Classification and Localization of Sea Animals in Brackish Waters with Diverse Visibility Using Deep Learning

Master Thesis

Lukas Stranovsky

Aalborg University
Department of Electronic Systems

May 10, 2020

This report is generated from source created with the document markup language LaTeX, originally developed by Leslie Lamport and based on Donald Knuth's TeX. The document was written with the use of overleaf.com and AAU report template created by Jesper Kjær Nielsen. Custom graphs were drawn with creately.com. Background of the cover page comes from unsplash.com and it is licensed under Unsplash license.

Department of Electronic Systems
Fredrik Bajers Vej 7B
9220 Aalborg Ø
Denmark
https://es.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Classification and Localization of Sea Animals in Brackish Waters with Diverse Visibility Using Deep Learning

Theme:
Computer vision

Project Period:
Spring Semester 2020

Participant(s):
Lukas Stranovsky

Supervisor(s):
Thomas B. Moeslund
Malte Pedersen

Copies: 1

Page Numbers: 92

Date of Completion:
May 10, 2020

Abstract:

With the increased occurrence of pollution in water bodies, there is a higher demand by the European Union for member countries to monitor their marine environments [1]. One of the reasons is that water contamination can be estimated by the behavioral change in animals living in the water body [2]. This creates a need for having a computer vision system that could monitor animals underwater. This system would highly depend on the underwater visibility that is influenced by several factors.
This master thesis builds on previous work by Pedersen et al. [3] and experiments with their unique underwater Brackish dataset which is highly influenced by turbidity. The aim of this project is to explore the effects of turbidity on the underwater object detector.
Firstly, the project uses real-time object detector YOLOv3 and manages to improve baseline results from the original paper by 9.2% mAP. The result is verified by a newly introduced manually annotated dataset called the Brackish X dataset which can be used for evaluating the generality abilities of a trained model.
Secondly, this project evaluates which turbidity features are best for estimating turbidity.
Lastly, the best model is evaluated on the original test set divided into 3 subsets based on their estimated turbidity. The result of this experiment set a new course for a future experiment in a controlled environment.

Contents

Preface

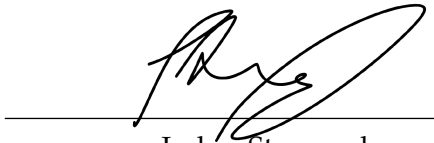
This report documents a Master's thesis at the Master's Programme in Vision, Graphics, and Interactive Systems (VGIS) at Aalborg University (AAU). The aim of this thesis is to investigate the effects of turbidity on the underwater animal detection system based on deep learning.

The thesis starts with chapter Introduction which explains the motivation behind the project and introduces the initial problem formulation. Chapter problem Analysis further narrows down the problem which results in defining the Final problem formulation in chapter 3. Subsequent chapter 4 - Theory, explains key topics in computer vision and deep learning required for proposing a suitable solution in chapter 5 Methodology. Chapter 6 presents the results acquired from the experiment. Chapter 7 - Discussion - explains the presented results. Finally, the project is wrapped up by future work in chapter 8, and the conclusion is made in the last chapter 9.

The citations utilize the IEEE reference style. Meaning that the author's name is not always specified, but instead, the source is referred to with a number in square brackets, e.g. [1]. This number corresponds to the full citation located at the end of the report. The citations are organized in the order they have been referenced within the report. Figures that do not have the reference in the caption were created by the author. Deep learning models in this project were trained on Google Colab.

I would like to express my gratitude to supervisors Thomas B. Moeslund and Malte Pedersen, for guidance and valuable advice that helped me during the process of writing this thesis.

Aalborg University, May 10, 2020



Lukas Stranovsky
<lstran17@student.aau.dk>

Chapter 1

Introduction

One of the essential substances for humankind and all other forms of life is water. However, because of the anthropogenic activities, many water sources are polluted which have led to the deterioration of water quality. According to Heath, [2] water pollution is the presence of hazardous material in water in excessive amounts to the extent that it is no longer safe for humans, animals, plants, or aquatic life. It usually occurs when water bodies; such as seas, oceans, aquifers, rivers, and groundwater are contaminated, mostly as a result of human activity resulting in an alteration of the physical, biological or chemical properties of the water to cause detrimental effects to live organisms.

Water pollution comes from different sources. The three major pollutant sources have agricultural, industrial, and urbanization origin [4]. The pollution of water affects the environment and every life form that depends on the water body [5]. It causes about 14 000 human casualties every day as a result of diseases caused by the consumption of contaminated water, especially in developing countries. It also causes an increased death of aquatic life; plastics present a choking hazard to aquatic life, blocking the breathing passages and stomachs of marine species. Pollution of water from toxic chemical wastes like lead and caladium disrupts the food chain when the pollutants are consumed by small organisms, which are then eaten by fish and eventually find their way to larger animals including people [6].

For these reasons, monitoring underwater life is essential to assess the risk in aquatic ecosystems [7]. Detecting disturbance from pollutants and toxic substances in water bodies can be an important early warning on the quality of water. Physiochemical characteristics are not sufficient in examining aquatic ecosystem disturbances as they only detect known pollutants for a specific period [2]. It is important to also apply behavioral monitoring that investigates the condition of the environment by studying how indicator species respond to the surrounding environment. It is efficient for monitoring aquatic ecosystems and the quality of water in the long term. Due to their sensitivity to alterations in environmental parameters, fish and daphnia are the best indicator species in assessing the quality of water.

The change in behavior of aquatic species, when stimulated by external factors like pollutants, has a regular pattern [7]. According to Heath [2], most of the behavioral indicators used to monitor and assess the quality of water include escape behavior; like swimming rapidly, floating, and circuitous frequency, breathing behavior; such as the respiration depth and rate of breathing, and motor behaviors; like the swerving frequency, swinging frequency, height,

velocity, and dispersion. Behavioral data is analyzed [7] to evaluate the status and degree of pollution in aquatic systems by check on for abnormal behavior. Monitoring the behavior of underwater life offers the opportunity to detect disturbances in aquatic ecosystems and mitigate them early enough to avoid the risks and losses that they would have later caused. Moreover, European Union introduced a strategy for all member states with the title "Marine Strategy Framework Directive (MSFD) which requires member states to observe their marine environments, in favor of improving decision making, law development, and economic expansion [1].

To be able to observe marine animals, biologists have to go through an enormous amount of data which can be manually demanding. The efficiency of this process can be improved by taking advantage of a reliable computer vision system that could assist in finding animals of interest in the data automatically [8]. For this purpose, scientists created various marine datasets. One of them is the Brackish dataset proposed by Pedersen et al. [3] in 2019. The Brackish dataset is one-of-a-kind because it was collected in a brackish channel with varying visibility that is caused by turbidity. This master thesis is going to build on previous research presented by Pedersen et al. by further investigating the area of object detection in underwater environments and identifying opportunities for enhancements.

1.1 Initial Problem Formulation

Based on the current knowledge about the topic, the initial problem formulation was formulated as follows:

How can a computer vision system be developed to detect animals in brackish water?

The consecutive chapter is going to elaborate on the problem in further detail and describe the key terms required for constructing the final problem formulation.

Chapter 2

Problem Analysis

The main goal of this chapter is to narrow the initial problem down which is necessary in order to form the final problem formulation. It starts by introducing the Brackish dataset, then it deals with underwater challenges and explores related work in the field of underwater vision.

2.1 Dataset Exploration

The dataset we are going to use in this project is called "The Brackish Dataset" which was proposed by Pedersen et al. [3] in 2019. It is the first dataset made in an estuarine environment, furthermore, it is the first publicly available dataset captured in Europe. The main motivation behind it is to monitor the marine environment as it is obligatory to reach the plans of the European Union's marine framework [1].

Figure 2.1: "Limfjordbroen" - a bridge over Limfjord. The place where the Brackish dataset was collected.

The dataset was captured in Limfjord which is approximately 170km long shallow part of the sea that connects the North Sea with Kattegat and separates Nordjutlandic island

(Vendsyssel). Eastern current has very high salinity - 32 Practical Salinity Units (PSU) - this is located in Thyborøn Channel, which is a place where the North sea enters the water network. Moving on further over the channel, salinity can get as low as 18 PSU [9].

More specifically, the dataset was captured under the bridge - Limfjordbroen - that connects Aalborg and Nørresundby (Figure 2.1). The data was collected using a setup shown in Figure 2.2 that contains 3 cameras and 3 lights placed in a grid. The whole setup is sunken 9 meters underwater pointing downwards. Cameras that were used are 1/3" Sony ExView Super HAD Color CCD providing resolution up to 1080x1920 pixels with a frame rate up to 30 fps. The intensity of the lights is about 1900 lumens. Annotation of the dataset was done under the supervision of a biologist. There are 6 categories of animals: Big fish, Small Fish, Crab, Jellyfish, Shrimp, and Starfish. Distribution of the frames for each category can be seen in Figure 2.3 and more precisely in table 2.1.

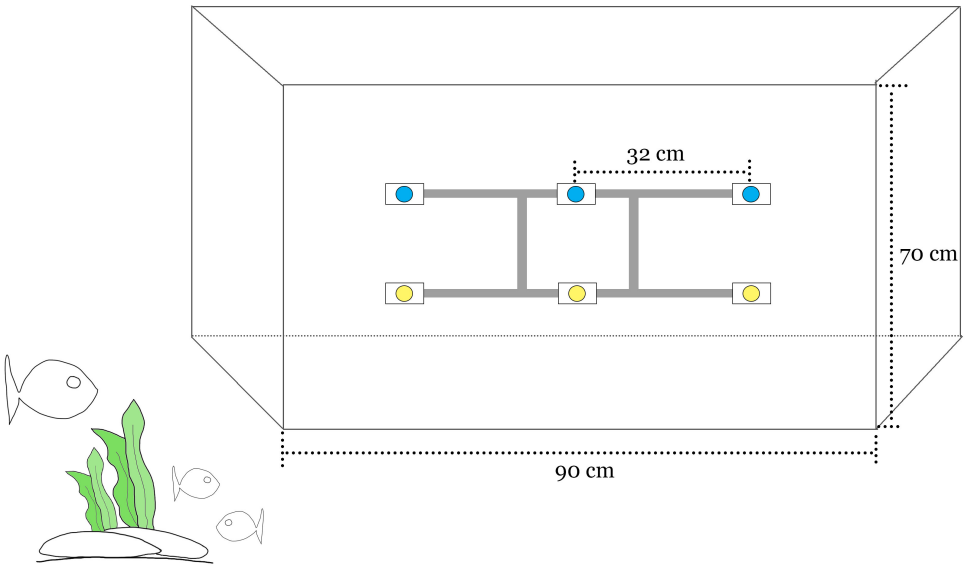
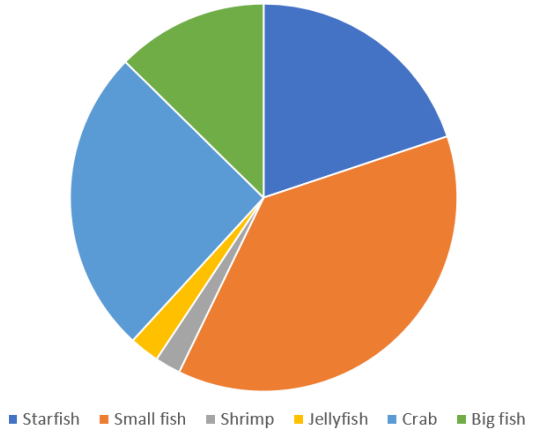


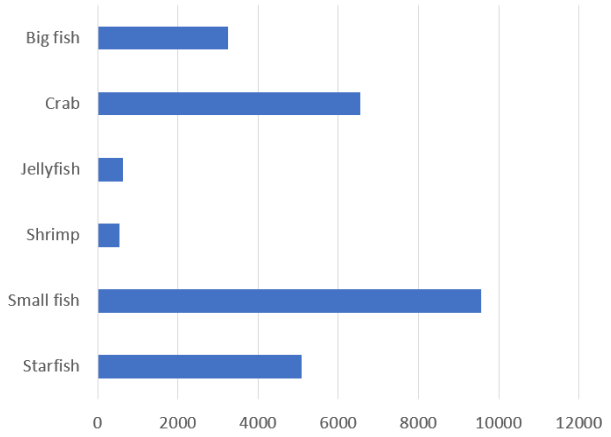
Figure 2.2: Setup of the underwater frame made from stainless steel. Blue color represents position where cameras are mounted, whereas yellow color represents lights.

Class	Annotations	Video Occurences
Big fish	3 241	30
Crab	6 538	29
Jellyfish	637	12
Shrimp	548	8
Small fish	9 556	26
Starfish	5 093	30

Table 2.1: Individual classes of the Brackish dataset together with the number of annotations and occurrences in videos. [3]



(a) Distribution of the data per each class in the Brackish dataset shown in pie chart.



(b) Distribution of the data per each class in the Brackish dataset shown in chart.

Figure 2.3

2.1.1 Baseline Results

Benchmark results presented by Pedersen et al. [3] their deep learning models are pre-trained on the OpenImages dataset. As stated by Pedersen et al., the reason behind this is that the OpenImages dataset contains relevant classes. Those classes are fish, starfish, jellyfish, shrimp, and crab. However, the comparison with classes of other benchmark datasets where pre-trained weights are also available for download is not made. Replacing the feature extractor could be a simple way of improving the baseline results. The available relevant feature extractors are pre-trained on Pascal VOC [10], ImageNet [11] and COCO [12].

Tables 2.2 and 2.3 show baseline results. We can see that YOLOv3 outperformed the older version of the algorithm YOLOv2 by 52.62% in AP_{50}. The most challenging class to detect is Small fish despite having the largest amount of annotations. The easiest class to detect is Starfish.

	AP	AP_{50}
YOLOv2	09.84	31.10
YOLOv3	**38.93**	**83.72**

Table 2.2: This table shows that YOLOv3 outperformed YOLOv2 in the original paper [3].

Class	YOLO Class	AP_{50}
Big fish	0	89.99
Crab	3	92.71
Jellyfish	4	82.05
Shrimp	2	76.62
Small fish	1	62.29
Starfish	5	98.67

Table 2.3: More detailed evaluation per individial classes from the original paper [3].

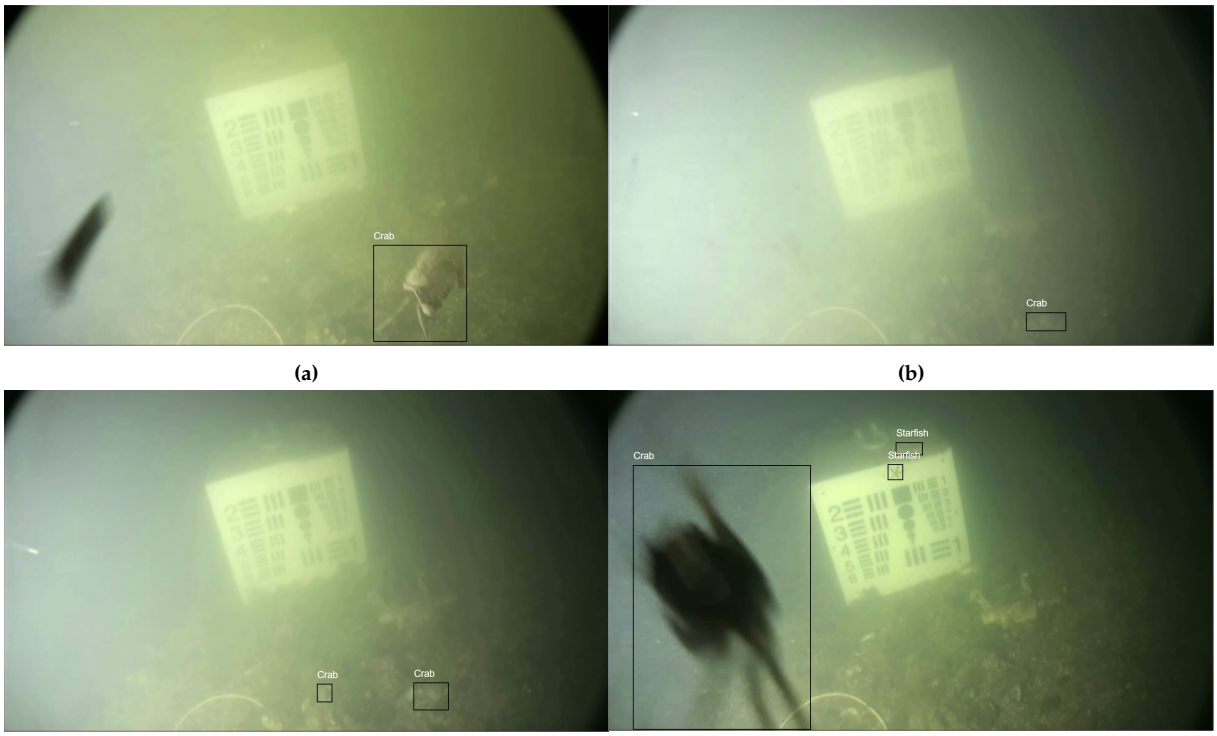
2.1.2 Crab

Cancer pagurus also known as **Edible Crab** or **Brown Crab** of class Malacostraca is easily distinguishable from other species because of its body shape. It is characterized by the typical brown-orange color of the carapace. The edible crab can grow up to 25cm, however, the average adult individuals have about 15cm. This crab lives in various depths, from coasts to depth of 100m populating North Sea, English Channel, and coasts of Portugal to the Mediterranean sea. Typical food of the Brown Crab consists of mollusks, crustaceans and also decaying flesh of dead animals [13].

Typical Characteristics [13]:

- Black or dark brown colored claws with teeth.
- Clump of strands of hair on legs.
- The last part of the legs ends with a spine tip.
- The carapace has a specific shape with front-lateral margins with 10 rounded lobes.

Figure 2.4: Picture of an adult Brown Crab in its natural habitat [13]



(a) (b)

(c) (d)

Figure 2.5: Interesting frames with crabs in them from the Brackish dataset.

Figure 2.5 shows how the crab looks like in the Brackish dataset. We can see that it looks much different from the reference Figure 2.4. We cannot see the distinct brown color, either the eyes or legs. The crabs in the dataset have various sizes from just very small sizes like in 2.5 (c) but they sometimes swim in the stream and can appear huge and block the camera vision as in 2.5 (d). The crabs look very similar to the background environment, especially the rocks. This can create a lot of false positives. Moreover, when the frame is turbid as in 2.5 (b), the clearly visible shape of crab is not that distinct as when turbidity is low. When the detector would be trained on low turbid frames, it could happen that it would not be able to detect crabs in high turbid frames.

2.1.3 Starfish

Asterias rubens of class *Asteroidea* are often referred to as Starfish or Sea stars (Figure 2.6). Although they have name fish in the title, they are not really fish. The correct naming of these underwater animals is echinoderms. Asterias Rubens occurs in the north-east Atlantic region, especially in the North Sea with high frequency. The typical size of this echinoderm for adults can be between 10 to 30 cm in diameter, rarely up to 52 cm. Size is highly dependant on food availability, thus it is not a reliable age indicator. Asterias Rubens is a predator that feeds upon a large scale of living organisms such as carrion, mollusks, sea worms, and also other echinoderms. It lives at the bottom of the sea where it slowly crawls or floats with the current [13].

Typical Characteristics [13]:

- Usually 5 tapering arms, infrequently 4-8.
- Adult individuals have an orange color, faded brown, or violet. Infants are more brownish. Color is dependant on the environment and water depth.
- Soft and flexible papulae in spongy areas.
- Ventro-lateral ossicles in slanting rows with white color.

Figure 2.6: 2 echinoderms hanging out in gravel benthic zone. Adult individual in the background, and infant damaged individual with 3 arms in the foreground [13]

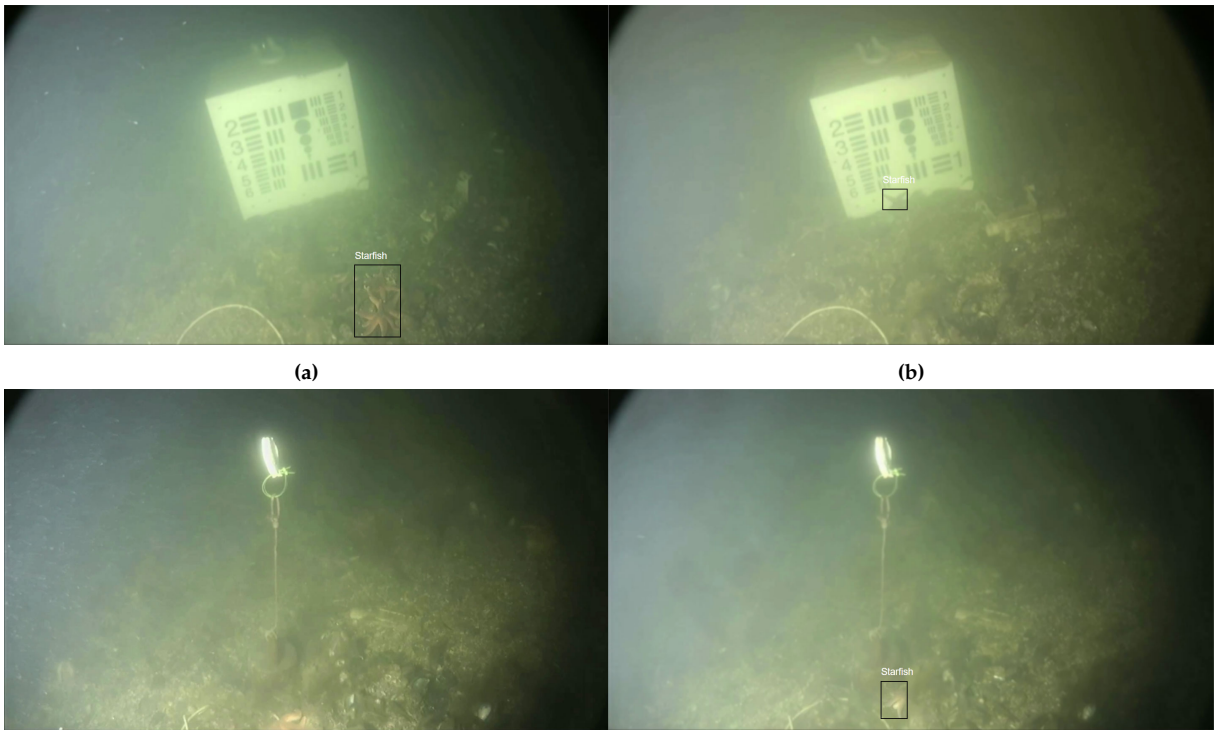
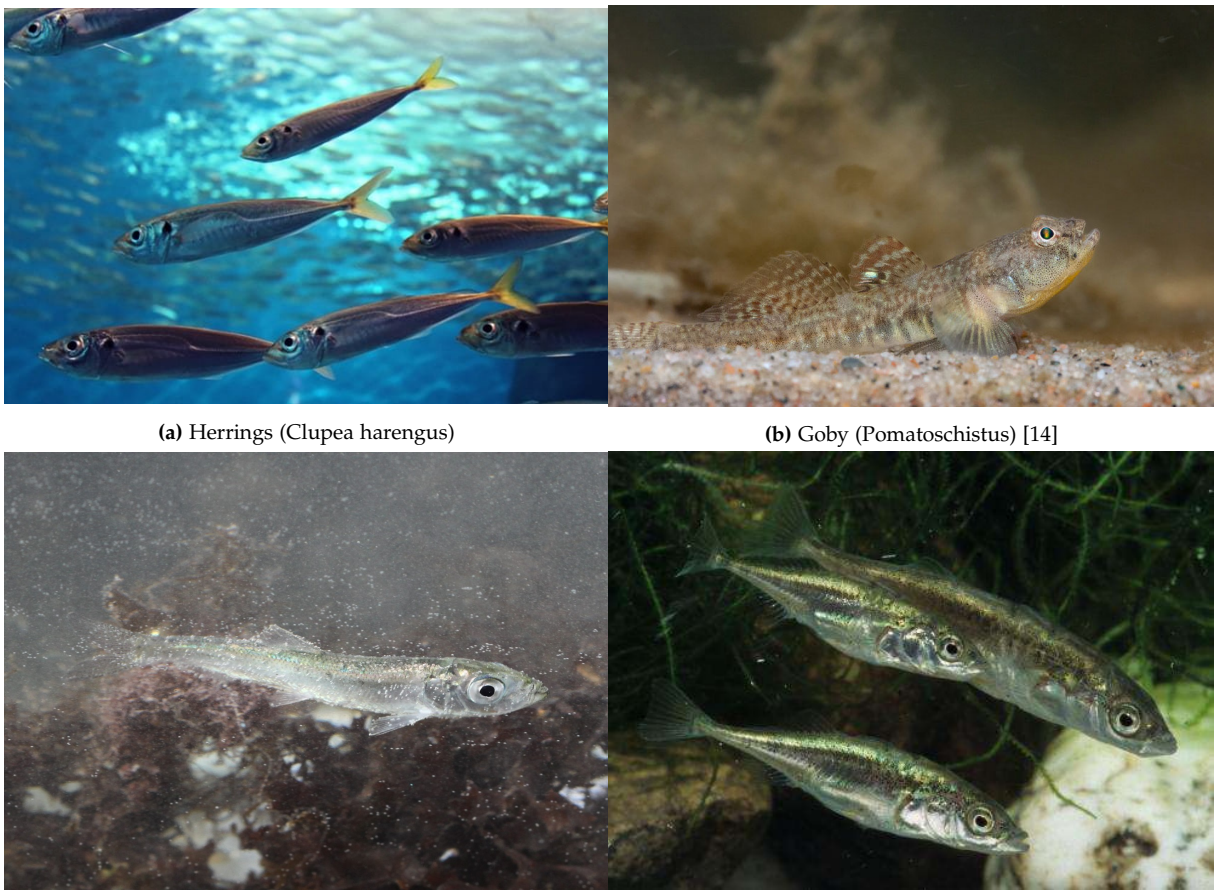


Figure 2.7: Interesting frames with starfish in them from the Brackish dataset.

