Master's thesis

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

Ingunn Kjønås

Maritime Object Detection in LWIR-images using Deep Learning methods with Data Augmentation

Master's thesis in Electronic Systems Design and Innovation
Supervisor: Edmund Førland Brekke
Co-supervisor: Rudolf Mester and Egil Eide
June 2021

Photo: Mikael Sætereid

NTNU
Norwegian University of Science and Technology

Ingunn Kjønås

Maritime Object Detection in LWIR-images using Deep Learning methods with Data Augmentation

Master’s thesis in Electronic Systems Design and Innovation
Supervisor: Edmund Førland Brekke
Co-supervisor: Rudolf Mester and Egil Eide
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

Abstract

The context of this project is the use of multisensor multitarget tracking for an Autonomous Surface Vehicle (ASV) in a harbour environment. The purpose of the research is to locate and track an ASV in combination with other targets for collision avoidance in autonomous navigation. The final goal is to improve robustness and reliability of the tracking system by means of sensor fusion. Infrared cameras can improve night vision, improve resolution and provide more feature information. Therefore, this thesis focuses on detection performance in infrared images.

To address this, a literature review is conducted covering approaches to object detection in maritime infrared images, with focus given to neural networks and data augmentation techniques.

There are few available annotated Long Wave Infrared (LWIR) images of boats, therefore more images are collected and annotated with the purpose of training and testing neural networks on the data. The neural network models YOLOv3 and EfficientDet-D0 are trained and tested on the available and collected data and their performance is compared.

Data augmentation is a frequently used technique in the general computer vision community in order to increase the variation in the training data, but no studies have previously examined the effect on maritime LWIR images. Because of this and motivated by the limited available dataset, the effect of data augmentation during training of the neural networks is examined in this thesis.

The results show that both models perform well with a probability of detection of 100% for two moving target boats when the pixel area size is above a threshold of 1800. For smaller objects, the detection performance is significantly reduced, showcasing a limited range of infrared camera object detection. The comparison of the models shows that YOLOv3 performing slightly better for smaller targets, although the effect is to small to conclude that one model is superior to the other.

The effect of the combined data augmentation techniques flip, scale and mosaic is significant increase in performance for both models, with mosaic providing the greatest improvement.

Finally, for the application of collision avoidance it can be useful to extract information related to the type of boat, which can be used for instance for estimation of velocity and heading. To test the possibility of separating motorboats from sailboats, the neural networks are tested with detection and classification combined, resulting in promising performance, although misclassifiactions are common and more false positive predictions are introduced than when training on one boat-class.

Sammendrag

Konteksten for denne oppgaven er bruk av multisensor målfølging av flere mål for et autonomt overflatefartøy i et havneområde. Hensikten med forskningen er å lokalisere og målfølge en autonom ferge kombinert med andre mål og hindringer for å unngå kollisjon i autonom navigasjon. Det endelige målet er å forbedre robustheten og påliteligheten til målfølgingssystemet ved hjelp av sensorfusjon. Infrarøde kameraer kan forbedre nattsynet og oppløsning, i tillegg til å gi mer informasjon knyttet til målets egenskaper. Derfor fokuserer denne oppgaven på deteksjon av objekter i infrarøde bilder.

For å adressere dette gjennomgås relevant litteratur som dekker tilnærming til deteksjon av objekter i maritime, infrarøde bilder, med spesielt fokus på nevrale nettverk og teknikker for forøkning av data.

Det finnes få tilgjengelige annoterte langbølge-infrarød bilder av båter, slik at mer data er samlet inn og annotert med mål om å trene og teste nevrale nettverk ved bruk av disse bildene. De nevrale nettverksmodellene YOLOv3 og EfficientDet-D0 er trent of testet på tilgjengelig og innsamlet data, og ytelsen deres er sammenlignet.

Dataforøknings-teknikker blir ofte brukt i det generelle datasyn domenet for å øke variasjonene i treningsdataen, men ingen studier er så langt utført for å undersøke effekten på maritime langbølge-infrarød bilder. På grunn av dette og kombinert med et størrelsesbegrenset datasett, blir den potensielle forbedringseffekten av dataforøkning under trening av de nevrale nettverkene testet i denne oppgaven.

Resultatene viser at begge modellene presterer bra med en deteksjonssannsynlighet på 100% for to båter i bevegelse når pikselarealet for båtene er over 1800. For mindre objekter blir deteksjonsresultatene betraktelig dårligere, hvilket viser at det er en avstandsgrense for deteksjon av målene i de infrarøde bildene. Sammenligning av de to modellene viser at YOLOv3 presterer litt bedre på deteksjon av små objekter, selv om effekten er for liten til å konkludere med at en modell er bedre enn den andre.

Effekten av å kombinere dataforøkningsteknikkene vending, skalering og mosaikk er signifikant forbedring av resultatene for begge modeller, hvor mosaikk gir den største forbedringen.

Når deteksjonene skal brukes til kollisjonsunngåelse kan det være nyttig å hente ut informasjon knyttet til type båt, noe som kan brukes blant annet til å estimere hastighet og vinkel på målet. For å undersøke mulightene til å skille motorbåter fra seilbåter er de nevrale nettverksmodellene testet med deteksjon og klassifikasjon kombinert. Dette resulterer i lovende ytelse, selv om misklassifiseringer er vanlige og det fører til flere falskt positive prediksjoner sammenlignet med trening på én båt-klasse alene.

Contents

Abbreviations

AGC Automatic Gain Correction. 35–38

AP Average Precision. iv, 17–19, 24, 27, 47, 50

AR Average Recall. iv, 18

ASV Autonomous Surface Vehicle. i, 1

BiFPN Bidirectional Feature Pyramid Network. 28, 29

CNN Convolutional Neural Network. 14, 21, 23, 24

COCO Common Objects in Context. v, 15, 23, 31, 32, 45, 46, 50, 56, 71

COLREGs Convention on the International Regulations for Preventing Collisions at Sea, 1972. 6

CSPNet Cross Stage Partial Network. 27

DDE Digital Detail Enhancement. 38, 39, 41, 72

EO Electro-Optical. 6, 7, 21

FN False Negative. 16

FOV Field of View. 37, 38, 57

FP False Positive. 16, 18

FPN Feature Pyramid Network. 14, 26, 27

FPS Frames Per Second. 24, 25

GAN Generative Adversarial Network. 29, 30, 72

GT Ground Truth. 15, 16

IoU Intersection over Union. iv, 15, 18, 26, 51, 52, 60–63, 65, 67, 69, 70

IR Infrared. iii–v, 1–3, 5–8, 22, 23, 25, 29, 32, 34, 35, 38–41, 46, 52, 55, 60, 64, 65, 69, 70

LWIR Long Wave Infrared. i, v, 2, 3, 6, 22, 32–35, 45, 46, 71, 72

mAP mean Average Precision. 18

MWIR Mid Wave Infrared. 6

NIR Near Infrared. 3, 6

NTNU Norwegian Uinversity of Science and Technology. 1

PANet Path Aggregation Network. 27

PCA Principal Component Analysis. 23

ROI Region of Interest. 37–39, 41, 72

RPN Regional Proposal Network. 25

SPP Spatial Pyramid Pooling. 27

SSD Single Shot MultiBox Detector. 22, 25, 26

SWIR Short Wave Infrared. 6

TIR Thermal Infrared. 6

TP True Positive. 16

YOLO You Only Look Once. 25, 26

Chapter 1

Introduction

1.1 Background

The background for this master thesis is the implementation of a collision avoidance system based on sensor fusion for the Autonomous Surface Vehicle (ASV) milliAmpere. ASVs are autonomous vehicles that operate on the surface of the water. Information from multiple sensor are processed in order to perceive the surroundings by identifying possible obstacles and finding appropriate navigation paths.

Norwegian Uinversity of Science and Technology (NTNU) leads a project called Autoferry which has developed a research platform for studies and experiments in the field of autonomous all-electric passenger ferries for urban water transport called milliAmpere. Currently a prototype is available for experiments while a complete ASV is under construction with the goal of transporting passengers and bicycles. The ferry will operate in Trondheim, Norway, between Ravnkloa and Fosenkaia.

As a part of this project, an important research topic is multitarget multisensor tracking for collision avoidance. This is the problem of estimating the states or trajectories of an unknown number of targets from multiple sensor measurements. The sensors used for observation of targets are radars, lidars and cameras, included Infrared (IR) cameras. By fusing data from several sensors, the information on which the collision avoidance system base its decisions is more secure and we thus minimize the risk. Specific advantages of including IR cameras are more feature information compared to radar measurements and better night vision than visible range cameras.

Previous work within the field of multisensor fusion with cameras include [Helgesen et al., 2019] which examines a measurement level sensor fusion system for tracking in a maritime environment using lidar, radar, visible light cameras and IR cameras. In this study, it was found that the system's performance and robustness were improved when including the IR cameras. A neural network based object detection applied on visible light maritime images was tested in [Kamsvåg, 2018] with promising detection performance on very close ranges up to 20 meters, and many misdetections for larger ranges. The results were fused with lidar-data, and the author shows that including the camera detections showed potential for tracking improvement at close ranges.

This master thesis builds upon previous work in the author's specialization project [Kjønås, 2021]. This project examined object detection on IR videos in the maritime domain where a mean and standard deviation based dynamic background subtraction method, which was compared to a neural network trained solely on RGB-images. The results were not very good, giving low probabilities of detection (recall) for both methods. For the background subtraction method the main problem was wakes causing distortion of bounding boxes and abrupt changes caused by internal camera noise. For the neural network the main problem was identified to be that training on relevant IR images would be necessary. In addition, the contrast in the collected IR images was very low, encouraging improvements in future studies.

Using neural networks for object detection in images is a popular and well performing method that has received increased attention from the maritime research field [Prasad et al., 2017]. Thus, it is decided to continue the work from the specialization project by focusing on neural networks for object detecition in maritime IR images in this thesis.

1.2 Contributions

The application of neural networks for object detection in maritime IR images is a topic where not much research has been conducted. One of the main contributions is [Schöller et al., 2019], who compare three neural networks on Long Wave Infrared (LWIR) images. However, this and other studies focus on the open sea domain, while milliAmpere will operate in a cluttered harbour environment. Thus the motivation behind this thesis is to highlight important aspects that needs to be considered moving forward on this topics through testing and comparison.

Data augmentation, which is a technique for increasing the variation in the training data frequently used on visible light images, has not been tested on maritime LWIR images as far as the author is aware. Therefore, this is an important research question examined in this thesis.

In addition, a lot of research is conducted within the computer vision field on modern neural network models that improve the general performance significantly. These studies are very recent and many models have therefore not been tested in maritime IR-images. [Schöller et al., 2019] uses models from 2015, 2017 and 2018, which will be further presented in the literature review in chapter 5. Similarly [Helgesen et al., 2019] uses a model from 2016 on Near Infrared (NIR) images. Thus another interesting problem investigated in this thesis is comparison of a new state-of-the-art model, EfficientDet [Tan et al., 2020], with a more frequently used model, YOLOv3 [Redmon and Farhadi, 2018].

Based on these research topics, this thesis searches to answer the following questions:

- Does training on LWIR-images, particularly the specific available and collected dataset, improve detection performance compared to only using networks pre-trained on RGB images?
- Can we provide an indication of whether newer models with better general performance also performs better on maritime IR images?
- Should data augmentation be used? In which case, which techniques improve the results the most and are recommended?
- Can we use the neural networks for the classification task of distinguishing between sailboats and motorboats in IR-images?
- How can the results be improved moving forward?

Another important contribution from this thesis is the improvements of the contrast in collected LWIR images by methodological adjustments of camera parameters.

1.3 Outline of the thesis

Theory regarding IR imaging and the motivation for including IR cameras will be presented in chapter 2. Chapter 3 covers theory on the topic of deep learning and neural networks. The following chapter 4 presents the theory related to the evaluation metrics that are used for evaluation of the results in this thesis.

Next, a literature review is given in chapter 5 covering previous work of object detection in IR images and maritime domain, as well as a comparison of neural network models and data augmentation techniques.

Available datasets and the method for collection, and annotation of more training and test data is presented in chapter 6. Included in this is the method for adjustments of camera parameters in order to improve the contrast in the IR images.

Next, chapter 7 covers the choice, setup and training of the two neural network based object detection models with data augmentation techniques. In addition, the method and implementation of associated code for further analysis is covered.

The results of the analysis is presented in chapter 8, aiming to answer the questions provided in section 1.2, including comparison of the models, the effect of data augmentation and the classification performance.

Chapter 2

Theory: Infrared (IR) imaging

2.1 IR radiation

Infrared radiation is radiation of electromagnetic waves at a given range of wavelengths. In general, every object that has a temperature above absolute zero will emit thermal radiation which will be distributed over a range of wavelengths [Rees, 2012]. Spectral radiance L_λ is the differential of radiance for the wavelength λ, and for a black body it is given by (2.1) as:

$$L_{\lambda,P} = \frac{2hc^2}{\lambda^5(\exp(\frac{hc}{\lambda kT}) - 1)} \tag{2.1}$$

where h is the Planck's constant, c is the speed of light, k is the Boltzmann constant and T is the absolute temperature of the body. The subscript P stands for Planck, as this is the black body behaviour, meaning that the body is a perfect emitter of thermal radiation. Real materials do not behave as a black body, thus the parameter emissivity $\varepsilon(\lambda)$, dependent on the wavelength, is introduced to relate $L_{\lambda,P}$ to the actual radiance of a body [Rees, 2012]:

$$L_\lambda = \varepsilon(\lambda)L_{\lambda,P} \tag{2.2}$$

As the emissivity is material dependent, (2.2) shows that the thermal radiation will vary for different materials and thus uniform temperatures can result in different pixel values when using thermal imaging systems.

2.2 IR spectrum

The infrared wavelength spectrum is often divided in sub-bands as shown in figure 2.1 because they have different properties.

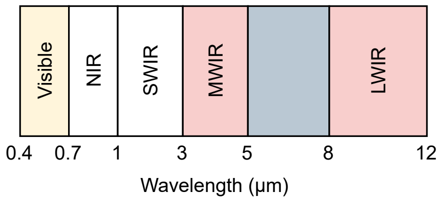


Figure 2.1: Infrared wavelengths on the electromagnetic spectrum. Thermal Infrared in red and visible light in yellow. Atmospheric absorption is very high in the grey area, so this is excluded. Figure inspired by [Berg, 2016]

While Near Infrared (NIR) and Short Wave Infrared (SWIR) cameras mostly depict radiation reflected from the surrounding scene, Mid Wave Infrared (MWIR) and Long Wave Infrared (LWIR) measure the emitted radiation and temperature[Berg, 2016]. Therefore, the latter two are often referred to as Thermal Infrared (TIR). In this thesis a LWIR camera is used, thus we will focus on LWIR imaging. The camera functionality and specifications are described in section **??**.

2.3 Advantages and disadvantages of IR imaging

All ships are required to have radars and use them for determining risk of collision according to Convention on the International Regulations for Preventing Collisions at Sea, 1972 (COLREGs)[1]. However, radar data have drawbacks such as sensitivity to rain and fog as well as the shape, size, and material of the targets [Prasad et al., 2017]. In experiments from [Helgesen et al., 2019] a radar reflector was mounted on the kayak, showing that this is an object that is particularly difficult for the radar to detect due to non-reflective materials and smaller size. The humans in kayaks should make them easier to detect with IR-cameras due to thermal radiation.

In addition, for situational awareness of autonomous vessels sensor fusion is considered a very important aspect. Fusion of data from several sensors such as radars, lidars, sonars and Electro-Optical (EO)-cameras can result in more information on which decision making for collision avoidance can be based and increased security.

Some advantages of including EO-sensors such as IR-cameras and visible light cameras are [Prasad et al., 2017]:

[1]https://www.imo.org/en/About/Conventions/Pages/COLREG.aspx

- Intuitive for users, no need for specific training to interpret the data
- We can extract more information related to type of vessel, size and angle of object appearance (front, side, back) to give indication of heading than e.g. radar

There are some obvious drawbacks with EO-sensors, meaning that the information used for situational awareness from these sensors alone is limited. Some of these drawbacks are [Prasad et al., 2017]:

- Difficult to predict distance to detected objects
- Shorter range due to atmospheric propagation losses. The detection range of the IR-camera used in this project will be further investigated in the result section.
- Sensitive to illumination and weather changes
- Computationally heavy

Furthermore, [Prasad et al., 2017] summarizes some of the advantages of including IR-cameras compared to visible light cameras:

- Longer range
- Better nightvision as thermal radiation is measured
- Less dynamic water movements which is an advantage for processing methods exploring temporal features

On the other side, the resolution and optic parameters are in general worse than for visible light cameras, and color features are lacking.

Chapter 3

Theory: Deep learning

The following chapter giving an introduction to the theory behind deep learning and neural networks is fetched from the author's project thesis [Kjønås, 2021] as the same theory was relevant here, however it is edited and adapted to this master thesis. Sections 3.2.1 and 3.4 are new.

Machine learning has received an increased amount of attention from the computer vision community the last two decades due to its impressive performance on detection and classification tasks in images. This section aims to give a brief introduction to deep learning using neural networks which is a subgroup of machine learning methods. The theory in this section is based on [Goodfellow et al., 2016].

Deep Learning is based on deep graphs with many layers on top of each other. The idea is that abstract, high-level features can be defined/computed in relation to simpler ones. The first layers in the deep learning method are then responsible for extracting low-level features, which in the case of images can be edges and colors, and high-level features such as wheels and leafs are extracted through a combination of the previous ones. Finally, a mapping is made from the features to output on the desired form. Included in the concept of machine learning is automating the process based on data used as training input.

3.1 Neural networks

A feedforward deep neural network is a model for approximating a function based on a composition of simpler functions. This is done by connecting a set of neurons responsible for simpler operations through multiple layers, inspired by neural connections in a brain. In its simplest form, each neuron learns an affine (linear) transformation of several inputs

$\boldsymbol{x}$ to an output y by applying a set of weights $\boldsymbol{w}$ and a bias b:

$$y = \boldsymbol{w}^T \boldsymbol{x} + b \tag{3.1}$$

The complete network consists of an input layer and an output layer, each containing several neurons, with multiple hidden layers in between that extract features. An example with one hidden layer is visualized in figure 3.1.

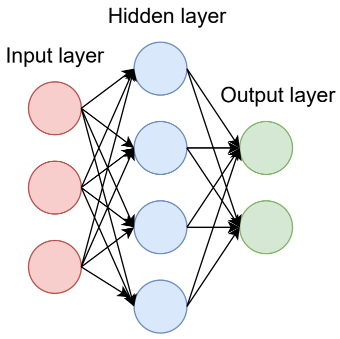


Figure 3.1: *Figure from [Kjønås, 2021]*
A neural network with an input layer, one hidden layer and an output layer

However, such a network where each neuron represents an affine transformation can never result in a more complex mapping of input to output, and the hidden layers would inevitably be of no use. This is due to its linear properties where combing two affine functions results in another affine transform. We can show this if we have $y_j = \boldsymbol{w}_j^T \boldsymbol{x} + b_j = \sum_i w_{ij} x_i + b_j$, then the final output from one neuron z is

$$\begin{aligned}
z &= \boldsymbol{v}^T \boldsymbol{y} + c \\
&= \sum_j v_j \left(\sum_i w_{ij} x_i + b_j \right) + c \\
&= v_1 w_{11} x_1 + v_1 w_{12} x_2 + ... + v_2 w_{21} x_1 + v_2 w_{22} x_2 + ... + v_1 b_1 + v_2 b_2 + ... + c \\
&= (v_1 w_{11} + v_2 w_{12} + ...) x_1 + (v_1 w_{21} + v_2 w_{22} + ...) x_2 + ... + (v_1 b_1 + v_2 b_2 + ... + c) \\
&= \sum_i u_i x_i + d \\
&= \boldsymbol{u}^T \boldsymbol{x} + d
\end{aligned}$$

where the terms that are independent on the input at step 4 are replaced by the new weights $\boldsymbol{u}$ and bias d, which is just another affine transform.

