DEEP LEARNING APPROACHES TO CLASSIFY AND TRACK AT-RISK FISH SPECIES

by

Vishnu Vardhan Kandimalla

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
August 2021

The thesis is dedicated to my family whose love always shows me the way.

Table of Contents

List of Tables

List of Figures

Abstract

In marine ecosystems, fish play a crucial role. They are linked to other organisms through the food web and other processes. Because they provide food, humans have had an especially close relationship with them for decades. Fish requires a healthy living environment to survive and grow. Due to poor living conditions, the numbers of several large fish species have decreased. Scientists have long recognized the importance of sufficient fish habitat in maintaining a healthy fish population. As a result, marine biologists and conservationists must detect and track fish species in the real world frequently to determine relative abundance and track population changes. For this purpose, one common method of doing so is to use acoustic transmitters which are surgically implanted in fish that transmit a unique id and sensor data. Energy companies are legislated to not harm species at risk around their fixed infrastructures. A species at risk cannot be tagged using conventional fish tracking technology. Therefore, without harming the species and obtaining the required information, the alternative idea is to use sonars, cameras, etc to collect the data and use deep learning algorithms to analyze the data for fish species detection, classification, and tracking. Accordingly, two deep learning models called YOLOv3 and Mask-RCNN were applied to acoustic images. Even different augmentation techniques such as hue, saturation, and random rotation were applied to achieve 0.73 mean average precision(mAp) using YOLOv3, and about 0.62 mean average precision(mAp) using Mask-RCNN at the intersection over union (IOU) 0.4. These results helped in understanding that deep learning models can be applied along with different augmentation techniques to achieve the best results on acoustic data. For tracking of fish species, YOLOv4 along with the integration of the Norfair tracking algorithm was tested on the Wells dam dataset. The model was able to achieve the maximum Multiple objects tracking accuracy (MOTA) of about 66.9% on 20fps videos. Our findings show that deep learning models can replace human effort in watching hundreds of thousands of videos for fish species detection and classification, and that tracking algorithms and video cameras can also replace fish tagging in some situations.

Acknowledgements

I would like to express my gratitude to my supervisor Dr.Luis Torgo who has been in support with my thesis. I would also like to express my gratitude to Dr.Christopher Whidden, who helped me with his insightful information. Both have been a strong support and great sense of hope for my thesis. Their guidance throughout my thesis helped me to solve critical problems and to shape my thesis. I wish to thank DeepSense, Innovasea and Mitacs in funding my thesis and providing me great support. I also wish to thank my family in all support.

Chapter 1

Introduction

The main goal of this thesis is to see how well deep learning models perform in the detection, classification, and tracking of fish species on acoustic and video data. The importance of fish detection, classification, and tracking is discussed in the Section 1.1 of this chapter; the Section 1.2 addresses the various methods used for fish detection, classification, and tracking; and the third Section 1.3 discusses the thesis contribution and organisation.

1.1 Importance of Fish Detection, Classification and Tracking

Fish are an important part of marine ecosystems. They are intricately linked to other organisms via the food web and other mechanisms. Since they provide food for humans, mankind has had a particularly close relationship with them for centuries. Around 43 million people [2] around the world depend on fishing or fish farming for a living. On the other hand, humans are not always kind to this natural resource. Besides, the ocean is polluted by waste from industry, cities, agriculture, and other sources. Development has a direct impact on many habitats, such as mangrove forests [2]. Fish, like all animals, need a healthy living environment, or habitat, to survive, grow, and reproduce. Temperature, depth of water, currents, waves, bottom kinds, cover, and other physical and chemical effects in a fish's environment, as well as oxygen levels, dissolved minerals, and other substances, make up a fish's habitat. In areas where fish populations have been changed or damaged by humans, many large fish species have reduced in numbers, become extinct, or been replaced by other species that are more tolerant of environmental changes [1]. Anglers and scientists have long recognized the importance of sufficient fish habitat in maintaining a healthy fish population [1]. As a result, marine biologists and environmentalists frequently count and detect fish species using techniques such as fish tagging, catch-and-release fishing, and video and image analysis to determine relative abundance and track population changes in their habitats. As a consequence, fish detection, classification and tracking are considered a crucial problems that must be tackled.

1.2 Methods Used For Fish Detection, Classification and Tracking

Fishermen and researchers use fish tagging as one of the most popular methods for monitoring the growth of different fish. This is done to gain a greater understanding of marine life's lifespan and migratory patterns [13]. For decades, a variety of marine and freshwater animals have been tagged externally with electronic tags. It was the most common form of attachment in the early years of fish telemetry studies, but later internal implants became the preferred method. With the introduction of archival tags, pop-up satellite archival tags (PSATs), and other environment-sensing tags, the number of telemetry studies using external tagging has increased recently. The external attachment has advantages over other tagging methods, such as speed of implementation, and it may be the only choice for fish with body shapes that make surgical implantation impossible, or when using tags with sensors that record the external environment. Tissue injury, premature tag loss, and decreased swimming ability are the most recorded problems with external tags, but the effects are highly context-dependent and species-specific. External tagging has been linked to reduced growth and longevity, but direct mortality from external tagging appears to be uncommon [37].

These techniques such as external tagging with electronic tags, surgical-implan tation of transmitters are not permitted to be used because they damage fish species. Sonars, such as the Dual-frequency Identification Sonar (DIDSON) and Adaptive Resolution Imaging Sonar (ARIS), have lately emerged as a feasible alternative to tagging for monitoring fish behaviour. When the fish passes a sensor, the camera is turned on and a video is recorded. These videos are often manually classified into fish species by humans [61] as there are not automated methods available to do it. The number and distribution of different fish species can provide useful information about the ecological system's health and can be used as a parameter for tracking environmental changes [67]. Visual classification of fishes can also aid in tracking their movements and revealing patterns and trends in their behaviour, allowing for a more in-depth understanding of the species [67]. Identifying, classifying, and counting fish individuals on photographs and videos is a critical task for monitoring marine biodiversity at a low cost. However, it remains difficult and time-consuming [89], as well as error-prone, and requires a trained expert because it is unrealistic to thoroughly analyse all the details in the collected videos [80], due to numerous challenges, such as environmental variations in luminosity, fish camouflage, complex backgrounds, water murkiness, low resolution, shape deformations of swimming fish, and subtle variations between some fish species and so on [35]. To overcome these challenges, we applied deep neural network algorithms to video data obtained using cameras and sonars to create automated fish detection and classification techniques. Ocean-aware project [8], funded by Canada's Ocean Supercluster, plans to build, and commercialize world-class solutions for tracking fish health and innovative approaches to assessment. This Ocean-aware project is led by Innovasea, with Dalhousie University as a supporting partner. The main aim of this project is to get a clearer understanding of the nature and movement of fish in real-time so that facilities and mitigations can be designed to preserve fish habitat. All Innovasea's fish tracking technologies currently rely on the use of a transmitter tag on the fish being tracked. A biologist normally surgically implants this transmitter tag into the fish. There are several cases where tagging fish is impossible, illegal, or prohibitively expensive, and thus tagging fish is a technical barrier to fish tracking. Fish species on the endangered species list, for example, cannot be legally tagged because they are protected from human harm. As a result, this limitation complicates and restricts scientific research of endangered fish species, as well as regulatory monitoring of endangered fish species. As part of this Ocean-aware project, our research focuses on acoustic and video camera data processing techniques that aid in the detection, classification, and tracking of fish. The main contribution to this project is testing and validating two deep learning models, YOLOv3 [70] and Mask-RCNN [29], to see how feasible they are in performing fish detection and classification, and if so, optimising them for better results, as well as using Norfair [6] algorithm in combination with YOLOv4 [14] to see how well it performs in fish tracking. YOLOv4 [14] is used here, which is an enhanced version of YOLOv3 and the most recent version of YOLO available at the time of the tracking task.

1.3 Thesis Contribution and Organisation

This thesis contributions are summarised below.

- Converted acoustic data to video data using sound metrics software(Didson V5) and later extracted images from the video data.
- Designed a workflow to label the images with the help of the Innovasea team so that images are suitable as an input for deep learning models.
- Used two deep learning models, YOLOv3 and Mask-RCNN, for performing fish classification and detection and showed that deep learning models can detect and classify fish from visual acoustic data.
- Various augmentation techniques were needed to improve the results of both deep learning models, YOLOv3 and Mask-RCNN.
- Analysed and compared the results of both the YOLOv3 and Mask-RCNN deep learning models to determine YOLOv3 is the better deep learning to use on acoustic data.
- The Norfair tracking algorithm was integrated with YOLOv4 to track fish in videos.
- Different parameters in the Norfair algorithm were tweaked to achieve the best object tracking accuracy.

The following is how the remainder of the thesis is structured; Chapter 2 describes the background and related work. Chapter 3 discusses the datasets and methods used in solving the problem. Chapter 4 explains the experimental methodology and results obtained and Chapter 5 discusses the conclusion, limitations, and future work.

Chapter 2

Background and Related Work

2.1 Background

This Section 2.1 introduces topics that we used in this thesis, making it easier for the reader to understand the approach that we took to solve the problem. In this subsection 2.1.1, we look at what computer vision is and some of its subtopics. In subsection 2.1.2 we discuss about what is deep learning and some of its subtopics. Subsection 2.1.3 describes the metrics we use in evaluating object detection and classification, and subsection 2.1.4 discusses the metrics we used in evaluating object tracking.

2.1.1 Computer Vision

Computer Vision is a field of research that focuses on developing techniques that assist computers in seeing and understanding the content of digital media such as images and videos, as well as identifying and processing objects in images and videos in the same way that humans do [55]. There are several steps that help machines to understand and feel their surroundings in the context of computer vision. Image classification, object detection, image segmentation, 3D scene reconstruction, video and image background indexing, scene reconstruction, image restoration, and so on are some of them [54].

We'll go through in detail about image classification, object detection, and image segmentation since these are the topics we used to solve the problem in our thesis.

Image Classification

The labeling of a pixel or a group of pixels based on their grey value is called image classification. Multiple features are typically used for a collection of pixels in classification, implying that several images of a particular object are needed [18]. Digital image classification aims to classify each pixel using spectral information expressed by digital numbers in one or more spectral bands and tries to identify each pixel using this spectral data. The word spectral pattern recognition describes this form of classification [59]. This is one of the techniques used in solving computer vision applications. Convolutional neural networks, AlexNet [41], GoogLeNet [83] , and VGGNet [79] are some of the essential architectures used for image classification. Figure 2.1 shows an example of image classification.

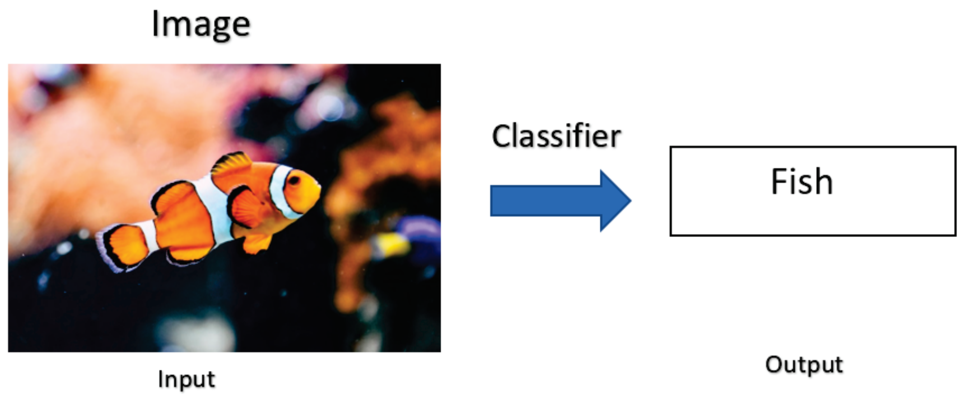


Figure 2.1: Example of a image classification

Image classification can be done in two ways: supervised or unsupervised classification.

Supervised Image Classification

The idea behind supervised classification is that a user can select sample pixels in an image that are representative of specific classes and then instruct image processing software to use these training sites as references for the classification of all other pixels in the image. The computer is used to identify spectrally similar areas for each class by using numerical information in all spectral bands for the pixels that make up these areas. To determine the numerical ”signatures” for each training class, the computer utilizes a special program or algorithm. After the machine has calculated the signatures for each class, each pixel in the picture is compared to these signatures and digitally labeled with the class that it most closely resembles. In supervised classification, we classify the information classes first, then use those to decide the spectral classes that describe them [59].

Unsupervised Classification

Unsupervised classification is a type of classification in which the results, which are groupings of pixels with similar characteristics, are based on software analysis of an image without the user providing sample classes. The analyst groups the spectral classes first, based solely on the numerical details in the data, and then match them to information classes. Clustering algorithms are programs that are used to determine the natural groupings or structures in data. Typically, the analyst specifies the number of groups or clusters to look for in the data. In addition to the number of classes desired, the analyst may also specify parameters relating to the separation distance between clusters and the variation within each cluster This iterative clustering process may yield some clusters that the analyst will want to combine later, or clusters that should be broken down further, each of which will necessitate another application of the clustering algorithm. As a result, the unsupervised classification does eliminate the need for human intervention [59].

Object Detection

Object detection is a computer vision technique that involves using a bounding box to locate one or more objects in an image and predicting the object's class. Even before CNN's became common in computer vision, object detection was being studied. Although CNN's are capable of extracting more complicated and better features automatically, a look at traditional methods can serve as a minor diversion at worst and inspiration at best [65]. Single Shot Detector(SSD) [50], Faster-RCNN [72], Mask-RCNN [29], YOLOv3 [70], and others are some of the examples of object detection algorithms. The Figure 2.2 shows an example of object detection, with an image containing fish as input to the object detection algorithm, and we can see a bounding box drawn on fish, as well as the label(fish) on it.

Different forms of object detection are discussed as follows, including one-step object detection and two-step object detection as we have used both types of object detection models in our thesis.

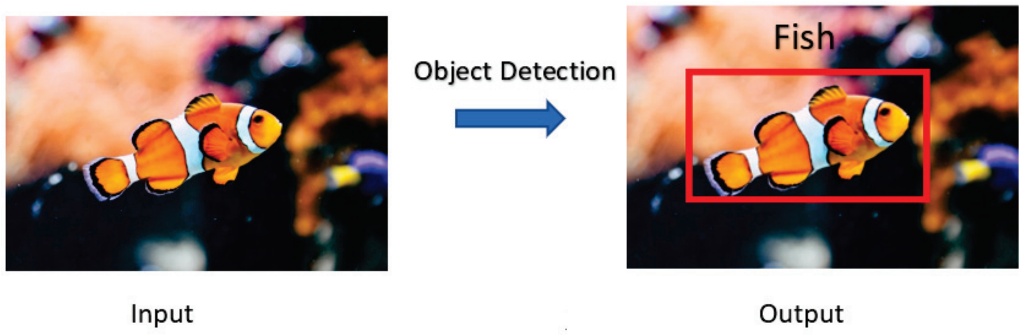


Figure 2.2: Example of object detection

One-Step Object Detection

One-step object detector treats object detection as a simple classification problem by taking an input image and learning the class probabilities and bounding box coordinates. Many one-step object detection architectures have been proposed in response to the need for real-time object detection, such as YOLO [68], YOLOv2 [69], YOLOv3 [70], SSD(Single Shot MultiBox Detector) [50] and others, which attempt to combine the detection and classification steps. When each bounding box can be easily expressed with a few values, it's much simpler to combine the detection and classification steps, resulting in a significantly faster pipeline [65]. Overview of one-step object detector is shown in Figure 2.3.

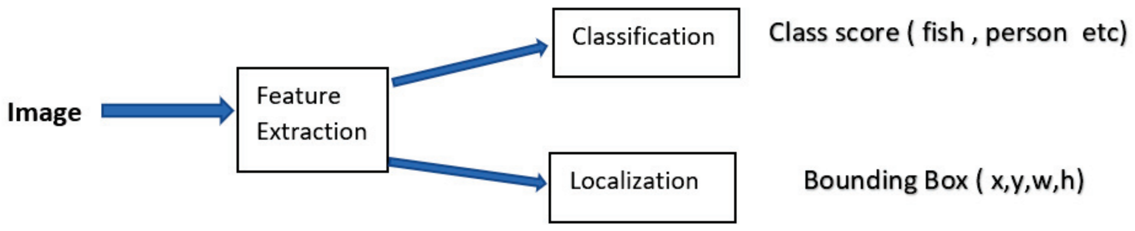


Figure 2.3: Overview of one-step object detection [43]

Two-Step Object Detection

Two-Step Object Detection uses algorithms to distinguish bounding boxes that might contain objects and then classify each bounding separately. Detectors such as Faster-RCNN (Region-based convolutional neural networks) and Mask-RCNN are two examples of RCNN's that use a region proposal network which is a convolutional network devoted to detect regions in the image where objects may be found to produce regions of interest in the first stage and then send those regions down the pipeline for object classification and bounding-box regression in the second stage [65]. Overview of two-stage objects detection is shown in Figure 2.4.

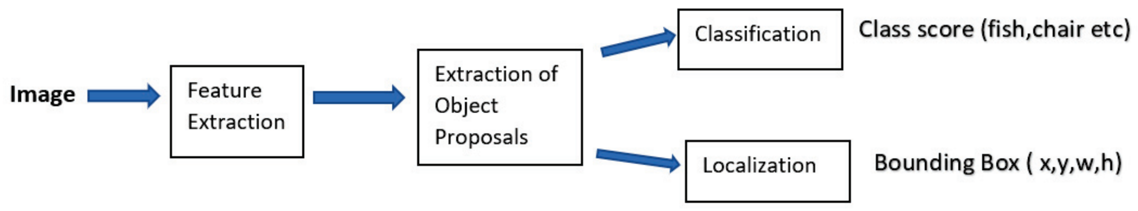


Figure 2.4: Overview of two-step object detection [43]

Image Segmentation

Image segmentation is a computer vision technique that produces pixel-by-pixel masks for each object in an image. This allows us to consider the object in the image at a much better level. The semantic segmentation technique and the instance segmentation technique are the two forms of this technique [77].

1. In semantic segmentation every pixel belongs to a specific class, and all pixels in the same class are characterized by the same color which is shown in Figure 2.5.

2. In instance segmentation different objects of the same class have different colours which is shown in Figure 2.6. .

Object Tracking

Object tracking is the process of estimating the state of a target object in a scene based on previous data. To put it another way, given a video, we want to figure out

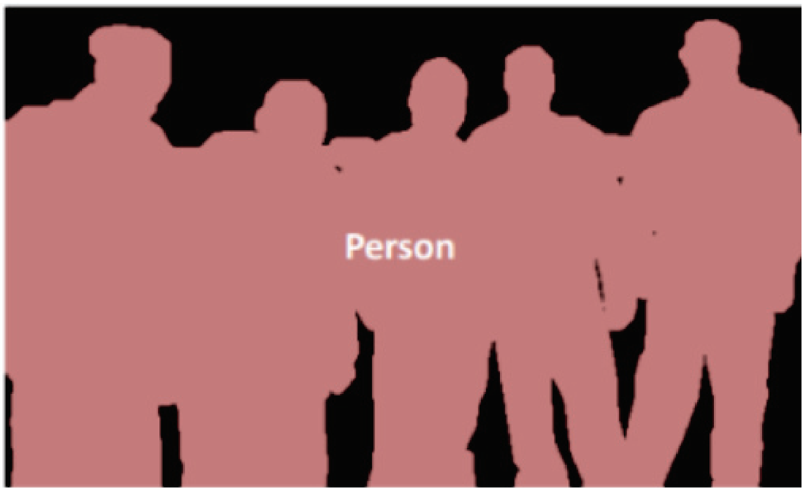


Figure 2.5: Example of semantic segmentation [77]

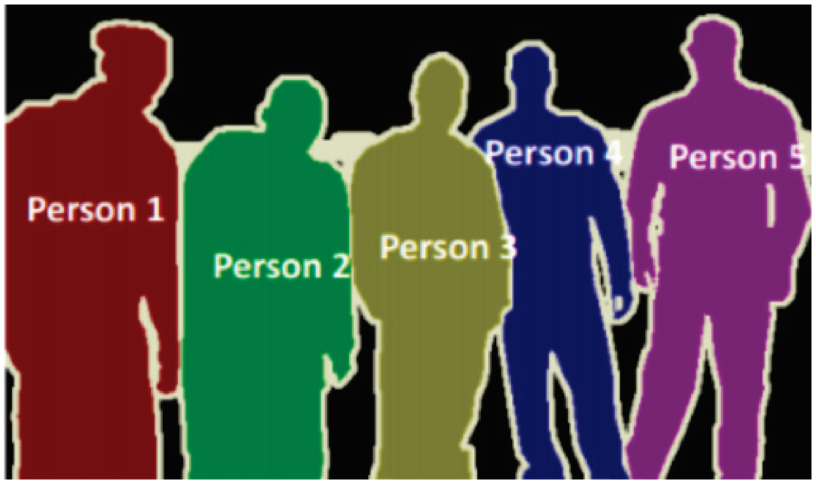


Figure 2.6: Example of instance segmentation [77]

the parts of the image that represent the same object in different frames first [24]. There are two different types of object tracking:

1. Single object tracking (SOT)
2. Multiple object tracking (MOT)

Single Object Tracking

The tracker is given the target's bounding box in the first frame of single object tracking(SOT). The tracker's goal is to find the same target in all of the subsequent frames. Since the first bounding box is manually provided to the tracker, SOT falls into the category of detection-free tracking. This means that single object trackers should be able to track any object they are given, even if no classification model has been trained on it [24].

