Freie Universität Berlin

Department of Mathematics and Computer Science

Thesis

In partial fulfillment of the requirements for the degree of:
Master of Science (M. Sc.)

in Data Science

Topic:

Enhancement of Underwater Images for Object Detection

submitted by:

Anton Haberland

Matriculation-Nr:	4981997
Date of birth :	18.07.1996
Born in:	Magdeburg

1. Corrector Prof. Dr. Daniel Göhring
2. Corrector Prof. Dr. Giancarlo Troni

Magdeburg, 16th of June, 2022

Abstract

Marine habitats are an increasingly relevant research field, and with more capable and affordable options for capturing images in underwater environments, the need for more effective pipelines to process these images becomes apparent. We compare different image enhancement methods, namely contrast limited adaptive histogram equalization(CLAHE), multi-scale retinex with color restoration(MSRCR), and a fusion-based approach on their efficacy in an object detection pipeline. Their specific use in the training and inferencing process with convolutional neural network(CNN) models are evaluated. In our setup, we build flexible pipelines to train several models with different enhancement strategies for the training dataset and assess their detection capabilities by measuring their inferencing precision with differently enhanced test datasets. We chose the regions with CNN(R-CNN) architectures Faster R-CNN and Mask R-CNN for our analysis, as they are widely used and deployed in all sorts of practical applications. We found that the use of these enhancement methods during the training phase results in better models, though their application in an inferencing pipeline is still inconclusive. Our data shows, that a significant subset of the images would gain from some form of enhancement as these methods show to mitigate some of the image degradations introduced by the underwater environment. We, therefore, argue with this work for the necessity of a reliable method that determines the best enhancement procedure for each image as part of an extended detection process.

Contents

Chapter 1: Introduction

1.1 Motivation

The deployment of cameras has a wide range of possible applications in the realm of marine sciences. Mounted on a Remotely Operated Vehicle(ROV) or Autonomous Underwater Vehicle(AUV) to aid the navigation process, dragged behind a boat for exploration, or positioned on the seafloor for monitoring a fixed area. These scenarios have obvious use cases for object detection that can replace tedious human labor or complement information gathered by other sensors. A human operator as well as an autonomous system that navigates an underwater vehicle relies on a multitude of different sensors to safely traverse difficult environments like the open sea or any marine habitat [1]. In this context, image enhancement can play a crucial role to make images more easily readable for a human controller and object detection to enable the autonomous system to make decisions based on image data [2] [3]. Besides that, several studies employ some form of camera monitoring of objects, individual animals, or full ecosystems with a range of different methods. One common approach is Baited Remote Underwater Video Systems(BRUVS) [4] which are stationary cameras monitoring the same location for an extended amount of time. These generate a large amount of video data but demand a costly labeling process with hours of work by domain experts to identify very specific objects in all the video frames. MaxN is a common metric that quantifies the occurrence of a species in a video by using the maximum number of individuals within a single frame. Applications like this are well suited for the deployment of object detection methods as no tracking of individuals over multiple frames is required. There is already a body of research working on this topic [5] though because of different requirements between projects, a lack of publicly available data, and the peculiarities of underwater imaging, no high-performing solutions have been developed that can be readily applied in most circumstances.

The mentioned peculiarities of underwater imaging are a wide range of image degradations caused by the environment. Light behaves differently when passing through the water instead of our atmosphere, different wavelengths are absorbed differently which in practice leads to an early loss in red followed by green chan-

nel information. The turbidity also varies widely caused by floating particles obstructing the view and influencing the image's sharpness, texture, brightness, and color distribution [6]. Another phenomenon is optical turbulences [7] which distort the image through the different temperature and salinity layers in sizable bodies of water. These peculiarities can significantly reduce the ability to correctly identify objects in images, not only for humans but also presumably for detection models. Showing that incorporation of image enhancement techniques in the training and inference pipeline can improve the detection accuracy and comparing strategies to do so, is the goal of this thesis.

1.2 Image Enhancement

Image enhancement is a broad field of research that aims to improve some aspects of the images for specific use cases. Image enhancement is considered to be a difficult problem, as there is no such thing as an objective solution for an optimally enhanced image, as based on your specific project, different image properties need to be enhanced. Figure 1.1 shows a basic example for image enhancement, with contrast limited adaptive histogram equalization(CLAHE)[8] and multi-scale retinex with color correction(MSRCR)[9] displayed.
Another difficulty in the field of image enhancement is the variety of underlying image degradations, that can be present in the frame caused by all sorts of influences during the capturing. This complexity of the problem leads to a vast range of solutions, all covering specific niches in the field. One class of algorithms works in the spatial domain and alters image contents based on the evaluation of pixel values. A common technique in this domain is the utilization of kernel matrices which are applied step by step to the source image. In those algorithms, the resulting pixel values are dependent on the values their neighboring pixels hold as well as on the configuration of the kernel matrix. Different effects can be achieved, for example, smoothing, edge enhancement, or denoising. Another common approach is shifting the image's color distribution to resemble a more desirable state. Those methods are called histogram processing methods, as the main way to inspect the color distribution of an image, is to generate its histogram representation. These and other techniques can be used and combined to construct more complex enhancement methods like in [10] for Magnet Resonance Imaging or [11] for underwater images.
A different approach for image enhancements, that is also used in common im-

plementations for methods like MSRCR, is utilizing 2D Fourier transformation to alter the image in its frequency representation. The requirement for a function to be fourier transformable is to be absolutely integrable, which makes it also applicable for images, as they can be thought of as functions $f : \mathbb{N}^2 \to \mathbb{N}$, with $f(x, y)$ being the image intensity at point (x, y) *with* $x \in [0, ImageWidth]$; $y \in [0, ImageHeight]$. The frequency representation is useful, as the transformation function is reversible and any change in the frequency domain can be observed in the restored spatial domain image. Two examples for that are shown in figure 1.2, where the ideal low pass filter and gaussian filter both mask out a circular shape in the frequency domain, which leads to a smoothing effect in the original image.

Figure 1.1: *Example for a differently enhanced image* ***Left****: raw image* ***Middle****: CLAHE enhanced image* ***Right****: MSRCR enhanced image (raw image from [12])*

1.3 Object Detection

Object detection is a part of the computer vision field, which aims to enable machines to extract information from images [13], that include the detection of movements, distances, or in the case of object detection, the exact positions of relevant objects. While early approaches were only able to recognize basic shapes, contemporary methods are capable of differentiating between several different classes and finding accurate bounding boxes as is shown in figure 1.3. One of the very early approaches is an algorithmic solution to the problem which utilizes edge or color information which can be obtained very efficiently, to find different kinds of shape or object types. Published in 1972 the Hough Transformation [14] is well suited to detect simple shapes like lines, circles, and curves but exponentially grows in computational cost with more complex shapes like ellipses. The general procedure includes first an edge detection algorithm to reduce the relevant pixel count and then for each remaining pixel, each config-

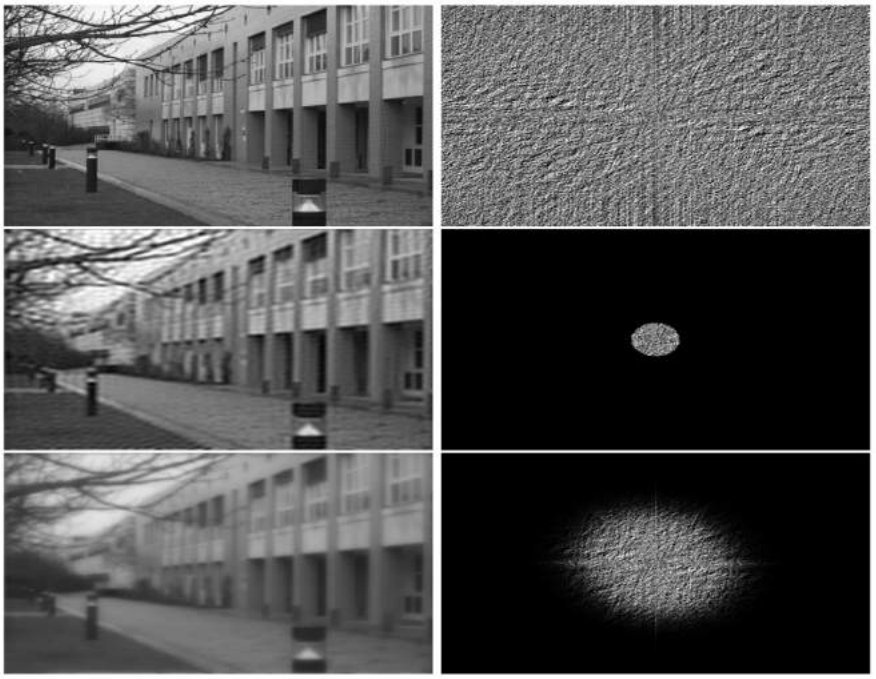


Figure 1.2: ***Left**: spacial domain images. **Right**: frequency domain images. **Top**: original image. **Middle**: Ideal Low Pass Filter. **Bottom**: Gaussian Low Pass Filter (raw image from [12])*

uration of the desired shape the pixel could lie on, is calculated. The result of the algorithm is those configurations, that are shared by many of the relevant pixels. As complex shapes require more parameters to be represented, these are often not feasible in terms of runtime and memory requirements. It should be noted, that effort has been made to optimize the algorithm for specific purposes. For example in [15] they have developed an efficient alteration of the algorithm, that can detect cylinders in point clouds.

Another classical approach called camshift, explained by Bradski et. al. [16], uses the color distribution of the target image section to efficiently find it in subsequent images, especially applicable in tracking objects in video data. To achieve this, the probabilities of pixels in a defined starting frame belonging to the desired object are calculated. These calculations quantify the similarity in color distribution between the current search frame and the target frame. The position of the search frame is iteratively moved until the probability does not continue to improve. The algorithm is also able to resize the search frame to

accommodate for different distances to the camera and tilt the frame to be able to detect the object even after different movements.
Those methods which are designed to find very specific shapes or objects are useful in those very specific applications but lack any kind of generalizability. While camshift would be able to find a single person on different images, it would struggle to recognize different persons or even the same person wearing different clothes. Therefore, methods that are able to reliably recognize objects of a specific class even with high variability in individual class members were a research goal of many teams in the computer vision community. In the early 2010s, the rapid development of deep learning-based approaches began, which completely overtook the field of object detection and is to this day in the process of producing new and more efficient models on a regular basis. The main idea of those methods is a feature extraction network, followed by some form of classifier, both usually trained in a supervised manner. The feature extraction network, as the name implies, is able to find a variety of abstract properties contained in the image, while the classifier then decides based on the provided features, whether or not one of the target classes is present. Most of the techniques were not particularly novel in the 2010s, though the high computational requirements for the training process limited the development to well-funded institutions. The success of AlexNet in 2012 [17] then showed that new and record-breaking models can be trained with only off-the-shelf graphics processing units, which kickstarted the rapid development during the following decade resulting in numerous state-of-the-art detection networks like you only look once(YOLO) or regions with convolutional neural networks(R-CNN).

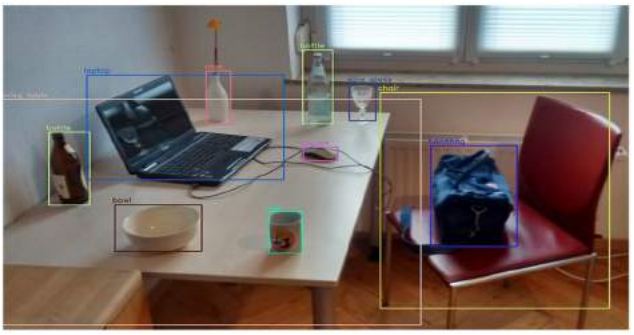


Figure 1.3: *Example for object detection [18]*

Chapter 2: Methods

2.1 State of the Art

For both problems, image enhancement and object detection exist several viable methods that perform well in comparable situations. With that range of options, it is necessary to explore the different approaches and limit the thesis to a select few.

For the enhancement of underwater images, there are several categories of non-commercial approaches that are explored in current research. The first is physical model-based approaches which try to account for the different behavior of light in underwater environments. They model different light absorption rates and diffusion patterns to calculate an alternative version of the image without these underwater image degradations. Many of these employ dark channel prior(DCP) [19] for dehazing which states that each small section on a haze-free image has at least one value in one of the color channels that is very close to zero. Chiang et. al. [20] proposed a method for underwater image enhancement that uses DCP in order to estimate a depth map of the scene, followed by a correction of the color channels based on the presence of artificial light, travel distances of the light and background light estimation utilizing the aforementioned depth map. While they achieve good results on a range of images, for optimal results, some calibration images should be captured before the actual data collection.

