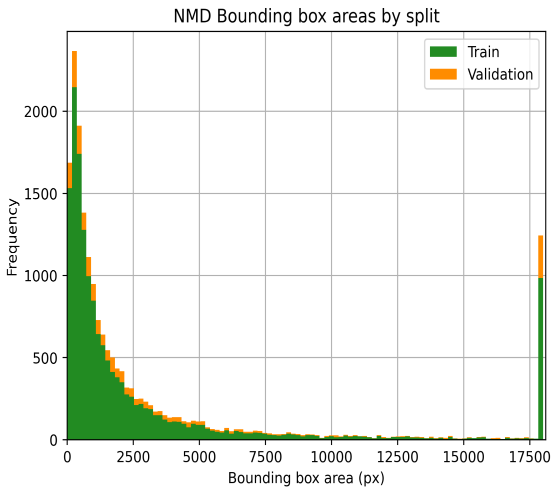
Model-specific hyperparameters

In tuning the learning rate, the same two learning rates tested in section 6.2.1 are adopted.

The NMD is a challenging dataset compared to the SMD. The Hurtigruten dataset, treated in section 4.3, composes most of the samples in the dataset, representing 27 different instance classes captured in various lighting and background conditions before agnostic relabeling. The Hurtigruten$_{BL}$ samples, additionally add diverse samples captured closer to the ocean surface and in harbours. It is therefore not unlikely that a higher learning rate and more training epochs are necessary for generalizing properly to the dataset.



(a) Aspect ratio barplot distribution by split



(b) Area histogram by split

Figure 6.5: NMD bounding box aspect ratios and areas by split, after adjusting for bilinear interpolation image resizing to 896 × 896. All areas larger than 18,000 are accumulated in the rightmost column in figure 6.5b. *Green* is training set samples, *orange* is validation set samples.

Model runs

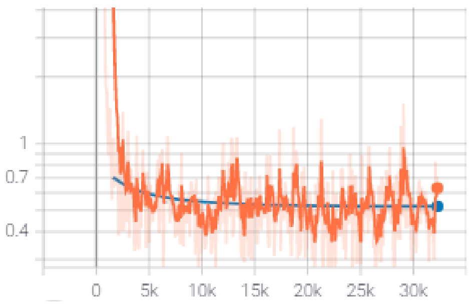
Similar to section 6.2.1, we make successive runs for deciding on the best combination of hyperparameters. The RGB mean and standard deviation are adjusted to the values of table 6.4 and all models are initially trained for 20 epochs. All other baseline settings are utilized as specified in section 5.4.

To firstly pinpoint a suitable learning rate, the following two models are considered:

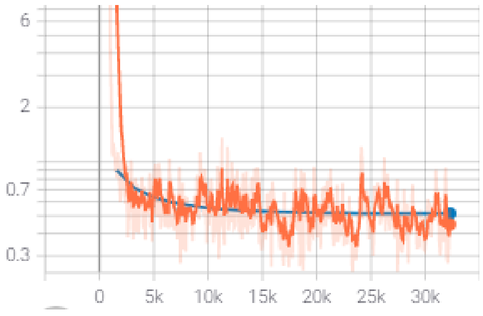
1) D3-lr$_1$: Baseline settings with learning rate 1e-3.
2) D3-lr$_2$: Baseline settings with learning rate 5e-4 and warm up learning rate of 1e-4.

The respective training and validation loss curves for the two 20 epoch training runs are presented in figure 6.6. D3-lr$_1$ is saved at epoch 20 at 32,280 training iterations, while D3-lr$_2$ is saved on epoch 15 at 24,210 training iterations due to a stagnating validation loss, which is not easily observable from figure 6.6b. D3-lr$_1$ trains significantly faster than D3-lr$_2$, unsurprisingly considering the higher learning rate. Overall, D3-lr$_1$ utilizes approximately 12 hours for training 20 epochs, while D3-lr$_2$ requires close to 17 hours.

Based on the presented validation set scores in table 6.5, it is observed that D3-lr$_1$ outperforms D3-lr$_2$ on all reported performance metrics. It is not unlikely that D3-lr$_2$ got stuck in a local minima, causing the somewhat worse validation set scores.



(a) Loss curves D3-lr$_1$



(b) Loss curves D3-lr$_2$

Figure 6.6: Loss curves D3-lr$_1$ and D3-lr$_2$ on NMD. *Orange* is training loss, *blue* is validation loss. 0.6 TensorBoard smoothing, logarithmic y-axis scale for visibility of training loss.

Following the experiment methodology, D3-lr$_1$ is adjusted with the proposed decreased anchor scale of 3.0 to create a new model. This model is denoted as D3-A$_{s=3.0}$.

3) D3-A$_{s=3.0}$: D3-lr$_1$ hyperparameters with anchor box scale of 3.0.

Since the loss curves of D3-A$_{s=3.0}$ resembles the loss-curves of D3-lr$_1$, these are omitted. D3-A$_{s=3.0}$'s evaluated validation set scores are presented in table 6.5. Overall, the AP metric has a slight improvement of 0.2 percentage points from D3-lr$_1$. It seems that this improved performance is mostly explained from lower IOU thresholds, as the AP_{50} improves significantly more than AP_{75}. In addition, the AP_s clearly benefits from the decreased anchor scale of D3-A$_{s=3.0}$, improving with 2.5 percentage points. However, it is also observed that the improved AP_s is trading-off of a decreased AP_l of 1.4 percentage points.

	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	Epoch
D3-lr$_1$	47.4	83.9	46.9	23.5	52.2	**70.2**	20
D3-lr$_2$	46.1	81.6	44.9	21.6	51.8	68.6	15
D3-A$_{s=3.0}$	**47.6**	**85.3**	**47.2**	**25.9**	**52.3**	68.8	20

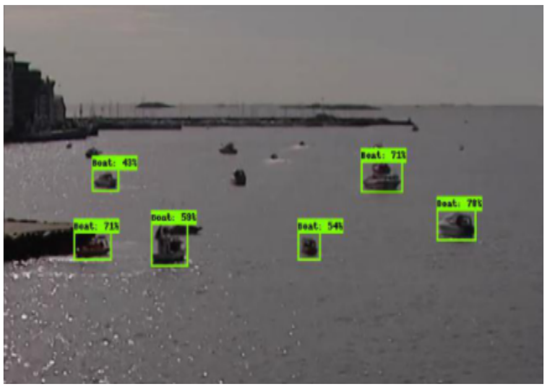
Table 6.5: NMD models validation set scores. **bold** is the overall best reported score.

Best model

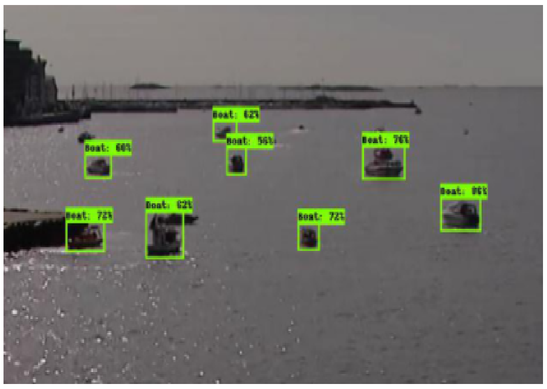
D3-A$_{s=3.0}$ obtains the highest AP validation set score and is therefore selected as the model for obtaining the targeted detection pre-trained weights for the NMD. In section 7.1, the same model is evaluated on the target domain test set under the name D3-NMD.

Sample predictions

To further assess the trained models' performance on the NMD, some sample predictions are visualized. Firstly, it is of interest to visualize a prediction with smaller objects in the scene, as to observe the effect of the changed anchor box scale of D3-A$_{s=3.0}$, compared to D3-lr$_1$. Figure 6.7 presents such a sample for Ep16 from the Hurtigruten dataset. It can be observed that only D3-A$_{s=3.0}$ manages to make any predictions for the smaller background boats, though still missing the very smallest ones.



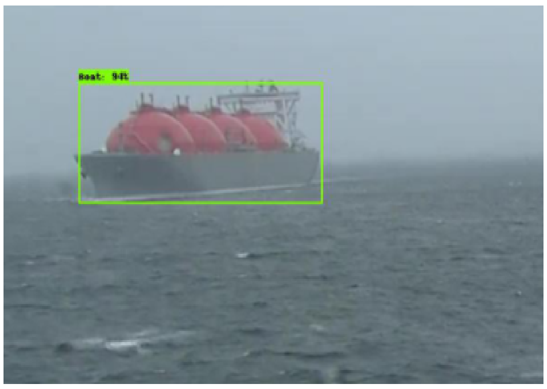
(a) Sample prediction D3-lr$_1$



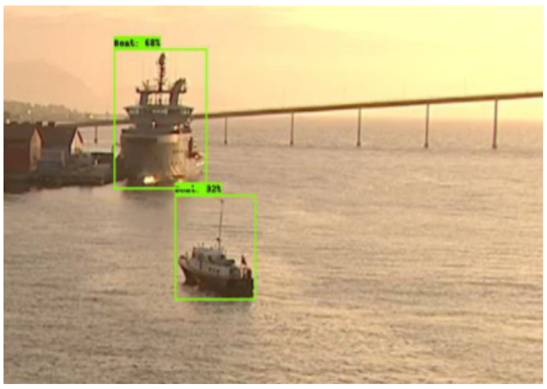
(b) Sample prediction D3-A$_{s=3.0}$

Figure 6.7: Sample predictions from Ep16 from Hurtigruten dataset D3-lr$_1$ and D3-A$_{s=3.0}$.

In addition, figure 6.8 presents some predictions for the best model, D3-A$_{s=3.0}$, in different weather conditions. D3-A$_{s=3.0}$ predicts all objects satisfactory and showcases the detector's robustness for different weather conditions. In addition, D3-A$_{s=3.0}$ is also capable of enclosing the complete object contours, including antennas, as visualized in figure 6.8b.



(a) Rainy conditions Ep29

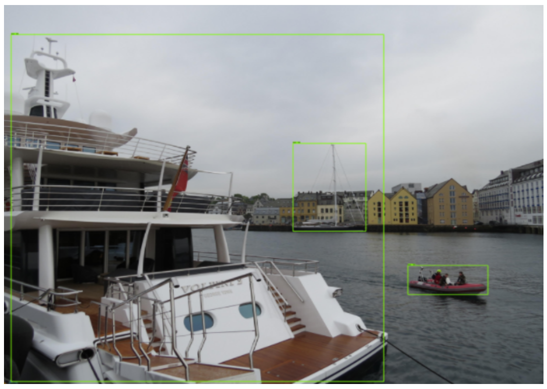


(b) Sunny conditions Ep19

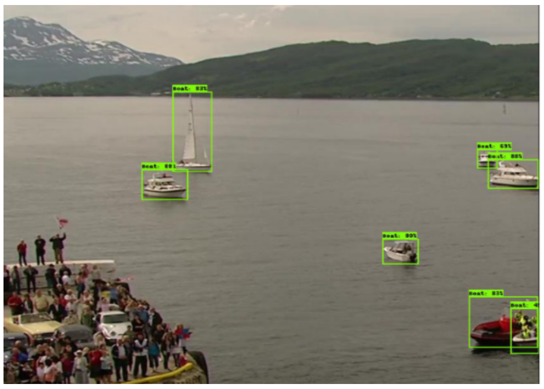
Figure 6.8: Sample predictions D3-A$_{s=3.0}$ from Ep29 and Ep19 Hurtigruten dataset

Additionally, some examples from more harbour-like conditions are presented. Figure 6.9a presents a sample image from Hurtigruten$_{BL}$. The detection setting demonstrates that D3-A$_{s=3.0}$ has no problems detecting large scale foreground objects simultaneously with smaller background objects. The large motorboat on the left side of the scene is well-captured, while the background sailboat and motorboat are also tightly captured in the predicted bounding boxes.

In figure 6.9b, D3-A$_{s=3.0}$ detects all the scattered objects in the sample image from Ep22. What is more, the two occluded objects on the down right corner of the scene are separated between as two unique instances. As the reader might remember, this was one of the reasons for designing the tiling procedure of the Hurtigruten dataset (see section 4.3.2), as to include partial objects in each tile. In cases like this, the tiled samples seen during training seem to benefit, as both objects are detected based on a subset of their features.



(a) Hurtigruten$_{BL}$



(b) Harbour condition Ep22

Figure 6.9: Sample predictions D3-A$_{s=3.0}$ from Ep22 and Hurtigruten$_{BL}$ dataset

6.2.3 Mixed maritime dataset

The final unseen target domain experiment is conducted on the MMD dataset, treated in section 5.2. All hyperparameters are presented in a similar manner as in the preceding unseen target experiments.

Dataset-specific hyperparameters

The MMD RGB mean and standard deviation values to adjust are presented in table 6.6.

MMD RGB mean and std	
Mean	**Std**
[129.735, 141.977, 156.679]	[43.859, 38.916, 40.808]

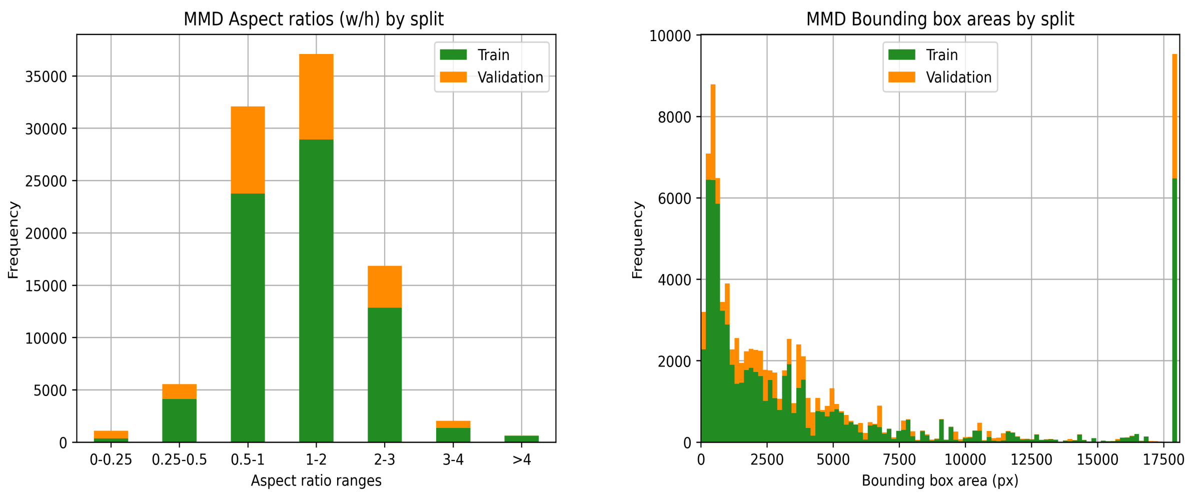
Table 6.6: MMD RGB mean and standard deviation.

Figure 6.10 presents the bounding box aspect ratio and area plots of the training and validation set of MMD. For both of the plots, the training and validation set seems to be adequately distributed for different aspect ratio and area ranges. Since the MMD inherits dataset properties of its two sub datasets, the SMD and NMD, some of the similar dataset-specific hyperparameter observations as in section 6.2.1 and section 6.2.2 are highlighted here.

The bounding box aspect ratios presented in figure 6.10a, have the largest count for the "0.5-1" and "1-2" columns. For the unseen target domain experiments on the SMD in section 6.2.1, it was unsuccessfully attempted to increase the highest anchor box aspect ratio due to the high count of samples in the "2-3" column. The same experiment is therefore not pursued here, as additionally there are even more samples in the lower aspect ratio columns. The default anchor box aspect ratios, are considered well-suited for the MMD.

The bounding box area distribution in figure 6.10b presents a more right-skewed distribution, similar to NMD. However, due to the additional samples from the SMD, there are almost 10,000 stacked samples with an area equal to or greater than 18,000 pixels. As such, a decrease in anchor box scale, similar to the best NMD model in section 6.2.2, might more strongly affect the detection performance on larger sized objects.

By decreasing the anchor box scale from the default value of 4.0 to 3.0, it is quantitatively observed that a very small percentage of large bounding boxes are excluded (0.5%) while including 11.4% more of the smallest bounding boxes in the dataset.



(a) Aspect ratio bar plot distribution by split

(b) Area histogram by split

Figure 6.10: MMD bounding box aspect ratios and areas by split, after adjusting for bilinear interpolation image resizing to 896 × 896. All areas larger than 18,000 are accumulated in the rightmost column in figure 6.10b. *Green* is training set samples, *orange* is validation set samples.

Model-specific hyperparameters

The same two learning rates as tested in section 6.2.1 and section 6.2.2 are adopted.

The MMD consists of 13,929 mixed training images from the NMD and the SMD (see table 5.1). Therefore, it is firstly expected that training might be rather time-consuming. Secondly, as the MMD is arguably more diverse and each batch of images might contain completely different boat instances and scenes, convergence might prove slow. As such, the highest of the two proposed learning rates might be more beneficial for finishing training in reasonable time with sufficiently fast convergence on the single GPU resource available.

Model runs

All models are adjusted for the RGB mean and standard deviation values presented in table 6.6 and attempted to train for 20 epochs with default baseline hyperparameters.