This means that a neural network with these properties would not be able to approximate functions that include non-linearities, which is often the case in real world problems. In order to solve this issue, nonlinear activation functions are applied to each neuron after the affine transform. An example of an activation function is the Sigmoid Linear Unit (SiLU) shown in (3.2), which was introduced in [Elfwing et al., 2018].

$$a(y) = y\sigma(y) = y \cdot \frac{1}{1+e^{-y}} \tag{3.2}$$

where $\sigma()$ is the sigmoid function. Note that y is the output of an affine function, which is the same as in (3.1).

3.2 Learning using gradient descent

Supervised learning algorithms train on input data with known desired output in order to find the optimal weights and biases that approximate the input to output mapping. This optimization problem is phrased in terms of minimizing a **cost function**, also known as criterion or loss function, that tells us how well the network has performed by evaluating the probability that the network gives for the desired output z conditional on the input $\boldsymbol{x}$ and parameter set $\boldsymbol{\theta}$ containing weights and biases. In a classification task, the desired output is typically on the form of a vector with a probability for each class z, $p(z)$, which evidently must sum up to 1 over all classes for a given input and parameter set. A commonly used cost function for classification problems is the cross-entropy:

$$C(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} -\log p(z^i|\boldsymbol{x}^i;\boldsymbol{\theta}) \tag{3.3}$$

where we are averaging over m samples of training data, and $p(z^i|\boldsymbol{x}^i;\boldsymbol{\theta})$ is the probability for the sample i of the true class z given the input $\boldsymbol{x}$ and the parameter set $\boldsymbol{\theta}$. Finally, the parameter set $\boldsymbol{\theta}$ is updated through back-propagation of the error given by the cost function. This done in terms of **gradient descent**, where we can decrease a function by moving a small enough step in the direction of the negative gradient:

$$\boldsymbol{\theta}' = \boldsymbol{\theta} - \epsilon\nabla_{\boldsymbol{\theta}}C(\boldsymbol{\theta}) \tag{3.4}$$

Here, the learning rate ϵ is determining the step-size, while $\nabla_{\boldsymbol{\theta}}$ is the gradient operator that gives all partial derivatives of the cost function $C(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$.

3.2.1 Hyperparameters

Batch size

Computing the gradient descent from the entire training set of input data is very computationally expensive and time consuming. Limitations of GPU-memory is another obstacle for using all training images for gradient descent computation. Since the gradient is an expectation it can be approximated using a smaller set of samples [Goodfellow et al., 2016]. By using a subset of the training samples in a minibatch of **batch size** m' we can estimate the gradient cross-entropy cost function $\boldsymbol{g}$ as

$$\boldsymbol{g} = \frac{1}{m'} \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m'} - \log p(z^i | \boldsymbol{x}^i; \boldsymbol{\theta}) \tag{3.5}$$

where the input $\boldsymbol{x}^i$ is drawn from the minibatch. Then we can update (3.4) by replacing $\nabla_{\boldsymbol{\theta}} C(\boldsymbol{\theta})$ with the estimate $\boldsymbol{g}$.

The batch size thus determines the number of inputs on which to calculate the gradient. An advantage of limiting the batch size is that this results in a noisy gradient which gives a regularizing effect that can give more robust model.

Epoch

An **epoch** is when all input data in the training propagated through the neural network once. After each epoch the neural network is evaluated on a validation set and based on the results we can make choices regarding the hyperparameters e.g. by updating the learning rate in the gradient descent. The reason for this is to avoid overfitting. Overfitting is when we create a complex mapping that is too closely related to the training data so that the model is not able to *generalize* when prediction is performed on new unseen data.

Step size

The **step size** is related to the batch size and epochs as the following equation:

$$steps = \frac{num\_train\_imgs \cdot epochs}{batch\_size} \tag{3.6}$$

This parameter is typically decided indirectly by defining the number of epochs and the batch size.

3.3 Convolutional neural network (CNN)

Convolutional neural networks (CNNs) are neural networks where the affine function $\boldsymbol{y} = \boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{x} + \boldsymbol{b}$ of at least one layer of neurons is replaced by a convolution [Goodfellow et al., 2016]. This has proven to be very efficient for images and other data with a grid-like topology. The regular neural network using matrix multiplication can perform all the same tasks as a CNN, however due to the large number of pixels that can be contained in an image, a fully connected layer will have a large memory requirement and large amount of operations needed. Thus, the reader should be aware that the main motivation for using CNNs is the efficiency award.

3.3.1 Convolutional layers

In a convolutional layer each output neuron $S(i, j)$ is computed from the multi-dimensional input image I $(H \times W \times M)$ by using a multi-dimensional kernel K $(D_K \times D_K \times M \times N)$ in a convolutional operation, which in the case of a two dimensional input and one channel N for the kernel becomes:

$$S(i,j) = (I * K)(i,j) = \sum_{k_1} \sum_{k_2} I(i - k_1, j - k_2) K(k_1, k_2) \tag{3.7}$$

The complete output of this layer would be of dimensions $H \times W \times N$. However, we can also introduce the parameter stride S, that determines the number of pixels moved after each convolution, resulting in a scaling of the height and width for the output.

We can see that the convolutional operation is the same linear function using weights as for the regular neural network, except that the output is only connected to a local region of the input. To represent the same function in a fully connected fashion, there would be a lot of weights equalling zero, meaning that we save a lot of computational power and memory of stored weights. In addition, re-using the same weights in a kernel over the entire image for multiple output neurons means that the network does not have to learn the same weights several times to represent the same output. Another advantage of this is that a movement (translation) of an object in the image will result in the same output after the convolutional layer, but with a translation factor.

The convolutional layers can be viewed as filters for extracting features, while a fully connected layer is often used afterward for mapping the features to an output.

3.4 General architecture for modern object detectors

Modern object detectors based on CNNs are composed of several modules. The first part is a **backbone** which is responsible for feature extraction and consists of convolutional layers. Next, most detectors use a **neck** such as Feature Pyramid Network (FPN) to improve scale invariance.

FPN as introduced in [Lin et al., 2017b] is an architecture that exploits the different scales from the bottom-up convolutional feature-pyramid in the backbone and combine multiple levels with an inverse pyramid consisting of upsampled top-down layers. The upsampling of the layers in the top-down pyramid results in spatially coarser, but semantically stronger (deeper) features for the lower layers and when combined with the bottom-up pyramid spatial resolution is improved. Prediction are made at each level which results in better scale invariance.

Finally, a **head** composed of a box predictor and a classifier is implemented in order to obtain the final prediction: bounding boxes and class labels.

Figure 3.2 shows neural network consisting of a backbone, neck and head.

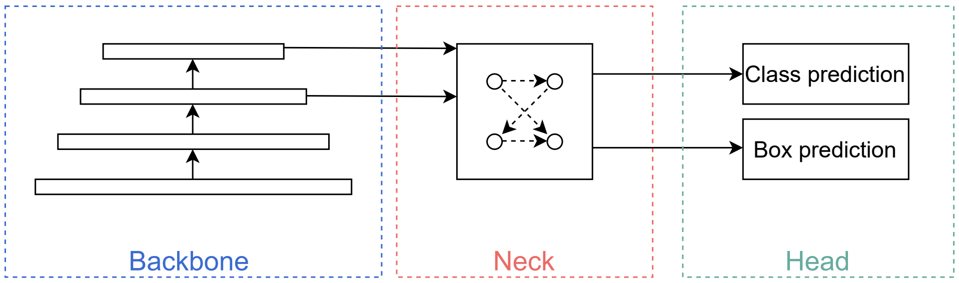


Figure 3.2: General architecture of a modern object detector with backbone, neck and head.

Chapter 4

Theory: Evaluation metrics

The literature uses a common set of evaluation metrics when comparing the performance of object detection models using bounding boxes. These metrics descend from detection challenges for two large annotated datasets with multiple classes known as PASCAL VOC [Everingham et al., 2010] and Microsoft Common Objects in Context (COCO) [Lin et al., 2015]. The latter is described in more detail in section 6.1.2. As these challenges are made to find the best performing model for object detection, the evaluation metrics must be commonly defined for comparison. In this chapter, the metrics are defined and described. Finally, in the result chapter of this thesis, they will be applied to analyze the results.

As the metrics IoU and precision and recall were also used in the analysis in the author's project thesis[Kjønås, 2021], the sections describing these values are similar. However they are edited and adapted to this master thesis. Section 4.3 is new.

4.1 Intersection over Union (IoU)

In order to quantify how well a predicted bounding box matches a ground truth bounding box, Intersection over Union (IoU) is calculated as follows:

$$IoU = \frac{\text{Area of intersection}}{\text{Area of union}} = \frac{B_p \cap B_{gt}}{B_p \cup B_{gt}} \tag{4.1}$$

where B_p is the predicted bounding box, and B_{gt} is the Ground Truth (GT) annotated bounding box.

Figure 4.1 illustrates the area of intersection and the area of union.

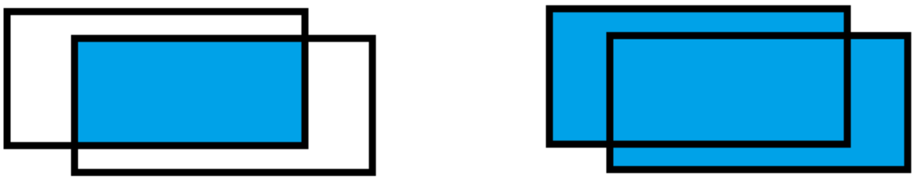


Figure 4.1: Area of intersection to the left and area of union to the right

A detection is defined as correct if the IoU of a predicted bounding box and the GT bounding box is above a given threshold value, T_{IoU}.

4.2 Precision and recall

Before we define precision and recall, we must define some other metrics related to correct detection, false alarms and misdetections.

The number of True Positive (TP) is the number of correctly detected objects, i.e. where the $IoU > T_{IoU}$ for a predicted bounding box and a GT bounding box.

The number of False Positive (FP) is the number of predicted bounding boxes from the model that are classified as boats, but not present in the GT annotations, also known as false alarms.

The number of False Negative (FN) is the number of boats that are present in the image and thus has GT bounding boxes, but they are not detected by the model, also known as misdetections.

Based on this, precision and recall are defined as follows.

$$precision = \frac{TP}{TP + FP} \tag{4.2}$$

$$recall = \frac{TP}{TP + FN} \tag{4.3}$$

Note that recall corresponds to the detection probability P_D of targets. The trade off between precision and recall can be decided by the user by adjusting the confidence threshold for the bounding boxes. This results in a precision-recall curve similar to figure 4.2. The green curve is the traditional curve, while the blue one is interpolated where the interpolation step will be described in the following section.

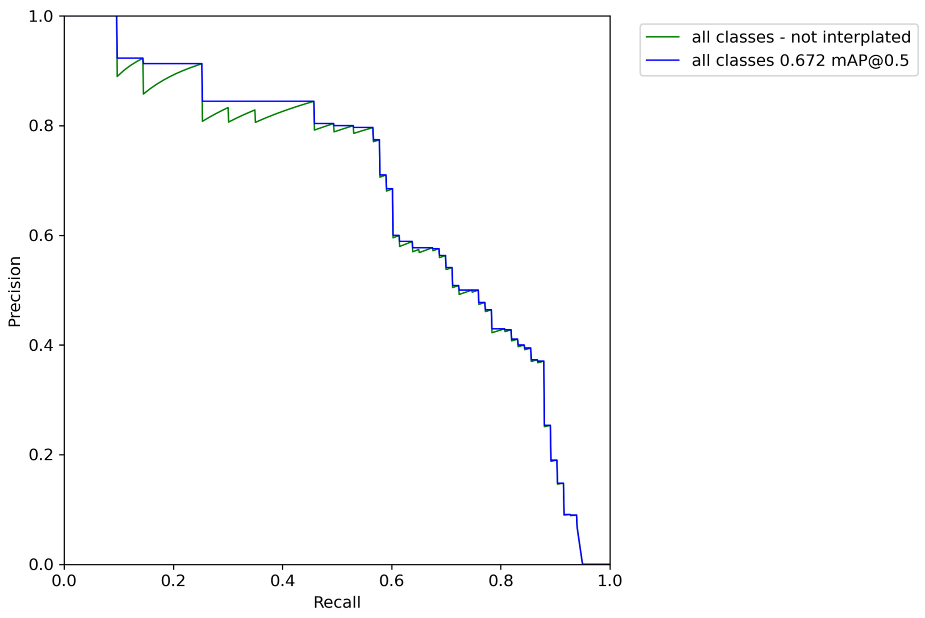


Figure 4.2: Example of a precision-recall curve in green and the corresponding interpolated precision-recall curve in blue.

4.3 Average Precision (AP)

A common evaluation metric for object detection is Average Precision (AP), used for comparison of models in [Everingham et al., 2010].

In general, AP is the area under the precision-recall curve. However, when used for calculation of AP the curve is interpolated.

$\{R_i\}$ is a set of recall values in the interval $[0, 1]$ with a given incremental step i so that we have a total of $\frac{1}{i} + 1$ recall values. For PASCAL VOC evaluation, the incremental step is 0.1 so that $\{R_{0.1}\} = \{0, 0.1, ..., 1\}$ [Everingham et al., 2010], while for COCO evaluation the incremental step i is 0.01 resulting in 101 recall values. As modern object detectors based on neural networks are evaluated on the COCO dataset, this evaluation method will be used in this thesis.

For each recall value $\tilde{r} \in \{R_i\}$ the interpolated precision $p_{interpolated}(\tilde{r})$ is found to be the maximum precision value for any recall value $\geq \tilde{r}$ [Everingham et al., 2010]:

$$p_{interpolated}(r) = \max_{\tilde{r}:\tilde{r}\geq r} p(\tilde{r}) \tag{4.4}$$

We can then find the AP with the following equation [Everingham et al., 2010]:

$$AP = \frac{1}{(1/i)+1} \sum_{r \in \{R_i\}} p_{interpolated}(r) \tag{4.5}$$

Graphically the AP corresponds to the area under the interpolated precision-recall curve in figure 4.2.

The reason for using the interpolated curve when calculating AP is to reduce the impact of small variations in the ranking [Everingham et al., 2010].

AP can be calculated for each class or as the average of all classes. The latter is often referred to as mean Average Precision (mAP). However, in evaluation of COCO no distinction is made between AP and mAP and thus the same notation is used in this thesis.

4.3.1 AP$_{\mathbf{IoU}}$

Furthermore, for COCO the AP is averaged over a set of IoU values between $[0.5, 0.95]$ with an incremental step of 0.05 which corresponds to the notation $AP@[.50:.05:.95]$. PASCAL only uses a single threshold of $IoU \geq 0.5$. In addition to $AP@[.50:.05:.95]$, which hereafter will be referred to as AP, the metrics $AP@[.50] = AP_{.50}$ and $AP@[.75] = AP_{.75}$ are often presented as well, where a single threshold of $IoU \geq 0.5$ and 0.75, respectively, are used. These metrics give an indication of how accurate the bounding boxes are by showing the difference when the IoU-threshold is increased. The $AP_{.50}$ for the precision-recall example curve in figure 4.2 is given in the legend.

4.3.2 AP$_{\mathbf{pixel\ area}}$

In order to evaluate how well the model detect objects dependent on their spatial extent, one can differentiate between pixel area sizes. AP_{small} is the AP for objects with $area < 32^2$ pixels, AP_{med} is for $32^2 < area < 96^2$ pixels, and AP_{large} is AP for objects with $area > 96^2$ pixels.

4.3.3 Average Recall (AR)

Average Recall (AR) is another evaluation metric that measures the object detectors performance on the ground truth annotated objects present in the input images. FPs are not considered for this metric which is calculated as the integral of recall values r over different IoU thresholds t_{IoU} [Padilla et al., 2021]:

$$AR = 2 \int_{0.5}^{1} r(t_{IoU}) dt_{IoU} \tag{4.6}$$

Similarily as for AP, AR is averaged over all classes and it is possible to differentiate between pixel area sizes as AR_{small}, AR_{med} and AR_{large}.

Chapter 5

Literature Review

5.1 Object detection in the maritime domain

Object detection is a computer vision method for images with the goal of recognizing and locating instances from a predefined set of classes such as *person* and *boat*.

In the maritime domain, videos from Electro-Optical (EO) cameras have traditionally been used for surveillance. During the last few years such cameras have shown to be useful in complementing radar and other sensors for situational awareness at sea through object detection of relevant boats [Prasad et al., 2017]. With the development of autonomous ships including systems for collision avoidance and navigation, maritime object detection and tracking methods are essential.

In [Prasad et al., 2017], the authors present and compare approaches to maritime object detection and tracking in videos using EO-sensors in the infrared and visible light range. Several approaches to background subtraction are presented and tested on a maritime dataset. The main challenge specific to the maritime domain when using background subtraction methods is identified to be the dynamic water from waves and wakes. This results in poor performance from static methods. Dynamic background approaches show a significant improvement, but are still challenged by dynamic movements at sea. In addition, abrupt changes in weather and illumination and unexpected events may cause problems for these methods. Finally, it is concluded that exploring state-of-the-art background modelling techniques from the general computer vision community such as CNNs may be rewarding in the maritime domain.

It is also worth noting that the current literature in maritime background subtraction almost exclusively deals with the case of open seas as opposed to urban harbour areas, according to the survey.

5.1.1 Related work using IR-images

Object detection using LWIR images in the maritime domain has been and tested with promising results in [Schöller et al., 2019]. The authors compared three neural network models trained on 21322 LWIR images: RetinaNet [Lin et al., 2017a], YOLOv3 [Redmon and Farhadi, 2018] and Faster R-CNN [Ren et al., 2015]. Faster R-CNN was the best performing model, but with a slow inference time which was found to be 5 times that of YOLOv3, and approximately 10 times that of RetinaNet. Thus, the authors argue that this might not be ideal for a real-time object detector in the maritime domain as a part of a tracking algorithm. YOLOv3 had better recall and precision than the original RetinaNet. They emphasized that recall is the most important evaluation metric when used for tracking, and conclude that given their recall results neither model could be used for stand-alone detection in an navigation setting, but complement the use of other sensors.

[Helgesen et al., 2019] used the neural network model SSD [Liu et al., 2016] trained on a total of 2035 NIR maritime images. The detection results were showed to be very good at target distances up to 400 m.

In [Hølland, 2019], the same neural network SSD was trained on LWIR maritime images. The object detector showed promising validation performance during training. However the model was trained and tested on the same data with the only exception being data augmentation. This results in potentially misleading detection performance and thus the need for more annotated IR images for both training and testing was addressed by the author.