Interesting frames from the Brackish dataset can bee is seen in Figure 2.7. It can be observed that the number of their arms is not always 5. Moreover, some of the starfish have color if they are closer to the camera, but some do not mainly if there are attached to the block and exposed to direct artificial light as in 2.7 (b). Some of the frames are annotated as one starfish even though it is visible that there are more of them (2.7 (a)). The size of them also varies, there are adults and infant individuals. For example in 2.7 (c), the starfish is not even annotated.

2.1.4 Small fish

Small fish consists of 4 types of fish (Figure 2.8). Namely Sticklebacks (Gasterosteus aculeatus), Gobies (Pomatoschistus microps), European Sprats (Sprattus sprattus), and Atlantic Herrings (Clupea harengus). Sticklebacks, Gobies, and Sprats are small fish about 5 to 10 cm long which makes them extremely difficult to recognize. Sticklebacks, Sprats, and Herrings are colored in silver, and in their natural habitat, they swim in schools. However, Gobies are mottled with sandy color. Gobies are able to tolerate a wide range of water salinity, although they prefer low salinity waters. It is an abundant fish that migrates to shallow waters during the breeding season. Herrings can grow up to 45cm, however, in the Brackish dataset they appear as infants. Thus, they are classified as small fish [13].



(a) Herrings (Clupea harengus)

(b) Goby (Pomatoschistus) [14]

(c) European Sprat (Sprattus sprattus)

(d) Sticklebacks

Figure 2.8: Species belonging to the class Small fish in the Brackish dataset.

According to the authors of the Brackish dataset [3], the first intention was to classify fishes into classes according to their species. However, this was not possible with certainty because of turbidity, light attenuation, and color weaken. Thus, they were all put into the same class

Sources of the images: 2.8 (a) - `https://animals.net/herring/`; 2.8 (c) - `http://www.freenatureimages.eu/`; 2.8 (d) - `https://phys.org/news/2013-06-brother-art-thou-sticklebacks-relatives.html`

"Small fish". This can create issues in the future because the difference with the "Big fish" class is only the size of the fishes. Therefore, we cannot be completely sure whether the fish belongs to those 4 species, but it can also be species that are included in the "Big fish" class.

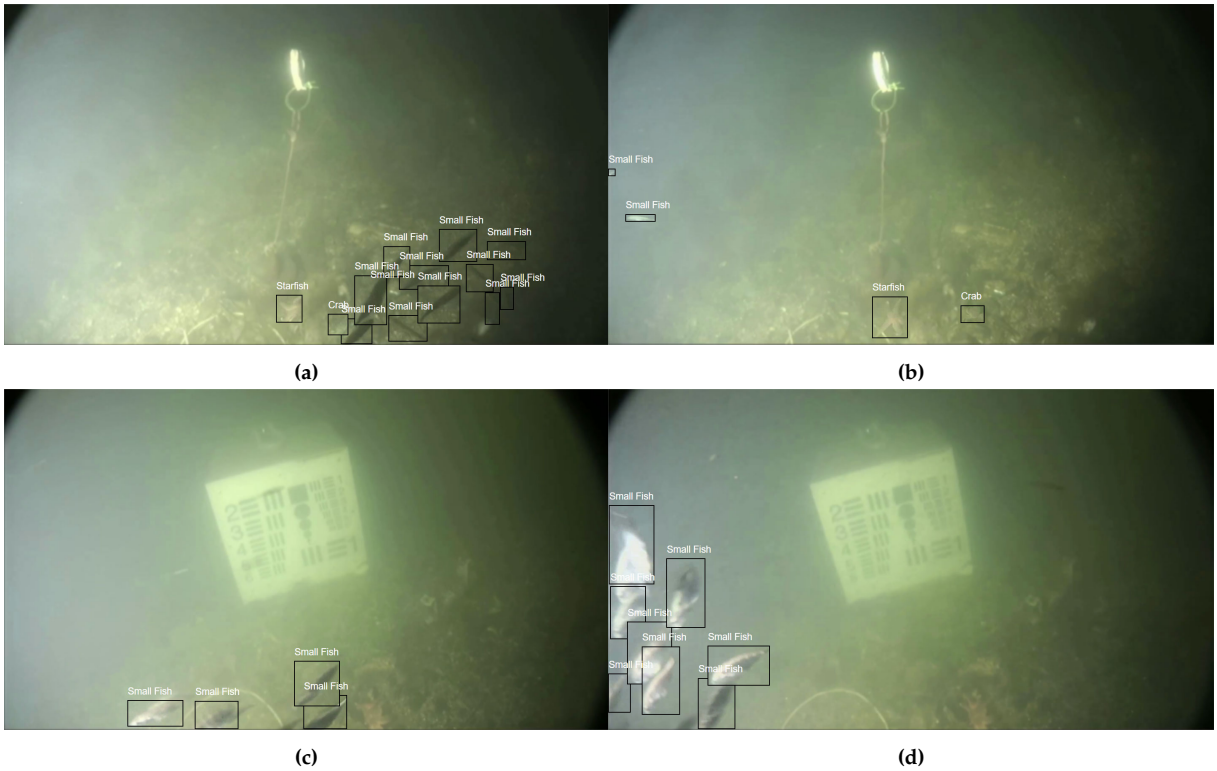


Figure 2.9: Interesting frames with "Small fish" class in them from the Brackish dataset.

Regarding the representation of the class in the dataset which can be seen in Figure 2.9. The annotation of the small fish can be very small as in 2.9 (b), but when they swim in front of the camera they can appear big as in 2.9 (d). When they swim in the schools there is a lot of occlusion going on which would make them not only difficult to classify by computer vision system but also by an annotator. In Figure 2.9 (d) 2 fishes are annotated as one. Furthermore, when they swim they are rotating so sometimes there is their back visible, sometimes it is stomach and sometimes just head, tail, and so on.

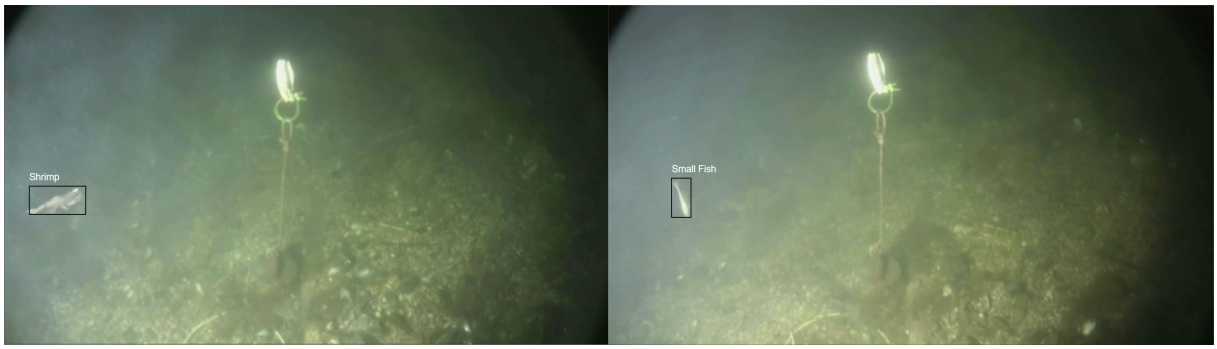
2.1.5 Shrimp

Brown shrimp (*Crangon crangon*) (Figure 2.10) belongs to the phylum of Arthropoda - animals with segment body and limbs. It has mottled brown color but this can vary depending on the environment. The body shape is cylindrical narrowing to the posterior end that ends with a fanned tail. Adult individuals are usually about 8.5 cm in length. It has 2 short antennae and 2 long antennae coming from its head that can be almost as long as the whole body. It occurs on sandy and muddy ground and sometimes can be hard to spot because it likes to dig under the surface with only eyes and antennae sticking out. It's often hunted by predators for instance birds and fish. [13].

Typical Characteristics [13]:

- The last segment of the body has two pairs of lateral spines.
- Cylindrical body with fin tail.
- 5 pairs of walking legs and 5 pairs of swimming legs (swimmerets).

Figure 2.10: Adult shrimp crawling on the bottom of the aquarium [13]



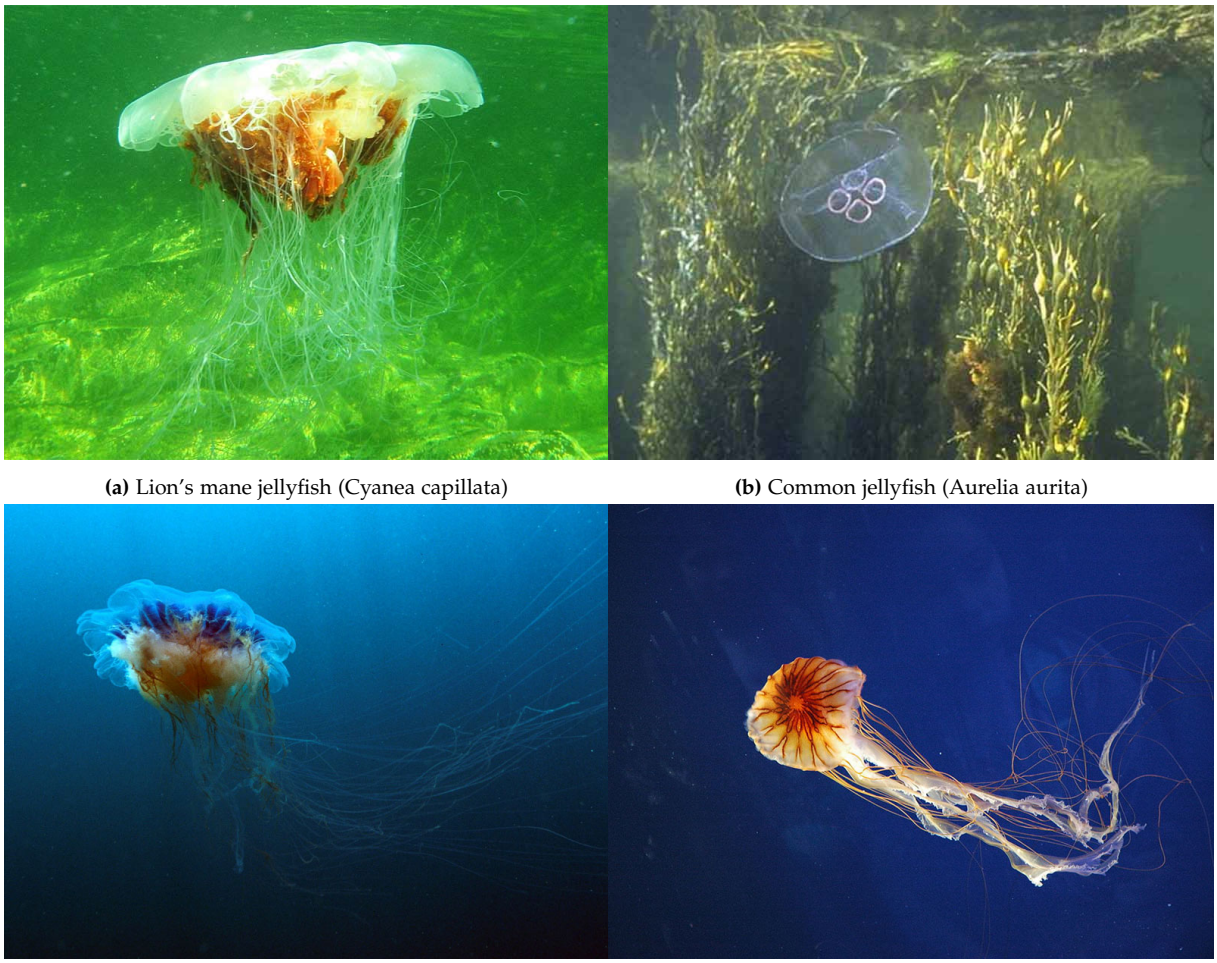
(a) **(b)**

Figure 2.11: Interesting frames with shrimp class in them from the Brackish dataset. Adult individual in (a) and infant in (b).

First of all, the number of frames of shrimps is low in comparison with other classes. The uneven number of classes can create an unwanted bias for our computer vision detector. Then the problem is in the appearance of infant individuals swimming around because they look similar to small fish (Figure 2.11 (b)), moreover the shrimp in figure 2.11 (b) is wrongly annotated as small fish. When shrimp crawls on the bottom of the body water, it might be falsely classified as crab.

2.1.6 Jellyfish

5 species of Jellyfish can be found in Danish marine environments. Those are Blue jellyfish (Cyanea lamarckii), Lion's mane jellyfish (Cyanea capillata), Moon jellyfish (Aurelia aurita), Compass jellyfish (Chrysaora hysoscella), Barrel Jellyfish (Rhizostoma pulmo)[1]. Because of the occlusion, turbidity in the water that leads to light attenuation, it is problematic to reliably distinguish between them in the dataset. Probably, for this reason, creators of the brackish dataset did not mention which one of them is occurring in the frames. To our best knowledge, we think that the jellyfish in the Brackish dataset is the Common jellyfish, which is the most wide-spread. Thus we have a closer look into this species.



(a) Lion's mane jellyfish (Cyanea capillata)

(b) Common jellyfish (Aurelia aurita)

(c) Blue jellyfish (Cyanea lamarckii)

(d) Moon jellyfish (Chrysaora hysoscella)

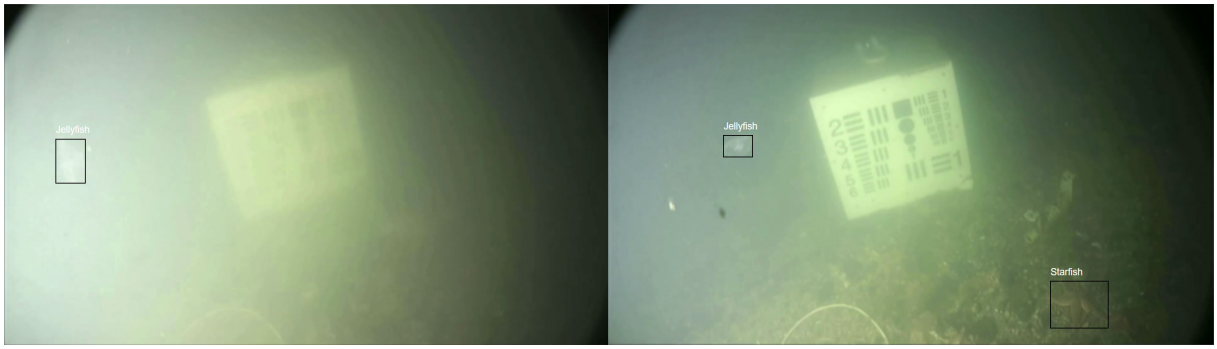
Figure 2.12: Figure shows 4 types of jellyfish species that occur in the North Sea [13]

[1]https://wildaboutdenmark.com/jellyfish-along-the-danish-coast/

Moon jellyfish have a completely transparent smooth, flatten, thick umbrella. The gonads can have various colors, e.g. mauve, violet, reddish, pink, or yellowish. The diameter of an adult individual can be between 25 cm to 40 cm [13]. They can be often seen washed up on the shores. It is harmless to humans.

Typical Characteristics [13]:

- It has a thin transparent umbrella.
- Gonads can have various colors.
- Short, hollow tentacles.
- 4 purple-blue gonads formed into the "horseshoe" shape contained into the gastrovascular cavity.
- Gonads do not exceed the umbrella surface like in most of the jellyfish species.



(a) **(b)**

Figure 2.13: Interesting frames with jellyfish class in them from the Brackish dataset.

Similarly to the shrimp, the jellyfish class is also lack of frames with jellyfish in them. We have observed the dataset and the only species of jellyfish occurring there is the Common jellyfish (Figure 2.12 (b)). The problem with this jellyfish is that it is transparent and in the turbid water it is hard to spot as can be seen in Figure 2.13 (a) where it looks like some floating abiotic object. Whereas when the turbidity is low as in 2.13 (b), the umbrella and the gonad can be distinctly visible.

2.1.7 Big fish

There are 2 kinds of big fishes occurring in the dataset (Figure 2.15). Those are Lumpsuckers (Cyclopterus lumpus) and Sculpins (Myoxocephalus scorpius).

Lumpsuckers live in depth from 50m to about 300m, however, during spawning, they migrate to shallow waters. The spawning season is between February and May. Male changes its color during the spawning season to reddish. Adults grow between 30 to 50 cm [13]. The ventral fins from the bottom side of Lumsuckers are adjusted into a suction disk that allows them to attach on solid surfaces.

- Various colors from reddish, bluish to slate gray.
- Distinct dorsal ridge.
- Can be seen attached to a rock in a fixed position.

Sculpins also are known as Bull-rout are fish residing at the bottom of the benthic layer. It is a very flexible fish, it lives in salt waters the same as in fresh rivers. The common length is 30cm in southern parts, however, in northern parts, it can grow up to 90cm. Male's stomach is cherry red and light orange in females [13].

- It has a large head in comparison with the rest of the body.
- Two dorsal fins narrowly attached on the back.
- Black stripes.
- Mottled skin fins and skin color.

(a) Male Lumpsucker (Cyclopterus lumpus) **(b)** Sculpin (Myoxocephalus scorpius)

Figure 2.14: Fish that are occuring in the Brackish dataset under the label "Big fish" [13].

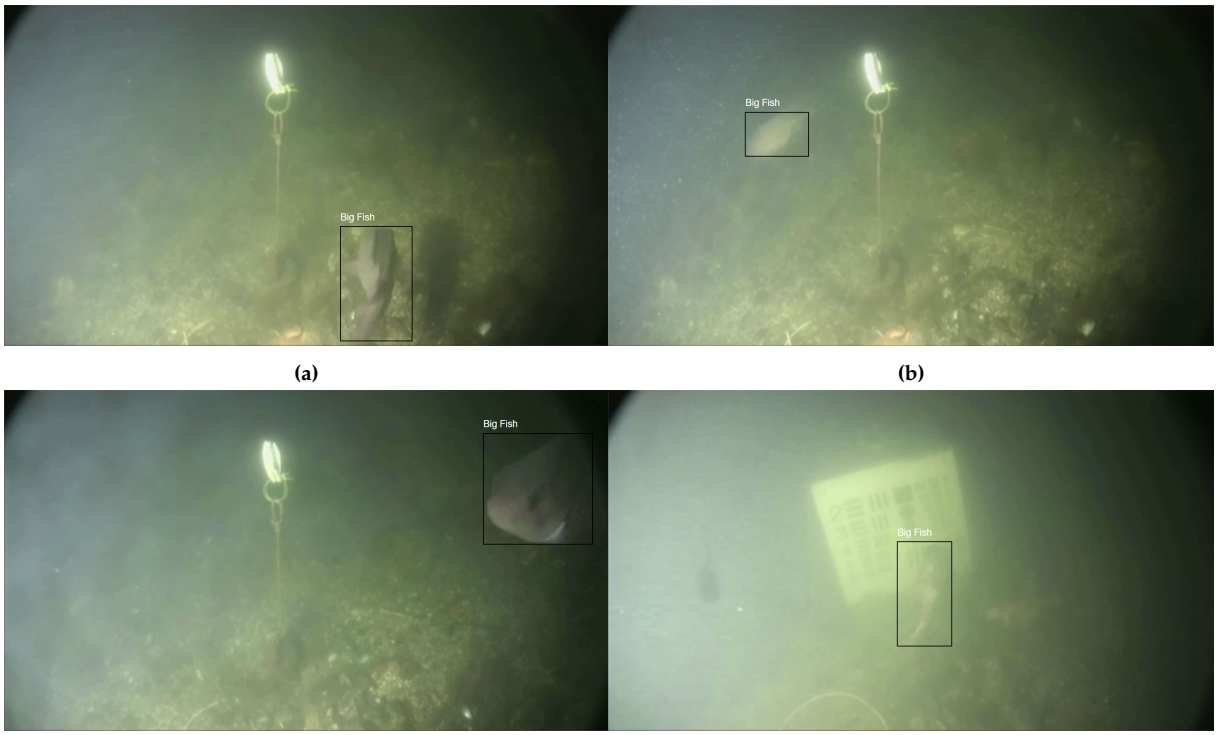


Figure 2.15: Interesting frames with big fish class in them from the Brackish dataset.

Big fishes from the Brackish dataset can be seen in Figure 2.15. It can be seen that fish can be partly occluded and it can have different shapes when it swims under different angles relative to the camera. In the 2.15 (a) there is a big fish swimming in the opposite direction to the camera. It casts a shadow that could confuse a computer vision detector to falsely classify the shadow as a fish. In the next frame 2.15 (b) when fish is too far away from the camera, it appears just as a blot. The frames 2.15 (c) show big fish occluded with only head in the field of the camera vision. The last frame 2.15 (d) shows again the back of the fish in mid turbid water, however, it moves which makes the shape blurry and it appears to be like some underwater floating stick.

2.1.8 Review

By reviewing the dataset, we have learned how the animals in the Brackish dataset look like and what difficulties there are for an object detector to deal with. The major challenges are turbidity, occlusions, animals overlapping, changing illumination conditions, blur, incomplete annotations, animals have different scales and varying angles. Floating objects such as leaves, seaweed can cause false positives. The bottom of the water body can also cause many false positives because animals are naturally camouflaged which means that they share certain features with the environment they live in. This helps them to stay away from predators. Some animals have a similar shape, for example when shrimp are swimming it has similar features as a small fish. Moreover, the dataset classes are uneven. The class "Small fish" has 9 556 annotations, whereas class "Shrimp" has only 548. This can create an unwanted bias. Lack of data is also an issue. A rule of thumb when training deep learning object detectors is having at least 2 000 annotations per class for training.
There are also phenomena typical for underwater vision such as color diminishing and halo effect. Color diminishing [15] illustrated in Figure 2.16 causes low contrast of colors underwater. This is because light with higher wavelength intervals does not travel far in water substance. Therefore, red color occurs only up to 5m underwater, whereas blue color and violet colors travel the furthest. The Halo effect, on the other hand, is caused by turbidity. It occurs when the amount of dissolved particles in the water is too high. Rays of light scattered from an artificial light source reflect those particles in all directions. This decreases the overall visibility and creates false edges.