The two following models are initially trained to find a satisfactory learning rate.

1) D3-lr$_1$: Baseline settings with default learning rate 1e-3.
2) D3-lr$_2$: Baseline settings with learning rate 5e-4.

Due to an issue encountered during training, the training runs are not successfully finished for both models and the loss curves are therefore omitted. In essence, the Google Colaboratory provided GPU resource struggles to finish the training runs in a reasonable time even with the time-out limit of 24 hours. It is therefore decided to do an intermediate assessment of the validation and loss curves due to time-constraints. At this inspection, D3-lr$_2$ is the only model with smoothly converging loss curves and a gradually smooth increase in the validation AP curve. As follows, D3-lr$_2$ is the only model finished for training. The model weights are saved from epoch 20, at training iteration 69,640. The final validation set score of D3-lr$_2$ is presented in table 6.7.

	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	Epoch
D3-lr$_2$	**42.5**	78.7	**41.0**	8.7	33.9	**58.6**	20
D3-A$_{s=3.0}$	42.4	**80.4**	39.5	**9.7**	**34.7**	57.5	12

Table 6.7: MMD models validation set scores. **bold** is the overall best reported score.

The succeeding model, adjusting the anchor box scale of D3-lr$_2$, is thereafter similarly trained for 20 epochs. We refer to this model as D3-A$_{s=3.0}$.

3) D3-A$_{s=3.0}$: D3-lr$_2$ hyperparameters with anchor box scale of 3.0.

The loss curves of D3-A$_{s=3.0}$ are displayed in figure 6.11. The training finished at epoch 20 after approximately 29 hours training, with the best model saved at epoch 12 for 41,784 training iterations. Even though it is not very clear from the validation loss, it did indeed start stagnating at around 40,000 iterations.

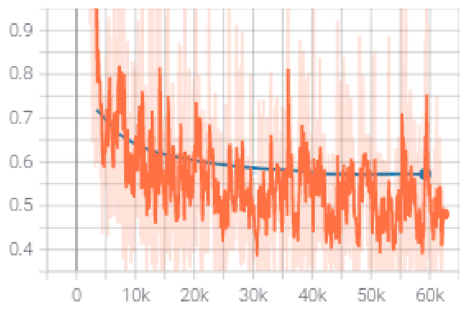


Figure 6.11: D3-A$_{s=3.0}$ loss curves. *Orange* is training loss, *blue* is validation loss. 0.6 TensorBoard smoothing.

Best model

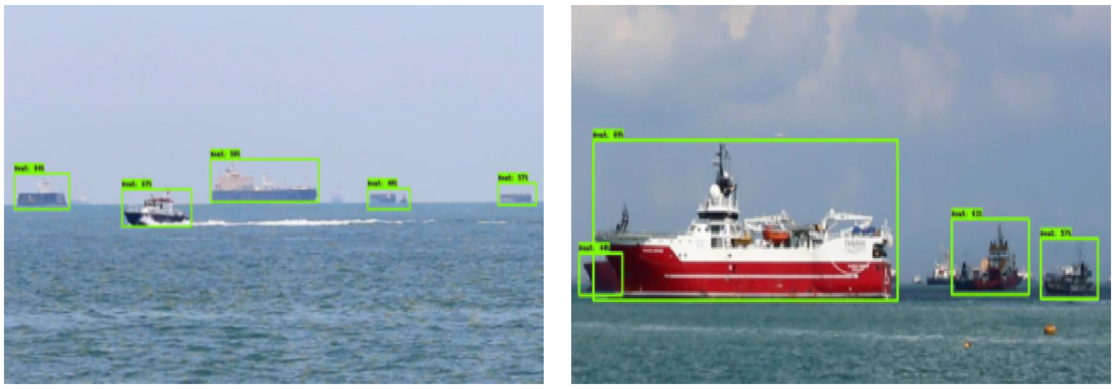
Overall, as presented in table 6.7, D3-A$_{s=3.0}$ is not necessarily presenting an improvement of the reported performance metrics scores of D3-lr$_2$. The AP metric is on the contrary inferior to the baseline AP of D3-lr$_2$. This is mostly a consequence of the decreased performance on higher IOU thresholds (AP_{75}) and for larger objects (AP_l). Intuitively, D3-lr$_2$ is the natural choice of final model trained on the MMD.

Nevertheless, since both of the models are rather similar in reported performance, two more considerations are taken into account. Namely, the fact that both models perform quite bad on AP_s and that a majority of the ground truth samples (belonging to the SMD) are inherently imprecise (see section 4.4.1). Due to the ground truth impreciseness, it is considered more important to weight AP_{50} than AP_{75} as representative for model performance. Since AP only covers thresholds from 0.5 and upwards, an improvement on AP_{50}, as displayed by D3-A$_{s=3.0}$, is considered more important than a higher AP_{75}. As for the small objects, due to the general low scores on this object size, we argue an increase in AP_s is important on the cost of trading-off a decrease in the higher reported AP_l scores.

Therefore, the final model selected for providing the targeted detection pre-trained weights and further evaluation on the target domain test set in section 7.1, is D3-A$_{s=3.0}$.

Sample predictions

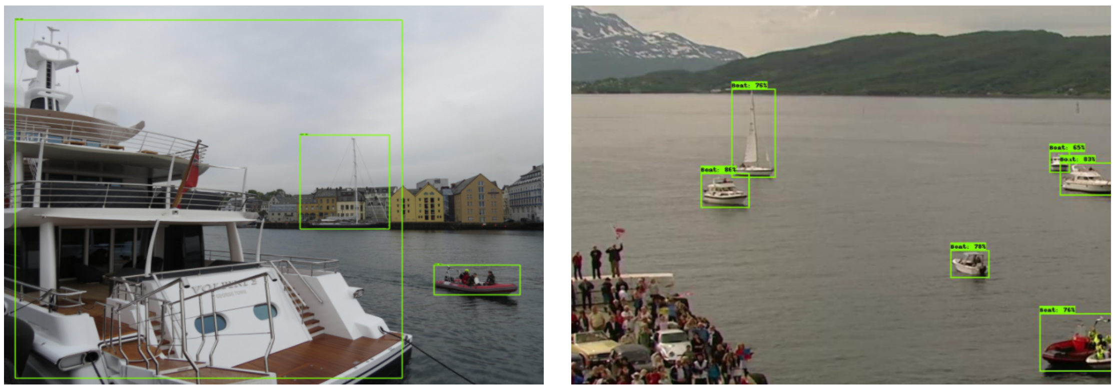
To assess how D3-A$_{s=3.0}$ performs on each of the underlying source domains, a few sample predictions from the SMD and NMD are presented. Figure 6.12 presents two of the sample images tested for predictions in figure 6.4 for the best unseen target domain model trained on the SMD. D3-A$_{s=3.0}$ produces very similar predictions. However, for figure 6.12b, D3-A$_{s=3.0}$ manages to also predict the left-most occluded boat hull, previously failed to detect in figure 6.4b.



(a) Distant boats horizon (b) Close medium and small boats

Figure 6.12: Sample predictions D3-A$_{s=3.0}$ on SMD samples. Images cropped for visibility.

The same sample images from the NMD presented in figure 6.9, are also tested for prediction of D3-A$_{s=3.0}$ in figure 6.13. Overall, D3-A$_{s=3.0}$ predicts almost all of the objects, even though the predicted bounding box of the large boat in figure 6.13a has a larger offset than observed in figure 6.9a. Moreover, D3-A$_{s=3.0}$ does not manage to separate between the two small and highly occluded instances in the bottom right corner of figure 6.13b, as previously observed in figure 6.9b. Based on the presented sample predictions, D3-A$_{s=3.0}$ generalizes quite well to samples from both the NMD and SMD. However, there are no indications that training on the MMD consistently improves predictions in the two underlying source domains.



(a) Hurtigruten$_{BL}$ (b) Harbour condition Ep22

Figure 6.13: Sample predictions D3-A$_{s=3.0}$ from Ep22 and Hurtigruten$_{BL}$ dataset.

Target domain RGB mean and std	
Mean	**Std**
[114.766, 129.052, 148.058]	[53.783, 56.067, 58.949]

Table 6.8: Target domain dataset RGB mean and standard deviation.

6.3 Seen target domain

This section covers all seen target domain experiments, which fine-tune into the target domain. The experiments are designed as to first establish baseline models and suitable hyperparameters for each of the fine-tuning settings (HO, FB, FF).

Further on, models initialized from the targeted detection pre-trained weights obtained in section 6.2, are fine-tuned for the same established baseline hyperparameters. Reusing the same hyperparameters for all seen target domain experiments is considered most fair for benchmarking the baselines and the targeted detection pre-trained models.

1) Section 6.3.1 covers baseline experiments conducted on the target domain dataset from initialized COCO pre-trained weights. All final models presented in this section represent the seen target domain model baselines and respetive established hyperparameters for each fine-tuning setting.
2) Section 6.3.2 covers all experiments utilizing targeted detection pre-trained weights obtained in section 6.2.

All hyperparameter considerations are presented in the same manner of section 6.2, separating between dataset-specific and model-specific hyperparameters. The models are evaluated on the target domain validation set with presented validation losses for a subset of the trained models. Sample predictions are presented on the target domain test set uniquely, covered in section 7.2. Lastly, all model names presented in this section have a global scope and are reused for reported target domain test scores section 7.2.

6.3.1 Baselines

This section establishes the baseline models for all fine-tuning settings (HO, FB, FF), enabling fair inter-model comparison in section 7.2. In addition to present hyperparameter considerations in-detail, the AutoAugment [24] data augmentation scheme is also tested to address the small target domain dataset size.

Dataset-specific hyperparameters

The RGB mean and standard deviation for the target domain dataset are presented in table 6.8. The standard deviation values are observed to be between 10-15 pixels higher than of previous presented datasets in section 6.2. Due to the small size of the target domain dataset, there are clearly a larger colour variation between the images.

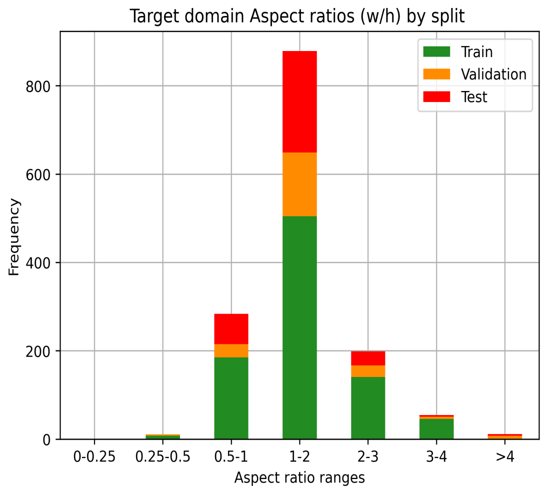
This observation potentially highlights one important characteristic of the target domain dataset. As the colour channel standard deviation is high for the entire dataset, the smaller validation and test set might be representative of quite different colour means than the training set. This is not unlikely and could potentially affect the model selection monitored on the validation set to provide models generalizing worse to the test set.

The bounding box aspect ratios and bounding box areas are presented in figure 6.14. Based on the aspect ratio plot in figure 6.14a, the aspect ratios are evenly distributed around the aspect ratio column “1-2”. The default anchor box aspect ratios of 0.5, 1.0 and 2.0 should therefore sufficiently cover all the bounding boxes in the dataset.

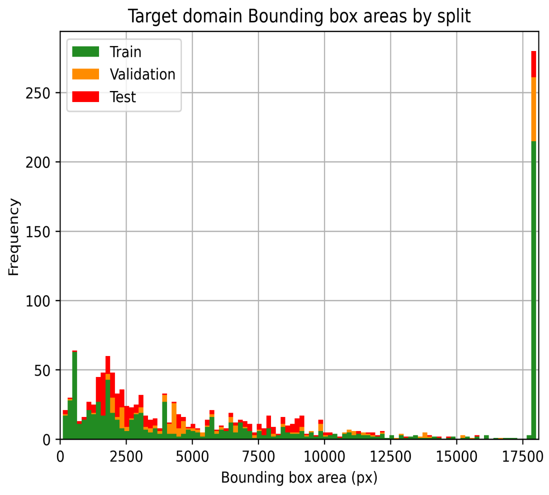
The bounding box area distribution is presented in figure 6.14b. The majority of the target domain instances are rather large, as observed in figure 6.14b from the rightmost column of stacked samples over 18,000 pixels. Only 10.5% of the dataset are small objects (see table 5.2) while 29.7% are large. Consequently, there is little support for decreasing the anchor box scales. Moreover, the target domain bounding box area distribution is quite different from all of the treated datasets in section 6.2.

A last observation from the bounding box area distribution of figure 6.14b, is the bounding box area distribution per split. As presented in the target domain dataset split in table 4.16, the validation and test set should respectively account for 15.1% and 20.2% of the labeled images.

However, there are largely more test set instance samples in the area ranges 0-2,500 and 7,500-10,000, while there are more validation set samples around 5,000 pixels. Additionally, for the stacked column at 18,000 pixels, the validation set portion is more than two times larger than the test set. This observation further gives ground for expecting discrepancies between the reported scores on the validation and test set, particularly for the size dependent AP metrics.



(a) Aspect ratio bar plot distribution by split



(b) Area histogram by split

Figure 6.14: Target domain bounding box aspect ratios and areas by split, after adjusting for bilinear interpolation image resizing to 896 × 896. All areas larger than 18,000 are accumulated in the rightmost column in figure 6.14b. *Green* is training set samples, *orange* is validation set samples, *red* is test set samples.

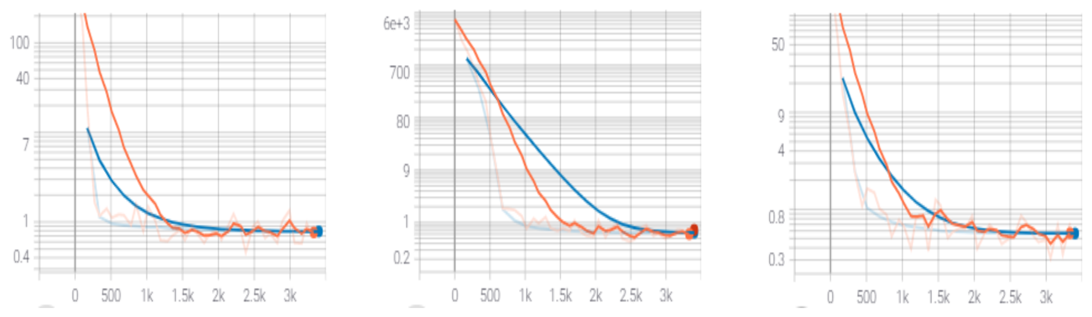
Model-specific hyperparameters

The lower learning rate of 5e-4 from previous experiments is adopted, before attempting adjustments. Considering that the target domain is small and training time is likely quite short, an even lower learning rate is unproblematic to test, if empirically observed to be necessary.

The same learning rate is tested for all fine-tuning settings. As the HO and FB fine-tuning setting update fewer parameters during training, a lower learning rate could be beneficial for smoother convergence. Nevertheless, adjustments to the proposed learning rate is only carried out if observed necessary from the validation and training losses.

Data augmentation

As the target domain dataset solely consists of 1,061 images, it is only fair to consider more aggressive data augmentation schemes in addition to the default horizontal flipping and scale jittering. Additional data augmentation has not been significantly tested in the unseen target



(a) Loss curves D3$_{HO}$-$\mathcal{D}_b$ (b) Loss curves D3$_{FB}$-$\mathcal{D}_b$ (c) Loss curves D3$_{FF}$-$\mathcal{D}_b$

Figure 6.15: Loss curves seen target domain baseline models. *Orange* is training loss, *blue* is validation loss. 0.6 TensorBoard smoothing, logarithmic y-axis scale for visibility of training loss.

domain experiments in section 6.2, as typically more epochs were needed for convergence due to the increased dataset size, and training time was already quite slow for these models.

Instead of applying a variety of different single augmentation transformations, the AutoAugment [24] augmentation scheme treated in section 5.4 is adopted. The V2 augmentation policy, consisting of variety of different geometrical and colour distribution augmentations are chosen.

Model runs

Initially, the default hyperparameters and baseline learning rate of 5e-4 with warm up learning rate 1e-4 are adopted for each fine-tuning setting, before considering further adjustments to the learning rate. No adjustments are carried out for the anchor box parameters and each model is trained for 20 epochs. We refer to each model according to their fine-tuning setting as the first subscript, D3$_{\{\cdot\}}$ (HO, FB, FF), and with the last subscript $\mathcal{D}_b$ indicating the source domain baseline model with COCO pre-trained weights.

- D3$_{HO}$-$\mathcal{D}_b$: Head-only fine-tuning.
- D3$_{FB}$-$\mathcal{D}_b$: Frozen backbone fine-tuning.
- D3$_{FF}$-$\mathcal{D}_b$: Full fine-tuning.