5.1.2 Related work using visible light images

In [Grini, 2019], the author Grini compared object detection results from training the neural networks SSD and YOLOv3 on a collected dataset of 1916 visible light maritime images. YOLOv3 was the better performing detection model on the author's dataset. An identified problem was frequent false detections of buildings that were misclassified as boats. The author addressed this by testing to train on a separate building class and investigate whether this could improve the problem of false boat-detection of buildings. Although this seemed to improve this issue somewhat in the training and validation datasets, when tested on a video collected by Kamsvåg in [Kamsvåg, 2018], including this class actually gave a higher misclassification rate of buildings.

Landsnes trained Faster R-CNN with FPN [Lin et al., 2017b] and Mask R-CNN [He et al., 2018] on Grini's collected dataset merged with another dataset giving a total of 2520 visible light maritime images with good results in [Landsnes, 2021]. Some interesting observations are that the results were improved when including the data augmentation methods horizontal flipping and random cropping, while rotation degraded the results. In addition, the author recommends using a unique dataset for testing, as random splitting from a merged dataset can result in sample leakage.

5.1.3 Spatio-temporal object detection for videos

As the application of the object detection is tracking in video, temporal features may also be exploited, and ideally in combination with spatial features. Background subtraction, as thoroughly summarized within the maritime field in [Prasad et al., 2017], is a group of such methods. The main idea is to model the background and detect objects based on comparison with and signal processing of incoming frames.

These methods are especially interesting for stationary cameras such as shore mounted ones. This was investigated in the author's specialization project [Kjønås, 2021] with the use of a simple temporal Gaussian approach to background subtraction, including Markov random fields for spatial filtering. The problems highlighted in [Prasad et al., 2017] such as wakes and abrupt changes from camera noise were symptomatic for this solution. However more robust solutions including subspace learning such as Principal Component Analysis (PCA) might be of interest for future research. Subspace learning in general are methods where blocks of the video is considered as matrices and background modeling features are represented more compactly through matrix decomposition. These subspace features can be learned and updated in an efficient manner and used for object detection [Prasad et al., 2017].

In the general computer vision field, CNNs are the leading research topic as they have shown superior performance on available image datasets used for evaluation such as the COCO dataset. In the extension of this, [Zhu et al., 2020] presents a survey of available datasets, metrics and methods for video object detection with a main focus on deep learning approaches as they have shown to be more effective. Such approaches would require several annotated videos as training data and are therefore not considered in this thesis with limited available data.

Furthermore, the object detection in the IR video is supposed to contribute to a tracking algorithm processing data from several sensors. Thus temporal filtering will be performed through this algorithm, hopefully resulting in more stable detection results.

5.2 Object detection using neural networks

As previously mentioned, the use of neural networks for object detection in images is a modern and successful method of great popularity in several application domains including face recognition and autonomous driving [Zhao et al., 2018]. It is also a popular approach in the maritime domain, and will thus be the focus of this thesis. The following section aims to give insight into well performing CNN models for real-time application in object detection.

5.2.1 Models

In this section we will present and compare state-of-the-art object detection models and some of the most frequently used in the literature.

In table 5.1 some selected models are presented with metrics comparing accuracy and efficiency. Average Precision (AP), or equivalently $mAP@[0.5 : 0.05 : 0.95]$, is the main parameter used to indicate the accuracy of a given model as it is used for COCO evaluation [Lin et al., 2015]. The meaning of this metric is described in section 4.3. In the table the AP is the measured results from evaluating on the COCO *test-dev* subset from the COCO dataset [Lin et al., 2015].

Frames Per Second (FPS) is the inference time of the given model on a single image with the input resolution given as *size*. This parameter thus indicates the efficiency of the model. The inference time is highly dependent on the machine on which the inference is running, therefore the GPU used is indicated in parentheses, all from NVIDIA[1]. According to [Redmon and Farhadi, 2018] NVIDIA Titan X and NVIDIA Tesla M40 are very similar GPUs and their times are thus comparable, while NVIDIA V100 is a newer and faster GPU so the models where this GPU is used to measure FPS are separated into another group.

Model	*AP*	*FPS*	*Size*
SSD512* (VGG) [Liu et al., 2016]	28.8	19 (Titan X)	512 × 512
Faster R-CNN with FPN [Lin et al., 2017b]	36.2	5.8 (M40)	800
Mask R-CNN [He et al., 2018]	**39.8**	5.1 (M40)	800
YOLOv3 [Redmon and Farhadi, 2018]	33.0	**19.6** (Titan X) 73 (V100)	608 × 608
YOLOv4-CSP [Wang et al., 2021]	46.2	93 (V100)	512 × 512
YOLOv4-CSP [Wang et al., 2021]	47.5	70 (V100)	640 × 640
YOLOv4-P5 [Wang et al., 2021]	51.8	41 (V100)	896 × 896
YOLOv4-P6 [Wang et al., 2021]	**54.5**	30 (V100)	1280 × 1280
EfficientDet-D0 [Tan et al., 2020]	34.6	**98** (V100)	512 × 512
EfficientDet-D1 [Tan et al., 2020]	40.5	74 (V100)	640 × 640
EfficientDet-D2 [Tan et al., 2020]	43.9	56.4 (V100)	768 × 768
EfficientDet-D3 [Tan et al., 2020]	47.2	34.5 (V100)	896 × 896

Table 5.1: Comparison of object detectors with real-time application in terms of AP, FPS and size. AP measured on *test-dev* COCO test-set [Lin et al., 2015]. The best performance in each group is marked in bold

[1] https://www.nvidia.com/

SSD [Liu et al., 2016], Faster R-CNN with FPN [Lin et al., 2017b], Mask R-CNN [He et al., 2018] and YOLOv3 [Redmon and Farhadi, 2018] are included in the table because they are important contributions to modern object detectors and frequently used in the literature in the maritime domain [Helgesen et al., 2019, Schöller et al., 2019, Hølland, 2019, Grini, 2019], while EfficientDet [Tan et al., 2020] and Scaled-YOLOv4 [Wang et al., 2021] are current state-of-the-art object detectors with real-time application. The last two models are scalable and only the versions that satisfy the requirement of FPS $\geq$ 30 are included, as this is a common threshold for defining real-time [Bochkovskiy et al., 2020]. In addition 9 FPS is the framerate of the IR-camera used in this thesis, and it is assumed that the final hardware for processing might not be as powerful as NVIDIA V100 and thus we need some margin.

For object detection using CNNs, the models can generally be classified into two groups. The first, one-stage detectors, are grid based and will be further explained in the following section. The other group is called two-stage detectors, which use a Regional Proposal Network (RPN) to predict a set of bounding boxes and another network to decide whether or not there is an object in each box, fine-tune the proposals and classify the objects.

One-stage detectors are in general known to be faster, but at the cost of worse precision than the two-stage detectors [Jiao et al., 2019] . Examples of two-stage detectors are R-CNN based models such as Faster R-CNN [Ren et al., 2015] and Mask R-CNN [He et al., 2018]. As real-time application is desired in this project, the focus will be on one-stage detectors.

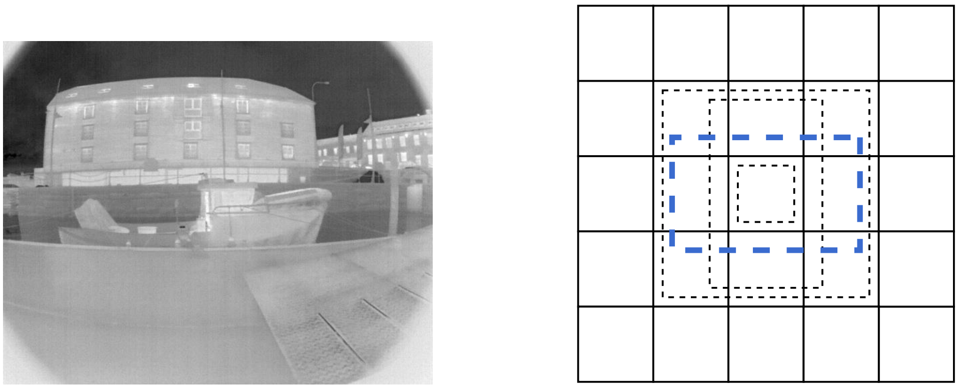
SSD

Single Shot MultiBox Detector (SSD) was first introduced in [Liu et al., 2016] as a one-stage detector out-performing its predecessor YOLO, which will be presented in following sections.

A fixed sized grid divides an image into cells. Each grid-cell produces a set of prior bounding boxes. In SSD the prior is decided by a set of manually selected aspect ratios, where these aspect ratios are the ratio between the width and height and are combined with different scaling factors to produce the prior boxes. The predefined total number of predictions is then $N \times N \times B$ where N is the grid size and B is the number of boxes for each cell defined by the number of scaling factors and aspect ratios.

The cell located in the center of a ground truth bbox is responsible for the detection of the given object. For each cell, the dimensions of the boxes in terms of offset from prior is predicted together with the class predictions, including a background class suggesting that there is no object in the box.

Figure 5.1 shows an example of the grid-based feature maps used for box predictions. One grid size is shown where prior boxes are displayed for the center cells of the two ground truth objects. The location and size offset is predicted for each prior box together with a confidence score for each predefined class.



(a) An example image with one boat present

(b) 5×5 feature map with pre-defined anchor boxes. The closest box is highlighted in blue. Note that it still needs adjustments to fit correctly

Figure 5.1: SSD box prediction
Figure inspired by [Liu et al., 2016]

The box predictions and classifications are based on features from differently scaled convolutional layers, without the top-down pyramid layers from the FPN. The features are extracted by the backbone network. The original paper uses VGG-16 as backbone [Liu et al., 2016], while both Helgesen [Helgesen et al., 2019] and Grini [Grini, 2019] used MobilenetV2 [Sandler et al., 2018], which is a faster backbone network with lower accuracy.

Finally, non-max suppression is applied to the boxes in each cell with the same class prediction and IoU above a threshold value to keep only the highest confidence box.

YOLO

You Only Look Once (YOLO) was the first one-stage object detector and introduced in [Redmon et al., 2015]. This model is quite similar to SSD, but one important difference is that it has no background class and instead uses a probability of there being an object present for each bounding box in order to filter out non-objects, thus keeping class predictions separated. In addition, the bounding boxes were predicted directly without the use of priors.

After the publication of YOLO in [Redmon et al., 2015], it has been improved in several versions. In this thesis, we will focus on YOLOv3 [Redmon and Farhadi, 2018] because it is a frequently used model for real-time object detection and in the maritime domain [Schöller et al., 2019][Grini, 2019], and present scaled-YOLOv4 [Wang et al., 2021] which at the point of the publication in November 2020 set a new state-of-the-art performance in terms of AP on the COCO-dataset while also being scaled to real-time frame-rates.

YOLOv3

YOLOv3 [Redmon and Farhadi, 2018] references an incremental improvement showing that the methodology is based on the framework from YOLO, but the model is improved: A drawback with the first version was that only one class could be predicted per cell as can be seen in figure **??**, while YOLOv3 allows multiple classes per cell for overlapping objects. In YOLOv3, the offset compared to prior bboxes are predicted, similarly to SSD using anchor boxes based on aspect ratios. The author proposes that the best ratios are determined through k-means clustering of the training data. By including predictions on multiple scale levels similarly to an FPN, YOLOv3 outperforms its predecessors in terms of detecting smaller objects.

The backbone is also improved: Darknet-53 is based on the original convolutional network Darknet from YOLO but with more layers to improve accuracy. In addition a residual mapping is added to the convolutional layers to avoid degradation, inspired by [He et al., 2015] who shows that the residuals are easier to optimize, especially for deeper networks.

Scaled-YOLOv4

Scaled-YOLOv4 [Wang et al., 2021] is a further improvement of YOLOv4 [Bochkovskiy et al., 2020]. YOLOv4 uses a CSPDarknet53 backbone, Spatial Pyramid Pooling (SPP) and Path Aggregation Network (PANet) in the neck and the same head as YOLOv3. SPP increases the receptive field, i.e. the size of the input region where the convolutional layer's feature is affected. PANet is an alternative to FPN by detecting and combining features from different layers.

In addition, YOLOv4 introduces two concepts for enhancing performance: *Bag of freebies* are methods that does not affect inference time, but can increase training time and change training strategies such as data augmentation. *Bag of specials* are methods that improves the model at a low cost and examples in YOLOv4 are SPP blocks and PANet in the neck. For full description of all the methods in the two bags, the reader is referred to the original paper [Bochkovskiy et al., 2020].

Scaled-YOLOv4 [Wang et al., 2021] further improves the model by using Cross Stage Partial Network (CSPNet) to reduce number of parameters and computations while improving accuracy in the backbone and neck. CSPNet as introduced in [Wang et al., 2019] divides the output signal of a base layer, e.g. a feature map from the first convolutional layer in the backbone, into two parts: Half of the signal follows the main path of the network. This results in more semantic information. The other half of the signal is bypassed and combined with the other part in a transitional layer. This preserves more spatial information.

Finally, it was found in [Wang et al., 2021] that compound scaling, a concept introduced in [Tan et al., 2020] and explained in the next section, of input size and number of stages in backbone and neck gave best performance. Then depth and width was dynamically scaled according to real-time requirements.

EfficientDet

EfficientDet [Tan et al., 2020] is a modern one-stage object detector. The backbone of EfficientDet is called EfficientNet [Tan and Le, 2019], and this is combined with a BiFPN for more complex feature combination at different scales and a shared box and class prediction network.

The EfficientNet [Tan and Le, 2019] backbone has contributed with the methodology for scaling of the backbone of an object detection model. This allows the user to choose the trade-off between *accuracy* and *efficiency* based on application specific constraints for inference time.

A convolutional network consists of several layers which can devide the neurons in different manners. Parameters determening this related to the depth and size of the model must be decided.

- Number of channels N in the kernel and output, determining the **width** of the model
- Number of layers, determining the **depth** of the model
- Input size $H \times W$, determining the **resolution** of the model

Note that the same notation as in section 3.3 is used. A visualization of scaling of the different parameters can be seen in figure 5.2.

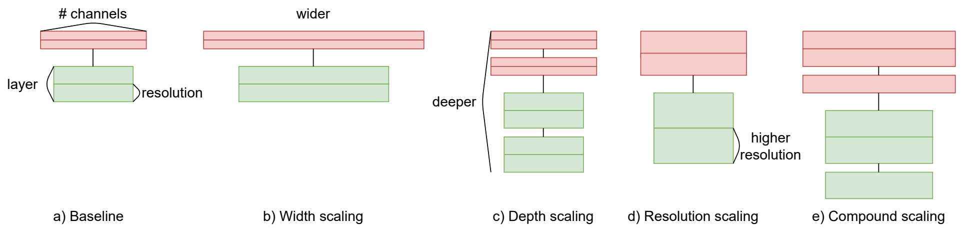


Figure 5.2: Scaling of convolutional network compared to a baseline. (a) is a baseline network with the tree parameters indicated. (b) shows width scaling. (c) shows depth scaling. (d) shows resolution scaling. (e) is compound scaling used in EfficientNet.
Figure inspired by [Tan and Le, 2019].

In EfficientNet, compound scaling is used. This methods consists of finding a balance between the scaling parameters through an architecture search.l Then model scaling is performed by scaling each parameter with a constant ratio, meaning that when we increase the model size, depth, width and resolution are all scaled uniformly.

EfficientDet [Tan et al., 2020] contributes to the model with the method that is used to extract features in the different layers are combine them in order to make detections. A Bidirectional Feature Pyramid Network (BiFPN) is used, which fuses information from low-level features with high-level ones just as much as the opposite way around. The BiFPN layer is repeated multiple times in the neck dependent on the scaling factor for more complex fusion of features.

The final network is then provided with 9 different scaling parameters, D0-D7 and D7x, each providing different trade-offs between efficiency and accuracy.

The choice of models to investigate in this thesis is presented in section 7.

5.3 Data augmentation

Data augmentation is a set of techniques that introduces manipulated copies of existing data in order to increase the total amount of data [Shorten and Khoshgoftaar, 2019]. It is applied to the training images in order to increase the training set size.

There are two main motivations for using data augmentation. The first is generalizability, which refers to the comparison of performance for a model when evaluated on validation data that is previously seen and test data that is unseen [Shorten and Khoshgoftaar, 2019]. A common problem with neural network is overfitting to the training data, the opposite of generalizability. By applying data augmentation techniques, one can introduce more variations to the collected data and thus use more data points for training of the model. Because of this, data augmentation is a powerful method for avoiding overfitting.

The second advantage, which is highly related to avoiding overfitting, is that neural network rely on big data in order to learn features and detect objects while in many cases the available annotated data is limited. For the application of training neural networks on IR-images in the maritime domain, approximately 2000 images were used for training in [Helgesen et al., 2019] and 20000 in [Schöller et al., 2019]. For this project we have a total annotated training and validation dataset of 261 images, which constitutes a considerably smaller dataset. This motivates exploring the effect of data augmentation in this thesis.

[Shorten and Khoshgoftaar, 2019] is a survey that covers data augmentation techniques for images used in deep learning. They present an overview of existing methods from simple geometrical and color transformation to advanced deep learning based Generative Adversarial Network (GAN). The survey states that not many studies are performed on comparison of different augmentation techniques, but cite [Taylor and Nitschke, 2017] that compares some frequently used geometric and color transformations and find that cropping gives best detection performance. In addition [Shijie et al., 2017] also compares geometric and color transformations and include GANs, Wasserstein GANs and combinations of different techniques. They found that cropping, flipping, Wasserstein GAN, and rotation generally performed better than others. Time was not found in this thesis to explore the more advanced methods such as Wasserstein GANs. Instead, we focus on geometrical transformations, as they are easier to implement and are highlighted in the augmentation survey.

Another topic discussed in [Shorten and Khoshgoftaar, 2019] is the safety of an augmentation. This refers to the likelihood that the annotation is correct after the transformation. For instance, in text recognition tasks, rotation of the letter ”n” results in the letter ”u” and consequently a wrong label. Augmentation methods must thus be useful for the general task, which in this case is object detection in terms of bounding boxes and classes, and domain, where questions such as ”Should the object detector consider a boat upside down?” arise.

Finally, it is worth noting that it is not possible to solve all problems related to biases in a small dataset through data augmentation. It cannot help the detector learn kayaks if there are no kayaks present in the training data.

5.3.1 Online and offline augmentation

In an offline augmentation scheme, the images are augmented before training and stored separately. Then the same set of training images is shown each epoch. This method takes up a larger storage space, but makes training faster and the augmented images can easily be verified.

With online augmentation, the dataset varies each epoch, as the augmentation is performed during training. If the augmentations are too strong, they can potentially deteriorate the images relative to the ground truth, and it is difficult to verify this. Thus, safety is especially important when using online augmentation.

5.3.2 Geometrical augmentation methods

Geometrical augmentations are methods that affects the geometrical properties of an image, i.e. the location of the pixels. The motivation for using geometrical augmentation is to avoid overfitting and overcome positional biases. An example of a positional bias is if the objects of interest in the training set are located in the bottom left corner of the image in almost all images, the neural network might learn that the probability of detecting this object is greater in this specific area of the image. However, the applied detector should be able to detect boats regardless of where in the image they are located.

The augmentation techniques investigated in this thesis will be presented in section 7.2.3.