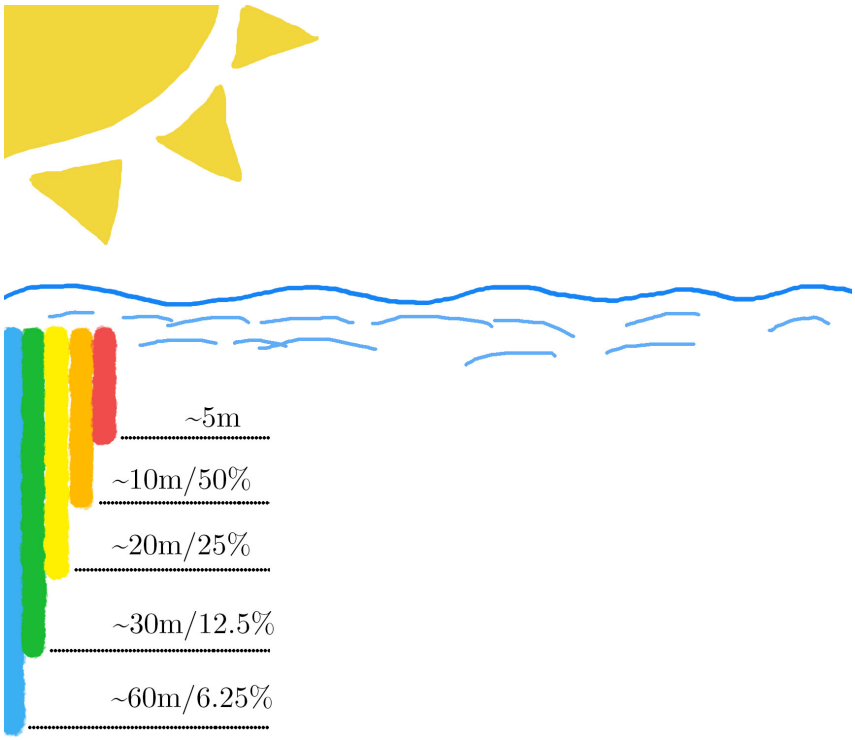


Figure 2.16: Illustration of color diminishing correlated with the depth of light. It shows the percentage of light that penetrates to certain depths. The reach of red color is the shortest whereas, the blue light is the longest.

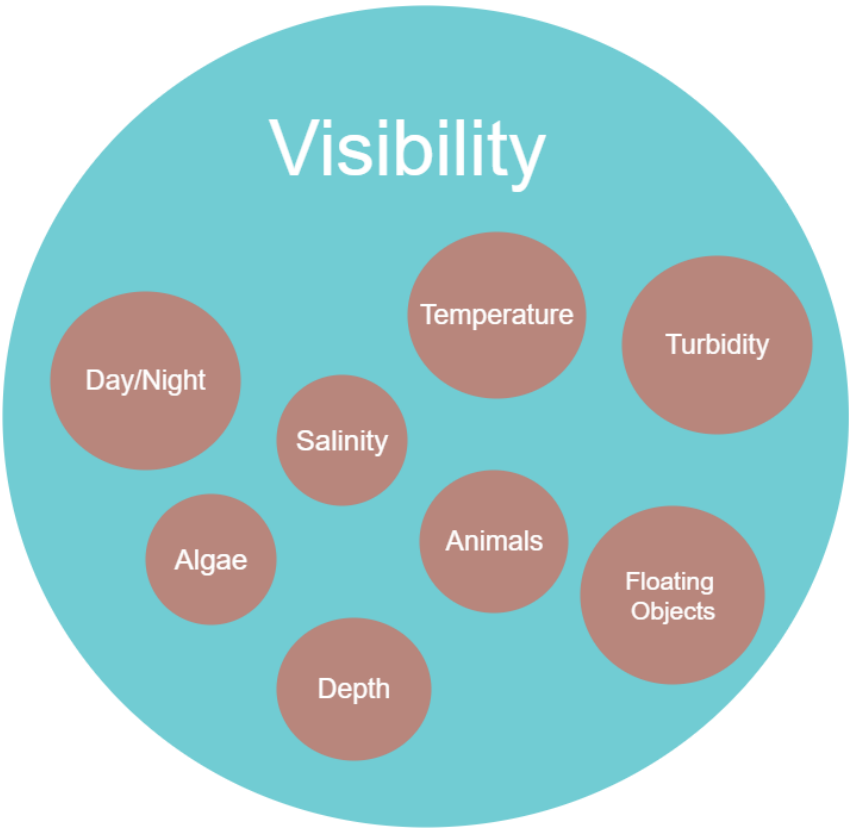


Figure 2.17: Venn diagram with factors influencing visibility underwater.

Turbidity has the highest influence on the vision in the Brackish dataset, we will analyze it in more detail in the following section. More precisely we will look into what turbidity is, what it causes, and how it can be measured. Since the overall visibility is mostly affected by turbidity, the difference between visibility and turbidity might be not completely clear. According to definitions, turbidity measures the number of dissolved particles in the water[2], whereas, visibility is defined as the distance that object can be readily recognized[3]. Therefore, it can be concluded that turbidity is just one of the influencing factors of the overall visibility. The other influencing factors are illustrated in Figure 2.17.

[2]https://www.lenntech.com/turbidity.htm
[3]https://dtmag.com/thelibrary/visibility-illuminating-facts-unclear-situation/

2.2 Turbidity

Turbidity is the optical property of a fluid or transparent solid which describes how clear or hazy the sample is (Figure 2.18). It comes as a result of the suspension or dissolution of particles in a fluid, which scatters light giving the water a cloudy or murky appearance [16]. Water is characterized as turbid when the particulate matter in it is conspicuous enough to reduce the penetration of light. According to Lawler [17], particulate matter may include sediments-like silt and clay, substances that could either be inorganic or organic, algae, colored organic compounds that can dissolve in water, and microscopic organisms.

The causes of turbidity are complex and greatly varied and are influenced by the physical, chemical, and microbiology characteristics of the water [16]. The particles causing turbidity also vary in size. In natural waters, turbidity may be caused by soil and rock weathering. Human activity has also contributed greatly, for instance, sewage and wastewater releases from industries, the use of powerboats and vehicles on water bodies, etc. Inorganic silt and clay and natural organic substances from decaying plant and animal material are among the most common particulate matter in water [17]. Other causes of turbidity include inorganic precipitates like metals, biological organisms like algae, macro, and micro bacteria, and naturally occurring minerals. Water treatment additives and components of the system can also cause turbidity.

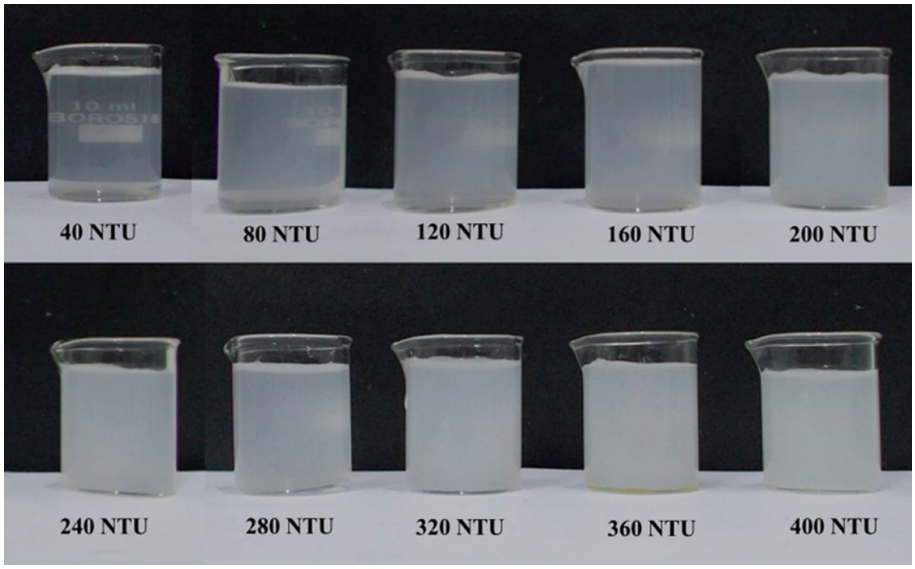


Figure 2.18: Glass flasks with water samples of various turbidity [18]. The sample with 40 NTU is almost transparent, but sample with 200 NTU is almost opaque.

According to guidelines for drinking water, there are two major types of turbidity [19];

- Inorganic particles include silt, clay, natural precipitants, and mineral fragments. They affect the water pH, affect its taste, and give it a cloudy appearance. They also act as a source of micronutrients, metals, and metal oxides in the water.

- Organic particles include decomposing plant and animal debris, organic macromolecules, and microorganisms. They affect the taste, color, pH, and smell of the water.

Measuring the turbidity is a key test of the quality of water [20]. The use of a turbidity tube is the most simple and cheapest way to measure turbidity. This is a tube which has a black X at the bottom. The analyst continuously pours water into the tube until the cross disappears and notes the scale on the tubes outside in NTU (Nephelometric Turbidity Units) [17]. Secchi disks can be also used; they involve lowering a colored disk that is attached to a line into a fluid until the disk disappears. The part of the line dipped into the fluid is noted and then recorded as NTU. Even though these methods are simple, they are not accurate as they rely on a user's consistency and are unable to measure very low turbidity. According to Kitchener et al. [20] more accurate measurements can be obtained using optoelectronic meters that work by emitting a known light intensity through a sample. The particles suspended in the sample then absorb or scatter this light. The last step is the measurement of the scattered light for the turbidity of the sample. This measurement of turbidity is classified into two categories:

- Nephelometry for low turbidity samples (less than 40 NTU) where the light after scattering is measured at a 90° angle.
- High turbidity samples (between 500-1000 NTU) which are visibly colored are measured at an angle of 180°.

Regarding turbidity annotation in the Brackish dataset. As mentioned previously, turbidity causes light attenuation which has an influence on overall visibility and might have an effect on the results of the object detector. By taking advantage of the reference block occurring in the dataset we can observe 3 levels of Turbidity: low turbid, mid turbidity, high turbidity (Figure 2.19). Low turbidity occurs when all the numbers on the reference block are clearly visible. Mid turbidity can be observed when the smallest number 6 on the reference block is not visible due to turbidity. And subsequently, the high turbidity, where any of the numbers cannot be distinctly recognized.

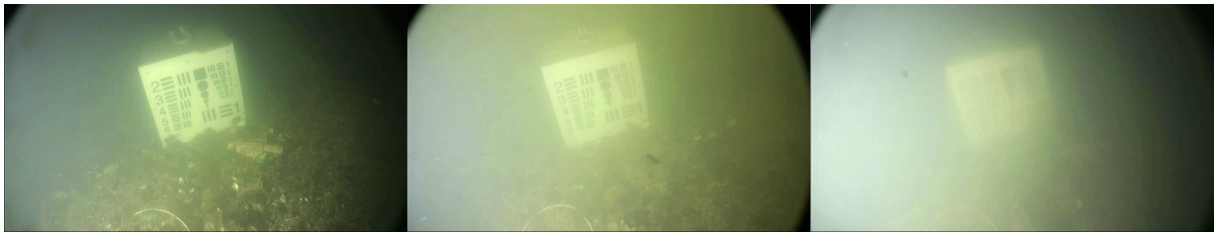


Figure 2.19: 3 levels of Turbidity that can be observed with the unaided eye. From left to right: Low turbidity, semi turbid, high turbid.

By looking into possibilities of detecting turbidity, it was found out that turbidity can be also detected using computer vision. Since our computer vision system should operate in varying visibilities where turbidity has the most effect on the camera vision, it would be beneficial if our system could detect different turbidity levels. Thus, the next section will proceed with analyzing related research focused on detecting turbidity using computer vision.

2.3 Traditional Computer Vision vs Deep Learning

Deep Learning achieved better performance in many computer vision tasks and surpassed traditional computer vision methods such as SIFT, SURF, FAST, HOG, Haar-like features, etc. Does this mean that the traditional approach is now obsolete? Mahony et al. [21] tries to answer this question in their paper.

The latest improvements in computing power, memory sizes, power consumption requirements, cameras improvement were the main factors that helped deep learning to rise. DL has achieved state-of-of-the-art performance in Image classification, semantic segmentation, object detection, and simultaneous localization and mapping (SLAM). Even though utilizing DL has led to jump in terms of accuracy, this has its cost which is in the form of billions of additional math operations. It is necessary to have a high powered GPU or TPU available.

The pipeline of traditional CV techniques starts firstly by feature engineering which consists of manual feature extraction. Then after the features are obtained a classifier with shallow structure is employed to perform prediction. Classifiers with a shallow structure are considered to be for instance Support Vector Machines, K-Nearest Neighbours, shallow neural networks. On the other hand, DL has introduced the concept of end-to-end learning which consists of feeding data into the network where it is up to the network to decide which features are meaningful and how much impact they will have on final prediction. The main difference between DL and traditional technique is that traditional CV models have to be programmed whereas DL models are trained. DL requires less proficiency, thus it is criticized to be a black box. The challenge with the traditional approach is that it is up to the developer to decide which features are important. This can become challenging especially when the difficulty of a problem increases. DL is often criticized to be a black box because it is not always clear why and what is deep neural network learning, whereas a traditional CV is fully transparent. Furthermore, the issue where DL pulls the shorter end is, for instance, 3D CV. 3D vision requires hand-crafted information about smoothness, silhouette, and lighting conditions.

Another approach is a combination of handcrafted features and DL, these are called Hybrid approaches. One of the examples is to utilize CV techniques to find the area of interest and then perform DL just on small patches. This will result in increased performance because the network does not have to process entire frames.

Mahony et al. [21] claim that a lot of CV algorithms developed in the past 20 years have become irrelevant. However, knowing only DL would greatly limit the abilities of a CV engineer. There are areas where DL is not the best option and traditional CV techniques are still necessary to know. However, we think that a deep learning solution is more suitable for our problem. The next section is going to explain evaluation methods that are common in the field of deep learning object detectors and are approved by the scientific community.

2.4 Related Work in Marine Object Detection

There has been a lot of research done recently in the field of object detection. Thanks to improvements in the field of marine optical imaging technology. Namely, improvements in digital cameras, autonomous underwater vehicles (AUV), and unmanned underwater vehicles (UUV). Moreover, increasing requirements by the European Union for member countries to monitor their marine environments has contributed to the popularization of the field of marine object detection.

The main difference between a common object detection and marine object detection is that underwater vision lacks natural illumination that is caused by light attenuation. According to Moniruzzaman et al. [22] underwater marine detection is focused on several different categories based on the target group they are attempting to detect. Target groups that are subjects of Moniruzzaman et al research are Fish, Planktons, Corals, and Seagrass.

Villon et al. [8] compares traditional approach with deep learning based approach and proves that deep learning outperforms traditional CV technique utilizing HOG and SVM classifier. Mahmood et al [23] uses VGGNet to detect corals on Moorea coral dataset[4].

When talking about the detection of plankton, Py et al. utilizes a deep convolutional network inspired by GoogleNet on the National Data Science Bowl dataset [5]. Dai et al [24] propose a network called ZooPlanktoNet which is an adjusted network inspired by AlexNet and VGGNet. They employ data augmentation techniques such as rotation, translation, rescaling, shering, and flipping. Then the networks were trained on ZooScane System Dataset[6].

By studying related research in the field of marine vision, it was found out that deep learning architectures utilized underwater do not differ from those used for overland datasets.

By studying the related work in the field of marine object detection, it was found out that researchers use traditional CV approaches and deep learning neural networks. Deep neural networks were prevalent. This is because deep learning has surpassed traditional CV in many tasks. Therefore, the next section is going to strive to answer the question of whether the traditional techniques are still viable.

[4]http://vision.ucsd.edu/content/moorea-labeled-corals
[5]https://www.kaggle.com/c/datasciencebowl
[6]https://www.seanoe.org/data/00446/55741/

2.5 Related Work in Turbidity Estimation based on Computer Vision

As mentioned in the section about turbidity, turbidity is measured using various methods. According to [25], these methods are either very inaccurate because they rely on the human observer or they are expensive (thousands of dollars), require a professional operation, and are not able to run real-time in large scale. This opens an opportunity to take advantage of computer vision. The general framework proposed by Liu et al. [25] for estimating turbidity with computer vision can be seen in figure 2.20.

The essential hardware requirements for the system based on computer vision require 3 main components: light source, image acquisition device, and image processing unit.

Furthermore, Liu et al. divide turbidity detection systems based on computer vision into 4 types depending on how the source image was acquired. They distinguish 4 types: sampled image, water surface image, underwater image, and invisible light image.

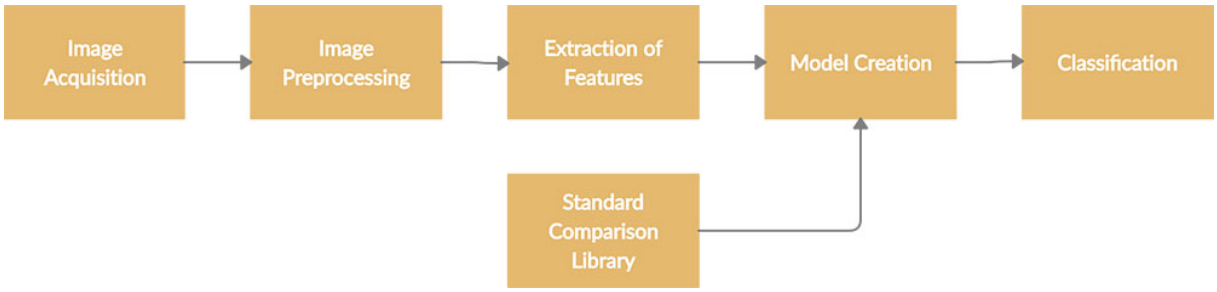


Figure 2.20: A common framework for turbidity estimation systems based on computer vision.

Sampled Image

Mullis et al [26] use a method called Sampled Image. The principle behind this method is to create an artificially controlled setup where the source image is acquired. Subsequently, the features are extracted and the relation between features and turbidity values is established. The setup used in the research consists of a vessel with sample water where the camera points to the vessel under angle $0°$ and artificial light is placed in the same plane in $90°$ The vessel is placed on top of a magnetic stirrer. The setup can be seen in Figure 2.21.

Underwater Image

As the name suggests, the source of the features is an image acquired from a camera submerged underwater. This camera can detect turbidity in various depths and it is suitable for fixed-point, long-term detection running in real-time. The drawback of this method is the cost because it requires an underwater camera, artificial light source, and additional operational requirements. This also brings new challenges because the background on underwater image constantly changes [25].

Water Surface Image

This remote sensing method utilizes water surface images to determine the turbidity. An interesting approach was proposed by Leeuw and Boss [27] where they use a mobile camera

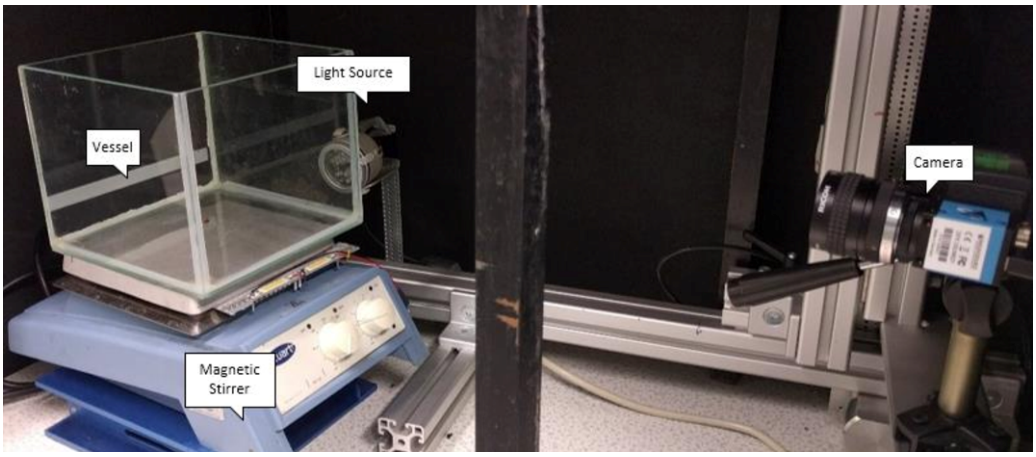


Figure 2.21: Controlled setup for measuring turbidity with a camera by Mullis et al [26].

which is a low cost and efficient solution. This method can also be used in real-time.
The problem with this approach is, however, a requirement for a huge amount of training data and small differences in sensors since the quality of cameras varies depending on the age of smartphone and company.

Invisible Light Image
Invisible light images used for turbidity detection can be captured with infrared or hyperspectral cameras. These images provide different features because the proportion of wave range from RGB cameras is limited. One of the approaches is presented by Hussain et al [18] based on Mie-scattering principle[7] where they made a portable tool that can be attached to a mobile phone. Advantage of this method is that the infrared sensor is able to detect turbidity of low NTU units with values such as drinking water has.

According to Liu et al [25] state-of-the-art performance can be achieved with manual feature extraction combined with the neural network. Manually extracted features by other researchers were luminance, color, shape, gray histogram, frequency-domain feature, gray feature, and gray gradient.

Now when we know how turbidity can be detected, we can finally look into related work in the field of underwater object detectors. The subsequent section will analyze relevant researches and find a suitable object detector algorithm for our problem.

[7]https://www.sciencedirect.com/topics/engineering/mie-scattering-theory

2.6 Evaluation Methods

In order to be able to evaluate object detectors, we have to define evaluation methods. In the field of computer vision, there are accepted metrics used for evaluating object detectors. In the Brackish dataset [3] the main metric they used is Average Precision (AP) which was also used by public domain datasets such as COCO [12] and Pascal VOC [10]. For the comprehension of AP, we have to first introduce primary building blocks that are used in calculating AP. Each prediction by a model can be divided into one of the 4 categories: FP (False Positives), FN (False Negatives), TP (True Positives), and TN (True Negatives). FP is when a model predicts that given condition will happen but it actually does not happen which is a type I error. FN, when giving condition, will not happen but it does that is a type II error. We are talking about TP and TN, on the other hand, when the model predictions are correct. These 4 categories are commonly summarized in the so-called Confusion Matrix (figure 2.22), which is frankly designed to explain the result of a classifier.

		$\hat{y}$ (Predicted)	
		0	1
y (Actual)	0	35	5
	1	10	35

Figure 2.22: Confusion matrix with example values. 35 in upper columns is TP, 5 is FP, 10 represents FN and last 35 is number of TN.

2.6.1 Intersection over Union

Intersection over Union (IoU) is a metric used for the evaluation of computer vision algorithms. It is also a base for other important metrics such as Average precision. It calculates the ratio between the ground truth bounding box and the predicted bounding box. This ratio is then thresholded. The prediction is either true or false if it passes the threshold or not. A popular public domain benchmark dataset Pascal VOC considers prediction as correct if the IoU is above 50% [10].
In order to apply IoU it is necessary to have:

1. Ground truth bounding boxes of objects from the testing set.
2. Predicted bounding boxes by a machine learning model.

If F and G denote rectangles with arbitrary lengths, the Intersection over Union can be

expressed with the following equation 2.1 The equation is visualized in figure 2.23 [10].

$$IoU = \frac{Area(\square F \cap \square G))}{Area(\square F \cup \square G)} \tag{2.1}$$

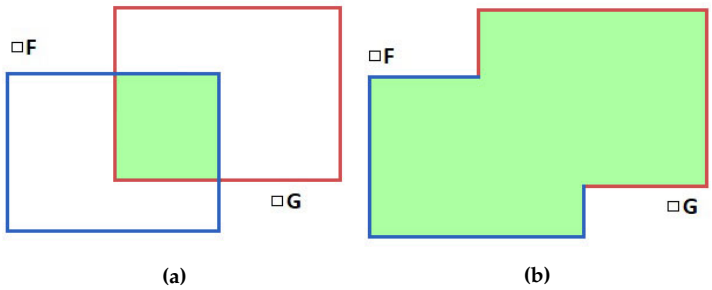


Figure 2.23: Figure (a) shows the Overlap area of the rectangles F and G. Figure (b) shows the Union area of the identical rectangles.

Algorithm 1 demonstrates an implementation of the IoU algorithm in pseudocode.

Algorithm 1 Intersection over Union

```
1:  procedure INTERSECTION_OVER_UNION(F,G)
2:      FArea = |F[2], F[0]| * |F[3], F[1]|
3:      GArea = |G[2], G[0]| * |G[3], G[1]|
4:
5:      Fx = max(F[0], G[0])
6:      Gx = max(F[1], G[1])
7:      Fy = min(F[2], G[2])
8:      Gy = min(F[3], G[3])
9:
10:     InArea = (Fy - Fx) * (Gy - Gx)
11: return InArea / (FArea + GArea - InArea)
```

2.6.2 Precision

The precision (equation 2.2) of a given class is defined as a ration between True positives divided by the sum of True positives with False positives. It measures the accuracy of the predictions, in other words, the percentage of predictions being correct.

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

2.6.3 Recall

Recall (equation 2.3) is defined as ratio between True positives and sum of True positives and False negatives. It describes how well we find all the positives.