Multiple Object Tracking

Multiple object tracking(MOT), as the name implies, involves tracking multiple objects. The tracking algorithm should first decide the number of objects in each frame and then keep track of each object's identity from frame to frame. MOT is a difficult problem to solve as identity switches are difficult to avoid, particularly in crowded videos, and the existence and number of objects in each frame are unknown, so MOT algorithms depend heavily on detection algorithms [19]. We are interested in tracking multiple objects in this thesis.

2.1.2 Deep Learning

Deep learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks. It's a method for extracting features and tasks from data like images, text, and sound. Deep learning models are also referred to as deep neural networks because most deep learning approaches use neural network architectures. The term "deep" usually refers to the number of hidden layers in the neural network. Convolutional neural networks are one of the most popular deep neural networks, and they're particularly good at working with image data.

Neural Networks

Neural networks are a set of algorithms that attempt to recognize patterns, correlations, and information from data using a process inspired by and working in the same way as the human brain/biology. Neural networks take in data and train themselves to recognize patterns in the data so that they can anticipate the outcomes of a new set of similar data [22]. Three components make up a basic neural network as shown in Figure 2.7

Input Layer Also known as input nodes are the inputs/information from the outside world that the model uses to learn and draw conclusions. The information is passed on to the next layer, the hidden layer, using input nodes [22].

Hidden Layer The hidden layer is a collection of neurons that execute all computations on the input data. A neural network can have any number of hidden layers. A single hidden layer makes up the simplest network [22].

Output Layer The output layer contains the model's output/conclusions produced from all calculations. The output layer might have a single or several nodes. The output node in a binary classification problem is 1, while in a multi-class classification problem, the output nodes can be more than 1 [22].

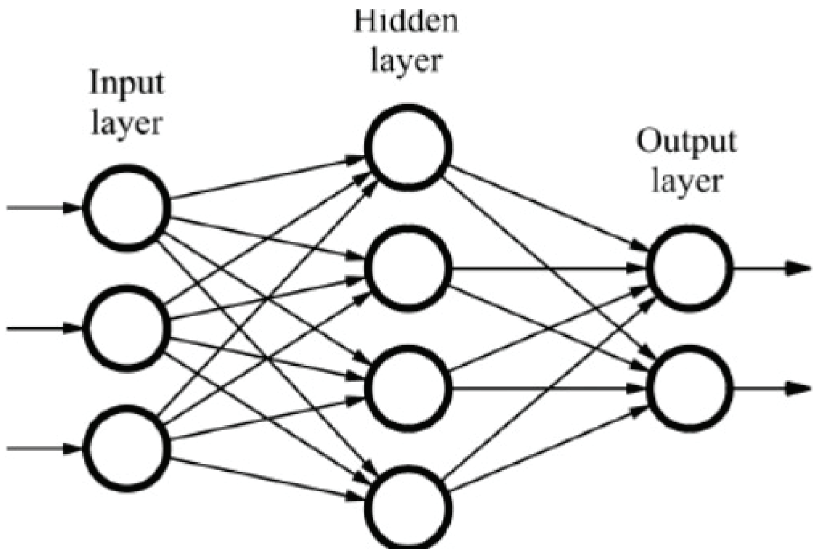


Figure 2.7: Example of Neural network [21]

Acivation Functions

In a neural network, activation functions compute the weighted total of inputs and biases, which is then used to determine whether a neuron can be activated or not. It manipulates the input and generates an output for the neural network that includes the data's parameters.

ReLU is the rectified linear activation function, or ReLU for short is a piecewise linear function that, if the input is positive, outputs the input directly; else, it outputs zero. Because a model that utilizes ReLU is quicker to train and generally produces higher performance, it has become the default activation function for many types of neural networks [36]. Mathematically, it is defined as $y = max(0, x)$. Figure 2.8 shows overview of ReLU activation function.

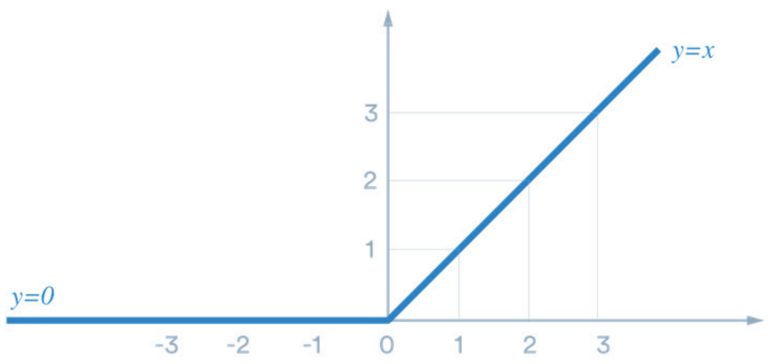


Figure 2.8: ReLU activation function
[20]

Leaky ReLU is the improvement of ReLU function. The dying ReLU condition occurs when the ReLU function kills some neurons in each repetition. Instead of returning 0 for negative values, a leaky ReLU will compute output using a relatively small component of input, x = 0.01 (non-zero constant gradient), and hence will never kill any neuron [76]. The equation of leaky ReLU is as follows and Figure 2.9 shows the overview of leaky ReLU function.

$$f(x) = \begin{cases} x & ifx > 0 \\ 0.01x & otherwise \end{cases}$$

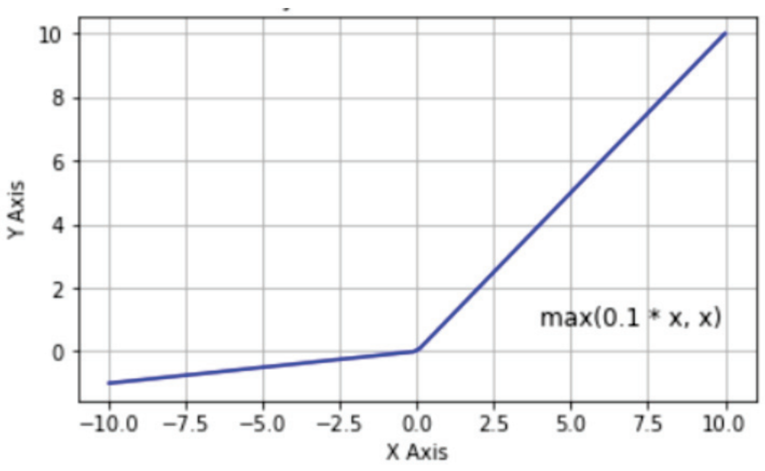


Figure 2.9: Leaky ReLU activation function
[76]

Convolutional Neural Network

A convolutional neural network, or CNN, is a form of artificial neural network that has been widely used for image analysis. Although CNN's are most commonly used for image analysis, they may also be used for other data analysis or classification problems. CNN is most commonly thought of as an artificial neural network with some kind of specialisation for detecting and understanding patterns. Some of the accomplishments achieved with CNN include classifying handwritten digits using the MNIST [44] dataset and identifying images using the CIFAR-10 dataset [40]. CNN's ability to detect patterns is what makes it so effective for image analysis. A convolutional layer receives input and then transforms the input in some way and then outputs the transformed input to the next layer and with a convolutional layer this transformation is known as convolutional operation. Figure 2.10 is an example how CNN looks like.

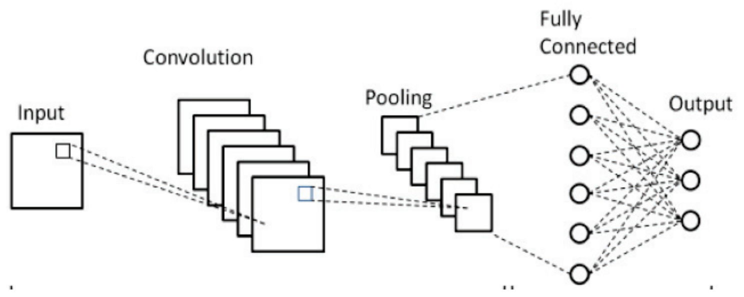


Figure 2.10: Example of convolutional neural network [62]

The pooling layer is a downsampling operation used after a convolution layer to achieve spatial invariance. Max and average pooling, in particular, are special types of pooling that take the maximum and average value, respectively. The number of pixels that move over the input matrix is known as the stride for a pooling layer. When the stride is set to 1, the filters are moved one pixel at a time. We switch the filters two pixels at a time when the stride is two, and so on. Convolution will function with a stride of 2 as seen in Figure 2.11. As the stride is 2 the filter is moved across the image from top to bottom, left to right, with two-pixel column changes on horizontal movements and two-pixel row changes on vertical moves. The output size is calculated using the formula $(W - F + 2P)/S + 1$ where W are the input size, F is the filter size, P stands for padding and S is the stride size. As we see in Figure 2.11

the input size is 9, with a filter size of 3, padding of 0, and stride size of 2, the final output size is (9-3+2(0))/2+1 =4(4 × 4 matrix) and each dot in the 4 × 4 matrix represents the max value of each filter on the input matrix. Since the filter does not always perfectly match the input image, we use zero padding. The method of adding P zeroes to each side of the input's boundaries is referred to as zero-padding. This value can be set manually or automatically [64].

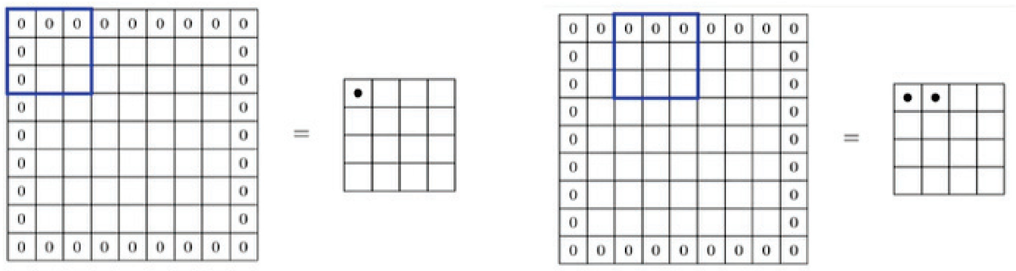


Figure 2.11: Example of strides of 2 pixels [78]

DenseNet

DenseNet (Densely connected convolutional networks) [33] are built on a straightforward connectivity pattern in which each network layer is directly connected to every other one. A dense block and a transition layer are present at each stage of a DenseNet, and each dense block is made up of k dense layers dense block is a collection of layers that are all related to each other. A batch normalization, ReLU activation function, and a 3×3 convolutional layer are included in every single layer in the dense block. A batch normalization, 1×1 convolutional layer, and an average pooling layer make up a transition layer. The output of ith dense layer will be concatenated with the input of the ith dense layer, and the result will be the $(i + 1)$th dense layer's input [25]. Each layer takes inputs from all preceding levels and passes on its feature maps to all following layers to maintain the feed-forward nature. The DenseNet is depicted in Figure 2.12.

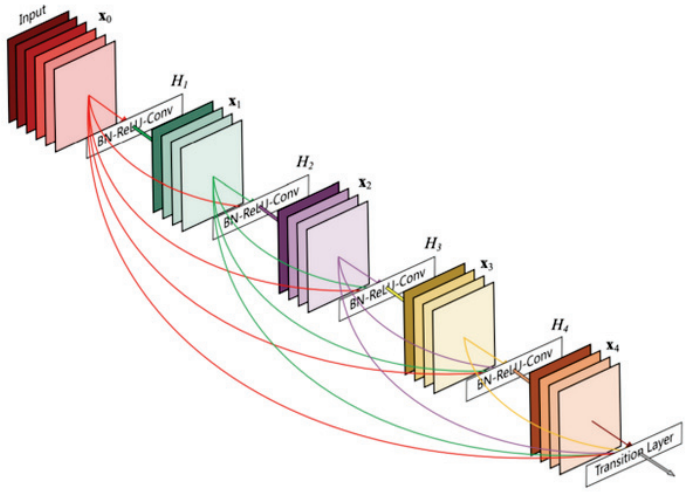


Figure 2.12: Overview of DenseNet architecture [33]

2.1.3 Evaluation Metrics of Object Detection and Classification

In this thesis, various metrics are used to assess the efficiency of the YOLOv3 and Mask-RCNN models in terms of object detection and classification. A brief description of each metric is as follows.

Confidence Score

confidence score for each prediction box measures the confidence on both the classification and the localization.. A classifier is typically used to predict it. For instance, in the example of Figure 2.13, a classifier is 91 percent certain that the object is present in the bounding box and the object present is fish.

Intersection Over Union

The area of the intersection divided by the area of the union of a predicted bounding box Bp and a ground-truth box Bgt is known as intersection over union (IOU). The ground truth box is the one that we labeled with the annotation tool, and the predicted bounding box is the one that the detector draws on the object in the image. The overview of IOU is shown in Figure 2.14.

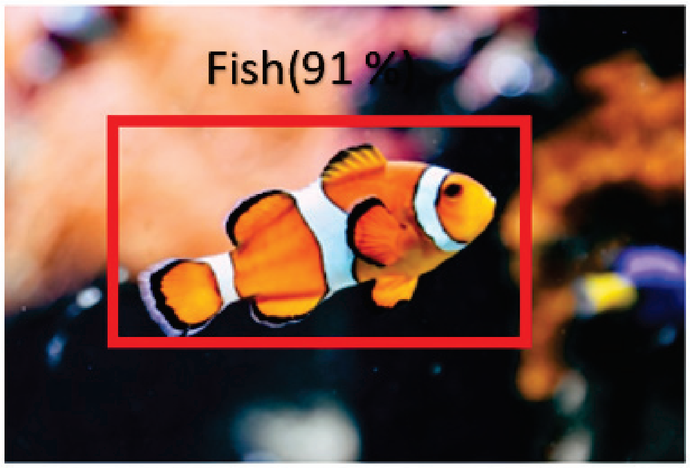


Figure 2.13: Example of confidence score

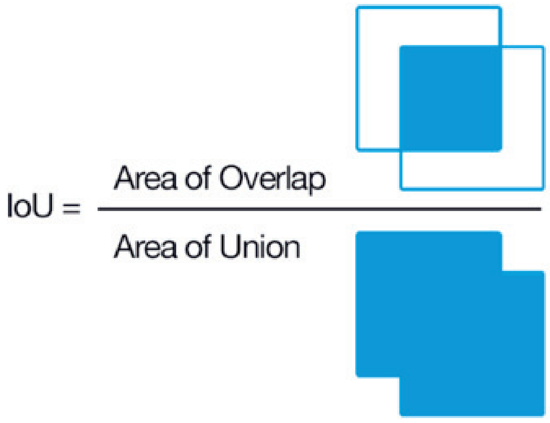


Figure 2.14: Overview of intersection over union [26]

True Positive(TP)

The detection made by the detector is said to be true positive(TP) if the predicted bounding box has IOU greater than or equal to a given object detection threshold, which is frequently selected as 0.5 with the labeled ground truth box.

False Positive(FP)

The detection made by the detector is said to be false positive(FP) if the predicted bounding box has IOU less than the given object detection threshold, which is frequently less than 0.5 with the labeled ground truth box.

False Negative(FN)

The detection is said to be false negative if there is no bounding box predicted by the detector although there is a ground truth box available.

Precision

Precision is the number of true positives divided by the sum of true positives and false positives.

$$Precision = \frac{TP}{TP + FP} \tag{2.1}$$

Recall

Recall is the number of true positives divided by the sum of true positives and false negatives

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

Precission-Recall Curve

The precision-recall curve is a graph that depicts the trade-off between precision and recall for various object detection thresholds [27]. This graph helps in determining the best threshold for maximizing both metrics. Precision values are plotted on the y-axis, while recall values are plotted on the x-axis. Plotting a curve for each object class is a reasonable way to test the output of an object detector since the confidence is changed [66]. An example of a precision-recall curve is shown in Figure 2.15.

Average Precision(AP)

The average precision (AP) is a way to summarise the precision-recall curve into a single value representing the average of all precisions. The AP is calculated according to the equation 2.3. Using a loop that goes through all precisions/recalls, the difference between the current and next recalls is calculated and then multiplied by the current precision. In other words, the AP is the weighted sum of precisions at each threshold where the weight is the increase in recall [27].

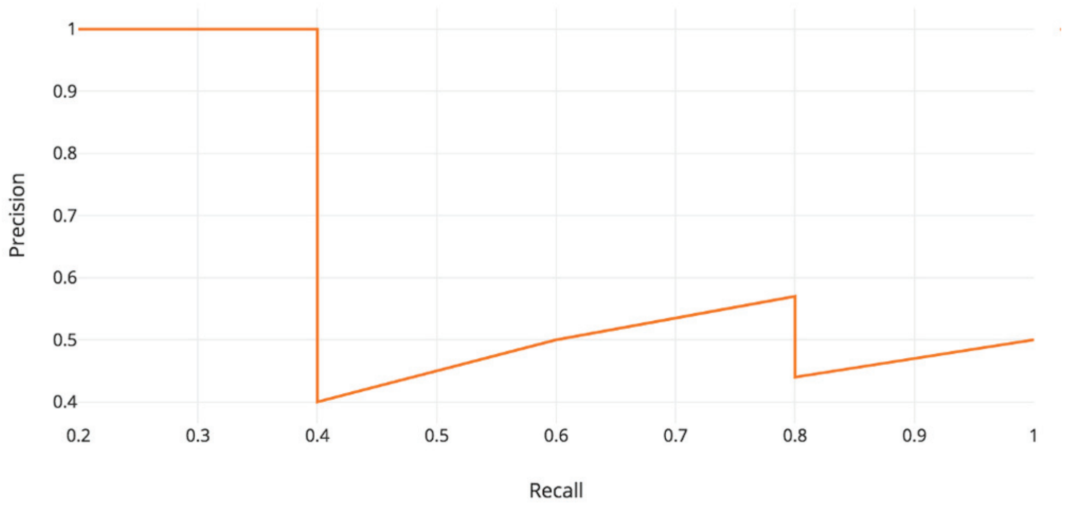


Figure 2.15: Example of precision-recall curve [34]

$$\sum_{k=0}^{K=n-1} [recalls(k) - recalls(k+1)] * precissions(k) \tag{2.3}$$

where n = Number of thresholds.

Mean Average Precision(mAp)

The calculation of AP only involves one class. However, in object detection, there are usually j >1 classes. Mean average precision (mAp) is defined as the mean of AP across all j classes.

$$mAP = \frac{1}{j}\sum_{i=1}^{j} AP_i \text{ [60]} \tag{2.4}$$

with AP_i being the Average Precision(AP) in the ith class and j is the total number of classes being evaluated [60].

2.1.4 Evaluation Metrics of Object Tracking

Various metrics are used in this thesis to evaluate Norfair's effectiveness which is the object tracking algorithm we will use. Below is a summary of each metric.

Mostly Tracked(MT)

It is considered mostly tracked if an object is successfully tracked for at least 80% of its life span. For example, if a video has 20 frames, the fish must be tracked successfully in at least 16 of those frames. It's worth noting that whether the ID stays the same in the track has no bearing on this metric [23].

