

Intelligent Ship Detection and Classification on Remote Sensing Images using Deep Learning

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology in Electronics and Communication Engineering

by

**ANANYA K R
17BEC0857**

Under the guidance of

Internal Guide

Dr. Valarmathi J

Professor

School of Electronics Engineering

VIT, Vellore.

External Guide

Dr. R Rajesh

Scientist 'F'

Centre for Airborne Systems

DRDO, Bangalore.



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

May, 2021

DECLARATION

I hereby declare that the thesis entitled “Intelligent Ship Detection and Classification on Remote Sensing Images using Deep Learning” submitted by me, for the award of the degree of *Bachelor of Technology in Electronics and Communication Engineering* to VIT is a record of bonafide work carried out by me under the supervision of Dr. Valarmathi J, School of Electronics Engineering, VIT, Vellore and Dr. Rajesh R, Scientist ‘F’, Centre for Airborne Systems, Defense Research and Development Organisation, Bangalore.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore

Date: 25th May 2021

Ananya K R

CERTIFICATE

This is to certify that the thesis entitled “Intelligent Ship Detection and Classification on Remote Sensing Images using Deep Learning” submitted by **Ananya K R (17BEC0857)** School of Electronics Engineering, VIT, Vellore for the award of the degree of *Bachelor of Technology in Electronics and Communication Engineering*, is a record of bonafide work carried out by her under my supervision during the period, 01.12.2020 to 30.05.2021, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place: Vellore

Signature of the Guide

Date: 25th May 2021

Internal Examiner

External Examiner

Head of the Department

SENSE



भारत सरकार - रक्षा मंत्रालय
Government of India - Ministry of Defence
रक्षा अनुसंधान तथा विकास संगठन
Defence Research & Development Organisation
वायुवाहित प्रणाली केंद्र
CENTRE FOR AIRBORNE SYSTEMS
एस 9100 डी & आई एस ओ 9001:2015 प्रमाणित
संस्थान
AS 9100D & ISO 9001:2015 Certified Establishment
बेलूर, येमलूर तपाल, बेंगलूरु - 560037, भारत
Belur, Yemlur Post, Bengaluru - 560 037, INDIA

CERTIFICATE


This is to certify that Ms. Ananya K R (17BEC0857), a student of Vellore Institute of Technology, Vellore has completed her capstone project at Centre for Airborne Systems (CABS), Defence Research and Development Organisation (DRDO), Bangalore, for a period of six months from December 2020 to May 2021.

Her project title was “Intelligent Ship Detection and Classification on Remote Sensing Images using Deep Learning”, and was guided by Dr. R Rajesh, Scientist – F, CABS, DRDO, Bangalore.

During the period of the internship, she was found to be regular and hard-working.

Place: Bangalore

Date: 01/06/21


Project Guide's Signature
Dr R RAJESH
SC 'F'

ACKNOWLEDGEMENTS

This report is a result of a collective effort of a lot of precious people who have directly or indirectly supported me over the years.

First of all, I would like to express my sincere gratitude to my guide, Dr. Valarmathi J, Professor, SENSE, for guiding, motivating and helping me overcome any obstacle that came along the way. Her guidance and knowledge added with the exquisite help from Dr. R. Rajesh and Shri. Dhipu T.M., scientists at the Centre for Airborne Systems, DRDO, Bangalore, who kept me motivated throughout this project. Adding to this, their constant guidance made this research topic all the more enthralling and invigorating.

I would like to thank the Almighty for blessing us all with this opportunity to learn and give something back to the society. I am extremely grateful and would like to pay sincere and deepest gratitude to our Honorable Chancellor, Dr. G. Viswanathan for providing all the students of this University all the required facilities and literally a place to learn and grow beyond boundaries. We are extremely privileged to be here under his guidance.

I owe my profound respect to the Dean, School of Electronics Engineering, Dr. Harish Kittur Mallikarjun for his continuous support and enlightenment. My sincere thanks also go to Dr. P. Prakasam, our Head of Department for believing in me and providing us with great opportunities to gain maximum knowledge and expand our horizons.

Last but not the least, I would also like to thank my family for being a constant pillar of support spiritually and providing me with everything I needed and my friends for keeping me going strong whenever I needed.

Ananya K R

EXECUTIVE SUMMARY

Intelligent detection and classification of ships from high-resolution remote sensing images is an extraordinarily useful task in civil and military surveys. However, this job comes with a lot of hassles since various disturbances are present in the sea such as clouds, mist, islands, coastlines, ripples, and so on. Moreover, two primary setbacks in this regard are cluttered image scenes and varying ship sizes. This project utilises machine learning and neural network techniques clubbed with few methodologies to carry out ship detection from images with ease. The advantages and disadvantages of different detection and classification techniques of ship detection are highlighted as well. Machine learning techniques such as Region Based Convolutional Neural Networks (R-CNNs), Mask R-CNNs, You Only Look Once (YOLO), and Convolutional Neural Network (CNN) Image Classifiers are applied on numerous datasets such as Dat-Tran's Raccoon dataset, Kaggle's Airbus dataset, and MARVEL Maritime dataset. It was observed that the YOLO model clubbed with a CNN Classifier gave the best results for ship detection and classification.

CONTENTS

1. INTRODUCTION.....	1
1.1 Objective.....	1
1.2 Motivation.....	1
1.3 Background.....	2
2. PROJECT DESCRIPTION AND GOALS.....	5
2.1 Object Localisation.....	5
2.1.1 R-CNN.....	6
2.1.2 Mask R-CNN.....	7
2.1.3 YOLO.....	8
2.2 Object Classification.....	10
2.2.1 Components of CNN Classifier.....	10
2.2.1.1 Convolution.....	11
2.2.1.2 The Rectified Linear Unit (ReLU).....	11
2.2.1.3 Pooling.....	11
2.2.1.4 Flattening.....	12
2.2.1.5 Full Connection.....	12
3. TECHNICAL SPECIFICATION.....	14
3.1 Laptop Specifications.....	14
3.2 Google Colaboratory.....	15
3.3 Advantages of using Google Colaboratory.....	16
4. DESIGN APPROACH AND DETAILS.....	18
4.1 Phase 1: Implementation of R-CNN.....	18
4.1.1 Objective.....	18
4.1.2 Dataset.....	18
4.1.3 Methodology and Code Snippets.....	19
4.1.4 Results.....	22
4.1.5 Key Takeaways.....	24
4.2 Phase 2: Implementation of Mask R-CNN.....	24
4.2.1 Objective.....	24
4.2.2 Dataset.....	24
4.2.3 Methodology and Code Snippets.....	25
4.2.4 Results.....	28
4.2.5 Key Takeaways.....	31

4.3 Phase 3: Implementation of YOLO.....	32
4.3.1 Objective.....	32
4.3.2 Dataset.....	33
4.3.3 Methodology and Code Snippets.....	33
4.3.4 Results.....	37
4.3.5 Key Takeaways.....	40
4.4 Phase 4: Implementation of CNN Classifier.....	40
4.4.1 Objective.....	40
4.4.2 Dataset.....	41
4.4.3 Methodology and Code Snippets.....	41
4.4.4 Results.....	46
4.4.5 Key Takeaways.....	47
5. PROJECT TIMELINE.....	48
6. SUMMARY.....	48
7. REFERENCES.....	49