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

2.6.4 Average Precision

Benchmark datasets (COCO, OpenImages) are evaluated with Average Precision (AP). AP is calculated as an area under the precision-recall curve and can be expressed by Equation 2.4. An example of this curve can be seen in Figure 2.24.

$$AP = \int_0^1 p(r)dr \tag{2.4}$$

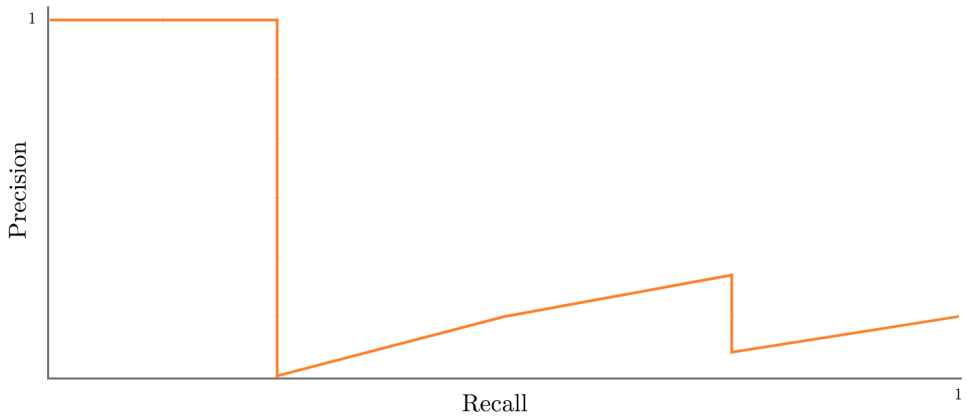


Figure 2.24: Precision and Recall plotted against each other.

To avoid approximation the precision is adjusted to be monotonically decreasing by replacing precision value with the maximum precision value to the right. This is illustrated with green line in Figure 2.25 [10]. After this step the are under the green curve is equal to average precision.

2.6.5 Mean Average Precision

Mean Average Precision denoted as mAP is AP calculated for each class of the dataset divided by number of classes. This can be shown by Equation 2.5 where i is number of classes and $AP(i)$ is AP of a single class.

$$mAP = \frac{\sum_1^i AP(i)}{i} \tag{2.5}$$

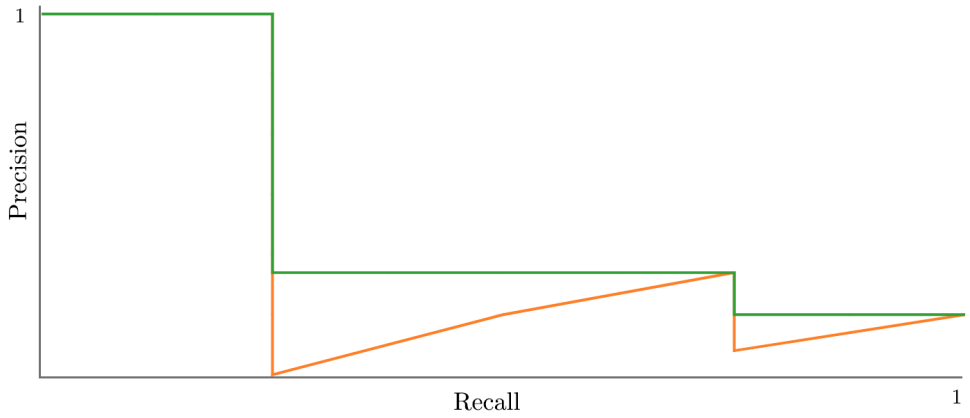


Figure 2.25: Precision and Recall plotted against each other with adjusted precision illustrated by gree line.

This is the end of the problem analysis chapter. The next chapter is going to introduce the final problem formulation of this master's thesis.

Chapter 3

Problem Formulation

By performing the problem analysis, we narrowed the problem down into the following:

- **Match or improve baseline results from Pedersen et al. [3]**:
  - Big Fish AP@[IoU = 0.5]: 0.8999
  - Crab AP@[IoU = 0.5]: 0.9271
  - Jellyfish AP@[IoU = 0.5]: 0.8205
  - Shrimp AP@[IoU = 0.5]: 0.7662
  - Small fish AP@[IoU = 0.5]: 0.6229
  - Starfish AP@[IoU = 0.5]: 0.9867

Moreover, the problem formulation consists of following question we will try to answer in this thesis:
How does turbidity affects performance of the models?
To our best knowledge there is no research that would study effects of turbidity on underwater computer vision object detector.

The next chapter is going to examine theory that is necessary for comprehension and carrying out the experiment.

Chapter 4

Theory

This chapter introduces what computer vision is and explains foundations of artificial neural networks which are currently the state-of-the-art in 2D object detection task.

4.1 Computer Vision

Computer vision (CV) is a computer science field that involves an analysis of digital images and videos. The type of information extracted could vary from augmented reality applications, measurements in space, or just identification [28]. It entails a set of algorithms that can understand image content and use it in different applications. It is an interdisciplinary field bringing together mainly physics, math, artificial intelligence, computer graphics, and others which are needed for the development of CV models.

Recent advances in machine learning computing capabilities increased data storage and lowered cost, high-quality input devices have improved computer vision. With the advent of camera phones, the world is increasingly being filled with billions of digital images whose viewing and analysis are past human capability. CV helps consumers in organizing and accessing their photos without the need for tags. The list of practical uses of CV is endless. Any futuristic situation would most likely require a CV application [29]. Facial recognition used in surveillance and security systems, payment portals, in social media, and even in retail stores to monitor inventories and tracking clients. According to Alhaija et al. [30], computer vision has made autonomous vehicles like Tesla and Ford a reality as they need to continuously process visual data. In healthcare, it helps in diagnosis, improves surgeon sight in operations and with most medical data being in the form of images, CV helps with the analysis of medical reports using algorithms.

The most common tasks in CV are:
Image classification - also referred to as object classification or image recognition, involves labeling images. It can either be a binary classification like marking an x-ray image as cancer or not or multiclass classification like naming a photograph or handwritten digits [31].
Object detection - a more advanced version of image classification is image classification with localization which involves labeling an image and showing its location by drawing a box around it (bounding box). Object detection is more advanced than image classification with

localization as it involves several objects in different types of images [32].
Object (semantic) segmentation - entails drawing a line around each object detected in the image. Unlike object detection, Chen [32] suggests that it identifies the specific pixels of an object in an image, segmenting them to different categories [32].
Image reconstruction - entails filling in parts of an image that may be missing or corrupt through a photo filler or transform that may not necessarily have an objective evaluation.
Image super-resolution - is the generation of a new version of an image with more detail and higher resolution than the original image [31].

When evaluating CV systems, it is necessary to have ground truth labels that are compared with predictions made by a CV model. The next section will break down the process of creating ground truth labels which are called data annotation.

4.2 Data Annotation

According to Schreiner [33], the data annotation or data labeling is a process of marking data acquired from a sensor. The data can be of any type, such as audio, images, text, time-series, etc. These annotations are used as ground truth for training and evaluating the accuracy of machine learning algorithms. For labeling the data, in this project, we are going to use software proposed by Bahnsen et al. [34]. Screenshot of the interface is presented in figure 4.1. The annotation software is divided into two annotation tools: Multimodal Pixel Annotator and Bounding Box Annotator. Pixel Annotator is used for marking pixels and assigning them into classes, while the Bounding Box Annotator is employed to create 2D bounding boxes which are rectangles enclosing the object area. The Annotation tool was implemented in C++ using OpenCV[1] and Qt [2]. The app is cross-platform supporting Windows, Linux, and macOS.

Annotations are usually stored as string in different formats such as XML[3], JSON[4], csv, etc.

In order to understand how the up-to-date CV systems work, we will first look into artificial neural networks in the subsequent section.

[1]https://opencv.org/
[2]https://qt.io/
[3]https://en.wikipedia.org/wiki/XML
[4]https://www.json.org/json-en.html

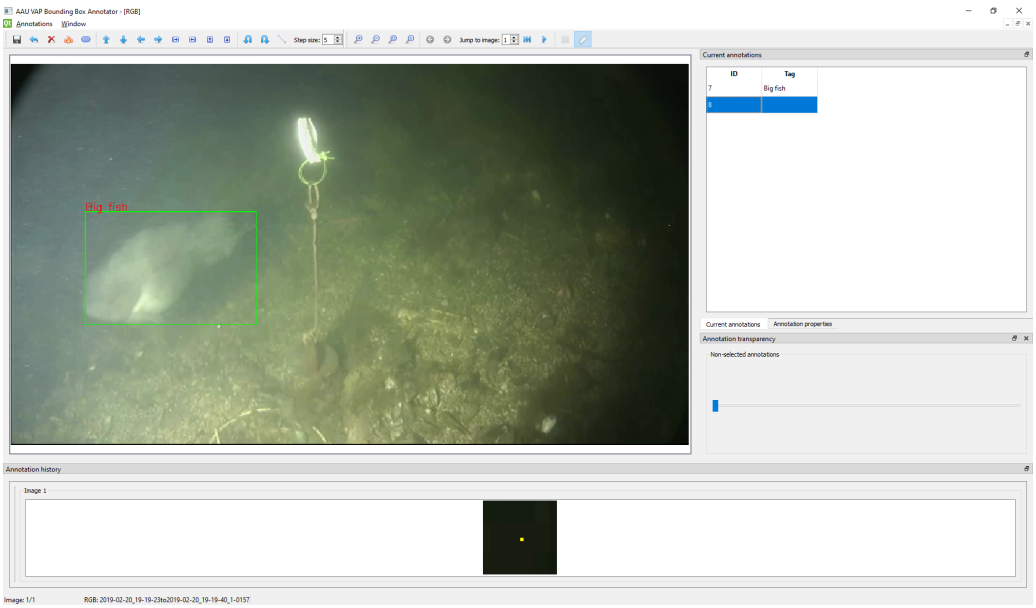


Figure 4.1: Annotating fish with the "AAU Multimodal Annotation Toolboxes" tool proposed by Bahnsen et al.

4.3 Artificial Neural Networks

Neural Networks, also known as Artificial Neural Networks (ANNs) are made to imitate biological neural network located in the brain of humans and animals. The general concept is to utilize the process of training or learning rather than using a precise set of programmed rules [35].

The Neuron (figure 4.8), also known as perceptron is the main building block of the ANNs. Neuron takes an input or multiple inputs from previous nodes, applies weights (learning parameters) to generate a weighted sum. Weights represent what is important and what signal will be passed through. The higher the weight is, the more impact the feature is going to have on the final result. Each neuron has also a bias that describes how flexible the neuron is. The sum of learning parameters is then passed to the activation function (ϕ) that computes the prediction or probability. The output of the perceptron is denoted as $\hat{y}$ and will be referred to as Predicted value. Predicted value can be continuous, regression, binary, or categorical. This architecture is also known as *Single Layer Perceptron* and can be seen in Figure 4.8 [35].

Basic ANN consists of 3 types of layers: Input layers, Hidden layers, and Output layers. The illustration of the ANN can be seen in figure 4.3. The Input layer depicts features that enter the network, for instance, color, height, price, or pixels of an image. The output layer is responsible for the output we want to predict, for example, what should be the price of an apartment, or whether skin stain is carcinogenic or not. Hidden layer stores neurons that are used for learning the features. The number of hidden layers depends on the complexity of our data and task we are trying to solve. First, the hidden layer can learn low-level features such as whether pixels are light or dark, the next layer is then able to extract higher-level features, for example, shapes and edges [35].

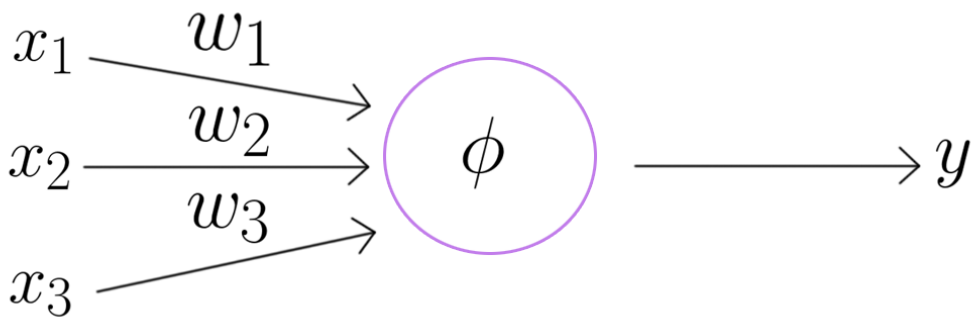


Figure 4.2: Illustration of an artificial neuron that has 3 inputs x_1, x_2, x_3 with weights w_1, w_2, w_3, activation function ϕ and output y.

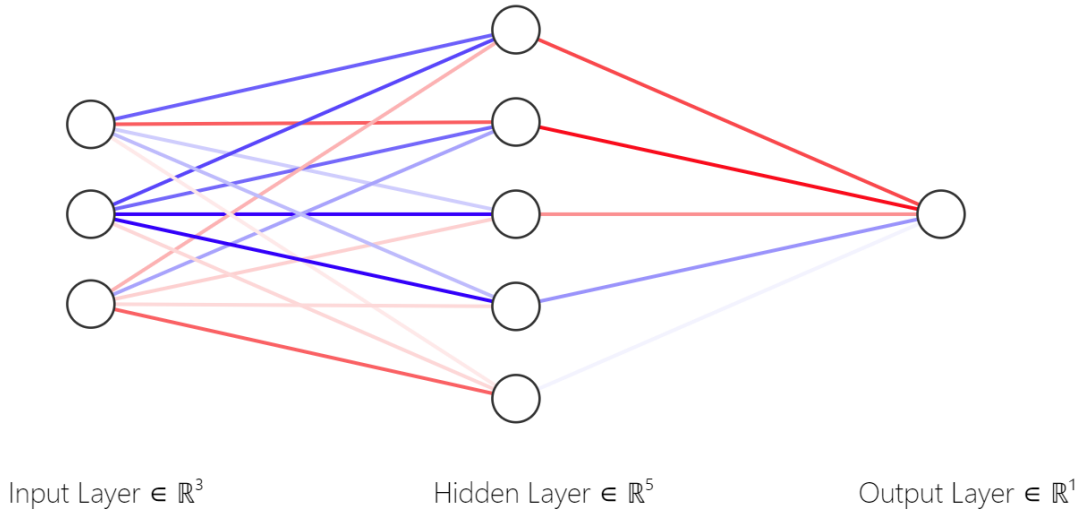


Figure 4.3: Illustration of fully connected artificial neural network with one hidden layer.

Activation function ϕ has to be chosen based on a specific problem there are attempting to solve. We will now look into 6 most used activation functions and explain their advantages and disadvantages. The mathematical equations of the activation functions are presented in table 4.1.

Binary Step is the simplest activation function which just outputs 1 or 0 based on whether the input value is below or over an arbitrary threshold. The binary step can be used for binary classifiers but for predicting multiple classes it is not useful.

Linear function is suitable only for simple tasks because the derivative of the linear function is only a constant which will result in biases and weights being updated by the same factor every time.

The **Sigmoid** function is non linear function best for predicting probabilities. For example, what is the probability that the picture is a dog or a cat? It is one of the most used activation

Activation Function	Equation	Range
Linear	$f(x) = x$	$(-\infty, \infty)$
Binary Step	$f(x) = \begin{cases} 0, if x < 0 \\ 1, otherwise \end{cases}$	$\{0, 1\}$
Sigmoid	$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$	$(0, 1)$
Hyperbolic Tangent	$f(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$(-1, 1)$
ReLU	$f(x) = \begin{cases} 0, if x < 0 \\ x, otherwise \end{cases}$	$< 0, \infty)$
Leaky ReLU	$f(x) = \begin{cases} 0.01 * x, if x < 0 \\ x, otherwise \end{cases}$	$(-\infty, \infty)$

Table 4.1: Comparison of the most used activation functions [36].

function.
Hyperbolic tangent works best for binary prediction, it is basically zero centered sigmoid function.
Rectified Linear Unit (ReLU) is a linear function that has been rectified. ReLU is less computationally demanding than Sigmoid and Hyperbolic Tangent function. Problem with ReLU is that some neurons might never get activated, this phenomenon is called dead neuron and it is addressed by Leaky ReLU.
Leaky ReLU is an improved version of ReLU where instead of function outputting 0 for all negative x values, Leaky ReLU returns small component of x to tackle the dead neurons problem.

The training process of the neural network is done utilizing *Forward propagation* and *Back propagation*. Backpropagation is used to minimize the error of the neural network by adjusting the weights until we get such an error which is satisfying enough for our system. The error is the difference between Predicted value and Actual value and it is expressed by a Cost function. However, calculating the optimal weights for the neural network may require a huge amount of computing power. This problem is solved by using *Gradient Descent*.

Gradient descent is a great way of finding global minima when our function is convex. Nevertheless, if the function is more complex, there is a chance that gradient descent finds a local minimum instead of the global one. This can lead to weights not being optimized which is undesirable. The solution is to use Stochastic Gradient Descent. The difference between them is that Gradient Descent adjusts weights based on the batch of samples, whereas Stochastic Gradient Descent updates them based on each sample.

When calculating gradient descent one has to first define cost function (Equation 4.1). Then the gradient descent is defined by first-order derivative as shown by equation 4.2 [35].

$$f(m, b) = \frac{1}{N} \sum_{i=1}^{n} (y_i - (mx_i + b))^2 \tag{4.1}$$

$$f^{'}(m,b) = \begin{bmatrix} \frac{df}{dm} \\ \frac{df}{db} \end{bmatrix} = \begin{bmatrix} \frac{1}{N}\sum_{i=1}^{n} -2x_i(y_i - (mx_i + b)) \\ \frac{1}{N}\sum_{i=1}^{n} -2(y_i - (mx_i + b)) \end{bmatrix} \tag{4.2}$$

The size of steps the gradient descent makes depends on the *learning rate* denoted as α. Larger the learning rate is, the learning process will be faster but might never reach the local minima. On the other hand, when the learning rate is too low, the learning process can be way too long.

In this section, we have described important building blocks of artificial neural networks and the process of learning. However, neural networks with only a single input and output layer are not sufficient for computer vision tasks on their own. In order to use neural networks for end-to-end object detection, deeper architectures are needed and we will analyze these in the following section called Deep learning.

4.4 Deep Learning

ANN with more than 1 hidden layer is referred to as Deep Neural Network (DNN). Utilizing multiple layers, DNNs are able to represent data with numerous levels of abstraction that leads to imitating the human brain even better. In comparison, ANN requires handcrafted features to be set as input, while DNN (figure 4.4) will find the features itself, without human interference. This, however, comes with drawbacks, for instance, having a high amount of training data is required, training time is much longer. Taking advantage of the recent improvements in hardware allowed deep learning to outperform previous state-of-the-art methods in various tasks such as object detection, image classification, and many others [28].

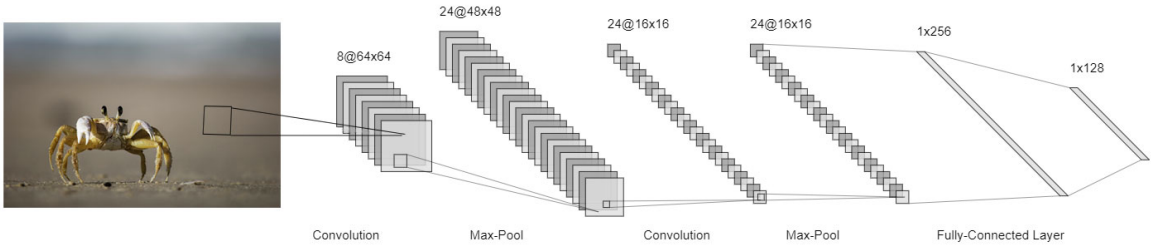


Figure 4.4: Architecture of Convolutional Neural Network.

Convolutional Layer
Convolution is a method for extracting feature maps from the input image using a feature detector (kernel). Multiple feature maps are created to obtain a convolutional layer. The goal of the convolution layer is to make the image smaller which will lead to lower processing times. By performing a convolutional operation, some of the information from the original image will be lost, but at the same time, the goal is to detect features of the image that are essential. This is done by using multiple feature detectors which results in multiple feature maps that preserve features that are important for the network [37]. When the feature maps are extracted an activation function takes into effect. In this case, ReLU is used in order to increase non-linearity which makes sense because images have lots of non-linear elements [38]. The convolution equation can be seen in equation 4.3 and it is visualized in figure 4.5. The feature detector iterates on image in a sliding window style.

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{4.3}$$

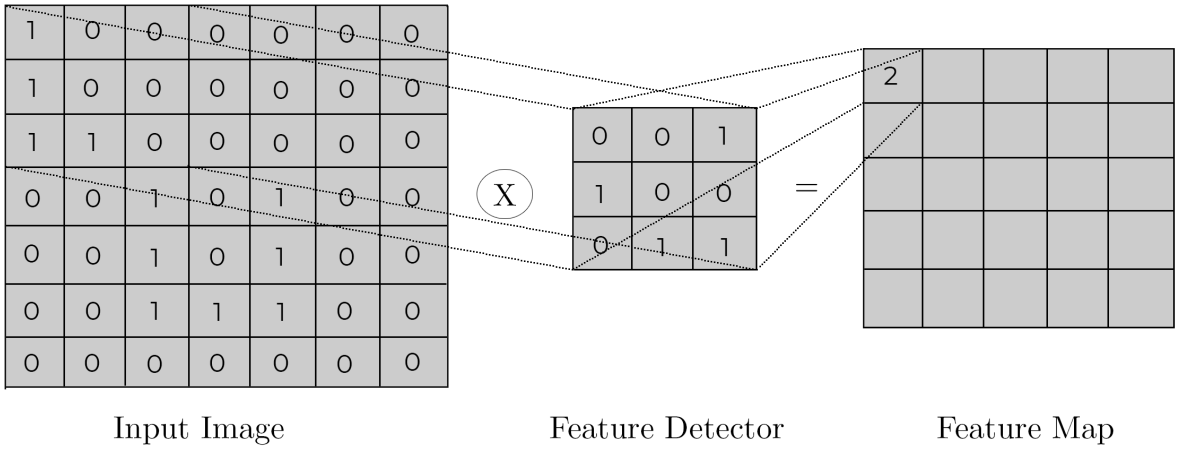


Figure 4.5: Illustration of the convolution operation. Feature detector is a kernel that iterates on the input image. Number of matches is then recorded in feature map.

Max Pooling Layer (Downsampling)

Max Pooling is another vital layer utilized in CNNs. The main purpose is to decrease the size of feature maps by disregarding information that is not necessary, while still being able to preserve important features. This results in CNN being more robust - invariant to rotations and other distortions, moreover it prevents over-fitting as we are removing information. The process of Max Pooling is straightforward. Similarly, as during convolution, we are using a sliding window with an arbitrary size that iterates over a feature map (Figure 4.6). The sliding window moves by steps which are also called strides. From each position of the window, the highest number is extracted it is saved in Pooled Feature Map that will be used in the next layer of the network. [39]

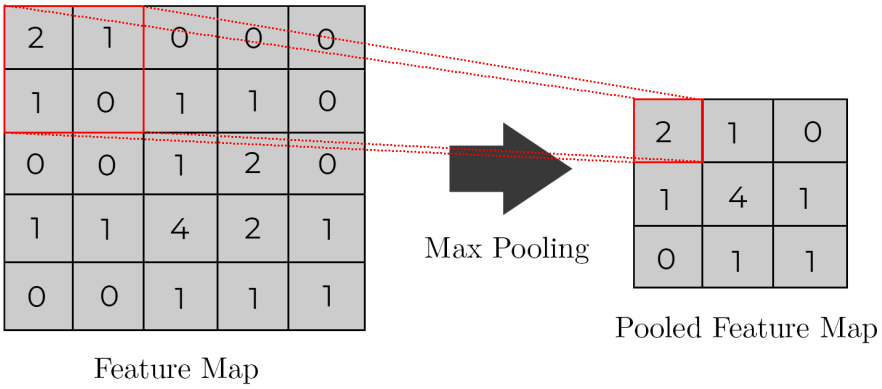


Figure 4.6: Illustration of Max-Pooling. The highest number from the red square in the right is recorded into the Pooled feature map.

Flattening Layer

Flattening Layer as the name suggest flattens the feature map by converting matrix of *m* rows and *n* columns into 1 dimensional vector with dimension of *1x(m*n)*. The values are then used as an input layer of an ANN.