Partially Tracked(PT)

If an object is successfully tracked for less than 80% of its life span but more than 20% of its life span, it is called partially tracked. If a video contains 20 frames, the fish must be successfully tracked in at least five of them. It's worth noting that the ID of the track has no impact on this metric[23].

Mostly Lost(ML)

An object is considered mostly lost if it is tracked for less than 20% of its life span. If a video has 20 frames, the fish will be tracked in less than 5 of them. It's important to note that the track's ID has no bearing on this metric[23].

False Negative(FN)

Number of frames where ground truth contains at least one object, while tracker either does not contain any object or none of the system's objects fall within the bounding box of any ground truth object on all the frame [10].

False Positive(FP)

Number of frames where the tracker results contain at least one object, while ground truth either does not contain any object or none of the ground truth's objects fall within the bounding box of any system object [10].

Identity Switching(IDSW)

Identity switching(IDSW) refers to the number of times a tracked trajectory changes its matched ground-truth identity. An IDSW occurs when the routing changes from the previously allocated red track to the blue track, as shown in the Figure 2.16. Red represents the trajectory direction that is aligned with the ground truth identity, and blue represents the alternative path, so an IDSW occurs when the routing changes from the previously allocated red track to the alternative path [23].

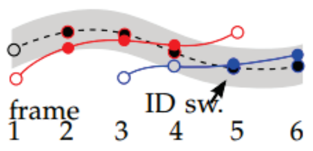


Figure 2.16: Example of identity switching. [23]

Fragmentation(FM)

The number of times a ground truth trajectory is interrupted(untracked) is known as fragmentation in object tracking. In other words, fragmentation is counted every time a trajectory's status shifts from tracked to untracked, and then tracking is resumed at a later time [23].

Multiple Object Tracking Accuracy

The most widely used metric for evaluating a tracker's output is multiple object tracking accuracy. This metric takes into account three types of errors: false positives, missed targets, and identity switches. The Formula for calculating MOTA is shown below

$$MOTA_t = 1 - \frac{\sum_t (FN_t + FP_t + IDSW_t)}{\sum_t GT_t} \text{ [23]} \tag{2.5}$$

where $IDSW_T$ is the identity switches respectively for time t, GT is the number of ground truth objects, FN_t is the number of false negatives (missed targets), FP_t the number of false positives [23].

2.2 Related Work

The different devices used for fish monitoring are discussed in Section 2.2.1, while the different algorithms used in solving fish classification and detection are discussed in Section 2.2.2, and the diverse research works for implementing fish tracking are discussed in Section 2.2.3.

2.2.1 Fish Monitoring

We worked on two datasets: one is the Fishery-acoustic-observation dataset [53], which was gathered using acoustic cameras and used for fish detection and classification, and the other is the Wells-dam dataset [93], which was collected using optical cameras and used for fish tracking. This section describes the different fish monitoring and dataset gathering studies that have been carried out using acoustic and optical cameras.

Using an acoustic camera, which is a sort of imaging equipment used to detect and classify sound sources and consists of a collection of microphones, sometimes known as a microphone array, from which signals are captured and processed in real-time to generate a representation of the position of sound sources [91]. Another tool is ProViewer 4.2 Software [15], which is used for fish monitoring and allows for the analysis of large amounts of data at a lesser cost. Acoustic cameras technical features, such as video viewing to collect data, can greatly enhance diadromous fish population monitoring. The method proved to be a reliable, flexible, and compact method for monitoring fish in coastal lagoons and other transitional environments, allowing researchers to characterize schooling activity and abundance of marine migratory fish during the winter migration season. Intensive long-term monitoring of fish movements like this will help researchers better understand the biological and environmental factors that influence their migration patterns, as well as deal with quantitative assessments of local fish stocks.

The use of an acoustic camera in shallow water [52] is another method as a new hydroacoustic method for tracking migratory fish populations. This approach appears to be a reliable and cost-effective method for estimating fish abundance for management purposes. The acoustic lens will focus on objects up to 1 meter away. It enhances fish behavior by direct video-like visualization of passages in the detection beam, has no effect on most species migratory behavior, and allows the owner of a long data sequence, which is important in long-term monitoring in a changing environment. Because of these features, an acoustic camera is an effective tool in a variety of applications where fish populations are difficult to observe using traditional methods.

For fish monitoring, another approach is to use deep neural networks following a combination of sonar and optical camera images [85]. It's a generative adversarial network that learns how to translate sonar and optical images into each other. During the training and evaluation stage, they used the captured sonar images as input data and the camera images as realistic images. The advantage of this approach is that it could produce realistic images using sonar data with some degree of accuracy even if the dark factor reached 1(completely dark).

DIDSON multi-beam sonar is another method [87], which is a widely used tool for fish monitoring. During stationary applications in lakes, Tushar et al. [87] used this system to detect and measure fish. This unit was able to track all of the deployed fish, which ranged in length from 10 to 60 cm. When the fish were perpendicular to the transducer at every point from the middle to the edges of the beam array, detection of all the fish was not an issue. There is no question that DIDSON's 96-beam array horizontal plane has the best resolution for detecting a target [57]. This study also demonstrated that the DIDSON is the most effective and precise instrument for obtaining biological data.

Using acoustic cameras and optical cameras for fish monitoring and data collection is a more reliable source for applying deep learning algorithms to determine fish movement, as well as the best source for performing fish detection and classification tasks, as shown by some of the methods. The following subsection go through the research that was done to detect fish and classify them by species using various deep learning algorithms.

2.2.2 Fish Detection and Classification

The numerous deep learning methods for solving the problem of detection and classification of fish species using acoustic camera data are described in this section. Since we used deep learning algorithms to solve the detection and classification problem of fish species in our thesis, these research works give us a general understanding of the approach taken by other researchers to solve the problem we are trying to solve. To the best of our knowledge, no one has used deep learning to detect and classify fish using visual sonar.

One of the methods proposed for fish classification is the use of a novel technique based on the use of convolution neural networks and image processing techniques [67].

In the first step, image processing is used to eliminate underwater obstacles, dirt, and non-fishing bodies from underwater images. In the second step, a deep learning approach is used by implementing a convolutional neural network with 32 filters and a max-pooling layer for processing input features, as well as a fully connected layer that classifies images into relevant categories. While this is a good method for fish species classification, it cannot work better in situations where there are so many water bodies and background noise.

The vgg-16 network architecture [79] is used in for fish classification on the noisy dataset of fish boat images [5]. These authors used two methods to test classification tasks: one is to use the vgg-16 architecture, which is initially untrained, and the other is to apply transfer learning to the vgg16 model from a pre-trained network on the ImageNet dataset [73]. Both approaches involve the same model architecture which contains 5 blocks of 13 convolutional layers, a max-pooling layer between successive convolutional blocks with a unit stride of downsampling, and three fully connected layers. This study showed that deep convolutional neural networks performed better in noisy images because the network was able to learn category-specific deep learning features, which aided incorrectly classifying the fish species. This study also found that although transfer learning reduced training time, the output of the initially untrained model is superior, implying that transfer learning works best when we have a pre-trained dataset that is similar to the fish classification problem which we are trying to solve.

The usage of a distributed pipeline for processing DIDSON data that was built using the Hadoop framework for performing classification is another method discussed in [45]. A java software converts the raw DIDSON acoustic data produced by the DIDSON hardware into the proper format and saves it to the HDFS. Filtering, tracking, and target identification code executed through YARN and the mapReduce api extracts potential targets as thumbnails and saves them to hdfs. Thumbnails are converted into a feature vector using feature generation code, which is performed via yarn. Finally, using a streaming task, the feature vector is passed through a classifier. This pipeline can ingest raw DIDSON data, convert the acoustic data to images, filter the images, detect, and extract motion, and generate feature data for machine learning and classification.

Using underwater vision data with high attenuation of lights, extreme noise, and haze in the images, a system for real-time fish detection based on the you only look once (YOLO) architecture‘[68] has been proposed. The HOG (Histogram of Directed Gradients) algorithm was compared to the YOLO model in this study. YOLO outperformed the HOG algorithm in terms of detection accuracy and processing speed and was able to reliably detect fish in noisy, dimlight, and hazy underwater images [82].

In [35] a method for fish detection and species classification in underwater environments using deep learning with temporal information, was presented. They used the combination of optical flow and gaussian mixture models with YOLO deep neural network which is a unified approach to detect and classify fish in unconstrained underwater videos. This method was able to detect freely swimming fish that have poor visibility due to water murkiness, low-resolution imagery, and low light conditions.

A deep learning model known as convolutional neural network (CNN) architecture with 22 layers is proposed for accurate and fast identification of coral reef fish in underwater images [89]. They proposed a system for aiding in the identification of fish species on underwater images, and they compared model performance to human skill in terms of speed and accuracy. In this study, the classification of species is done using a CNN model. About 900,000 images were used to train the CNN, which also included: (i) entire fish bodies, (ii) partial fish bodies, and (iii) the climate (e.g., reef bottom or water). The CNN was also able to recognise fish partly concealed behind coral or other fish, and was better than humans at identifying fish on tiny or fuzzy images, whereas humans were better at identifying fish in unusual positions (e.g., twisted body). This approach is useful for identifying fish on underwater images and has the potential to develop new video-based protocols for monitoring fish biodiversity cost-effectively and efficiently.

The Mask-RCNN model [29] is proposed for the detection of fish species from videos collected during fish practices of vessels using electronic monitoring system(EMS) [86]. Using mask regional-based CNN, the fish in the frames of the EMS Video were identified and segmented from the background at the pixel level(Mask RCNN). A feature extractor, region proposal network, bounding box recognition (BBR), and a mask network(MN) were all part of the model. The RPN(Regional proposal Network) then suggested the regions of interest in the feature maps. The ROI and corresponding feature maps were then placed into the BBR to evaluate the object's classes and precise bounding boxes. To create binary masks for the objects, the ROI and corresponding feature maps were also fed into the MN. Its ability to detect fishes in videos at a higher frame rate per second with greater precision, as well as its ability to build segmentation masks around detected objects, make this model one of the most effective models for fish species detection and classification.

We've seen many deep learning models proposed for solving the problem of fish classification and detection in various scenarios where the dataset includes a large number of fuzzy, murky, algae-covered videos, background noise, and so on. However, various image processing techniques were used to clean the data, and we also saw one of the approaches known as transfer learning, which allows us to use an architecture that has already been trained with a similar dataset and can act as a perfect feature extractor while reducing training time.

2.2.3 Fish and Multiple Object Tracking

One of the problems addressed in this thesis is tracking fish. Fish tracking is important because it helps us understand the distribution and availability of fish, as well as providing a basis for long-term fisheries management [3]. In this section, we discuss previous research work on fish and other object tracking using various algorithms.

This research paper [81] introduces a covariance-based approach for fish tracking that helps marine biologists in understanding the complex environment by allowing them to conduct intelligent video analysis. The tracked object is initially regarded as an entity that represents a single fish and contains details about the fish's appearance background and current co-variance model, whereas the detected object is regarded as a moving object that has yet to be identified with any tracked object. The corresponding covariance matrix for each detected object is then computed by constructing a feature vector for each pixel made up of the pixel coordinates. This feature vector is then used to calculate the covariance matrix, which models the object, and it is then linked to the detected object. The object is then compared to the currently tracked objects using this matrix to determine which one it most closely resembles. This method was tested using hand-labeled ground truth data from 30000 frames from 10 separate underwater videos, and it performed well under a variety of conditions in the dataset, including poor image contrast due to bad weather, murky water, multiple fish occlusions, and so on [81].

Another approach used in this research paper [12] for object tracking is the SORT algorithm. SORT is a tracker that operates on the concept of detection-based tracking. To detect objects, the design includes a powerful detector known as YOLO. Following detection, the Hungarian algorithm [42] and Kalman filter [42] are used to track objects using YOLO's detections. SORT keeps track of each detection by giving each bounding box a unique ID. When an object is lost due to occlusion, incorrect detection, or other factors, the tracker assigns a new ID and begins monitoring the newly discovered object. This architecture was put to the test on a variety of videos containing objects of various classes. They were able to track the objects in their dataset with high precision, and they discovered that tracking performance is influenced by detection performance [11].

This paper [92] proposes the deepsort algorithm, which is a tracking-by-detection technique for real-time multiple object tracking and an improvement of the SORT algorithm[12]. Multiple Object Tracking entails calculating the trajectory of multiple objects at the same time in a sequence of video frames. To get all the detections from a given frame, the YOLO and Retinanet [47] frameworks were used, and then the detections from each frame were fed into a pre-trained CNN model, which produced an association matrix related to each detection that included the appearance features of the objects. This vector which becomes the "appearance descriptor" of the object [51] is fed into the kalman filter, which predicts the position of the bounding boxes in future frames. The IOU score was used by the hungarian algorithm to create an assignment cost matrix that linked projected bounding boxes to previously created tracks. This approach was put to the test on the VisDrone 2018 dataset [95], which contains several video clips shot with drone-mounted cameras in a variety of scenarios [39]. This method is able to track objects through longer periods of occlusions, effectively reducing the number of identity switches [92].

There are many other methods proposed for solving fish tracking which was discussed in [4, 32, 58]. These methods are hardcoded to a fixed distance function and to tracking boxes where the function used to calculate the distance between tracked objects and detections is not defined by the user. So for having more flexibility in this functionality we choose a tracker known as Norfair [6] which is a customizable lightweight Python library for real-time 2D object tracking. In this tracker the function used to calculate the distance between tracked objects and detections is defined by the user, making the tracker extremely customizable. Norfair operates by predicting each point's future location based on its previous positions. It then attempts to align these approximate locations with the detector's newly observed points. Norfair can use any distance function defined by the user to perform this matching. As a result, each object tracker may be as basic or as complex as required. As a result, we chose norfair as the algorithm to solve the fish tracking problem.

Chapter 3

Materials and Methods

In this chapter, details about the datasets used and different pre-processing steps applied are discussed in Section 3.1 and Section 3.2 discuss about the methods used for solving fish detection and classification and methods used for fish tracking are discussed in Section 3.3

3.1 Materials

Two different publicly available datasets were used in this thesis, one for fish detection and classification and the other for fish tracking. The following sub-sections describe the datasets in detail, as well as the pre-processing steps performed on the datasets.

3.1.1 Dataset for Fish Detection and Classification

Fisheries-acoustic-observation dataset [53] is the dataset that we used for fish detection and classification. The original data was obtained by a DIDSON imaging sonar on the Ocqueoc River. The data in this dataset are stored in two formats: raw format which is acoustic data as collected from the DIDSON device and binary format that contains images of the visualised acoustic data. We used raw data in our analysis. The raw data directory contains two subdirectories that separate the data by year. The naming of the raw data follows a pattern of yyyy-mm-dd_hhmmss_HF.ddf that encodes the year, month, day, and the start time of collection. Each video file is 30 minutes in duration. In total, the raw data contains 105 raw DIDSON files from 2013 and 95 from 2016. These data represent approximately 100 h of data collection, 524 clips with known targets were extracted from the 100 hours of DIDSON data. These clips are stored in separate folders called OC13-by-Expert, OC13-by-PIT, and OC16-by-VIDEO, and are divided by year of collection and specified methods such as Passive Integrated Transponders (PIT) tags, video surveillance, or experts. These directories are further broken down by species of fish that clips contain. Table 3.1 displays how many video clips of each species are contained in these directories [53]. As we have used images for performing classification and detection tasks, Section 3.1.2 explains the steps performed in converting these video to images.

Count of number of videos of each species	
Species	Number of Videos in Dataset
Carp	51
Lamprey	190
Largemouth bass (Lmbass)	1
Pike	2
Smallmouth-bass (Smbass)	65
Steelhead	6
Sucker	100
Walleye	109

Table 3.1: Count of number of videos of each species.

3.1.2 Pre-Processing Steps for Fish Detection and Classification

To match the requirements of the deep learning models, we processed the video files of the Fisheries-acoustic-observation dataset. YOLOv3, the first deep learning model, requires data to be in YOLO format for training. As a result, each image file in all directories is labelled, and the labelled coordinates, which include the object type, object coordinates, height, and width of the object, are saved in a text file with the same name as the image file. For the second deep learning model Mask-RCNN, the data must be in Pascal-VOC format, which is an XML file generated for each image in the same directory. Each XML file contains annotations for the corresponding image file, such as object type, object coordinates, height, and width. The Wells dam dataset used for fish tracking is already available with labels for all of the images, so no pre-processing steps were applied for this dataset.

Converting DDF to AVI format

Multiple steps were performed to prepare the data for deep learning models. Initially, the Fishery-acoustic-observation dataset's raw data is in DDF format. We have used software known as Didson-V5 software to open these DDF files. Didson-V5 is a sound metrics software that was developed to help users get more features from their image sonars. Using this software all the DDF video files present in folders OC13-by-Expert, OC13-by-PIT and OC16-by-VIDEO were converted into AVI files. In total 524 AVI files were obtained of all species. The frame rate of these videos was about 7 frames/second. A sample view of a frame from a video is shown in Figure 3.1.

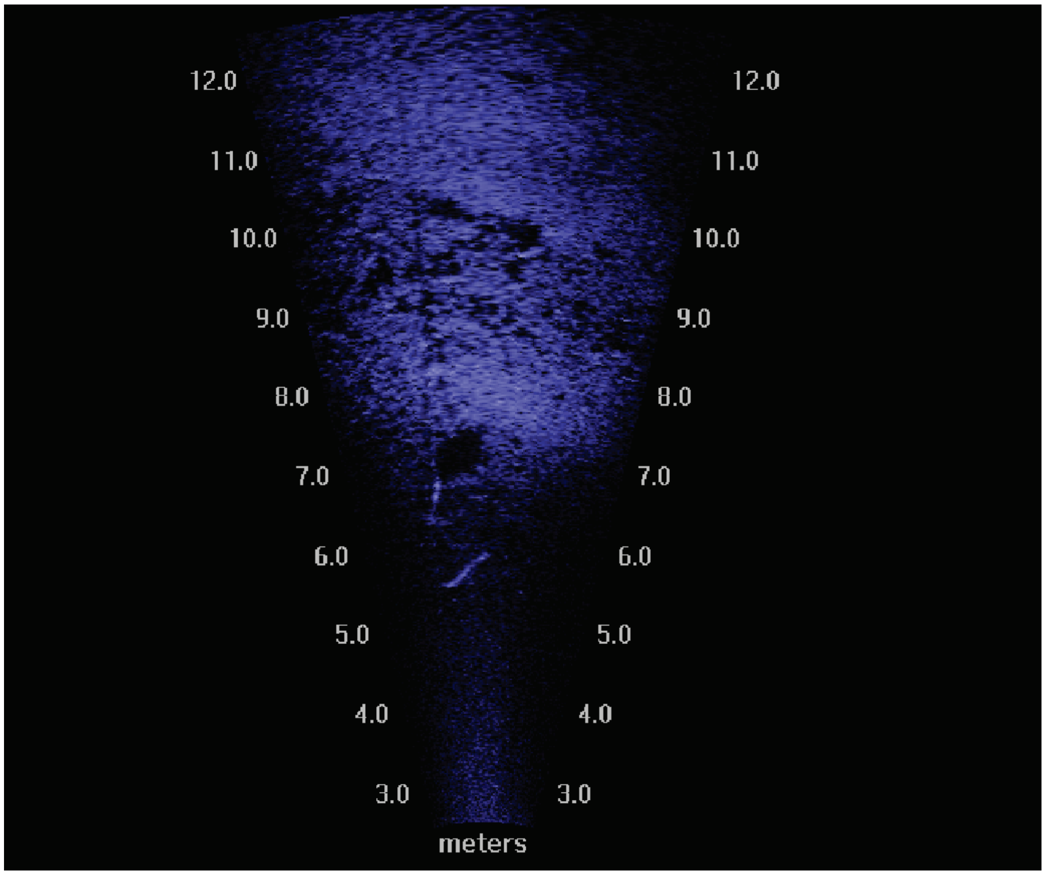


Figure 3.1: Sample view of a frame in a AVI file

Converting AVI files Into Images

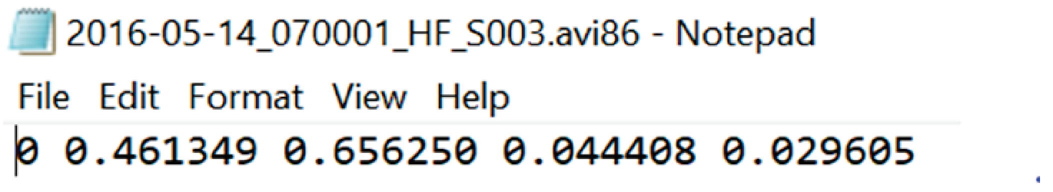
Images are the necessary format required for training deep learning models that were used in this thesis. As a result, we used a python script to extract images from all AVI files at 4 fps (frames per second). The images of each species retrieved from these videos were saved in their own folder. In total 7953 images were extracted of "Carp" species, 6986 images of Lamprey, 160 images of Lmbass, 350 images of Pike, 11690 images of Smbass, 582 images of Steelhead, 21676 images of the Sucker, 65 images of

Trout, and 11973 images of Walleye were extracted from the videos.