D3$_{HO}$-$\mathcal{D}_b$ and D3$_{FB}$-$\mathcal{D}_b$ are trained for approximately 2.5 hours, while D3$_{FF}$-$\mathcal{D}_b$ is trained for four hours. The training and validation loss of each model is presented in figure 6.15 with logarithmic y-axis scale to more accurately observe the loss curves. All training losses and validation losses decrease smoothly. Even though the validation loss of D3$_{FB}$-$\mathcal{D}_b$ in figure 6.15b possibly indicates that training a few more epochs could be beneficial, this is attempted with no improvements. D3$_{FB}$-$\mathcal{D}_b$ is therefore, similar to the other models, saved at epoch 20.

As to additionally test more data augmentation and preferably only for one model as a starting-point, the fully fine-tuned D3$_{FF}$-$\mathcal{D}_b$ is adapted to include the AutoAugment-V2 augmentation policy. This model is denoted as D3$_{FF}$-$\mathcal{D}_b$-AA

- D3$_{FF}$-$\mathcal{D}_b$-AA: D3$_{FF}$-$\mathcal{D}_b$ hyperparameters with AutoAugment-V2 augmentation policy.

D3$_{FF}$-$\mathcal{D}_b$-AA is trained for 26 epochs before the validation loss starts stagnating at a total training time of 7h30 minutes. The model weights are saved at epoch 23 for the highest evaluated validation AP score.

All evaluated validation set scores are presented in table 6.9. AP_s metric is not reported due to insufficient samples of small objects in the validation set for generating performance scores. D3$_{FF}$-$\mathcal{D}_b$ reports the highest validation scores on all metrics, implying that the target domain dataset is

	AP	AP_{50}	AP_{75}	AP_m	AP_l	Epoch
D3$_{HO}$-$\mathcal{D}_b$	37.4	71.8	31.3	37.5	38.3	20
D3$_{FB}$-$\mathcal{D}_b$	45.7	83.0	45.3	46.7	46.6	20
D3$_{FF}$-$\mathcal{D}_b$	**51.4**	**85.6**	**56.3**	**57.3**	**49.7**	20
D3$_{FF}$-$\mathcal{D}_b$-AA	48.4	84.1	52.0	54.2	46.8	23

Table 6.9: Seen target domain baseline models validation set scores. AP_s not reported due to insufficient samples in validation set. **bold** is the overall best reported score.

large enough for a updating all parameters in a full fine-tuning scheme. The two models with frozen backbones, D3$_{HO}$-$\mathcal{D}_b$ and D3$_{FB}$-$\mathcal{D}_b$ performs significantly worse. Updating more parameters of EfficientDet-D3 is superior to more freezing, also considering that D3$_{FB}$-$\mathcal{D}_b$ outperforms D3$_{HO}$-$\mathcal{D}_b$.

D3$_{FF}$-$\mathcal{D}_b$-AA does not manage to outperform its baseline model D3$_{FF}$-$\mathcal{D}_b$. It is possible that the D3$_{FF}$-$\mathcal{D}_b$-AA could benefit from a different learning rate and by fewer and more tailored augmentations for the target domain. However, data augmentation is not a fool-proof technique for augmenting model performance and can, as observed here, even diminish performance.

In conclusion, all the baseline models D3$_{HO}$-$\mathcal{D}_b$, D3$_{FB}$-$\mathcal{D}_b$, D3$_{FF}$-$\mathcal{D}_b$ are selected for evaluation in the target domain test set in section 7.2.

6.3.2 Targeted detection pre-training

In this section, all the targeted detection pre-training experiments are conducted. EfficientDet-D3 is in separate experiments initialized from the three different pre-trained weights from section 6.2, before fine-tuning into the target domain dataset for all fine-tuning settings (HO, FB, FF), resulting in nine unique models. All hyperparameters from section 6.3.1 are reused for training.

Hyperparameter discussion

As introductory mentioned, all hyperparameters from section 6.3.1 are adopted. However, this decision deserves a short discussion.

In particular, the learning rate found in section 6.3.1, is the only hyperparameter subject to discussion. The pre-trained weights used for initialization are obtained in section 6.2 after fine-tuning on three experimental maritime datasets. As such, it is expected that these pre-trained weights encapsulate some transferable and useful knowledge from each of the source domains. A too high learning rate consequently risks forgetting substantial amount of this knowledge, resulting in catastrophic forgetting, as discussed by Montone et al. [68].

A gradual fine-tuning scheme, not implemented in this thesis, has been found to counter this [68]. However, from the detection pre-training formulation in section 5.1, it is not intended to conserve previously learned knowledge to maintain source domain performance, but rather use it to examine improved target domain performance. In any case, the proposed learning rate from section 6.2 may be sub-optimal. On the other hand, reusing the same learning rate establishes a solid basis for benchmarking of all the seen target domain models. Additional trial and error testing of different learning rates would also be inherently time-consuming for all of the nine models treated in this section.

In conclusion, more fine-tuning of the learning rates in this section is possible and might prove beneficial, but the reuse of the learning rate in section 6.3.1 is considered both meaningful and more suitable for practical reasons.

Model runs

Following the hyperparameter discussion, all models are trained for exactly the same hyperparameters as the baseline models of section 6.2. As it is not possible to exclude slower convergence from the pre-trained weights, models still improving until the last trained epoch (20) are further trained until validation loss stagnation.

Longer trained models have seen more images during training. For fair comparison with the baselines, all models in this section should optimally be trained 20 epochs only, similar to the baselines. However, as the baseline models are trained until validation loss stagnation, following good training conduct, it is considered reasonable to do the same here.

Similar to section 6.3.1, each model is named according to the fine-tuning setting as the first subscript $D3_{\{.\}}$, with the last subscript, $\mathcal{D}_{\{.\}}$, indicating the source domain pre-trained weights. For instance, $D3_{HO}$-$\mathcal{D}_{SMD}$, denotes the model fine-tuned into the target domain with HO fine-tuning setting and pre-trained weights from the SMD, or more specifically, the targeted detection pre-trained weights from the globally named model, D3-SMD, in section 7.1.

In general, all models finish the initial training run of 20 epochs in the range of two-three hours, with the HO and FB fine-tune settings unsurprisingly being the fastest. Loss curves are inspected in detail for model selection and considering necessary extra training epochs. Due to space restrictions in this section, we omit presenting any loss curves. All validation scores and trained epochs for model selection are presented in table 6.10.

Similar to observations from the baseline runs in section 6.3.1, the FF transfer settings arguably report the best validation scores. Even though $D3_{FB}$-$\mathcal{D}_{SMD}$ exceptionally reports the overall highest AP_{75} amongst all models, all other AP metrics are highest reported for one of the FF models.

Furthermore, considering the AP, AP_{50} and AP_{75} metrics, the NMD pre-trained models generally outperform the SMD models. In addition, there are no consistent observations indicating that the pre-trained weights from the larger MMD dataset gives any significant performance gains. The overall best performing model on the validation set is however $D3_{FF}$-$\mathcal{D}_{MMD}$, marginally outperforming $D3_{FF}$-$\mathcal{D}_{NMD}$.

	Model	AP	AP_{50}	AP_{75}	AP_m	AP_l	Epoch
HO	$D3_{HO}$-$\mathcal{D}_{SMD}$	48.4	78.6	55.3	36.5	$\underline{51.9}$	24
	$D3_{HO}$-$\mathcal{D}_{NMD}$	$\underline{51.1}$	$\underline{85.3}$	$\underline{56.1}$	$\underline{57.1}$	49.9	24
	$D3_{HO}$-$\mathcal{D}_{MMD}$	49.4	83.0	51.6	52.6	49.7	20
FB	$D3_{FB}$-$\mathcal{D}_{SMD}$	57.1	86.2	**67.9**	56.5	$\underline{57.9}$	22
	$D3_{FB}$-$\mathcal{D}_{NMD}$	$\underline{57.2}$	$\underline{87.2}$	64.5	$\underline{58.1}$	57.4	19
	$D3_{FB}$-$\mathcal{D}_{MMD}$	56.4	87.0	62.7	56.3	57.5	16
FF	$D3_{FF}$-$\mathcal{D}_{SMD}$	58.0	87.9	66.3	**60.1**	57.3	25
	$D3_{FF}$-$\mathcal{D}_{NMD}$	58.2	87.7	$\underline{66.9}$	59.2	58.2	15
	$D3_{FF}$-$\mathcal{D}_{MMD}$	**58.3**	**88.0**	$\underline{66.9}$	58.4	**58.7**	26

Table 6.10: Seen target domain, targeted detection pre-trained models validation set scores. AP_s not reported due to insufficient samples in validation set. Underlined metrics is the best for the group. **Bold** is the overall best reported score. For tied values, both are highlighted. HO=Head-only, FB=Frozen backbone, FF=Full-fine tuning.

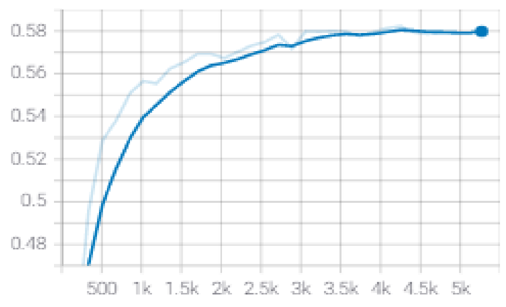
Another observation which gives more insight into the transferability of the features learned during pre-training, is the AP evaluation on the validation set executed during training time. The validation scores reported in table 6.10 present the evaluated validation scores of each model from its saved model weights, with the corresponding epoch. However, the same evaluation is also automatically executed from the model weights temporarily saved every epoch, providing an AP curve with training iterations on the x-axis. To illustrate, the validation AP curves of all the FF models together with the FF baseline model $D3_{FF}$-$\mathcal{D}_b$ from section 6.3.1, are presented in figure 6.16.

One observation to highlight is the difference in convergence speed for the pre-trained models and the baseline model. All the pre-trained models achieve an evaluated AP on the validation set equal to or better than 40 percentage-points after only 500 training iterations, which is approximately three epochs. That is, after seeing the target domain training set only three times[1]. Additionally, all of the pre-trained FF models obtain an AP better than 50 percentage-points in the first 1,000 iterations. In comparison, D3$_{FF}$-$\mathcal{D}_b$ needs over 2,000 iterations to obtain a comparable score, before slowly converging to its final reported validation AP score of 51.4 after 3,400 training iterations, or 20 epochs (see table 6.9).

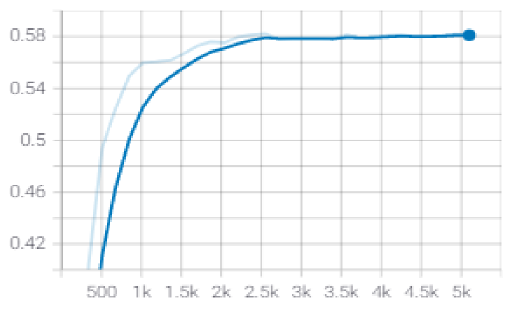
However, there is limited benefit from solely converging faster, considering the short training time on the target domain training set. Thus, another key observation is the fact that the pre-trained models also obtain a much higher validation AP score. As reported, without bells and whistles, all FF pre-trained models manage to surpass 58.0 on the validation AP score. D3$_{FF}$-$\mathcal{D}_b$ on the other hand, has no indications of ever achieving such a high score, considering its slow convergence from the 2,000-3,000 training iterations, observable in figure 6.16d, as well as its validation loss stagnation in figure 6.15c.

What is more, D3$_{FF}$-$\mathcal{D}_{SMD}$, which obtains the highest initial validation AP score, observable from the y-axis in figure 6.16a, reports an astonishing initial AP of 43.3 after one epoch. That is, after one pass over the target domain training dataset.

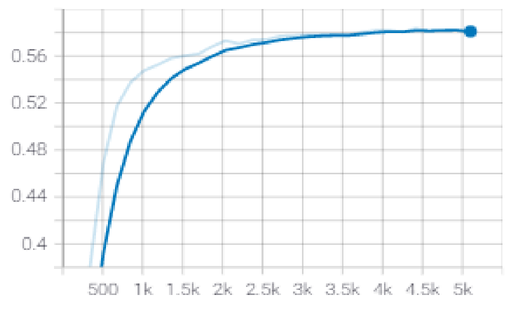
In conclusion, targeted detection pre-training clearly has an effect on convergence speed and initial performance in the target domain, but also the highest achievable validation set score. Further observations regarding the performance on the target domain test set is reported in section 7.2.



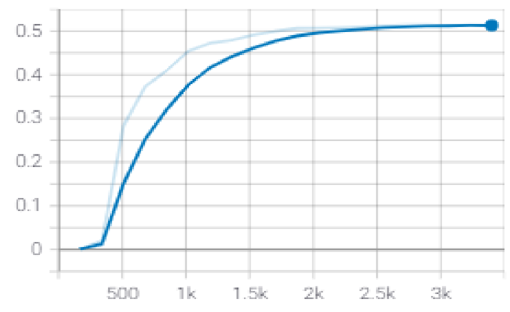
(a) D3$_{FF}$-$\mathcal{D}_{SMD}$



(b) D3$_{FF}$-$\mathcal{D}_{NMD}$



(c) D3$_{FF}$-$\mathcal{D}_{MMD}$



(d) D3$_{FF}$-$\mathcal{D}_b$

Figure 6.16: Target domain validation AP curves; D3$_{FF}$-$\mathcal{D}_{SMD}$, D3$_{FF}$-$\mathcal{D}_{NMD}$, D3$_{FF}$-$\mathcal{D}_{MMD}$ and D3$_{FF}$-$\mathcal{D}_b$. 0.6 TensorBoard smoothing.

[1]One epoch indicates seeing the whole training set one time.

CHAPTER 7 RESULTS

In this chapter, evaluation on the target domain test set is conducted for all selected unseen and seen target domain models in chapter 6. An additional study measuring the impact when introducing class-aware labels in the target domain dataset is presented in section 7.3. In line with our problem formulation, several case-studies are presented in section 7.4, as well as a video inference test scenario in section 7.5. The overall best model is measured from the AP metric.

7.1 Unseen target domain

Model	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	Epoch
D3-SMD	25.7	61.3	16.4	**20.0**	27.5	26.3	7
D3-NMD	**35.2**	**77.7**	23.4	10.0	29.6	**42.5**	20
D3-MMD	33.0	74.8	**23.5**	10.0	**31.2**	36.7	12

Table 7.1: Target domain test set scores, unseen target domain models. **bold** is the overall best reported score. *Epoch* is the trained epochs used for model weight saving.

Table 7.1 presents the target domain test set scores for the unseen target domain models, also referred to as the *unseen target models*. The unseen target models are discussed stand-alone, leaving a comparison with the seen target models to section 7.2.

7.1.1 Main results

- *The source domains contain useful knowledge transferable into the target domain.*

Even though the reported scores of the unseen target models vary, there is no doubt that all the source domains contain useful information transferable into the target domain. The overall best model, D3-NMD, reports AP of 35.2, AP_{50} of 77.7 and AP_{75} of 23.4, on inference in an entirely unseen target domain.

As to give a sense of what this implies, the reader should consider that EfficientDet-D3 reports AP of 47.2, AP_{50} of 65.9 and AP_{75} of 51.2, after training and evaluation on COCO test-dev [101]. Certainly, these datasets are not comparable, however if considering the circumstances, the reported test scores of D3-NMD are indeed quite good.

The largest performance decline is caused by penalized fine-localization of objects, represented by AP_{75}. It is also for this metric that there is the largest difference between the D3-NMD score (23.4) and the EfficientDet-D3 COCO trained model (51.2). [101]. This observation does make sense, taking into consideration that the class and box prediction heads of the unseen target models are not tuned for the target domain.

- *Features learned from the NMD transfer best to the target domain.*

It is observed that D3-NMD is the overall best model, followed by D3-MMD and lastly D3-SMD. D3-MMD is trained on the combination of the SMD and NMD datasets, which is by far the largest dataset in terms of labeled images and instances. D3-NMD is trained on 45.7% of the dataset size of D3-MMD (inferred from table 5.1) and nevertheless outperforms D3-MMD significantly for AP and AP_{75}.

Thus, for the dataset sizes treated in this thesis, a larger size and variety of images seen during training do not indicate to improve feature transfer. However, NMD samples seem to consistently improve target domain performance. This observation is firstly inferred from the fact than D3-NMD standalone is the best model. Secondly, combining the worst performing source domain dataset, SMD, together with NMD, delivers a better model than not doing so (D3-MMD > D3-SMD). As follows, NMD appears to be the most suitable source domain for DA into the target domain.

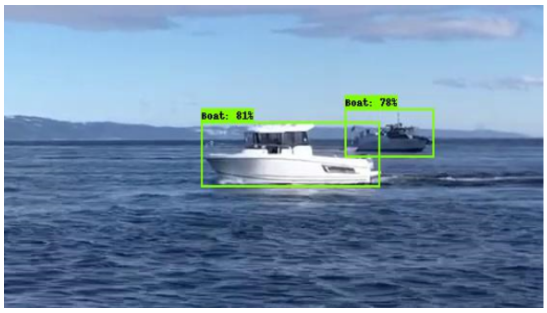
- *Object sizes are not linked to transferability and target domain performance benefits*

The object size ratio distribution of the source domains and the target domain are inherently different, as presented in section 5.2 and from the area histograms presented in section 6.2 & section 6.3. Generally, the target domain contains very few small objects (10.5%, see table 5.2), and is heavily weighted by medium sized objects (60.2%), additionally with the highest weight of largest objects (29.7%) of all treated experimental datasets.