Fully Connected Layer
Fully Connected Layer is a fully connected ANN that takes care of final classification predictions.

Softmax
Softmax is an activation function utilized in the output layer when we want from DNN to predict the probability of multiple mutually exclusive classes.

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}} \tag{4.4}$$

This is done by equation 4.4 where k is the number of neurons in the last layer, in other words, the number of classes we want to predict, z is a vector with length k - unnormalized predictions. z_j is jth unnormalized prediction from the last layer we want to normalize. This is divided by sum of e with exponent being unnormalized predicted values. The result is then normalized prediction where $\sum_{k=1}^{k} f_k(z) = 1$.

Cross-Entropy
Cross-Entropy (Formula 4.5) is the common loss function used to calculate the error of the network. Cross-Entropy is preferred over mean squared error for classification tasks, whereas mean squared error performs better for regression tasks. For this reason, Cross-Entropy is utilized in CNNs since it has the ability to punish more the classification error which affects the computation of the gradient and has an impact on the overall performance of the neural network.

$$H(y, \hat{y}) = -\sum_i y_i \log \hat{y}_i \tag{4.5}$$

4.4.1 Overfitting

When training deep learning models it is important to understand the concept of overfitting. This is because a model with the highest accuracy does not always mean this model is the best one. Overfitting is a phenomenon when deep learning models exactly model the training data so it does not generalize on testing data. This can be spotted when looking at an error/iteration graph (Figure 4.7). Overfitting occurs when error on the training dataset still decreases but the error on the test dataset changes its direction and starts increasing. When this happens, we choose a model from checkpoint before the error on the test dataset started increasing as Figure 4.7 shows [35].

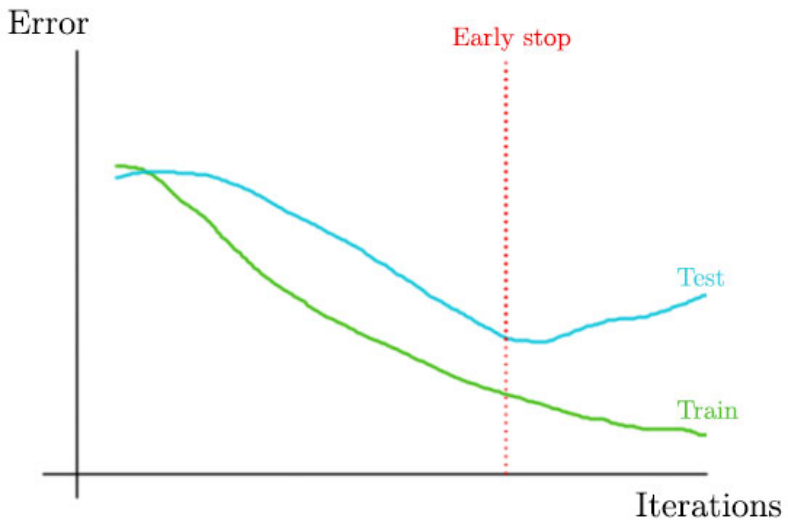


Figure 4.7: This figure demonstrates phenomenon of overfitting and early stopping point.

In this section, we have explained how deep neural networks can represent features from images. We have also learned that training deep neural networks is time-consuming and that phenomena such as overfitting can occur. Therefore, in the next section, we will look into transfer learning which is an inevitable part of training deep learning models and a common way of tackling the flaws in deep learning.

4.4.2 Transfer Learning

One of the common techniques to decrease training time is to transfer learning [40]. Transfer learning can be described as taking knowledge learned from other tasks and applying it to another task. This can be highly beneficial because a lot of the low-level features learned from a large relevant dataset can help our model to perform better on the new task. Using a transfer learning makes sense for example when pre-trained model is trained on a large amount of data, while the problem we are transferring our knowledge to, has notably fewer data. For instance, a model can be trained on millions of samples where it learns a lot of low-level features. Then this can be transferred to a problem where we have only 1000 samples.

To summarize: Transfer learning of task A to task B makes sense when:

- Task A and B have the same input x (similar task and same nature of data).
- We have a lot more data for Task A than Task B.
- If low-level features from Task A could be helpful in task B.

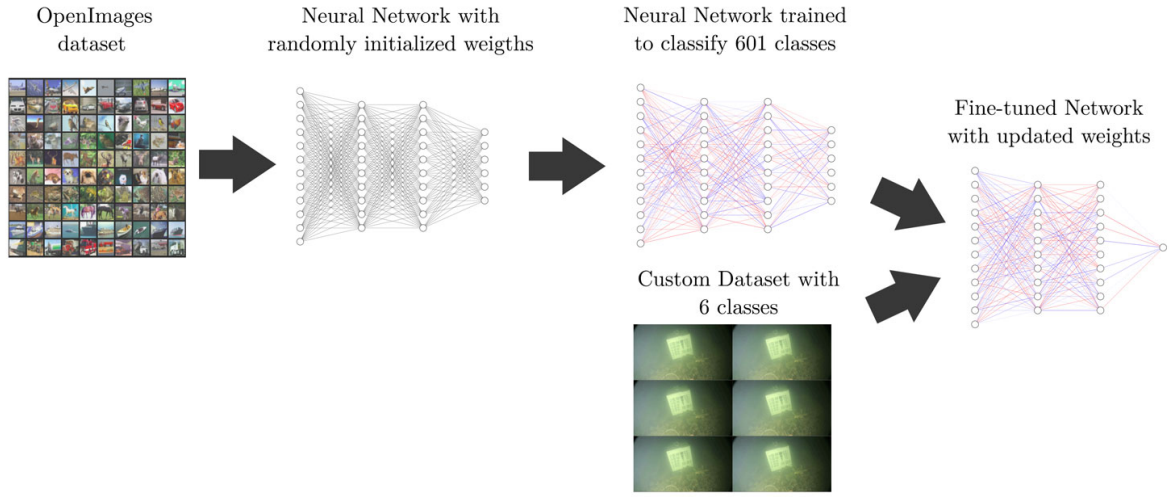


Figure 4.8: Illustration of Transfer learning in used in the Brackish dataset.

The current most used public domain benchmark datasets compared in table 4.2 are ImageNet[11], OpenImages [41], COCO[12] and Pascal VOC [10]. OpenImages is by far the biggest one. This, however, comes as a drawback because it contains a lot of false and ambiguous annotations. The deep learning model that performed the best in the results provided by Pedersen et al. [3] on the Brackish dataset was pre-trained on OpenImages. The relevant classes in the OpenImages for our problem are 6: Fish, Goldfish, Starfish, Jellyfish, Crab, and Shrimp. COCO and Pascal VOC have no relevant objects that could be used for animal detection underwater. In the ImageNet there are 11 relevant classes: Starfish, Goldfish, Lionfish, Anemone fish, Crayfish, Jellyfish, Dungeness crab, Rock crab, Fiddler crab, King crab, Hermit crab. Using backbone pre-trained on ImageNet appears to be a better choice than on OpenImages. This is because the focus of OpenImages is spread among 600 classes whereas ImageNet only 200. Furthermore, there are more relevant classes for our problem in ImageNet than in OpenImages.

Dataset	Images	Boxes	Classes	Boxes per Image	Annotated
Pascal VOC	11.5k	27k	20	2.4	Yes
ImageNet	477k	534k	200	1.1	Yes
COCO	123k	896k	80	7.4	Yes
OpenImages	1,515k	14,815k	600	9.8	Partially

Table 4.2: Comparison of the benchmark public domain datasets used for computer vision. More specifically for the 2D object detection task.

The next section is going to explain a deep learning algorithm called YOLO. It was utilized in the original Brackish dataset [3] and it is considered as state-of-the-art among 2D real-time object detectors.

4.5 You Only Look Once Algorithm

You Only Look Once algorithm also known as YOLO is a novel object detection system. It was firstly introduced in 2016 by Redmon et al. [42], later that year improved by introducing YOLOv2 [43] and finally the last upgraded version YOLOv3 was published in 2018 [44]. The reason why YOLO is so powerful is that a single neural network is used for predicting bounding boxes and class probabilities directly from an image in one run. Approaches before YOLO were mostly utilizing sliding window [45] or more recent approaches such as R-CNN [46] used region proposals. In other words, these approaches were originally object classifiers but were repurposed for object detection. Redmon et al. claim that these approaches had complex pipelines which resulted in slow detection that was hard to optimize because each of the individual components had to be trained separately. That is where YOLO stands out since they frame detection as a single regression problem without the need for a complex pipeline.

Figure 4.9: Shows predictions made by grid cells filtered with non max suppression. Thickness of a bounding box represents confidence of the detector that there is an object inside. Class of the object is going to be predicted in the next step.

YOLO uses a Grid Cell that predicts all bounding boxes simultaneously. It divides the image into a grid of SxS cells, each grid is then responsible for predicting 1 object. Each grid also predicts a fixed amount of bounding boxes B together with confidence scores for those boxes (figure 4.9). Confidence scores represent how confident the model is that the predicted box contains some object in it. Moreover, each grid cell also predicts C conditional class probabilities $P(Class|Object)$ which is the probability that the detected object belongs to a class. [42]

Using non-max suppression to remove low probability predictions. Then for each class use non-max suppression to generate final predictions.

Advantages in comparison with other methods: generating less false positives on back-

ground because it looks into the whole image and can see the larger context. YOLO can also learn generalizable representations of objects relatively well, in the original paper they trained YOLO on natural images and then compared Each cell predicts a fixed number of bounding boxes, however only one object, thus there is a limitation on how close the objects can be. It can run in real-time.

Since the initial YOLO publication in 2015, there have been 2 upgrades: YOLOv2[43] and YOLOv3[44]. They have stopped using arbitrary guesses of boundary boxes and started using the concept of anchor boxes. Moreover, batch normalization was added into convolutional layers, the new backbone was employed and a new activation function was used. These adjustments led to increased accuracy, speed, a decline of false positives which makes YOLO the state-of-the-art among real-time object detectors.

Describing the YOLO algorithm wraps up the chapter Theory. The subsequent chapter is going to describe what and how the experiments will be performed.

Chapter 5

Methodology

This chapter describes methods with reasonings that will be used for carrying out the research experiment in order to answer the outlined problem in Chapter 3. It concists of 3 modules that are outlined with all the steps thei contain in figure 5.1. Those modules are: 1. Deep learning solution, 2. Turbidity estimation, 3. Effect of turbidity on detection result.

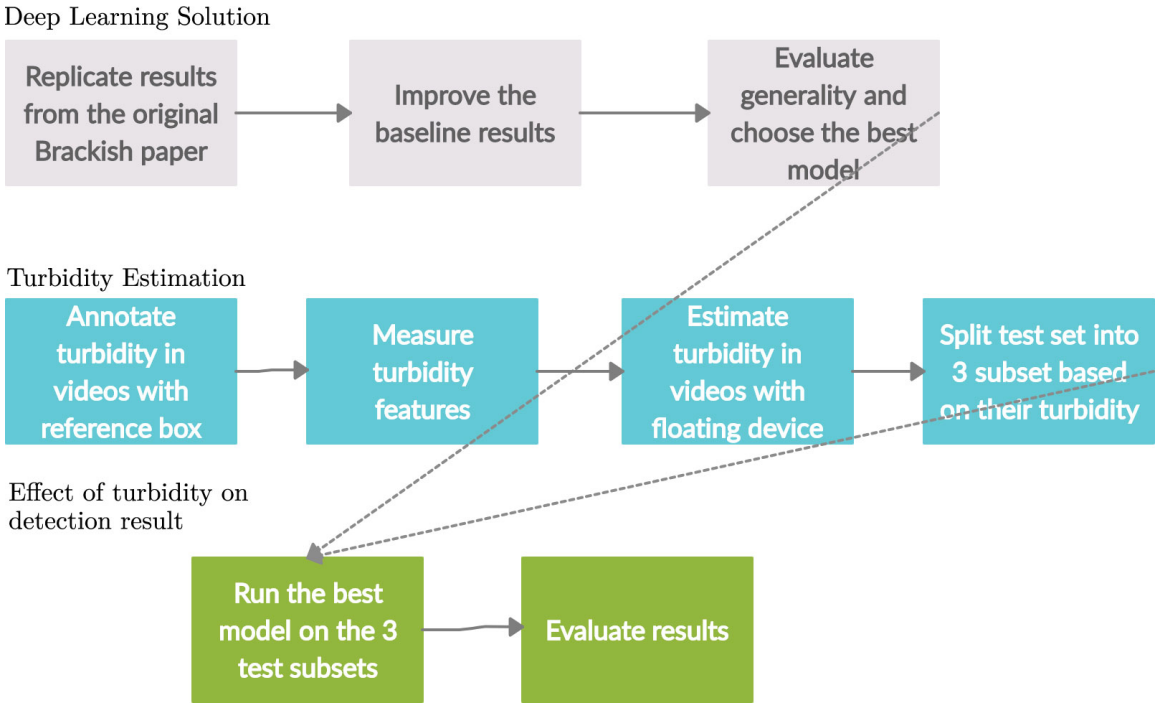


Figure 5.1: Overview of the proposed solution for the final problem formulation. It consists of 3 main parts: 1. Deep learning solution, 2. Turbidity Estimation and 3. Effect of turbidity on detection result.

Deep Learning solution

First of all, we need to decide whether we will create our own custom deep neural network or we will use some of the existing object detection networks. Authors of the Brackish dataset used state-of-the-art YOLOv2 and YOLOv3. YOLOv3 has shown to be a better option even

though they used different backbones in their experiment. This was expected becauseYOLOv3 has better accuracy on smaller objects than YOLOv2. The authors of the dataset also mention that their results are just baseline and have the potential to be improved. Because we see potential in improving their best results achieved with YOLOv3 and at the same time, YOLO is still considered a state-of-the-art, we will also use YOLOv3 for our experiment as well. The improvements could be accomplished by utilizing different feature extractors described in subsection 4.4.2 - Transfer learning. Moreover, increasing input resolution of the network could increase the accuracy of Small fish since according to Mahony et al. [21], image resolution is important when we want to detect objects in distance.

After the training, we will choose the best performing model and proceed to the next stage. The best performing model will be chosen based on mAP metrics described in section 2.6. However, as we learned, mAP does not describe how well the deep learning model generalizes on a new unseen data. To tackle this, we will introduce a new testing dataset called the **Brackish X** dataset. This dataset will consist of new data acquired from the same place in the Limfjord, however, with the camera tilted to a different side.

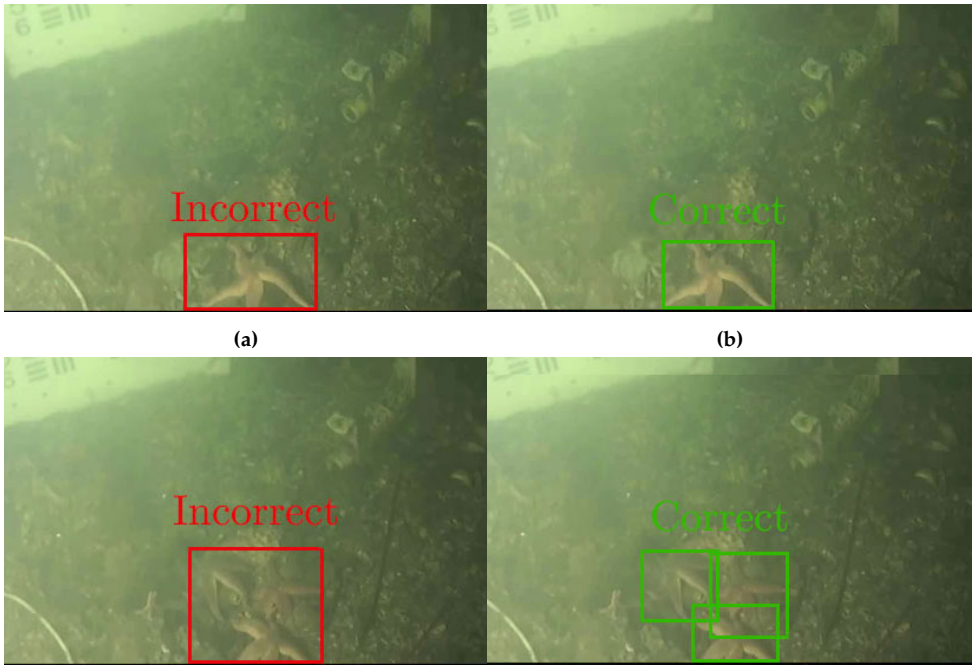


Figure 5.2: This figure shows examples of the most frequent mistakes during annotation and how the correct annotations should be. (a) - the bounding box is not tight enough, (b) - bounding box is correctly tight, (c) - single bounding box covers incorrectly multiple starfish (d) - each starfish has its own bounding box.

The original paper of the Brackish dataset [3] lacks annotating guidelines. Thus, based on a heuristic review of the Brackish dataset, we introduce the following rules that will be applied for annotating the new Brackish X dataset.

- The bounding box has to cover all the visible area of the animal.

- The bounding box has to be drawn as tightly as possible.
- The bounding box should include only one instance of the animal.
- If the animal is occluded, it will be annotated only when the visible part of the animal is more than 30%.
- If it is not possible to identify an animal from a single frame, for instance when turbidity is too high. In this case, the later sequence of the video can be observed to find out what class the animal belongs to. If it is not possible to identify the animal, these frames are not going to be included.

The common mistakes that annotators made are shown in figure 5.2.

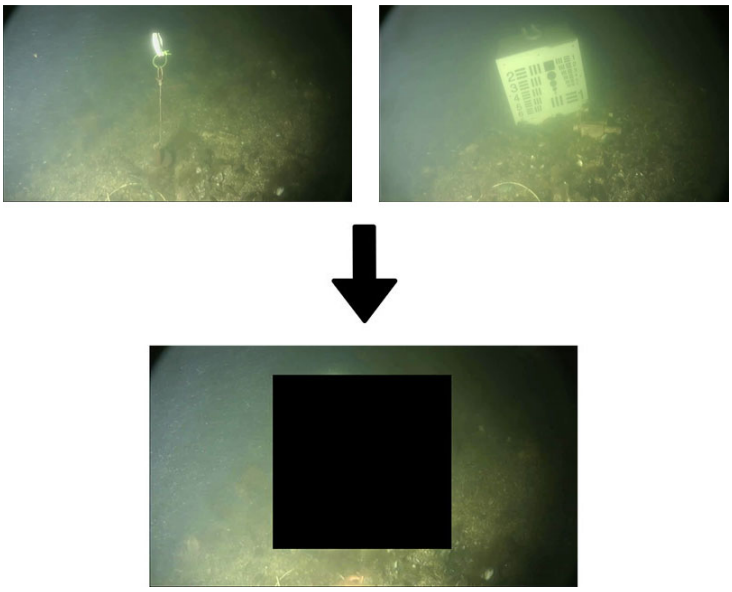


Figure 5.3: The upper left side of the image contains a floating device. The upper right side contains a reference block. The central part of the bottom image is cropped out. The outer area is our region of interest.

Turbidity Estimation

In order to understand how does the turbidity influence results of the underwater object detector, it would be beneficial to have a system that could detect turbidity levels in brackish water. In section 2.5 it was found out that most used features for detecting turbidity are gradient image, sharpness, contrast, and luminance. An experiment will be carried out to find out which of the mentioned features can characterize turbidity levels in the Brackish dataset. This information would be useful for future turbidity detector in brackish waters.

In the Brackish dataset, there are frames with reference block and floating device in them. These were used for other research purposes. We want to remove possible bias and thus, in order to use all the frames, we cropped the central part of the image that differs and kept only the outer part (Figure 5.3). The part of the image that remained will be referred to as Region of interest (ROI).

In order to find out which feature is the best for detecting turbidity in brackish water, we will employ Forest of trees method[1] used for evaluating feature importance and K-means clustering algorithm[2] [47].

To increase the reproducibility of this research, we will calculate the turbidity features with the following equations:

Brightness - for computing brightness we have employed the following pseudocode 2.

Algorithm 2 Computing brightness from a single image.

```
get image histogram
number of all pixels = width x height
for all the bins in histogram do
    ratio = number of pixels in current bin / number of all pixels
    brightness = brightness + ratio * (index - scale)
end
return brightness / scale
```

Sobel edge intensity (Sobel) is defined by applying 2 kernels to the image. G_x takes care of the vertical direction, whereas G_y of the horizontal direction. This is demonstrated by equation 5.1.

Then the gradient image is expressed by gradient magnitude defined by equation 5.2. Then we summed up all the intensities and divided it by the number of pixels.

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * I; G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * I \tag{5.1}$$

$$G = \sqrt{G_x^2 + G_y^2} \tag{5.2}$$

Sharpness can be estimated by averaging the gradient magnitude as illustrated by equation 5.3.

$$\|\Delta F\| = \sqrt{\left(\frac{\delta F}{\delta x}\right)^2 + \left(\frac{\delta F}{\delta y}\right)^2} \tag{5.3}$$

Luminance of an image is calculated with equation 5.4. M stands for width and N for height of an image. R, G, B stands red, green and blue component respectively.

$$Y = \frac{1}{MN} \sum_M \sum_N .2126R + .7152G + .0722B \tag{5.4}$$

[1] https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html
[2] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

Root Mean Square (RMS) Contrast is computed with equation 5.5 which is just a standard deviation of R,G,B pixel intensities. M stands for width and N for height of an image. I_{ij} is the i-th j-th pixel, and $\bar{I}$ is the average intensity of all pixels.

$$RMS = \sqrt{\frac{1}{MN}\sum_{M}\sum_{N}(I_{ij}-\bar{I})^2} \tag{5.5}$$

We have included python snippets for all the feature in Appendix E.

Effect of Turbidity on Detection Results
Once the turbidity will be annotated, and deep learning model will be trained, we will split the original Brackish test dataset into 3 sub-datasets based on the estimated level of turbidity. The best performing deep learning model will be chosen based on their mAP performance on the new Brackish X dataset. Detection runs on each of the datasets. Finally, the results will be analyzed.

This is the end of the Methodology chapter. The next chapter is going to take care of presenting results.

Chapter 6

Results

This chapter presents results from the proposed solution in the previous chapter 5 - Methodology. It follows the same structure as outlined in Figure 5.1: 1. Deep learning solution, 2. Turbidity estimation and finally 3. Effect of turbidity on the detection result.

6.1 Deep Leaning Solution

The first task is to match or improve baseline results from Pedersen et al. [3]. For this we have trained 6 YOLOv3 models with different settings.

6.1.1 Description of the models

All models were trained for 30 000 iterations except model 4# where early checkpoint on 12 000 iteration was used. All models were evaluated using AP and mAP which were introduced previously. A confidence treshold of 0.01 is chosen, same as in original Brackish paper [3]. Summary of the results can be seen in Table 6.1.

0#Pedersen et al. - Baseline results from the original paper.

1#YOLOv3 - This is our attempt to replicate results from the original paper utilizing the same training settings. Resolution 416x416, pre-trained weights on OpenImages and freeze of the starting layers.

2#YOLOv3 - Using the same settings as the model 1#, however, with increased precision from 416x416 to 608x608.

3#YOLOv3 - This model is overfitted on purpose. We changed the backbone to ImageNet, we used the original 416x416 resolution, and turned off the freeze.

4#YOLOv3 - This model uses the same settings as 3#, however, early checkpoint on 12 000th iteration is used.

5#YOLOv3 - Same settings as the original paper, however, the different backbone is used.

6#YOLOv3 - Increased precision to 608x608.

Model	Backbone	Input size	**mAP**	Big Fish	Small Fish	Shrimp	Crab	Jellyfish	Starfish	TP	FP	FN
0#Pedersen et al. [3]	★	416x416	83.72	89.99	92.71	82.05	76.62	62.29	98.67	-	-	-
1#YOLOv3	★	416x416	87.24	90.27	73.66	80.09	97.64	82.67	99.12	2243	3373	156
2#YOLOv3	★	608x608	86.4	92.84	83.75	79.97	96.76	65.87	99.19	2257	2462	142
3#YOLOv3	△	416x416	97.96	99.26	96.35	99.97	99.99	92.57	99.59	2367	336	32
4#YOLOv3	△	416x416	95.71	96.48	90.65	96.94	99.34	91.27	99.59	2329	1444	70
5#YOLOv3	△	416x416	91.87	94.48	83.81	86.62	98.25	88.46	99.59	2254	1172	145
6#YOLOv3	△	608x608	92.92	95.78	84.48	87.65	98.23	91.81	99.59	2264	1033	135

★ - OpenImages
△ - ImageNet

Table 6.1: Results evaluated with mAP for all our trained models compared with the results from the original paper.