LIST OF FIGURES

Figure No.	Title	Page No.
1	An overview of recent object detection performance	2
2	Object detection	3
3	Bounding box representation used for object localization	5
4	Evolution of Deep Learning Object Detection over the years	6
5	The three stages of a typical R-CNN model	7
6	The Mask R-CNN framework for Instance Segmentation	8
7	YOLO Detection Algorithm	9
8	YOLO Algorithm Architecture	10
9	The Convolution Function	11
10	The SoftMax Function	12
11	Google Colaboratory Logo	15
12	A snippet of the Colab Interface	15
13	Easily transferable to Drive or GitHub	16
14	The pre-installed libraries that come with Colab	16
15	A snippet of Dat-Tran's Raccoon Dataset	18
16	The Object Detection Workflow	19
17	IOU Equation	19
18	Ground-truth and predicted bounding boxes of an image	20
19	Code snippet of the IOU function	21
20	A summary of NMS	22
21	Results obtained upon execution	22
22	Training Loss and Accuracy of the model	23
23	A few output images of the R-CNN model	23
24	(a) Before NMS was applied, (b) After NMS was applied	24
25	Graph depicting the 80:20 split	25
26	A Code Snippet of the Mask Generation Function	26
27	A few examples of masks generated for ship images	26
28	A Code Snippet with the Encode-Decode Mask Function	27
29	Encoded and Decoded Ship Masks	28

30	Training the Mask RCNN Model	28
31	Training Loss vs Epoch for Mask R-CNN	29
32	A few snippets of the Mask R-CNN results	31
33	Drawback of using Mask R-CNN	32
34	The Marvel Dataset	33

Figure No.	Title	Page No.
35	The LabelMe Annotation Tool Interface	34
36	The output from the LabelMe Annotation Tool	34
37	The RoboFlow Process	36
38	Results using the YOLOv4 model	37
39	The mAP formula	37
40	YOLO Loss and accuracy vs epoch graph	38
41	YOLO results	40
42	Steps followed on the RoboFlow interface	43
43	File organisation	44
44	A snippet of the .csv file with one-hot encoded data	44
45	The Classifier Model	45
46	Training the CNN Classifier	45
47	Accuracy for each class in a confusion matrix	46
48	Confusion Matrix of the CNN Classifier	46
49	Project Timeline	48

LIST OF TABLES

Table No.	Title	Page No.
1	Summary of related object detection surveys since 2000	3

1. INTRODUCTION

1.1 Objective

There is a growing need for effective ship recognition and detection for ensuring maritime security and civil management. Apart from this, there are a wide range of applications for ship detection as well right from traffic monitoring, and the defence of territory and naval battles, to harbour surveillance, fishery management, and sea pollution management [1].

Under the guidance of Defence Research and Development Organisation (DRDO), Bangalore, this project aims to develop an effective methodology using machine learning algorithms and models such as R-CNNs, Mask R-CNNs, YOLO, and CNN Classifiers to meet two primary objectives -

(A) **Object Detection:** to draw a bounding box around the most feasible position of the object/ship in images.

(B) **Classification:** to classify the detected maritime object/ship into one of the classes, namely, cargo, cruise, carrier, and so on.

1.2 Motivation

The ubiquitous and wide applications like scene understanding, video surveillance, robotics, and self-driving systems triggered vast research in the domain of computer vision in the most recent decade. Being the core of all these applications, visual recognition systems which encompasses image classification, localization and detection have achieved great research momentum. Due to significant development in neural networks especially deep learning, these visual recognition systems have attained remarkable performance. Object detection is one of these domains witnessing great success in computer vision.

Moreover, in recent years, with the continuous enhancement of hardware computing power, deep learning algorithms have been rapidly developed and applied in the field of object detection. Deep learning-based methods are widely used to detect common objects in daily life and achieve extremely high performance.

Even though a lot of object detection algorithms have surfaced over the past decade, classifying and detecting maritime objects such as ships still remains a challenge owing to the cluttered image scenes and varying ship sizes. This project aims to mainly use and implement deep learning techniques involving convolutional neural networks to develop a model that

can effectively reach the required objective. The beauty of convolutional neural networks is that they do not rely on manually created feature extractors or filters. Rather, they train per se from raw pixel level up to final object categories. Also, deep neural architectures handle complex models efficiently than shallow networks.

1.3 Background

As a longstanding, fundamental and challenging problem in computer vision, object detection, has been an active area of research for several decades [4]. The goal of object detection is to determine whether there are any instances of objects from given categories (such as humans, cars, bicycles, dogs or cats) in an image and, if present, to return the spatial location and extent of each object instance (e.g., via a bounding box [5]; [6]). As the cornerstone of image understanding and computer vision, object detection forms the basis for solving complex or high-level vision tasks such as segmentation, scene understanding, object tracking, image captioning, event detection, and activity recognition. Object detection supports a wide range of applications, including robot vision, consumer electronics, security, autonomous driving, human computer interaction, content-based image retrieval, intelligent video surveillance, and augmented reality. Recently, deep learning techniques ([7]; [8]) have emerged as powerful methods for learning feature representations automatically from data. In particular, these techniques have provided major improvements in object detection, as illustrated in Fig. 1, where VOC2012 stands for The Pascal Visual Object Classes Challenge conducted in 2012, and ILSVRC stands for ImageNet Large Scale Visual Recognition Challenge.

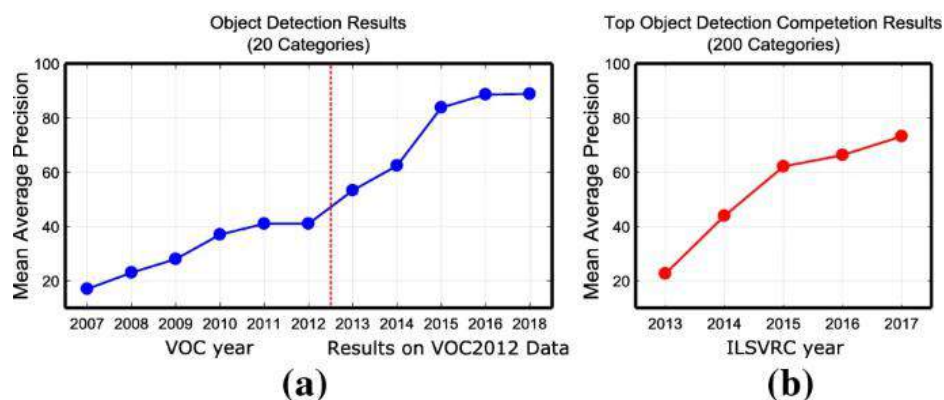


Fig. 1 - An overview of recent object detection performance: a significant improvement can be observed in performance (measured as mean average precision) since the arrival of deep learning in 2012. **(a)** Detection results of winning entries in the VOC2007-2012 competitions, and **(b)** top object detection competition results in ILSVRC2013-2017 [7]

As illustrated in Fig. 2, object detection can be grouped into one of two types ([9]; [10]): detection of specific instances versus the detection of broad categories. The first type aims to

detect instances of a particular object (such as Donald Trump's face, the Eiffel Tower, or a neighbour's dog), essentially a matching problem. The goal of the second type is to detect (usually previously unseen) instances of some predefined object categories (for example humans, cars, bicycles, and dogs). Historically, much of the effort in the field of object detection has focused on the detection of a single category (typically faces and pedestrians) or a few specific categories. In contrast, over the past several years, the research community has started moving towards the more challenging goal of building general purpose object detection systems where the breadth of object detection ability rivals that of humans.