Labelling the Images

At this pre-processing level, labeling the images is the most important step. Since we are using two deep learning models for training, one is YOLOv3 and the other is Mask-RCNN, we need the data to be labeled in two different formats. So, for the first model which is YOLOv3, YOLO format labeling is required, while the Pascal-VOC format is required for the second deep learning model which is Mask-RCNN. The YOLOv3 labeling method will be discussed first, followed by a discussion of the Pascal-VOC data format. Recall that we extracted videos of fish from raw data, from which later images were extracted at a rate of 4 frames per second. To train with deep learning algorithms, we require the images as well as the bounding boxes created on the objects in the images. LabelImg [88] is the tool that is used to draw the bounding boxes around images for this purpose. Together with me, three people from Innovasea helped me in labelling the images using the guidelines I wrote on how to label the images, resulting in a total of 72 hours of labeling time. When each image is labelled, a text file containing information about the bounding box coordinates and class number of each object present in the image is generated.

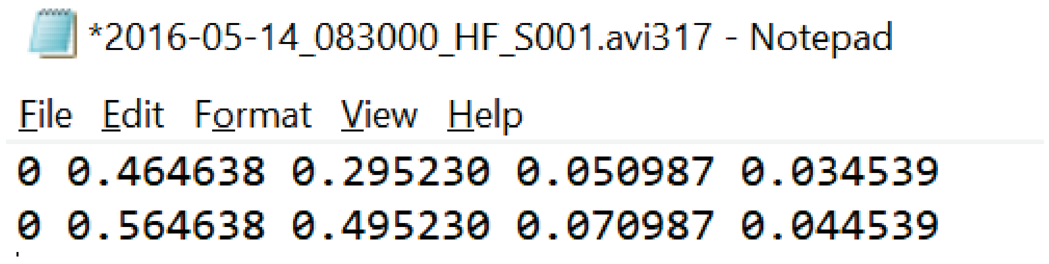
LabelImg tool creates text files for all images with objects, and a python script is written to create empty text files for images without objects. The image name and the text file generated during labelling must be identical. The YOLO labeling data format is depicted in Figure 3.2. The first value in the text file shows which class the labeling object belongs to and the next four values are the object's bounding box coordinates ($x-min$, $x-max$, $y-min$, and $y-max$). This is an example of how the text file looks when only one object is present in the image, as shown in Figure 3.2.


2016-05-14_070001_HF_S003.avi86 - Notepad
File Edit Format View Help

```
0 0.461349 0.656250 0.044408 0.029605
```

Figure 3.2: Example of a text file where only one object is present in the image

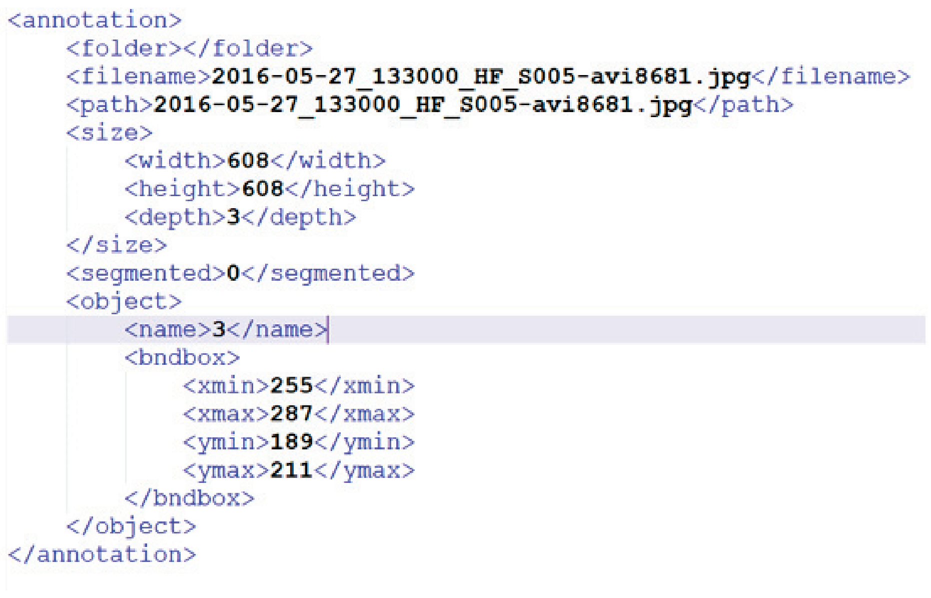
Figure 3.3 depicts how the text file looks when multiple objects are present in the same image. The values in the text file represent the same as previously discussed.


*2016-05-14_083000_HF_S001.avi317 - Notepad
File Edit Format View Help

```
0 0.464638 0.295230 0.050987 0.034539
0 0.564638 0.495230 0.070987 0.044539
```

Figure 3.3: Example of a text File if multiple Objects are Present in the image

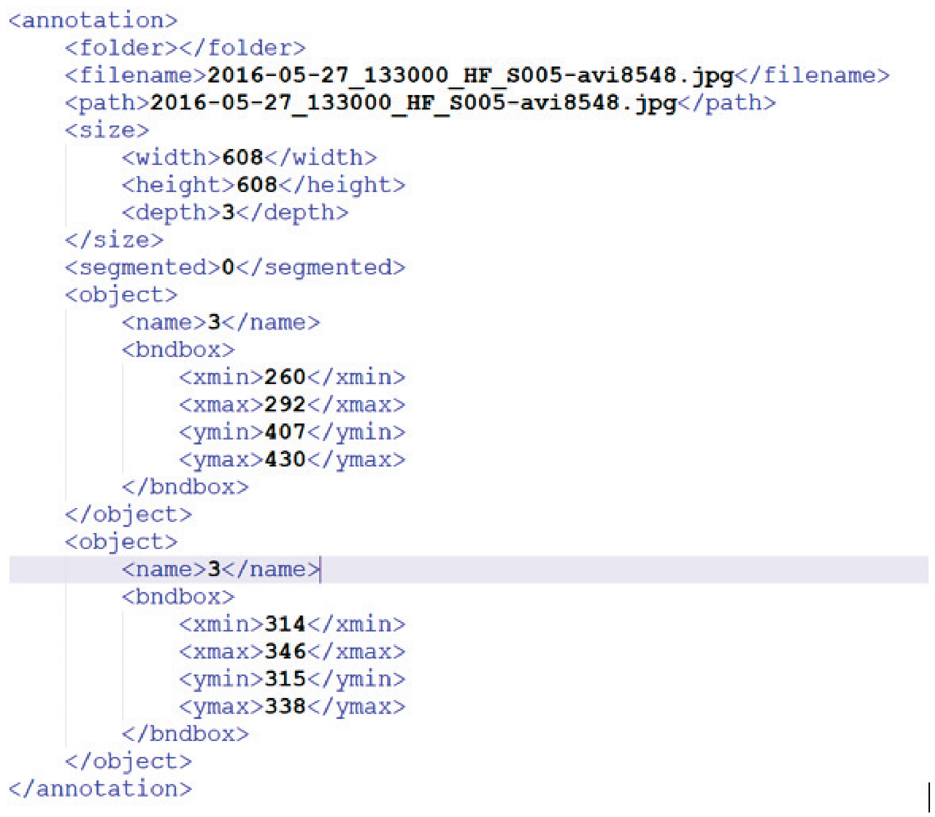
Since labeling all of these images is time-consuming and difficult for the second deep learning model which is Mask-RCNN, a python script is written to convert the YOLO labeling formats to the Pascal-VOC format. The Pascal-VOC format will be an XML file. The image of a Pascal-VOC file is shown in Figure 3.4. The filename in the XML file indicates which image this labeling belongs to, the path indicates the image's location, the width, height, and depth indicate the image's dimensions, the name indicates which class the object in the image belongs to, and the *xmin*, *xmax*, *ymin*, and *ymax* indicate the bounding box coordinates of the object in the image.



```
<annotation>
    <folder></folder>
    <filename>2016-05-27_133000_HF_S005-avi8681.jpg</filename>
    <path>2016-05-27_133000_HF_S005-avi8681.jpg</path>
    <size>
        <width>608</width>
        <height>608</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>3</name>
        <bndbox>
            <xmin>255</xmin>
            <xmax>287</xmax>
            <ymin>189</ymin>
            <ymax>211</ymax>
        </bndbox>
    </object>
</annotation>
```

Figure 3.4: Example of a XML file if single object is present in the image

The image in Figure 3.5 shows how the annotation file of the Pascal-VOC format looks when there are multiple objects in the image.



```xml
<annotation>
    <folder></folder>
    <filename>2016-05-27_133000_HF_S005-avi8548.jpg</filename>
    <path>2016-05-27_133000_HF_S005-avi8548.jpg</path>
    <size>
        <width>608</width>
        <height>608</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>3</name>
        <bndbox>
            <xmin>260</xmin>
            <xmax>292</xmax>
            <ymin>407</ymin>
            <ymax>430</ymax>
        </bndbox>
    </object>
    <object>
        <name>3</name>
        <bndbox>
            <xmin>314</xmin>
            <xmax>346</xmax>
            <ymin>315</ymin>
            <ymax>338</ymax>
        </bndbox>
    </object>
</annotation>
```

Figure 3.5: Example of a XML file if multiple objects are present in the image

3.1.3 Dataset for Fish Tracking

The dataset we used for fish tracking is referred to as the Wells dam dataset [93]. Fish were recorded through a passage viewing window at a dam in eastern Washington to build this dataset. These videos include Chinook, Jack Chinook, and Sockeye species. The frame rate was exactly 30 frames per second, and the video image size was exactly 1280×960 pixels. A total of 24000 frames are available, with 13405 of them containing fish. As this dataset contains images along with labels, there are no pre-processing steps applied to this dataset.

3.2 Methods for Fish Detection and Classification

For fish detection and classification, we used two deep learning models, YOLOv3 and Mask-RCNN, in this thesis. We chose YOLOv3 because of its ability to detect free-moving fish that are camouflaged in the background, its processing speed, as well as its ability to detect and classify fish from videos with varying frame rates. The ability to detect and clasify fish with high precision, as well as the ability to create a mask around the detected object, lead us to choose the Mask-RCNN as the other model. More information about these models is provided in the subsections that follow.

3.2.1 YOLOv3 Model

YOLOv3 [70] is one of the most effective models for detecting objects. We chose it because of its average detection rate of about 24 frames per second, its ability to detect objects in real-time, and its ability to detect and locate multiple objects on a single image.

Architecture of YOLOv3

Figure 3.6 depicts how the YOLOv3 architecture looks like. The feature extractor, Darknet-53 (a convolutional neural network with 53 layers), extracts features from images at three different scales and feeds them into a detector to predict bounding box coordinates and class probabilities of objects in the image. The detections are rendered on three different scales here as well [94].

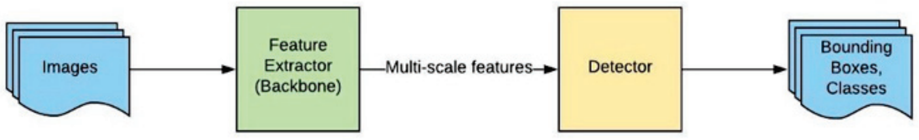
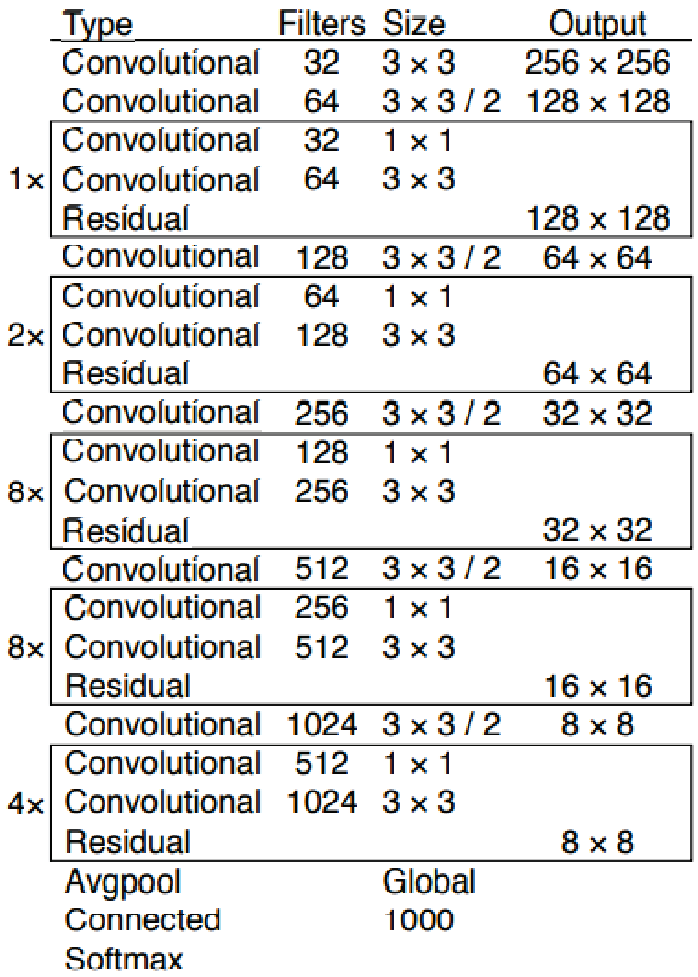


Figure 3.6: Overview of Yolov3 architecture [94]

Figure 3.7 depicts the darknet-53 [70] network, which is made up of 53 convolutional neural layers. Each rectangle in Figure 3.7 represents a residual block that was introduced to solve the gradient disappearance problem so that networking training will be easier. Batch normalization and a leaky rectified linear activation function(ReLU) layer follow each convolutional layer. A bottleneck structure of 1×1 followed by 3×3 followed by a skip connection can be found within each residual block. The extracted features from the last three residual blocks are fed into the detector for object detection in images [70].



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
	Convolutional	32	1 × 1	
1×	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
	Convolutional	64	1 × 1	
2×	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
	Convolutional	128	1 × 1	
8×	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
	Convolutional	256	1 × 1	
8×	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
	Convolutional	512	1 × 1	
4×	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 3.7: Darknet-53 architecture [70]

YOLOv3 detects at three different scales and in three different locations in the network as shown in Figure 3.9. The layers where the detections are rendered are 82, 94, 106 layers. Using 32, 16, and 8 network strides, the YOLOv3 network down-samples images. If the input of the network is 608×608, the output of the detectors will be 608/32 =19(19×19) for strides 32, 608/16=38(38×38) for strides 16, and 608/8=76(76×76) for strides 8, which are known as grids, where the first size of the output grid will be responsible for the detection of large objects, second size of the output will be responsible for the detection of medium objects, and the last output is responsible for the detection of small objects. Each grid cell is in charge of predicting B bounding boxes and C class probabilities for objects whose centers fall inside the cell. The number of anchors used is represented by the letter B. Every bounding box's attributes are $(5 + C)$. The letter C denotes the number of classes available. The trust score indicates the likelihood of an item being found in a box. The level of confidence varies from 0 to 1 [75].

YOLOv3 generates a 3-D tensor with the form $[S, S, B \times (5 + C)]$ where S is the number of grid cells, B is the bounding boxes and C is the class probabilities after applying a single forward pass convolutional neural network to the entire image since we have a $S \times S$ grid of cells [75]. This is depicted in Figure 3.8 below.

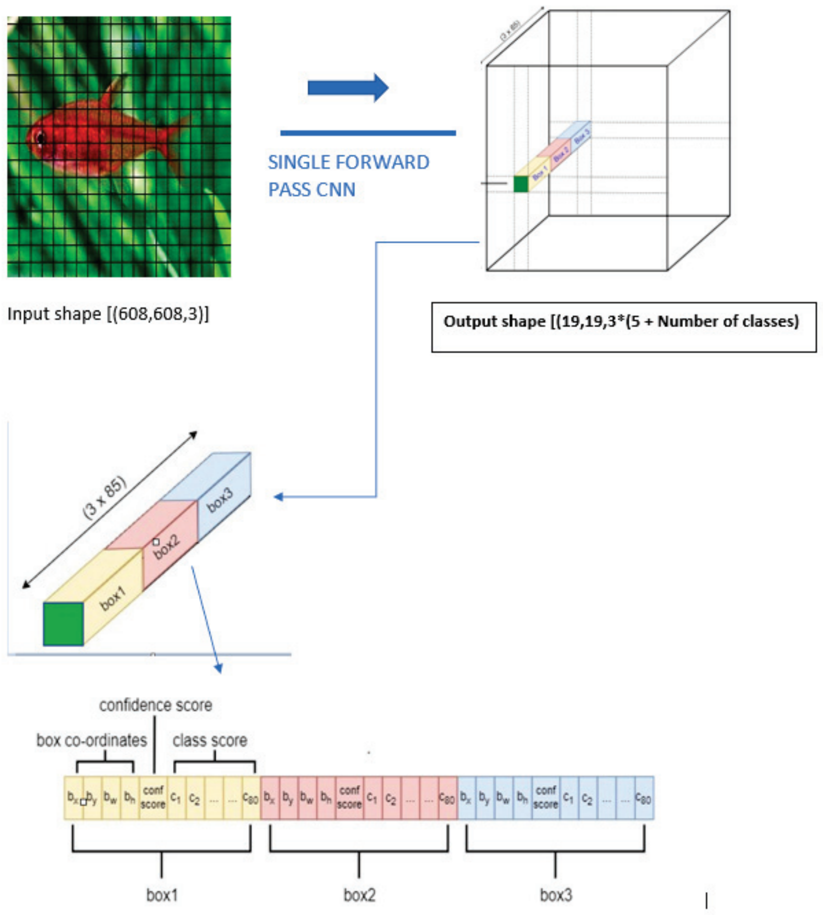

SINGLE FORWARD PASS CNN
Input shape [(608,608,3)]
Output shape [(19,19,3*(5 + Number of classes)]
(3 x 85)
box1
box2
box3
confidence score
box co-ordinates
class score
b_x b_y b_w b_h conf score c_1 c_2 ... c_80
box1
box2
box3

Figure 3.8: Overview of how the prediction of bounding boxes and class probabilities are calculated during single forward pass of network [75]

Figure 3.9 shows the 106 layers fully convolutional underlying architecture for YOLOv3. YOLOv3 downsamples the input image to 19×19 pixels for the first scale and predicts the 82nd layer, assuming the network size is 608×608 pixels. The first detection scale generates a 3-D tensor of size $19 \times 19 \times B \times (5 + C)$. YOLOv3 then adds one convolutional layer to layer 79's feature map before upsampling it by a factor of two to a scale of 38×38 pixels. The upsampled feature map is then concatenated with the feature map from layer 61. Before being subjected to the 2nd detection scale at layer 94, the concatenated feature map is passed through a few more convolutional layers [75].

The second prediction scale yields a 3-D tensor with dimensions of $38 \times 38 \times B \times (5 + C)$. The same design is used once more to estimate the third scale. After one convolutional layer is added, the feature map of layer 91 is concatenated with a feature map from layer 36. The final prediction layer is Layer 106, resulting in a 3-D tensor with dimensions of $76 \times 76 \times B \times (5 + C)$ [75].