Chapter 6

Method: Datasets and collection of more data

Data is extremely important in machine learning, thus this chapter aims to present the datasets used for training, validation and testing of the object detector.

6.1 Datasets used for pre-training

6.1.1 ImageNet

[Russakovsky et al., 2015] introduced ImageNet as a benchmark dataset for image classification: the task of identifying whether there are objects of pre-defined classes present in the image or not. There are a total of over 14 million images containing objects from more that 1000 classes [Russakovsky et al., 2015]. This dataset is typically used for pre-training of the backbone in order to learn feature extraction. When the backbone is used for object detection such that the instances must also be localized with bounding boxes, the last soft-max layer is removed and the neck and head is then connected to the backbone before further training on other datasets.

6.1.2 COCO-dataset

Microsoft Common Objects in Context (COCO) is a large annotated dataset containing complex images from everyday scenes with common objects in their natural context [Lin et al., 2015]. There are 2.5 million annotations in a total of 328k images, which are divided into a trainval set and a testdev set. There are 80 object classes of which each instance is annotated with a segmentation mask and class label. Examples of this can be seen in figure 6.1. Note that one of the classes is *"boat"* which means that when a network is pre-trained on this dataset we can detect boats in the images.

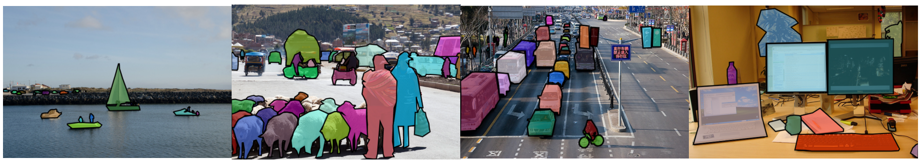


Figure 6.1: *Images from [Lin et al., 2015]*
Example images from the COCO-dataset with instance segmentation masks

6.2 Available LWIR datasets

6.2.1 Hurtigruten

The images in the Hurtigruten dataset are collected and annotated by Michael Ernesto Lopez and Edmund Brekke in June 2019. The complete dataset consists of both IR and visual spectrum images, but only the IR images are considered here. The images are collected along the Norwegian coast from onboard the cruise ship *Hurtigruten* with a LWIR camera.

The IR images are of size 512×640 $(H \times W)$ and there are a total of 313 images. The annotations are made as bounding boxes with two different detail levels of the class-labels: fine-detailed and simplified. There are a total of 21 fine-detailed classes which represent different types of boats. Noting that some of these classes are constituted of very few samples, it is decided to focus on the simplified labels of which there are 6, and they are summarized in table 6.1. As there are very few barges, this class is considered a part of "Motorboat with priority" hereafter.

Total number of images	**313**
Barge	4
Building	739
Motorboat	55
Motorboat with priority	342
Sailboat with sails down	29
Sailboat with sails up	3

Table 6.1: Overview of simplified class labels and the number of instances in Hurtigruten dataset

Examples of pictures with annotations from this dataset is shown in figure 6.2

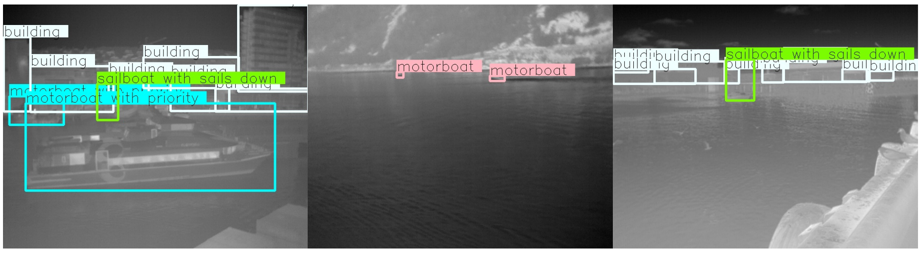
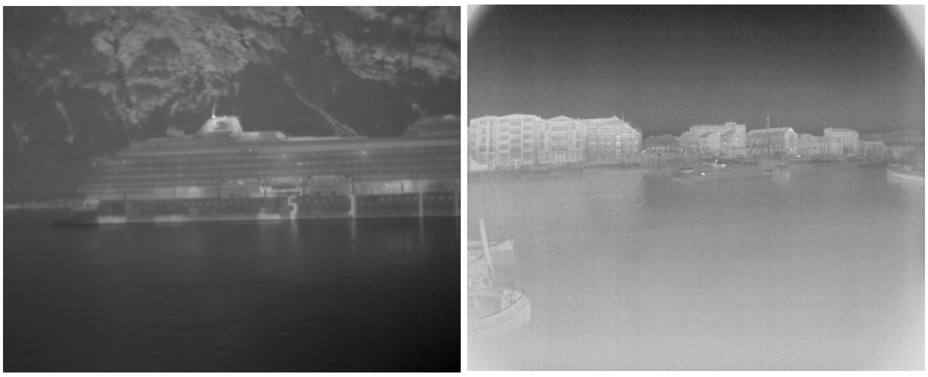


Figure 6.2: Example images from the Hurtigruten-dataset with bounding box annotations and class labels

Two of the most frequently used class labels from the detailed labels are cruise ship and cargo ship. These are both very large ships which are not very relevant in the small canal where milliAmpere will be operating, and in the infrared images they can resemble buildings as shown in figure 6.3.



(a) Picture of cruise ship from Hurtigruten dataset. **(b)** Picture from Fosenkaia dataset in section 6.2.2

Figure 6.3: The ship in figure (a) can resemble the buildings on the left side of figure (b) due to windows and similar structure.

During early testing false detection of buildings which were classified as boats were frequently present. This motivated changing the dataset to remove data that potentially could make performance worse because of the similar appearance in the images. Thus, a subset of the dataset called Hurtigruten_small was created by excluding images of very large ships that are not relevant in the narrow canal in Trondheim, based on manual inspection of the images. For example, the image in figure 6.3a was removed. Hurtigruten_small is summarized in table 6.2

Separating cruise-ships from buildings may very well be feasible for the neural network, but it is assumed that it might require more data, as the current dataset is already quite small. In addition, it is not relevant for the particular case of harbour environments in a narrow canal in this thesis.

Total number of images	**167**
Building	368
Motorboat	42
Motorboat with priority	159
Sailboat with sails down	20
Sailboat with sails up	3

Table 6.2: Overview of simplified class labels and the number of instances in the subset Hurtigruten_small dataset

6.2.2 Fosenkaia

An important part of the author's specialization project [Kjønås, 2021] was to collect a dataset synchronized between several sensors, including an LWIR camera mounted on a sensor rig at Fosenkaia in Trondheim. One scenario consisting of 911 frames in a video has been annotated with bounding boxes. An example of an image can be seen in figure 6.3b. This dataset called Fosenkaia was originally planned to be used for evaluation and testing in this thesis. However, as the contrast is very low, a newly collected dataset with better contrast due to change in camera parameters is used instead. This will be further explained in the following sections.

Since the camera and thus the background is stationary, the Fosenkaia dataset is probably prone to overfitting if used for training. There is a risk that if a large part of the total training-data contains pictures with the same background and the same weather condition etc. the background will be learned instead of features of the objects of interest. This will surely give problems for the object detector that handles the IR-cameras onboard milliAmpere with a dynamic background. In addition, changes in the background, such as different weather conditions or additional docked boats may cause trouble for the stationary IR-camera object detection if the detector is overfitted.

For future interest in this sort of dataset, it is recommended to change the camera parameters as will be described in section 6.3.1 and record new data.

6.3 Collection of datasets

The motivation for collecting more data is that previous work with similar goals have used more data for training. For instance, Helgesen used 2035 IR-images in [Helgesen et al., 2019], Grini used 1916 images in [Grini, 2019]. Hurtigruten very small dataset by itself, and when some image are excluded in the Hurtigruten_small dataset the result is even less data. Therefore it is considered necessary to collect more data in order to get decent results from training the neural networks. In addition, there is an overweight of large ships and open-sea images in the Hurtigruten dataset, while not as many are from harbour environments. The sailboat-classes are also quite small.

Another goal when collecting more data is to investigate hypotheses from the author's supervisors regarding improvement of the contrast in the images. It is suggested that exclusion of the sky might help and potentially one can look at other camera settings for the IR-camera as well.

6.3.1 IR camera specifications

The camera used when collecting new data and for the Fosenkaia dataset from [Kjønås, 2021] is a Boson640 LWIR-camera. Table 6.3 presents an overview of the key specifications for the camera.

Name	Boson 640
Sensor Technology	Uncooled VOx Microbolometer
Resolution	640x512
Horizontal Field of View (HFOV)	95°
Effective Focal Length	$4.9\,\mathrm{mm}$
Spectral range	LWIR: $7.5\,\mu\mathrm{m}$ - $14\,\mu\mathrm{m}$
Frame rate	9 FPS

Table 6.3: IR-camera specifications from [FLIR Systems Inc., 2019a]. A similar table is presented in [Kjønås, 2021].

A full description of the functionality of the IR-camera is given in [FLIR Systems Inc., 2019b]. The sensors consists of a 2D array of vanadium-oxide (VOx) microbolometers. These all have a temperature dependent resistance that varies with incident radiation flux, and thus measures the thermal radiation of the surroundings. Each bolometer is applied with an internal bias, and the measured current signal is digitized into a measurement of width 14.2 bits (i.e. 2 digits after the decimal point and 16 bits in total). The signal is then pre-processed in terms of Non-uniformity Correction and Spatial/Temporal Filtering according to factory default settings given in [FLIR Systems Inc., 2019b].

Automatic Gain Correction (AGC) algorithm

The 14.2 bits values need to be converted into 8 bits because imaging system displays represent each pixel value (for a given channel) using 8 bits. In order to keep as much information from the signal as possible, an Automatic Gain Correction (AGC) algorithm is used to convert the data to 8-bit output signals [FLIR Commercial Systems, 2018]. Using a linear mapping or classical histogram equalization could result in low contrast for the foreground objects, as fairly uniform background parts such as the sky would actually consume a large portion of the dynamic range during mapping.

The AGC algorithm plots the input pixel values vs. number of pixels in a histogram. Optimizations adjusted by a set of parameters can change the calculation of the histogram. These parameters are briefly described in 6.4. Then the mapping function that transforms the 16-bit data to an 8-bit space is applied to the histogram.

Parameter	**Short description**	**Default value**
Information-Based Mode	Dynamic range is proportional to the amount of scene information in the pixel values. Ideal when the background is uniform, for instance the sky. A High-Pass and Low-Pass filter is used to extract details and background, and High-Pass content is weighted more.	On
Tail rejection	Sets a percent-wise lower and upper threshold in the histogram where all values outside will be mapped to min and max, respectively, leaving more bits to the central grey-shades.	0
Max Gain	Most effect in scenes with a narrow dynamic region. In such images increased value results in a better exploitation of the entire 8-bit spectrum, but risks overexposure and more grainy noise.	1.38
Damping Factor	Changes the update rate of the AGC algorithm between video frames.	85
Adaptive Contrast Enhancement	Adjust the perceived brightness of the image by prioritizing lower or upper part of histogram.	0.97
Plateau Value	When increased, the most frequent pixel values are prioritized, while these are otherwise clipped. FLIR recommends using the default value.	7
Linear Percent	How linear the mapping is. With a high value the relative temperature difference is more accurate.	20
Detail Headroom	Increases the amount of the 8-bit range that is dedicated to the Low-Pass filter data.	12
Digital Detail Enhancement	Increases the amount of the 8-bit range that is dedicated to the High-Pass filter data.	0.95
Smoothing Factor	Defines the cut off for the HP filter. FLIR recommends using the default value.	1250
Region of Interest	Decides which parts of the image that will be used for optimization of the AGC-algorithm.	0-639, 0-511

Table 6.4: AGC parameters with name, description and default value. Information from [FLIR Commercial Systems, 2018].

All these setting were tuned and tested by taking pictures with the same view of a boat in a relevant setting from Ravnkloa in Trondheim. The results were compared to the default values and the most interesting setting parameters are further investigated in the following sections. The most difficult part is considered to be to separate boats from background objects such as buildings. But improving the contrast between the targets and the sea is also important. Thus this is the focus when tuning the parameters. Parameters that do not have a significant effect or degrades the contrast are not further discussed, and the default values were used.

Region of Interest (ROI)

As stated in the application note [FLIR Commercial Systems, 2018], when the background is fairly uniform such as the sky or sea, it dominates the histogram, and is therefore allocated a large portion of the dynamic range. It would thus be preferable to exclude the sky from the histogram used in the AGC-mapping. However, as the vertical FOV is large (77° [FLIR Systems Inc., 2019b]) and the lens is convex, meaning that the corner parts of the image are distorted, it is difficult to exclude the sky from the images. This is especially difficult in the upper corners without using an unnatural angle on the pointing direction of the camera. Thus, a better method is to set the ROI for the AGC as described in [FLIR Systems Inc., 2019b]. This parameter lets you define the start and stop column and row which defines the region on which the AGC-algorithm is based when mapping the 16 bits to 8. This does not mean that the part that it excluded from the ROI is not displayed, only that it does not affect the AGC-optimization.

Figure 6.4 shows two images where one has the entire image as ROI, while in the other image the AGC-algorithm is based on the pixel values from row 120 (counted from the top) and down, meaning that most of the sky is excluded.

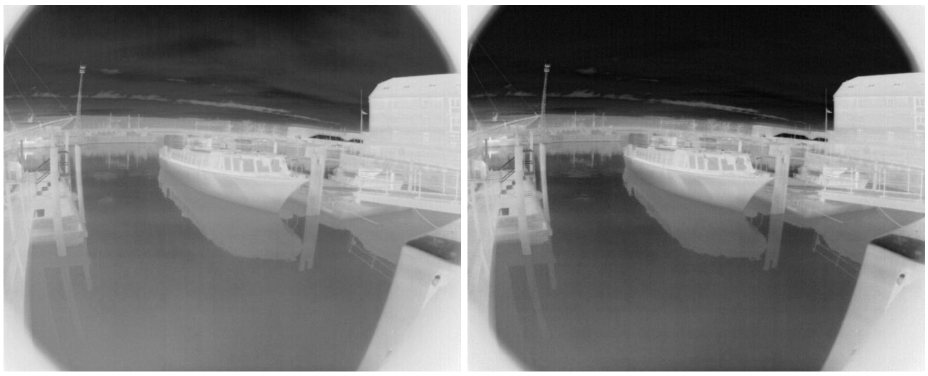

(a) ROI is the entire image **(b)** ROI is limited to row 120 and down

Figure 6.4: Comparison of effect of limiting ROI

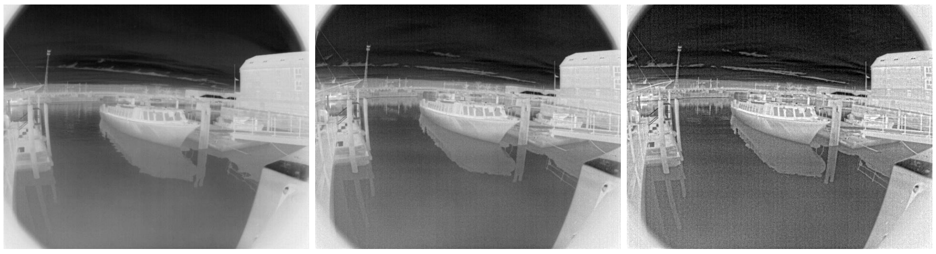
As can be seen from this figure, excluding the sky when using the AGC-algorithm by changing the ROI clearly affects the contrast in the image. Especially the contrast between the boat in the middle of the image and the sea is improved, but also details on foreground objects such as the pier are more visible.

For the shore mounted camera, the ROI limit can easily be determined, as the background with the sky region is stationary. However, for the cameras that will be mounted on board the autonomous vessel, the region containing sky in the images will vary as the ferry and thus the camera FOV is affected by wave movements. Excluding parts of the sky from the ROI also improves the contrast because it would be less dominant in the histogram, although not as much and it is thus recommended to limit the ROI for the on board mounted cameras as well. The exact row to set as start row for the ROI depends on the mounting angle of the IR cameras.

Digital Detail Enhancement (DDE)

Besides ROI, the most interesting parameter to adjust was found to be Digital Detail Enhancement (DDE), keeping in mind that the goal was to improve the contrast between the objects of interest from background objects such as buildings and piers.

DDE determines if the AGC-mapping should allocate more bits to the contents of the High-pass filter compared to the Low-pass filter [FLIR Commercial Systems, 2018]. If the parameter is turned up, the edges and transitions between objects are more detailed at the cost of more noise in the image. This is shown in figure 6.5.



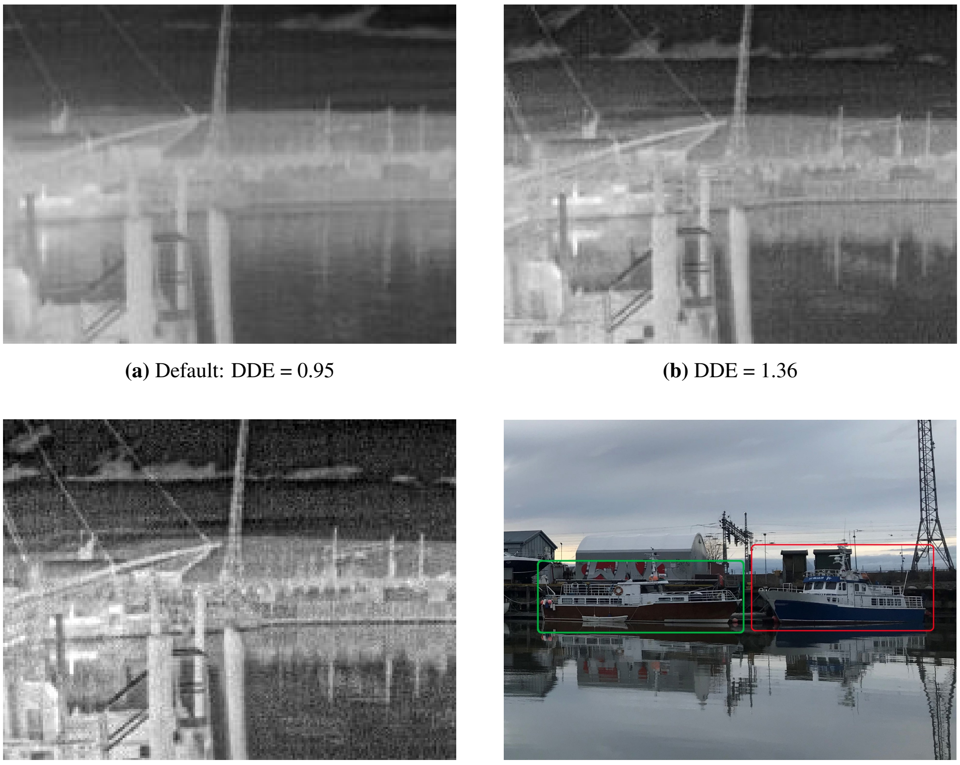
(a) Default: DDE = 0.95 **(b)** DDE = 1.36 **(c)** DDE = 2

Figure 6.5: Comparison of effect of DDE

The difference is clear: the details e.g. on the boat are more visible, but the image becomes more noisy. The middle boat is considered easy to detect, however this image contains some difficult background boats prone to low contrast. In figure 6.6, we zoom in on these boats to showcase the effect of DDE.