Non-physical model-based approaches, on the other hand, do not model any properties of underwater light propagation but usually only try to alter the image to conform closer to some desired state in terms of color channel distribution, histogram shape, or brightness thresholds. We will have a closer look at MSRCR[9], CLAHE [8] and a fusion-based approach [21] in this chapter. A third category is deep learning-based enhancement methods which mostly employ some form of generative adversarial network(GAN) arrangement in which one model learns to convert images captured above water to look like underwater images in order to train enhancement network in a supervised manner. Examples are UWGAN [22] which requires depth maps for the training of the GAN network and waterGAN [23]. In all publications, they proved the relevance

of their networks on datasets with images too small for object detection tasks. Meaning the application for our issue at hand would have required modification of the networks and a lengthy training process. A comprehensive comparison of different methods has been conducted by Li et. al. [24] where they found the fusion-based enhancement to perform best, though they also state that no method outperforms all the others in a consistent manner.
A comparison of contemporary deep learning object detection networks can be found in the very recent publication by Zaidi et. al. [25]. They found new transformer-based approaches like the model by Liu et. al. [26] to outperform more established CNN-based models in their experiments. These models, though well-performing are still a novelty in the field and we will focus on the practically proven models built on convolutional layers. Those more common methods, that are already in use in numerous practical applications, do not perform considerably worse in the aforementioned review and their high adoption rate makes them a good choice for experiments. Example architectures include popular approaches like YOLO [27], R-CNN [28] and also models like EfficientDet [29]. As image segmentation in the underwater domain would be desirable at some point, we choose to continue with R-CNN-based models, as Mask R-CNN [30] is able to provide this functionality. Also, Song et. al. [31] already showed that the use of image enhancement with MSRCR can have a positive influence on the segmentation capabilities of Mask R-CNN. In that light, we decided to use Faster R-CNN as it is the latest development in the R-CNN publications that only focuses on object detection and also Mask R-CNN models that we train without real segmentation data. For image enhancement, we only use methods where no extra training or additional data like depth maps are required. Therefore the non-physical model-based methods, namely CLAHE, MSRCR, and fusion are within the scope of this thesis.

2.2 Enhancement

2.2.1 CLAHE

Contrast limited adaptive histogram equalization is a continuation of the basic histogram equalization technique, which aims to transform the image in such a way, that the resulting histogram representation of the image covers the pixel intensity range more evenly (figure 2.1). By utilizing the cumulative distribution function(CDF) of the pixel intensities, the histogram is stretched with equation

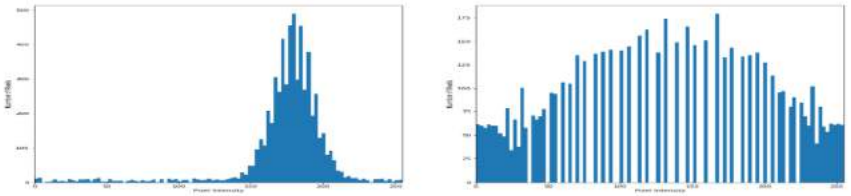


Figure 2.1: *Global Histogram Equalization*

(2.1) where I is the raw and R the enhanced image and MAX the maximum image intensity.

$$R(x,y) = (MAX - 1) * CDF(I(x,y)) \tag{2.1}$$

This method usually improves the image contrast but often leads to distortions because of different image regions having differently distributed pixel intensities and to an overenhancement of image noise.

Addressing this, adaptive histogram equalization(AHE) was proposed to enhance pixel intensities by exclusively regarding the intensity distribution of pixels in their immediate neighborhood. In the naïve approach, the transformation shown in equation (2.1) has to be calculated for each pixel with a different CDF of the respective neighborhood. This approach is very slow, as a new CDF has to be calculated for each pixel and therefore not feasible in many applications. Pizer et. al. proposed a solution in [8] where they proved that they could achieve equal results as the naïve approach, with a fraction of its computational cost. They fully divide the image into equally sized separate regions and only use equation (2.1) to calculate the enhancement of their respective center pixels. The remaining pixel values are enhanced by interpolation with those center points. For the majority of pixels, bilinear interpolation with the four closest center pixels is applied, while pixels close to the image boundaries are interpolated linearly with the two closest center pixels. Pixels in the corner of the image are not interpolated but enhanced with equation (2.1) using the CDF of their respective image region. Pizer et. al. also introduced a contrast limiting approach to limit noise amplification. They describe clipping the pixel intensities at a fixed value for each of the image regions and redistributing lost intensity equally over all the region's pixels (figure 2.2).

A sample of CLAHE enhancements is shown in figure 2.3 where the contrast enhancement is visible to different extends in all of the examples. There, the enhancement only made a minor difference in the subjective quality of the dark

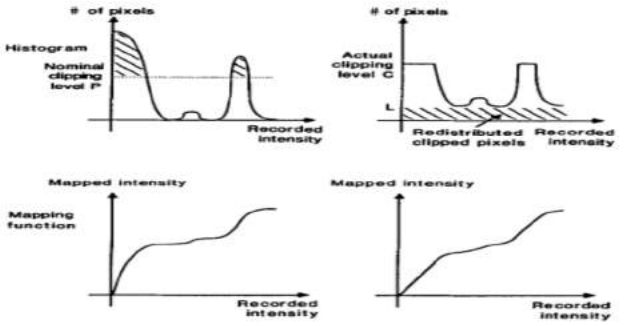


Figure 2.2: *Visualization of Clipping in Clahe Algorithm (figure from* [8])

image which depicts a broader trend with CLAHE and our dataset. Namely, low-quality images in which strong enhancements are desired, usually benefit less from CLAHE enhancements than images that can be considered good quality to begin with. Also, it should be noted, that in contrast to the other algorithms that will be discussed, CLAHE does not alter the overall color tone of the image while the other methods also aim for color correction.

2.2.2 MSRCR

The Multi-Scale Retinex with Colour Restauration algorithm is based on the Retinex theory of color vision by Edwin Land [35] in 1977 and the subsequent work on modeling his finding into usable algorithmic solutions for digital image processing. The theory builds on the assumption that an image consists of two independent parts. The reflectance and luminance part which retinex algorithms try to estimate and separate from each other. Based on this theory further research like hulbert et al.[36] started working on modeling these findings mathematically. Based on work like this, Rahman et. al. [37] proposed MSR which was able to enhance the color/lightness rendition and dynamic range compression at the same time. MSR has a comparable approach to the Difference of Gaussian(DOG) algorithm as convolution operations with a gaussian Kernel and subsequent subtraction from some form of the original image is required.
Equation (2.2) shows how the MSR enhancement is achieved, with R being the enhanced image, I the original image, i specifying the current color channel (blue, green, red), K being the number of scales, with each scale resulting in a different gaussian surround matrix S_k, the $*$ symbol being the convolution operator and W_k a weight value for which $\sum_{k=1}^{K} W_k = 1$ holds. S_k has the the

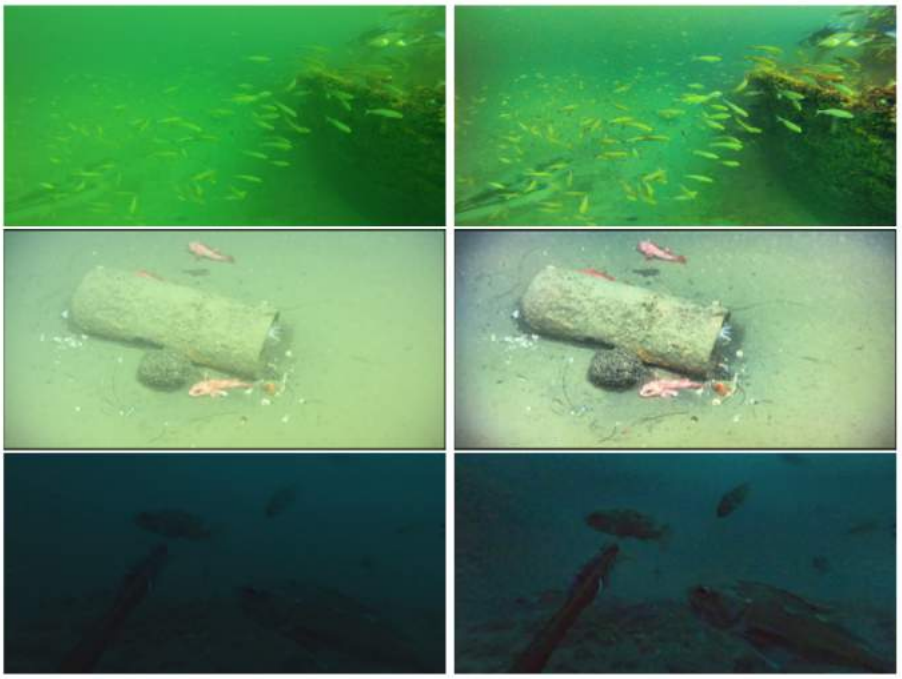


Figure 2.3: ***Left**: Raw Image, **Right**: CLAHE Enhanced Image(top from [32], middle from [33], bottom from [34])*

same dimensions as I_i and is calculated as shown in equation (2.3) with M_k chosen, so that $\sum_{a=0}^{X}\sum_{b=0}^{Y} S_k(a,b) = 1$ holds, considering that (X,Y) are the image dimensions for one color channel. The σ_k^2 for each of the K scales can be chosen based on the implementation, smaller values will lead to more dynamic range compression while larger ones are shown to enhance color constancy [37]. Choosing three different values like $\sigma_1^2 = 50$, $\sigma_2^2 = 100$, $\sigma_3^2 = 150$ is shown to enhance both of those traits.

$$R_i = \sum_{k=1}^{K} W_k(lg(I_i) - lg(S_k * I_i)) \tag{2.2}$$

$$S_k(x,y) = M_k * exp(-(x^2 + y^2)/\sigma_k^2) \tag{2.3}$$

As it is required to perform convolution operations between two equally and sufficiently large matrices, it is useful to convert this calculation into the frequency domain. When defining $\mathfrak{F}()$ as the Fast Fourier Transform(FFT) function, and $f()$ and $g()$ as two absolutely integrable functions, then $\mathfrak{F}(f() *$

$g()) = \mathfrak{F}(f()) \cdot \mathfrak{F}(g())$ holds. Based on this, the calculation from (2.2) can be reformulated to (2.4) which has has a better runtime for the matrices in this specific case.

$$R_i = \sum_{k=1}^{K} W_k(lg(I_i) - lg(\mathfrak{F}^{-1}(\mathfrak{F}(S_k) \cdot \mathfrak{F}(I_i)))) \tag{2.4}$$

Further work on this approach has shown that some images will appear desaturated after enhancement by MSR. To address this shortcoming, Rahman et. al [9] introduced MSRCR which adds a color correction term to the results of MSR like equation (2.5)

$$R_{MSRCR} = CR \cdot R_{MSR} \tag{2.5}$$

The color correction term has different definitions based on the publication. In Song et. al. [31] they introduced the following version for a color correction term. As they specifically work with underwater images as well, it is also suited for the problem this thesis explores. Its definition is shown in equation (2.6), g and α are constants that have been determined empirically by the authors and set to 1 and 128 respectively.

$$CR_i = g \cdot [lg(\alpha \cdot I_i + 1) - I'] \tag{2.6}$$

$$I' = lg(\sum_{i=1}^{3} I_i + 1) \tag{2.7}$$

The sample of MSRCR enhanced images in figure 2.4 shows some of the usefulness of the method. A green color shift in the original image gets usually corrected and more realistic-looking color distribution is achieved. When lacking a significant color shift in the clean image, the result tends to be skewed too much into the red channel with occasional strong red artifacts on objects close to the camera. Some of this might be explained by the light-absorbing properties of water, which absorbs low wavelength light i.e. red light earlier than higher wavelength light. Especially in artificially lit underwater scenes, close objects will still contain information in their red color channel while further ones will have lost most of it due to the absorption.