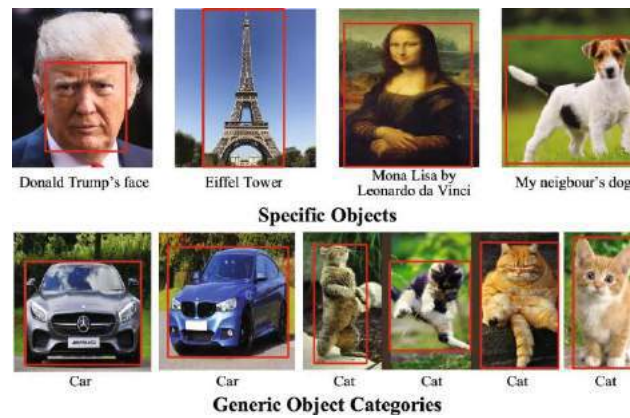


Fig. 2 - Object detection includes localizing instances of a *particular* object (top), as well as generalizing to detecting object *categories* in general (bottom). [12]

A brief insight into the evolution of deep learning mechanisms and techniques to carry out object detection since the beginning of the year 2000, has been summarised in Table 1. Every paper has been linked in the References section as well.

Table 1 - Summary of related object detection surveys since 2000

Year	Paper	Content
2009	Monocular pedestrian detection: survey and experiments [11]	An evaluation of three pedestrian detectors
2010	Survey of pedestrian detection for advanced driver assistance systems [12]	A survey of pedestrian detection for advanced driver assistance systems
2012	Pedestrian detection: an evaluation of the state of the art [13]	A thorough and detailed evaluation of detectors in monocular images
2002	Detecting faces in images: a survey [14]	First survey of face detection from a single image
2015	A survey on face detection in the wild: past, present and future [15]	A survey of face detection in the wild since 2000
2006	On road vehicle detection: a review [16]	A review of vision based on-road vehicle detection systems
2015	Text detection and recognition in imagery: a survey [17]	A survey of text detection and recognition in colour imagery
2007	Toward category level object recognition [18]	Representative papers on object categorization, detection, and segmentation

Year	Paper	Content
2009	The evolution of object categorization and the challenge of image abstraction [19]	A trace of the evolution of object categorization over 4 decades
2010	Context based object categorization: a critical survey [20]	A review of contextual information for object categorization
2013	50 years of object recognition: directions forward [21]	A review of the evolution of object recognition systems over 5 decades
2011	Visual object recognition [22]	Instance and category object recognition techniques
2013	Object class detection: a survey [23]	Survey of generic object detection methods before 2011
2015	Feature representation for statistical learning-based object detection: a review [24]	Feature representation methods in statistical learning-based object detection, including handcrafted and deep learning-based features
2014	Salient object detection: a survey [25]	A survey for salient object detection
2013	Representation learning: a review and new perspectives [26]	Unsupervised feature learning and deep learning, probabilistic models, auto-encoders, manifold learning, and deep networks
2015	Deep learning [27]	An introduction to deep learning and applications
2017	A survey on deep learning in medical image analysis [28]	A survey of deep learning for image classification, object detection, segmentation and registration in medical image analysis
2017	Recent advances in convolutional neural networks [29]	A broad survey of the recent advances in CNN and its applications in computer vision, speech and natural language processing
2019	Deep learning for generic object detection [30]	A comprehensive survey of deep learning for generic object detection

2. PROJECT DESCRIPTION AND GOALS

Object detection is used to determine where objects are located in a given image, i.e., object localization, and which category each object belongs to, known as object classification.

2.1 Object Localisation

Humans can easily detect and identify objects present in an image. The human visual system is fast and accurate and can perform complex tasks like identifying multiple objects and detect obstacles with little conscious thought. With the availability of large amounts of data, faster GPUs, and better algorithms, computers can easily be trained to detect and classify multiple objects within an image with high accuracy.

An image classification or image recognition model simply detect the probability of an object in an image. In contrast to this, object localization refers to identifying the location of an object in the image. An object localization algorithm will output the coordinates of the location of an object with respect to the image. In computer vision, the most popular way to localize an object in an image is to represent its location with the help of bounding boxes. Fig. 3 shows an example of a bounding box.



Fig. 3 - Bounding box representation used for object localization

A bounding box can be initialized using the following parameters:

- a. bx, by : coordinates of the centre of the bounding box
- b. bw : width of the bounding box w.r.t the image width
- c. bh : height of the bounding box w.r.t the image height

A brief summary of the various machine learning algorithms used to carry out object localisation/bounding-box representation over the years can be presented in the flowchart in Fig. 4.

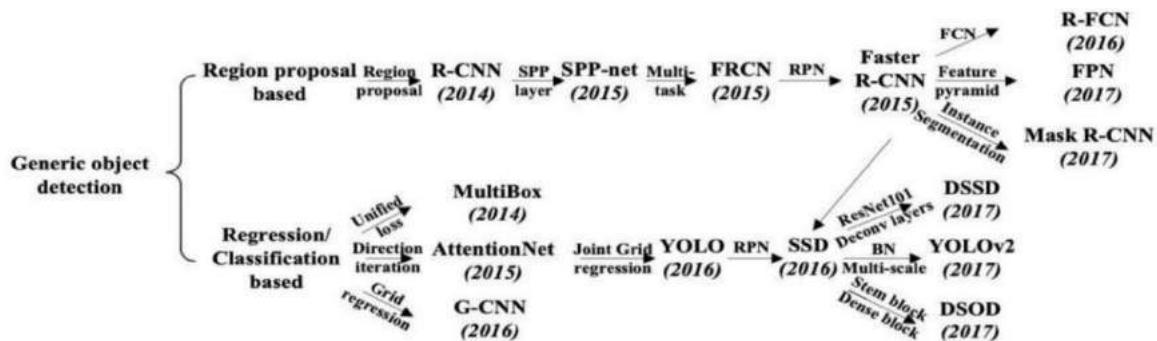


Fig. 4 – Evolution of Deep Learning Object Detection/Localisation techniques over the years [2]

In this project, the primary focus of interest is object localisation or bounding box detection using three major and proved to be widely successful machine learning algorithms as seen in the following sections.