It seems like these boats are more visible when DDE is increased, indicating that the contrast is improved. Especially the boat marked in red is nearly impossible to observe with the default parameters, while slightly easier when DDE is increased.

There may be reasons to not set the DDE too high given possible disadvantages of a noisy image. The behaviour of the neural network on very noisy images is unknown. Thus it is considered safe to not use a value that is to high, but still increase the value in order to get the advantages of the contrast improvement.



(a) Default: DDE = 0.95

(b) DDE = 1.36

(c) DDE = 2

(d) RGB image of the same boats that are present in the IR-images, marked in green and red

Figure 6.6: Comparison of effect of DDE when zooming in

6.3.2 Nidelva

As the parameter testing was performed in parallel with collecting more data, the new settings were not applied before the third day of data collection, where DDE was set to 1.05 and ROI was limited to row 120 and down. However, weather conditions, distance to target and other factors may also affect the contrast, so for training and validation data all collected images are highly relevant.

The plan was to merge the new images with the Hurtigruten dataset, thus some areas to focus on when taking new pictures were identified based on manual inspection of the Hurtigruten dataset:

- Harbour environments
- Smaller boats (typical sizes in the canal)
- Kayaks if possible

- Different distances, especially close up as these are lacking for smaller boats in the Hurtigruten dataset
- More sailboats, because there are very few sailboat instances compared to motorboat (especially with priority) in Hurtigruten

The procedure for collecting data was to walk along the canal-area and different parts of Nidelva in Trondheim and take pictures of boats on the 18th, 20th and 23rd of March 2021. In order to ease the annotation process, visible light images were taken of the same objects as in the IR-images, so that they could be compared. The pictures were not directly overlapping, so it was only intended to be a visual aid.

Most of the pictures are taken of docked boats, but a few boats where passing by and depicted. Unfortunately, no kayaks was observed. These are considered crucial for the IR sensor to detect, so future work includes collecting images of kayaks. An overview of the collected images with the number of class instances are shown in table 6.5 for the new dataset hereafter called Nidelva.

Total number of images	**94**
Building	0
Motorboat	283
Motorboat with priority	1
Sailboat with sails down	135
Sailboat with sails up	0

Table 6.5: Overview of class labels and the number of instances in the Nidelva dataset

The same class labels as in the Hurtigruten dataset were used in order to easily merge the two datasets. Although this project focuses on bounding boxes, future work may include segmentation tasks, and it was thus decided to annotate the images with polygons of up to 20 points, which automatically generates bounding boxes as well. The annotation procedure for the collected images will be discussed in section 6.3.4. Some examples of annotated images from the Nidelva dataset are shown in figure 6.7.

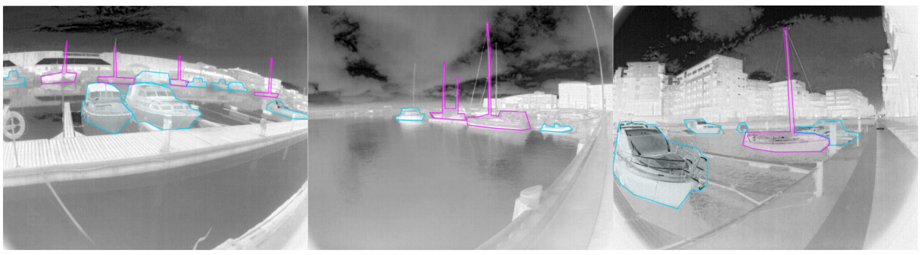


Figure 6.7: Example images from the collected Nidelva dataset with instance segmentations and class labels. Pink is ”sailboat with sails down”, blue is ”motorboat”

6.3.3 milliAmpere

Øystein Helgesen in cooperation with Kjetil Vasstein collected a new synchronized dataset from several sensors in the harbour area between Ravnkloa and Fosenkaia on the 5th of May 2021. This included videos from three IR-cameras taken on board milliAmpere. The new camera setting adjustments as explained in section 6.3.1 were used with DDE = 1.32 and ROI set to the lower 2/3 of the image. With these settings, Helgesen, who has collected several similar datasets previously, stated that the contrast in the IR-images with the new settings seemed much better than before.

A video from one of the three cameras is annotated with bounding boxes and class labels by the author according to the annotation rules in the next section. An overview of the number of class instances is shown in table 6.6.

Total number of images	**628**
Building	0
Motorboat	2723
Motorboat with priority	0
Sailboat with sails down	2490
Sailboat with sails up	0

Table 6.6: Overview of class labels and the number of instances in the milliAmpere dataset

The scenario in the video is one boat approaching the camera before passing and exiting the frame, then a new motorboat enters the frame from the opposite side. In addition, several sailboats and motorboats are docked on each side of the canal. The detection performance on the two moving boats will be interesting to analyze separately as they will give a clear indication of how the pixel size and thus the distance from the camera affects performance. Example images with annotations are shown in figure 6.8.

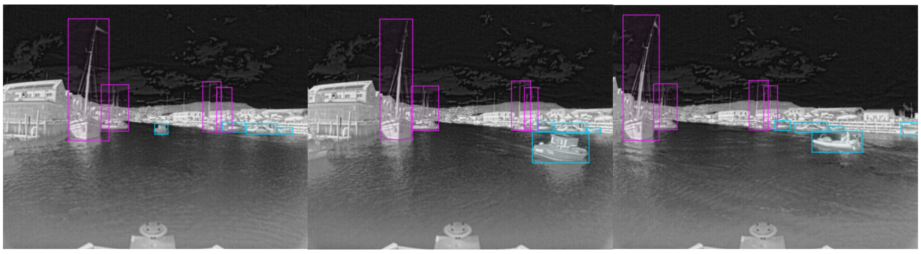


Figure 6.8: Example images from the collected milliAmpere dataset with bounding box annotations and class labels. Pink is ”sailboat with sails down”, blue is ”motorboat”.

The two sailboats on the leftmost part of the images in figure 6.8 in addition to the two moving boats are marked as *not difficult* when the view is reasonable. The other boats in the background are marked as *difficult*. This distinction will be used in the analysis of the results.

6.3.4 Annotation of datasets

Computer Vision Annotation Tool (CVAT) [1] was used to annotate the collected images. This includes a tracking functionality that is very useful when annotation videos.

Two concerns were made during annotation of these datasets:

- General guidelines for object detection
- Consistency with the Hurtigruten-dataset annotations

The Pascal VOC guidelines as described in [Everingham et al., 2010] and recited here were used as a basis for the annotation, where the relevant points are:

What to label: All objects of the defined categories, unless:

- you are unsure what the object is
- the object is very small (at your discretion)
- less than 10-20% of the object is visible.

Bounding box: Mark the bounding box of the visible area of the object (not the estimated total extent of the object). Bounding box should contain all visible pixels, except where the bounding box would have to be made excessively large to include a few additional pixels ($< 5\%$) e.g. a car aerial.

Truncation: If more than 15-20% of the object lies outside the bounding box mark as Truncated. The flag indicates that the bounding box does not cover the total extent of the object.

Difficult: Labeled objects which are particularly difficult to detect due to small size, illumination, image quality or the need to use significant contextual information. In the challenge evaluation, such objects are discarded, although no penalty is incurred for detecting them. The aim of this annotation is to maintain a reasonable level of difficulty while not contaminating the evaluation with many near unrecognisable examples.

Furthermore, it is important to be consistent across multiple images. For instance, background boats with the same contrast and distance must always be labeled if one is labeled.

The second consideration is consistency with the annotation of the Hurtigruten dataset. An important change from the guidelines is that even though the masts on sailboats with sails down often constitute less than 5% of the object, they are included in the bounding box annotation. This is done in the Hurtigruten dataset. For differentiating between sailboats and motorboats the mast is an essential feature. Therefore, the masts are also included in the annotations of the sailboats.

[1] `https://cvat.org/`

The same class labels as the simplified version in the Hurtigruten dataset are used for annotation of the collected data, except the class ”barge”, which is merged with ”motorboat with priority”.

Furthermore, it was decided to not annotate the buildings in the collected datasets. The main reason for this is that it is a time consuming task and difficult to do correctly. Buildings are often situated next to each other, making it difficult to decide how to separate their bounding boxes. In addition they are often obscured by other foreground objects. In the harbour area, there are a lot of buildings, including low contrast background buildings which are very difficult to annotate. Grini examined the hypothesis of whether including a boat class would improve boat detection performance by giving fewer false alarms from buildings in [Grini, 2019]. It was found that in harbour environments, the inclusion of a boat class had little effect on the detection performance, while in open sea environments, including a building class helped lowering false detections of land as boats. Since this thesis focuses on harbour environments, it is not considered worthwhile to annotate buildings in all collected images.

6.4 Dataset split: training, validation and testing

Before training the neural network on the available data, we need to split the dataset into three subsets: train, validation and test. During training, the training set is showed to the neural network for minimization of the loss function and after each epoch the updated weights are evaluated on the validation set. The motivation for using a separate validation set is to avoid overfitting and for aiding hyperparameter search, such as updating the learning rate. If we exclude the validation set it is likely that the network will not be able to generalize.

In this thesis, the collected dataset Nidelva is merged with Hurtigruten_small for training and validation. Each of the two are randomly split into a training set and a validation set with a ratio of 85-15. This solution of a random split might not be ideal as some images are similar as they are taken of the same object with only a slightly different angle. However, several images will contain unique instances, and the time constraint makes this the preferred choice.

It is decided to merge the ”motorboat” and ”motorboat with priority” classes and ”sailboat with sails down” and ”sailboat with sails up” classes for training, validation and test. There are two main reasons for this: Firstly, there are only two classes present in the test set, namely ”motorboat” and ”sailboat with sails down”. Therefore, the other classes can not be properly evaluated. Secondly, there are very few ”sailboats with sails up” instances in the merged dataset. Such an unbalanced class distribution would cause problems for the neural network when it comes to learning this class. The resulting label distribution for the train and validation datasets is shown in table 6.7 and 6.8.

Finally, we need unseen an independent data as a test set in order to properly evaluate the

Total number of images	**222**
Motorboat	417
Sailboat	143

Table 6.7: Overview of the training dataset, which is a part from fusion of Hurtigruten_small and Nidelva

Total number of images	**39**
Motorboat	68
Sailboat	15

Table 6.8: Overview of the validation dataset, which is the other part from fusion of Hurtigruten_small and Nidelva

general performance of the neural network. This allows us to compare different models. The milliAmpere dataset is used as a test set in this thesis. For the test set it is much more important to avoid sample leakage, which is ensured when using this new test set.

The reason for choosing the milliAmpere dataset over the Fosenkaia dataset is that initial training showed poor performance on the Fosenkaia dataset, where low contrast is considered the most important reason. As new and better camera settings have been tested, future application is expected to be more similar to what we can see in the milliAmpere dataset, giving more realistic results.

Chapter 7

Method: Object detection using Neural Networks

7.1 Choice of model

The choice of models to investigate in this thesis is based on the literature review in section 5.2.1.

An interesting hypothesis to test is if the newest neural network object detection models that achieves state-of-the-art on the COCO-dataset also performs better in the specific domain of maritime detection in LWIR-images. Thus it is decided to compare one well known model that is frequently used in the literature with a newer model. YOLOv3 [Redmon and Farhadi, 2018] is chosen as the frequent model as it is tested with promising results in the same domain in [Schöller et al., 2019] and is preferred in terms of real-time application over the slower two-stage detector Faster R-CNN. YOLOv3 also outperforms SSD in terms of AP on the COCOeval dataset, and when compared in the maritime domain in [Grini, 2019].

EfficientDet [Tan et al., 2020] is chosen as the modern object detector. This was state-of-the-art for real-time application at the start of the author's specialization project [Kjønås, 2021] and barely tested here, so it was decided to continue to investigate this model. However, at similar frame rates Scaled-YOLOv4 outperforms EfficientDet, and future work on this area may including using this improved model. The scaled version D0 is chosen as this has the fastest inference time.

For the implementation of EfficientDet a PyTorch implementation called ”Yet Another

EfficientDet Pytorch”[1] was used. The reason for this was that this was an extension of the unpublished work of Thomas Skarshaug within the Autoferry-project. However this led to bad results after training. The detection performance was actually better when pre-trained on the COCO-dataset alone than when trained on the IR-images. A lot of effort was put into troubleshooting, among this tuning of hyperparameters, change of data and more. This was not leading to large progress with the training so finally, it was decided to abandon this implementation and use the original implementation in Tensorflow[2] made available by the authors of [Tan et al., 2020]. This has been an expensively learned lesson and it is recommended for everyone interested in performing future work with similar projects to verify the implementation and use the original work if possible.

For YOLOv3 the original implementation was first tested, but it was unfortunately difficult to make it work. Due to the delayed progress from struggling with the EfficientDet implementation, it was decided to use a PyTorch implementation by a company called Ultralytics[3], which has been acknowledged by the authors of [Bochkovskiy et al., 2020] for their contribution to improvements to the YOLO-models.

7.2 Experimental method

As described in the introduction, this thesis aims to explore different aspects of object detection in LWIR-images using neural networks. In order to systematically investigate the focus areas of comparing models, the effect of data augmentation and classification of motorboats and sailboats the following method has been used.

EfficientDet-D0 backbone, EfficientNet-B0, is pre-trained on the RGB-images dataset ImageNet, while this is not the case for the YOLOv3 backbone. Then both the EfficientDet-D0 and YOLOv3 models are pre-trained on the RGB-images COCO-dataset as introduced in section 6.1.2. The main advantage of this is that the neural network needs less training data in order to learn object features. A disadvantage of this is that RGB-images have three color channels, while IR-images only have one. This is currently solved by copying the IR-image and sending in all three color channels. This means that there will be additional weights in each convolutional layer in order to represent color features that are not necessary for the IR-images. Maybe the network could have been smaller with same results or deeper instead with same amount of weights? However, it is assumed that the size of the current available IR-dataset is not sufficient for training a neural network from scratch.

The pre-trained weights are downloaded from the repositories for the two models. Then both models and their corresponding pre-trained weights and the training, validation and test set that are described in section 6.4 are uploaded to a GeForce GTX 1080 Ti GPU for training and evaluation.

[1]https://github.com/zylo117/Yet-Another-EfficientDet-Pytorch
[2]https://github.com/google/automl/tree/master/efficientdet
[3]https://github.com/ultralytics/yolov3

First, all labels were merged into a combined ”boat” class for evaluation of the general detection performance. Then the original labels ”sailboat” and ”motorboat” were used for training and testing in order to evaluate classification performance as will be described in section 7.2.5.

7.2.1 Training: Choice of hyperparameters

Both models come with a set of hyperparameters that control the learning process of the neural network. The key hyperparameters are desicrbed in section 3.2.1. In order to improve the performance hyperparameter optimization can be applied. This has not been considered for this thesis due to limited time and scope. However, optimization is shown to have positive effect on the performance and is thus a recommended subject for further investigation in future work. [Yu and Zhu, 2020] is a survey that covers some major optimization algorithms and their applicability and can be a good starting point.

Nevertheless, the hyperparameters need to be set for training of the neural networks. Thus the strategy used for determining the key parameters are described in the following.

Epochs

The necessary number of epochs is found trough testing. It can be said that the network is done training when it starts overfitting so that the validation loss starts to increase while the training loss continues to decrease. Thus, the weights with the best validation performance in terms of AP is kept instead of the last available weights in order to avoid overfitting. Training is initially run with fewer epochs, e.g. for testing all data augmentation methods, and then extended if it is found necessary because the network does not show indications of overfitting. The main motivation for limiting the number of epochs is that it is time consuming to have many epochs. When presenting the results we will be operating with step size as described in (3.6) because the batch size is different for the two models.

Batch size

Typical batch sizes range from one to a few hundred [Goodfellow et al., 2016]. If a large batch size is chosen the estimated gradient is more accurate. However the hardware results in upper bounding of the value. It is decided to use the maximum value that the GPU allows. For YOLOv3 the batch size is set to 16 while for EfficientDet-D0 it is 32.

Besides these hyperparameters the default values described in the original papers are used [Tan et al., 2020][Redmon and Farhadi, 2018].

7.2.2 Evaluation

When evaluating the results of the training the evaluation metrics from chapter 4 are used on the test set. This is done for initial testing in order to quatify the results and compare them e.g. different data augmentation techniques. The best results are further investigating by plotting precision-recall curve etc. as will be further described in section 7.2.4.

7.2.3 Data augmentation

When examining the possible performance improvement of data augmentation, a selected set of augmentation techniques are applied individually during training and evaluated on the validation set for both neural network models. Then the best performing methods are combined and tested in the same manner. The validation set performance gives an indication of which methods improves the results and these are evaluated on the test set.

Online augmentation is used, primarily because it saves a lot of storing space, but also because it does not limit the amount of variations in images, as new augmentations can be applied each epoch.

This thesis will focus on geometrical data augmentation methods. [Shorten and Khoshgoftaar, 2019] recommends that the geometric transformations random cropping and translation should be manually inspected in order to ensure that the correct label is preserved, thus these methods are not tested in this thesis with the use of online augmentation. Otherwise, the selected techniques are horizontal flipping, scaling, rotating and mosaic transformation.

Horizontal flipping

Horizontal flipping is flipping the image and the overlaying bboxes in the horizontal direction, i.e. from left to right. By flipping the images we double the number of training images, where a flipped image is considered different than the original. In an online augmentation scheme, this means that for each image there is a probability of 50% that it will be flipped when loaded and prepared for training. An example of a flipped image where the ground truth annotation bounding box is flipped correspondingly is shown in figure 7.1.

An important motivation for using horizontal flipping is that it is a safe augmentation technique in the maritime object detection domain, as the bounding boxes and class labels do not risk being distorted.

In general it is also possible to use vertical flipping. However, given the maritime domain and the mounting of the cameras that this detection method will be used on, it does not really make sense to train on boats that are upside down, and it is therefore not applied.

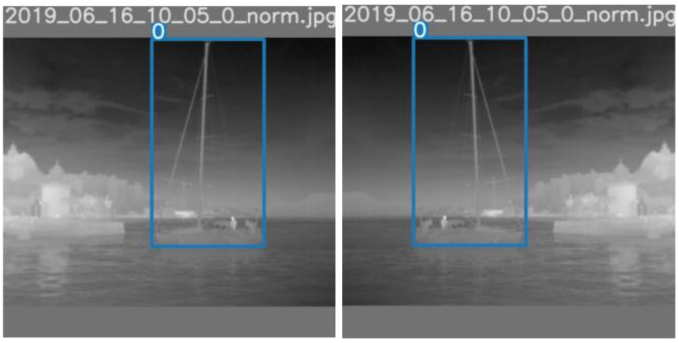


Figure 7.1: Example of a flipped image with ground truth annotation. The label 0 signifies ”boat”.

Rotation

The camera on board milliAmpere will be subject to rotations as waves can change the view of the camera given the movements of the ferry. Thus, it is interesting to train on rotated images for this case. In addition, it can help improve positional biases and overfitting as previously discussed.

When enabling rotation for training, the method is implemented so that each image has a 50% chance of being rotated, and the rotation angle is randomly drawn from a uniform distribution of degrees between $[-10, 10]$. When an image is rotated, the size of the bounding boxes is affected. Since they are described in terms of 4 positional arguments, $xmin$, $ymin$, $xmax$, $ymax$, some information is lost when they are rotated, and they might cover a larger area than the rotated object. An example of a rotated image that illustrates this is shown in figure 7.2.