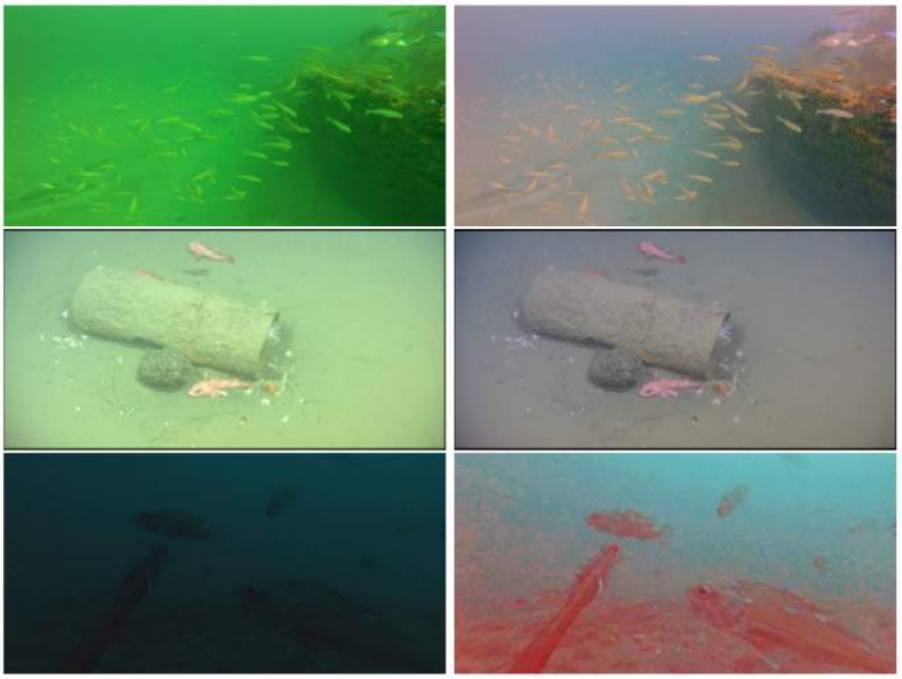


Figure 2.4: ***Left**: Raw Image, **Right**: MSRCR enhanced image(top from [32], middle from [33], bottom from [34])*

2.2.3 Fusion Based

Fusion-based approaches are a broad group of algorithms that aim to combine the outputs of multiple separate methods to utilize each of their respective advantages. In 2017 Ancuti et. al. [21] published a method specifically tailored to the problem of underwater image enhancement with an emphasis on color correction. They start with a custom white balancing method (equation (2.8)), as they found the existing methods tend to introduce some unwanted artifacts when applied to underwater scenes. In the equation, I^w is the white balanced image that only differs from I by its enhanced red channel I_r^w. I_r and I_g are the normalized red and green channels of the raw images with values in range [0,1] and $\bar{I}_g$ and $\bar{I}_r$ are the mean values of the respective normalized matrices. In this formula we denote a matrix exclusively filled with ones as 1.

$$I_r^w = I_r + \cdot(\bar{I}_g - \bar{I}_r) \cdot (1 - I_r) \cdot I_g \tag{2.8}$$

From this white-balanced output, two separate images are created and independently modified to be combined to a final result image at the end. The first is a gamma-corrected and the second an unsharp masked version of I^w. For both of these images, I_1 and I_2 a weight map is then constructed with contrast, saliency, and saturation values. The contrast is calculated by applying a laplacian filter on the luminance channel of the images, which will enhance edges and contours. For saliency they use an approach published by Achanta et. al. [38] in which they use the LAB representation of the image from which they subtract the mean values of its respective channels and follow this up by calculating the euclidean norm between the 3 channels. Finally, the image saturation is obtained by subtracting the luminance channel of the image from the three color channels and combining their values as shown in equation (2.9) with I_r, I_g and I_b being the three color channels and I_l the luminance channel.

$$W_{sat} = \sqrt{1/3\left[(I_r - I_l)^2 + (I_g - I_l)^2 + (I_b - I_l)^2\right]} \tag{2.9}$$

The contrast, saliency, and saturation values need to be combined into a single weight map for both of the two input images. To achieve this all weight matrices are first normalized by dividing by 255 and then combined to the weight maps W_k by summing $W_{k,sat}$, $W_{k,sal}$ and $W_{k,cont}$ with k indicating to which of the two input images the weight map belongs. Those values are then normalized by equation (2.10), where the divisor is the sum of both previously calculated weight maps. The δ is used to prevent divisions by zero and in the paper, by Ancuti et. al. [21] is chosen to be 0.1. In our investigations smaller terms lead to improved results and we decided on setting δ to 0.001. Our approach also differs from the publication by the θ term which we define as $\theta = 1 - (1/K)$ with $K = 2$ being the number of different images. It distributes the values of the weights matrices around 1 rather than $1/K$ which helps to reduce the overall darkening of the result images.

$$\bar{W}_k = \theta + (W_k + 2 \cdot \delta) / \sum_{k_2=1}^{K} (W_{k_2} + \delta) \tag{2.10}$$

At this stage, there are the gamma-corrected image I_1 and the white balanced image I_2 plus their respective weight maps $\bar{W}_1$ and $\bar{W}_2$. These matrices are now combined to a single output image by a multiscale fusion process involving gaussian and laplacian pyramids formulated in equation (2.11). There L is the

maximum depth of the pyramids which should depend on the size of the image and can usually be set to a value between 5 and 10. Each successive level in the gaussian pyramid is a downsampled version of the previous level and each level in the laplacian pyramid is the difference between successive levels of this gaussian pyramid. To add the differently sized matrices of the L levels they need to be upsampled to the size of the original image before summation which is omitted in the definition of equation (2.11) for better readability.

$$R = \sum_{k=1}^{K} \sum_{l=1}^{L} (G_l(W_k) \cdot L_l(I_k)) \tag{2.11}$$

The resulting image R is the solution of the fusion algorithm and some samples are depicted in 2.5. In comparison to the other two presented algorithms, fusion alters the underlying color distribution most and addresses color shifts in the raw image more aggressively. In doing so, it consistently introduces a range of artifacts often caused by the enhancement of existing noise. Also when examining low-light input images, fusion tends to exacerbate the problem as the results are usually darker than the respective inputs.

2.3 Object Detection with R-CNN

2.3.1 Overview

The first R-CNN model was proposed in 2014 by Girshick et. al. [39] and was able to achieve a 30% improvement in Mean Average Precision(mAP) on the VOC2012 dataset over the previous top model. While research on image classification at the time already heavily utilized the power of convolutional neural networks(CNN) since the publication of 2012 by Krizhevsky et. al. [17], it still found limited use in the field of object detection. The difference is simply, that image classification aims to recognize objects within the image and return the probabilities of target objects being in that image, while object detection also localizes those objects within the frame. R-CNN was therefore proposed to bring the performance gains CNNs promised to the field of object detection. The main idea of the model is to extract regions of interest within the image and feed those regions separately into an image classifier and ultimately decide based on confidence thresholds, which regions are considered to be objects. While their approach is not bound to a specific region proposal method,

Figure 2.5: ***Left**: Raw Image, **Right**: FUSION enhanced image (top from [32], middle from [33], bottom from [34])*

they decided on using the selective search algorithm by Uijlinks et. al. [40]. They extract approximately 2000 regions per image and feed them into the network proposed by Krizhevsky et. al. [17] for feature extraction. This is then followed by a support vector machine classifier(SVM) to decide based on the 4096-dimensional feature vector how likely the presence of the target classes for this image segment is. This structure makes R-CNN inherently a two-stage detector as the region proposal is separated into distinct processes, while models like YOLO [41] combine the stages into one process.

They finally employ a bounding box regressor to improve the accuracy of the identified objects to better fit the ground truth bounding boxes of the training dataset with a method proposed by Felzenszwalb et. al. [42]. While this model outperformed competing models at the time, it suffered from a lengthy runtime during the training, and more importantly also during the inference. They state that they achieved 13 s/image with a top-of-the-line graphics card, which makes it far from useful for any real-time application which would require around 30 images/s at least. This limitation cannot be overcome by improving hardware

alone, as the described design requires a high number of region proposals per image to be passed through the network. Because of this limitation, all parts of the model were reworked over the following years and significant improvements in terms of accuracy and more crucially, performance was achieved. A more in-depth look into the separate parts and their development over time with their respective impacts are explored in the following sections.

2.3.2 Convolutional Neural Network (CNN)

As the following sections will deal with specific implementations of Convolutional Neural Networks and the advantages of different iterations, the underlying ideas and methods of CNNs need to be clarified beforehand, as well as the use of terminologies clearly specified.
CNNs are neural networks mainly used in the context of image processing but can be applied to any application, in which the consecutive uses of kernel matrices can lead to the extraction of increasingly complex features. The unique characteristic of CNNs is as already implied in their name, the use of convolutional layers. A very simplified depiction of standard operations in such a network is shown in figure 2.6, where the first operation is a convolution with a 3x3 kernel on a single channel 6x6 image with a stride of 1 and no padding. In practical applications, multiple of these convolutional operations with several different kernels per layer are successively used, with the shape of the kernels, the stride, and padding type determining the height and with of the output. As the convolution operation is a series of matrix dot products, the kernel, and the input sections need to be equally sized at every step, therefore the kernel needs to have a depth equal to the depth of its direct input matrix. A depiction for kernel and image sizes is shown in figure 2.7 where you can also find the connection between kernel count and output depth. During the training, these models alter the values in the kernel matrices to optimize the feature extraction capabilities over the course of the training. Convolutions in combination with other common layer types like ReLU (equation (2.12)) and max pooling are basically a tool to drastically scale down the input while preserving relevant contents in some abstracted form. This scaled-down version, also called a feature vector is then often used by some form of classifier to identify objects of interest.

$$ReLU(x) = max(0, x) \tag{2.12}$$

A schematic for a real-world application is shown in figure 2.8 with AlexNet/ ImageNet [17]. It uses five convolutional layers with each output having the ReLU function applied, and interspersed with three max-pooling layers. When comparing the input size of $227 \cdot 227 \cdot 3 = 154587$ with the dimensions of the resulting feature vector after the convolutions $6 \cdot 6 \cdot 256 = 9216$, the dramatic reduction in dimensionality becomes obvious and we can also show a reduction in trainable parameters in comparison to this model without the convolutions. If you would apply the first fully connected layer directly to the input image, you would have $227 \cdot 227 \cdot 3 \cdot 4096 = 633188352$ trainable parameters alone. The whole convolution process, plus the first fully connected layer only requires 41494560 weights[43], which is a reduction by a factor of 15. While this not only drastically improves the training time and memory requirement, it is also important to note that the feature extraction process itself is necessary for the classification to begin with.
For training, an optimization function like stochastic gradient descent is utilized to find an optimal combination of weight values that minimizes some error value. This error is some difference between the model's prediction and the ground truth. Then with backpropagation, each weight is updated in a process that iteratively updates the trainable parameters from last to first layer.

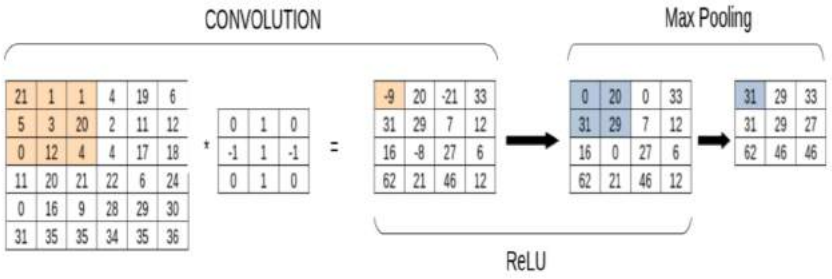


Figure 2.6: *Simplified illustration of common operations in a CNN*

2.3.3 Region Proposal

The first major iterations of the R-CNN model family, including R-CNN [39] and Fast R-CNN [45] utilize the already mentioned selective search algorithm [40]. This method first clusters connected pixels based on their light intensity value together by adopting a graph-based image segmenting algorithm by Felzenszwalb et. al. [46]. Based on this initial segmentation, similar regions

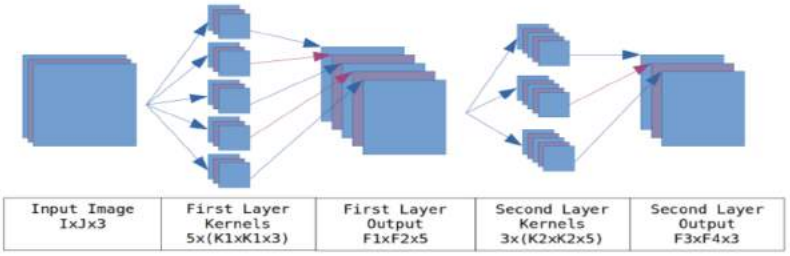


Figure 2.7: *Two successive convolutional layers on a multi channel image (e.g. RGB). The kernel counts (5 and 3) are arbitrarily chosen in this example*

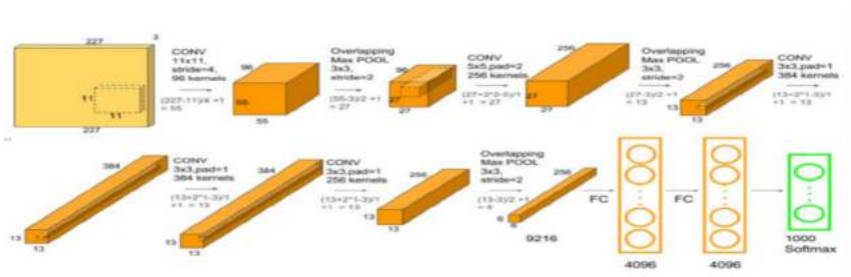


Figure 2.8: *Schematic representation of the AlexNet model (figure from [44])*

are progressively merged into larger regions, using a similarity measure that includes the region's color, texture, size, and relative position to each other (shown in figure 2.9).