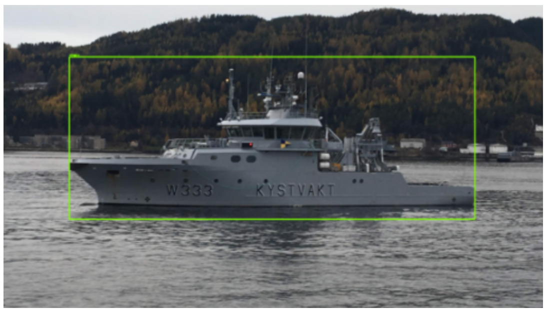
The best unseen target model is D3-NMD, which contrarily has the highest weight of small objects (48.6%, see table 5.2). Additionally, D3-NMD also reports the highest AP_l target domain test set score, even though containing the lowest weight of large objects (10.7%) of all experimental datasets. As such, there are no consistent indications that object size ratios in the experimental training datasets are linked to target domain performance or size-dependent detection.

Sample predictions As to support the initial claim that the unseen target models perform well in the target domain, a few sample predictions for the reported best model, D3-NMD, is visualized in figure 7.1. The ground truths are omitted as there are very few foreground objects and the predictions are inherently accurate.

Both in the multi-object scenario of figure 7.1a, and the single-object case of figure 7.1b, D3-NMD accurately predicts all of the foreground objects. As such, we claim to have demonstrated one of the initial objectives with the unseen target models; maritime source domains are inherently useful for transferring useful knowledge into an unseen maritime target domain.



(a) Open-sea multiple objects



(b) Fjord single object

Figure 7.1: Sample predictions unseen target domain; D3-NMD

7.2 Seen target domain

In this section, baseline models and targeted detection pre-trained models from section 6.3 are evaluated on the target domain test set, as presented in table 7.2. Additionally, AR is reported as to give more insight into the models' production of FN predictions.

	Model	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l	AR	Epoch
HO	D3$_{HO}$-$\mathcal{D}_b$	31.7	65.6	23.7	**40.0**	35.2	31.2	60.0	20
	D3$_{HO}$-$\mathcal{D}_{SMD}$	43.7(+12.0)	79.7	37.7	20.0	40.0	49.9	62.8	24
	D3$_{HO}$-$\mathcal{D}_{NMD}$	42.5(+10.8)	82.7	36.3	20.0	37.8	49.8	62.5	24
	D3$_{HO}$-$\mathcal{D}_{MMD}$	36.4(+4.7)	77.9	28.1	10.0	32.2	43.1	61.9	20
FB	D3$_{FB}$-$\mathcal{D}_b$	32.1	65.5	25.2	**40.0**	34.8	32.5	59.9	20
	D3$_{FB}$-$\mathcal{D}_{SMD}$	45.4(+13.3)	81.7	41.3	20.0	**42.4**	50.2	63.3	22
	D3$_{FB}$-$\mathcal{D}_{NMD}$	47.2(+15.1)	**85.9**	42.9	20.0	40.1	56.8	63.9	19
	D3$_{FB}$-$\mathcal{D}_{MMD}$	45.8(+13.7)	83.6	42.6	20.0	40.2	53.3	65.2	16
FF	D3$_{FF}$-$\mathcal{D}_b$	42.9	77.8	39.5	02.2	37.8	50.3	61.8	20
	D3$_{FF}$-$\mathcal{D}_{SMD}$	48.1 (+5.2)	84.8	44.3	20.0	42.1	56.1	64.2	25
	D3$_{FF}$-$\mathcal{D}_{NMD}$	**48.8**(+5.9)	85.4	**45.9**	20.0	40.7	**59.6**	64.5	15
	D3$_{FF}$-$\mathcal{D}_{MMD}$	48.1 (+5.2)	84.9	45.8	05.0	40.5	57.7	**65.5**	26

Table 7.2: Target domain test set scores, seen target domain models. Underlined metrics is the best for the group. **bold** is the overall best reported score. For tied values, both are highlighted. HO=Head-only, FB=Frozen backbone, FF=Full-fine tuning. For the AP metric, all percentage points improvement are compared to the baseline per group. *Epoch* is fine-tuned epochs on the target domain, used for model weight saving.

As a first note, a discrepancy is discovered for the target domain validation and test set. For all seen target domain models, or *seen target models*, there is a large performance difference between reported scores on the validation and test set.

Overall, all seen target models perform considerably better on the validation set than on the test set, which can be observed by comparing table 6.9 and table 6.10 with reported test scores of table 7.2. The discrepancy is in-depth elaborated upon in section 8.1. As of now, we underline the two relevant consequences of this discrepancy for the remainder of this section.

1) The validation and test set undergo a covariate shift, which do not invalidate any reported test scores, but motivates refraining from a thorough comparison of performance between the split sets.
2) The AP_s metric is unreliable for benchmarking purposes.

7.2.1 Main results

- *All baseline models are outperformed by the targeted detection pre-trained models.*

Based on the model results, excluding details from the size dependent AP metrics, there is no baseline model which outperforms its targeted detection pre-trained counterparts. In addition, most of the pre-trained models are trained a comparable amount of epochs (± 5) to the baselines, but still vastly surpass their performance in many cases, particularly for the HO and FB fine-tuning settings. These observations are consistent with reported validation scores from section 6.3.

- *FF is the superior fine-tuning setting, while FB is better than HO.*

Similar as observed from the target domain validation set, the FF fine-tuning setting is the best performing fine-tuning option. Furthermore, FB outperforms HO, both for the baselines and the targeted detection pre-trained models. These observations support experiments on fine-tuning in the literature [23] and indicates that the target domain dataset is sufficiently large for adapting all EfficientDet-D3 parameters and additional freezing of parameters is sub-optimal. As follows, the target domain dataset is likely too large and the source and target domain distance not close enough, for the FB models to further improve the FF models.

- *Pre-training dataset similarity to the target domain is more important than dataset size.*

The NMD pre-trained weights delivers the best FF and FB model, while the best HO model is based on SMD pre-trained weights. The MMD pre-trained weights do not manage to deliver the best model in any of the fine-tuning settings. Hence, similar to observed in section 7.2, it seems that dataset feature similarity is more important than dataset size for successfully transferring knowledge into the target domain. The results from the unseen and seen target models suggest that the NMD dataset is most suitable for DA into the target domain.

- *Targeted detection pre-training delivers better reported target domain performance, even for fewer fine-tuned epochs.*

Based on the validation curves in figure 6.16 and the author's observations from HO and FB validation AP curves, the pre-trained models experience an initial boost in AP followed by a faster convergence. Subject to the fine-tuning setting, the targeted detection pre-trained models also converge to better reported test set scores for similar or slightly more trained epochs compared to the baseline models.

However, several targeted detection pre-trained models also report better test scores for *fewer* trained epochs. The overall best model, D3$_{FF}$-$\mathcal{D}_{NMD}$, improves on the baseline model, D3$_{FF}$-$\mathcal{D}_b$, with +5.9 percentage-points on the AP metric, while training five epochs shorter. Targeted detection pre-training therefore also provide a performance boost after fine-tuning, similar to observed for DCNN fine-tuning [111].

- *Mixed source domain pre-training produces higher target domain recall when adapted properly.*

The overall results indicate that benefiting from mixed source domains, represented by targeted detection pre-training on MMD, also requires updating the feature-fusion (FB) or backbone weights (FF). This is observable from the HO setting, where D3$_{HO}$-$\mathcal{D}_{MMD}$ strongly deteriorates performance compared to the other targeted detection pre-trained HO models.

On the other hand, the largest increase in the AR metric compared to the baseline scores, is reported for the FB and FF models pre-trained on the MMD, D3$_{FB}$-$\mathcal{D}_{MMD}$ and D3$_{FF}$-$\mathcal{D}_{MMD}$, with a + 5.3 and + 3.7 increase compared to D3$_{FB}$-$\mathcal{D}_b$ and D3$_{FF}$-$\mathcal{D}_b$. As such, we observe that pre-training on mixed source domains mitigates FN predictions, on the cost of a lower AP. We argue this observation supports an intuitive hypothesis. Increased dataset diversity expose the models to a larger variety of object instances during pre-training training, delivering models capable of more easily detecting new objects observed during fine-tuning.

- *Detector inference times are in line with the problem formulation in section 1.1.*

The inference time is not reported for all models, due to time-constraints. For the best reported model, D3$_{FF}$-$\mathcal{D}_{NMD}$, the frame-rate is 26 FPS for end-to-end processing. That is, image pre-processing, pure inference time, post-processing and NMS. As such, the frame rate is above the required 25 FPS as discussed in section 1.1. Similar inference times are found for the other trained detectors, with slight differences depending on model weights. We note that this inference time is in line with the TitanV GPU latency reported for EfficientDet-D3 of 27 FPS in the original paper [101]. The allocated GPU resource for our reported inference time is the Nvidia Tesla P100.

7.3 Class-awareness

In this section, an additional study introducing class-aware labels in the target domain is presented. The intention of this study is to observe the effects of targeted detection pre-training when the source and target domain task changes. Specifically, following definition 5.1.1, $\mathcal{T}_{SM} \neq \mathcal{T}_{TM}$, due to a changing source and target label space.

The three classes *ferry*, *kayak* and *motorboat (mb)* are introduced in the target domain dataset. The *class aware seen target models* are trained under exactly similar settings as the agnostic seen target models in section 7.2. Due to time-constraints, training is conducted only for the baselines and the targeted detection pre-training on the consistently best dataset from section 7.2, the NMD.

	Model	AP	AP_{50}	AP_{75}	AP_{ferry}	AP_{kayak}	AP_{mb}	Epoch
HO	D3$_{HO}$-$\mathcal{D}_b$-AW	13.1	29.0	10.5	6.6	**11.6**	20.9	24
	D3$_{HO}$-$\mathcal{D}_{NMD}$-AW	$\underline{16.4}$(+3.3)	$\underline{34.8}$	$\underline{13.1}$	$\underline{14.9}$	3.3	$\underline{31.2}$	22
FB	D3$_{FB}$-$\mathcal{D}_b$-AW	13.1	27.4	8.7	6.8	$\underline{8.3}$	24.2	23
	D3$_{FB}$-$\mathcal{D}_{NMD}$-AW	**19.2**(+6.1)	**38.2**	$\underline{15.3}$	**16.7**	6.7	**34.3**	24
FF	D3$_{FF}$-$\mathcal{D}_b$-AW	13.0	24.6	11.7	6.5	1.7	$\underline{31.0}$	18
	D3$_{FF}$-$\mathcal{D}_{NMD}$-AW	$\underline{16.7}$(+3.7)	$\underline{31.8}$	**15.8**	$\underline{15.3}$	$\underline{4.9}$	29.8	11

Table 7.3: Target domain test set scores, class aware target models. Underlined metrics is the best for the group. **bold** is the overall best reported score. For tied values, both are highlighted. HO=Head-only, FB=Frozen backbone, FF=Full-fine tuning.For the AP metric, all percentage points improvement are compared to the baseline per group. *Epoch* is fine-tuned epochs on the target domain, used for model weight saving.

7.3.1 Main results

- *Target domain class imbalance and changed source-target label space strongly deteriorates performance.*

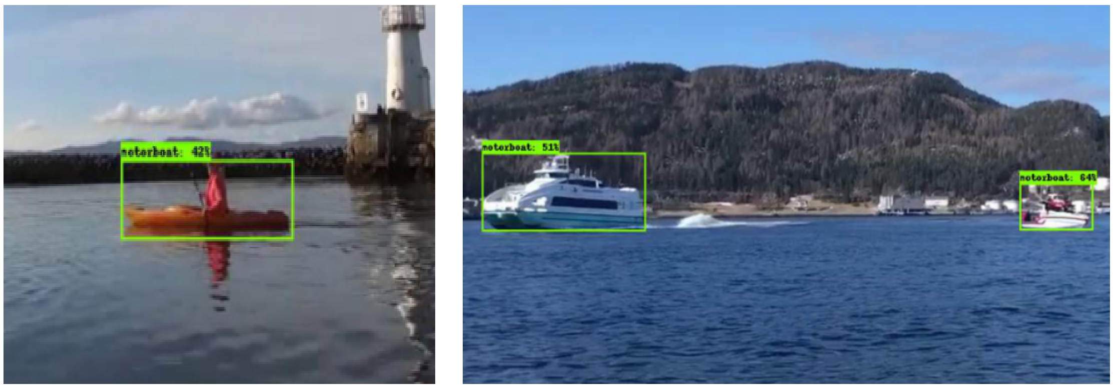
As observed in table 7.3, all the class-aware models performs worse than the class-agnostic seen target models from section 7.2. Firstly, as the source and target domain label space are now different, it is natural to observe worse target domain performance [111]. More importantly, as the target domain test set is inherently class imbalanced, fewer training samples are available for generalizing the detector to each class. As such, an overall decline in reported performance test scores are observed for all models, both baselines and targeted detection pre-trained models. In particular, the AP_{kayak}, which is based on 154 instance labels (see table 4.15), strongly suffers from this fact.

- *Targeted detection pre-training consistently outperforms the baseline models, for similar and less trained epochs.*

Similar to the agnostic seen target models, targeted detection pre-training proves to ameliorate the class aware baseline models. Even though there are some variety as to the class-reported AP metrics, the AP, AP_{50} and AP_{75} are consistently best for the targeted detection pre-trained models for each of the fine-tuning setting.

- *More freezing delivers superior target domain performance when introducing fewer available instances per class.*

The overall best model is D3$_{FB}$-$\mathcal{D}_{NMD}$-AW, which reports the highest AP score. In contrary to the agnostic seen target models, FB is the best fine-tuning setting both for the baselines (tied with HO) and the targeted detection pre-trained models. For the same fine-tuning setting, the largest AP baseline improvement of +6.1 is observed. The FF models seem to require more training instances per class for proper generalization. As follows, freezing the backbone and reusing learned features from NMD is the superior option to obtain the best target domain performance for our target domain dataset, which is not a general result with regard to freezing.



(a) Kayak.

(b) Ferry (left) and motorboat (right).

Figure 7.2: Sample predictions best class-aware seen target model, D3$_{FB}$-$\mathcal{D}_{NMD}$-AW. Figure 7.2a *prediction:* Motorboat, 42% confidence. Figure 7.2b *Predictions*: (left) Motorboat 51% confidence, (right) Motorboat 64% confidence.

Sample predictions A few samples predictions for the best model, D3$_{FB}$-$\mathcal{D}_{NMD}$-AW, are presented as to demonstrate some of the characteristics from the class-aware models. D3$_{FB}$-$\mathcal{D}_{NMD}$-AW localizes the boats satisfactory in both of the sample images, even though struggling to predict the correct classes. All predicted instances in the two samples images are predicted as to belong to the motorboat class. Additionally, the confidence score is rather low, supporting previous arguments regarding the models' decreased robustness.

7.4 Case-studies

The case-studies present some scenarios encountered in the wild and are extracted from the target domain test set, based on frames from 5 FPS scenario videos. We prioritize to present predictions from the best reported seen target model, D3$_{FF}$-$\mathcal{D}_{NMD}$, as well as its baseline, D3$_{FF}$-$\mathcal{D}_b$, to shed light on scenarios where the targeted detection pre-training may benefit particularly. Additionally, as the best reported models are the models subject for deployment in the target domain, D3$_{FF}$-$\mathcal{D}_{NMD}$ needs to be tested for robustness.

The case-studies are additionally designed as to analyse predictions in a spatio-temporal aspect. The target domain dataset is largely based on generating frames from videos in the target domain, as treated in section 4.5. As follows, each scenario in this section is conducted on the consecutive generated frames originating from one of the sampled target domain test set videos.

7.4.1 Occluded boats open-sea

Scenario This scenario is designed as to observe the detectors' capabilities in detecting an object which disappears due to occlusion, before reappearing at a different spatial location. The scenario is presented in figure 7.3. The leftmost gray boat passes behind the closest white boat on the left side, before reappearing on the right side. This scenario isolates the fine-localization capabilities of the detectors, as the two boats are overlapping in some the frames. In total, the scenario is generated from a five second video, where we use three particular frames which demonstrates the occlusion transition.