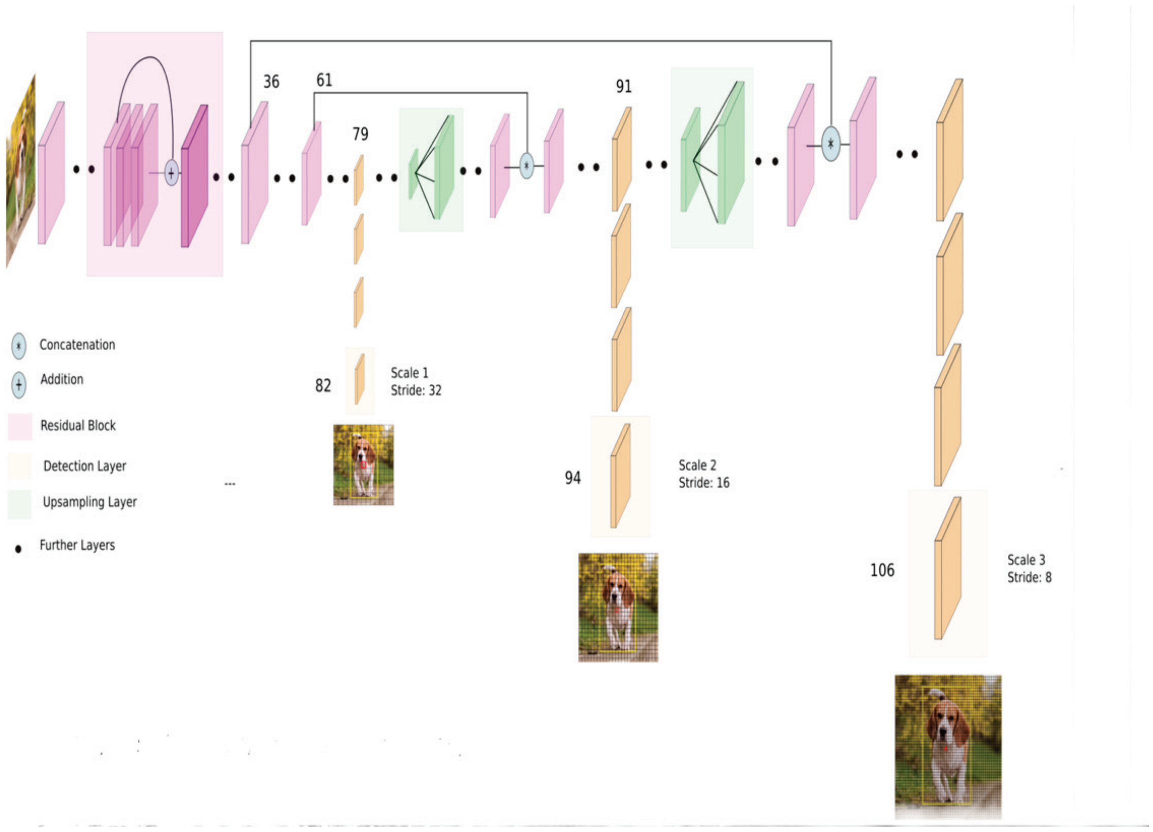


Figure 3.9: Overview of Yolov3 network architecture [9]

Anchor Boxes

To predict bounding boxes YOLOv3 uses pre-defined bounding boxes that are called anchor boxes. The estimated bounding box's actual width and height are calculated using these anchor boxes. A total of nine anchor boxes are used, with three anchor boxes for each scale. Using three anchors, each grid cell of the feature map will predict three bounding boxes. The default width and height of anchor boxes are (116×90), (156×198), (373×326) for scale-1, (30×61), (62×45), (59×119) for scale-2, and (10×13), (16×30) for scale-3 (33×23) [75].

Bounding Box Predictions

Bounding priors are anchors that are determined using the k-means clustering process. YOLOv3 calculates offset to pre-defined anchors to predict the real width and height of bounding boxes. This transformation is also known as log-space transform. YOLOv3 uses the sigmoid function to predict the center positions of bounding boxes. The equations that are used to obtain predicted bounding box distance, height, and center coordinates are shown in Equation 3.1.

$$b_x = \sigma(t_x) + c_x \tag{3.1}$$

$$b_y = \sigma(t_y) + c_y \tag{3.2}$$

$$b_w = p_w e^{t_w} \tag{3.3}$$

$$b_h = p_h e^{t_h} \tag{3.4}$$

where bx, by, bw, bh are center, width, and height of predicted bounding box. tx, ty, tw, th are the output of the network after training [75].

The below Figure 3.10 shows an overview of prediction.

Non-Max Suppression

YOLOv3 predicts ($19 \times 19 \times 3$) = 1083 bounding boxes for scale-1, ($38 \times 38 \times 3$) = 4332 bounding boxes for scale-2, and ($76 \times 76 \times 3$) = 17328 bounding boxes for scale-3 using the input image of 608×608 pixels. On a single pass on an image, the YOLOv3 predicted approximately 22743 bounding boxes. But, as seen in Figure 3.8,

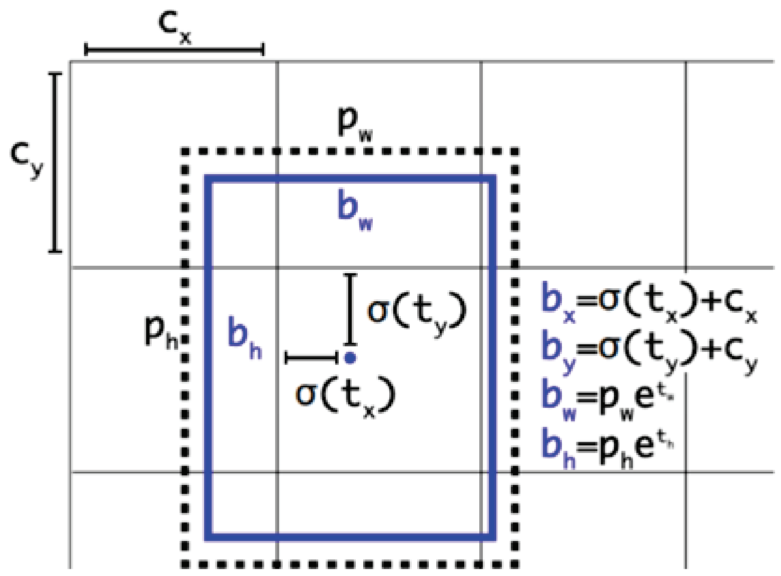


Figure 3.10: Overview of prediction of bounding boxes [70]

we just have an entity (fish), so how do we get rid of the rest of the boxes? The non-max suppression algorithm is used for this purpose. Non-max suppression eliminates YOLOv3's numerous detections of the same object on a single image and retains the object's best bounding box.

Algorithm

1 First the non-max suppression selects the boxes with the highest Confidence score(2.1.3) from the multiple boxes present on the image.

2 Then finds the IOU (Intersection over union) of the selected box with other boxes and remove the boxes with IOU <0.5

3 After step-1 and step-2, it selects the box with the next highest Confidence score and repeats step-1 and step-2 until a unique box is selected for the objects present in the image.

Below Figure 3.11 shows the overall view of the non-max suppression, first, it shows the condition of the bounding box over objects before non-max suppression is applied, and after it shows the output after non-max suppression is applied.

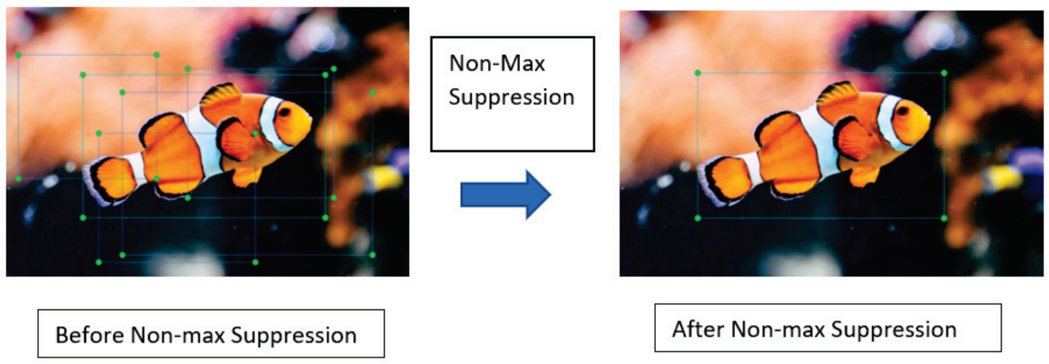


Figure 3.11: Figure showing how the image looks before and after non-Max suppression is applied

3.2.2 Mask-RCNN Model

Mask-RCNN is the second-deep learning model we have used in performing fish classification and detection. The ability to create a mask around the detected objects along with confidence score and bounding box, usage of the fully connected neural network is the reason we have chosen this model. The overview of the Mask-RCNN architecture is explained in the following subsections.

Architecture of Mask-RCNN

Mask-RCNN [29](Mask Regional Convolutional Neural Network) is the RCNN family's fourth model. This is the most recent and advanced model, and it is a modification of the Faster-RCNN model [72]. The architecture of Mask-RCNN is depicted in Figure 3.12. It is divided into two stages, the first of which is the backbone stage, which is built with FPN (feature pyramid network) [46] and Resnet-101 [31], Regional Proposal Network [72] and ROI(Region of interest) [28] align layer for proposing regions which contains objects, and the second is the head stage which contain fully connected layers where the classification, bounding box prediction and mask prediction happens using the proposed regions of each object as input from first stage.

First Stage

The First stage in Mask-RCNN consists of two networks, one is the backbone network and the other is Regional Proposal Network. The backbone network is used to extract features from raw images. The image is fed into the backbone network, and feature maps are extracted. In Mask-RCNN, the backbone network is Resnet101-FPN (feature pyramid network).

In the second step of this stage, the feature maps extracted from the backbone network are fed into the region proposal network. The Regional Proposal Network is used to decide where there is a possibility of object presence in the image. The first step in this network is to slide a window over the CNN feature map that has been generated, with anchor boxes being generated at each window. Anchor boxes are bounding boxes of a fixed height and width that are predefined. Different scales of anchor boxes are used since the objects in the image can have different dimensions. The intersection over union (IOU) method will be used to calculate the overlapping between bounding boxes after the anchor boxes have been created, the bounding box with the highest IOU is chosen, and the remaining bounding boxes are ignored. Finally, Regional Proposal Network gives the area where the object is present a foreground class and the area where the object is not present a background class. The foreground class of the feature map with bounding box area, also called region proposals, will be forwarded to the next level.

Second Stage

Now that the Regional Proposal Network has given the area proposals, which are offsets of each anchor box, we must obtain the proposed bounding box, also known as regional proposals, whose coordinates are centered on the original image scale. The coordinates must be adjusted to match the scale of the feature maps. The ROI align layer, which is a modification to the ROI pooling layer, is used for this purpose. ROI align is introduced to perform data pooling more accurately.

The ROI align process consists of three steps: ROI division, ROI interpolation, and max pooling. ROI division divided each coordinate of each regional proposal obtained after the Regional Proposal Network network by k (where k is the size of the ROI align layer and $k = 7$ according to the research paper [29]. The float values are the new coordinates obtained in this case. The required part is chopped using new coordinates in order to obtain it from the feature map that responds to the supposed object. This cropped portion is also divided into grids, but ROI align chooses four points in each bin using bilinear interpolation on a regular basis to establish concrete values in these bins. The maximum or average value from each bin is then determined using these four points. The output from the ROI align layer is feed into the fully connected layers where the object class, bounding box coordinates, and mask of the object is obtained.

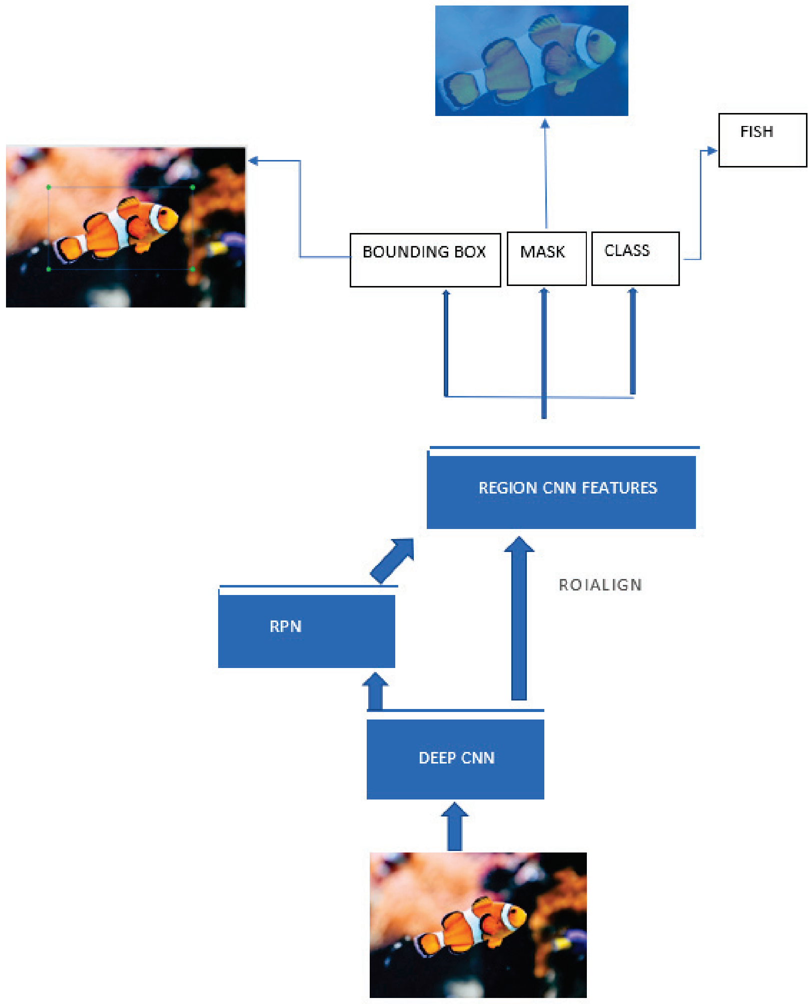


Figure 3.12: Overview of Mask-RCNN architecture [38]

3.3 Method for Fish Tracking

For fish tracking, we used the Norfair tracking library[6] in combination with YOLOv4[14] that is the updated version of YOLOv3 [70], which provides detections and serves as an input to the norfair algorithm. YOLOv4 was chosen because it was the most recent version of YOLO at the time the fish tracking task was performed.

3.3.1 Yolov4

As we saw in Section 3.2.1, YOLOv3 contains mainly two components, one is the backbone or feature extractor which is darknet53 [70] for extracting the features from the images and head or detection blocks which is used for bounding box localization and identifying the class of the object inside the box. YOLOv4 [14] has three components, a feature extractor known as CSP(Cross-Stage-Partial-Connections)Darknet53 [90], a neck that connects the backbone to the head, and contains the spatial pyramid pooling additional module (SPP) [30] and PANet path-aggregation-network [49], and a head that is identical to YOLOv3. Figure 3.13 below depicts a high-level overview of YOLOv4.

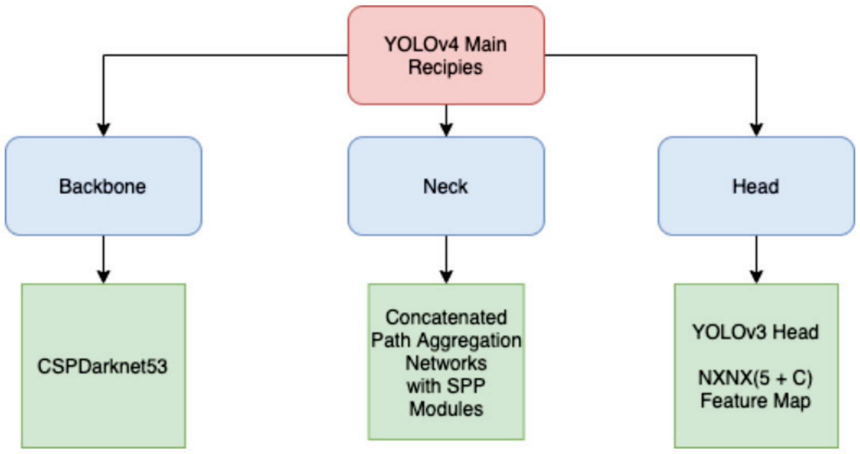


Figure 3.13: Overview of Yolov4 architecture [7]

Cross-Stage-Partial-Connections (CSP) Darknet53

CSPDarknet53 [90] is a DarkNet53-based convolutional neural network and backbone for object detection. It uses a CSPNet(Cross Stage Partial Network) approach to split the base layer's feature map into two sections, then merges them using a cross-stage hierarchy. The adoption of a split-and-merge approach allows the network to have more gradient flow. CSPNet(Cross Stage Partial Network) divides the base layer's feature map into two parts, one of which will pass through a dense block and a transition layer, and the other of which will be integrated with the transmitted feature map in the following stage [90]. Figure 3.14 shows how the splitting happens.

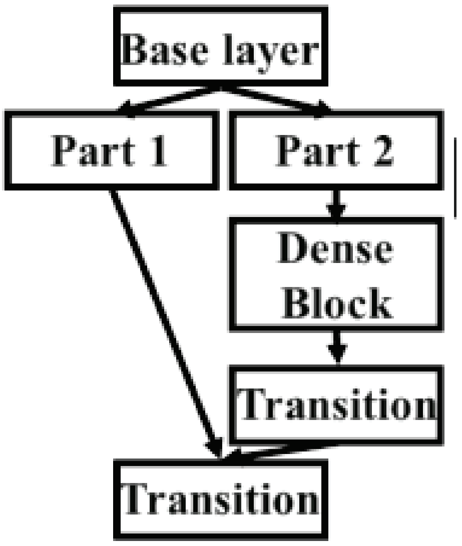


Figure 3.14: Overview of CSPDarknet53 partition architecture [90]

Spatial Pyramid Pooling

Pooling spatial information in small spatial bins is achieved through spatial pyramid pooling [30]. The number of bins and their dimensions is predetermined. Each filter's results are pooled in each spatial bin. Three-level pooling is demonstrated in the sample Figure 3.15. The feature map output has 256 filters and can be any size (depends on input size). The output of the first pooling layer (grey in the illustration) contains a single bin and covers the entire image. This is like the global pooling scheme [84]. This pooling produces a 256-d output. The feature map is pooled to have four bins in the second pooling, resulting in an output of size 4×256. The feature map is pooled to have 16 bins in the third pooling, resulting in an output of size 16×256. The output of all the pooling layers is flattened and concatenated to produce a fixed-dimension output that is independent of the input size [84]. The overview of spatial pyramid pooling architecture is shown in Figure 3.15.

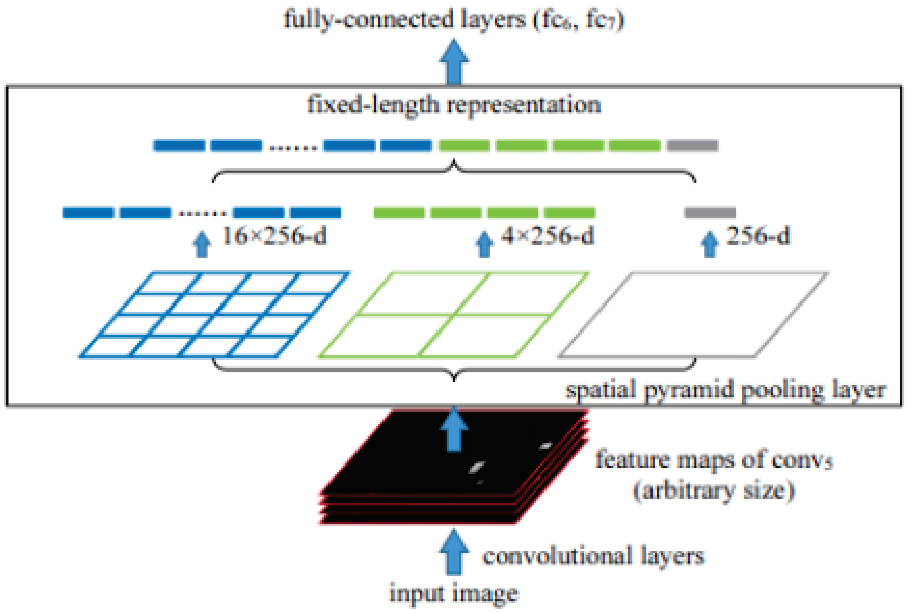

fully-connected layers (fc6, fc7)
fixed-length representation
16×256-d
4×256-d
256-d
spatial pyramid pooling layer
feature maps of conv5
(arbitrary size)
convolutional layers
input image

Figure 3.15: Overview of spatial pyramid pooling architecture [30]

Path Aggregation Network

Path Aggregation Network(PANet) [49] is found in the YOLOv4 model's neck, and it is primarily used to improve the process of instance segmentation by conserving spatial information. PANet is being used for instance segmentation in YOLOv4 because of its ability to reliably preserve spatial information, which aids in proper pixel localization for mask generation [56]. The architecture of PANet is shown below in Figure 3.16.