The difference between R-CNN and Fast R-CNN is how the region proposals are fed into the succeeding network. Fast R-CNN can speed up the training process by more than 45x mainly by not passing each proposal into the network separately and applying the full image classification process to all of the proposals. They instead feed the full image into the feature extraction network and provide the region proposal coordinates as a separate input. The feature matrix is then constructed in a way, that a specially designed Region of Interest(RoI) pooling layer can extract regions of interest within the network using the provided region coordinates from the selective search algorithm. This results in several feature vectors, each representing the contents of a single region of interest which are then each classified in the final part of the model.

While the speed enhancement of Fast R-CNN is very significant, the problem remains that selective search, especially with its common CPU implementations,

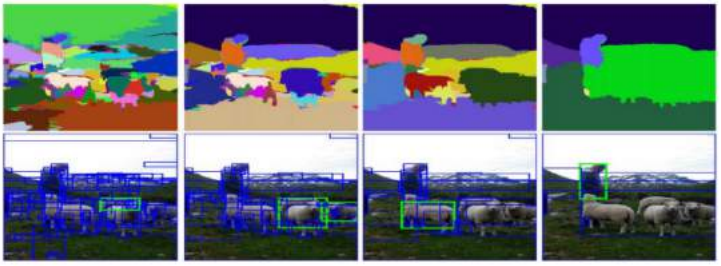


Figure 2.9: *Selective search algorithm. From left to right: progressively combining similar regions. (figure from [40])*

is a major bottleneck in the entire pipeline. For that reason, Ren et. al. [28] published Faster R-CNN, an evolution of Fast R-CNN that incorporates the whole region proposal process into a small region proposal network(RPN) that runs parallel to the feature extraction and shares the convolutional layers to save on memory and computation. Instead of extracting relevant regions from the raw image, the regions are extracted from the feature map after the image has been passed through several convolutional layers. The number of shared layers depends on the backbone network chosen for the model.
In figure 2.10 the structure of the RPN is depicted. It firstly convolves the feature map with $k = 9$ differently sized anchor kernels. These anchor kernels have the aspect ratios 2:1, 1:2, and 1:1 with three different sizes per aspect ratio, to account for differently sized and orientated objects on the image. Each position is mapped by those anchor frames onto a lower-dimensional feature vector on which 2 parallel fully connected layers follow, one trained to determine an objectness score and one to improve the bounding box coordinates.
This method improves the inference time by 10x and more, depending on the utilized backbone model, but makes the training process less straightforward as the region proposal network needs to be trained as well. The fact that the region proposal and the classification share layers and are dependent on each other requires a more sophisticated training procedure.

2.3.4 Feature Extraction

The feature extraction network also called the backbone is interchangeable in all of the R-CNN models as minimal work is required to make different backbones work in the overall process. The main changes that have to be made are adding, removing, or resizing some of the layers to make feature maps compat-

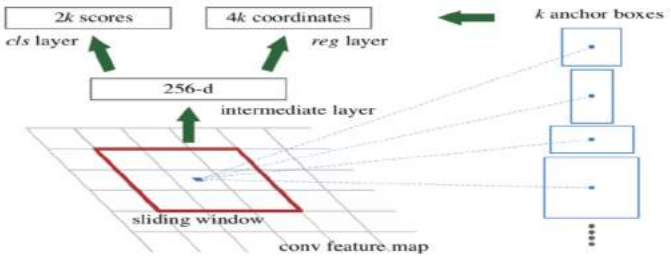


Figure 2.10: *Region proposal network(RPN) (figure from [28])*

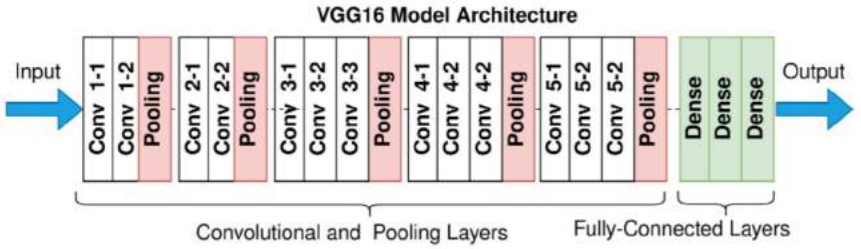


Figure 2.11: *VGG16 network architecture (figure from [47])*

ible with the other parts of the pipeline, like the classifier or the RoI pooling layer. Therefore the authors of all the R-CNN iterations publish their versions with the at-the-time top-of-the-line feature extraction networks.
The original R-CNN from Girshick et. al. [39] used AlexNet which we already covered in a previous section. Girshick then utilized in his next publication, Fast R-CNN [45] the VGG16 network [48] that builds on the basic ideas explored in the previous section, but has a higher accuracy by building a deeper network with more layers (figure 2.11). The addition of an RoI-pooling right after the feature extraction is the most notable addition of Fast R-CNN. This layer extracts only the parts of the whole feature map from the CNN, that correspond to a separately calculated region of interest. These regions are individually divided into a fixed number of equally sized sections and each section is max pooled with an equally sized filter matrix. With this method, a different feature vector for each region of interest is generated and can be used for classification.
Faster R-CNN [28] published their model with a ResNet50 [49] backbone. This backbone, published in 2016 by He et. al., was a milestone in the development of convolutional neural networks as they introduced the idea of residual blocks that made the construction of even larger networks feasible. They stated that

deeper networks, though computationally more expensive, should, in theory, be as accurate as their shallower versions as the additional layers could just be identity mappings and therefore not change the end result. In practice though, when far exceeding 20 layers, models start performing worse with growing depth. Introducing skip connections in a network model addresses this problem. A very basic depiction of a residual block is in figure 2.12. The premise of a block like this is to use convolutional layers to extract features from an input, but add the input to the resulting feature map. This leads to each block not learning weights that will result in an optimal output itself, but weights that are trained to optimally contribute to its input. Mathematically speaking, when a traditional network section transforms an input matrix x into $F(x)$, then we can define the difference/residual between these two as $R(x) = F(x) - x$. While the traditional network learns to produce $F(x)$, the network within the residual block learns to output $R(x)$. This reduces the ability of singular weights to negatively influence the overall result as outputs from previous layers are more likely to be preserved.

The example of figure 2.12 is just a depiction of a very simple version of a residual block, and there can be several operations within this block, which do not even need to preserve the dimensions of the feature vector. In that case, the skip connection needs to account for these changes by transforming the feature vector with methods like max-pooling as well. Figure 2.13 is an example architecture for a network model utilizing residual blocks.

The integration of the region proposal into the network structure has already been touched on in the section *Region Proposal*. The difference this change makes in the whole training process is far from trivial as both the classification and the RPN would train the shared convolutional layers independently. Solving that, the method proposed by the authors firstly trains the full RPN, including the layers it would normally share with the classifier. For that, they only consider proposals that have a sufficiently large intersection over union(IoU) with the ground truth data as valid object proposals. By using the accuracy of these predictions with stochastic gradient descent and backpropagation, they train the RPN weights. The second step uses the trained RPN to train a Fast-RCNN model, that has the same backbone as the desired final Faster R-CNN model. After sufficiently good results with the Fast R-CNN, they use its weights to initialize the full Faster R-CNN model. This model can then be trained in 2 stages, firstly by only tuning the weights that only belong to the RPN, and finally, the classifier part with the shared weights frozen.

Another iteration of the R-CNN model is Mask R-CNN from He et. al. [30] from 2017. While its main contribution is adding image segmentation to the model's capabilities, which will be explored in a later section of this work. They also incorporated new developments in the feature extracting network, namely a Feature Pyramid Network(FPN) [50] (figure 2.14). It is based on the idea that higher layers have a higher resolution while lower layers contain semantically more relevant features. The classifier can profit from feature maps with both of those attributes, so upsampling the deepest feature maps and iteratively combining them with the higher feature maps retain the semantically complex features while expanding the resolution. This new bottom-up constructed pyramid is also well suited to streamline the RoI-Pooling process, or in the case of the Mask R-CNN network, RoI-Align. RoI-Align works similar to RoI-Pooling but they avoid any rounding operations when extracting the anchor boxes from the feature maps, as the rounding operations significantly affect the results of the subsequent segmentation. Using one of these RoI operations on the pyramid, 3 equally sized kernels, but with different orientations (1:1, 2:1, 1:2) are extracted from each feature matrix of the bottom-up pyramid. As the resolution of the matrix increases with each step in the pyramid, the size of the extracted region of interest also does. Meaning, that instead of using nine anchor kernels on one single matrix, three kernels are used on multiple differently sized matrices which will result in a similar amount of differently shaped regions of interest.

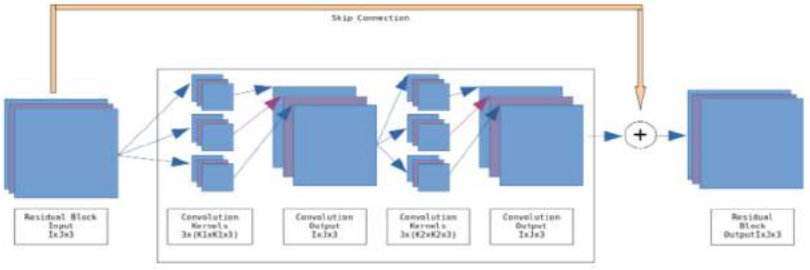


Figure 2.12: *Visualization of an example residual block. Within the block classic convolution operations are applied to the input matrix. The difference is the addition of the input matrix to the output of the convolutional part. Only an example, more and different operations can be part of a residual block.*

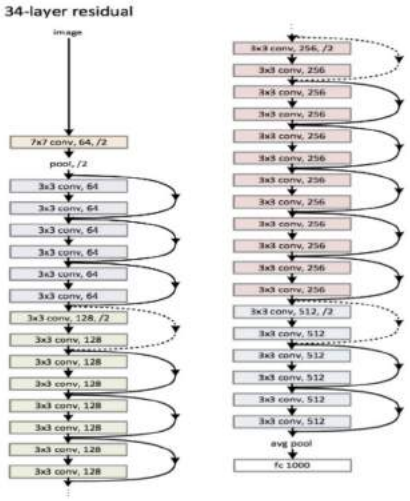


Figure 2.13: *Flowchart of the ResNet34 architecture. Dotted residual connection change matrix dimension. (figure from [51])*

2.3.5 Classification

The classification part of these models were developed during the progress as well. The original R-CNN was published with a support vector machine(SVM) per target class. A support vector machine is a simple supervised classification method that aims to find an optimal N-dimensional hyperplane to separate an N+1-dimensional dataset by a specified target label, see figure 2.15 for a simple example. In the R-CNN implementation, each SVM is fitted onto a 4096 dimensional set of outputs, as these are the dimensions of the feature map created by AlexNet. Each Support vector machine is trained to decide if the feature map implies the presence of one single target class.

Fast R-CNN does not continue using the SVM approach, as it complicates the training process to fit these models in a separate process after the CNNs training. Instead, they opt for two parallel fully connected layers, the first outputs K+1 probabilities with K being the number of target classes. Each probability is the likelihood of the specific class being present in the particular region of interest. The second output layer is trained to output a $K \cdot 4$ sized feature matrix that contains the bounding box offsets $b^k = (x_1^k, y_1^k, x_2^k, y_2^k)$ for each of the target classes to enhance the exact coordinate of the current region of interest.