2.1.1 R-CNN

R-CNN or Regional Convolutional Neural Networks were majorly adopted when there were two pressing requirements persistent in the field of object localisation - (a) to improve the quality of candidate boundary boxes, and (b) to use a deep architecture for extraction of high-level features. To solve these requirements, R-CNN [31] was proposed by Ross Girshick in 2014 and obtained a mean average precision (mAP) of 53.3% with more than 30% improvement over the previous best result.

Fig. 5 shows the flowchart of R-CNN, which can be divided into three stages as follows:

- I. **Region proposal generation:** The R-CNN adopts selective search to generate about 2k region proposals for each image. The selective search method relies on simple bottom-up grouping and saliency cues to provide more accurate candidate boxes of arbitrary sizes quickly and to reduce the searching space in object detection.
- II. **CNN based deep feature extraction:** In this stage, each region proposal is warped or cropped into a fixed resolution and the CNN module in is utilized to extract a 4096-dimensional feature as the final representation. Due to large learning capacity, dominant expressive power and hierarchical structure of CNNs, a high-level, semantic and robust feature representation for each region proposal can be obtained.

III. **Classification and localization:** With pre-trained category specific linear SVMs for multiple classes, different region proposals are scored on a set of positive regions and background (negative) regions. The scored regions are then adjusted with bounding box regression and filtered with a greedy non-maximum suppression (NMS) to produce final bounding boxes for preserved object locations.

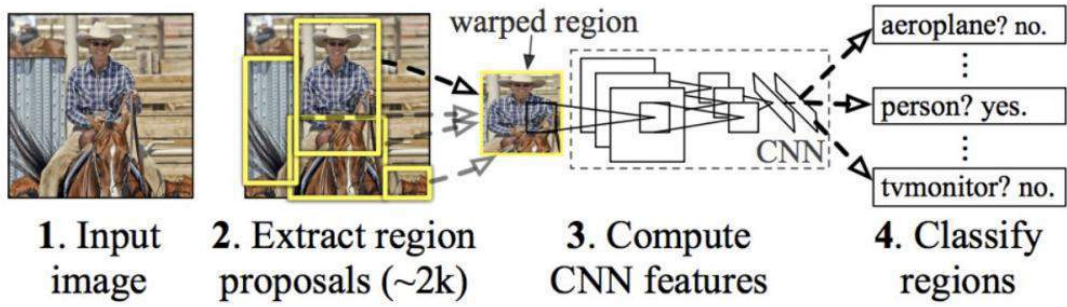


Fig. 5 – The three stages of a typical R-CNN model [9]

The advantages and disadvantages of the R-CNN model have been discussed in the later part of the report, after its practical implementation on Dan-Tran's Raccoon Dataset in section 4.1.

2.1.2 Mask R-CNN

Instance segmentation is a challenging task that requires detecting all objects in an image and segmenting each instance (semantic segmentation). These two tasks are usually regarded as two independent processes. The most viable option before the discovery of Mask R-CNNs, i.e., the multi-task scheme had a disadvantage that would create spurious edge and exhibit systematic errors on overlapping instances.

To solve this problem, parallel to the already existing branches in Faster R-CNN for classification and bounding box regression, the Mask R-CNN adds an additional branch to predict segmentation masks in a pixel-to-pixel manner as seen in Fig. 6. Different from the other two branches which are inevitably collapsed into short output vectors by FC layers, the segmentation mask branch encodes an $m \times m$ mask to maintain the explicit object spatial layout. This kind of fully convolutional representation requires fewer parameters but is more accurate than that of traditional R-CNN.

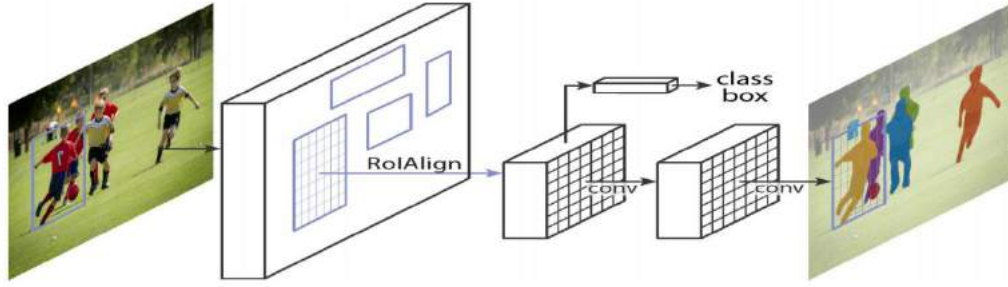


Fig. 6 – The Mask R-CNN framework for Instance Segmentation [13]

Formally, besides the two losses for classification and bounding box regression, an additional loss for segmentation mask branch is defined to reach a multi-task loss. This loss is only associated with ground-truth class and relies on the classification branch to predict the category. Because Region of Interest (RoI) pooling, the core operation in Faster R-CNN, performs a coarse spatial quantization for feature extraction, misalignment is introduced between the RoI and the features. It affects classification little because of its robustness to small translations. However, it has a large negative effect on pixel-to-pixel mask prediction. To solve this problem, Mask R-CNN adopts a simple and quantization-free layer, namely RoIAlign, to preserve the explicit per-pixel spatial correspondence faithfully. RoIAlign is achieved by replacing the harsh quantization of RoI pooling with bilinear interpolation, computing the exact values of the input features at four regularly sampled locations in each RoI bin.

In spite of its simplicity, this seemingly minor change improves mask accuracy greatly, especially under strict localization metrics. Given the Faster R-CNN framework, the mask branch only adds a small computational burden and its cooperation with other tasks provides complementary information for object detection. As a result, Mask R-CNN is simple to implement with promising instance segmentation and object detection results. In a nutshell, Mask R-CNN is a flexible and efficient framework for instance-level recognition, which can be easily generalized to other tasks (e.g., human pose estimation) with minimal modification.

The advantages and disadvantages of the Mask R-CNN model have further been discussed in the later part of the report, after its practical implementation on the Airbus Kaggle Dataset in section 4.2.

2.1.3 YOLO

YOLO (You Only Look Once) uses a totally different approach. YOLO is a clever convolutional neural network (CNN) for doing object detection in real-time. The algorithm

applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.

YOLO is popular because it achieves high accuracy while also being able to run in real-time. The algorithm “only looks once” at the image in the sense that it requires only one forward propagation pass through the neural network to make predictions. After non-max suppression (which makes sure the object detection algorithm only detects each object once), it then outputs recognized objects together with the bounding boxes.

With YOLO, a single CNN simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This model has a number of benefits over other object detection methods:

- YOLO is extremely fast
- YOLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance.
- YOLO learns generalizable representations of objects so that when trained on natural images and tested on artwork, the algorithm outperforms other top detection methods.