The next section is going to present the Brackish X test dataset which we proposed for verifying generality of a computer vision system trained on the Brackish training dataset.

6.2 The New Brackish X Dataset

The Brackish X dataset is a manually annotated dataset proposed in this master thesis. Its main purpose is to evaluate the generality capabilities of the underwater computer vision system trained on the original Brackish training dataset.

The Brackish X dataset was capture mostly in the same position as the original Brackish dataset. However, it also contains frames from a completely new position which was achieved by tilting the whole underwater capturing setup. An example frame can be seen in Figure 6.1 together with annotations containing crabs, small fish, and starfish. The whole environment together with the animals differs a lot from the original training data. As demonstrated in Figure 6.2, 71.67% of the frames in the Brackish X dataset are from the original position and the rest 28.33% is from the new one, similar to Figure 6.1.

Figure 6.1: Frame from the new camera position containing 3 crabs, 1 starfish and 1 small fish.

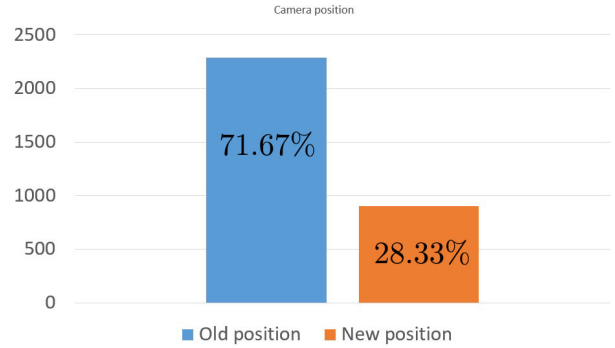


Figure 6.2: Ratio of frames from the old camera position and new camera position in the new Brackish X dataset.

The whole Brackish X dataset consists of 3 254 manually annotated frames. All in all, there are 14 545 annotations. The distribution between classes can be seen in Table 6.2. Visual distribution of the classes based on their position is shown in Figure 6.3. Note that there are no shrimps in the new position and there are no jellyfish at all in the whole dataset.

Class	Annotations
Big fish	789
Crab	5 582
Jellyfish	0
Shrimp	139
Small fish	3 331
Starfish	4 704

Table 6.2: Number of annotations for each class in the new Brackish X dataset.

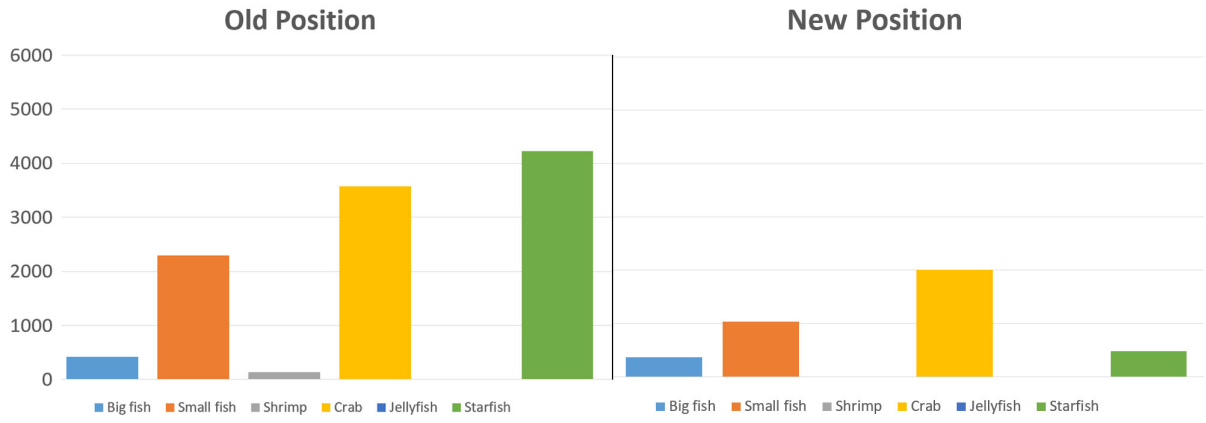


Figure 6.3: Distribution of the annotations based on position they were taken from.

The next section is going to present results from evaluating our trained models on the Brackish X dataset.

6.3 Evaluation on the Brackish X

This section presents the results of our 6 trained models that were trained on: 1. The whole Brackish X dataset (Table 6.3), 2. The Brackish X dataset old camera position (Table 6.4) and 3. The Brackish X dataset new camera position (Table 6.5).

6.3.1 Result on the Brackish X dataset

Model	Backbone	Input size	**mAP**	Big Fish	Small Fish	Shrimp	Crab	Jellyfish	Starfish	TP	FP	FN
1#YOLOv3	★	416x416	13.8	0	25.82	11.23	13.56	-	18.39	2888	31959	8463
2#YOLOv3	★	608x608	13.63	0	16.88	12.74	23.69	-	14.86	2846	18237	8505
3#YOLOv3	△	416x416	18.2	0	36.99	24.82	20.22	-	8.99	2122	4233	9229
4#YOLOv3	△	416x416	17.98	0	26.37	32.92	20.96	-	9.67	2283	4944	9068
5#YOLOv3	△	416x416	7.674	0	20.31	4.93	6.28	-	6.85	1355	9926	9996
6#YOLOv3	△	608x608	11.1	0	13.34	12.86	17.52	-	11.78	2358	10986	8993

★ - OpenImages
△ - ImageNet

Table 6.3: Results evaluated with mAP for all our trained models on the whole Brackish X dataset.

6.3.2 Result on the Brackish X dataset old position

Model	Backbone	Input size	**mAP**	Big Fish	Small Fish	Shrimp	Crab	Jellyfish	Starfish	TP	FP	FN
1#YOLOv3	★	416x416	25.06	0	67.93	14.08	22.79	-	20.5	2583	4674	5779
2#YOLOv3	★	608x608	23.14	0	52.31	15.38	31.08	-	16.95	2524	4446	5838
3#YOLOv3	△	416x416	27.5	0	69.76	28.24	29.47	-	10.04	2019	2402	6343
4#YOLOv3	△	416x416	22.35	0	55.31	17.85	28.71	-	9.89	2199	5944	6163
5#YOLOv3	△	416x416	15.44	0	54.54	6.6	9.66	-	6.39	1292	5074	7070
6#YOLOv3	△	608x608	21.63	0	58.21	13.31	26.78	-	9.86	2104	4111	6258

★ - OpenImages
△ - ImageNet

Table 6.4: Results evaluated with mAP for all our trained models only on the old camera position of the Brackish X dataset.

6.3.3 Result on the Brackish X dataset new position

Model	Backbone	Input size	**mAP**	Big Fish	Small Fish	Shrimp	Crab	Jellyfish	Starfish	TP	FP	FN
1#YOLOv3	★	416x416	1.79	0	0	-	1.31	-	5.83	305	27285	2684
2#YOLOv3	★	608x608	2.85	0	0	-	11.39	-	0	322	13791	2667
3#YOLOv3	△	416x416	1.11	0	0	-	4.44	-	0	103	1831	2886
4#YOLOv3	△	416x416	3.49	0	0	-	13.46	-	0.48	260	4098	2729
5#YOLOv3	△	416x416	3.4	0	0	-	0.55	-	13.03	63	4852	2926
6#YOLOv3	△	608x608	8.55	0	0	-	0.64	-	33.56	254	6875	2735

★ - OpenImages
△ - ImageNet

Table 6.5: Results evaluated with mAP for all our trained models only on the new camera position of the Brackish X dataset.

The next section is going to present results of the turbidity features proposed in the chapter 5 - Methodology.

6.4 Turbidity Estimation

For each of the turbidity features the same experiment was performed where we measured and analyzed signals from frames with the referenced box in them. Namely, we have extracted Sobel edge intensity, sharpness, luminance, contrast, and brightness. Results can be seen in the Appendix B.

For evaluating which of the features describe the turbidity the best we have utilized Random forest feature importance. Result is shown in Figure 6.4. From the experiment we have learned that sharpness is the best for detecting turbidity in our research case, second is Sobel edge intensity. Contrast and luminance are not that descriptive and brightness is negligible.

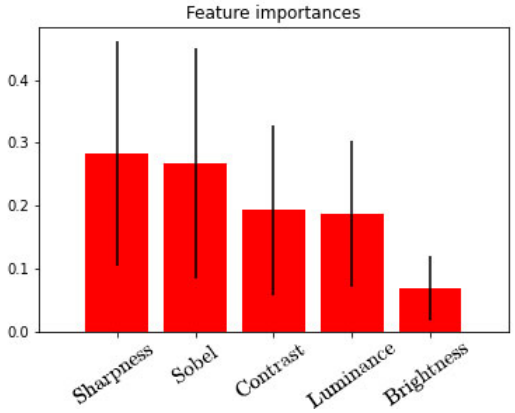


Figure 6.4

The Sobel and Sharpness were selected for classifying turbidity levels in videos with a floating device in them. For this, we have utilized the unsupervised clustering algorithm K-means. The result can be seen in Figure 6.5.

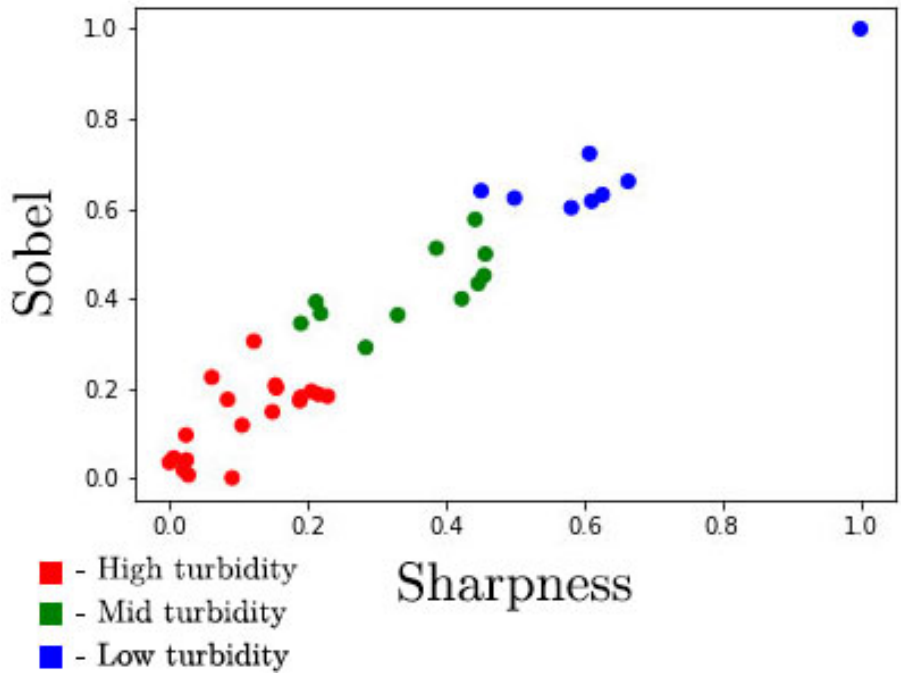


Figure 6.5: Visualized output of the K-means. Each point on the Figure represents averaged signal from single video with floating device.

6.5 Effects of Turbidity on Detection Results

The model that performed the best on the Brackish X dataset was model 3# which is the overfitted one. This means we will have to choose the best model based on the mAP result only from the Brackish X new position subset. The best performing model was model 6#. Table 6.6 shows the performace of the model 6# on 3 subsets of the original Brackish dataset divided by estimated turbidity levels.

Model	**Turbidity**	**mAP**	Big Fish	Small Fish	Shrimp	Crab	Jellyfish	Starfish	TP	FP	FN
6#YOLOv3	Low	90.89	97.21	87.33	70	99.63	91.91	99.29	1104	388	63
6#YOLOv3	Mid	93.61	97.21	82.35	94.01	96.98	-	97.5	342	182	10
6#YOLOv3	High	92.66	94.58	80.2	84.51	97.26	100	99.41	818	463	62

Table 6.6: Results evaluated with mAP for 3 subsets of the Brackish test dataset divided by estimated turbidity levels.

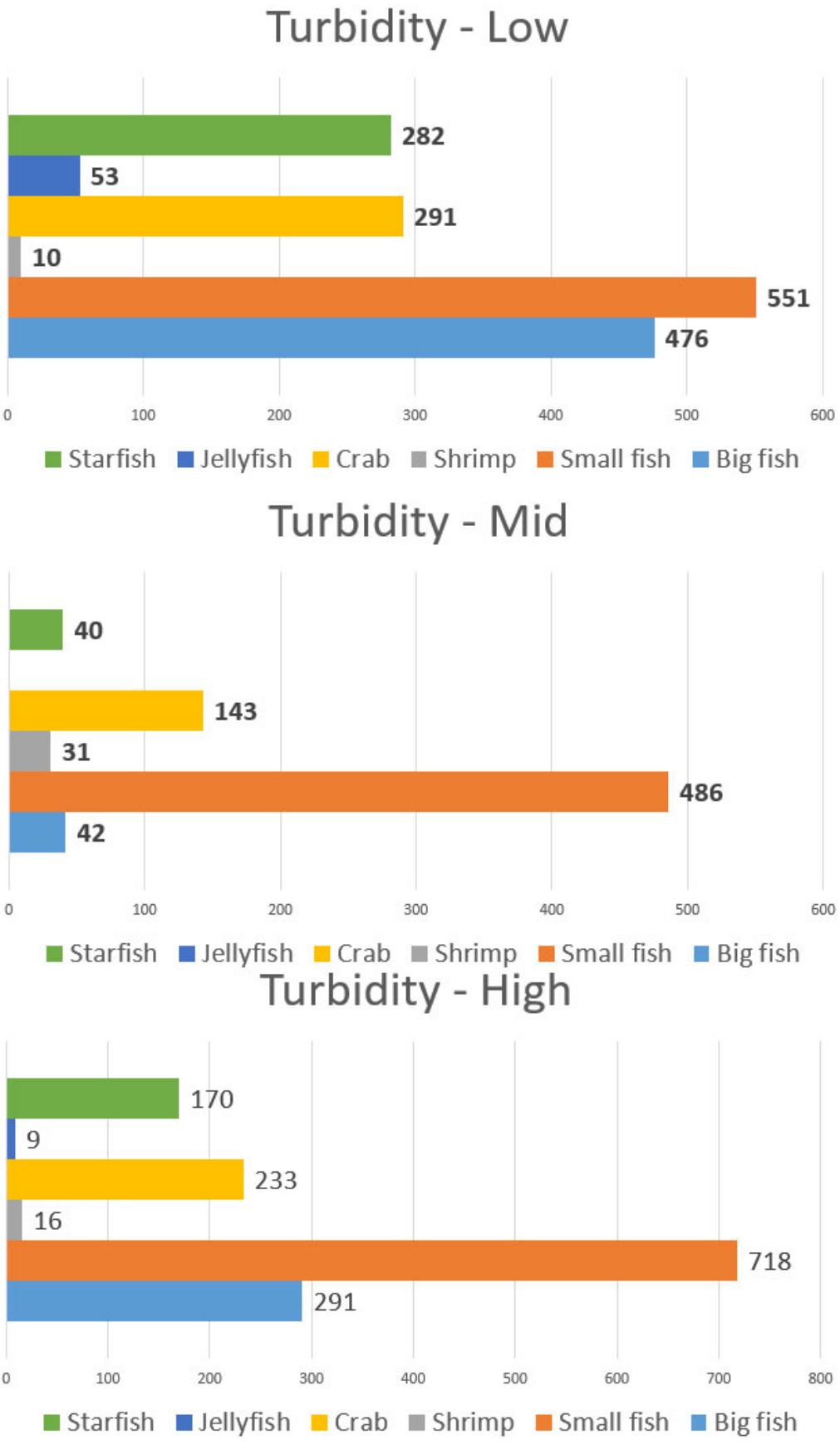


Figure 6.6: The figure shows the unbalanced distribution of objects among classes in the original Brackish dataset divided based on estimated turbidity levels.

Chapter 7

Discussion

Deep Learning Solution
From our first performed experiment presented in Table 6.1 the best model which was over-fitted on purpose achieved mAP of 97.96%. When comparing the baseline results from the original paper with 83.72 mAP (model 0#) using weights pre-trained on OpenImages we have achieved an increase by 8.15 mAP just by changing the backbone to ImageNet which is much more similar to our detection problem. Then changing the input size, first on the model 2# pre-trained on OpenImages to 608x608 it increased mAP on small Fish from 73.66 to 83.75. However, the jellyfish decreased from 82.67 mAP to 65.87. It seems like when there are more features, it is harder to detect jellyfish. When increasing the input size of the model 5# pre-trained on ImageNet it led to a slight increase of mAP by 1.05 and a decrease of false detections by 149.

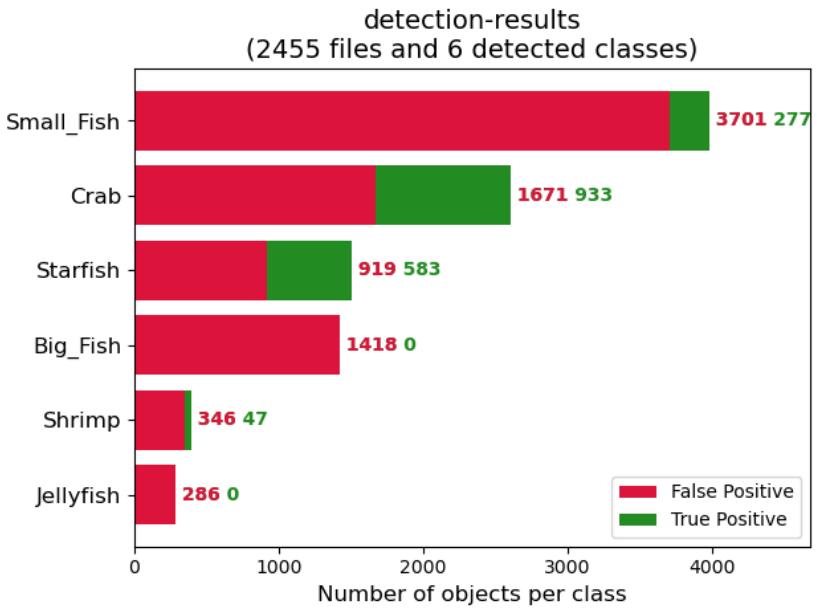
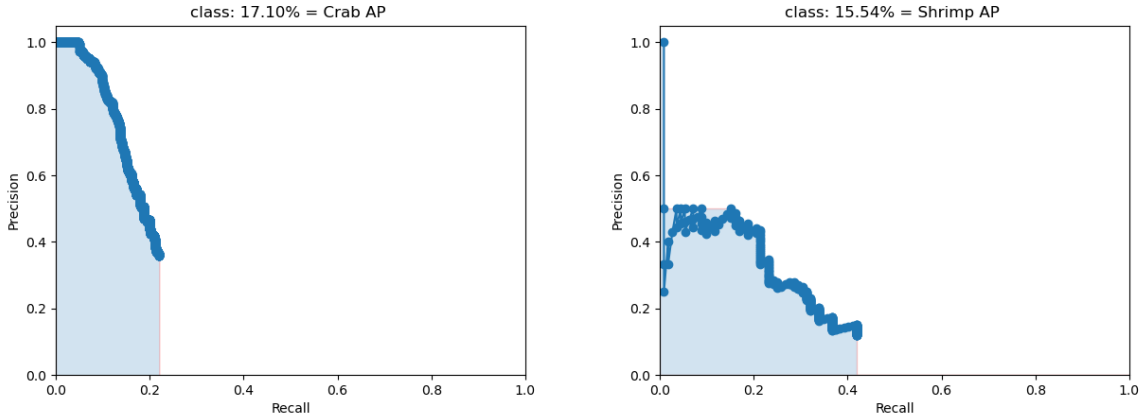
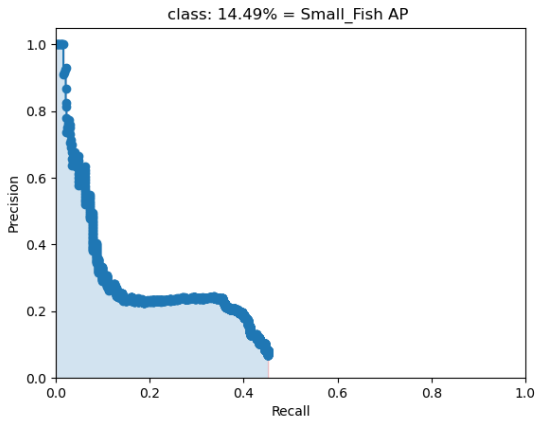


Figure 7.1: Detection results from the whole Brackish X dataset. Evaluation of the model 6# which performed the best on the Brackish X new position.

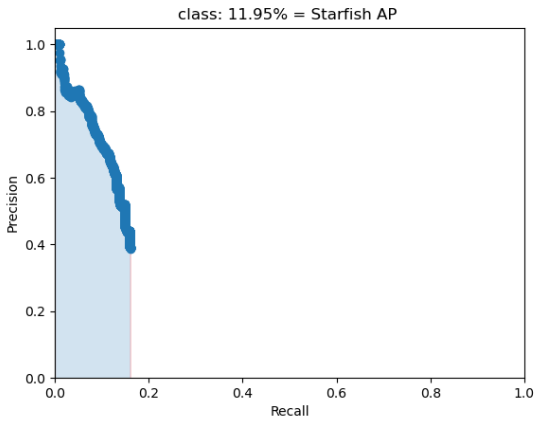


(a) AP Crab

(b) AP Shrimp



(c) AP Small Fish



(d) AP Starfish

Figure 7.2: Precision recall curves plotted for calculating AP. Big fish class is not plotted because there were no true positives and Jellyfish class because there are no jellyfish in the Brackish X dataset whatsoever.

In the next step, we have presented the new Brackish X dataset where the majority of data comes from the old camera position and only 28.33% from the new one. All the data are videos that the models have not seen during the training.

Firstly, we have evaluated all our 6 trained models on the whole Brackish X dataset that can be seen in Table 6.3. The highest mAP of 18.2 achieved the overfitted model 3#. This would mean that the Brackish X dataset is not suitable for the purpose it was created that was testing the generality capabilities of the trained model. Thus, we further split the dataset into the old position and the new position. Results from the new position can be seen in Table 6.4 where model 3# achieved the highest accuracy of 27.5 mAP. The results of the models evaluated on the new position can be seen in Table 6.5. The overfitted model 3# had the lowest mAP of 1.11. The best model was the model 6# with mAP of 8.55. From the table, we can see that the number of false detections on models pre-trained on OpenImages is 3-6 times higher than on the models pre-trained with ImageNet.

The best model achieved performance on Starfish class as high as 33.56, this is probably because of its distinctive star shape. The Crab AP was only 0.64 and Big fish together with

Small fish class had AP equal to zero. The reason behind the poor performance (Figure 7.1 and Figure 7.2) is that the animals in the new camera position look very different from the animals in training data. This means that there is a need for more training data.
The models in on the original test Brackish dataset achieved great performance mainly because the test dataset consists of randomly chosen frames from the training data. The time difference between the frames is low, thus the test data look very much alike as the training data.

Turbidity Estimation
First of all, we have manually annotated the videos with the reference box in them. Then we performed an experiment where we measured the importance of turbidity features in the Brackish dataset. The best performing features were Sharpness and Sobel edge intensity. Those 2 features were used for estimating turbidity in the videos with the floating device in the using K-means clustering (Figure 6.5).

The issue with this approach is that we are not sure about the real turbidity in the videos. Even though using computer vision for estimating turbidity can be a relatively cheap and easy way, we lack ground through information validated by a turbidimeter.

By cropping out the central part of the image, we have lost the most distinct turbidity features. However, it was a necessary step to make all the frames equal. We were thinking about utilizing only the frames with the reference box, however, the majority of videos contain floating device. We would lose the majority of testing data in this case.

Effects of turbidity on detection result
The best model was the model 6# since it achieved the best accuracy of 8.55 on the Brackish X new position. We have divided the original Brackish test dataset into 3 subsets based on their estimated turbidity level. The results can be seen in Table 6.6. Looking at the table we can observe that mAP is quite similar. We expected that there will be a higher gap between the mAP. Also, we hypothesized that low turbidity will create much more false detections because of the higher amount of features present in the frames. This cannot be concluded from our experiment. The reason is that the distribution of the animals in the 3 subtests is uneven (Figure 6.6). Moreover, the animals in the scenes behave differently, which makes it difficult to compare.