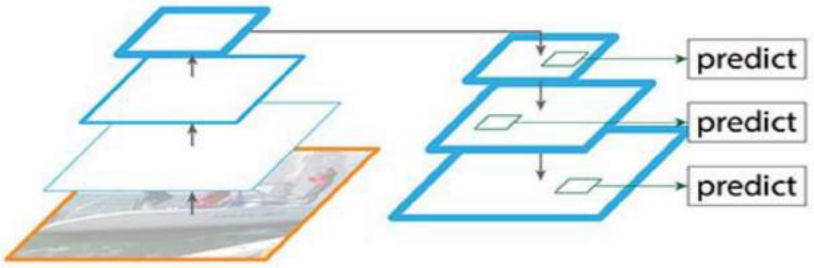


Figure 2.14: *Feature Pyramid Network (figure from [50])*

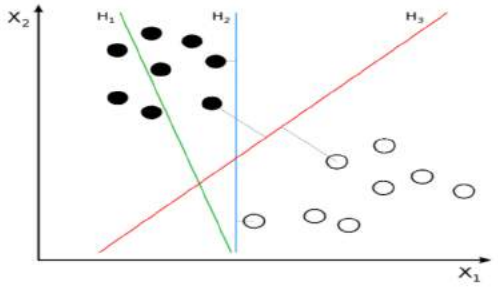


Figure 2.15: *Support vector machine on a 2 dimensional feature map. H3 as optimal separation between the groups (figure from [52])*

2.3.6 Image Segmentation

Image segmentation expands on the idea of object detection, instead of only finding bounding boxes, the exact contours of target objects are found by labeling each pixel according to the object they belong to (figure 2.16). This functionality was introduced in 2017 by He et. al. with the Mask R-CNN network [30]. For this, a third output layer is added parallel to the classification and bounding box regression layer. This new layer is trained to create an $K \cdot m \cdot n$ output per region of interest, being a binary mask for each of the K classes, mapping pixels to being either part of the object or the background. To achieve image segmentation with this model, the training process needs to be tweaked again to accommodate the new output layer. Firstly, the training data needs to contain the ground truth segmentation data in order to calculate the difference with the predictions. Then for every mask prediction that corresponds to a bounding box prediction having a sufficiently large IoU with the ground truth, the binary cross-entropy loss between the predicted and actual mask is calculated.

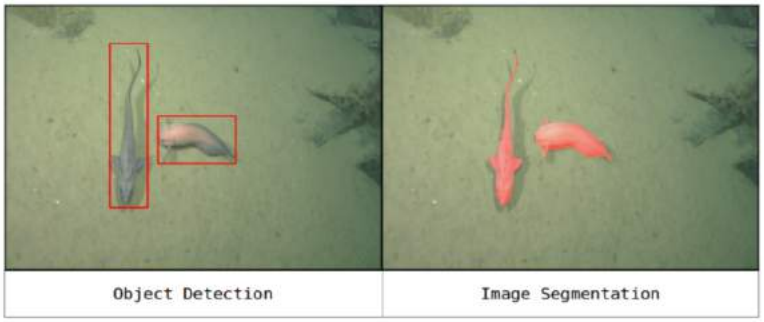


Figure 2.16: *Object Detection vs Image Segmentation (raw image from [33])*

Chapter 3: Experimental Setups

3.1 Training Setup

3.1.1 Training Dataset

For the training process images from multiple sources are included in the dataset. In Cutter et. al. [53] they collected and labeled a number of underwater images for detection purposes. As they focus on the detection of fish as well, the dataset can be used in this project with only some minor work on the label format.
The main fraction of the data is obtained from the FathomNet API [33]. It contains a vast collection of labeled images of underwater habitats with bounding box annotation on the species level. To obtain a set of images depicting fishes, it is necessary to call the API for samples of specific species. To find out which species can be counted as fish, we needed to filter all the available species by their membership to a specific group in their taxonomy. This could be done by querying the species name to the API of the Ocean Biodiversity Information System(OBIS)[54]. For this dataset, only species belonging to the phylum Gnathostomata were considered. This grouping should contain every species one would consider as fish, though many species beyond that, as for example mammals, birds, and for that matter all land-dwelling vertebrates also belong to that category. After an evaluation of the resulting dataset, no image contains labels for species that are clearly not fishes, though no expert has controlled the dataset for difficult to classify outliers.
Additionally, the Estacion Costera de Investigaciones Marinas (ECIM)[34], a marine research institute of the Pontificia Universidad Católica de Chile provided video material from one of their current research projects. These videos did not contain any bounding box annotations, so some of the frames had to be chosen and labeled by hand. While choosing the images, we focused on images with many fishes at the same time, as the previous datasets contain only a few comparable examples.
While the FathomNet API and the collection from Cutter et. at. are useful sources for our dataset, we found a considerable amount of images having incomplete bounding box labeling (figure 3.1), and the number of samples we

gathered from these sources made it infeasible to amend all the missing labels. In order to prevent the model to find fish in every image presented, we also add samples with a second non-fish class. Our entire training set therefore also contains 280 images of individuals from the phylum Cnidaria(jellyfish etc.) but our evaluations will not focus on the detection quality for this second class.

Figure 3.1: *Samples from Fathomnet Database [33] with missing bounding box labels*

3.1.2 Training Goals

This project explores possible accuracy gains of object detection networks on underwater images when applying algorithmic image enhancements beforehand. We focus on the previously discussed enhancements of CLAHE, MSRCR, and Fusion and compare the two related detection networks Faster R-CNN and Mask R-CNN. All implementations were done in python3.7 using common libraries like numpy, cv2, scipy, pandas, and PyTorch as the machine learning framework. The training pipeline was run on google colab as the provided GPU capacity is sufficient for the chosen models and the dataset at hand. The training loop for the Faster R-CNN is custom built with PyTorch, while Mask R-CNN as a comparison model is trained using the detectron2 library [55]. We only use dummy segmentation data for Mask R-CNN as shown in figure 3.2 as real data is currently not accessible in large enough quantities. In projects like those conducted in ECIM, models being able to segment objects are highly desirable for numerous different studies, like behavioral analysis. When more data is available our implementation can easily be converted to use real instead of dummy data at a later stage. As the main focus lies on the Faster R-CNN network, the object detection results of Mask R-CNN are used to confirm trends

we found concerning the influence of different enhancement methods.
Both Faster R-CNN and Mask R-CNN are initialized with a pre-trained ResNet50 Backbone plus a Feature Pyramid Network(FPN) neck, as described in section 2.3. All models are trained using statistical gradient descent with a learning rate of 0.003, momentum of 0.9, and a weight decay of 0.0001.
To compare the effect of the enhancement methods we train numerous models per network architecture. A raw, CLAHE, MSRCR, and a fusion model, each of them trained with the whole training set being enhanced by the respective method. We furthermore train a randomly enhanced model, where for each image in the training set, a random enhancement is chosen in each epoch. See table 3.1 for an overview of the 10 trained models.
The training dataset contains 1229 images from different sources as described in section 3.1.1, which is enough to train these networks up to a reasonable accuracy for our analysis, though not to confidently use it in any practical application, caused by the complexity of the problem itself. Additionally, the several incomplete labels in the dataset will negatively influence the weight optimization in the model and require special consideration when analyzing the results in the following sections.

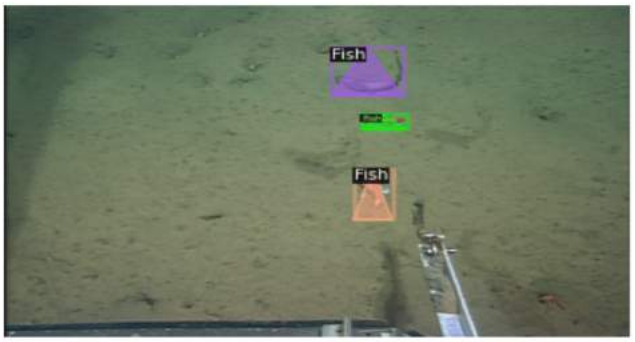


Figure 3.2: *Example for dummy masking (raw image from [33])*

3.2 Testing Setup

3.2.1 Average Precision Score (AP)

For evaluating the model's object detection capabilities, we use the average precision(AP) [56] score per model on the test dataset. Figure 3.3 shows an

TRAINING:	Model-1	Model-2	Model-3	Model-4	Model-5
Faster R-CNN	Raw Data	Clahe enhanced data	MSRCR enhanced data	Fusion enhanced data	Random enhanced data
Mask R-CNN	Raw Data	Clahe enhanced data	MSRCR enhanced data	Fusion enhanced data	Random enhanced data

TESTING:	Model-1	Model-2	Model-3	Model-4	Model-5
Raw Test Set					
Clahe Test Set					
MSRCR Test Set					
Fusion Test Set					

Table 3.1: *The 10 models and the datasets they are trained with. The Testing table shows the relevant testing scenarios.*

exemplary comparison of three prediction scenarios and an estimation of their relative AP scores. This score is a numerical value quantifying the relationship between precision and recall of the model under different confidence thresholds. When using a model, there are a few parameters to consider, one is the confidence score per detected object that is given by the model. Each confidence score is a numerical value between zero and one on how confident the model is, that the associated prediction is a real target object. When deploying a model, the confidence threshold needs to be chosen high enough that false positives are minimized, but low enough that more true objects are detected. When comparing models, it is not trivial which threshold to choose for each model, as both will have different optimal thresholds, and even at these thresholds, the comparison is not as straightforward as expected, as one model might have the higher recall and the other a higher precision with their optimal parameters. Another threshold when evaluating a model is deciding how close the predicted bounding box has to be to the real one to consider it as a true positive. For that different intersection over union(IoU) thresholds are usually considered (figure 3.4). But as well as for the confidence score, an optimal IoU threshold for that model would need to be found for comparisons.

The AP score can incorporate these two scores into its calculation. We use the version of the COCO AP implementation from the python library mean-average-precision [57]. In that implementation, the precision and recall of the whole dataset with different IoU thresholds from 0.5 to 1 in 0.05 steps are calculated. The calculations for precision and recall are shown in equation (3.1)

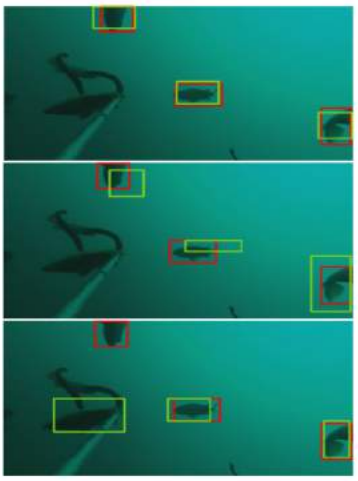


Figure 3.3: *Comparison of predictions with different AP scores(red = ground truth, green = prediction).* ***Top****: High AP score* ***Middle****: lower AP score because of worse bounding boxes(lower IoU scores)* ***Bottom****: lower AP score because of False Positive(single green) and False Negative(single red) (raw image from [34])*

and (3.2) with TP being the number of true positives, FP the number of false positives and FN the number of false negatives, or in this case, the number of ground truth objects that have not been found. When plotting these precision-recall pairs, you get a curve in which the precision decreases with increasing recall. For the Average Precision value, you then obtain the precision values at specific recall intervals, in our case at each recall from 0 to 1 in 0.01 steps, and average the resulting precisions.
With more than one relevant object class this evaluation would be done for all classes separately and the average of these class specific AP scores would be the mean average precision score(mAP).

$$Precision = \frac{TP}{TP + FP} \tag{3.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.2}$$

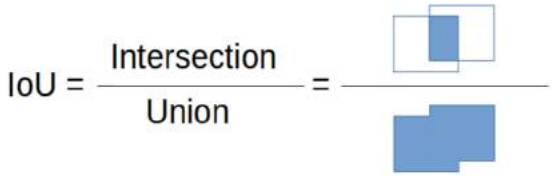


Figure 3.4: *Intersection over Union (IoU)*

3.2.2 Testing Scenarios

We test the precision of the model by calculating the AP score of the model on the test dataset (35 images) under different enhancement strategies to make relevant comparisons. We only test the single enhancement models, meaning the models that have only been trained with one enhancement method, on the test dataset enhanced with the respective method. Only the models that are trained with all enhancement methods present during the training will be evaluated with all enhancement strategies, see table 3.1 for an overview.
We will also choose the best performing model and check for each image in the training and test dataset which enhancement strategy results in the best object detection result for this specific image. With this, we aim to find how possible improvements are distributed as it might be possible, that the use of enhancement methods can yield a big improvement for some images, while those images not profiting from the enhancement only perform a little worse as a side effect. This may help inform the definition of an enhancement strategy for the inference process as the gains in some images might outweigh the losses in others. We also explore why some images might benefit while others do not and if underlying image properties may predict the use of one or another enhancement method.