The major concept of YOLO is to build a CNN network to predict a $(7, 7, 30)$ tensor. It uses a CNN network to reduce the spatial dimension to 7×7 with 1024 output channels at each location. YOLO performs a linear regression using two fully connected layers to make $7 \times 7 \times 2$ boundary box predictions (the middle picture in Figure 9). To make a final prediction, we keep those with high box confidence scores (greater than 0.25) as our final predictions (the right picture in Fig. 7).

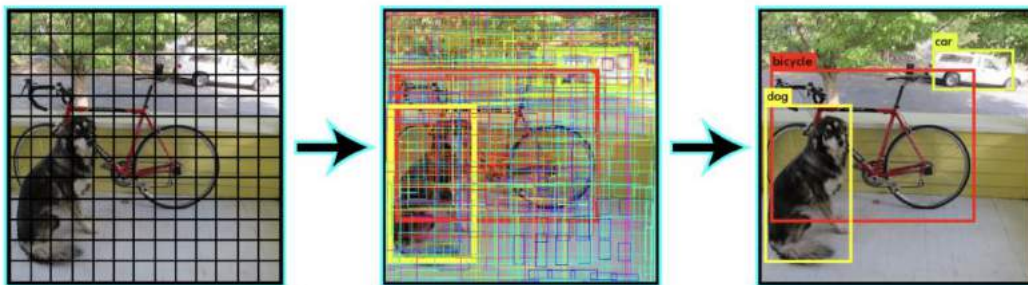


Fig. 7 – YOLO Detection Algorithm

The class confidence score for each prediction box is computed as a product of the box confidence score and its conditional class probability. YOLO measures the confidence on

both the classification and the localization (where an object is located). Moreover, YOLO has 24 convolutional layers followed by 2 fully connected layers (FC). Some convolution layers use 1×1 reduction layers alternatively to reduce the depth of the feature's maps. For the last convolution layer, it outputs a tensor with shape (7, 7, 1024). The tensor is then flattened. Using 2 fully connected layers as a form of linear regression, it outputs $7 \times 7 \times 30$ parameters and then reshapes to (7, 7, 30), i.e., 2 boundary box predictions per location. This can be visualised in Fig. 8.

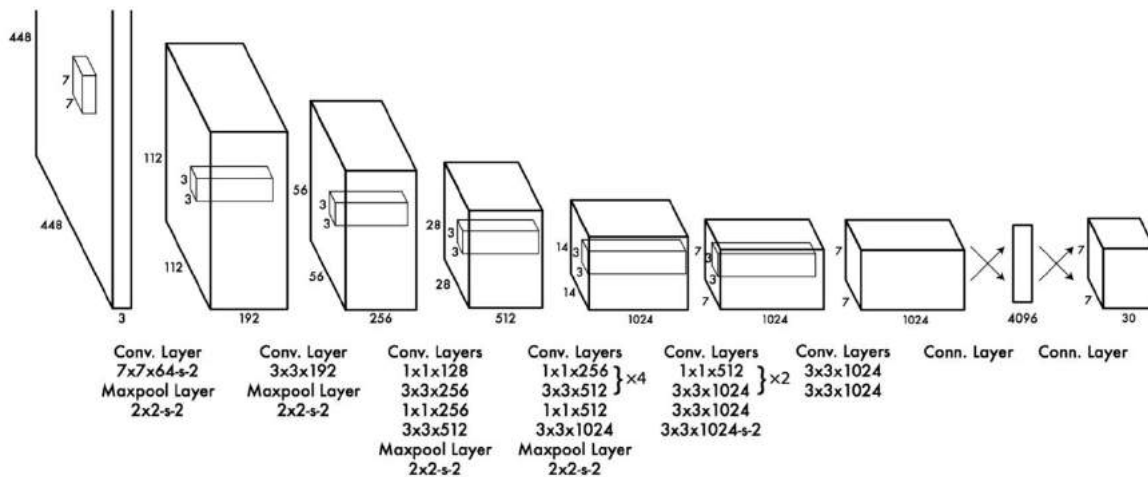


Fig. 8 – YOLO Algorithm Architecture [34]

A deeper insight into the advantages and disadvantages of the YOLO model have been discussed in the later part of the report, after its practical implementation on the MARVEL Maritime Dataset in section 4.3.

2.2 Object Classification

Apart from implementing object localisation, this project also uses a convolutional neural network for object classification. A Convolutional Neural Network (CNN) is a multi-layered neural network with a special architecture to detect complex features in data. CNNs have been used in image recognition, powering vision in robots, and for self-driving vehicles.

2.2.1 Components of CNN Classifier

Images are made up of pixels. Each pixel is represented by a number between 0 and 255. Therefore, each image has a digital representation which is how computers are able to work with images.

2.2.1.1 Convolution

A convolution is a combined integration of two functions that shows us how one function modifies the other.

$$\begin{aligned}(f * g)(t) &= \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau \\ &= \int_{-\infty}^{\infty} f(t - \tau)g(\tau)d\tau\end{aligned}\dots\dots\dots (1)$$

Fig. 9 – The Convolution Function

There are three important items to mention in this process: the input image, the feature detector, and the feature map. The input image is the image being detected. The feature detector is a matrix, usually 3x3 (it could also be 7x7). A **feature detector** is also referred to as a kernel or a filter.

Intuitively, the matrix representation of the input image is multiplied element-wise with the feature detector to produce a feature map, also known as a convolved feature or an activation map. The aim of this step is to reduce the size of the image and make processing faster and easier. Some of the features of the image are lost in this step.

However, the main features of the image that are important in image detection are retained. These features are the ones that are unique to identifying that specific object. For example, each animal has unique features that enable to identify it. The way the loss of image information is reduced by having many feature maps. Each feature map detects the location of certain features in the image.

2.2.1.2 The Rectified Linear Unit (ReLU)

The rectifier function is used to increase non-linearity in the CNN. Images are made of different objects that are not linear to each other. Without applying this function, the image classification will be treated as a linear problem while it is actually a non-linear one.

2.2.1.3 Pooling

Spatial invariance is a concept where the location of an object in an image doesn't affect the ability of the neural network to detect its specific features. Pooling enables the CNN to detect

features in various images irrespective of the difference in lighting in the pictures and different angles of the images.

There are different types of pooling, for example, max pooling and min pooling. Max pooling works by placing a matrix of 2x2 on the feature map and picking the largest value in that box. The 2x2 matrix is moved from left to right through the entire feature map picking the largest value in each pass.

These values then form a new matrix called a pooled feature map. Max pooling works to preserve the main features while also reducing the size of the image. This helps reduce overfitting, which would occur if the CNN is given too much information, especially if that information is not relevant in classifying the image.

2.2.1.4 Flattening

Once the pooled featured map is obtained, the next step is to flatten it. Flattening involves transforming the entire pooled feature map matrix into a single column which is then fed to the neural network for processing.

2.2.1.5 Full Connection

After flattening, the flattened feature map is passed through a neural network. This step is made up of the input layer, the fully connected layer, and the output layer. The fully connected layer is similar to the hidden layer in ANNs but in this case it's fully connected. The output layer is where we get the predicted classes. The information is passed through the network and the error of prediction is calculated. The error is then backpropagated through the system to improve the prediction.