Chapter 8

Future Work

Underwater Object Detection
The results of our deep learning solution on the Brackish test dataset was satisfactory (mAP 92.92%). However, the results on the new Brackish X dataset were poor even in the old position which open possibilities for improvements. Firstly, we have found annotation errors and ambiguities in the original Brackish dataset. The videos with incorrect annotations are mentioned in Appendix C. Secondly, the results could be further improved by utilizing more complex data augmentation such as mixing Images, meta-learning or deep learning augmentation approaches. We have studied and described these methods and can be found in Appendix F.

Thirdly, for future work, there is a need for a much larger dataset in case there is a need for a reliable deep learning-based computer vision system.

Moreover, The manual annotations we made in this master thesis are only student annotations and cannot be considered expert level. The annotation process should consist of a pipeline with multiple people called annotators and validators as Shao et al. suggest [48]. Because of the nature of our dataset, a biologist should be present to verify the annotations. Thus, because of the limited workforce, there might be minor errors in the dataset that should be verified before using it in future work.

Lastly, the crucial thing is that researcher will have to keep in mind when working with data acquired from Limfjord is that there is much more fish species that they occur in the Brackish dataset. This is because there are about 200 living fish species in the North Sea and some of them can also occur in the Limfjord.

Effects of turbidity on detection result
As we found out in this project, using the Brackish dataset [3] it was not possible to come with any useful conclusion. This is because the amount of animals in each test dataset divided by turbidity into low, mid, and high is always different. To explore this, a controlled environment would have to be made. The setup of our proposed artificial environment can be seen in Figure 8.1. It consists of an aquarium that is placed on a stand containing a steering wheel located in the center. The purpose of the steering wheel is to stir the water so the particles will not sink to the bottom. The aquarium has a rectangular shape with length at least 1m and width 35cm. A light would be provided by the artificial light source from the longer side and camera from the shorter. In addition, an artificial fish could be used in the aquarium to simulate a living

animal. For this purpose, a fishing swimbait could be utilized by attaching it to a fixed wire that would be mounted on a mobile robotic arm in order to pass the same trajectory for each of the individual experiments. Moreover, it is necessary to know the exact turbidity values instead of just guessing them as we did in this master thesis. This should not be difficult in a controlled environment.

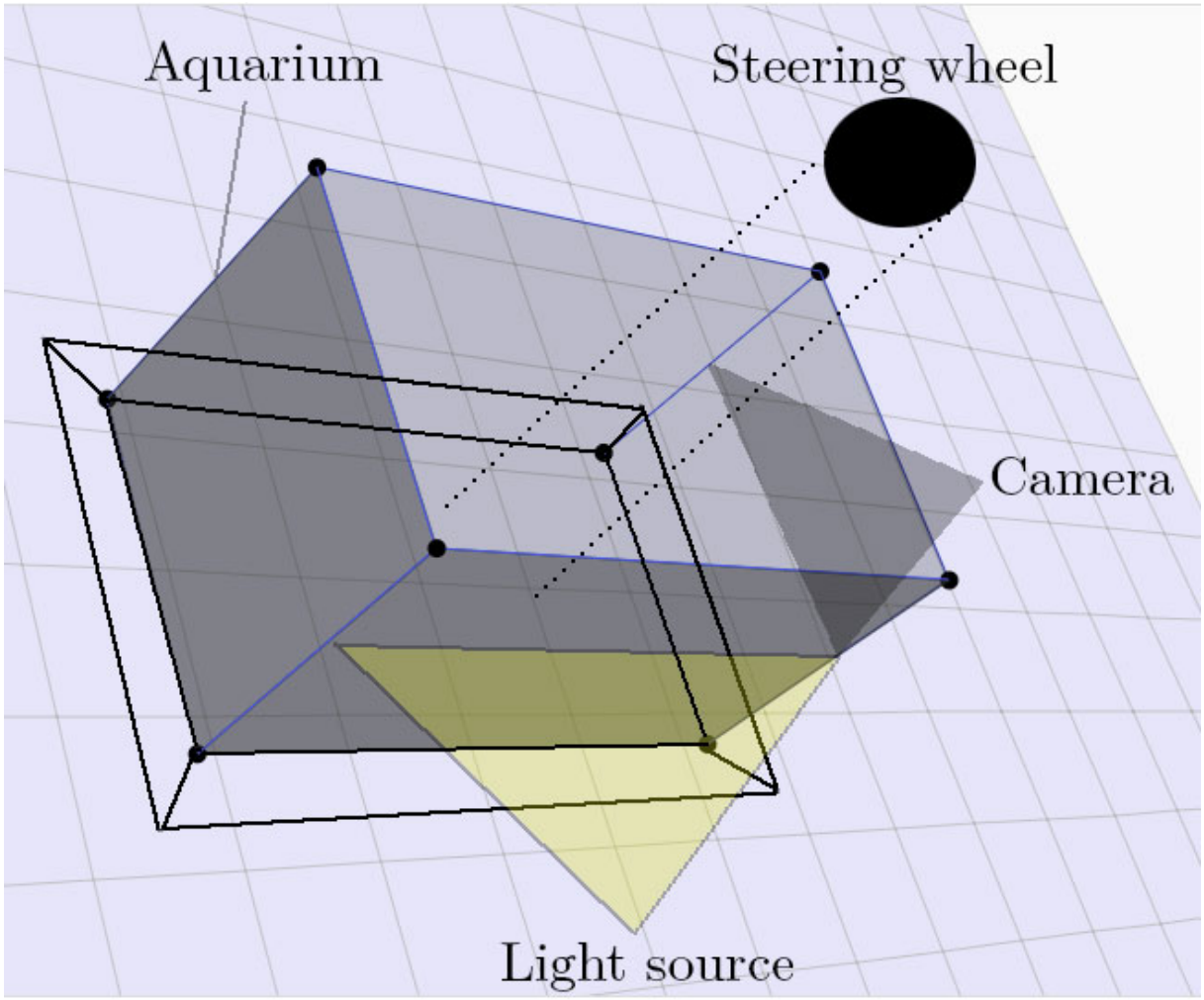


Figure 8.1: Proposed setup of the controlled environment for the future experiment.

Chapter 9

Conclusion

This master thesis aimed to improve the baseline results presented by Pedersen et al. [3] on the unique Brackish dataset. Moreover, the direction was to investigate the effect of turbidity on detection results produced by real-time object detector YOLOv3. In chapter 5 we proposed methodology that split problem into 3 modules: 1. Deep learning solution, 2. Turbidity estimation and 3. Effect of turbidity on the detection result.
Deep learning solution: Our improvements in object detection led to increase in mAP of 9.2%. The results were verified on a dataset called the Brackish X dataset which was proposed in this thesis. It consists of 3 254 frames including 14 545 manually created annotations. The dataset was evaluated with 6 different models. From the experiment, we can conclude that only the part from the new camera position can be used for evaluating the generality capabilities of deep learning models trained on the original Brackish dataset. Furthermore, there is in general lack of training data for such a complex problem which can lead to a false impression of good results.
Turbidity estimation: This module tested 5 different turbidity features that were introduced in section 2.5. We found out that sharpness and Sobel edge intensity are best for detecting turbidity for our case.
Effect of turbidity on detection result: The results of modules 1 and 2 were used as inputs for module 3 as it was indicated in chapter 5 - Methodology. Based on the analysis of the result we can conclude that it was not possible to find out what effect different turbidity levels have on deep learning object detector. A new experiment will have to be done in the future. We have proposed an artificial setup for measuring the effects of turbidity which was outlined in chapter 8.

Bibliography

[1] Millard et al. *Marine Pilot - EMODnet and INSPIRE: benefits of closer collaboration and a framework for action, Marine Strategy Framework Directive (MSFD). European Commision - DG Joint Research Centre*. 2015.

[2] Alan G. Heath. *Water Pollution and Fish Physiology. United States of America : CRC Press, 384 p. ISBN 9780262038034*. 2019.

[3] Pedersen et al. "Detection of Marine Animals in a New Underwater Dataset with Varying Visibility". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2019.

[4] Yacout MHM. *Sources of Water Pollution and Procedures Necessary for the Prevention of Pollution (A Review). Journal Of Dairy, Veterinary & Animal Research*. 2016.

[5] Q. Wang and Z. Yang. *Industrial water pollution, water environment treatment, and health risks in China. Environmental Pollution, 218, 358-365. doi: 10.1016/j.envpol.2016.07.011*. 2016.

[6] Aladesanmi et al. *Comparative Assessment and Source Identification of Heavy Metals in Selected Fishpond Water, Sediment and Fish TissuesOrgans in Osun State, Nigeria. Journal Of Health And Pollution*. 2014.

[7] Xia et al. *Aquatic Toxic Analysis by Monitoring Fish Behavior Using Computer Vision: A Recent Progress. Journal Of Toxicology*. 2018.

[8] Villon et at. *Coral Reef Fish Detection and Recognition in Underwater Videos by Supervised Machine Learning: Comparison Between Deep Learning and HOG+SVM Methods. ACIVS: Advanced Concepts for Intelligent Vision Systems, Lecce, Italy. ffhal-01374123*. 2016.

[9] Groslier et al. "Status of the Pacific Oyster Crassostrea gigas (Thunberg, 1793) in the western Limfjord, Denmark - Five years of population development". In: *Aquatic Invasions* 9 (June 2014), pp. 175–182. DOI: 10.3391/ai.2014.9.2.06.

[10] Everingham et al. *The PASCAL Visual Object Classes (VOC) Challenge*. 2009.

[11] Deng et al. "ImageNet: a Large-Scale Hierarchical Image Database". In: *IEEE Conference on Computer Vision and Pattern Recognition* (June 2009), pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[12] Lin et al. "Microsoft COCO: Common Objects in Context". In: *Computer Vision - ECCV 2014*. Springer International Publishing, 2014, pp. 740–755. ISBN: 978-3-319-10602-1.

[13] The Marine Biological Association of the UK. *The Marine Life Information Network (MarLIN)*. https://www.marlin.ac.uk/. 2020.

[14] Vallon et al. "You eat what you are: Personality-dependent filial cannibalism in a fish with paternal care". In: *Ecology and Evolution* 6 (Mar. 2016). DOI: 10.1002/ece3.1966.

[15] Ghani and Shahrizan. "Underwater image quality enhancement through integrated color model with Rayleigh distribution". In: *Applied Soft Computing* 27 (Nov. 2014), 219–230. DOI: 10.1016/j.asoc.2014.11.020.

[16] J.U. Grobbelaar. "Turbidity". In: *Encyclopedia of Inland Waters*. Dec. 2009, pp. 699–704. ISBN: 9780123706263. DOI: 10.1016/B978-012370626-3.00075-2.

[17] D. Lawler. "Turbidity, Turbidimetry, and Nephelometry. In: Reedijk, J. (Ed.) Reference Module in Chemistry, Molecular Sciences and Chemical Engineering. Waltham, MA: Elsevier. 15-July-2016 doi: 10.1016/B978-0-12-409547-2.11006-6., 13pp." In: July 2016, pp. 1–13. DOI: 10.1016/B978-0-12-409547-2.11006-6.

[18] Hussain et al. "Water turbidity sensing using a smartphone". In: *RSC Advances* 6 (Feb. 2016). DOI: 10.1039/C6RA02483A.

[19] Federal-Provincial-Territorial Committee on Health and the Environment. *Guidelines for Canadian Drinking Water Quality. Canada*. 2019.

[20] Kitchener et al. "A review of the principles of turbidity measurement". In: *Progress in Physical Geography: Earth and Environment* 41.5 (2017), pp. 620–642. DOI: 10.1177/0309133317726540.

[21] Mahony et al. "Deep Learning vs. Traditional Computer Vision". In: *Computer Vision Conference (CVC)* (Apr. 2019). DOI: 10.1007/978-3-030-17795-9_10.

[22] Moniruzzaman et al. "Deep Learning on Underwater Marine Object Detection: A Survey". In: *Advanced Concepts for Intelligent Vision Systems*. Springer International Publishing, 2017. ISBN: 978-3-319-70353-4. DOI: 10.1007/978-3-319-70353-4_13.

[23] Mahmood et al. "Coral classification with hybrid feature representations". In: Sept. 2016, pp. 519–523. DOI: 10.1109/ICIP.2016.7532411.

[24] Dai et al. "ZooplanktoNet: Deep convolutional network for zooplankton classification". In: *OCEANS 2016 - Shanghai*. 2016, pp. 1–6.

[25] Liu et al. "A review of turbidity detection based on computer vision". In: *IEEE Access* (Oct. 2018), pp. 60586–60604. DOI: 10.1109/ACCESS.2018.2875071.

[26] Mullins et al. "A novel image processing-based system for turbidity measurement in domestic and industrial waste-water". In: *Water Science and Technology* 77 (Jan. 2018). DOI: 10.2166/wst.2018.030.

[27] Leeuw and Boss. "The HydroColor App: Above Water Measurements of Remote Sensing Reflectance and Turbidity Using a Smartphone Camera". In: *Sensors* 18 (Jan. 2018), p. 256. DOI: 10.3390/s18010256.

[28] Voulodimos et al. "Deep Learning for Computer Vision: A Brief Review". In: *Computational Intelligence and Neuroscience* 2018 (Feb. 2018), pp. 1–13. DOI: 10.1155/2018/7068349.

[29] Leo et al. "Computer vision for assistive technologies". In: *Computer Vision And Image Understanding* 154 (2017), pp. 1–15. DOI: 10.1016/j.cviu.2016.09.001.

[30] Alhaija et al. "Augmented Reality Meets Computer Vision : Efficient Data Generation for Urban Driving Scenes". In: *International Journal of Computer Vision* (Aug. 2017). DOI: 10.1007/s11263-018-1070-x.

[31] R. Klette. *Concise Computer Vision - An Introduction into Theory and Algorithms*. Jan. 2014, p. 429. ISBN: 978-1-4471-6319-0. DOI: 10.1007/978-1-4471-6320-6.

[32] C. H. Chen. *Handbook of Pattern Recognition and Computer Vision*. 5th. USA: World Scientific Publishing Co., Inc., 2016. ISBN: 9789814656528.

[33] Schreiner et al. "Using Machine Learning Techniques to Reduce Data Annotation Time". In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 50 (Oct. 2006). DOI: 10.1177/154193120605002219.

[34] Bahnsen et al. *The AAU Multimodal Annotation Toolboxes: Annotating Objects in Images and Videos*. Sept. 2018. URL: https://www.researchgate.net/publication/327570312_The_AAU_Multimodal_Annotation_Toolboxes_Annotating_Objects_in_Images_and_Videos.

[35] Andreas C. Müller & Sarah Guido. *Introduction to Machine Learning with Python. United States of America : O'Reilly Media, Inc., 376 p. ISBN 9781449369415*. 2016.

[36] Nwankpa et al. "Activation Functions: Comparison of trends in Practice and Research for Deep Learning". In: *CoRR* abs/1811.03378 (2018). arXiv: 1811.03378. URL: http://arxiv.org/abs/1811.03378.

[37] Shelhamer et al. "Fully Convolutional Networks for Semantic Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39 (May 2016), pp. 1–1. DOI: 10.1109/TPAMI.2016.2572683.

[38] C. Kuo. "Understanding Convolutional Neural Networks with A Mathematical Model". In: *Journal of Visual Communication and Image Representation* 41 (Sept. 2016). DOI: 10.1016/j.jvcir.2016.11.003.

[39] Scherer et al. "Evaluation of pooling operations in convolutional architectures for object recognition". In: Jan. 2010, pp. 92–101. DOI: 10.1007/978-3-642-15825-4_10.

[40] Stanford University. *CS231n: Convolutional Neural Networks for Visual Recognition. Accessed 30 Mar 2020*. http://cs231n.github.io/transfer-learning/.

[41] Kuznetsova et al. "The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale". In: *arXiv:1811.00982* (2018).

[42] Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

[43] Joseph Redmon and Ali Farhadi. "YOLO9000: Better, Faster, Stronger". In: *arXiv preprint arXiv:1612.08242* (2016).

[44] Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement". In: *arXiv* (2018).

[45] Felzenszwalb et al. “Object Detection with Discriminatively Trained Part-Based Models”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.9 (2010), pp. 1627–1645. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2009.167.

[46] Girshick et al. “Region-Based Convolutional Networks for Accurate Object Detection and Segmentation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.1 (2016), pp. 142–158. ISSN: 1939-3539.

[47] MacQueen J. B. “Some methods for classification and analysis of multivariate observations”. In: vol. 1. 1967, 281–297.

[48] Shao et al. “Objects365: A Large-Scale, High-Quality Dataset for Object Detection”. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 8429–8438. DOI: 10.1109/ICCV.2019.00852.

[49] Agnieszka Mikołajczyk and Michał Grochowski. “Data augmentation for improving deep learning in image classification problem”. In: May 2018, pp. 117–122. DOI: 10.1109/IIPHDW.2018.8388338.

[50] Shorten & Khoshgoftaar. “A survey on Image Data Augmentation for Deep Learning”. In: *Journal of Big Data*. Vol. 6. 2019. DOI: 10.1186/s40537-019-0197-0.

[51] Wong et al. “Understanding Data Augmentation for Classification: When to Warp?” In: *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. 2016.

[52] Lemley et al. “Smart Augmentation Learning an Optimal Data Augmentation Strategy”. In: *IEEE Access* 5 (2017), pp. 5858–5869.

[53] Hiroshi I. “Data Augmentation by Pairing Samples for Images Classification”. In: *CoRR* abs/1801.02929 (2018). URL: http://arxiv.org/abs/1801.02929.

[54] Zhong et al. “Random Erasing Data Augmentation”. In: *CoRR* abs/1708.04896 (2017). arXiv: 1708.04896. URL: http://arxiv.org/abs/1708.04896.

[55] Ross B. Girshick. “Fast R-CNN”. In: *ICCV* (2015). URL: http://arxiv.org/abs/1504.08083.

[56] LeCun et al. “MNIST handwritten digit database”. In: *ATT Labs [Online]. Available: http://yann. lecun. com/exdb/mnist* 2 (2010).

Appendix A

Attachments

This chapter describes attachments in alphabetical order that were submitted together with the master thesis for the sake of reproducibility of the results achieved in the chapter 6 - Results.

- **Predictions/** - Raw predictions from YOLO for each model on the whole original Brackish dataset and Brackish X dataset.
- **The_Brackish_X_annotations/** - Manual annotations of the Brackish X dataset.
- **YOLO_config_files/** - Darknet config files of all 6 YOLOv3 models.
- brackish_x.txt - List of frames in the whole Brackish X dataset.
- brackish_x_new.txt - List of frames from the new position of the Brackish X dataset.
- brackish_x_old.txt - List of frames from the old position of the Brackish X dataset.
- Feature_importance.ipynb - Google Colab notebook containing code of computing the turbidity feature importance and estimating turbidity level using K-means.
- features_floating_device.csv - Sharpness, Sobel edge intensity, and predicted turbidity level computed for all videos in the brackish dataset with the floating device in them.
- features_reference_block.csv - Normalized turbidity features computed for all videos in the original Brackish dataset with reference block in them.
- test_high.txt - List of frames from the original Brackish test dataset with turbidity estimated to be high.
- test_low.txt - List of frames from the original Brackish test dataset with turbidity estimated to be low.
- test_mid.txt - List of frames from the original Brackish test dataset with turbidity estimated to be mid.

Appendix B

Turbidity Measurements

This chapter presents individual measurements of turbidity features extracted from the Brackish dataset from frames containing a reference box in them. Each line in the presented figures represents one video. The blue color stands for low turbidity, greed for mid turbidity, and red for high.