Chapter 4: Results

4.1 Faster R-CNN and Mask R-CNN

The first insight gained from the numerous models is the comparison of the two applied network architectures. To assess the network's object detection quality, we apply them to a testing dataset containing 35 images from the same sources as the training set though only consisting of pictures unknown to the models. The performance of each model is quantified by the AP score that captures the quality of the network on all the images of the test set. Figure 4.1 combines the results into one graph for easy comparison between the two architectures. Each curve only displays the performance of the best model built with this architecture at the given epoch. To reiterate, we created numerous models per architecture, each trained with a different enhancement strategy for the training data. When testing the different models, we apply the same enhancement technique on the test dataset as has been on the training data. Besides a small lead on early epochs, Faster R-CNN achieves a higher AP score on the test dataset with a maximum after around 70 epochs and for Mask R-CNN at around 90 epochs. From figure 4.2 we can examine the different models for each architecture. It shows the different model performances for Faster R-CNN on the left and for Mask R-CNN on the right. The randomly trained model was tested in different configurations and this plot only displays its best values per training stage. Not all models have been trained for the same number of epochs, and not all early stages for the models have been saved during the training process, which makes data for some of the models extend further to the left or right of the plot.

Not only the better performance of Faster R-CNN is noteworthy from these results, but also the similar behavior of both models concerning the different enhancement strategies. The Fusion-based enhancement results in models with the worst AP score in both of the architectures while the randomly enhanced models perform best with the not enhanced raw and then the CLAHE enhanced models following closely.

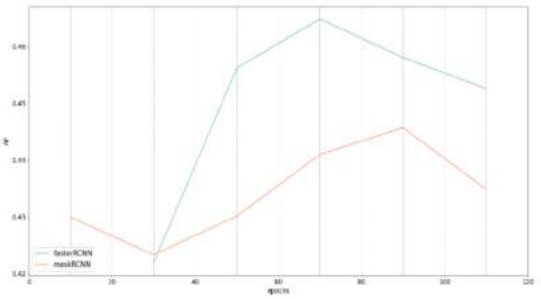


Figure 4.1: *Comparison Faster R-CNN vs Mask R-CNN. Best models per architecture over different training stages*

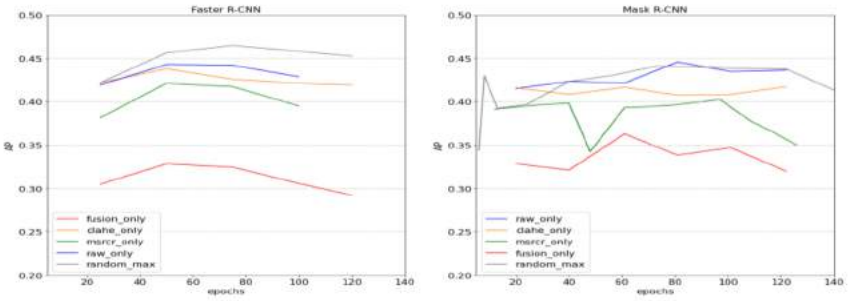


Figure 4.2: *Comparison Faster R-CNN vs Mask R-CNN over different training stages(epochs).* ***Left****: AP curves for Faster R-CNN models* ***Right****: AP curves for Mask R-CNN models. (random_max refers to the best test performance for the randomly trained model)*

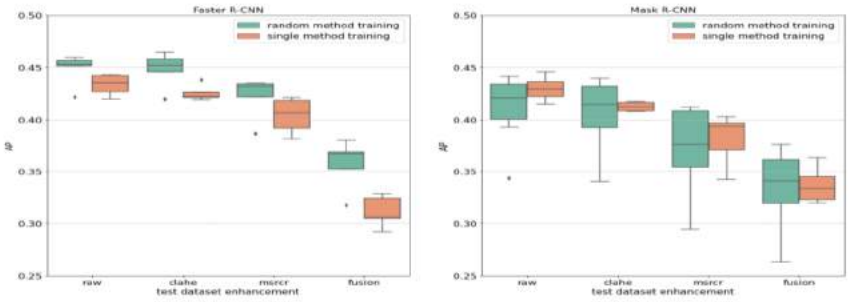


Figure 4.3: *Comparison between models only trained with one enhancement method against model trained with all enhancement methods. The test dataset used for comparison is enhanced by the respective method given on the horizontal axis. The boxes include models after different training stages(epochs).* ***Left****: Faster R-CNN* ***Right****: Mask R-CNN*

4.2 Training Strategies

The comparison of different training strategies from figure 4.2 already shows a performance lead of the models that are trained with the random enhancement strategy. To understand the impacts of the enhancement methods, we evaluate these random enhanced models in multiple test runs with a different enhancement method in each run. Figure 4.3 shows the AP scores of these different runs in direct comparison with the respective single method enhanced networks. In other words, we first plot the AP scores of the different stages of the random enhanced model on the raw test dataset against the model that has been trained with only the raw training dataset. Then the same comparisons are done for the three enhancement methods CLAHE, MSRCR, and fusion with all these for both Faster R-CNN and Mask R-CNN.

The randomly enhanced model consistently outperforms any single enhanced model in every testing scenario for Faster R-CNN. They also have a higher or similar AP score than their Mask R-CNN counterparts, while the difference for the single enhancement networks between Faster and Mask R-CNN is not as pronounced.

The single best score is achieved by a Faster R-CNN model with a randomly enhanced training dataset that is evaluated over a CLAHE enhanced test dataset which is just barely an improvement over the same network over the raw test dataset. These results suggest a positive influence of incorporating several enhancement techniques into the training pipeline though not necessarily their usefulness during the inferencing process. From this result, one could reasonably assume that the use of MSRCR and fusion during the inferencing process reduces the model's capability to recognize the target objects within the image, while CLAHE brings very little improvement over keeping the images in a raw state. While this is cumulatively the case for this specific and arguably small test dataset, we need to investigate the extent to which these methods improve or worsen the detection quality on single images and if a generalization can be made on which types of images would gain or lose from any method.

4.3 Inference Strategies

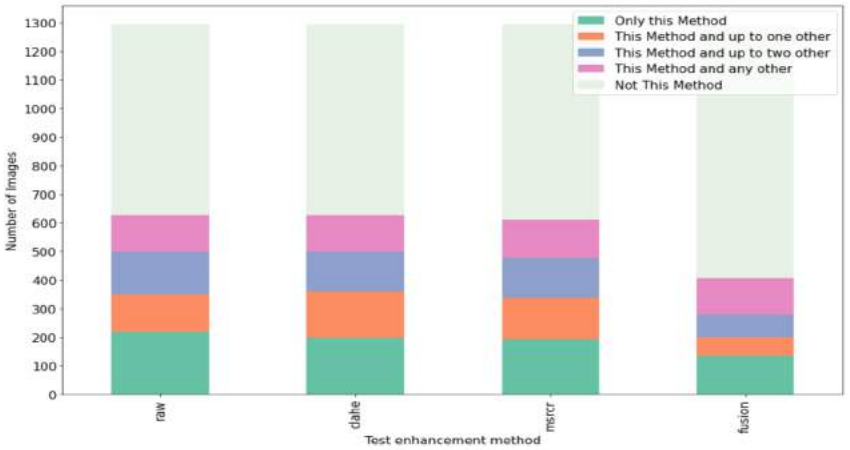
To calculate the influences of the methods on the detection process on singular images, we calculate the average precision for each image with each enhance-

ment method on a single detection model. For the following analysis, we use the randomly enhanced Faster R-CNN-based model after 75 training epochs as it leads to the highest AP scores on the test dataset. We use the full training dataset for the evaluation because we believe that it can still provide useful information about the differences in the enhancement methods, even though the model has been trained on these images. After obtaining four precision scores per image, one with each of the three enhancement methods and a baseline score for the raw image, the results show all methods have a positive influence on some images. Figure 4.4a shows how many images improve with which enhancement method, with the different colors depicting the set of images that have multiple enhancement methods achieving an equal highest precision score. All entries in green, being the number of images achieving their highest precision with only a single method, add up to 744 pictures or 60% of the 1294 entries sized training dataset. A slim majority of these perform best when kept in a raw state. This on the one hand shows that no enhancement during the inferencing performs overall the best when only one method can be used. On the other hand, only 17% of the entire dataset (figure 4.4b) performs worse with any of the enhancement methods, and subsequently, 83% can with at least one of these methods match the performance of the raw image version or outperform it. The raw image underperforms in 49% of samples and therefore an inference strategy that only uses the raw images would lose prediction quality on almost half of the images.

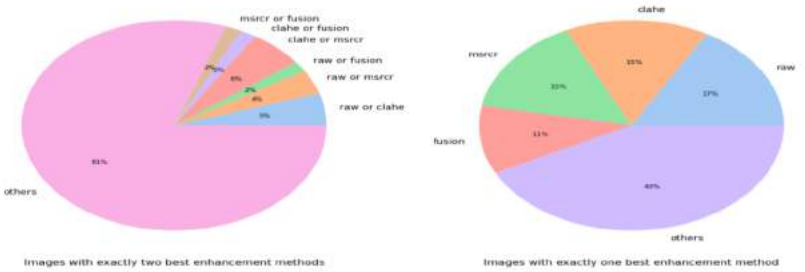
To further investigate the extent to which the enhancement methods influence the detection precision, it is also important to see how much is gained or lost with the application of the methods. A possible scenario would be that one method achieves a very high gain in precision with the images that it benefits and just a minuscule loss with those that benefit from staying raw. To quantify this, we calculate the precision deltas per image as follows $deltaClahe = AP_{Clahe} - AP_{raw}$; $deltaMsrcr = AP_{Msrcr} - AP_{raw}$; $deltaFusion = AP_{Fusion} - AP_{raw}$ for each image. In figure 4.5 these deltas are displayed in a Boxplot for each of the 3 enhancement methods. Every entry above 0 is an image that gained in precision when using the enhancement method given on the horizontal axis, while every entry below 0 is an image where the particular method resulted in a loss in detection precision. Doing this analysis on the training dataset might lead to unreliable results as it has already been trained on all versions of the images. Therefore we also perform the same analysis on the significantly smaller though unknown test dataset, which seems to confirm the trends in the training

dataset. The worst performing method is once again the fusion-based enhancement method. While some images gain detection precision from it, a broad application during inferencing would decrease the model's accuracy by a substantial margin for a majority of the images. The results for CLAHE and MSRCR on the other hand are not as clear, as most of the samples are distributed around zero, meaning that the increase or decrease in detection quality caused by the method is either very low or none. Both methods have very similar results, with CLAHE being slightly more concentrated around a delta of zero. The plot of the 35 images test dataset from figure 4.5 indicates a lower median AP score for the MSRCR method, though further analysis suggests this is only due to the low number of samples in this set. A pairwise T-Test with the AP values of the CLAHE test method results in a p-value of 0.15 substantiating the observation that both groups are equally distributed. Adding to that, every pairwise T-Test involving the AP deltas of the fusion method results in p-values below 0.01, further cementing the previous results, that fusion performs measurably worse in our experiments than the other methods. With these results, viable inferencing strategies could be the use of either CLAHE, MSRCR, no enhancement, or some combination of them. When using one of these two methods, or choosing them randomly per image, overall an equally good detection quality to no enhancement could be achieved. When enhancing all images in some form, the data indicates a worse detection performance in a number of images. Identifying those images might reveal some underlying properties that make them less susceptible to the enhancement methods.