Here (a) in the Figure 3.16 is the backbone of the feature pyramid network, which allows localised spatial information to move upwards in the red arrow, with the red path passing through 100+ layers, but PANet offers a shortcut path that only passes through roughly 10 levels to reach the top N5 layer, (b) is bottom-up Path Augmentation, which makes it easier to propagate lower layer information to the top, (c) adaptive feature pooling, in which the PANet uses features from all layers to create a network that determines which ones are useful. The network is then able to adapt

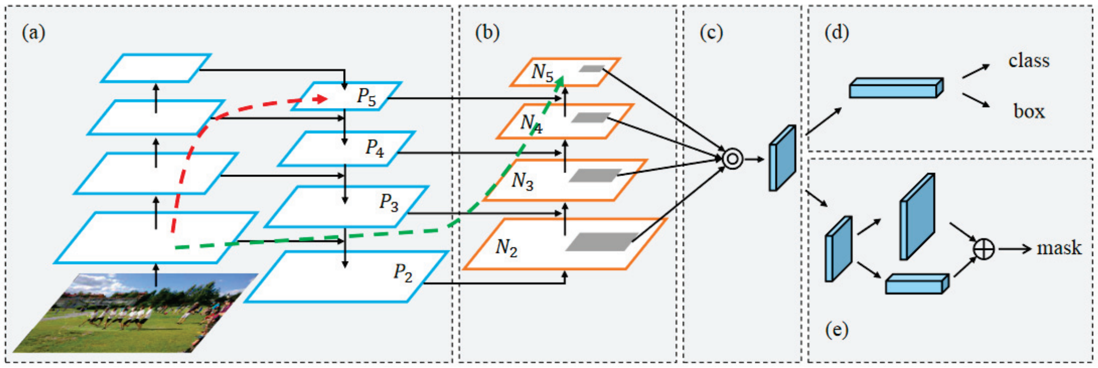


Figure 3.16: Overview of path aggregation network architecture [49]

to new features by an element-wise max fusion procedure, (d) is the box branch, which predicts the class and bounding box coordinates, and (e) is the fully connected fusion, which leverages these layers to offer a more accurate mask prediction using PANet [56].

3.3.2 Norfair Algorithm

Norfair [6] is a lightweight Python library for real-time 2D object tracking that can be customised. Norfair is designed to add tracking capabilities to any object detection model using a few lines of code. For Norfair to operate, we must provide input in the form of detections made by the detector; in our case, YOLOv4 is used as the detector, which first makes detections on the images or videos and then passes those detections per frame to Norfair for object tracking. In the following subsections, we will go through the Norfair tracking library in detail.

How Norfair Works

Norfair operates by predicting each point's future location based on its previous positions. It then attempts to align these approximate locations with the detector's newly observed points. Norfair can use any distance function defined by the library user to perform this matching. As a result, each object tracker may be as basic or as complex as required. There are a few key parameters in the Norfair algorithm that we should be aware of [6].

- **distance_function:** is the function that is used for calculating the distance between newly observed objects and the objects it is currently monitoring. This function should accept two arguments: a detection of type Detection and a tracked object of type TrackedObject, and return a float containing the calculated distance.

- **distance_threshold:** defines the maximum distance at which a match will occur. The tracker can not fit detections or tracked items that are further away than this threshold.

- **hit_inertia_min:** Each tracked object maintains an internal hit inertia counter that tracks how frequently it is matched to a detection; when it matches, this counter rise, and when it does not, it falls. The object is destroyed if it does not find a match for a specified number of frames and then falls below the value set by this claim. It is set to 10 by default.

- **hit_inertia_max:** Each tracked object maintains an internal hit inertia counter that tracks how frequently it is matched to a detection; when it matches, this counter rise, and when it does not, it falls. This argument specifies how large this inertia can become, and thus how long an object can exist without being matched to any detections. The default value is 25.

- **initialization_delay:** To be considered a potential object to be returned to the user by the Tracker, each tracked object must wait until its internal hit inertia counter exceeds hit inertia min. To be considered initialised and returned to the user as a real entity, the object's hit inertia counter must surpass hit inertia min by the amount specified in the argument initialization delay. This is an important parameter since it helps to understand how the tracker behaves when assigning a unique id to the fish in videos with different frame rates per second.

Features

- Norfair can be used with any detector that expresses its detections as a sequence of (x, y) coordinates. Detectors that perform object identification, pose estimation, and instance segmentation is included.

- The user defines the function that calculates the distance between tracked objects and detections, making the tracker extremely customizable.
- This feature may make use of any additional data, such as appearance embeddings, to enhance tracking performance significantly
- To add tracking to existing projects, it can be easily integrated into complex video processing pipelines. Simultaneously, a video inference loop can be built from scratch using only Norfair and a detector quickly. The detection network feeding Norfair's detections will be the only thing limiting inference speed.

Chapter 4

Experimental Methodology and Results

This chapter discusses the methods used to perform fish detection, classification, and tracking, as well as the results obtained after performing these methods. The methods and results of fish detection and classification are discussed in Section 4.1. The methods and results of fish tracking are discussed in Section 4.2.

4.1 Fish Detection and Classification

Two deep learning models YOLOv3 and Mask-RCNN are used in performing fish detection and classification tasks. The methods, how many images of each species were taken from the dataset, and how the data was divided into train and test sets are all discussed in Subsection 4.1.1. Subsection 4.1.2 discusses the initial results of each model and then goes on to discuss the challenging cases of the model, the augmentation methods applied, and the final results of each model.

4.1.1 Methods

As we saw in Section 3.1.1 about the overview of the dataset for fish detection and classification, pre-processing steps were applied to the videos to convert them into images that could be used as input to deep learning models. The following subsection discusses the count of each species that were taken into consideration as train and test set and how the training was done.

Train and Test Set

Following the completion of the pre-processing steps, the images containing fish have text files generated while labeling. However, there were no text files for the images that do not contain any fish. We wrote a python script to create text files for these types of images.

After creating text files, all of these images are organised into folders based on the species. A python script was written to extract 80% of the images randomly as the train set and 20% as the test set randomly, and then reshuffle them. Table 4.1 shows an overview of the count of each species included in the train and test sets, which is taken into account for both the YOLOv3 and Mask-RCNN models. The second column in the Table 4.1 indicates the total count of each species image which were labelled from the images which we extracted from Fishery-acoustic-observation dataset, the third column indicates the count of each species image taken into consideration for training, the fourth column indicates the number of positive samples of each species in the train set which indicates the presence of fish in the images, the fifth column indicates a number of images of each species in the test set and sixth column indicates the percentage of positive samples of each species in the test set.

Species	Total number of Images	Total Number of train images	Number of positive samples in train set	Number of test images	Number of positive samples in test set
Carp	1235	988	64.2%	247	64.77%
Lamprey	2741	2193	36.6%	548	34.4%
Lmbass	160	128	25%	32	21.8%
Pike	350	280	46.7%	70	50%
Smbass	1425	1140	70%	285	72.9%
Steelhead	582	465	44.3%	116	45.6%
Sucker	1155	924	86.4%	231	86.1%
Walleye	573	450	71.3%	114	72.8%

Table 4.1: Overview of train and test set used for training YOLOv3 and Mask-RCNN

Training of YOLOv3 Model

We used convolution weights that have been pre-trained on the ImageNet [74] dataset for training. The model was pre-trained on a large dataset with about 80 different object classes, but to use it for our purposes, we need to change some parameters in the yolov3.config configuration file [63]. yolov3.config is the file where YOLOv3 model network architecture parameters are stored.

The first change that was made in the yolov3.config file is the number of classes, which is set to 8 in our case as we have a dataset with 8 classes of objects. The *batch* value, which indicates how many images and labels are used in the forward pass to compute a gradient and update the weights via backpropagation, is set to 64. This *batch* value was selected according to the capacity of NVIDIA Tesla V100-GPU which we used for training. The *Subdivision* parameter indicates that the *batch* should be divided again into blocks of images, and it is set to 16. The *width* and *height* parameters, which indicate that every image will be resized to fit the network size during training and testing, are set to 608,608 as this is the best network size for obtaining best accuracy using YOLOv3 according to the article [71]. $Max\_batches$ parameters are set to $(numberofclasses) \times 2000$, since we have 8 classes in our dataset, it will be 16000. The steps parameter should be set to 80 and 90 percent of $Max\_batches$, so in our case, it is 12800,14400. The $filter$ parameter which indicates the number of output feature maps is calculated as $(classes + 5) \times 3$, which in our case is 39. After the dataset and config file are ready, the Google Colab Pro is utilized for training, which has an NVIDIA Tesla V100-GPU compute processor with 16GB of RAM.

Training of Mask-RCNN Model

We used convolution weights that have been pre-trained on the MS-COCO (Microsoft Common Objects in Context) [48] dataset for training. The model was pre-trained on a large dataset with about 80 different object classes, but to use it for our purposes, we need to change some parameters in the mask_rcnn.config file [17].

Initially, the changes were made on the parameter known as `Images_per_GPU` which was set to 4 according to the GPU memory. The Number_of_classes parameter was set to 9 as we have 8 classes of objects in our dataset and 1 is the background. The backbone network architecture parameter is set to Resnet-101, which is the Convolutional Network architecture that will be utilised in the first step of Mask-RCNN. Because this backbone network architecture paramter supports both Resent-50 and Resenet-101, I chose Resnet-101 because it produces the best mAp when combined with Mask-RCNN according to research paper [29]. Image_Min_Dim and Image_Max_Dim are set to 608,608. After the dataset and config file is ready then for the training purpose Google Colab Pro has been used which consisted of an NVIDIA Tesla V100-GPU computing processor with 16Gb of RAM.

4.1.2 Results

This section discusses the outcomes of both the YOLOv3 model and the Mask-RCNN model on the Fishery-acoustic-observation dataset. Following that, there is a discussion of examples where the model underperformed. Following that, we talked about the different augmentation techniques applied to improve model results. Later, we discussed the improved results of both the YOLOv3 and the Mask-RCNN models.

Intial Results of YOLOv3

Here the model performance of YOLOv3 is assessed in two different IOU values. Table 4.2 shows the initial results of the YOLOv3 model on the Fishery-acoustic-observation dataset. The first column shows the species which is present in the dataset. The second and third column shows the values of true positive and false positive of each species at IOU@ 0.5. The fifth and sixth column shows the True positive and False positive of each species at IOU@ 0.4. The fourth and last column shows the average precision of the YOLOv3 model at IOU@ 0.5 and IOU@ 0.4.

Species	TP at IOU = 0.5	FP at IOU = 0.5	AP at IOU = 0.5	TP at IOU = 0.4	FP at IOU = 0.4	AP at IOU = 0.4
Carp	81	100	0.30	108	73	0.54
Lamprey	30	37	0.16	44	23	0.36
Lmbass	0	0	0	0	0	0
Pike	9	13	0.17	15	7	0.42
Smbass	63	13	0.42	69	7	0.50
Steelhead	12	12	0.24	9	15	0.28
Sucker	73	40	0.36	81	32	0.47
Walleye	44	3	0.66	45	2	0.70

Table 4.2: Results of YOLOv3 model on Fishery-acoustic-observation dataset

Table 4.3 shows the mean average precision of the YOLOv3 model. At IOU@ 0.5, the model achieved a mean average precision of 0.29, while at IOU@ 0.4, the model achieved an mean average precision of 0.41.

We can infer from the above results that YOLOv3 performance on both IOU is not so good and that YOLOv3 mAp is higher at IOU@ 0.4 than at IOU@ 0.5. Another observation is that the walleye species has the best average precision in comparison to other species, while Lmbass has the lowest, at 0%, indicating that the more images we have for training, the better the YOLOv3 will perform in terms of detection and classification.

IOU	Mean Average Precission(mAp)
IOU@ 0.5	0.29
IOU@ 0.4	0.41

Table 4.3: Mean average precision of YOLOv3 Model on Fishery-acoustic-observation Dataset

Initial results of Mask-RCNN

Here the model performance of Mask-RCNN is assessed in two different IOU values. Table 4.4 of the YOLOv3 model on the Fishery-acoustic-observation dataset. The first column displays the species present in the dataset. The second and third columns show the true positive and false positive values for each species at IOU 0.5. The fifth and sixth columns shows each species' true positive and false positive at IOU@ 0.4. The fourth and last column show the YOLOv3 model's arithmetic precision at IOU@ 0.5 and IOU@ 0.4.

Species	TP at IOU = 0.5	FP at IOU = 0.5	AP at IOU = 0.5	TP at IOU = 0.4	FP at IOU = 0.4	AP at IOU = 0.4
Carp	114	116	0.38	142	88	0.57
Lamprey	5	58	0.01	7	56	0.06
Lmbass	1	7	0	2	4	0.25
Pike	15	30	0.31	18	27	0.47
Smbass	106	100	0.24	127	79	0.38
Steelhead	9	52	0.07	13	48	0.19
Sucker	142	214	0.17	180	177	0.30
Walleye	55	26	0.24	60	21	0.35

Table 4.4: Results of Mask-RCNN model on Fishery-acoustic-observation dataset

The below Table 4.5 shows the mean average precision of Mask RCNN model. At IOU@ 0.5, the model achieved a mean average precision of 0.18, while at IOU@ 0.4, the model achieved a mean average precision of about 0.32.

Mask-RCNN model performance is also underwhelming, with IOU@ 0.5 achieving 0.18 mAp and IOU@ 0.4 achieving 0.32 mAp. In comparision to IOU@ 0.4, this model has a higher mAp for IOU@ 0.5. Another finding is that the Lmbass species achieves the lowest average precision when compared to the other species, indicating that the Mask-RCNN model requires more training images to perform better on both classification and detection.

IOU	Mean Average Precission(mAp)
IOU@ 0.5	0.18
IOU@ 0.4	0.32

Table 4.5: Mean average precision of Mask-RCNN model on Fishery-acoustic-observation dataset

Evaluation of Difficult Cases

The overall performance of both models on the dataset is not particularly impressive. To figure out where the model's performance went wrong, we ran a little experiment to discover where the models are under-performing. Initially, 1000 images with objects in them were considered for this experiment, which were chosen randomly from the test set which included a mix of all species. As a first phase YOLOv3 and Mask-RCNN model is run on all 1000 images at IOU@ 0.5, and the images where the objects were classified and detected by the models were taken separately and images which were not classified and detected by the models were taken separately. The YOLOv3 and Mask-RCNN models are run on images gathered in the first phase of images where object were not detected at various threshold IOU values ranging from 0.2 to 0.4 in the second phase, and images which were not classified and detected by the models were gathered. As a third phase, all of the collected images from the second phase which are 804 in total were classified into three categories: edge object images, partly visible images, and clearly visible images.

Edge Object Images

Edge object images are the images where the object(fish) is present at the edge in the image which is shown in the Figure 4.1.

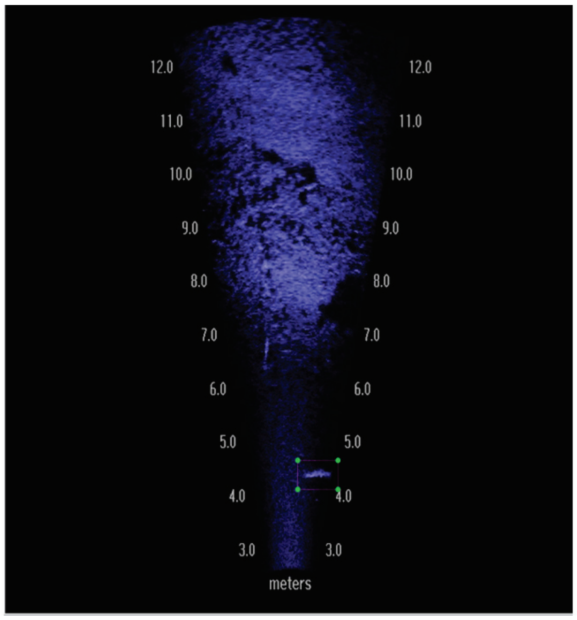


Figure 4.1: Image where the fish is present at the edge

Partially Visible Object Images

Partially visible object images are the images where the object(fish) is not so clearly visible. This is shown in Figure 4.2.

Clearly Visible Object Images

Clearly visible object images are the images where the object(fish) in the image is clearly visible which is shown in Figure 4.3.

After the categories have been divided, bounding boxes are drawn on each category of images using the ground truth and prediction coordinates made by the model at IOU@ 0.1 for each image, where the ground truth is shown as the green box and the prediction is shown as the red box, to see how the models rendered detections on these images and how much variance those predicted boxes have with respect to the ground truth bounding boxes. Figure 4.4 shows one of the case where the model is unable to draw the bounding box at the exact location of the partially visible object (fish) in the image. We could see there is a deviation from the ground truth box which is why models are not able to detect objects at different IOU values ranging from 0.2 to 0.5. Table 4.6 shown the count of a number of each category where the models are not able to make any detection.

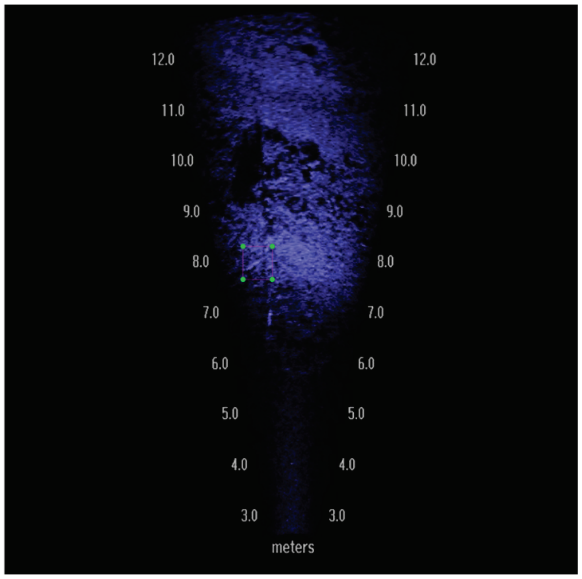


Figure 4.2: Image where the fish is partially visible.

We could see that from the Table 4.6 that, out of 804 images, models fail to detect and classify objects on 702 partially visible object images which prove that we need to train the models with more images with different color intensities making the models better to detect on this kind of images, 58 clear visible object images, and 44 images where the object is present at the image's edge proving that we need to train the model with images with different rotational angles to make models perform better on this kind of images. The following Subsection named Image Augmentation gives more details about this.

Image Augmentation

As we see in Table 4.6, we have a count of each category where the model failed. So, to make the model better on these kinds of images, we have chosen to go with different augmentation techniques which will make the model to detect and classify objects at higher IOU values. So, we have seen in the above experiment, models were failing in the majority of cases when the objects are partially visible, and the detection was not made on certain species where there are low number of training images. So, to make the model better in these kinds of situations and to increase the size of the training dataset as well as to bring variation in types of images being trained, different data augmentation techniques have been applied. By providing extra training data

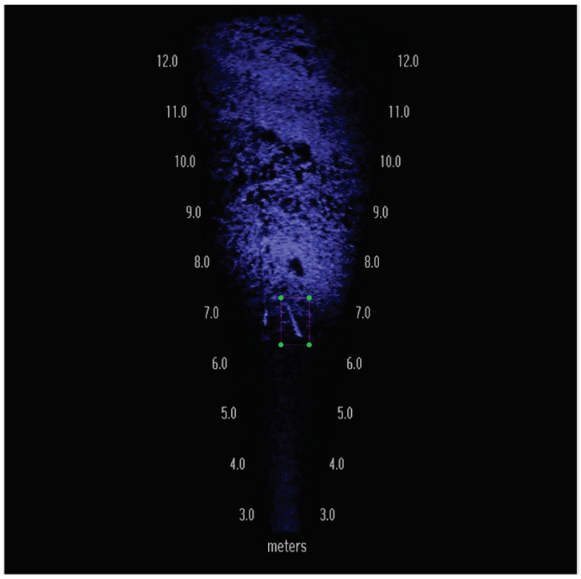


Figure 4.3: Image where the fish is clearly visible .

to the models increase model detection and classification accuracy. Saturation, hue, contrast, and rotation were different augmentation techniques applied.