The final figures produced by the neural network don't usually add up to one. However, it is important that these figures are brought down to numbers between zero and one, which represent the probability of each class. This is the role of the SoftMax function.

$$\sigma : \mathbb{R}^K \rightarrow (0, 1)^K$$

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \quad \text{for } j = 1, \dots, K. \quad \dots\dots\dots (2)$$

Fig. 10 – The SoftMax Function

A deeper insight into the advantages and disadvantages of the CNN Classifier have been discussed in the later part of the report, after its practical implementation on the MARVEL Maritime Dataset in section 4.4.

3. TECHNICAL SPECIFICATION

3.1 Laptop Specifications

Mac Book Pro 2017

Display

- Retina display
- 13.3-inch (diagonal) LED-backlit display with IPS technology; 2560-by-1600 native resolution at 227 pixels per inch with support for millions of colors
- Supported scaled resolutions:
 - 1680 by 1050
 - 1440 by 900
 - 1024 by 640
- 500 nit's brightness
- Wide colour (P3)

Processor

- 2.3GHz dual-core Intel Core i5, Turbo Boost up to 3.6GHz, with 64MB of eDRAM
Configurable to 2.5GHz dual-core Intel Core i7, Turbo Boost up to 4.0GHz, with 64MB of eDRAM

Storage

- 128GB
128GB SSD
Configurable to 256GB, 512GB, or 1TB SSD

Memory

- 8GB of 2133MHz LPDDR3 onboard memory
Configurable to 16GB of memory

Graphics

- Intel Iris Plus Graphics 640

Operating System

- macOS is the operating system that powers everything one does on a Mac. macOS Mojave brings new features inspired by its most powerful users but designed for everyone.

3.2 Google Colaboratory

Google Colaboratory has been used primarily to implement multiple ML Models throughout the project duration. Colaboratory, or “**Colab**” for short, is a product from Google Research. **Colab** allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.



Fig. 11 – Google Colaboratory Logo

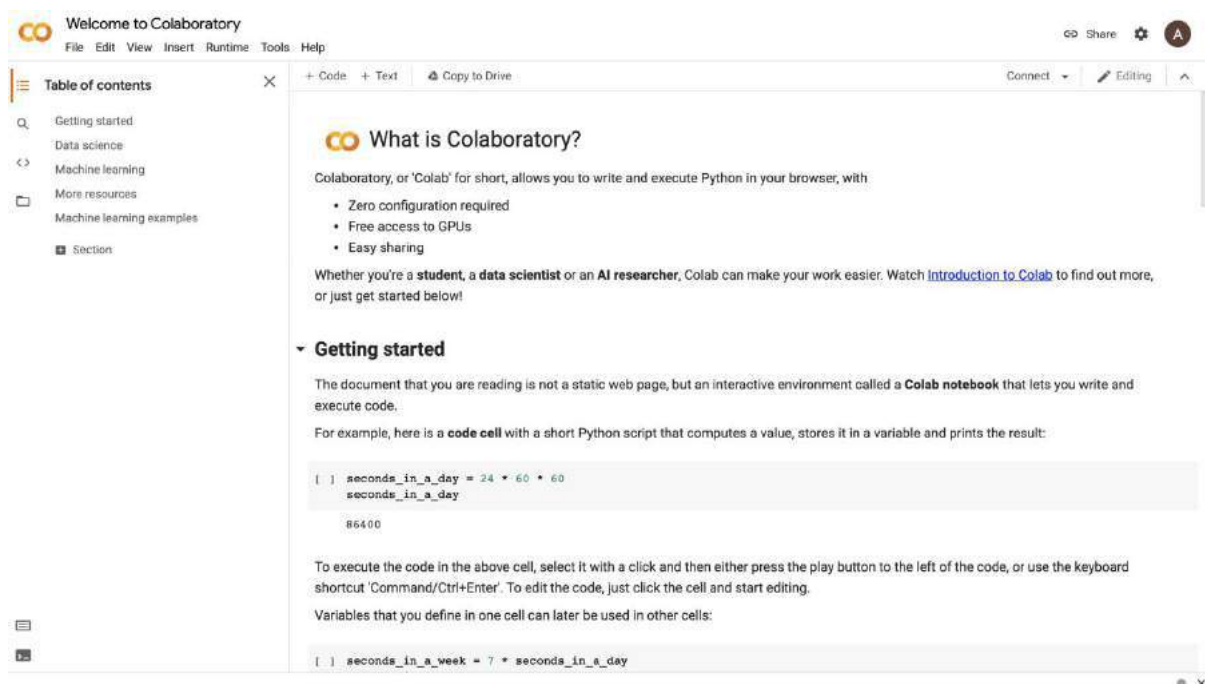


Fig. 12 – A snippet of the Colab Interface

3.3 Advantages of using Google Colaboratory

I. Easy to Share across multiple platforms

One can share his/her Google Colab notebooks very easily. Thanks to Google Colab everyone with a Google account can just copy the notebook on his/her own Google Drive account. There is no need to install any modules to run any code, modules come preinstalled within Google Colab.

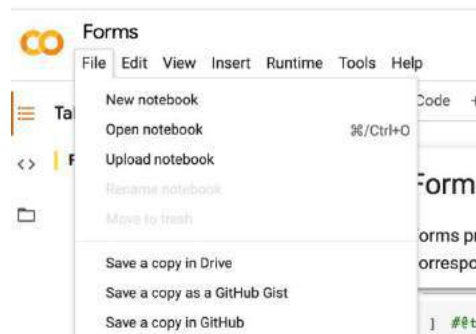


Fig. 13 – Easily transferable to Drive or GitHub

II. Pre-Installed Libraries

One of the main benefits of using Colab is that it has most of the common libraries that are needed for machine learning like *TensorFlow*, *Keras*, *ScikitLearn*, *OpenCV*, *numpy*, *pandas*, etc. pre-installed. Having all of these dependencies means that one can just open a notebook and start coding without having to set up anything at all. Any libraries that are not pre-installed can also be installed using standard terminal commands. While the syntax for executing terminal commands remains the same, one must add an exclamation mark (!) at the start of the command so that the compiler can identify it as a terminal command.



Fig. 14 – The pre-installed libraries that come with Colab

III. Cloud-Based Interface

Another feature is that the *Colab* environment is independent of the computing power of the local computer itself. Since it is a cloud-based system, as long as one has internet connectivity, he/she can run even heavy machine learning operations from a relatively old computer that, ordinarily, wouldn't be able to handle the load of executing those operations locally. Additionally, Google also offers a **GPU (*Graphics Processing Unit*)** and a **TPU (*Tensor Processing Unit*)** for free. These hardware accelerators can enable users to run heavy machine learning operations on large datasets much faster than any local environment.

4. DESIGN APPROACH AND DETAILS

The project has been broken down into four phases, with respect to the algorithms and dataset chosen, namely the implementation of R-CNN, Mask R-CNN, YOLOv4 and CNN Classifier. The details regarding the objectives, datasets, methodologies, code snippets and results have been outlined in the following sections.