When examining images with outlier AP scores, one commonality that does not reveal problems in the enhancement method, but rather the problems in the underlying data is shown in figure 4.6. Most fishes on these images are not labeled as ground truth, so a model recognizing more fish on these images correctly would be scored worse, as those fish would be considered false positives in the AP scoring. All these three images have relatively low AP deltas for CLAHE as well as MSRCR, meaning that the raw images perform better in this detection model. When doing the same analysis but with more complete ground truth labels (figure 4.7), the AP values shift in favor of the enhancement methods. In all 3 cases, the MSRCR enhanced images achieve the highest AP score afterward, showing that the lack of labels not only limits the model's training capabilities but also directly influences the results we gather from our experiments. After an analysis of the complete dataset, we found that from the 1294 images, 218 do not have the complete ground truth labeling, of which 154 only have up to five



(a) *Number of images profiting from specified enhancement method (or raw) based on precision score. An image can have multiple equally best methods. Based on how many it has, it is grouped into the corresponding color*



(b) ***Left**: Proportion of images having exactly two enhancement methods (or raw) as best performing. **Right** Proportion of images having exactly one enhancement method (or raw) as best performing*

Figure 4.4

missing bounding boxes and 64 images exceed 5 missing ground truth labels. When removing these images from our analysis, the overall distribution does not change significantly with the only notable difference being a slight increase of the AP delta mean for the CLAHE method. The other observation is a slight shift of the samples with outlier AP deltas, which now, after the removal of incomplete samples, shift slightly more in favor of the enhancement methods.
When comparing the subjective effect of the methods with the resulting detection quality, quantified by the AP score, we sometimes find discrepancies between the perceived enhancement and the model's detection precision. When looking at figure 4.8 we can see in the first example a reduction in image qual-

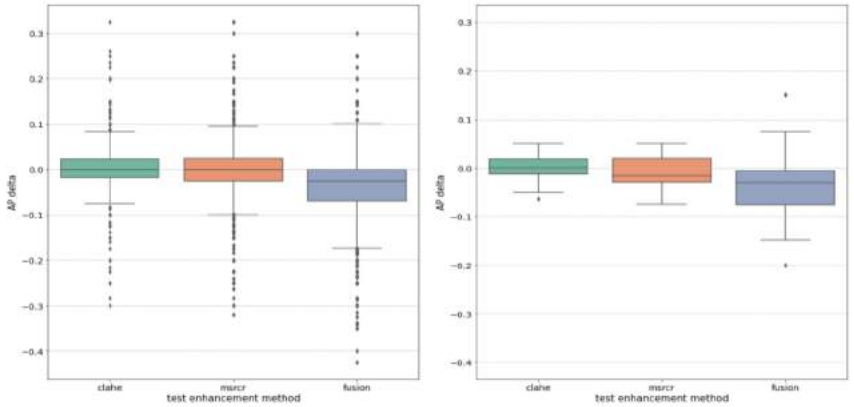


Figure 4.5: *AP score difference between enhanced and raw version for each image. Positive delta shows better detection on enhanced image and negative delta shows better detection on raw image.* ***Left****: Training dataset(1294 images)* ***Right****: Test dataset(35 images)*

ity with the MSRCR method that also translates into a lower AP score. The second example depicts a barely visible fish behind agitated sediment that has its contours revealed with the application of MSRCR and CLAHE, which also results in improved detection with the model. The third example in the figure on the other hand shows the discrepancy, while a human observer perceives a better image quality with the two methods, the best AP score is achieved with the raw image.

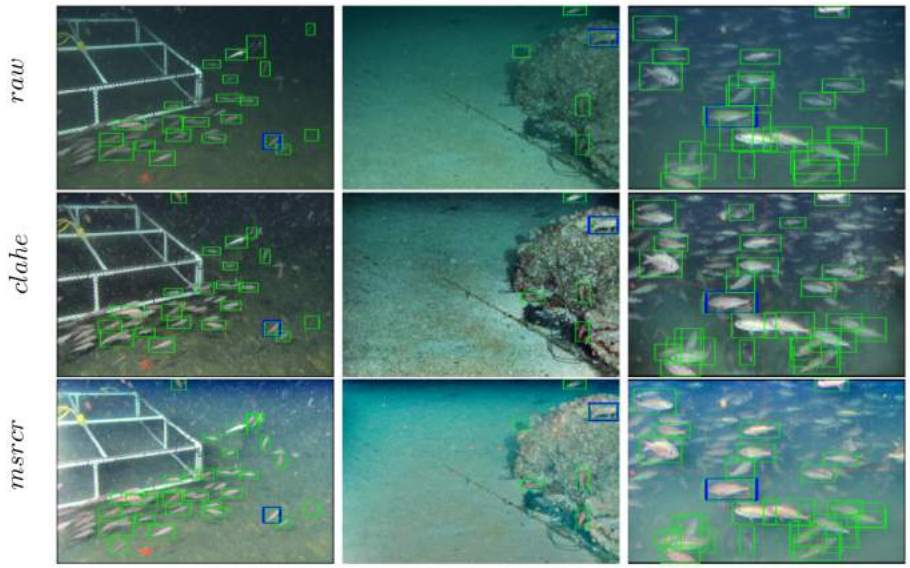


Figure 4.6: *Images with negative AP delta values, indicating better performance on raw image version.* ***Blue****: ground truth bounding box* ***Green****: prediction. Insufficient ground truth labeling*

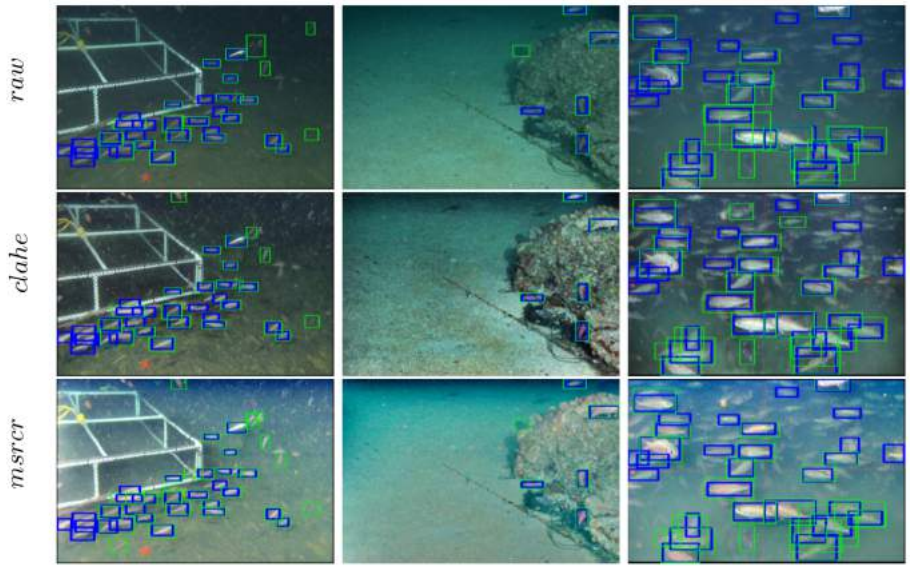


Figure 4.7: *Images with more complete ground truth data(harder to recognize fishes still not labeled).* ***Blue****: ground truth bounding box* ***Green****: prediction*

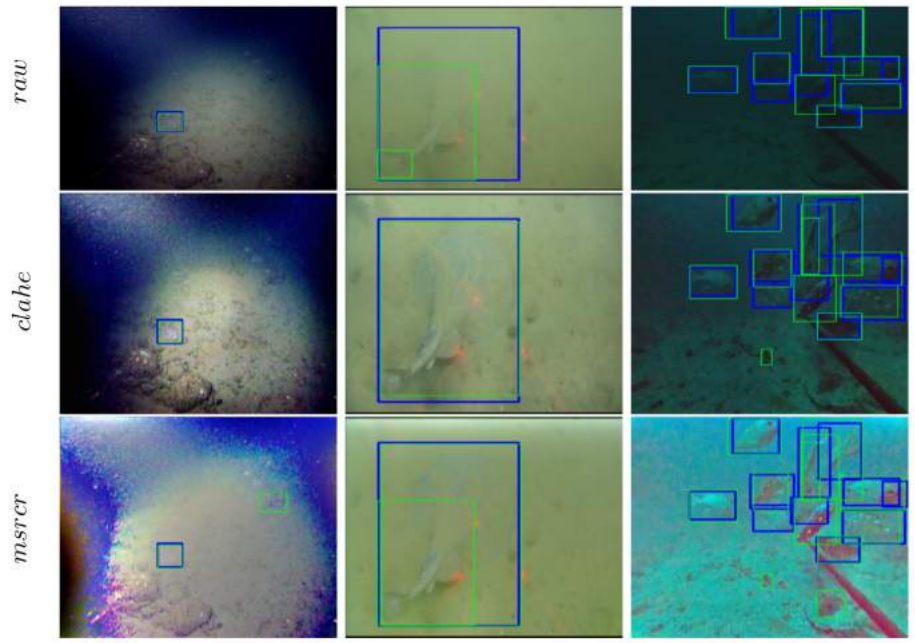


Figure 4.8: *Some detection examples,* ***Blue****: ground truth bounding box* ***Green****: prediction* ***Left****: Example where image quality is negatively influenced by MSRCR* ***Middle****: Example where MSRCR and CLAHE enable correct detection of the object* ***Right****: Enhanced images have better subjective quality, though best AP score for the raw version*

Chapter 5: Discussion

5.1 Insights

We were able to show that the right use of enhancement methods during the training is a useful strategy to improve the model's overall detection precision. We have no definitive answer to why this improvement is achieved, but we believe the cause is twofold. On the one hand, there are some images that benefit from the enhancement methods where the target objects are more easily distinguishable. From these images, relevant features are more likely to be extracted and recognized as such. This knowledge can then be used to already make better predictions in later epochs when the same image is reintroduced in one of its worse versions. The other driver of performance gain during the training is the expansion of the dataset. When using 3 additional versions per image, we virtually quadruple the training set. Strategies like that are called data augmentation and are a common practice to inflate the data especially when only limited training samples are available. While this artificial expansion does not bring the same effects as adding the same amount of completely new and unknown samples, it still enables the model to be hardened against noise, or different distributions of image parameters like color, contrast, or sharpness [58]. The comparison of Faster R-CNN and Mask R-CNN clearly favored the older Faster R-CNN network model in our experiment. This can partly be explained by the use of a more advanced backbone network than the authors used at the time of publishing in 2015. The authors of Mask R-CNN explored the advantages of using an FPN-ResNet50 as a backbone for this type of model, which improved their detection results, implying that replacing the older backbone of Faster R-CNN with this newer structure will also improve Faster R-CNN. The only differences remaining in the two model architectures are the difference between RoIPool from Faster R-CNN and RoIAlign from Mask R-CNN, which should not significantly contribute to this discrepancy, and the addition of a mask error to the overall error term in Mask R-CNN. This error quantifies the difference between the actual object mask and the predicted object mask. As our data only includes dummy object masks, we hypothesize that the limited correlation of the dummy mask with the actual object features regularly intro-

duces skewed error terms that might lead the model in suboptimal directions. Another factor that could influence our results, is the use of different frameworks for building both models, while both utilize PyTorch, we use detectron2 for the Mask R-CNN network and a custom training loop for Faster R-CNN. Both approaches should not differ in theory, as hyperparameters were chosen to match, though some minor differences in implementation could be responsible for some of the observed discrepancies. The single best-performing model we obtained is a Faster R-CNN network, trained on randomly enhanced data after 75 epochs.

From the methods we chose, all experiments show an advantage of MSRCR and CLAHE over the fusion-based approach. We showed that models trained with CLAHE enhanced or MSRCR enhanced training dataset have a performance almost on par with the model trained on the raw dataset, while the fusion-based approach falls short in any metric we tested.

The evaluation of different inferencing strategies showed that no single method strategy performs well enough that the other methods should be ignored. While the dataset in its raw state has a slight edge against the enhanced versions, we found that 49% of the images would achieve a higher detection score with at least one of the enhancement methods, leading us to the belief that a comprehensive strategy on how to preprocess images can have tremendous value for the complete pipeline.

5.2 Limitations

The major limitation of our experiment is the small number of samples in our training and test dataset, combined with the inconsistencies in their labeling. This causes an insufficient learning process that results in models with precision and recall too low for a practical application. While a model built for practical application was not the goal of this work, this unreliability also influences the results we obtain as we can never be sure that detection errors are caused by the low image quality, or by the limited capacity for generalization of the model. This extends to our analysis of which image prefers which enhancement method for optimal object detection. If we were to make predictions on other images based on this data, we could not claim to have found the optimal preprocessing strategy for any object detection pipeline, but only the optimal strategy for our limited model. We also tested and were not able to confirm strategies to pick

the optimal method before the inferencing process. All the data we gathered by comparing the detection results of the 4 different image versions, was not able to establish useful correlations between underlying image properties and the enhancement method with the highest AP score. The idea was to decide based on image metrics like sharpness, brightness, or the specific underwater image quality metrics UIQM [59] and UCIQE [60] which enhancement method would result in the highest AP score. The lack of any useful correlation might be attributed to the lack of actual correlation between the efficacy of the enhancement methods and the image parameters we chose, but might also stem from decisions in our experimental design.

The choice of enhancement methods might be one reason, the difference between the enhanced and the raw images might not be relevant for the detection process and other methods, those applying physical models or neural networks themselves, might alter the images more usefully for this particular problem. Another issue might be an incorrect or incomplete implementation for MSRCR or fusion, as we were not able to find useful source codes and reimplemented them from the descriptions in their respective publication. The results of Li et. al. [24] suggest some problems with our version of the fusion-based algorithm, as they found it to be one of the best algorithms to enhance the subjective quality of underwater images, while we found several examples in our experiments where the enhanced image seems to be degraded in comparison to the raw version.