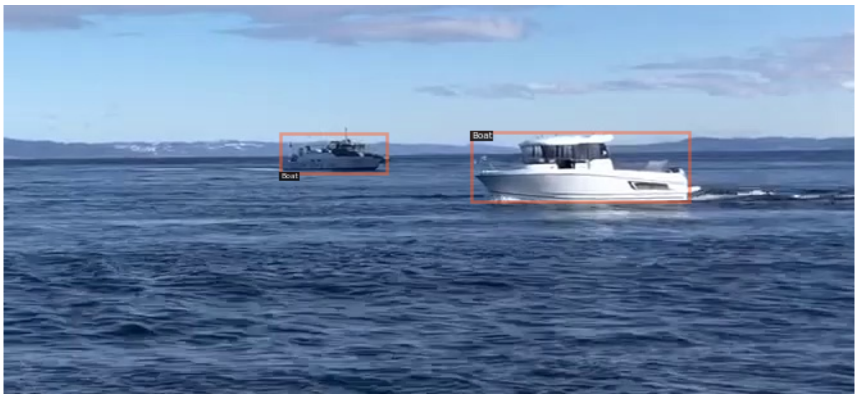
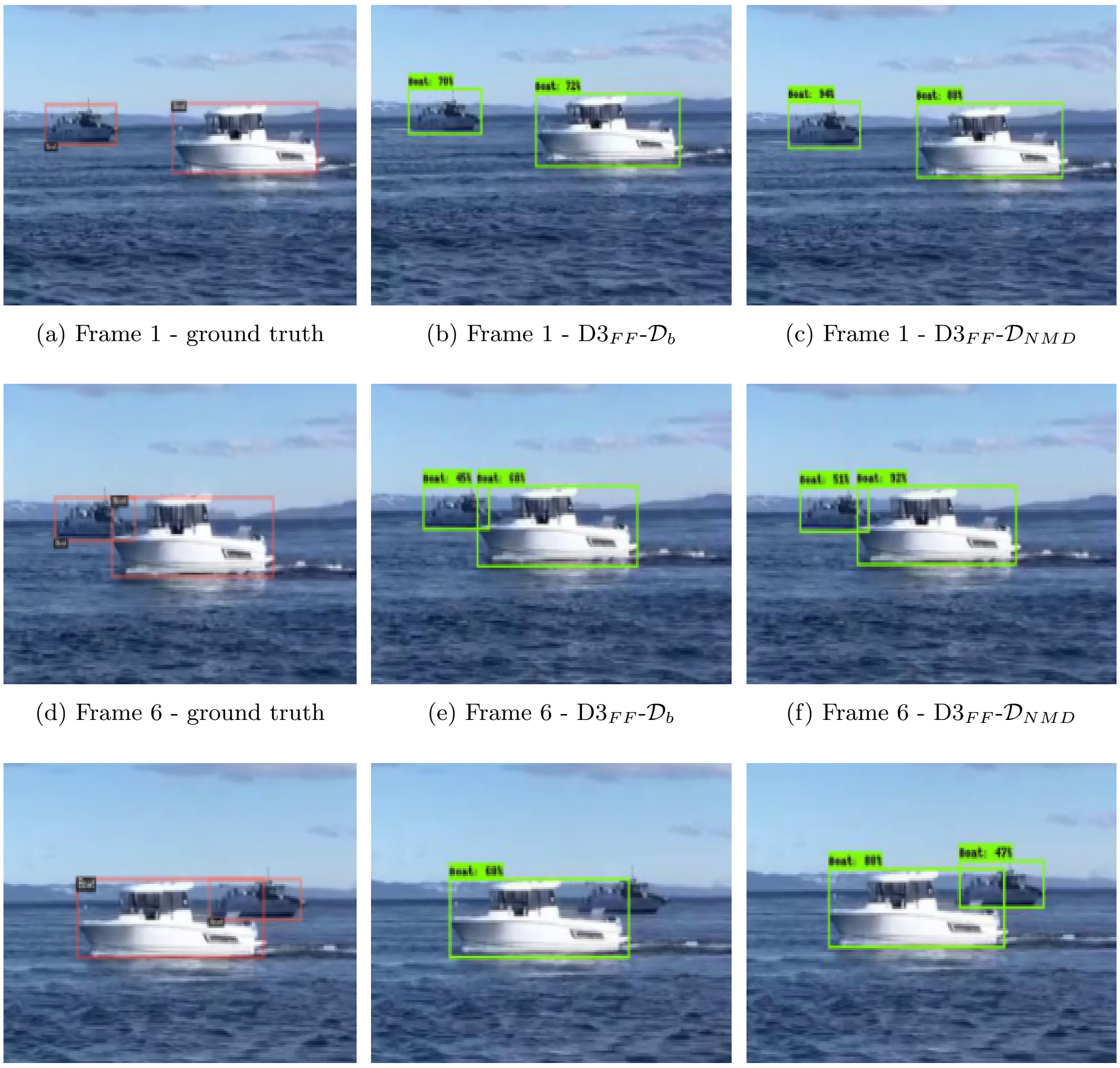


Figure 7.3: Multi-object occlusion scenario, first frame.

Prediction observations Figure 7.4 presents three hand-picked frames from the scenario and the respective ground truths and predictions of D3$_{FF}$-$\mathcal{D}_b$ and D3$_{FF}$-$\mathcal{D}_{NMD}$. The frames are picked as to best illustrate the moment where the gray boat is not too clustered to the white boat, and thus should be detectable.

Both D3$_{FF}$-$\mathcal{D}_b$ and D3$_{FF}$-$\mathcal{D}_{NMD}$ accurately predict the separated boats of the first frame. Similarly, when the gray boat approaches the white boat in the sixth frame, right before becoming entirely occluded, both of the models accurately detect the two boats. However, for the last presented frame, the models differ in their predictions. D3$_{FF}$-$\mathcal{D}_b$ only detects the leftmost white boat, as presented in figure 7.4h, while D3$_{FF}$-$\mathcal{D}_{NMD}$ detects both of the boats, as seen in figure 7.4i.

By testing predictions for slightly lower score confidence thresholds (0.3), it is observed that D3$_{FF}$-$\mathcal{D}_b$ is also capable of detecting the gray boat in the last presented frame. Implicitly, the targeted detection pre-training seems to increase model confidence in its predictions. The same observation, though hard to observe for the reader, is supported by the marginally tighter bounding boxes of D3$_{FF}$-$\mathcal{D}_{NMD}$ compared to D3$_{FF}$-$\mathcal{D}_b$. To conclude the scenario, D3$_{FF}$-$\mathcal{D}_{NMD}$ seems to be more robust model in occlusion scenarios than D3$_{FF}$-$\mathcal{D}_b$.



(a) Frame 1 - ground truth (b) Frame 1 - D3$_{FF}$-$\mathcal{D}_b$ (c) Frame 1 - D3$_{FF}$-$\mathcal{D}_{NMD}$

(d) Frame 6 - ground truth (e) Frame 6 - D3$_{FF}$-$\mathcal{D}_b$ (f) Frame 6 - D3$_{FF}$-$\mathcal{D}_{NMD}$

(g) Frame 17 - ground truth (h) Frame 17 - D3$_{FF}$-$\mathcal{D}_b$ (i) Frame 17 - D3$_{FF}$-$\mathcal{D}_{NMD}$

Figure 7.4: Multi-object occlusion scenario, D3$_{FF}$-$\mathcal{D}_b$ and D3$_{FF}$-$\mathcal{D}_{NMD}$ predictions. Images cropped for visibility.

7.4.2 Ferry front-on

Scenario This scenario presents one of the speed ferries which operates in the harbour orifice towards Trondheimsfjorden. An overview of the scenario is presented in figure 7.5. All frames are generated from a four second video in clear sky and weather conditions. This scenario is considered important as to give a sense of the time for detecting approaching vessels. In order to manoeuvre according to the movements of the ferry, early detection is crucial.

The scenario is considered rather simple as the ferry is relatively large as well as its contrasting contours to the background buildings.

Prediction observations Figure 7.6 presents the ground truth and predictions for D3$_{FF}$-$\mathcal{D}_b$ and D3$_{FF}$-$\mathcal{D}_{NMD}$ for three frames, selected as the start, mid and end of the generated frames from the scenario video. Even though the cropped frames are not entirely aligned in figure 7.6, they are indeed the same frames.

For this scenario, the accuracy of the EfficientDet-D3 detector is demonstrated. Both D3$_{FF}$-$\mathcal{D}_b$ and D3$_{FF}$-$\mathcal{D}_{NMD}$ solely generate TP predictions for all of the sample frames. Actually, this is the

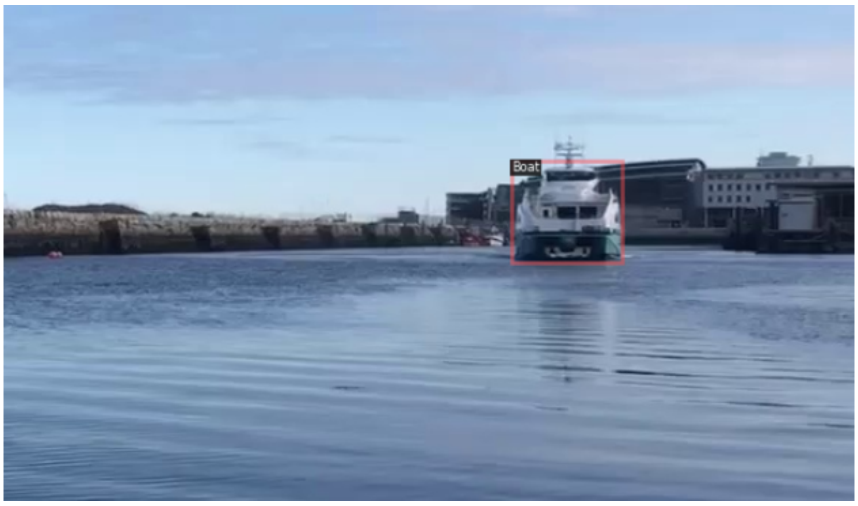
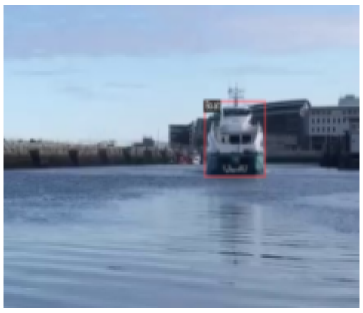


Figure 7.5: Ferry front-on scenario, first frame.

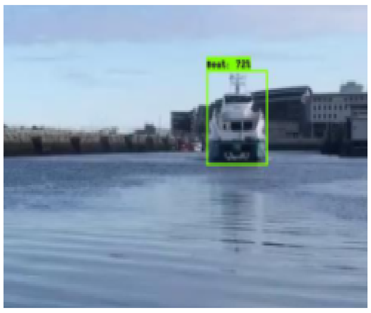
case for all of the total 20 frames for both of the detectors[1]. That is, the trained models accurately detects the ferry without error in all of the 20 frames of the scenario video.

There are few other remarks to make as both detectors perform impeccably. As a last mention, not observable in the prediction samples, D3$_{FF}$-$\mathcal{D}_{NMD}$ generally produces predictions of higher confidence. As an example, the first frame prediction from D3$_{FF}$-$\mathcal{D}_b$ in figure 7.6b has 72% confidence of finding a boat in the predicted bounding box, while D3$_{FF}$-$\mathcal{D}_{NMD}$ presents a confidence of 88% for the same frame, in figure 7.6c. Generally, we find D3$_{FF}$-$\mathcal{D}_{NMD}$ to produce more confident predictions. A consequence of this is also tighter predicted bounding boxes.

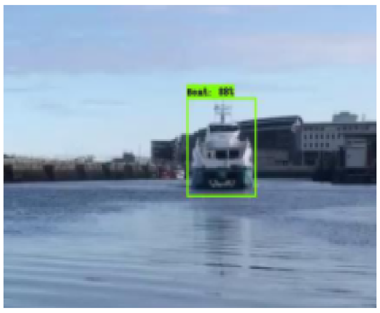
[1]Observed on inspection by the author



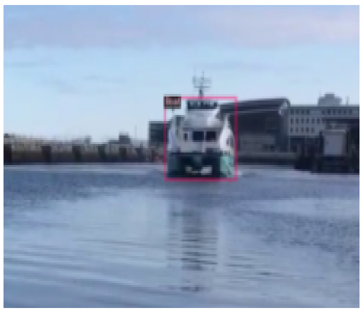
(a) Frame 1 - ground truth



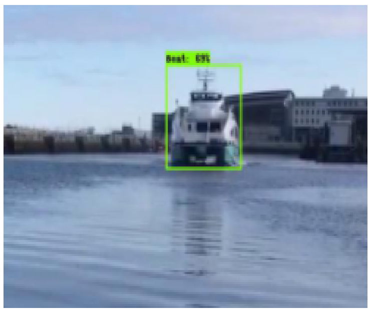
(b) Frame 1 - D3$_{FF}$-$\mathcal{D}_b$



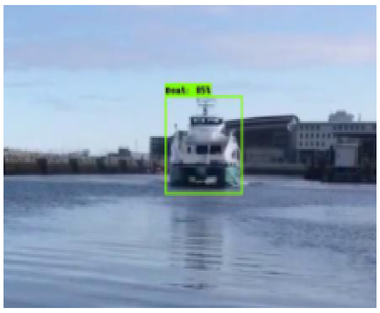
(c) Frame 1 - D3$_{FF}$-$\mathcal{D}_{NMD}$



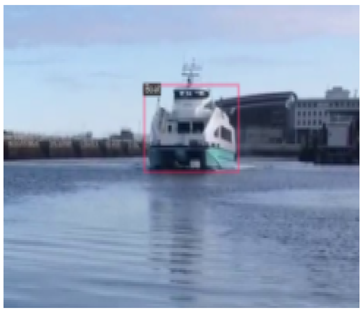
(d) Frame 10 - ground truth



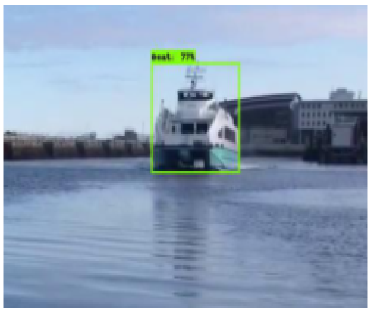
(e) Frame 10 - D3$_{FF}$-$\mathcal{D}_b$



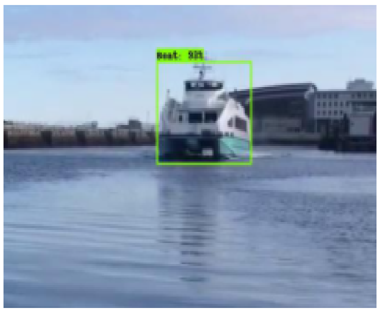
(f) Frame 10 - D3$_{FF}$-$\mathcal{D}_{NMD}$



(g) Frame 20 - ground truth



(h) Frame 20 - D3$_{FF}$-$\mathcal{D}_b$



(i) Frame 20 - D3$_{FF}$-$\mathcal{D}_{NMD}$

Figure 7.6: Ferry front-on scenario, D3$_{FF}$-$\mathcal{D}_b$ and D3$_{FF}$-$\mathcal{D}_{NMD}$ predictions. Images cropped for visibility.

7.4.3 Approaching kayak

Scenario The final case-study is likely one of the more challenging scenarios. It consists of 20 frames of a kayak instance, generated over a four second video captured from the sensor station of milliAmpere, in the harbour area around Ravnkloa. The first frame, presented in figure 7.7, represents the overall scenario. The kayak, marked with a red bounding box underneath the bridge, approaches milliAmpere front-on before passing by on the port side.

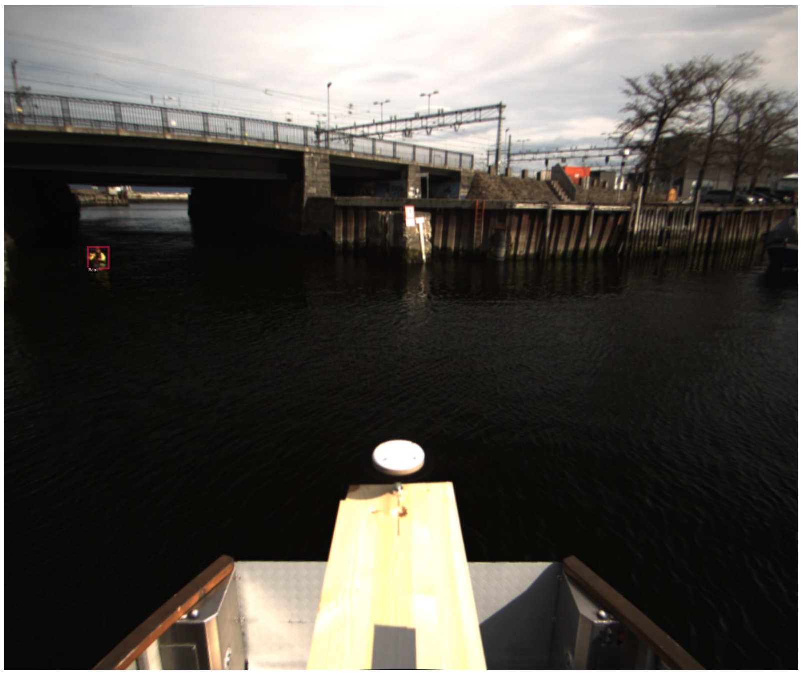


Figure 7.7: Approaching kayak scenario, first frame. The kayak instance is in the upper left part of the image, below the bridge, blending into the background.