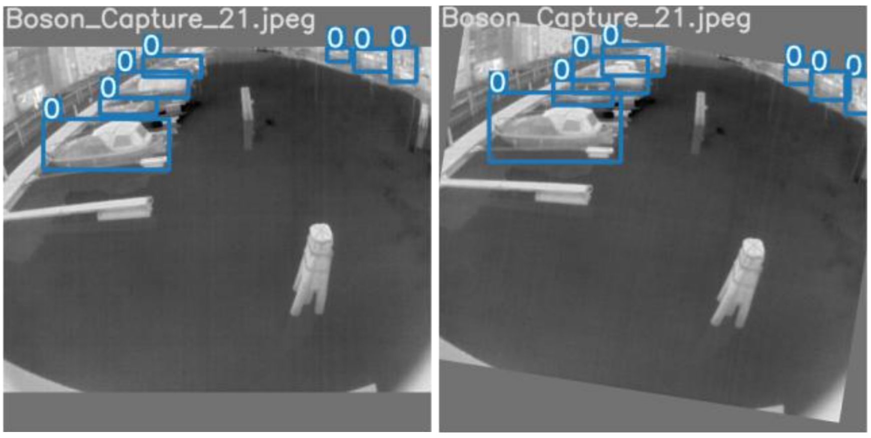


Figure 7.2: Example of the original and rotated image with ground truth annotation rotated correspondingly. The label 0 signifies ”boat”.

Note how the foremost boat has an annotated bounding box that is no longer tightly covering the object after rotation.

Scaling

The Hurtigruten dataset contains a substantial amount of instances that cover a large amount of the image. And the Nidelva dataset contains images of boats at a short distance, meaning that they also have large bounding boxes. During runs of training with no data augmentation, an early version of the detector seemed to prefer large bounding boxes that cover a large portion of the image. For this reason, downscaling might be a beneficial augmentation method, as the bounding boxes will be smaller in size during training.

In addition, zooming in, which is equivalent to scaling up, can result in a greater precision for small instances, thus scaling up is also performed.

Figure 7.3 shows an example image scaled both up and down.

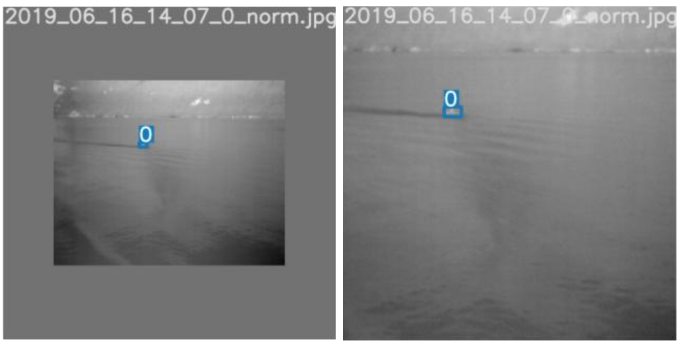


Figure 7.3: Example of a scaled image where the first is scaled down and the second is scaled up with ground truth annotation scaled correspondingly. The label 0 signifies ”boat”.

When scaling is enabled, each image is scaled by a scaling factor that is randomly drawn from a uniform distribution of $[0.5, 1.5]$, meaning that the size will be between 50% smaller and 50% larger.

Mosaic

[Bochkovskiy et al., 2020] presents a new data augmentation method which improves the detection performance on the COCO-dataset in terms of AP. The idea is to combine four images in a mosaic fashion so that different contexts are mixed so that detections can be less context dependent. Another advantage of this is that the batch size on which the gradient descent is performed is based on a larger subset of the images.

An example of a training image on which mosaic augmentation has been performed is shown in 7.4.

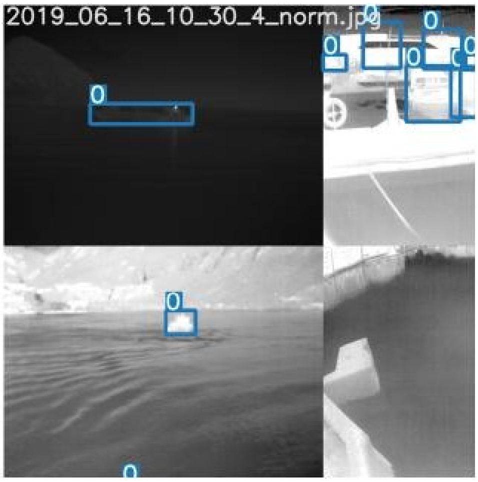


Figure 7.4: Mosaic composition of four training images, cropped in order to fit the input size. The label 0 signifies ”boat”.

As can be seen from the figure, the image is randomly cropped as the combined images exceeds the input size. When enabled, the augmentation is applied with a probability of 50%.

Unfortunately, it was difficult and would have been more time consuming to adapt the dataloader used for training for EfficientDet to this method when using an online augmentation scheme, and thus this was not prioritized. However, the method is tested with the YOLOv3 detector and it is assumed that the results can give an indication as to whether this technique should be prioritized and applied to models in future work.

7.2.4 Comparison of the best models

In order to properly compare the performance of the two neural network models, the evaluation metrics from section 4 are used.

Then, interpolated precision-recall curves are plotted for different IoUs according to the equations in section 4.3 inspired by [Schöller et al., 2019].

Based on these results tuning of evaluation parameters is necessary before inference on the test set, as these thresholds are used for inference.

Tuning of evaluation parameters

In order to quantify whether a prediction is counted as a detection of a ground truth object, an **IoU** threshold must be determined. However, this threshold will not affect the performance of a tracker as we do not have ground truth bounding boxes in real applications and so it is used purely for evaluation purposes. Because of this the choice is based on similar values found in the literature.

The **confidence** of a bounding box is a part of the prediction output describing objectiveness, i.e. if an object is present or not in the given predicted bounding box. This value determines the trade-off between precision and recall from the precision-recall curves and must be pre-defined by the user for detection applications. By setting the confidence threshold lower, we favor higher recall values, while increasing confidence threshold results in higher precision.

The ideal threshold depends on the specific application, thus the value should be tuned for the final tracking-algorithm which is beyond the scope of this project. However, as we know that tracking is the application, some reflections regarding the trade-off between precision and recall can be made.

A possible consequence of false negatives or misdetections is track loss. When the tracking-system is used for collision avoidance the worst consequence of not detecting a present boat is collision between the autonomous vessel and the other boat. The predictions from the IR-images will be fused with predictions from other sensors and if the misdetection rate is too high this can degrade the fused data, rather than improving them. As a result, we must limit the confidence threshold in order to avoid too many misdetections.

On the other side, false positives or false alarms can result in tracking of non-existing targets. The worst consequence of this is that false tracks hinders the autonomous vessel from operating, or even worse, collision avoidance of a false track could be prioritized over a real one in a situation where collision is unavoidable. However, false positive predictions may be handled by sensor fusion for target tracking. Furthermore, if the false alarm rate is too high, the total number of predictions can potentially cause problems for the processing time of the track management and data association algorithms and thus struggle to operate in real-time. This can be handled by limiting the number of allowed detections per image or by setting a lower limit for the confidence threshold.

The conclusion from this is that minimizing misdetections is considered more important and thus a high recall is prioritized, meaning that the confidence threshold can be decreased.

Based on the tuned thresholds, videos and images of the prediction results are made. Furthermore, it is especially interesting to analyze the detection performance on the moving motorboats in the test set as these are the most important targets to track. The pixel area of the ground truth bounding boxes are plotted vs. time, combined with which frames have true positives (detection) of the moving targets, and which result in false negatives (misdetection).

7.2.5 Classification

In order to test classification performence of the models, the original labels ”sailboat” and ”motorboat” were used for both training, validation and testing. Buildings were still not included as these are difficult to annotate.

The results will be shown on the test set in terms of videos, and confusion matrices showing how often sailboats are misclassified as ”motorboat” and the other way around. These matrices will be explained together with the results as this makes it easier for the reader to follow the explinations.

Chapter 8

Experimental results

This section aims to present and discuss results that can answer the questions provided in the introduction in section 1.2.

8.1 Effect of training on IR-images

Since ”boat” is a class label in the COCO-dataset and both models are pre-trained on this dataset, it is interesting to compare training-results to evaluation from the pre-trained weights. This will function as a benchmark for evaluating the effect of training on IR-images. The pre-trained weights are inferred on the test set and evaluated for both EfficientDet-D0 and YOLOv3. These results are compared to training of EfficientDet-D0 and YOLOv3 with no data augmentation in table 8.1. The data used for training, validation and test is described in section 6.4. For the training the batch size is set to 32 for EfficientDet and 16 for YOLOv3 due to hardware limitations. Otherwise, default hyperparameters are used except that all data augmentation is turned off. Steps signifies the number of times the weights are updated as given in (3.6) for the best epoch, while the total number of steps ran is shown in parentheses.

We can see from this table that training largely improves the results. We can also see that YOLOv3 performs slightly better than EfficientDet-D0 both on AP and AR. However, EfficientDet-D0 is better on medium and large objects which is later shown in table 8.4 to be true for AR as well.

Model	Steps	AP	$AP_{.50}$	$AP_{.75}$	AP_{small}	AP_{med}	AP_{large}	AR
E-D0	0	0.0	0.1	0.0	0.0	0.0	7.4	1.7
E-D0	297 (988)	17.7	40.0	13.1	1.6	24.0	34.6	33.2
YOLOv3	0	0.0	0.2	0.0	0.0	0.0	7.3	1.6
YOLOv3	1680 (1680)	18.0	36.4	15.4	6.8	22.7	27.1	35.7

Table 8.1: Comparison of results from using only COCO pre-trained weights and after training.

8.1.1 Overfitting

Overfitting as described in section 3.2.1 occurs when the network is not able to generalize by being overly adapted to the training data.

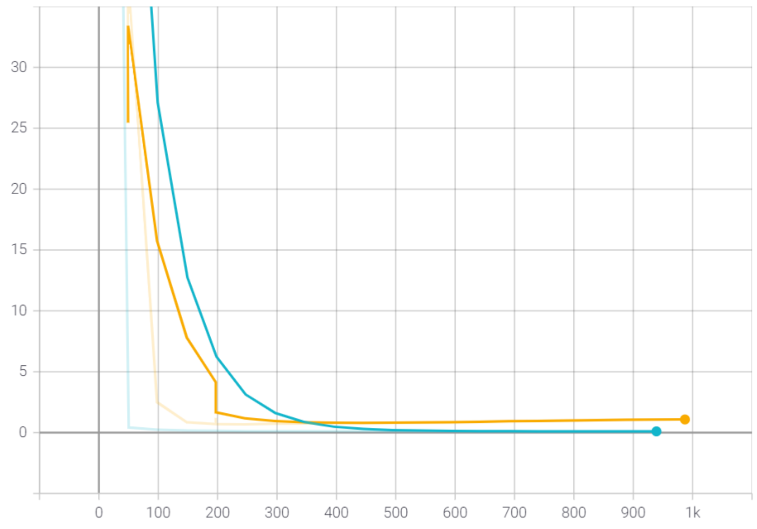


Figure 8.1: Loss vs. number of steps for EfficientDet-D0. Blue is training-loss and yellow is validation-loss. A visual smoothing factor of 0.5 is used for the plot.

After epoch 5 at step 246 the validation loss starts to increase even though the training loss continues to decrease. This is a sign of overfitting. The effect could have been even stronger was it not for the adaptive learning rate for these models which decreases the learning rate when the validation loss increases. The appearance of overfitting to the training data is the reason for using the weights from the best validation epoch instead of the last epoch. Note that although the validation loss is larger at epoch 6, this epoch provides the best AP_{val} and is thus presented in the results.

The overfitting happens quite early which is expected given the small size of the training set. This motivates using data augmentation in order to continue to decrease the loss for more steps.

8.2 The effect of data augmentation

When analyzing the effect of data augmentation the method described in section 7.2.3 is used. The evaluation results on the validation set using individual and combined augmentation techniques are shown in table 8.2. Steps denotes the best performing step, while the total number of steps ran is shown in parentheses.

Model	Steps	Flip	Scale	Rotate	Mosaic	AP_{val}	AR_{val}
EfficientDet-D0	297 (988)					40.3	56.7
EfficientDet-D0	642 (988)	✓				42.2	57.9
EfficientDet-D0	988 (988)		✓			51.8	65.5
EfficientDet-D0	544 (988)			✓		37.2	54.6
EfficientDet-D0	1482 (1679)	✓	✓			56.9	66.0
YOLOv3	1680 (1680)					46.2	59.7
YOLOv3	1386 (1400)	✓				46.1	60.7
YOLOv3	1372 (1400)		✓			55.2	64.6
YOLOv3	1400 (1400)			✓		50.1	59.2
YOLOv3	1400 (1400)				✓	58.9	66.8
YOLOv3	1358 (1400)	✓	✓			51.9	62.0
YOLOv3	1316 (1400)	✓	✓		✓	53.8	64.6

Table 8.2: Comparison of results on the validation set for different data augmentation methods.

The first thing to note from this table is that for EfficientDet the step and thus epoch that gives the best validation results is lower when no augmentation is used than when various techniques are applied. For the combination of augmentations, the network is trained for even more steps as the validation AP continued improving longer for this model. This means that the network was overfitting earlier when data augmentation was not used, which is expected behaviour and a good sign because this indicates that the variation in the data is so that the neural network learns more features. For YOLOv3 however, the steps executed do not seem to result in overfitting. With no augmentation, the model started to overfit around epoch 1960, and due to time constraints it was decided to train all models to epoch 1400 instead of arrival of overfitting to have comparable training length to EfficientDet. It is assumed that the compared metrics on the validation set give an indication of how well the techniques perform. The best performing models are trained more steps before evaluation on the test set with the results shown in table 8.3 and 8.4.

Another result that can be observed from table 8.2 is that applying rotation to the training data degrades the results for EfficientDet. This is unexpected, as the FOV of the image is indeed rotated by a few degrees in the test set. The same problem with rotation was found in [Landsnes, 2021], where rotation deteriorate precision. It is difficult to tell exactly why this augmentation methods works poorly because the training process of the neural network is not transparent. One possible explanation is that the rotation of the bounding boxes results in less accurate bounding boxes which leads to degraded precision. For YOLOv3 the performance in terms of AR is worse, while the AP is improved. From this it is decided to not combine rotation with the other methods for further training.

For EfficientDet, the individual tests of data augmentation show that flip and scale seem most promising and are thus combined for the final test. There is no guarantee that combining data augmentation techniques that are individually effective results in better performance for several reasons such as safety issues and that augmentation does not solves issues related to limited training data diversity [Shorten and Khoshgoftaar, 2019]. However, here the table shows that combining scale and flip further improves performance for the validation set.

Similarly, flip and scale are combined for YOLOv3 in order to compare the results to EfficientDet. Furthermore, we can see that mosaic outperforms all other techniques on the validation set. These tree methods are combined which actually results in worse metrics than both scale and mosaic individually. This is unexpected, but a possible reason is that the combined techniques might need more training steps.

The best performing augmentation methods from table 8.2 are evaluated on the test set and the results are shown in table 8.3 for AP-metrics and table 8.4 shows AR-metrics.

Model	Augment	AP	$AP_{.50}$	$AP_{.75}$	AP_{small}	AP_{med}	AP_{large}
E-D0	No aug	17.7	40.0	13.1	1.6	24.0	34.6
E-D0	Flip + Scale	18.3	45.4	15.4	1.0	22.0	50.4
YOLOv3	No aug	18.0	36.4	15.4	6.8	22.7	27.1
YOLOv3	Flip + Scale	20.4	48.0	15.0	5.2	27.0	32.7
YOLOv3	Flip + Scale + Mosaic	21.9	56.6	13.7	11.6	27.6	28.1
YOLOv3	Mosaic	27.7	54.1	23.9	7.5	34.0	46.4

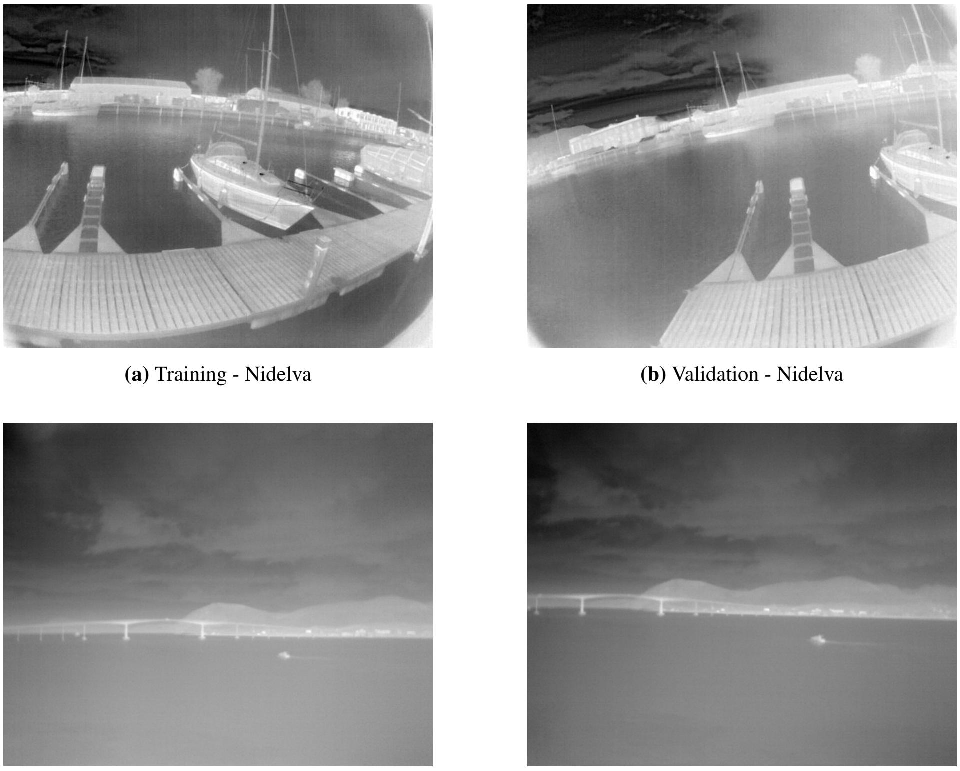
Table 8.3: Comparison of AP results on test set for different data augmentation methods.

Model	Augment	AR	AR_{small}	AR_{med}	AR_{large}
E-D0	No aug	33.2	9.4	42.1	55.6
E-D0	Flip + Scale	34.1	15.1	37.5	69.4
YOLOv3	No aug	35.7	28.7	36.8	49.5
YOLOv3	Flip + Scale	38.9	30.0	43.3	42.4
YOLOv3	Flip + Scale + Mosaic	43.7	36.8	46.1	51.0
YOLOv3	Mosaic	39.5	24.5	44.1	58.3

Table 8.4: Comparison of AR results on test set for different data augmentation methods.

The first thing to notice is that the evaluation metrics for the test set are all significantly lower than for the validation set. The most probable reason for this is that several images in the validation set are very similar to those in the training set because they are depicting the same objects shortly before or after the other image is taken. Examples of this is shown in figure 8.2.