Furthermore, there is the question if the evaluation of object detection quality of single images is best expressed with the AP score. This score is designed for standardized comparisons of different models on the same test dataset. Our analysis involves in one experiment the calculation of such a score for every single image. We found that the AP score shows some unfavorable behaviors for our analysis in some edge cases, which we believe are less noticeable with the more common applications of this metric. The number of false-positive detections by the model does not, or just very slightly, influence the resulting AP score for some images. This can be observed for images with a small number of ground truth objects, that are confidently and precisely detected by the model. As these samples will have a constant and high precision value over most of the recall range, a precision average that is mostly independent of the recall values results. This behavior is innate to the design of the AP score, as it uses the model's precision and recall at different threshold values to determine an average precision. Looking at these edge cases, when the confidence

and the IoU thresholds increase, the recall increases too, growing faster when fewer false detections are made by the model. The precision value on the other hand does not change for most of the threshold combinations, as all correct detections have high confidence and a high IoU with the ground truth label. When then averaging the precision values for the final AP score, the number of false detections will have had a very limited influence on this score for these specific cases. Any enhancement strategy that would be able to eliminate many false positives would not find these improvements fully displayed in our analysis.

5.3 Outlook

While we could show some positive effects of image enhancement in an object detection pipeline, our results only provide a narrow view into this problem. Any further research on this topic relies on the availability of a larger and more reliably labeled dataset in order to train a more robust model. The Monterey Bay Aquarium Research Institute (MBARI) [61] is currently working on the improvement and expansion of their fathomnet database [33] that has already been utilized in this work. As their labeling also includes species-level annotation, it might be possible with a growing dataset to not only label fish, but also differentiate between species or to find completely different organisms in various scenes.

Another interesting approach can be the utilization of additional information like altitude of capturing, image depth map, or water temperature, as this additional data can be used in physical model-based enhancement algorithms or as a deciding factor for which enhancement to choose for the detection pipeline. A dataset like that could be systematically generated in an experimental setup to generate images under certain light, turbidity, and depth conditions, with easily simulatable object types from different angles and distances. With an approach like this, a wide range of different scene compositions can be created and used to complement data captured in real ecosystems.

Without access to such a rich dataset, other experiments can be conducted to build on the work we have done so far. Though for any continued work we believe a reliable dataset without missing labels is required in order to remove one point of failure that has to be accounted for during the analysis. To ensure that no missing labels remain, any obtained dataset would probably need to be

controlled and subsequently, any missing labels added, which requires a significant time investment, depending on the size and quality of the dataset.
Different methods besides CLAHE, MSRCR, and fusion might grant additional insights, especially GAN-based approaches seem to be promising candidates for further analysis.Examples like WaterGAN [23] already provide pre-trained models but usually require some work to incorporate into the pipeline, as these models are trained on datasets with small-sized images, probably too small for usefulness in an object detection pipeline.
Altering this pipeline by comparing different architectures like YOLO[27] or EfficientDET[29] can provide additional information as it would be interesting to see if they will confirm the observations we now made with the R-CNN models.

Bibliography

[1] James C Kinsey, Ryan M Eustice, and Louis L Whitcomb. A survey of underwater vehicle navigation: Recent advances and new challenges. In *IFAC conference of manoeuvering and control of marine craft*, volume 88, pages 1–12. Lisbon, 2006.

[2] Fausto Ferreira, Gianmarco Veruggio, Massimo Caccia, Enrica Zereik, and Gabriele Bruzzone. A real-time mosaicking algorithm using binary features for rovs. In *21st Mediterranean Conference on Control and Automation*, pages 1267–1273. IEEE, 2013.

[3] Fausto Ferreira, Diogo Machado, Gabriele Ferri, Samantha Dugelay, and John Potter. Underwater optical and acoustic imaging: A time for fusion? a brief overview of the state-of-the-art. In *OCEANS 2016 MTS/IEEE Monterey*, pages 1–6, 2016. doi: 10.1109/OCEANS.2016.7761354.

[4] Matthew D Taylor, Jessica Baker, and Iain M Suthers. Tidal currents, sampling effort and baited remote underwater video (bruv) surveys: are we drawing the right conclusions? *Fisheries Research*, 140:96–104, 2013.

[5] Md Moniruzzaman, Syed Mohammed Shamsul Islam, Mohammed Bennamoun, and Paul Lavery. Deep learning on underwater marine object detection: A survey. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 150–160. Springer, 2017.

[6] Felipe Codevilla, Joel De O Gaya, N Duarte, and SSCC Botelho. Achieving turbidity robustness on underwater images local feature detection. *International journal of computer vision*, 60(2):91–110, 2004.

[7] Weilin Hou, Sarah Woods, Ewa Jarosz, Wesley Goode, and Alan Weidemann. Optical turbulence on underwater image degradation in natural environments. *Applied optics*, 51(14):2678–2686, 2012.

[8] Stephen M Pizer, E Philip Amburn, John D Austin, Robert Cromartie, Ari Geselowitz, Trey Greer, Bart ter Haar Romeny, John B Zimmerman, and Karel Zuiderveld. Adaptive histogram equalization and its variations. *Computer vision, graphics, and image processing*, 39(3):355–368, 1987.

[9] Zia-ur Rahman, Daniel J Jobson, and Glenn A Woodell. Retinex processing for automatic image enhancement. In *Human Vision and Electronic Imaging VII*, volume 4662, pages 390–401. SPIE, 2002.

[10] Hardeep Kaur and Jyoti Rani. Mri brain image enhancement using histogram equalization techniques. In *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pages 770–773. IEEE, 2016.

[11] Cosmin Ancuti, Codruta Orniana Ancuti, Tom Haber, and Philippe Bekaert. Enhancing underwater images and videos by fusion. In *2012 IEEE conference on computer vision and pattern recognition*, pages 81–88. IEEE, 2012.

[12] FUBerlin Takustraße 9 - Wikimedia. URL "`https://commons.wikimedia.org/wiki/File:FU_Berlin,_Takustra%C3%9Fe_9,_Institut_f%C3%BCr_Informatik_w.jpg`". [Online; accessed 12-June-2022].

[13] Computer Vision Definition. URL `https://www.ibm.com/topics/computer-vision`. [Online; accessed 07-April-2022].

[14] Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1): 11–15, 1972.

[15] Tahir Rabbani and Frank Van Den Heuvel. Efficient hough transform for automatic detection of cylinders in point clouds. *Isprs Wg Iii/3, Iii/4*, 3: 60–65, 2005.

[16] Gary R Bradski. Computer vision face tracking for use in a perceptual user interface. 1998.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[18] Detected with YOLO - Wikimedia. URL "`https://commons.wikimedia.org/wiki/File:Detected-with-YOLO--Schreibtisch-mit-Objekten.jpg`". [Online; accessed 15-June-2022].

[19] Kaiming He, Jian Sun, and Xiaoou Tang. Single image haze removal using dark channel prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12):2341–2353, 2011. doi: 10.1109/TPAMI.2010.168.

[20] John Y Chiang and Ying-Ching Chen. Underwater image enhancement by wavelength compensation and dehazing. *IEEE transactions on image processing*, 21(4):1756–1769, 2011.

[21] Codruta O Ancuti, Cosmin Ancuti, Christophe De Vleeschouwer, and Philippe Bekaert. Color balance and fusion for underwater image enhancement. *IEEE Transactions on image processing*, 27(1):379–393, 2017.

[22] Nan Wang, Yabin Zhou, Fenglei Han, Haitao Zhu, and Jingzheng Yao. Uwgan: Underwater gan for real-world underwater color restoration and dehazing. *arXiv preprint arXiv:1912.10269*, 2019.

[23] Jie Li, Katherine A Skinner, Ryan M Eustice, and Matthew Johnson-Roberson. Watergan: Unsupervised generative network to enable real-time color correction of monocular underwater images. *IEEE Robotics and Automation letters*, 3(1):387–394, 2017.

[24] Chongyi Li, Chunle Guo, Wenqi Ren, Runmin Cong, Junhui Hou, Sam Kwong, and Dacheng Tao. An underwater image enhancement benchmark dataset and beyond. *IEEE Transactions on Image Processing*, 29:4376–4389, 2019.

[25] Syed Sahil Abbas Zaidi, Mohammad Samar Ansari, Asra Aslam, Nadia Kanwal, Mamoona Asghar, and Brian Lee. A survey of modern deep learning based object detection models. *Digital Signal Processing*, 126:103514, 2022.

[26] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[27] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[28] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[29] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.

[30] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[31] Shaojian Song, Jingxu Zhu, Xiuhua Li, and Qingbao Huang. Integrate msrcr and mask r-cnn to recognize underwater creatures on small sample datasets. *IEEE Access*, 8:172848–172858, 2020.

[32] Example Photo - Otterventures.com (BLOG). URL `https://www.otterventures.com/photo-albums-1`. [Online; accessed 12-June-2022].

[33] Océane Boulais, Ben Woodward, Brian Schlining, Lonny Lundsten, Kevin Barnard, Katy Croff Bell, and Kakani Katija. Fathomnet: An underwater image training database for ocean exploration and discovery. *arXiv preprint arXiv:2007.00114*, 2020.

[34] Estacion Costera De Investigaciones Marinas. URL `http://ecim.bio.puc.cl/en/`. [Online; accessed 20-May-2022].

[35] Edwin H Land. The retinex theory of color vision. *Scientific american*, 237 (6):108–129, 1977.

[36] Anya Hurlbert. Formal connections between lightness algorithms. *JOSA A*, 3(10):1684–1693, 1986.

[37] Zia-ur Rahman, Daniel J Jobson, and Glenn A Woodell. Multi-scale retinex for color image enhancement. In *Proceedings of 3rd IEEE international conference on image processing*, volume 3, pages 1003–1006. IEEE, 1996.

[38] Radhakrishna Achanta, Sheila Hemami, Francisco Estrada, and Sabine Susstrunk. Frequency-tuned salient region detection. In *2009 IEEE conference on computer vision and pattern recognition*, pages 1597–1604. IEEE, 2009.

[39] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[40] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[41] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[42] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9): 1627–1645, 2010.

[43] Number of Parameters and Tensor Sizes in a Convolutional Neural Network (CNN). URL `https://learnopencv.com/number-of-parameters-and-tensor-sizes-in-convolutional-neural-network/`. [Online; accessed 12-June-2022].

[44] Schematics AlexNet. Alexnet — https://www.freesion.com/article/9144975072/. URL `https://www.freesion.com/article/9144975072/`. [Online; accessed 05-May-2022].

[45] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[46] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004.

[47] VGG16 — Wikipedia, the free encyclopedia, . URL `https://commons.wikimedia.org/wiki/File:VGG16.png`. [Online; accessed 04-May-2022].

[48] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[49] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[50] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[51] Stack Exchange - How to interprete ResNet50 types. URL `https://datascience.stackexchange.com/questions/33022/how-to-interpert-resnet50-layer-types`. [Online; accessed 10-June-2022].

[52] Support-vector machine — Wikipedia, The Free Encyclopedia, . URL `https://en.wikipedia.org/wiki/Support-vector_machine`. [Online; accessed 06-May-2022].

[53] George Cutter, Kevin Stierhoff, and Jiaming Zeng. Automated detection of rockfish in unconstrained underwater videos using haar cascades and a new image dataset: labeled fishes in the wild. In *2015 IEEE Winter Applications and Computer Vision Workshops*, pages 57–62. IEEE, 2015.

[54] Ocean Biodiversity Information System. Intergovernmental Oceanographic Commission of UNESCO (OBIS). URL `www.obis.org`. [Online; accessed 10-June-2022].

[55] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[56] Rafael Padilla, Sergio L Netto, and Eduardo AB Da Silva. A survey on performance metrics for object-detection algorithms. In *2020 international conference on systems, signals and image processing (IWSSIP)*, pages 237–242. IEEE, 2020.

[57] Sergei Belousov. PyPi: mean-average-precision. URL `https://pypi.org/project/mean-average-precision/`. [Online; accessed 20-May-2022].

[58] Barret Zoph, Ekin D Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V Le. Learning data augmentation strategies for ob-

ject detection. In *European conference on computer vision*, pages 566–583. Springer, 2020.

[59] Karen Panetta, Chen Gao, and Sos Agaian. Human-visual-system-inspired underwater image quality measures. *IEEE Journal of Oceanic Engineering*, 41(3):541–551, 2015.

[60] Miao Yang and Arcot Sowmya. An underwater color image quality evaluation metric. *IEEE Transactions on Image Processing*, 24(12):6062–6071, 2015.

[61] Monterey Bay Aquarium Research Center (MBARI). URL `https://www.mbari.org/`. [Online; accessed 10-June-2022].