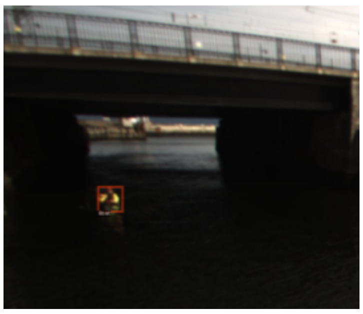
The lighting conditions are quite unusual, with the sea surface appearing completely black. Moreover, the boat hull of milliAmpere is present in all the 20 test frames adding extra noise and possible confusion for the detector. Since the target domain training set contains inherently few images captured from the sensor station, this is likely one of the more challenging in the wild prediction examples. Even though the colour of the kayak is in sharp contrast to the background, the kayak is partially covered in shadow, which makes particularly the first few frames extra challenging.

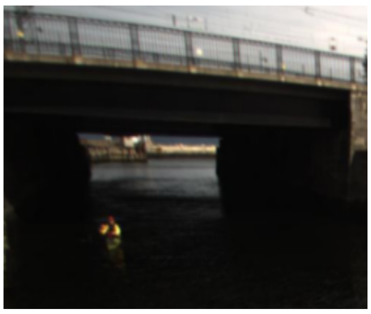
Prediction observations Figure 7.8 presents ground truths and predictions for the FF seen target models, D3$_{FF}$-$\mathcal{D}_b$ and the reported best model D3$_{FF}$-$\mathcal{D}_{NMD}$. Four sample frames of different distance and viewpoint of the kayak instance are chosen, by an interval of six frames.

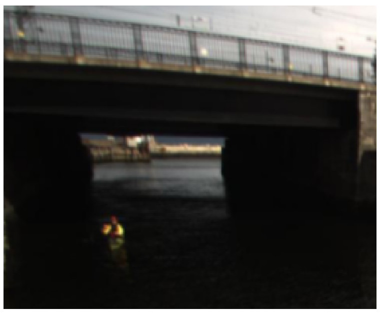
Firstly, D3$_{FF}$-$\mathcal{D}_b$ is observed to severely struggle in this scenario. In empirically assessing predictions for all 20 frames, D3$_{FF}$-$\mathcal{D}_b$ does not detect the kayak before the 13th frame, not visualized here. Contrarily, D3$_{FF}$-$\mathcal{D}_{NMD}$ initially detects the kayak on the third frame, also not visualized.

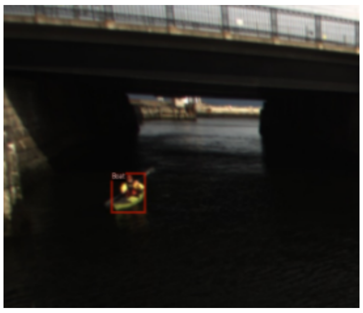
Moreover, the detections in figure 7.8 showcases the increased robustness of the detection pretraining on the NMD. D3$_{FF}$-$\mathcal{D}_{NMD}$ detects the kayak instance with high accuracy in all the presented frames except the initial frame. Additionally, from empirical observations there are no missed detection from the third frame and onward. One might also relate the consistent detection capability to the increased AR score of D3$_{FF}$-$\mathcal{D}_{NMD}$, mitigating FN predictions more frequently observed in D3$_{FF}$-$\mathcal{D}_b$.

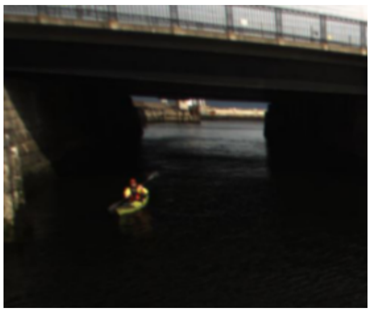
D3$_{FF}$-$\mathcal{D}_b$ is the only model which makes FP predictions in this scenario. The FP prediction appears for the boat hull of milliAmpere as presented in figure 7.9. Even though such FPs are easy to handle by filtering out predictions overlapping with the hull on the lower part of the scene, it indicates a lesser ability of separating foreground objects from noise.

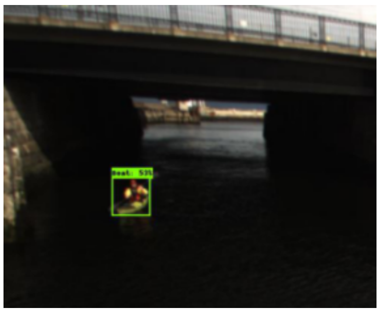

(a) Frame 1 - ground truth

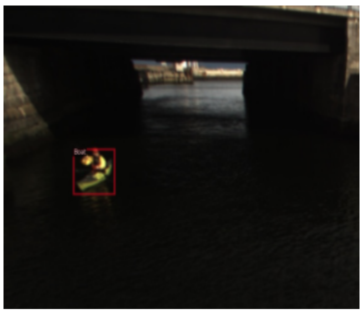

(b) Frame 1 - D3$_{FF}$-$\mathcal{D}_b$

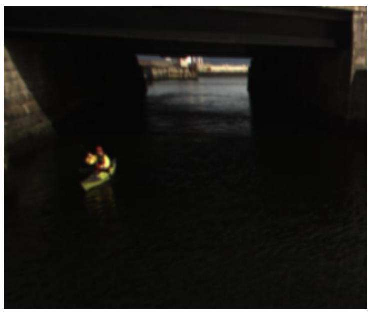

(c) Frame 1 - D3$_{FF}$-$\mathcal{D}_{NMD}$

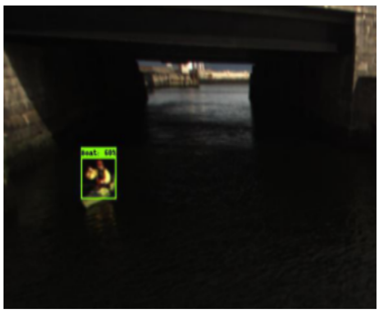

(d) Frame 6 - ground truth

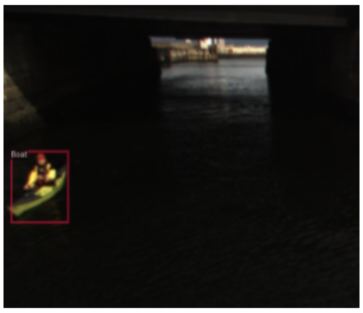

(e) Frame 6 - D3$_{FF}$-$\mathcal{D}_b$

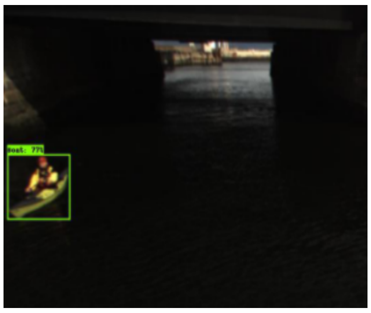

(f) Frame 6 - D3$_{FF}$-$\mathcal{D}_{NMD}$


(g) Frame 12 - ground truth


(h) Frame 12 - D3$_{FF}$-$\mathcal{D}_b$


(i) Frame 12 - D3$_{FF}$-$\mathcal{D}_{NMD}$


(j) Frame 18 - ground truth


(k) Frame 18 - D3$_{FF}$-$\mathcal{D}_b$

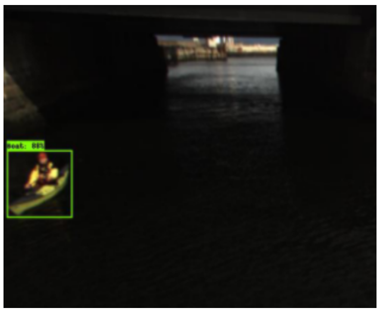

(l) Frame 18 - D3$_{FF}$-$\mathcal{D}_{NMD}$

Figure 7.8: Approaching kayak scenario, D3$_{FF}$-$\mathcal{D}_b$ and D3$_{FF}$-$\mathcal{D}_{NMD}$ predictions. Images cropped for visibility. The ground truth is marked by a red bounding box, slightly blending into the background.

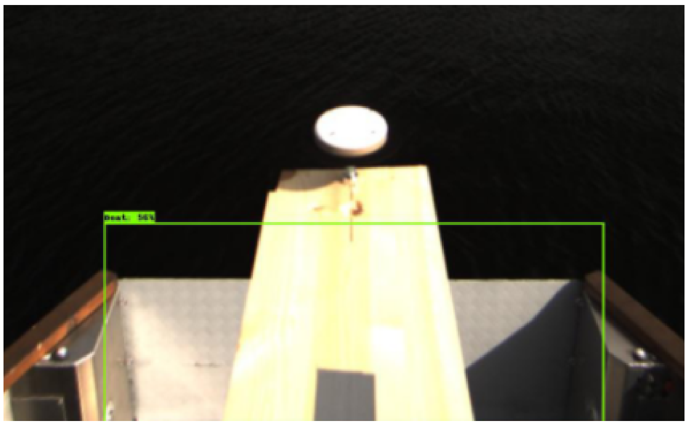


Figure 7.9: D3$_{FF}$-$\mathcal{D}_b$ FP prediction of boat hull - Frame 1.

7.5 Video inference

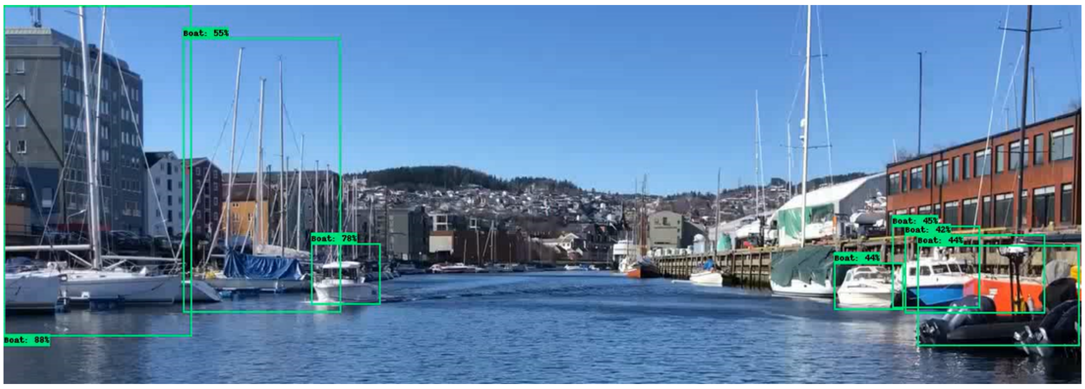
It is considered necessary to conduct a test on video inference in the wild, for a video of 25 FPS in accordance with the problem formulation treated in section 1.1. The best reported model, D3$_{FF}$-$\mathcal{D}_{NMD}$, is run for inference on a selected video scenario from the Ravnkloa harbour area.

The scenario is created with a white marker boat approaching and passing by the camera, situated on milliAmpere. It has been prioritized to present a challenging and realistic video inference scenario, as the best developed detector has been empirically observed to perform very well on single object scenarios. Four selected frames captured by pausing the video for different time duration are presented in figure 7.10.

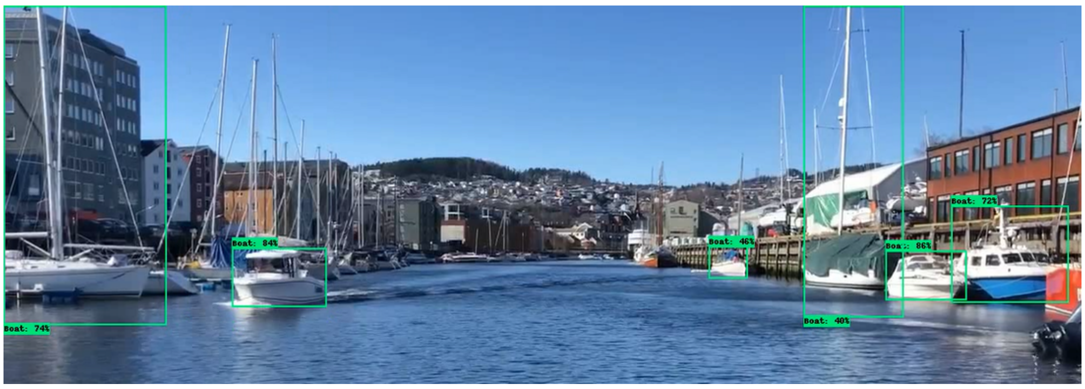
The detector has no problems in processing the video and generates predictions for all frames in the video. The approaching white marker boat on the left side of the scene, is well detected for the whole duration of the video, as may be observed in figure 7.10. One might consider this the most important instance to properly detect, as it is initially on collision course with milliAmpere.

However, unlike previous scenarios, the video scenario sheds light on some previously unseen weaknesses with D3$_{FF}$-$\mathcal{D}_{NMD}$, and also likely the other developed detectors. Namely, the detection of clustered boats in the background. Overall, the boats on the right side of the scene are detected, though not consistently and accurately over time for all instances.

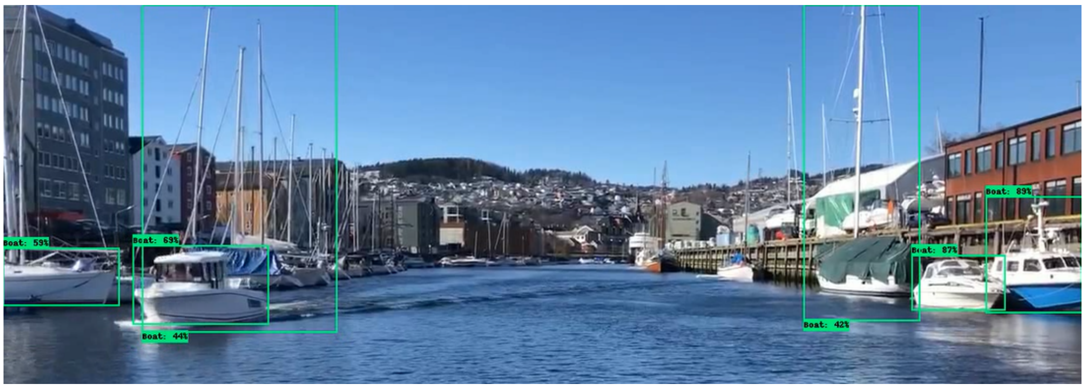
Nevertheless, on the left side of the scene, D3$_{FF}$-$\mathcal{D}_{NMD}$ generates very large bounding box predictions for the clustered sailboats in the background of the marker boat. For instance in figure 7.10d, one large predicted bounding box covers most of the docked sailboats. Clearly, this is no accurate prediction for separating between the docked boats. The poor performance for such instances may be linked to the target domain dataset composition, which is largely based on single-instance labeled images. Additionally, the images originating from the Grini dataset, where sailboats are labeled for the whole masts, possibly hinders the detector in properly separating such instances when closely clustered with multiple masts visible.



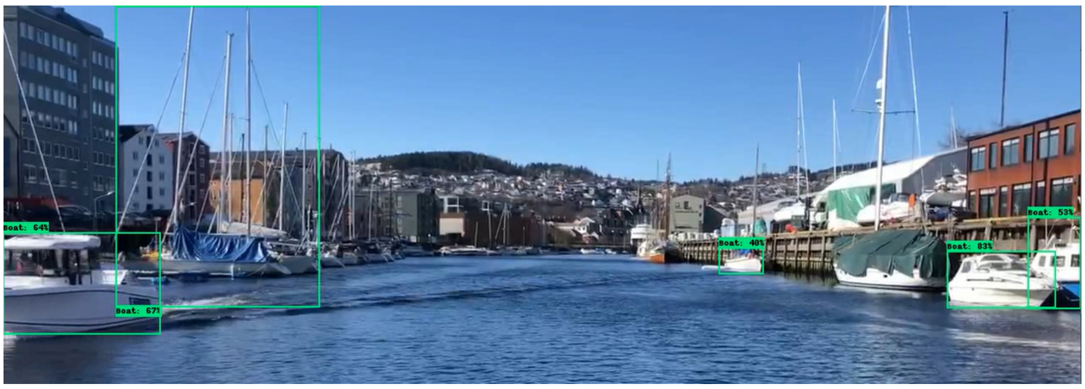
(a) Initial frame



(b) Intermediary second frame



(c) Intermediary third frame



(d) Last frame.

Figure 7.10: Video scenario, four sample frames acquired by pausing the video. D3$_{FF}$-$\mathcal{D}_{NMD}$ predictions.

CHAPTER 8 DISCUSSION

In this chapter, we present a further discussion on observations from chapter 7 as well as potential sources of errors in the conducted experiments.

8.1 Target domain dataset limitations

The target domain dataset used in this thesis is created from labeled images from the Grini dataset, as well as sampled images and videos by the author, counting a total of 1,061 images and three classes, if treated class-aware. Due to the small dataset size, there are some encountered problems which deserve a discussion.

Dataset bias Firstly, the target domain dataset is biased towards object instances and weather conditions encountered at the time of sampling. The data reused from the Grini dataset is sampled over one experimental day [35]. The similar is true for the author's collected data. The lighting and weather conditions occurring in the target domain dataset are therefore strongly affected by the short sampling periods. Likewise, the amount of unique instances are observably quite few, though not accounted for quantitatively.