This may cause overfitting and is unfortunate. From this perspective, the validation set



(a) Training - Nidelva

(b) Validation - Nidelva

(c) Training - Hurtigruten

(d) Validation - Hurtigruten

Figure 8.2: Example images from training and validation set that are similar.

should not have been chosen at random, in stead unique images should have been selected in order to avoid sample leakage. Unfortunately, time was not found to do this after the production of these results in this thesis and it is recommended as future work.

Another effect from this is that the significant performance improvement of the validation metrics when using data augmentation is not transferred to the test set. Although we can see an improvement it is not as large as we would have expected from the results of the validation metrics. Furthermore, with a biased validation set, the best performing results during validation might not be the best on unseen data if the weights are overfitted. This is illustrated by YOLOv3 where we see that flip + scale + mosaic performs better on the test set in terms of AR than mosaic alone, which is not the results on the validation set.

Nevertheless, AP and AR are both improved when adding the data augmentation. Table 8.3 shows that for EfficientDet, the greatest improvement in AP is AP_{large}. A possible reason for this is that the data augmentation technique scaling enables the possibility to learn object with a greater variation of sizes. However, AP_{small} and AP_{med} are worse. The author has no good explanation for this. Table 8.4 shows that AR are generally improved by introducing flipping and scaling in addition to improvements of both AR_{small} and AR_{large}. For YOLOv3, flip + scale performs similarly, where some metrics are improved while others are better without augmentation. By introducing mosaic almost all metrics are improved, both alone and with flip and scale. This shows great potential for using this data augmentation technique and is recommended for future work.

The overall results are considered improved when introducing the specific data augmentation methods. In addition, the significant improvement on the validation set shows great potential for using data augmentation techniques on IR-images.

8.3 Analysis and comparison of the best results

The best performing models from table 8.3 and 8.4 are considered to be flip + scale for EfficientDet because both AP and AR are better than without augmentation. For YOLOv3 it is considered to be flip + scale + mosaic because AR is better with these techniques than mosaic alone. Recall is the most important metric for the tracking application since it is of higher importance to detect all objects and avoid false negatives than to introduce false positives which can be suppressed during fusion with data from other sensors. These models with the given data augmentation techniques will be the base for further analysis.

The inference time of the two models are not tested as the final hardware is unknown. However, it is assumed that the times provided by their respective papers, which are recited in table 5.1, are valid so that EfficientDet-D0 is approximately 1.34 times faster than YOLOv3.

The precision-recall curves for different IoU thresholds are plotted in figure 8.3 for EfficientDet-D0 and 8.4 for YOLOv3.

The figures show that for high precision values and high recall values the performance of the two models are similar. Although there is a clear difference for larger IoUs where EfficientDet-D0 favors precision and YOLOv3 favors recall. The middle parts along the curves are better for YOLOv3 as better performing models have curves that are closer to the upper right corner, meaning that the trade-off between precision and recall is less expensive.

However, the differences are not very large and the two models will be further compared in the following analysis.

8.3.1 Choice of tuneable evaluation parameters

An IoU and a confidence threshold must be decided for further evaluation.

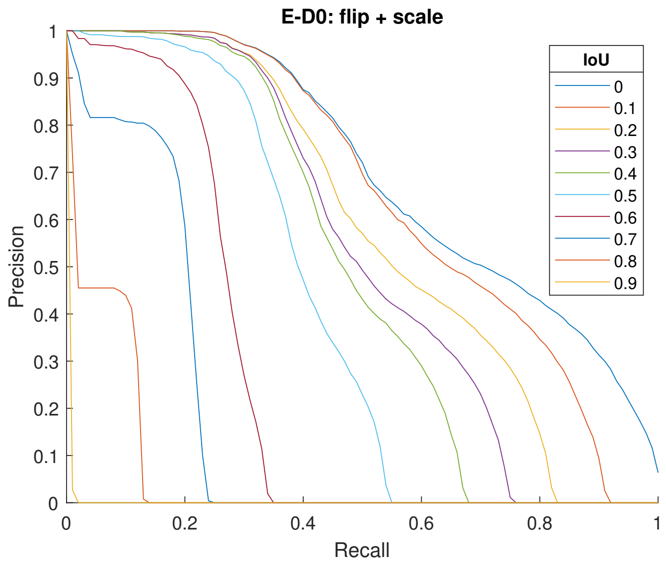


Figure 8.3: Precision-recall curve for different IoU-values. EfficientDet-D0: Flip + scale.

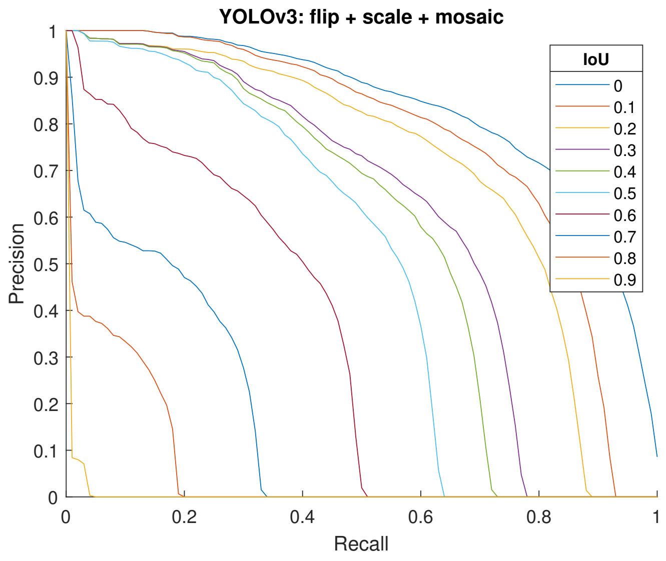


Figure 8.4: Precision-recall curve for different IoU-values. YOLOv3: Flip + scale + mosaic.

IoU threshold

In [Schöller et al., 2019] an IoU threshold of 0.3 was chosen in order to favor a high recall. They argue that for small objects, small errors in the predicted bounding boxes affects the

IoU a lot.

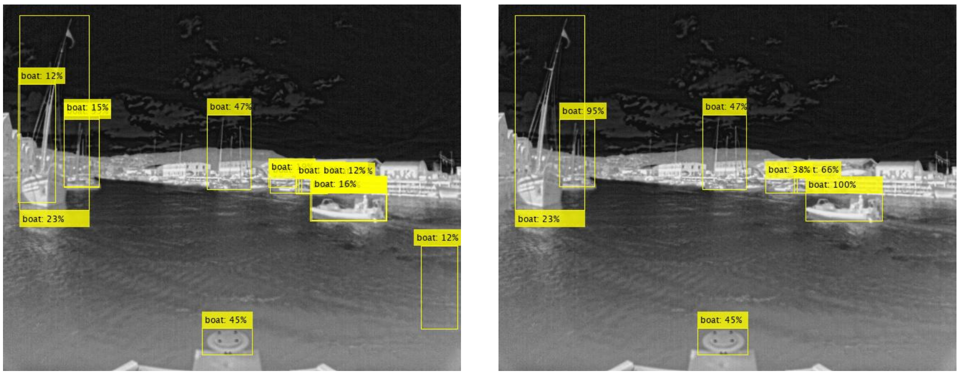
We see from the precision-recall curves in figure 8.3 and 8.4 that the difference between an IoU of 0.3 and 0.4 is quite small, especially for higher precision values, and thus we choose an IoU threshold of 0.4.

Confidence threshold

[Schöller et al., 2019] uses a confidence threshold of 0.6. However, it is noteworthy that their precision-recall curves resembles the shape of a 90 degrees angle meaning that the high threshold sacrifices less recall than for the curves presented in this thesis. In order to keep a high recall, the confidence threshold needs to be set lower in this case.

For the given IoU threshold it can be seen from figure 8.4 and 8.3 that we can go up to a recall of at least 0.4 without significant loss of precision. This corresponds to a confidence of 0.3 for EfficientDet-D0 and approximately 0.21 for YOLOv3. Based on these values, tuning is performed through manual inspection of the image inference results. Some observations will be presented in the following.

With a higher threshold of 0.3 we miss moving boats at large distances for both models. Reducing the threshold to 0.2 does not seem to result in significantly more false detections while detecting the moving boats at a greater distance. If the confidence threshold is further reduced to 0.1 we can observe more false positives which can be seen from figure 8.5a, comparing the same image with a threshold of 0.1 and 0.2, respectively.



(a) Confidence threshold of 0.1 **(b)** Confidence threshold of 0.2

Figure 8.5: EfficientDet-D0: Flip + scale. Inference on the same image with different confidence thresholds.

The first observation to note from this figure is the "random" prediction covering parts of the water that appear in the lower right corner of figure 8.5a with a confidence score of 0.12 and not present when the confidence threshold is increased to 0.2. It is present in the given frame and not the former or proceeding frames. Such jumpy predictions can easily be handled by track initialization or management. However, if there are too many such false positives in a neighboring area, they might result in false tracks.

The second noteworthy observation is the double detections of the moving boat in the right center in figure 8.5a, where a detection with confidence 0.16 is highly overlapping with the detection visible in figure 8.5b with a confidence of 1. By increasing the threshold we see that only one detection of this boat is present. However, this can also be handled by modifying the non-max suppression of the output predictions so that if the IoU of two predictions is too large, only the one with the highest confidence is kept. A similar double detection is made for the sailboat in the left side of the image, but with a much lower IoU. For such cases, it is preferable to increase the confidence threshold.

The last false positive prediction to note is present in the lower center of the image for both images. A part of the autonomous vessel present in all images from this camera is predicted to be a boat with a high confidence of 0.45. However, as this region of the image is equal in all frames, false predictions in these areas can easily be suppressed by ignoring predictions from the pre-defined area.

Similar observations are present in several frames for both EfficientDet-D0 and YOLOv3 when setting the confidence threshold to 0.1. When increasing the threshold to 0.2, many are removed, especially the double detections and jumpy predictions. In conclusion, a confidence threshold of 0.2 is chosen for further evaluation in this thesis. For comparison reason, the same confidence threshold is chosen for both models.

The final chosen evaluation parameters are summarized in table 8.5

Parameter	**Value**
IoU	0.4
Confidence threshold	0.2

Table 8.5: Choice of evaluation parameters

8.3.2 Video results

Videos showing the inference of the trained networks using the confidence threshold from table 8.5 can be found on

[https://drive.google.com/file/d/1dhRbIbwMPdq5G4-w_aVc42n130PHnzxp/view?usp=sharing] for EfficientDet-D0 and

[https://drive.google.com/file/d/1zj5GoqHDMj2KvSCpOiRihpuOpRKIGPnO/view?usp=sharing] for YOLOv3.[1]

[1]These links will be maintained in three months from submission of this thesis. If the reader is interested in the videos after this and the links don't work, feel free to contact the author by e-mail.

In addition to the videos, randomly chosen images comparing ground truth, EfficientDet-D0 and YOLOv3 are shown in figure 8.6, 8.7 and 8.8. These are meant as a supplement to the videos, which are recommended to watch as the text in the images is quite small.

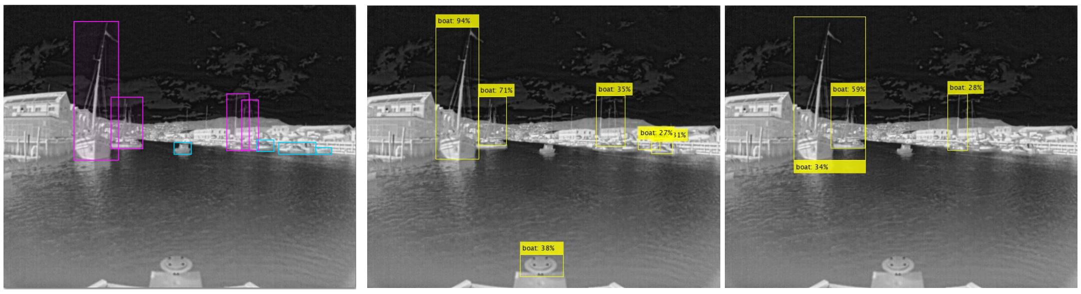


Figure 8.6: Example images from frame 20: Ground truth to the left, EfficientDet-D0 in the middle and YOLOv3 to the right. For ground truth pink is ”sailboat” and blue is ”motorboat”.

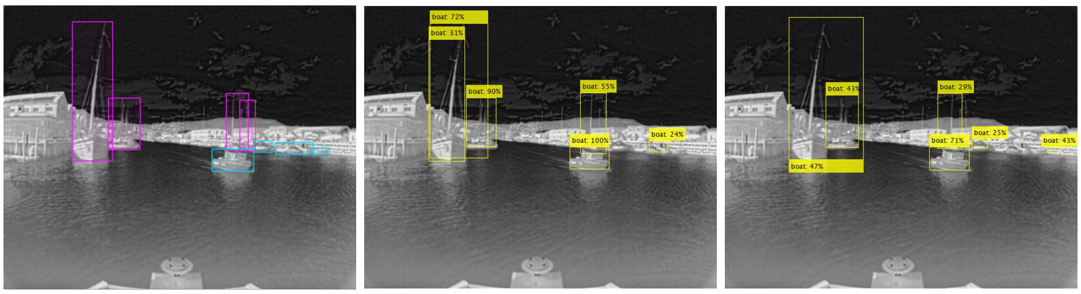


Figure 8.7: Example images from frame 111: Ground truth to the left, EfficientDet-D0 in the middle and YOLOv3 to the right. For ground truth pink is ”sailboat” and blue is ”motorboat”.

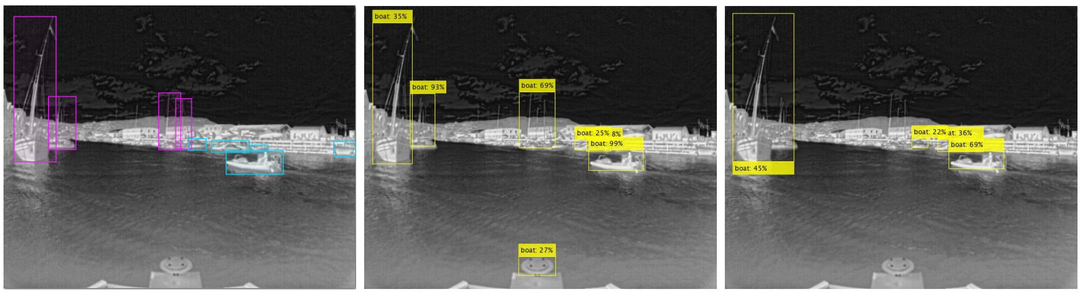


Figure 8.8: Example images from frame 253: Ground truth to the left, EfficientDet-D0 in the middle and YOLOv3 to the right. For ground truth pink is ”sailboat” and blue is ”motorboat”.

In general, the videos and images show that both models are able to detect boats in IR-images. The overall results seems promising in spite of some false predictions and misdetections. We can observe that both models struggle with the moving boats at large distances i.e. small pixel areas. In addition, the difficult docked sailboats and motorboats at the right side of the images are often misdetected. These are all marked as difficult and it was thus expected that the models would perform worse for these objects.

The videos show that YOLOv3 is better at large distances for the moving targets, while figure 8.7 and 8.8 show that at close ranges, the confidence score is higher for the moving boats when using EfficientDet-D0 than for YOLOv3.

8.3.3 Detection of moving boats

From the videos it seems that many misdetections are related to difficult small docked boats on the right side of the images, thus it is interesting to analyze the detection performance on the targets of interest, i.e. the two moving boats. As these are the only boats that are annotated with the combination of the label ”motorboat” and difficult ”0” in the test set, they can easily be extracted for evaluation.

Another interesting analysis we can perform on the moving boats is the correlation between detection performance and pixel area, which corresponds to the distance between the camera and the target. Since the moving targets are moving towards and away from the camera, the pixel area will increase and decrease correspondingly for the given targets. The combination of pixel area and whether the moving boat is detected or not is shown in figure 8.9 for EfficientDet-D0 and 8.10 for YOLOv3. The blue graph is the pixel area vs. time, while the orange scattering points are showing whether the moving boats are detected (”yes”) or not (”no”) and when neither boat is present (”gone”) at each time step in the test video. The boats are detected when a prediction is made with an IoU with the ground truth and a confidence score above the chosen thresholds.

For both models, we can observe that when the pixel area is large enough, i.e. the boats are close enough to the camera, the detection probability is 100%, which is very good. In the transition when the boats leave or enter the frame both models have some misdetections. This is not surprising as when the boats are truncated along the edges of the frame, their features can be difficult to recognize. As milliAmpere will be equipped with several IR-cameras with different while somewhat overlapping fields of view, the targets may be detected in images from neighbouring cameras in the transition region.

Some small differences for the two models are present. For EfficientDet-D0, the moving boat that enters the frame is detected in the majority of the frames up to approximately 50 s, while this is limited to approximately 40 s for YOLOv3. However, for both models some misdetections are present in some frames up to these points in time. It is assumed that the tracking algorithm can handle a few jumpy misdetections before loosing the track. Still, this might cause loss of the track for instance given the density of misdetections around 40 s for YOLOv3. This should be tested in analysis of the tracking performance.

On the other side, YOLOv3 performs slightly better for very small pixel areas and large distances in the end of the video. Note the upper right corner of figure 8.10 showing several

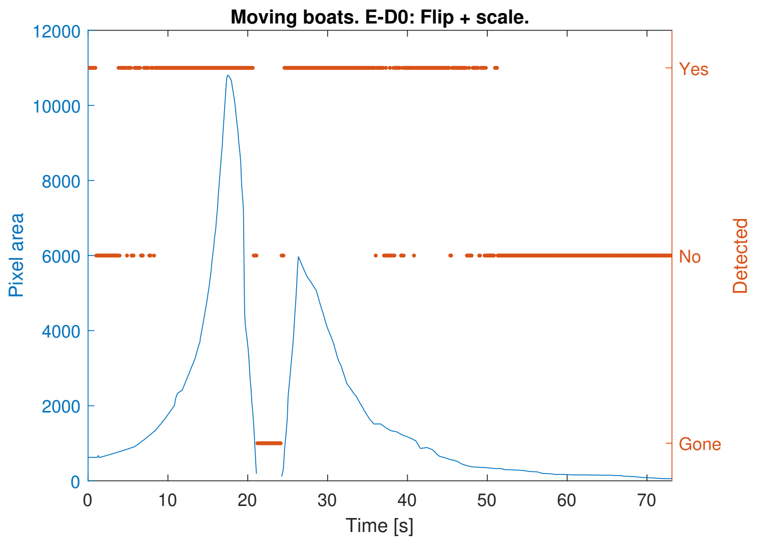


Figure 8.9: Pixel area and detection of moving targets vs. time for EfficientDet-D0.