4.1 Phase 1: Implementation of R-CNN

4.1.1 Objective

R-CNNs serve as a great starting point to solve any major object detection problem and to brush up skills of TensorFlow and Keras libraries, due to its well-maintained documentation and varied use cases.

4.1.2 Dataset

The dataset chosen to reach the objective was the Dat-Tran's 2017 Raccoon Dataset, that consists of 196 images of raccoons in varied sizes, shapes and orientation. Owing to more than a single raccoon present in a few images, the total number of bounding boxes in the ground-truth dataset equals to 213. As it can be observed, this is a single class problem, and a great starting point to perform and explore object detection. The images in this dataset vary from 0.04 to 2.67 megapixels and the median image size can be approximated at 480 x 360, i.e., 0.18 megapixels. A small snippet of the dataset can be seen in Fig. 15.



Fig. 15 – A snippet of Dat-Tran's Raccoon Dataset

4.1.3 Methodology and Code Snippets

Implementing an R-CNN object detector is a somewhat complex multistep process. A summary of the flow that has been followed has been summarised in the flowchart, as shown in Fig. 16.

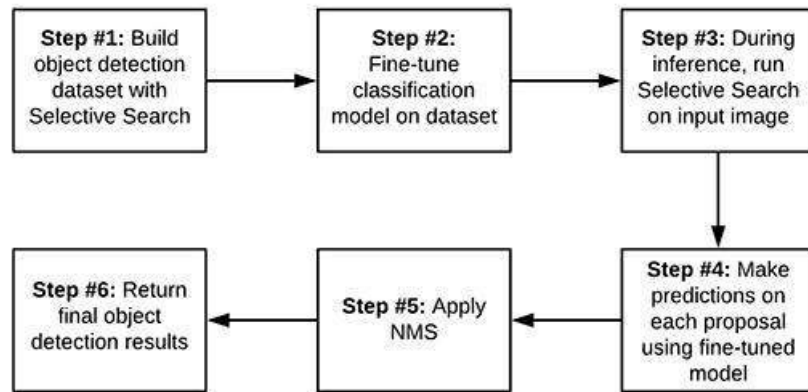


Fig. 16 – The Object Detection Workflow

Firstly, Selective Search was implemented, along with a bit of post-processing logic, to identify regions of an input image that *do* and *do not* contain a potential object of interest, in this case, a raccoon. These regions were used as the training data for the object detection model.

Next, a MobileNet (pre-trained on ImageNet) was fine-tuned to classify and recognise objects from the previously curated dataset.

Finally, the last step involves implementing a Python script for inference and prediction by applying Selective Search to an input image, classifying the region proposals generates, and then to display the final output of the R-CNN.

The accuracy of the object detection algorithm was measured by using a metric called “Intersection over Union (IOU)”, that can be calculated using the formula in Fig. 17.

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

The diagram shows two overlapping blue rectangles. The top rectangle is slightly offset to the right and up from the bottom rectangle. The area where they overlap is shaded a darker blue. The entire area covered by both rectangles is also shaded blue. This visualizes the 'Area of Overlap' and 'Area of Union' used in the IOU formula.

Fig. 17 – IOU Equation

Examining the equation in Fig. 17 and observing Fig. 18, it can be seen that Intersection over Union (IOU) is simply a ratio:

- In the numerator, the **area of overlap** between the predicted bounding box and the ground-truth bounding box is computed.
- The denominator is the **area of union**, or more simply, the area encompassed by *both* the predicted bounding box and the ground-truth bounding box.
- Dividing the area of overlap by the area of union yields the final score — ***the Intersection over Union*** (hence the name).

In this project, IOU is used to measure object detection accuracy, including how much a given Selective Search proposal overlaps with a ground-truth bounding box (which is useful when generating positive and negative examples for the training data).