B.0.1 Sobel Edge intensity

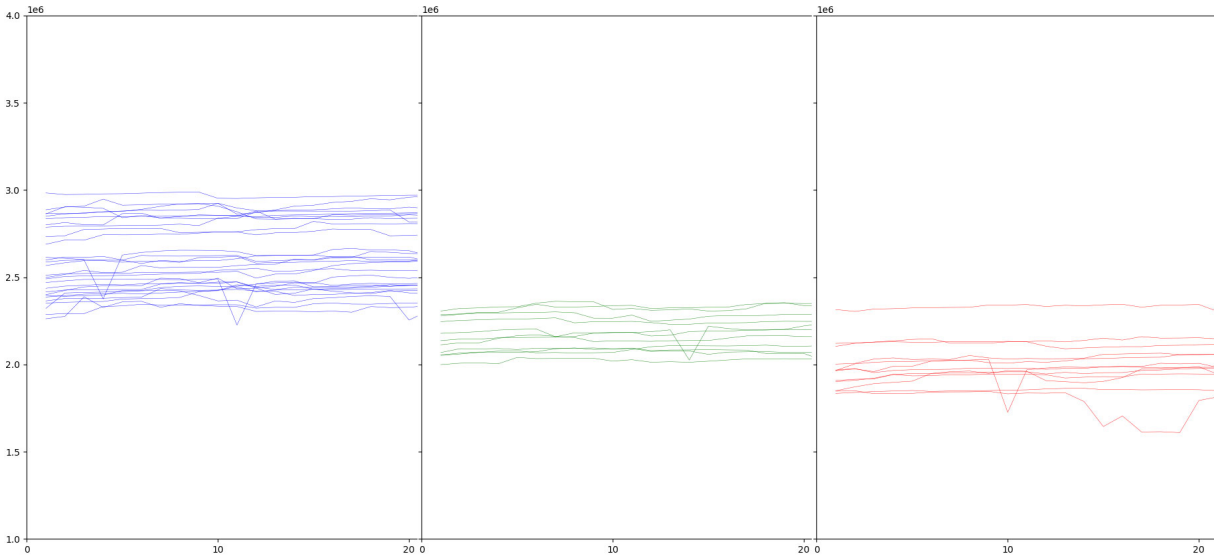


Figure B.1

B.0.2 Sharpness

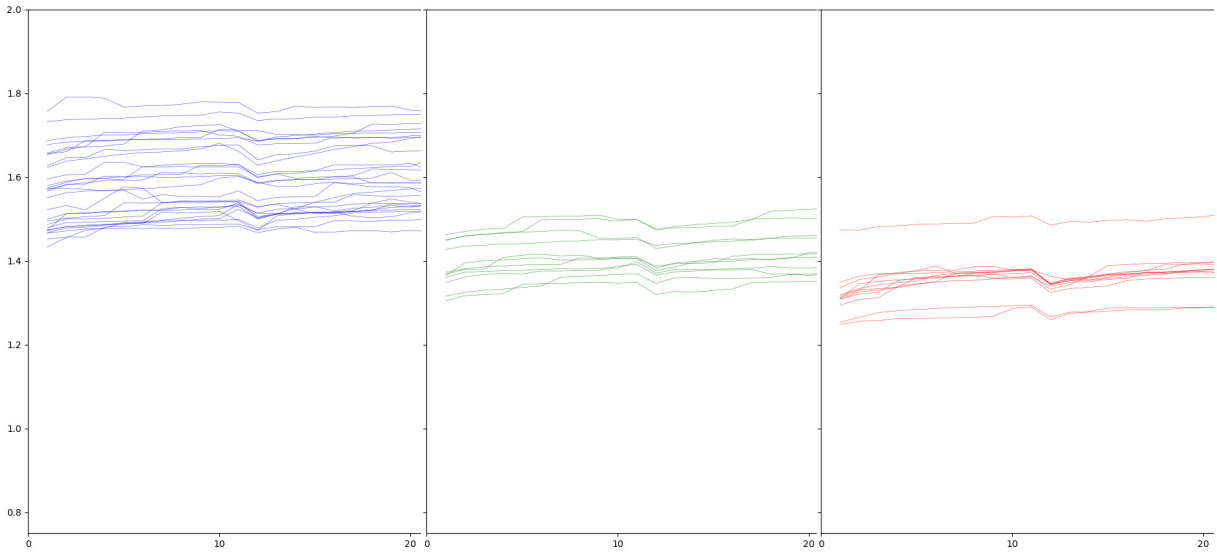


Figure B.2

B.0.3 Luminance

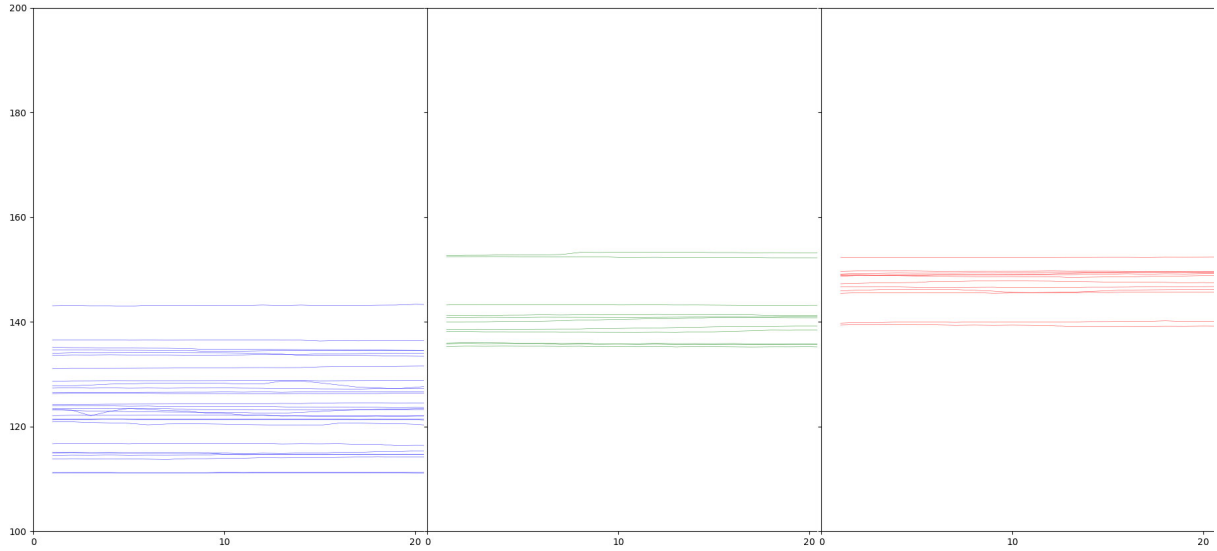


Figure B.3

B.0.4 Contrast

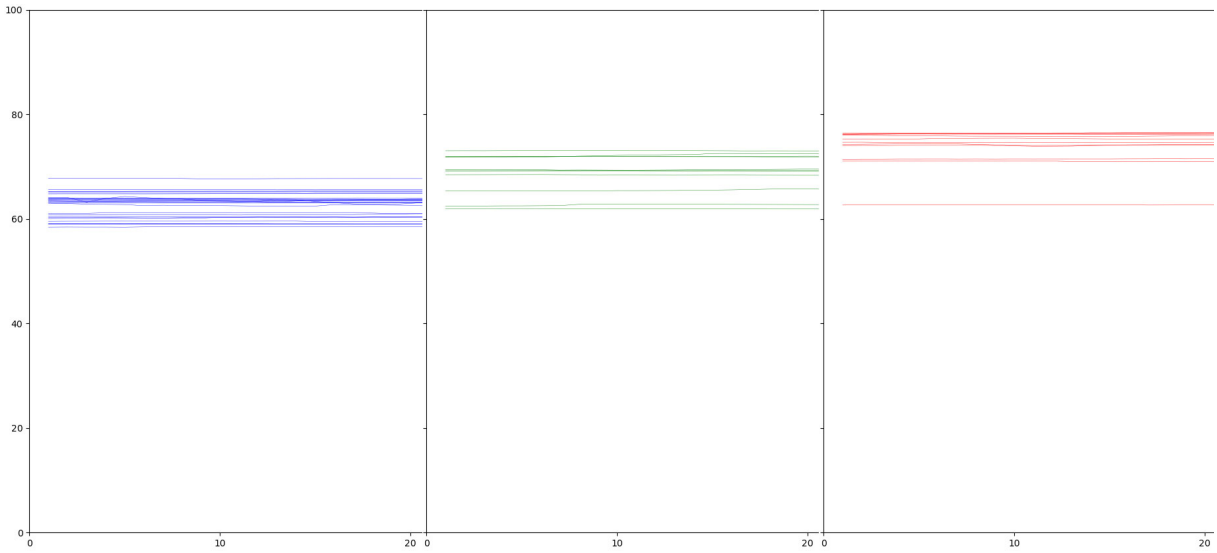


Figure B.4

B.0.5 Brightness

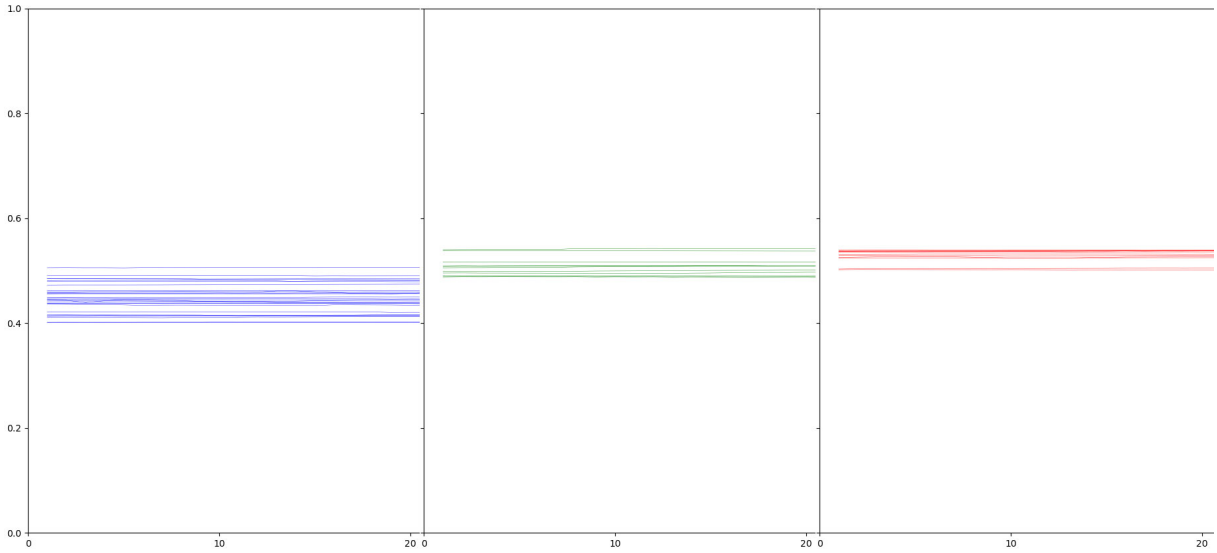


Figure B.5

Appendix C

Videos with incorrect annotations

These videos contain incorrect annotations and were use to manually check if a trained model was overfitted or not.

1. 2019-02-22_22-22-06to2019-02-22_22-22-14_1-0027 - big fish not annotated
2. 2019-03-06_22-27-01to2019-03-06_22-27-14_1-0179 - big fish not annotated
3. 2019-02-22_22-11-57to2019-02-22_22-12-10_1-0002 - starfish not annotated
4. 2019-03-19_17-01-06to2019-03-19_17-01-19_1-0029 - starfish not annotated
5. 2019-03-21_04-10-54to2019-03-21_04-11-06_1-0017 - crab not annotated
6. 2019-03-07_08-57-30to2019-03-07_08-57-40_1-0077 - big fish not annotated

Appendix D

Annotated Turbidity

Videos with the reference box that were annotated manually:

High Turbidity
2019-03-19_11-42-13to2019-03-19_11-42-21_1.avi
2019-03-20_23-30-18to2019-03-20_23-30-28_1.avi
2019-03-20_23-43-34to2019-03-20_23-43-42_1.avi
2019-03-20_23-44-38to2019-03-20_23-44-46_1.avi
2019-03-20_23-53-40to2019-03-20_23-53-56_1.avi
2019-03-20_23-54-24to2019-03-20_23-54-35_1.avi
2019-03-21_00-14-36to2019-03-21_00-14-49_1.avi
2019-03-21_00-26-58to2019-03-21_00-27-06_1.avi
2019-03-21_00-29-09to2019-03-21_00-29-17_1.avi
2019-03-21_00-58-45to2019-03-21_00-58-55_1.avi
2019-03-21_01-41-04to2019-03-21_01-41-16_1.avi
2019-03-21_03-22-18to2019-03-21_03-22-29_1.avi

Mid Turbidity
2019-03-19_11-41-37to2019-03-19_11-41-47_1.avi
2019-03-19_11-42-56to2019-03-19_11-43-14_1.avi
2019-03-19_13-33-15to2019-03-19_13-33-25_1.avi
2019-03-19_14-30-38to2019-03-19_14-30-50_1.avi
2019-03-19_17-02-30to2019-03-19_17-02-42_1.avi
2019-03-20_20-18-49to2019-03-20_20-18-57_1.avi
2019-03-21_03-26-50to2019-03-21_03-27-04_1.avi
2019-03-21_03-27-02to2019-03-21_03-27-17_1.avi
2019-03-21_04-10-54to2019-03-21_04-11-06_1.avi
2019-03-21_04-11-22to2019-03-21_04-11-31_1.avi

Low Turbidity
2019-03-19_14-02-47to2019-03-19_14-03-00_1.avi
2019-03-19_15-42-11to2019-03-19_15-42-25_1.avi

2019-03-19_17-01-06to2019-03-19_17-01-19_1.avi
2019-03-19_17-01-23to2019-03-19_17-01-32_1.avi
2019-03-19_17-02-04to2019-03-19_17-02-12_1.avi
2019-03-19_17-07-53to2019-03-19_17-08-34_1.avi
2019-03-19_18-01-56to2019-03-19_18-02-13_1.avi
2019-03-19_18-15-37to2019-03-19_18-15-45_1.avi
2019-03-19_18-26-35to2019-03-19_18-26-46_1.avi
2019-03-19_19-30-33to2019-03-19_19-30-45_1.avi
2019-03-19_19-56-16to2019-03-19_19-56-24_1.avi
2019-03-19_20-53-30to2019-03-19_20-53-39_1.avi
2019-03-19_22-56-30to2019-03-19_22-56-38_1.avi
2019-03-20_02-06-34to2019-03-20_02-06-42_1.avi
2019-03-20_03-33-35to2019-03-20_03-33-42_1.avi
2019-03-20_05-40-25to2019-03-20_05-40-33_1.avi
2019-03-20_06-00-48to2019-03-20_06-00-56_1.avi
2019-03-20_06-47-19to2019-03-20_06-47-29_1.avi
2019-03-20_10-50-55to2019-03-20_10-51-03_1.avi
2019-03-20_11-11-18to2019-03-20_11-11-25_1.avi
2019-03-20_11-52-23to2019-03-20_11-52-31_1.avi
2019-03-20_15-15-12to2019-03-20_15-15-19_1.avi
2019-03-20_20-08-39to2019-03-20_20-08-48_1.avi
2019-03-21_03-21-54to2019-03-21_03-22-09_1.avi
2019-03-21_05-01-24to2019-03-21_05-01-31_1.avi
2019-03-21_07-08-28to2019-03-21_07-08-35_1.avi
2019-03-21_07-40-40to2019-03-21_07-40-50_1.avi
2019-03-25_23-17-56to2019-03-25_23-18-04_1.avi

Videos with floating device with estimated turbidity levels based on Sobel and Sharpness:

High Turbidity
2019-02-21_06-50-24to2019-02-21_06-50-40_1.avi
2019-02-21_06-50-54to2019-02-21_06-51-01_1.avi
2019-02-21_06-52-16to2019-02-21_06-52-34_1.avi
2019-02-21_06-52-52to2019-02-21_06-53-01_1.avi
2019-02-21_06-55-09to2019-02-21_06-55-16_1.avi
2019-02-21_06-56-08to2019-02-21_06-56-28_1.avi
2019-02-22_22-11-57to2019-02-22_22-12-10_1.avi
2019-02-22_22-18-13to2019-02-22_22-18-20_1.avi
2019-02-22_22-19-45to2019-02-22_22-19-53_1.avi
2019-02-22_22-21-06to2019-02-22_22-21-16_1.avi
2019-02-22_22-21-31to2019-02-22_22-21-44_1.avi
2019-02-22_22-22-06to2019-02-22_22-22-14_1.avi
2019-02-22_22-22-16to2019-02-22_22-22-27_1.avi
2019-02-22_22-31-28to2019-02-22_22-31-38_1.avi

2019-02-22_22-32-01to2019-02-22_22-32-15_1.avi
2019-02-22_23-58-37to2019-02-22_23-58-49_1.avi
2019-03-06_22-03-36to2019-03-06_22-04-08_1.avi
2019-03-06_22-16-56to2019-03-06_22-17-10_1.avi
2019-03-06_22-27-01to2019-03-06_22-27-14_1.avi

Mid Turbidity
2019-02-20_19-01-02to2019-02-20_19-01-13_1.avi
2019-02-20_19-40-26to2019-02-20_19-40-35_1.avi
2019-02-20_19-43-37to2019-02-20_19-43-46_1.avi
2019-02-22_23-27-24to2019-02-22_23-27-39_1.avi
2019-02-22_23-28-21to2019-02-22_23-28-34_1.avi
2019-02-26_00-44-33to2019-02-26_00-44-47_1.avi
2019-03-06_22-00-43to2019-03-06_22-00-51_1.avi
2019-03-06_22-01-00to2019-03-06_22-01-09_1.avi
2019-03-06_22-01-44to2019-03-06_22-01-52_1.avi
2019-03-06_22-04-39to2019-03-06_22-04-49_1.avi
2019-03-06_22-24-20to2019-03-06_22-24-31_1.avi

Low Turbidity
2019-02-20_19-18-56to2019-02-20_19-19-06_1.avi
2019-02-20_19-19-23to2019-02-20_19-19-40_1.avi
2019-02-20_19-23-53to2019-02-20_19-24-12_1.avi
2019-02-21_06-36-31to2019-02-21_06-36-39_1.avi
2019-03-06_22-11-06to2019-03-06_22-11-17_1.avi
2019-03-06_22-12-29to2019-03-06_22-12-37_1.avi
2019-03-06_22-46-20to2019-03-06_22-46-32_1.avi
2019-03-07_08-57-30to2019-03-07_08-57-40_1.avi

Appendix E

Python Snippets

E.1 Sharpness

```
def getSharpness(image):
    image = image.astype("int32")
    """ Get the Image gradient in both directions """
    gy, gx = np.gradient(image)
    """ Calculate the magnitude of the gradient """
    gnorm = np.sqrt(gx ** 2 + gy ** 2)
    """ Average the magnitude """
    sharpness = np.average(gnorm)

    return sharpness
```

E.2 Luminance

```
def getLuminance(path):
    image = cv2.imread(path)
    b, g, r = cv2.split(image)
    b = b * 0.0722
    g = g * 0.7152
    r = r * 0.2126
    result = np.add(b,g)
    result = np.add(r,result)

    return result.sum() / 518400
```

E.3 Sobel

```
def getSobel(image):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    image = image.astype("int32")
    image = ndimage.gaussian_filter(image, 1)
    dx = ndimage.sobel(image, 0, mode="nearest")
    dy = ndimage.sobel(image, 1, mode="nearest")
    mag = np.hypot(dx, dy)
    mag *= 255.0 / np.max(mag)

    return mag
```

E.4 Brightness

Source[1]

```
def getBrightness(image):
    image = np.array(image)
    image = Image.fromarray(image)
    greyscale_image = image.convert('L')
    histogram = greyscale_image.histogram()
    pixels = sum(histogram)
    brightness = scale = len(histogram)
    for index in range(0, scale):
        ratio = histogram[index] / pixels
        brightness += ratio * (-scale + index)

    return 1 if brightness == 255 else brightness / scale
```

E.5 Contrast

```
def getContrast(image):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    return image.std()
```

[1] https://gist.github.com/kmohrf/8d4653536aaa88965a69a06b81bcb022

Appendix F

Data Augmentation and Image Synthesis

Data augmentation is a technique used for generating new training data in order to train robust models that can adapt better to different variations of detecting an object. Mikolajczyk & Grochowski [49] claims that this is one of the ways of tackling the problem of insufficient training data mentioned in the previous section. They utilized data augmentation for improving the results of the image classifier used for medical diagnosis. Shorten & Khoshgoftaar [50] divides Data augmentation techniques into 3 main categories (Figure F.1): Basic Image Manipulators, Deep Learning Approaches, and Meta-Learning.

Basic Image Manipulators This category of augmentation techniques is characterized by a relatively simple implementation. Those are for instance flipping, rotation, cropping, color jittering and edge enhancement, kernel filters, and others. When performing these transformations it is important to consider the "safety" of each of them which depends on a specific problem we want to solve. Safety can be defined as a probability that the label will be preserved after transformation [50].

Deep Leaning Approaches

The popularity of Deep Learning Approaches has recently raised and most of the research is currently focused on this category. One of the methods is Feature space augmentation. It consists of manipulating features in the feature space. Feature space is a name for high-level features that can be found in high-level layers of CNN. These features are represented by tensors that can be used for various vector level operations, for instance adding noise, interpolation, extrapolation, etc. Drawbacks of this approach are that these vector operations can be hard to comprehend. Wong et al [51] proved that augmentation in data space will in most cases outperform augmentation in feature space in terms of computation demand.

Meta Learning

Meta-learning could be explained in short as learning to learn which is a concept of applying evolutionary algorithms. One of the approaches is called *Smart Augmentation* proposed by Lemley et al [52]. It consists of using 2 neural networks: Network A - augmentation network and Network B - Classification network. Network A picks 2 images by random from

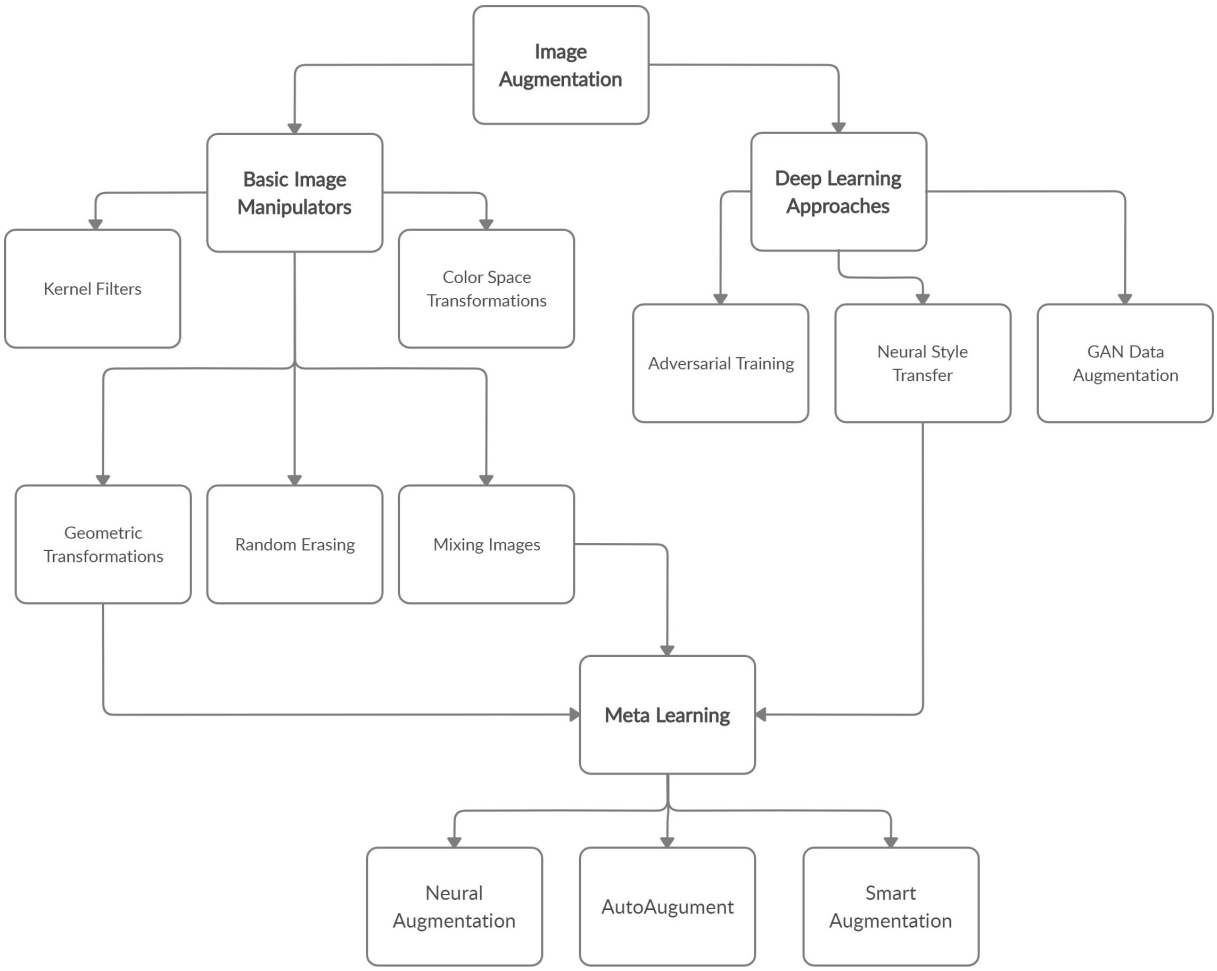


Figure F.1: 3 main categories of image augmentation techniques shown in a tree structure with subcategories.

the dataset and creates augmentation from them. These images are then used for training Network B. Eventually, the error is then backpropagated to Network A to update weights. Meta-learning is quite new and not enough research has been carried out. Moreover, it is time-consuming to implement and troublesome to understand.

F.0.1 Geometric Transformations

Flipping

Flipping one of the easiest augmentation techniques to implement and it is quite effective. It works well on the ImageNet dataset, however, it's not suitable for MNIST because for instance labels for 6 and 9 would not be preserved. Horizontal axis flipping is preferred instead of flipping the vertical axis. This makes sense because when having a picture with ground or sky, flipping picture vertically could confuse network which would have a negative influence on the results.

Rotation
The rotation consists of rotating an image by on an axis by the angle from interval 1° to 359°. Safety of this augmentation depends on the magnitude of the angle.

Translation
Image translation is an efficient way to tackle positional bias. This is done by moving the central patch to side in a random direction. If the image exceeds from its frame, then the empty area is usually colored with values 0 or 255, or filled with Gaussian noise. Positional bias can occur when dealing with datasets where are of interest is in the same place in every frame. This is the case for most face datasets.

Cropping
Cropping consists of cutting parts of the image. This can be useful when we work with the date of different widths and heights by preserving the central patch of the image. Also, it can be used by cutting random parts from images. Cropping is similar to translations, however, cropping alters dimensions of an image whereas translation will keep them.

Color Jittering
Color jittering is an augmentation method that operates in image color channels. Images are stores as tensors with dimensions height × width × color channels. This opens an opportunity to alter color channels, e.g. isolate a single color, apply single matrix operations to adjust brightness, alter histogram, increase or decrease pixel values by constant, restrict pixel values to a specific minimum or maximum values. This augmentation is suitable for handling lighting biases. However, it is important before applying this

F.0.2 Kernel Filters

The principle of applying kernels is similar to using a feature detector in a convolutional layer. A kernel is a matrix of size $n \times n$ which slides on the image and adjusts its pixel values. The most used filters are Gaussian blur or High contrast filter. Gaussian blur can be applied to blur images. The network might be better in terms of handling blurred images. Contrast filter, on the other hand, can make edges sharper which can lead to more features for a network to learn.

F.0.3 Mixing Images

Mixing images consists of taking image pairs and averaging their pixel values. Horoshi [53] describes a method called *SamplePairing* where for each image of the dataset he picks another image by random. Performs basic data augmentation on both of them. Then crops random patch of size 224 × 224 from both of the images. In the next step, both images are merged by averaging intensity of two patches for each pixel. According to Shorten and Khoshgoftaar [50] this is counter-intuitive because to a human, products of these transformations are not going look very useful. Utilizing *SamplePairing* has helped to reduce the error rate on CIFAR-100 dataset by 8.58 %.

F.0.4 Random Erasing

Random erasing introduced by Zhong et al [54] consists of removing random patches of size $n \times m$ from images and replacing them by rectangles with random pixels intensity from interval 0 to 255. This idea corresponds to the dropout layer where a certain percentage of the data is discarded. The dropout layer is ingrained within network architecture whereas random erasing is done in the input data space. This will influence the network to focus more on the entire image and find other descriptive features rather than some specific place. Moreover, it is a great way of tackling problems such as overfitting and occlusion. It improved the performance of Fast R-CNN [55] from 70.0 mAP to 76.2 mAP.

Random erasing is a relatively safe augmentation technique, however, there might be cases where random erasing is not suitable. For example MNIST dataset [56] of handwritten digits. When Erasing random patches, it can occur that number 8 will become 6.