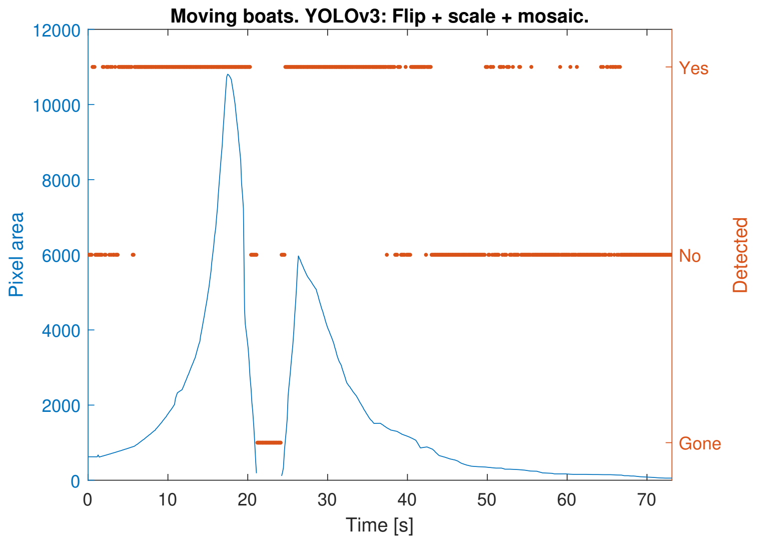


Figure 8.10: Pixel area and detection of moving targets vs. time for YOLOv3.

detections with very small pixel sizes. However, these are not consecutive and surrounded by many misdetections and may therefore not result in proper tracks.

The detection performance on the moving targets is very promising for the application of tracking alone or combined with other sensors. Including fusion of other sensor data might extend the pixel area threshold limit for detection and help the tracker to deal with the jumpy misdetections.

8.4 Detailed classes

The following analysis examines the classification performance of the two best models with data augmentation from section 8.3 in order to distinguishing ”sailboat” from ”motorboat”. The same confidence and IoU thresholds as in table 8.5 are used. The results from inference of the test video are shown for EfficientDet-D0 in: [https://drive.google.com/file/d/1opcpqPj83nyuRIskFj_zPBfu1SEYV3WR/view?usp=sharing]

and for YOLOv3: [https://drive.google.com/file/d/18TN2TREzoAKv8xqR29jBHc9MFteFRr5L/view?usp=sharing]

One example image frame comparing ground truth, EfficientDet-D0 and YOLOv3 is shown in figure 8.11.

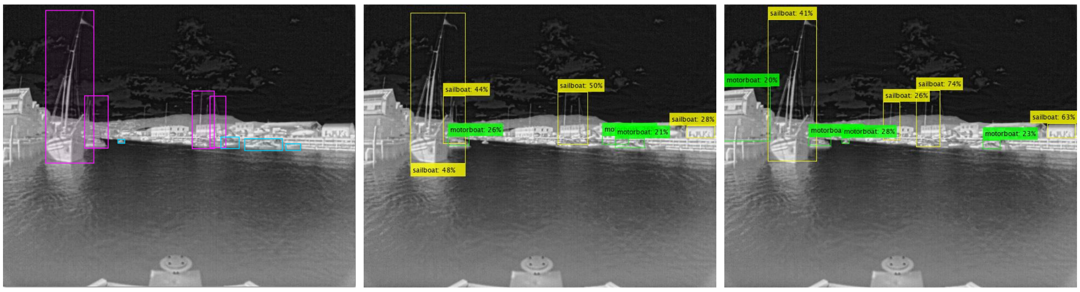


Figure 8.11: Example images for detailed classes from frame 598: Ground truth to the left, EfficientDet-D0 in the middle and YOLOv3 to the right. For ground truth pink is ”sailboat” and blue is ”motorboat”.

The first observation from the videos and images is that the general detection performance seems slightly worse than when training with only the one class ”boat”. Note for instance that we have more false positives predicting buildings as boats. This can be seen in figure 8.11 where the building to the right in the images is predicted as sailboat for both models. False predictions of buildings has been a problem for previous work within the domain [Kamsvåg, 2018][Grini, 2019]. When using only one ”boat” class almost no building were predicted to be boats. This change is strange behaviour as we train on the same data. One suggestion for why it happens is that when using two classes fewer training examples per class are available which may lead to the detectors not being able to learn and generalize features properly. For further investigation of this, it is recommended to include more training data.

In order to evaluate the networks ability to correctly predict the label can be examined through plotting of a confusion matrix. A confusion matrix shows a comparison between the ground truth classes in the images, and the predicted classes from our models.

The confusion matrix for EfficientDet-D0 is shown in figure 8.12 and for YOLOv3 in figure 8.13.

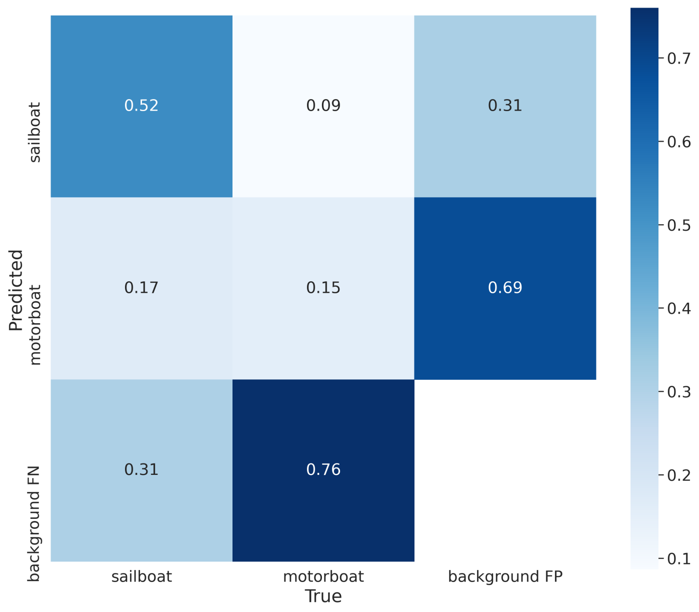


Figure 8.12: Confusion matrix for EfficientDet-D0.

The x-axis represent the true classes which are the ground truth labels for the bounding boxes for ”sailboat” and ”motorboat”, and a background class for evaluation of false positives. The y-axis shows the predicted classes which are the predicted labels for the bounding box outputs from the models. When we predict a ”sailboat” or ”motorboat” and the truth is background as in the rightmost column we have false positives. We have false negatives or misdetections when we predict background and the truth is either ”sailboat” or ”motorboat” as in the lowest row. In this case of object detection when using bounding boxes, it does not make sense to talk about true negatives, so the lower right box is empty. The remaining boxes represent true positives that are either correctly classified when predicted and true labels match, or misclassified.

From the confusion matrices we observe that we have some misclassifications for both models. EfficientDet-D0 more often misclassify ground truth ”sailboats” as ”motorboats” compared to YOLOv3, while the latter more often predict ”sailboat” when ”motorboat” is ground truth. Both models generally struggle to predict motorboats and performs much better at classifying sailboats.

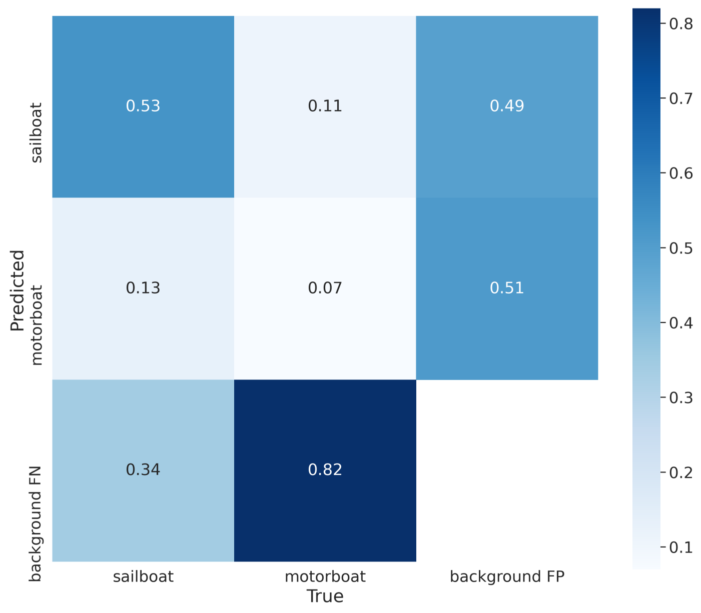


Figure 8.13: Confusion matrix for YOLOv3.

In the videos we see that the moving motorboats almost always are classified as ”motorboat” and the large sailboats to the left of the images are very often classified as ”sailboat” and the masts are included in the predicted bounding boxes. Thus, the low numbers in the confusion matrices might be a result of poor performance on the difficult targets to the right in the image. Note that when the moving targets passes a sailboat the motorboats are sometimes classified as ”sailboats” meaning that masts seem to be a learned and important feature for classifying sailboats.

As a conclusion we see potential for classification in terms of separating ”motorboats” from ”sailboats”, but we probably need more training data in order to improve performance.

8.5 Discussion of the evaluation metrics

Finally, it should be noted that in order to count a prediction as a true positive, the IoU threshold is used, although the adequacy of a prediction depends mostly on the lower edge of the predicted bounding box. When the IR-detections are used for tracking, we need to find the distance to the target and the bearing angle. The distance from the camera to the target can be found using georeferencing in the maritime domain, where we can apply a method presented in [Helgesen et al., 2020] using the lower bound of the target and camera properties and calibration. An accurate georeferencing estimation of position requires very high precision for the object pixel coordinates because of the sensor resolution [Helgesen et al., 2020]. In addition, for the bearing angle, the side edges of the bounding boxes should be accurate.

However, IoU is not directly presenting these edges accurately. For instance, some detections of sailboats only include the hull and not the mast. This can give a very accurately positional estimate although the IoU is low. Still, for classification, the mast seems to be an important feature for sailboats and can thus be essential to detect as a part of the bounding box.

[Prasad et al., 2020] highlight some problems using the general AP and AR metrics in the maritime domain and suggest a new metric called bottom edge proximity for a better performance evaluation. This metric addresses the previous discussion and focus on the accuracy of the lower edge of the bounding boxes.

8.6 Discussion of the datasets

As previously discussed, there are images that resemble in the validation set and training set, which can result in problems of avoiding overfitting. The test set used in this thesis is a video, motivated by independence from the training and validation data. In addition, another advantage of using this test set is that the video is a realistic application of the IR detector, providing useful information regarding detection of moving targets and the relation between pixel area (and target distance) and detection performance given the same target.

On the other hand, a disadvantage of using a video combined with metrics such as precision-recall, AP and AR is that the images are correlated across frames. This means that we test on a smaller variety of targets than when using independent images in different contexts. The background is also similar across the images. Thus, before the detector is applied for collision avoidance, it is recommended to test the performance on a greater variety of targets and with different backgrounds.

Chapter 9

Conclusion

This thesis has compared the neural network models EfficientDet-D0 and YOLOv3 on maritime LWIR images from a dataset partly collected by the author for this project. As a conclusion, we will aim to answer the questions from the introduction in section 1.2.

Compared to using COCO pre-trained weights, training the networks on LWIR images significantly improves the performance. This shows that the collected dataset has potential for future usage.

It is difficult to conclude which model is better because their performances are very similar. We observe from the inferred videos that both models perform well on close and large targets. When the pixel area is larger than approximately 1800, both models detect the moving target in 100% of the cases. For very small pixel area targets YOLOv3 seems slightly more promising. The superior performance might be because the data augmentation technique mosaic is used during training for YOLOv3 and not for EfficientDet-D0. We note that the COCO AP is only slightly better than for YOLOv3 in table 5.1 and we conclude that hyperparameters without complete tuning and augmentation differences counterbalance the improvement. As EfficientDet-D0 is 1.34 times faster than YOLOv3, one might test with a larger scaled model, for instance EfficientDet-D1 or D2, that can highlight the performance difference better for similar inference times. Larger models are expected to perform better on smaller targets.

Geometrical data augmentation techniques including flip, scale, rotate and mosaic have been tested in this thesis, where rotation degraded the results, while the other techniques improved the performance on the validation and test set alone or combined. Mosaic was only tested for YOLOv3 and showed very promising results and is thus recommended for future work.

Classification of types of boats, in terms of differentiating between motorboats and sailboats, is shown to work to some degree, especially on large and close targets. The models struggle more when motorboats and sailboats passes one another and at large distances. It is thus concluded that more training data is needed in order to improve the classification results.

Furthermore, excluding the sky from the ROI of the LWIR camera and increasing DDE has shown to significantly improve the contrast of the images.

9.1 Future work

Some suggestions for future work are:

- Collection of more data for testing and training. It is preferable to include more classes such as kayaks, and collect data under different weather and lighting conditions. Included in this point is the need for an independent validation set.
- Further tuning of hyperparameters should be performed, including hyperparameter optimization. One suggestions is to look into calculation of aspect ratios using k-means clustering.
- As data augmentation have shown to improve the current results, researching and testing even more augmentation methods, not only geometrical ones but also color transformations and GANs, is recommended. Other suggestions from the literature are CutMix and grid mask. Mosaic is recommended to include for EfficientDet if one wish to continue with this model.
- Depending on the final hardware, using larger models such as EfficientDet-D1 or D2 could be tested as this is expected to improve performance, especially for smaller targets.
- Sensor fusion for multitarget tracking.

Although several areas of improvement are identified, the detector has a high detection probability on large pixel area targets and can thus be tested with sensor fusion for multitarget tracking with the current results.

Bibliography

[Berg, 2016] Berg, A. (2016). Detection and tracking in thermal infrared imagery. *Licentiate dissertation, Linköping University Electronic Press*.

[Bochkovskiy et al., 2020] Bochkovskiy, A., Wang, C., and Liao, H. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934.

[Elfwing et al., 2018] Elfwing, S., Uchibe, E., and Doya, K. (2018). Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107.

[Everingham et al., 2010] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303 – 338.

[FLIR Commercial Systems, 2018] FLIR Commercial Systems (2018). *Datasheet: Boson Camera Adjustment Application Note*. 6769 Hollister Ave. Goleta, CA 93117, 221 edition. https://www.flir.com/products/boson/.

[FLIR Systems Inc., 2019a] FLIR Systems Inc. (2019a). *Datasheet: Boson Datasheet*. 27700 SW Parkway Ave. Wilsonville, OR 97070. https://www.flir.com/products/boson/.

[FLIR Systems Inc., 2019b] FLIR Systems Inc. (2019b). *Datasheet: Boson Engineering Data Sheet*, 330 edition. https://www.flir.com/products/boson/.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[Grini, 2019] Grini, S. V. (2019). Object detection in maritime environments. *MSc thesis, NTNU*.

[He et al., 2018] He, K., Gkioxari, G., Dollár, P., and Girshick, R. B. (2018). Mask R-CNN. *CoRR*, abs/1703.06870.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

[Helgesen et al., 2019] Helgesen, Ø. K., Brekke, E. F., Helgesen, H. H., and Engelhardtsen, Ø. (2019). Sensor combinations in heterogeneous multi-sensor fusion for maritime target tracking. In *2019 22th International Conference on Information Fusion (FUSION)*, pages 1–9.

[Helgesen et al., 2020] Helgesen, Ø. K., Brekke, E. F., Stahl, A., and Engelhardtsen, Ø. (2020). Low altitude georeferencing for imaging sensors in maritime tracking. In *proceedings of the IFAC World Congress*, Berlin, Germany.

[Hølland, 2019] Hølland, E. H. (2019). Maritime object detection using infrared cameras. *MSc thesis, NTNU, Trondheim.*

[Jiao et al., 2019] Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z., and Qu, R. (2019). A survey of deep learning-based object detection. *IEEE Access*, 7:128837–128868.

[Kamsvåg, 2018] Kamsvåg, V. (2018). Fusion between camera and lidar for autonomous surface vehicles. *MSc thesis, NTNU, Trondheim.*

[Kjønås, 2021] Kjønås, I. (2021). Detection of vessels in infrared images from a shore-mounted camera. *Specialization project, NTNU, Trondheim.*

[Landsnes, 2021] Landsnes, K. (2021). Weakly-supervised instance segmentation for improved detection in maritime environments. *Specialization project, NTNU, Trondheim.*

[Lin et al., 2017a] Lin, T., Goyal, P., Girshick, R. B., He, K., and Dollár, P. (2017a). Focal loss for dense object detection. *CoRR*, abs/1708.02002.

[Lin et al., 2017b] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017b). Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944.

[Lin et al., 2015] Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P. (2015). Microsoft coco: Common objects in context.

[Liu et al., 2016] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, page 21–37.

[Padilla et al., 2021] Padilla, R., Passos, W. L., Dias, T. L. B., Netto, S. L., and da Silva, E. A. B. (2021). A comparative analysis of object detection metrics with a companion open-source toolkit. *Electronics*, 10(3).

[Prasad et al., 2020] Prasad, D. K., Dong, H., Rajan, D., and Quek, C. (2020). Are object detection assessment criteria ready for maritime computer vision? *IEEE Transactions on Intelligent Transportation Systems*, 21(12):5295–5304.

[Prasad et al., 2017] Prasad, D. K., Rajan, D., Rachmawati, L., Rajabally, E., and Quek, C. (2017). Video processing from electro-optical sensors for object detection and tracking in a maritime environment: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 18(8):1993–2016.

[Redmon et al., 2015] Redmon, J., Divvala, S. K., Girshick, R. B., and Farhadi, A. (2015). You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640.

[Redmon and Farhadi, 2018] Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *CoRR*, abs/1804.02767.

[Rees, 2012] Rees, W. G. (2012). *Physical Principles of Remote Sensing*. Cambridge University Press, 3 edition.

[Ren et al., 2015] Ren, S., He, K., Girshick, R. B., and Sun, J. (2015). Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497.

[Russakovsky et al., 2015] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

[Sandler et al., 2018] Sandler, M., Howard, A. G., Zhu, M., Zhmoginov, A., and Chen, L. (2018). Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381.

[Schöller et al., 2019] Schöller, F. E., Plenge-Feidenhans'l, M. K., Stets, J. D., and Blanke, M. (2019). Assessing deep-learning methods for object detection at sea from lwir images. *IFAC-PapersOnLine*, 52(21):64 – 71. 12th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS 2019.

[Shijie et al., 2017] Shijie, J., Ping, W., Peiyi, J., and Siping, H. (2017). Research on data augmentation for image classification based on convolution neural networks. In *2017 Chinese Automation Congress (CAC)*, pages 4165–4170.

[Shorten and Khoshgoftaar, 2019] Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(60).

[Tan and Le, 2019] Tan, M. and Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946.

[Tan et al., 2020] Tan, M., Pang, R., and Le, Q. V. (2020). Efficientdet: Scalable and efficient object detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10778–10787.

[Taylor and Nitschke, 2017] Taylor, L. and Nitschke, G. (2017). Improving deep learning using generic data augmentation. *CoRR*, abs/1708.06020.

[Wang et al., 2021] Wang, C., Bochkovskiy, A., and Liao, H. M. (2021). Scaled-yolov4: Scaling cross stage partial network. *CoRR*, abs/2011.08036.

[Wang et al., 2019] Wang, C., Liao, H. M., Yeh, I., Wu, Y., Chen, P., and Hsieh, J. (2019). Cspnet: A new backbone that can enhance learning capability of CNN. *CoRR*, abs/1911.11929.

[Yu and Zhu, 2020] Yu, T. and Zhu, H. (2020). Hyper-parameter optimization: A review of algorithms and applications. *CoRR*, abs/2003.05689.

[Zhao et al., 2018] Zhao, Z., Zheng, P., Xu, S., and Wu, X. (2018). Object detection with deep learning: A review. *CoRR*, abs/1807.05511.

[Zhu et al., 2020] Zhu, H., Wei, H., Li, B., Yuan, X., and Kehtarnavaz, N. (2020). A review of video object detection: Datasets, metrics and methods. *Applied Sciences*, 10(21).