Category	Count
Edge	44
partial	702
Clear	58

Table 4.6: Count of each category where the model under-performed

Saturation This modifies the color intensity. The higher the value, the greater the variance. The saturation range used is [0 to 1.5], which is applied on images during training of the model.

Exposure It determines the amount of black or white that is added to colours. The higher the value, the greater the variance, possibly making it appear as if the images were over-or under-exposed the exposure range used is [0 to 1.5], which is applied on images during training of the model.

Hue Hue can be thought of as the 'shade' of the colors in an image. Hue range used is [0 to 0.5] which is applied on images during training. The hue augmentation changes the color channels of an input image at random, causing a model to explore several color schemes for objects and scenes in the image. This strategy is important for ensuring that a model does not memorise the colors of a given object or scene.

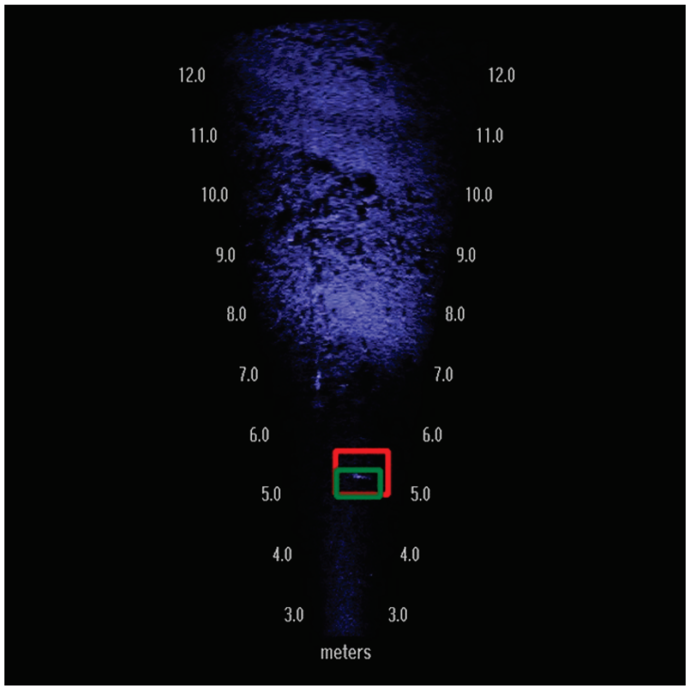


Figure 4.4: Showing the failure case of the model when the fish in the image is partially visible

Random Rotation changes the angle of objects present in the images. Objects can be skewed in either direction. The rotation range used is from [-90 to 90 degrees]

These are the data augmentation approaches that were applied to both the model configuration files and the training parameters. The modified results are discussed below.

Final Results of YOLOv3

This section discusses the results obtained by the YOLOv3 model on the Fishery-acoustic-observation dataset after applying various data augmentation techniques and retraining the model. Table 4.8 shows the results of the YOLOv3 model after applying data augmentation techniques.

Here YOLOv3 model is able to achieve 0.59 mAp at IOU@ 0.5 and 0.73 mAp at IOU@ 0.4

The value of change in mAp for different IOU values 0.4 and 0.5 before and after augmentation techniques applied is shown in Figure 4.5. The mAp of the YOLOv3 model has been increased from 0.29 to 0.59 for IOU@ 0.5 and from 0.41 to 0.73 for

Species	TP at IOU = 0.5	FP at IOU = 0.5	AP at IOU = 0.5	TP at IOU = 0.4	FP at IOU = 0.4	AP at IOU = 0.4
Carp	174	35	0.78	187	22	0.90
Lamprey	72	43	0.42	91	24	0.62
Lmbass	4	1	0.56	4	1	0.56
Pike	15	15	0.31	21	9	0.54
Smbass	168	41	0.82	184	25	0.91
Steelhead	21	13	0.39	26	8	0.57
Sucker	241	112	0.66	301	52	0.88
Walleye	67	21	0.86	70	18	0.91

Table 4.7: Improved results of YOLOv3 model on Fishery-acoustic-observation dataset after applying data augmentation techniques

IOU	Mean Average Precission(mAp)
IOU@ 0.5	0.59
IOU@ 0.4	0.73

Table 4.8: Mean average precision (mAp) of YOLOv3 model on Fishery-acoustic-observation dataset after data augmentation techniques applied

IOU@ 0.4, which is a substantial improvement. We can also see that the Walleye species achieves the highest average precision at IOU@ 0.5, while the Smbass species achieves the highest average precision at IOU@ 0.4, demonstrating that model detection performance varies with IOU. Another observation is that the number of false positive values for each species is higher for IOU@ 0.5 than for IOU@ 0.4, demonstrating that the YOLOv3 model prediction of bounding box changes as IOU value increases. Also, we can see that the average precision of Lmbass increased from 0 to 0.56 for IOU@ 0.5, which is quite impressive, and this also shows that increasing the number of training images has a positive impact on the performance of the YOLOv3 model.

Final Results of Mask-RCNN

This section discusses the results obtained by the Mask-RCNN model on the Fishery-acoustic-observation dataset after applying various data augmentation techniques and retraining the model. Table 4.9 shows the results of Mask-RCNN model after applying data augmentation techniques.

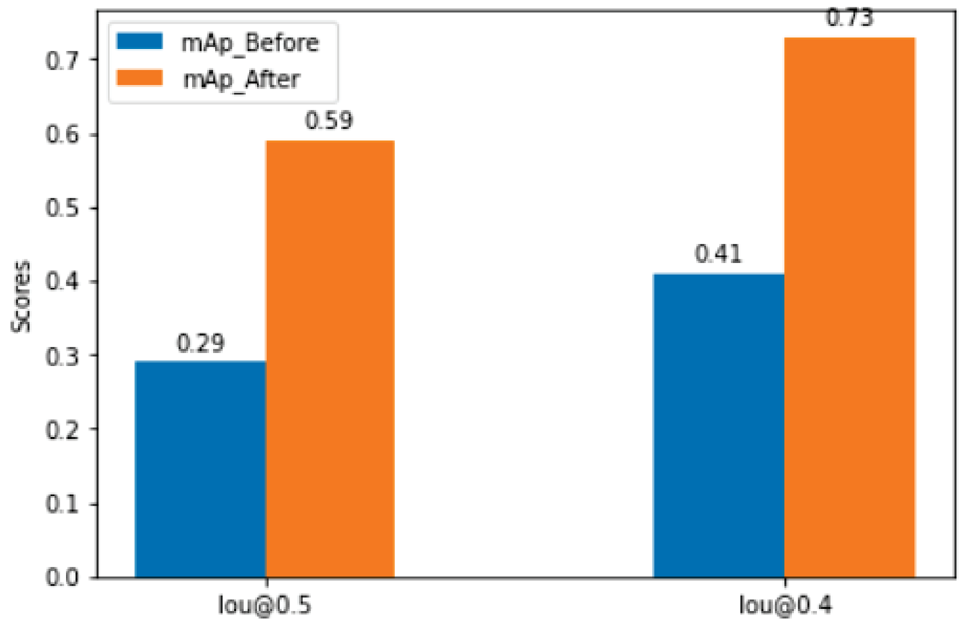


Figure 4.5: Comparison of Mean average precision of YOLOv3 model before and after data augmentation techniques.

Species	TP at IOU = 0.5	FP at IOU = 0.5	AP at IOU = 0.5	TP at IOU = 0.4	FP at IOU = 0.4	AP at IOU = 0.4
Carp	171	59	0.73	191	41	0.80
Lamprey	14	49	0.22	19	44	0.30
Lmbass	4	3	0.44	6	1	0.66
Pike	21	23	0.45	21	23	0.45
Smbass	145	60	0.68	153	55	0.70
Steelhead	19	43	0.30	26	37	0.39
Sucker	241	122	0.64	267	104	0.69
Walleye	69	10	0.82	73	7	0.85

Table 4.9: Improved results of Mask-RCNN model on Fishery-acoustic-observation dataset after applying data augmentation techniques

Table 4.10 shows the mAp of Mask-RCNN model after applying data augmentation techniques. The mAp of the Mask-RCNN model has been increased from 0.18 to 0.54 for IOU@ 0.5 and from 0.32 to 0.62 for IOU@ 0.4, which is a significant improvement. We can also see that the Walleye species achieves the highest average precision at both IOU@ 0.5 and IOU@ 0.4. Another observation is that the number of false positive values for each species is higher for IOU@ 0.5 than for IOU@ 0.4, demonstrating that the Mask-RCNN model prediction of the bounding box changes as IOU value increases. Also, we can see that the average precision of Lmbass increased from 0 to 44.44% for IOU@ 0.5, which is quite impressive, and this also shows that increasing the number of training images has a positive impact on the performance of the Mask-RCNN model as well.

IOU	Mean Average Precission(mAp)
IOU@ 0.5	0.54
IOU@ 0.4	0.62

Table 4.10: Mean average precission of Mask-RCNN model on Fishery-acoustic-observation dataset after applying data augmentation techniques

Conclusion of Results

We can see that the highest mAp achieved by both YOLOv3 and Mask-RCNN is around 0.73 and 0.62, indicating that models can detect and classify fish species using acoustic data. In comparision to Mask-RCNN, YOLOv3 achieves the maximum mAp, demonstrating that YOLOv3 is a better model to deploy in the case of acoustic data. The processing rate of each frame in videos is faster with YOLOv3(24fps) than with Mask-RCNN(8fps). Another finding from these studies is that models can achieve good average precision on species with more images such as carp, Smbass, and walleye when compared to species with fewer images such as lmbass, pike, and steelhead. This suggets that more training examples for less sampled species are needed. Using data augmentation approaches and hyper-parameter tuning, the model achieves good average precision on species with fewer images, as well as good mAp over the entire dataset.

4.2 Fish Tracking

For fish tracking, the Norfair algorithm and Yolov4 are used. Subsection 4.2.1 discusses the methods, how the dataset was divided into training and testing, and how the training was done with the YOLOv4 model. Subsection 4.2.2 discusses the model's results.

4.2.1 Methods

In Section 3.1.3, we provided an overview of dataset used for fish tracking. It contains a total of 24000 frames, with 19200 images available for training and 4800 images available for testing. The subsections that follow go into greater detail about the training process.

Train and Test Set

The main purpose of this experiment is to see how well Norfair performs with various types of camera data recorded at various frame rates. So we used Norfair on videos with varied frame rates per second, such as 20fps, 10fps, and 5fps, to have a better grasp of how well Norfair performs in fish tracking at varied frame rates. As we know, the dataset comprises 19200 images for training, all of which were retrieved at 30 frames per second. As a first training set, we down-sampled all the training images to 20 frames per second using a Python script, yielding 12800 images. Similarly, we down-sampled 20 frames per second to 10 frames per second, yielding 6400 images for training set 2. The 3200 images in the training set 3 were made by down-sampling the 10fps images into 5fps images.

The test set 1 has 4800 images retrieved at 30 frames per second, which were then down-sampled to 20 frames per second, yielding a total of 3200 images. Similarly, we down-sampled 20 frames per second images to 10 frames per second, yielding 1600 images for test set 2. The test set 3 was made by down-sampling the 10fps images to 5fps, yielding a total of 800 images.

Training

We used convolution weights that have been pre-trained on the ImageNet dataset for training. The pretrained model was trained on a large dataset that contains about 80 classes of objects, but in order to use it for our purposes, we need to change some settings in the yolov4.config configuration file [63].

The first adjustment is in the yolov4.config file is the number of classes, which is set to 1 in our case because all of the objects in our dataset belong to the same class, which is fish. The *batch* value, which indicates how many images and labels are used in the forward pass to compute a gradient and update the weights via backpropagation, is set to 64 which was selected according to the capacity of NVIDIA Tesla V100-GPU which we used for training. The *subdivision* parameter indicates that the batch should be divided again into blocks of images, and it is set to 16. The *width* and *height* parameters, which indicate that every image will be resized to fit the network size during training and testing, are set to 608,608 as this is the best network size for obtaining best accuracy using YOLOv4 according to research paper [14]. $Max\_batches$ parameters are set to $(numberofclasses) \times 2000$, so since we have 1 class in our dataset, it will be 2000. The steps parameter should be set to 80 and 90 percent of max batches, so in our case, it is 1600,1800. The filter parameter which indicates the number of output feature maps is calculated as $(classes + 5) \times 3$, which in our case is 18. Along with these settings, data augmentation parameters such as hue, saturation, exposure and random rotation were adjusted as well. After the dataset and config file is ready then for the training purpose google colab pro has been used which consisted of NVIDIA Tesla V100-GPU computing processor with 16GB of ram is used. The training was carried out for 1000 epochs and weights generated were saved.

4.2.2 Results

After training, three different weights were generated for three different training sets which are for 20fps, 10fps, and 5fps videos respectively. Now to check the performance of Norfair, we have considered a parameter is known as initialisation_delay which was discussed in Section 3.3.2. The performance of Norfair is first discussed using the default setting of initialisation delay of 17. Norfair's later results on various initialization delays are shown.

Norfair Performance on Different Initialisation Delay

There are three test sets, each with a distinct frame rate of 20fps, 10fps, and 5fps. The first 600 images were taken from the test set1 which totally has 3200 images, and a video of 30 seconds duration was made from those images using a Python script. Two videos were made from test set2 and test set3 in the same way. Norfair performance is initially checked at initialization-delay of 17, which is the default value, on each video that was generated at varying frames-per-second. The performance of Norfair is shown the Table 4.11. Here the GT is the ground truth value, MT is mostly tracked, PT is partially tracked, FP is false positive, ID'S is identity switches, FM is fragmentation and MOTA is Multiple object tracking accuracy as discussed in Subsection 2.1.4. We have checked Norfair performance on different intilaization_delay values as well. Figure 4.12 shows the performance of Norfair at intialization_delay 14. Figure 4.13 shows the performance of Norfair at intialization_delay 11. Figure 4.14 shows the performance of Norfair at intialization_delay 8. Figure 4.15 shows the performance of Norfair at initialization delay 6.

Frame rate	GT	MT	PT	ML	FP	FN	ID'S	FM	MOTA
Norfair@ 20fps	1	1	0	0	44	103	96	42	54.7%
Norfair@ 10fps	1	0	1	0	43	78	33	33	42.8%
Norfair@ 5fps	1	0	1	0	40	63	7	17	18.5%

Table 4.11: Results of Norfair at intialization delay 17

Frame rate	GT	MT	PT	ML	FP	FN	ID'S	FM	MOTA
Norfair@ 20fps	1	1	0	0	40	52	111	31	62.2%
Norfair@ 10fps	1	0	1	0	32	64	38	30	50.2%
Norfair@ 5fps	1	0	1	0	34	44	14	20	31.9%

Table 4.12: Results of Norfair at intialization delay 14

Frame rate	GT	MT	PT	ML	FP	FN	ID'S	FM	MOTA
Norfair@ 20fps	1	1	0	0	25	28	125	20	66.9%
Norfair@ 10fps	1	1	0	0	13	17	61	12	66.2%
Norfair@ 5fps	1	1	0	0	10	14	27	8	62.2%

Table 4.13: Results of Norfair at intialization delay 11

Frame rate	GT	MT	PT	ML	FP	FN	ID'S	FM	MOTA
Norfair@ 20fps	1	1	0	0	36	52	111	31	62.9%
Norfair@ 10fps	1	0	1	0	26	64	38	30	52.4%
Norfair@ 5fps	1	0	1	0	30	44	14	20	34.8%

Table 4.14: Results of Norfair at intialization delay 8

Frame rate	GT	MT	PT	ML	FP	FN	ID'S	FM	MOTA
Norfair@ 20fps	1	1	0	0	46	84	99	42	57.4%
Norfair@ 10fps	1	0	1	0	38	74	34	32	45.7%
Norfair@ 5fps	1	0	1	0	37	57	10	19	23%

Table 4.15: Results of Norfair at intialization delay 6

Discussion of Results

As we can see from the performance of Norfair at various initialization delay values, the greatest MOTA attained by Norfair is 66.9% for 20fps videos, 66.2% for 10fps videos, and 62.2% for 5fps videos at an initialization delay of 11.

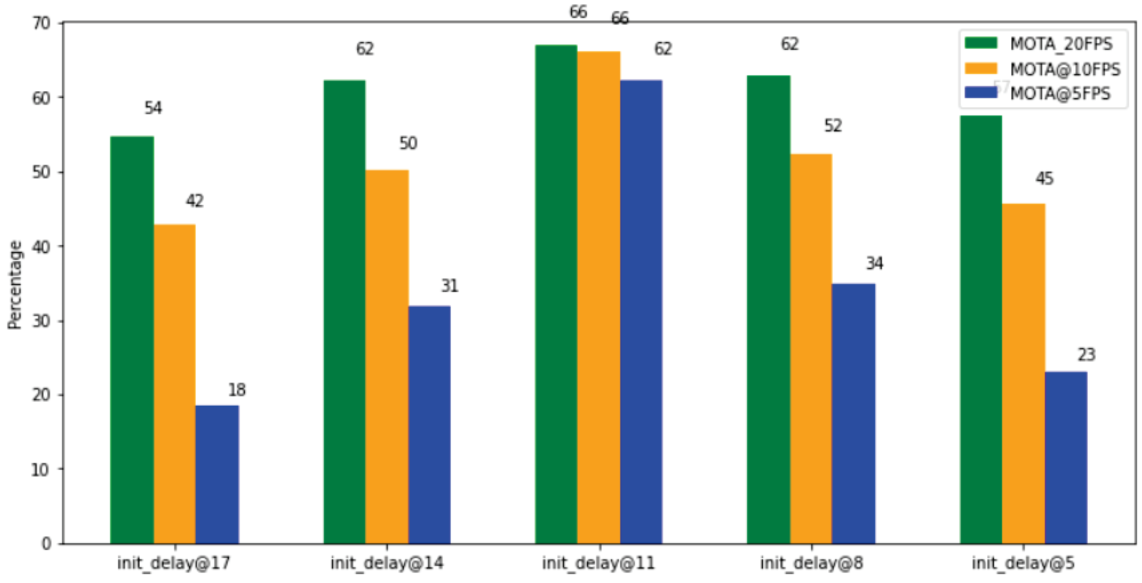


Figure 4.6: Comparison of MOTA of Norfair at different Intialization_delay values

Figure 4.6 shows a comparison of MOTA of Norfair at various frame rates. As shown in Figure 4.6, the MOTA value of Norfair attained its maximum value at initialization delay 11, and as the initialization delay is reduced, the MOTA value decreases. Even yet, we can see that MOTA values decline as fps decreases and that for all values of initialization delay, the highest MOTA is achieved for videos of 20fps and the lowest is for videos of 5fps, indicating that Norfair performs well on high-fps videos in comparison to low-fps videos. There is another metric which is known as fragmentation, and the value of fragmentation at different intilaization_delay is shown in Figure 4.7. Here we could see that the fragmentation value remains low for the

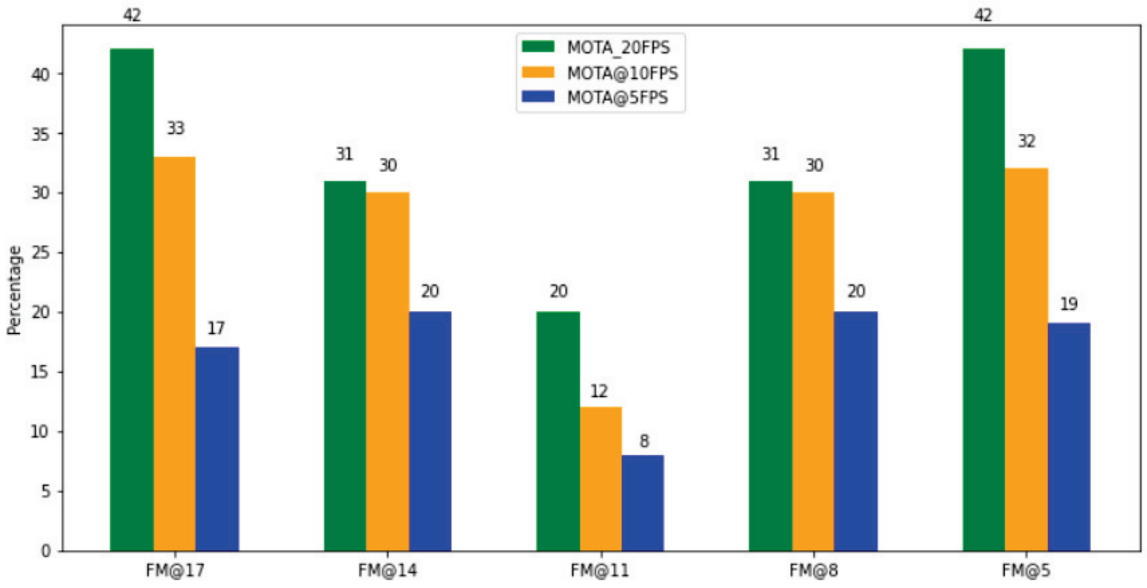


Figure 4.7: Comparison of Fragmentation of Norfair at different Intialization_delay values

intilaization_delay value 11. However, when compared to other fps, the fragmentation value for 5fps videos is low since the number of detections made in low fps videos is lower, resulting in a smaller number of fragmentation values. We can conclude from the above findings that Norfair tracks fish on videos with high frames per second(fps) and to make it work on lower frames per second it requires the parameter intialization_delay to be tuned. As we see in Table 4.13 Norfair performance is better for both higher frame rate videos as well as lower frame rate videos for the initialization_delay of 14.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

The main aim of the thesis was to see how well deep learning algorithms perform on acoustic data as well as video camera data for detection, classification, and tracking. We first tested the feasibility of using deep learning for detection and classification on a Fishery-acoustic-observation dataset of videos containing eight distinct species of fish captured by a high-resolution DIDSON imaging sonar in the ocqueoc River. The dataset contains metadata observations indicating which videos and frames contained fish, their species, and text descriptions of their approximate location which were later converted into images for performing detection and classification tasks using YOLOv3 and Mask-RCNN. For performing fish tracking, we have used the Well dam dataset which contains underwater optical videos recorded from the Wells Dam fish ladder on the Columbia River in Washington State, USA. This dataset contains images of fish which were given as an input to the YOLOv4 in integration with the Norfair algorithm for performing fish tracking.