In addition, while some instances are captured from camera pictures, other are captured on videos. There are rarely many camera pictures of the same instances. On the other hand, one video of a single instance, produces multiple frames of that same instance. This problem represents a sample-selection bias towards the filmed instances and is essentially a consequence of choosing to film certain instances, while taking pictures of other.

Furthermore, as our data collection was conducted with a marker boat, also used for simulating scenarios, as for instance the occlusion scenario in section 7.4.1, the marker boat occurs more frequently than other encountered motorboats. What is more, due to few encountered boat instances in the target domain, the majority of the generated frames contain single instances, as opposed to more challenging multi-object situations. As such, the designed detector's robustness on more challenging multi-object detection situations is uncertain. The challenging video inference scenario in section 7.5, indicates difficulties in detecting clustered background boats, typically observed in harbours.

Covariate shift Another related problem with the target domain dataset, which causes the previously mentioned validation and test set performance discrepancy in section 7.2, is a covariate shift between the split sets. The covariate shift is to some extent caused by the manual splitting procedure of the target domain dataset, as explained in section 4.5.1, as well as the scarce data foundation from the 1,061 images and the already established sample-selection bias from the target domain.

The dataset split was carried out manually as to avoid sample leakage[1] between the training and validation-test sets, while distributing the samples from each of the three classes evenly in each split set. Even though such a procedure is subject to bias from the author, a random dataset split would cause sample leakage as too many of the frames are generated from videos.

The majority of the 620 labeled images collected by the author, are generated from videos. These labeled images were assigned to each split set, by assigning all the frames generated from one unique video to one of the split sets. Especially for the small-scale validation and test set, this procedure causes problems. Namely, because the validation and test set are overrepresented by samples generated from quite few videos. As follows, the validation and test set internally undergo a sample-selection bias and a covariate shift.

As pointed out introductory in section 7.2, two particular problems transpire from the dataset split.

1) The validation and test set covariate shift causes very different reported scores for the two split sets.
2) The object size distribution between the split sets are inconsistent, also causing insufficient samples for evaluating AP_s properly.

The first problem is clear from the large performance difference of the seen target models' reported validation scores (see table 6.9 & table 6.10) and test scores (see table 7.2). It is inconclusive as to which of the validation and test set that reports the most representative target domain performance scores. However, as the validation and test set reported scores are consistent regarding the best fine-tuning setting and the superior performance of the targeted detection pre-training, we argue that the validity of the reported main results are conserved.

The second problem is an underlying problem of the internal sample-selection bias and the limited data foundation from the target domain dataset. To give more insight into the split set object size distributions, the COCO area classified object sizes of each target domain split set are presented in table 8.1.

Area	**Train**	**Val**	**Test**	**Total**
Small	134	2	10	146
Medium	443	136	288	867
Large	308	78	41	427

Table 8.1: Target domain dataset. Amount of bounding boxes per split, COCO area-classified.

Observably, the split sets have a very different object size distributions. We do not claim that an optimal split should have exactly equal weights of all object sizes. However, such disproportionate object size weighting as currently observed, also strongly suggests that each split set most likely contains very different images. In particular for small objects, which there are fewest samples of in the target domain dataset.

For instance, 134 of the total 146 small bounding boxes are restricted to the training set. Considering that the training set makes up 64.7% of the target domain dataset (see table 5.1), a larger quantity of the small object sizes should optimally be present in the validation and test set. Besides, if considering that the test and validation set is respectively 20.2% and 15.1% of the target domain dataset size (see table 5.1), the validation set also has an overweight of large samples compared to the test set. Lastly, the 10 small objects in the test set are clearly not sufficient for reporting metrics, causing AP_s to be quite unreliable.

[1]Also defined in section 4.3; Images containing strongly resembling instance viewpoint and scenes in training set and the validation-test set, potentially causing overfitting.

8.2 Domain distances

One of the largest drawbacks of our result analysis in this thesis, is the lack of a quantitative analysis of source to target domain distance. Optimally, such a measure would numerically or graphically measure the similarity between one source domain and the target, to support observations from the experiments and results.

As hypothesized in the experimental dataset creation of section 5.2, the NMD source domain is subjectively closest to the target domain. The results from section 7.1 and section 7.2 supports this hypothesis. The best reported unseen target model is D3-NMD, while the targeted detection pre-trained models, pre-trained on NMD, perform best for two out of three fine-tuning settings. As such, the features learned from NMD seems to be most transferable into the target domain.

To quantify why this is the case, is inherently more difficult, and have so far been avoided in the presented results. Intuitively, the NMD images are captured in Nordic maritime waters with the most similar boat instances and backgrounds to the target domain. Moreover, the RGB colour mean of the NMD (see table 6.4) is closest to the target domain (see table 6.8), indicating a possibly closer similarity in the RGB space. However, as the RGB mean is an accumulative metric, it is not very meaningful.

While the SMD represents an object-viewpoint more similar to the target domain than the NMD, it also includes a smaller variety of object types and is more heavily weighted towards larger ship/vessel types, rarely encountered in the target domain. Moreover, as pointed out in section 7.1 and also observable in section 7.2, NMD contains the lowest ratio of large objects, whereas the target domain contains the highest ratio among all datasets. Nevertheless, D3-NMD performs best on AP_l, and targeted detection pre-training on the NMD reports the best AP_l for two out of three fine-tuning settings. Further indicating, object size ratio similarity between source domain and target domain have little effect.

To summarize, most of the above-mentioned arguments are solely hypotheses based on subjectivity and some consistent reported test set observations. We therefore argue that a source to target domain measure is necessary for making further observations. It was not found time to implement any such measures in this thesis.

However, several options have been researched. Chu et al. [23] presented two distance measures directly calculated from the mean responses of one of the fully connected layers of a DCNN, namely the cosine distance and the maximum mean discrepancy between the mean responses. Similarly, an SVM and CNN classifier were trained to separate between source and target domain images, using the prediction accuracy as a similarity measure. It is however likely that the classifiers experience a bias related to the different amount of images in the source domains and the target domain, during training.

Additionally, generative models such as the Variational Auto Encoder [53] could be used for training on source and target domain images, before inspecting the projected latent space of the encoder for clusters of source and target domain image samples. Which would give a measure of the encoder's capability of separating between the images and implicitly the similarity of the domains.

8.3 Reported metrics & trustability

The reported metrics for all experiments in this thesis are the COCO AP metrics, which encapsulate model performance into comparable numerical values. By using these metrics, the performance gain delivered from targeted detection pre-training is suitably benchmarked with the baselines presented in table 7.2 of section 7.2. Moreover, the seen target models' performance can be compared with the reported EfficientDet-D3 test set scores on COCO [101] (AP=47.2, AP_{50}=65.9, AP_{75}= 51.2), to better understand the performance in a global perspective.

However, these metrics provide little insight into the explainability of certain detector predictions for use-cases in the target domain. Overall, in the presented case-studies of section 7.4 and from sample predictions, it is generally observed that D3$_{FF}$-$\mathcal{D}_{NMD}$ predicts tighter and more confident bounding boxes than D3$_{FF}$-$\mathcal{D}_b$. In the particular occlusion and kayak case-study, D3$_{FF}$-$\mathcal{D}_{NMD}$ proves to detect objects unobservable by D3$_{FF}$-$\mathcal{D}_b$. What is more, the case-study observations suggest D3$_{FF}$-$\mathcal{D}_{NMD}$ is better suited for continuously detecting the same object after initial detection than D3$_{FF}$-$\mathcal{D}_b$, implying a more reliable detector.

These observations can be linked to the better reported target domain test set scores of D3$_{FF}$-$\mathcal{D}_{NMD}$ versus D3$_{FF}$-$\mathcal{D}_b$. Similarly, it is possible that larger variety of boat instances and situations observed during pre-training for the the targeted detection pre-trained models, makes D3$_{FF}$-$\mathcal{D}_{NMD}$ more suitable for robust detection performance in difficult conditions where D3$_{FF}$-$\mathcal{D}_b$ struggles. Additionally, it is likely that the backbone features of the detection pre-trained models have higher neuronal activation distributed over the entire object regions, as found in [60].

Nevertheless, additional performance metrics, as presented below, would be useful for further confirming these observations. As such metrics have not been implemented in this thesis, the presented results in section 7.1 - 7.3 are intentionally presented in a quite generic manner, looking for patterns in the reported test score metrics.

To better understand the effects of the targeted detection pre-training, Explainable Artificial Intelligence (XAI) methods would be useful. Methods for explaining the neuronal activation in a DCNN are common in the literature [99] [60]. Li et al. [60] use class activation maps for demonstrating the effect of detection pre-training on the backbone. Methods such as Grad-CAM [90] could be implemented to further give insight into the learned backbone features with and without targeted detection pre-training.

Methods for explaining the detection rather than the feature activation, are not extensively researched to the same extent. Henriksen [43] used an XAI method named Local Interpretable Model-Agnostic Explanations (LIME) [87], which makes perturbations to input images for assessing the changes in the objectness scores and mask predictions of Mask R-CNN. The perturbed image is thereafter presented with a colour map of the superpixels with the highest significance for explaining the prediction. Such methods are also possible to implement by using the game-theoretical concept of Shapley values [92] as designed in the SHapley Additive exPlanations (SHAP) [66].

To better trust the detector in a spatio-temporal aspect and target domain deployment, where it is desired that the detector continuously detects the object after the initial detection, longer spatio-temporal case-studies, as well as quantitative metrics for observing lost track of detected objects would be useful. Detection histograms, as presented by Blanke et al. [15] for different object sizes, may be implemented to provide a visualization of detections over time, giving more insight into the spatio-temporal aspect. For multi-object detection, this would require tracking ID on the ground truth labels. Additionally, for trusting the detector in deployment, a simple tracking algorithm, such as DeepSORT [110] may also be implemented to model the position and velocity of the detected object, and improve detection performance for loss of detection, typically occurring in occlusion situations.

8.4 Targeted detection pre-training

The results reported in section 7.1 - 7.3, provides a thorough foundation for shedding light on the effects of targeted detection pre-training and source to target domain transferability. In this section we present a further discussion on the targeted detection pre-training experiments.

Pre-trained weights All trained models in this thesis are originally initialized from COCO pre-trained weights. The targeted detection pre-trained models are simply pre-trained a second time on a maritime source domain dataset for object detection.

An interesting experiment, not conducted in this thesis, would be to train EfficientDet-D3 from scratch on the same maritime source domain datasets. In such an experiment, it would be possible to ablate the pure benefit of maritime targeted detection pre-training. This is however not conducted, as the target domain dataset is too small in size for creating baseline models trained from scratch to compare against. However, such an experiment would provide additional information regarding the transferability between maritime source domains and the target domain.

Fine-tuning settings The three tested fine-tuning settings of this thesis (HO, FB, FF) are utilized as tools for conducting DA in the experiments. As such, the fine-tuning settings are not thoroughly compared against each other. Generally, it is observed that less freezing is superior to more freezing for the small-scale target domain dataset, except for the case of class-awareness, where there are fewer instances per class and freezing the backbone is more beneficial. A gradual fine-tuning scheme could prove to be more beneficial [43], but also inferior [85] or similar in performance [68] as full fine-tuning.

Targeted detection pre-training benefit The adoption of the targeted detection pre-training scheme has largely been motivated as to attempt reusing existing annotated maritime datasets to improve target domain performance. The reasoning behind this is simple; to avoid labeling data. Labeling data is time-consuming and manual labor. However, the reported results on the effects of targeted detection pre-training in chapter 7 are inherently linked to the target domain dataset size, which deserves a further discussion.

The agnostic seen target model results of section 7.2, demonstrate that full fine-tuning is the superior fine-tuning setting for all models, even for the target domain training set of solely 686 images (see table 5.1). Moreover, the FF baseline model, D3$_{FF}$-$\mathcal{D}_b$, is outperformed by all FF targeted detection pre-trained models. Also, as observed in section 6.3.2, the FF targeted detection pre-trained models have a much faster convergence and higher final AP than the baseline FF model.

As presented in [41], the effect of large-scale classification pre-training diminishes compared to from-scratch training if the detection architecture training schedule is modified accordingly, even for as "little" as approximately 10,000 images. He et al. also recommend to focus on collecting target data, while pointing out that classification pre-training clearly is beneficial for faster convergence and if limited target data is available.

Similar dataset size ablation studies have not been conducted for detection pre-training. Nonetheless, [91] shows that targeted detection pre-training achieves a faster convergence and a performance boost which is not achievable by from-scratch training, even for training on the full large-scale COCO dataset. This is similar to our observations. Additionally, it is shown that the neuronal activation between a classification and detection pre-trained model changes significantly [60].

The following question is therefore important to discuss: *Is targeted detection pre-training something which should be further pursued?*

As for the effects in the backbone neuronal activation, there is little basis for making conclusive statements. To allocate more resources for collecting target data is undoubtedly necessary to address problems with sample-selection bias and covariate shifts, as experienced in this thesis and previously discussed in section 8.1. It is not unlikely that the targeted detection pre-training benefits could be less significant for an overall larger target domain dataset.

However, depending on available resources and project time-constraints in designing and testing a detector for real-world applications, collecting more target data is not always possible. The author spent over a week designing the target domain dataset in this thesis. Allocating resources to do similar efforts for every specific maritime deployment use-case would be inherently expensive for the research community.

The explored targeted detection pre-training scheme presented in this thesis is a very good option for faster stream-lined detector development with increased target domain performance for small-scale maritime datasets. By using large existing annotated datasets, such as the publicly available SMD, it is possible to increase detector robustness and target domain performance, without extra cost in labeling and sampling more target domain data.

CHAPTER 9

CONCLUSION & FUTURE WORK

Camera-based object detection is one of the key sensors in the sensor suite of most autonomous systems. For ASVs, object detectors are inherently useful for recognizing close-by vessels and providing richer object-structure information than non-visual sensors.

In this thesis, a robust detector based on the EfficientDet-D3 architecture was designed and thoroughly use-case tested for maritime vessel detection on a meticulously designed target domain dataset, resembling the operational area of the autonomous ferry milliAmpere.

A literature survey was first conducted. Maritime object detection research was often found to be based on small-scale and custom annotated datasets [102] [15] [35]. Moreover, the consensus of classification pre-training for object detectors was found less beneficial than previously thought [41]. Detection pre-training, on the other hand, has proven to deliver faster convergence and overall better target domain performance for subsequent detection fine-tuning [60] [91]. Pre-training and fine-tuning for the detection task was defined as targeted detection pre-training.

Detection pre-training was conducted for EfficientDet-D3 with COCO initialized weights for three experimental maritime datasets, the largest covering a total of 17,871 optical RGB images with 95,398 instance labels. Fine-tuning into the target domain with detection pre-trained weights was conducted for three different fine-tuning settings; full fine-tuning (FF), frozen backbone (FB) and fine-tuning of the EfficientDet-D3 prediction heads only (HO). Baseline models were fine-tuned for the same fine-tuning settings, but with COCO initialized weights only.

The best reported model for evaluation on the target domain test set, was targeted detection pre-trained. Even though fine-tuned for 25% fewer epochs, it outperformed the best baseline with +5.9 on the COCO AP metric. The reported metrics and several case-studies indicated that targeted detection pre-training facilitated better detection robustness and fewer false-negative predictions. A video inference test in the wild demonstrated the best reported model to adhere with inference time requirements for deployment on milliAmpere, even though the detection of clustered docked boats could be improved.

We further present some of our main findings:

- *Targeted detection pre-trained models consistently converge faster and to higher performance scores than all baselines, even for fewer fine-tuned epochs.*
- *Targeted detection pre-trained models are more robust, mitigating false-negative predictions in challenging scenarios while producing tighter and more confident predicted bounding boxes.*
- *More freezing is inferior to full fine-tuning when the pre-training and target tasks and labels are the same.*

- *The benefits of targeted detection pre-training seem to depend more on pre-training dataset similarity to the target domain than the pre-training dataset size, even though this should be further researched.*

In conclusion, the targeted detection pre-training scheme has proven to highly benefit target domain performance by transferring knowledge from existing annotated maritime datasets, resulting in a robust detector suitable for deployment on milliAmpere. Our findings encourage the adoption of this scheme for faster and more robust detector development for small-scale maritime datasets.

However, our designed target domain dataset experienced biases and a covariate shift which motivates the collection of more target domain data to facilitate future detector design for milliAmpere. Further ablation studies should also be conducted to confirm the generality of results concerning targeted detection pre-training.

9.1 Future work

Based on the reported results in chapter 7 and discussion in chapter 8, there are several possibilities for future work.