In terms of fish classification and detection, the maximum mAp achieved by YOLOv3 in terms of fish classification and detection is 0.59 for IOU@ 0.5 and 0.73 at IOU@ 0.4 whereas Mask-RCNN was able to achieve the mAp of about 0.54 for IOU@ 0.5 and 0.62mAp for IOU@ 0.4. The results show that using acoustic data, models can detect and classify fish species. YOLOv3 achieves the highest mAp when compared to Mask-RCNN, demonstrating that YOLOv3 is a better model to deploy in real-time for performing fish detection and classification when using acoustic data. We also found that YOLOv3 was faster in terms of processing image frames per second which is 24fps in comparison to Mask-RCNN which is 8FPS which makes YOLOv3 the best model in deploying in cases where images need to be processed faster for performing detection and classification. As a note we saw that our results were not accurate in terms of classification and detection on the infrequently sampled fish dataset and we recommend at least having more than 400 images of a given species for accurate classification. So to achieve these results, some novel optimisations were applied to perform well on acoustic data which included some data augmentation techniques such as saturation, exposure, hue, and random rotation to improve performance on our imbalanced classification task.

Another task we performed after fish detection and classification is tracking of fish species. For this purpose, we have used the upgraded version of YOLOv3 which is known as YOLOv4 along with integration of Norfair tracking algorithm. As we have multiple variations of the frame rate of videos captured or available in deploying sites, this fish tracking was performed on three different frames of videos: 20fps, 10fps, and 5fps. We have tested YOLOv4 along with Norfair on each frame rate video for different Initialization_delay of 17,14,11 and 8. The greatest MOTA attained by Norfair is 66.9% for 20fps videos, 66.2% for 10fps videos, and 62.2% for 5fps videos at an Initialization_delay of 11 and the least MOTA achieved by Norfair was at Initialization_delay of 8 which is 54.7% for 20fps video, 42.8% for 10fps video and 18.5% for 5fps video. It concluded from the above findings that Norfair tracks fish on videos with high frames per second(fps) and to make it work on lower frames per second it requires the parameter initialization_delay to be tuned. The results also prove that Norfair can be a fair algorithm used along with the combination of YOLOv4 for deploying in cases where the fish tracking task needs to be done.

5.2 What are the Limitations?

We saw how well both deep learning models performed in terms of detection and classification, as well as how well Norfair tracked fish. These models, however, have some limitations, which are listed as follows.

- The first constraint is that the labeling of images by humans may lead to some inaccuracies such as wrong labeling of objects in the images and varied length of drawing bounding boxes around the images. This may lead to reducing the performance of deep learning models as training data is the most important part for the deep learning models to make perfect detections on the new images.

- The second constraint is that the acoustic images are challenging since fish are not always clearly visible in the images and there is a lot of background noise, which can be another reason for reduced performance of deep learning models while performing classification and detection tasks.

- The Norfair algorithm also has a limitation, as we can see in each frame of the video there are lot of identity switches happening and this number grows as the frame rate per video decreases. Because of the identity switches, a tracked trajectory changes its matched ground-truth identity in each frame of the video which will have a direct impact on multiple object tracking accuracy of the Norfair algorithm.

5.3 Future Work

- As we can saw, the first limitation is regarding human annotations, one way to make this better might be to automate the annotations. Using self-supervised vision transformers such as DINO(Distillation with no labels) [16] might help in doing this. The DINO can be trained with just images without any necessity of labels, can later predict the segmentation masks on the objects of images where we can extract bounding boxes from these segmentation masks and use them as an input to the YOLOv3 or YOLOv4 model. In this manner we might have the possibility of skipping the annotations stage by humans and replace with DINO to make the automation of labels possible. By achieving this we might see the increase in performance of classification and detection accuracy by the deep learning models as there might be decrease in inaccuracy of annotations. The second limitations is regarding the clarity of Didson sonar images which can be improved.

- The second limitation is about the image clarity of Didson sonar images, which have a direct impact of performance of deep learning model while preforming fish classification and detection. One way to make it better to have a more number of images while training by using different augmentation techniques on Didson sonar type of images which might increase the performance of deep learning model as this makes the deep learning models to avoid the problem of underfitting.

- Another limitation is regarding the number of identity switches by the Norfair tracking algorithm while tracking the fish in the videos. Tuning the parameters such as initialization_delay might help in reducing the identity_switches as this parameter decides when the tracked object to be considered as a potential object by allocating a unique_id to the object. The other way might help is to try with different tracking algorithms such Deepsort with the combination of YOLOv4.

To summarise, we observed that deep learning models are a viable approach for detecting and classifying fish using acoustic data. We also found that data augmentation strategies played a key role in improving the models detection and classification performance by increasing the number of training samples. Another observation is that Norfair worked well in terms of fish tracking on video camera data, and that tweaking the initialization delay parameter improved Norfair tracking performance on low frame rate videos as well.

.

Bibliography

[1] Fish habitat factsheet. https://tinyurl.com/yzadfa2z/, 2010.

[2] Ecosystems and species. https://worldoceanreview.com/en/wor-2/ecosystems-and-species/fish-habitats/, 2013.

[3] Why and how do we track the movements of fish. http://vro.agriculture.vic.gov.au/dpi/vro/vrosite.nsf/, 2021.

[4] Vaibhav Kumar Agarwal, N Sivakumaran, and VPS Naidu. Six object tracking algorithms: A comparative study. *Indian Journal of Science and Technology*, 9(30):1–9, 2016.

[5] Adamu Ali-Gombe, Eyad Elyan, and Chrisina Jayne. Fish classification in context of noisy images. In *International Conference on Engineering Applications of Neural Networks*, pages 216–226. Springer, 2017.

[6] Joaquín Alori, Alan Descoins, Braulio Ríos, and Agustín Castro. tryolabs/norfair: v0.3.1, July 2021.

[7] Hyacinth Ampadu. Yolov3 and yolov4 in object detection. https://ai-pool.com/a/s/yolov3-and-yolov4-in-object-detection//, 2021.

[8] Nancy Andrews. Ocean aware project announcement. https://oceansupercluster.ca/ocean-aware-project-announcement/, 2020.

[9] Kathuria Ayoosh. What's new in yolo v3? https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b, 2018.

[10] Faisal Bashir and Fatih Porikli. Performance evaluation of object detection and tracking systems. In *Proceedings 9th IEEE International Workshop on PETS*, pages 7–14, 2006.

[11] Akansha Bathija and Grishma Sharma. Visual object detection and tracking using yolo and sort. *International Journal of Engineering Research Technology*, 8(11), 2019.

[12] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[13] Hyunju Blemel, Andrew Bennett, Silas Hughes, Kathleen Wienhold, Thomas Flanigan, Molly Lutcavage, Chi Hin Lam, and Clayward Tam. Improved fish tagging technology: Field test results and analysis. In *OCEANS 2019-Marseille*, pages 1–6. IEEE, 2019.

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

[15] Fabrizio Capoccioni, Chiara Leone, Domitilla Pulcini, Massimo Cecchetti, Alessandro Rossi, and Eleonora Ciccotti. Fish movements and schooling behavior across the tidal channel in a mediterranean coastal lagoon: An automated approach using acoustic imaging. *Fisheries Research*, 219:105318, 2019.

[16] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *arXiv preprint arXiv:2104.14294*, 2021.

[17] cclauss. Mask$_r cnn.https://github.com/matterport/Mask_R CNN/$, 2019.

[18] B Chitradevi and P Srimathi. An overview on image processing techniques. *International Journal of Innovative Research in Computer and Communication Engineering*, 2(11):6466–6472, 2014.

[19] Trinh Cindy. A tour of video object tracking — part iii: Multiple object tracking. https://medium.com/@cindy.trinh.sridykhan/a-tour-of-video-object-tracking-part-iii-multiple-object-tracking-5e3a15ae0a7c, 2019.

[20] Liu Danqing. A practical guide to relu). https://medium.com/@danqing//, November 2017.

[21] databricks. Neural network. https://databricks.com/glossary/neural-network/, 2021.

[22] deepansh. Beginners guide to artificial neural network. https://www.analyticsvidhya.com/blog/2021/05/beginners-guide-to-artificial-neural-network/, May 2021.

[23] Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, Javen Shi, Daniel Cremers, Ian Reid, Stefan Roth, Konrad Schindler, and Laura Leal-Taixe. Cvpr19 tracking and detection challenge: How crowded can it get? *arXiv preprint arXiv:1906.04567*, 2019.

[24] Shah Deval. The surveillance phenomenon you must know about : Multi object tracking. https://medium.com/visionwizard/object-tracking-675d7a33e687, 2020.

[25] Arthur Douillard. Densely connected convolutional networks. https://arthurdouillard.com/post/densenet//, 2018.

[26] Hofesmann Eric. Iou a better detection evaluation metric. https://towardsdatascience.com/iou-a-better-detection-evaluation-metric-45a511185be1/, 2020.

[27] Ahmed Gad. Evaluating object detection models using mean average precision. https://www.kdnuggets.com/2021/03/evaluating-object-detection-models-using-mean-average-precision.html/, 2021.

[28] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[29] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

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

[32] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *European conference on computer vision*, pages 749–765. Springer, 2016.

[33] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[34] jonathan hui. map (mean average precision) for object detection. https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173/, 2018.

[35] Ahsan Jalal, Ahmad Salman, Ajmal Mian, Mark Shortis, and Faisal Shafait. Fish detection and species classification in underwater environments using deep learning with temporal information. *Ecological Informatics*, 57:101088, 2020.

[36] Brownlee Jason. A gentle introduction to the rectified linear unit (relu). https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks//, January 2019.

[37] Niels Jepsen, Eva B Thorstad, Torgeir Havn, and Martyn C Lucas. The use of external electronic tags on fish: an evaluation of tag retention and tagging effects. *Animal Biotelemetry*, 3(1):1–23, 2015.

[38] He Kaiming. Mask r-cnn. https://web.cs.ucdavis.edu/ yjlee/teaching/ecs289g-winter2018/Mask$_{R}CNN.pdf$/, 2018.

[39] Shivani Kapania, Dharmender Saini, Sachin Goyal, Narina Thakur, Rachna Jain, and Preeti Nagrath. Multi object tracking with uavs using deep sort and yolov3 retinanet detection framework. In *Proceedings of the 1st ACM Workshop on Autonomous and Intelligent Mobile Systems*, pages 1–6, 2020.

[40] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[41] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[42] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[43] Leal-Taixé Laura. Cv3dst - one-stage object detectors. https://www.youtube.com/watch?v=J9LSeOGoNW0t=32s, 2020.

[44] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[45] Liling Li, Tyler Danner, Jesse Eickholt, Erin McCann, Kevin Pangle, and Nicholas Johnson. A distributed pipeline for didson data processing. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 4301–4306. IEEE, 2017.

[46] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[47] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[48] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[49] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8759–8768, 2018.

[50] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[51] Shishira Maiya. Deepsort: Deep learning to track custom objects in a video. https://nanonets.com/blog/object-tracking-deepsort/, 2019.

[52] François Martignac, Aurélie Daroux, Jean-Luc Bagliniere, Dominique Ombredane, and Jean Guillard. The use of acoustic cameras in shallow waters: new hydroacoustic tools for monitoring migratory fish population. a review of didson technology. *Fish and fisheries*, 16(3):486–510, 2015.

[53] Erin McCann, Liling Li, Kevin Pangle, Nicholas Johnson, and Jesse Eickholt. An underwater observation dataset for fish classification and fishery assessment. *Scientific data*, 5(1):1–8, 2018.

[54] Bython Media. Waht is computer vision. https://www.youtube.com/watch?v=LTlZy-OVWkM/, 2021.

[55] Ilija Mihajlovic. Everything you ever wanted to know about computer vision. https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e/, 2019.

[56] Miracle. Panet: Path aggregation network in yolov4]. https://medium.com/clique-org/panet-path-aggregation-network-in-yolov4-b1a6dd09d158//, 2020.

[57] Russell A Moursund, Thomas J Carlson, and Rock D Peters. A fisheries application of a dual-frequency identification sonar acoustic camera. *ICES Journal of Marine Science*, 60(3):678–683, 2003.

[58] Guanghan Ning, Zhi Zhang, Chen Huang, Xiaobo Ren, Haohong Wang, Canhui Cai, and Zhihai He. Spatially supervised recurrent convolutional neural networks for visual object tracking. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4. IEEE, 2017.

[59] nrcan. Image classification and analysis. https://www.nrcan.gc.ca/maps-tools-publications/satellite-imagery-air-photos/remote-sensing-tutorials/image-interpretation-analysis/image-classification-and-analysis/9361/, 2013.

[60] Rafael Padilla, Sergio L Netto, and Eduardo AB da Silva. A survey on performance metrics for object-detection algorithms. In *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 237–242. IEEE, 2020.

[61] Thitinun Pengying, Marius Pedersen, Jon Yngve Hardeberg, and Jon Museth. Underwater fish classification of trout and grayling. In *2019 15th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, pages 268–273. IEEE, 2019.

[62] Van Hiep Phung and Eun Joo Rhee. A deep learning approach for classification of cloud image patches on small datasets. *Journal of information and communication convergence engineering*, 16(3):173–178, 2018.

[63] pjreddie. darknet. https://github.com/pjreddie/darknet/blob/master/cfg/yolov3.cfg/, 2018.

[64] Raghav prabhu. Understanding of convolutional neural network (cnn) — deep learning. https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148/, 2018.

[65] Ganesh Prakash. Object detection : Simplified. https://towardsdatascience.com/object-detection-simplified-e07aa3830954, 2019.

[66] rafael. Metrics for object detection. https://github.com/rafaelpadilla/Object-Detection-Metrics/, 2021.

[67] Dhruv Rathi, Sushant Jain, and S Indu. Underwater fish species classification using convolutional neural network and deep learning. In *2017 Ninth international conference on advances in pattern recognition (ICAPR)*, pages 1–6. IEEE, 2017.

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

[69] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[70] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[71] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.

[72] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.

[73] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[74] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[75] Rahmad Sadli. The beginner's guide to implementing yolov3 in tensorflow 2.0 (part-1). https://machinelearningspace.com/yolov3-tensorflow-2-part-1/, 2019.

[76] Kansal Sahadev. A quick guide to activation functions in deep learning. https://towardsdatascience.com/a-quick-guide-to-activation-functions-in-deep-learning-4042e7addd5b/, 2020.

[77] Pulkith Sharma. Introduction to image segmentation techniques (part 1). https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/, 2019.

[78] Tanuj Shrivastava. Cnn (convolution neural network). https://medium.com/analytics-vidhya/cnn-convolution-neural-network-17cc89802234, 2020.

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

[80] Concetto Spampinato, Daniela Giordano, Roberto Di Salvo, Yun-Heh Jessica Chen-Burger, Robert Bob Fisher, and Gayathri Nadarajan. Automatic fish classification for underwater species behavior understanding. In *Proceedings of the first ACM international workshop on Analysis and retrieval of tracked events and motion in imagery streams*, pages 45–50, 2010.

[81] Concetto Spampinato, Simone Palazzo, Daniela Giordano, Isaak Kavasidis, Fang-Pang Lin, and Yun-Te Lin. Covariance based fish tracking in real-life underwater environment. In *VISAPP (2)*, pages 409–414, 2012.

[82] Minsung Sung, Son-Cheol Yu, and Yogesh Girdhar. Vision based real-time fish detection using convolutional neural network. In *OCEANS 2017-Aberdeen*, pages 1–6. IEEE, 2017.

[83] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[84] Sanchit Tanwar. Review: Spatial pyramid pooling[1406.4729]. https://medium.com/analytics-vidhya/review-spatial-pyramid-pooling-1406-4729-bfc142988dd2//, 2020.

[85] Kei Terayama, Kento Shin, Katsunori Mizuno, and Koji Tsuda. Integration of sonar and optical camera images using deep neural network for fish monitoring. *Aquacultural Engineering*, 86:102000, 2019.

[86] Chi-Hsuan Tseng and Yan-Fu Kuo. Detecting and counting harvested fish and identifying fish types in electronic monitoring system videos using deep convolutional neural networks. *ICES Journal of Marine Science*, 77(4):1367–1378, 2020.

[87] Michal Tušer, Jaroslava Frouzová, Helge Balk, Milan Muška, Tomáš Mrkvička, and Jan Kubečka. Evaluation of potential bias in observing fish with a didson acoustic camera. *Fisheries Research*, 155:114–121, 2014.

[88] Tzutalin. labelimg. https://github.com/tzutalin/labelImg, 2015.

[89] Sébastien Villon, David Mouillot, Marc Chaumont, Emily S Darling, Gérard Subsol, Thomas Claverie, and Sébastien Villéger. A deep learning method for accurate and fast identification of coral reef fishes in underwater images. *Ecological informatics*, 48:238–244, 2018.

[90] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. Cspnet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 390–391, 2020.

[91] wikipedia. Acoustic camera. https://en.wikipedia.org/wiki/Acoustic$_camera, 2021$.

[92] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[93] Wenwei Xu and Shari Matzner. Underwater fish detection using deep learning for water power applications. In *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 313–318. IEEE, 2018.

[94] Ethan Yanjia. Dive really deep into yolo v3: A beginner's guide. https://towardsdatascience.com/dive-really-deep-into-yolo-v3-a-beginners-guide-9e3d2666280e/, 2019.

[95] Pengfei Zhu, Longyin Wen, Dawei Du, Xiao Bian, Haibin Ling, Qinghua Hu, Qinqin Nie, Hao Cheng, Chenfeng Liu, Xiaoyu Liu, et al. Visdrone-det2018: The vision meets drone object detection in image challenge results. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.