- *More target domain data* should be collected to mitigate the target domain dataset bias and covariate shift experienced in this thesis. Optimally; reusing the annotated data used in this thesis, while sampling more data from different weather conditions, more varied boat types and multi-object situations. Additionally, satisfactory class-aware detection would need a more class balanced target domain dataset.
- *An improved target domain dataset split* designed as to guarantee well-representative split sets. A stratified dataset split, splitting on object classes, sizes, and potentially weather conditions could be considered.
- *Quantifiable spatio-temporal performance metrics* to provide more insight into the detection performance over time. For instance, a detection histogram of detected and missed instances over time in different video scenarios in the wild.
- *A real-time implementation* on milliAmpere. As to observe constraints and challenges linked to deployment, not covered in our case-studies.
- *Better object detectors* recently published in 2021 [114] [64] [25] may be considered to further improve detection performance.
- *Targeted detection pre-training ablation studies* for different maritime target domains and dataset sizes. Such studies would be interesting to analyse the generality of our results and also the effect of targeted detection pre-training for smaller and larger target domain datasets.
- *XAI methods* for providing more insight into the explainability of detector predictions and also better quantifying the effects of targeted detection pre-training. Methods like Grad-CAM [90], LIME [87] or SHAP [66] could be considered.
- *Implementing a domain distance measure* would additionally give more insight into maritime source to target domain similarity, and its importance for well adopted feature transferring.
- *Gradual fine-tuning* as a fine-tuning strategy could potentially better conserve pre-trained features and further improve target domain performance, both with and without targeted detection pre-training.
- *Larger maritime pre-training datasets* would also be beneficial to further ablate the effect of pre-training dataset size for targeted detection pre-training.

BIBLIOGRAPHY

[1] Coco detection evaluation. https://cocodataset.org/#detection-eval, 2020. Accessed: 2020-15-12.

[2] Efficientdet automl, Github repository. https://github.com/google/automl/tree/master/efficientdet, 2020. Accessed: 2021-06-05.

[3] Efficientdet automl gradual fine-tuning issue, Github repository. https://github.com/google/automl/issues/798, 2020. Accessed: 2021-06-05.

[4] Shipspotting. http://www.shipspotting.com/gallery/, 2020. Accessed: 2020-23-10.

[5] Yet-another-efficientdet-pytorch,Github repository. https://github.com/zylo117/Yet-Another-EfficientDet-Pytorch, 2020. Accessed: 2021-07-06.

[6] Hurtigruten minute by minute. https://tv.nrk.no/serie/hurtigruten-minutt-for-minutt, 2021. Accessed: 2021-25-02.

[7] Autonomous ferry bastø fosen VI. https://www.kongsberg.com/fr/maritime/about-us/news-and-media/news-archive/2020/first-adaptive-transit-on-bastofosen-vi/, 2021. Accessed: 2021-28-01.

[8] Mayflower autonomous ship. https://www.ibm.com/industries/federal/autonomous-ship, 2021. Accessed: 2021-28-01.

[9] Yara birkeland. https://www.yara.com/news-and-media/press-kits/yara-birkeland-press-kit/, 2021. Accessed: 2021-28-01.

[10] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL http://tensorflow.org/. Software available from tensorflow.org.

[11] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 2018.

[12] M. E. Aidouni. Evaluating object detection models: Guide to performance metrics. https://manalelaidouni.github.io/Evaluating-Object-Detection-Models-Guide-to-Performance-Metrics.html, 2019. Accessed: 2021-06-05.

[13] Z. Alom, T. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. V. Essen, A. Awwal, and V. Asari. A state-of-the-art survey on deep learning theory and architectures. *Electronics*, 8, 2019.

[14] B. Alsallakh, N. Kokhlikyan, V. Miglani, J. Yuan, and O. Reblitz-Richardson. Mind the pad – CNNs can develop blind spots. *ICLR*, 2021.

[15] M. Blanke, S. Hansen, J. D. Stets, T. Koester, J. E. Brøsted, A. L. Maurin, N. Nykvist, and J. Bang. Electronic outlook - a comparison of human outlook with a robotic solution. In *Proceedings of ICMASS*, 2018.

[16] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao. Yolov4: Optimal speed and accuracy of object detection. *ArXiv preprint arXiv:2004.10934*, 2020.

[17] D. Bolya, C. Zhou, F. Xiao, and Y. J Lee. Yolact: Real-time instance segmentation. *arXiv preprint arXiv:1904.02689*, 2019.

[18] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[19] S. Brahmbhatt, H. I. Christensen, and J. Hayes. Stuffnet: Using 'stuff' to improve object detection. *WACV*, 2017.

[20] X. Chen, R. Girshick, K. He, and P. Dollár. Tensormask: A foundation for dense object segmentation. *arXiv preprint arXiv:1903.12174*, 2019.

[21] X. Chen, C. Xie, M. Tan, L. Zhang, C. J. Hsieh, and B. Gong. Robust and accurate object detection via adversarial learning. In *Proceedings of CVPR*, 2021.

[22] T. Cheng, X. Wang, L. Huang, and W. Liu. Boundary-preserving mask r-cnn. *ECCV*, 2020.

[23] B. Chu, V. Madhavan, O. Beijbom, J. Hoffman, and T. Darrell. Best practices for fine-tuning visual classifiers to new domains. In *ECCV Workshops*, 2016.

[24] E. D Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation policies from data. *CVPR*, 2019.

[25] X. Dai, Y. Chen, B. Xiao, D. Chen, M. Liu, L. Yuan, and L. Zhang. Dynamic head: Unifying object detection heads with attentions. In *Proceedings of CVPR*, 2021.

[26] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. *CVPR*, 2009.

[27] D.Qiao, G.Liu, F.Dong, S.Jiang, and L.Dai. Marine vessel re-identification: A large-scale dataset and global-and-local fusion-based discriminative feature learning. *IEEE Access*, 2020.

[28] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian. Centernet: Keypoint triplets for object detection. *ICCV*, 2019.

[29] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes (VOC) challenge. *IJCV*, 2009.

[30] K. Fukushima. A hierarchical neural network capable of visual pattern recognition. *Neural Netw*, 1988.

[31] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *ArXiv preprint arXiv:1811.12231*, 2019.

[32] S. Gidaris and N. Komodakis. Object detection via a multi-region & semantic segmentation-aware cnn model. *ICCV*, 2015.

[33] R. Girshick. Fast R-CNN. *ICCV*, 2015.

[34] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR*, 2013.

[35] S. Grini. Object detection in maritime environments: Systematic training and testing of deep learning-based detection methods for vessels in camera images. Master's thesis, NTNU, `http://folk.ntnu.no/edmundfo/msc2019-2020/grini_simen_msc_reduced.pdf`, 2019.

[36] E. Gundogdu, B. Solmaz, V. Yücesoy, and A. Koç. Marvel: A large-scale image dataset for maritime vessels. In *Lecture Notes in Computer Science*, volume 10115. Springer, Cham, 2016.

[37] W. Hammedi, M. Ramirez-Martinez, Metzli, P. Brunet, S.-M. Senouci, and M. A. Messous. Deep learning-based real-time object detection in inland navigation. *GLOBECOM*, 2019.

[38] B. Hariharan, P. Arbeláez, R. Girshick, and J.-T. Malik. Simultaneous detection and segmentation. *ECCV*, 2014.

[39] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[40] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. *ICCV*, 2017.

[41] K. He, R. Girshick, and P. Dollár. Rethinking imagenet pre-training. *ICCV*, 2019.

[42] Ø. K. Helgesen. Sensor fusion for detection and tracking of maritime vessels. Master's thesis, NTNU, `https://folk.ntnu.no/edmundfo/msc2019-2020/masters_thesis_helgesen_reduced.pdf`, 2019.

[43] A.T. Henriksen. Domain adaptation for maritime instance segmentation: From synthetic data to the real-world. a study on generation and use of synthetic data in convolutional neural networks and on model-agnostic explenations in instance segmentation. Master's thesis, NTNU, `https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2631161`, 2019.

[44] J. H Hosang, R. Benenson, P. Dollár, and B. Schiele. What makes for effective detection proposals? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[45] C.-C. Hsu, K.-J. Hsu, C.-C. Tsai, Y.-Y. Lin, and Y.-Y. Chuang. Weakly supervised instance segmentation using the bounding box tightness prior. In *Proceedings of NeurIPS*, 2019.

[46] Y. Huang, Y. Cheng, D. Chen, H. Lee, J. Ngiam, Q. V. Le, and Z. Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In *Proceedings of NeurIPS*, 2019.

[47] E. H. Hølland. Maritime object detection using infrared cameras. Master's thesis, NTNU, `https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2625682?locale-attribute=en`, 2019.

[48] IMO. Convention on the international regulations for preventing collisions at sea, 1972 (COLREGs). `https://www.imo.org/en/About/Conventions/Pages/COLREG.aspx`. Accessed: 2021-08-03.

[49] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ArXiv preprint arXiv:1502.03167*, 2015.

[50] Ayush Jaiswal, Yue Wu, Pradeep Natarajan, and Premkumar Natarajan. Class-agnostic object detection. *ArXiv preprint arXiv:2011.14204*, 2020.

[51] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu. A survey of deep learning-based object detection. *IEEE Access*, 2019.

[52] K. Kim, S. Hong, B. Choi, and E. Kim. Probabilistic ship detection and classification using deep learning. *Applied Sciences*, 2018.

[53] D. P. Kingma and M. Welling. An introduction to variational autoencoders. *ArXiv preprint arXiv:1906.02691*, 2019.

[54] S. Kornblith, J. Shlens, and Q. V. Le. Do better imagenet models transfer better? *arXiv preprint arXiv:1805.08974*, 2018.

[55] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, T. Duerig, and V. Ferrari. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV*, 2020.

[56] K. Landsnes. Weakly-supervised instance segmentation for improved detection in maritime environments. Specialization project at NTNU, 2021.

[57] M. N. S. Larcher. K-means anchors ratios calculator, Github repository. `https://github.com/mnslarcher/kmeans-anchors-ratios`, 2020. Accessed: 2021-07-05.

[58] H. Law and J. Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of ECCV*, 2018.

[59] V. Lempitsky, P. Kohli, C. Rother, and T. Sharp. Image segmentation with a bounding box prior. *ICCV*, 2009.

[60] H. Li, B. Singh, M. Najibi, Z. Wu, and L. Davis. An analysis of pre-training on object detection. *ArXiv preprint arXiv:1904.05871*, 2019.

[61] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. Microsoft COCO: Common objects in context. *ECCV*, 2014.

[62] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. *arXiv preprint arXiv:1612.03144*, 2016.

[63] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. *ECCV*, 2016.

[64] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *ArXiv preprint arXiv:2103.14030*, 2021.

[65] I. Loshchilov and F. Hutter. SGDR: stochastic gradient descent with restarts. *ArXiv preprint arXiv:1608.03983*, 2016.

[66] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Proceedings of NeurIPS*, 2017.

[67] V. Marie, I. Bechar, and F. Bouchara. Real-time maritime situation awareness based on deep learning with dynamic anchors. *AVSS*, 2018.

[68] G. Montone, J. K. O'Reagan, and A. V. Terekhov. Gradual tuning: a better way of fine tuning the parameters of a deep neural network. *NeurIPS*, 2017.

[69] S. Moosbauer. Objetkdetektion in infrarot- und visuell-optischen videos mittels deep learning. Master thesis at Leipzig University of Applied Sciences. Provided by Sebastian Moosbauer in personal correspondence., 2019.

[70] S. Moosbauer, D. König, J. Jäkel, and M. Teutsch. A benchmark for deep learning based object detection in maritime environments. *CVPRW*, 2019.

[71] Y. Niitani, T. Akiba, T. Kerola, T. Ogawa, S. Sano, and S. Suzuki. Sampling techniques for large-scale object detection from sparsely annotated objects. *CVPR*, 2019.

[72] C. Nita and M. Vandewal. Cnn-based object detection and segmentation for maritime domain awareness. *SPIE: Artificial Intelligence and Machine Learning in Defense Applications II*, 2020.

[73] W. Ouyang, X. Wang, C. Zhang, and X. Yang. Factors in finetuning deep model for object detection. *CVPR*, 2016.

[74] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 2010.

[75] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[76] D. K. Prasad. Singapore maritime dataset ground truth description. `https://drive.google.com/file/d/0B10RxHxW3I92NjRjZnN1bjVjelk/view`, 2021. Accessed: 2021-22-04.

[77] D. K. Prasad. Singapore maritime dataset. `https://sites.google.com/site/dilipprasad/home/singapore-maritime-dataset`, 2021. Accessed: 2021-22-04.

[78] D. K. Prasad, C.K. Prasath, D.Rajan, L.Rachmawati, E.Rajabally, and C.Quek. Challenges in video based object detection in maritime scenario using computer vision. *ArXiv preprint arXiv:1608.01079*, 2016.

[79] D. K. Prasad, D. Rajan, L. Rachmawati, E. Rajabally, and C. Quek. Video processing from electro-optical sensors for object detection and tracking in a maritime environment: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 2017.

[80] D. K. Prasad, C.K. Prasath, D. Rajan, L. Rachmawati, E. Rajabally, and C.Quek. Object detection in a maritime environment: Performance evaluation of background subtraction methods. *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[81] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for activation functions. *ArXiv preprint arXiv:1710.05941*, 2017.

[82] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[83] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CVPR*, 2016.

[84] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.

[85] M. Reiersen. Deep visual domain adaption: From synthetic data to the real world. Master's thesis, NTNU, `https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2576033`, 2018.

[86] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Proceedings of NeurIPS*, 2015.

[87] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should i trust you?": Explaining the predictions of any classifier. *SIGKDD*, 2016.

[88] S. Ritter, D. G. T. Barrett, A. Santoro, and M. M. Botvinick. Cognitive psychology for deep neural networks: A shape bias case study. *ArXiv preprint arXiv:1706.08606*, 2017.

[89] B. Russell, A. Torralba, K. Murphy, and W. T. Freeman. Labelme: a database and web-based tool for image annotation. *IJCV*, 2007.

[90] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of ICCV*, 2017.

[91] S. Shao, Z. Li, T. Zhang, C. Peng, G. Yu, X. Zhang, J. Li, and J. Sun. Objects365: A large-scale, high-quality dataset for object detection. *ICCV*, 2019.

[92] L. S. Shapley. A value for n-person games. *Contributions to the Theory of Games 2 (AM-28)*, 1953.

[93] R.A. Shenoi, J.A. Bowker, A.S. Dzielendziak, A.K. Lidtke, G. Zhu, F. Cheng, D. Argyos, I.Fang, J. Gonzalez, and S. Johnson et al. Global marine technology trends 2030. Technical report, University of Southampton: Southampton, UK, 2015.

[94] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 2000.

[95] Y. Shinya, E. Simo-Serra, and T. Suzuki. Understanding the effects of pre-training for object detectors via eigenspectrum. *ArXiv preprint arXiv:1909.04021*, 2019.

[96] B. Singh, M. Najibi, and L. S. Davis. Sniper: Efficient multiscale training. *NeurIPS*, 2018.

[97] B. Solmaz, E. Gundogdu, V. Yucesoy, A. Koç, and A. A. Alatan. Fine-grained recognition of maritime vessels and land vehicles by deep feature embedding. *IET Computer Vision*, 2018.

[98] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[99] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *ICML*, 2019.

[100] M. Tan, B. Chen, R. Pang, V. Vasudevan, and Q. V. Le. Mnasnet: Platform-aware neural architecture search for mobile. *arXiv preprint arXiv:1807.11626*, 2018.

[101] M. Tan, R. Pang, and Q. V. Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of CVPR*, 2020.

[102] E.J Tangstad. Visual detection of maritime vessels. Master's thesis, NTNU, https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2452113, 2017.

[103] A. Telea. An image inpainting technique based on the fast marching method. *Journal of Graphics Tools*, 2004.

[104] B. Tilemachos. Singapore maritime dataset frames ground truth generation and statistics, Github repository. `https://github.com/tilemmpon/Singapore-Maritime-Dataset-Frames-Ground-Truth-Generation-and-Statistics`, 2021. Accessed: 2021-22-04.

[105] K. Turøy. Closed-loop collision avoidance system for the revolt model-scale vessel. Master's thesis, NTNU, 2020.

[106] J. Uijlings, K. Sande, T. Gevers, and W. W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 2013.

[107] C. Y. Wang, H. Y. M. Liao, I. H. Yeh, Y. H. Wu, P. Y. Chen, and J. W. Hsieh. Cspnet: A new backbone that can enhance learning capability of CNN. *ArXiv preprint arXiv:1911.11929*, 2019.

[108] C. Y. Wang, A. Bochkovskiy, and H. Y. M. Liao. Scaled-yolov4: Scaling cross stage partial network. *ArXiv preprint arXiv:2011.08036*, 2021.

[109] K. Weiss, T. M. Khoshgoftaar, and D. Wang. A survey of transfer learning. *Journal of Big Data*, 2016.

[110] N. Wojke, A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric. *arXiv preprint arXiv:1703.07402*, 2017.

[111] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? *NeurIPS*, 2014.

[112] Y. Zhang, Q.-W. Li, and F.-N. Zhang. Ship detection for visual maritime surveillance from non-stationary platforms. *Ocean Engineering*, 2017.

[113] Y. Zhong, J. Wang, L. Wang, J. Peng, Y.-X. Wang, and L. Zhang. DAP: detection-aware pre-training with weak supervision. *CVPR*, 2021.

[114] X. Zhou, V. Koltun, and P. Krähenbühl. Probabilistic two-stage detection. *ArXiv preprint arXiv:2103.07461*, 2021.

Domain Adaptation for Detection of Maritime Vessels in Images