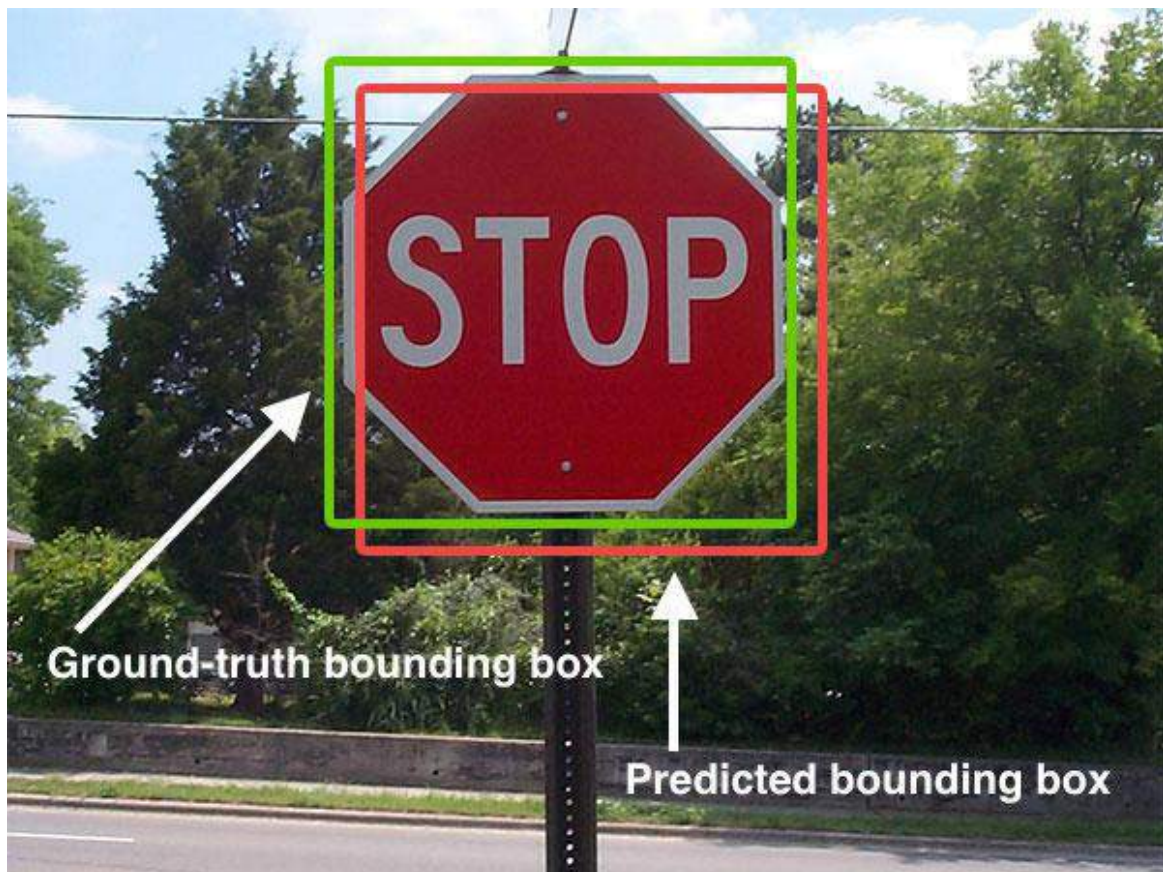


Fig. 18 – Ground-truth and predicted bounding boxes of an image

The Function defined in Python for detecting IOU for any image in an object detection problem has been elaborated, in Fig.19.


```

def compute_iou(boxA, boxB):
    # determine the (x, y)-coordinates of the intersection rectangle
    xA = max(boxA[0], boxB[0])
    yA = max(boxA[1], boxB[1])
    xB = min(boxA[2], boxB[2])
    yB = min(boxA[3], boxB[3])

    # compute the area of intersection rectangle
    interArea = max(0, xB - xA + 1) * max(0, yB - yA + 1)

    # compute the area of both the prediction and ground-truth
    # rectangles
    boxAArea = (boxA[2] - boxA[0] + 1) * (boxA[3] - boxA[1] + 1)
    boxBArea = (boxB[2] - boxB[0] + 1) * (boxB[3] - boxB[1] + 1)

    # compute the intersection over union by taking the intersection
    # area and dividing it by the sum of prediction + ground-truth
    # areas - the intersection area
    iou = interArea / float(boxAArea + boxBArea - interArea)

    # return the intersection over union value
    return iou

```

Fig. 19 – Code snippet of the IOU function

Another important concept that was implemented during the course of this project is the “Non-Maxima Suppression (NMS).” Any typical object detection pipeline has one component for generating proposals for classification. Proposals are nothing but the candidate regions for the object of interest. Most of the approaches employ a sliding window over the feature map and assigns foreground/background scores depending on the features computed in that window. The neighbourhood windows have similar scores to some extent and are considered as candidate regions. This leads to hundreds of proposals. As the proposal generation method should have high recall, it is often advised to keep loose constraints in this stage. However, processing these many proposals all through the classification network is cumbersome. This leads to a technique which filters the proposals based on some criteria called Non-maximum Suppression. This phenomenon can be observed in Fig. 20.

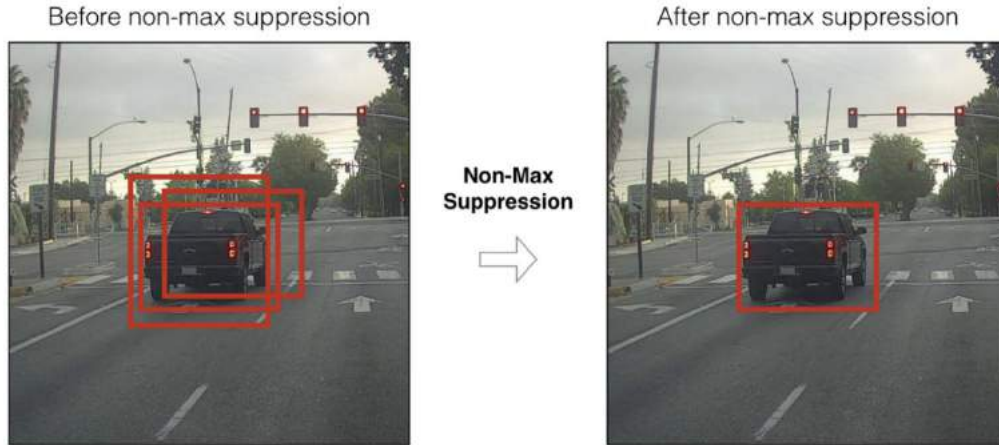


Fig. 20 – A summary of NMS

In the Results section, more about the effects of NMS on this particular project have been outlined.

4.1.4 Results

```
[INFO] loading images...
[INFO] compiling model...
[INFO] training head...
Train for 94 steps, validate on 752 samples
Train for 94 steps, validate on 752 samples
Epoch 1/5
94/94 [=====] - 77s 817ms/step - loss: 0.3072 - accuracy: 0.8647 -
val_loss: 0.1015 - val_accuracy: 0.9728
Epoch 2/5
94/94 [=====] - 74s 789ms/step - loss: 0.1083 - accuracy: 0.9641 -
val_loss: 0.0534 - val_accuracy: 0.9837
Epoch 3/5
94/94 [=====] - 71s 756ms/step - loss: 0.0774 - accuracy: 0.9784 -
val_loss: 0.0433 - val_accuracy: 0.9864
Epoch 4/5
94/94 [=====] - 74s 784ms/step - loss: 0.0624 - accuracy: 0.9781 -
val_loss: 0.0367 - val_accuracy: 0.9878
Epoch 5/5
94/94 [=====] - 74s 791ms/step - loss: 0.0590 - accuracy: 0.9801 -
val_loss: 0.0340 - val_accuracy: 0.9891
[INFO] evaluating network...
      precision    recall  f1-score   support

 no_raccoon       1.00      0.98      0.99        440
  raccoon         0.97      1.00      0.99        312

 accuracy
 macro avg       0.99      0.99      0.99        752
 weighted avg    0.99      0.99      0.99        752

[INFO] saving mask detector model...
[INFO] saving label encoder...

real    6m37.851s
user    31m43.701s
sys      33m53.058s
```

Fig. 21 – Results obtained upon execution

Upon training and testing the model for 5 epochs, results that have been highlighted in Figure 21 were obtained. A graph depicting the training loss and accuracy obtained during every epoch, the model was trained for, has been depicted in Figure 22. Once the accuracy and the loss ceased to change, as seen in the graph, the training of the model was stopped.

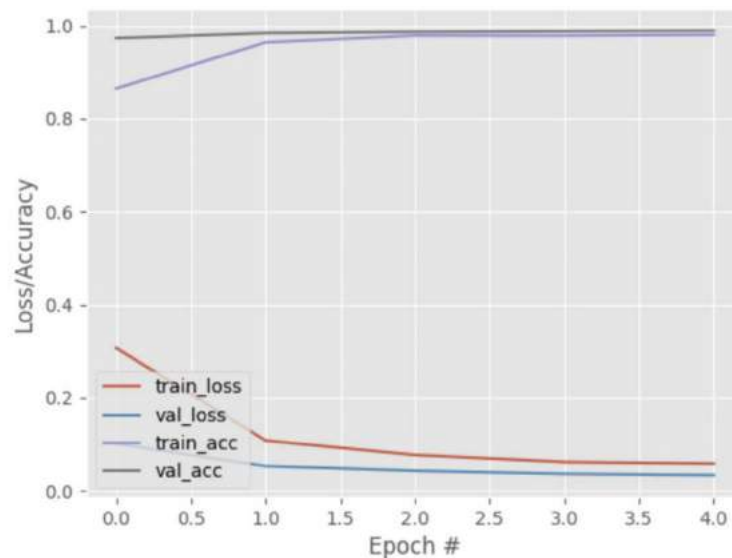


Fig. 22 – Training Loss and Accuracy of the model

Fig. 23 and Fig. 24 show snippets of a few output images with the bounding box as well as the IOU labelled on it.

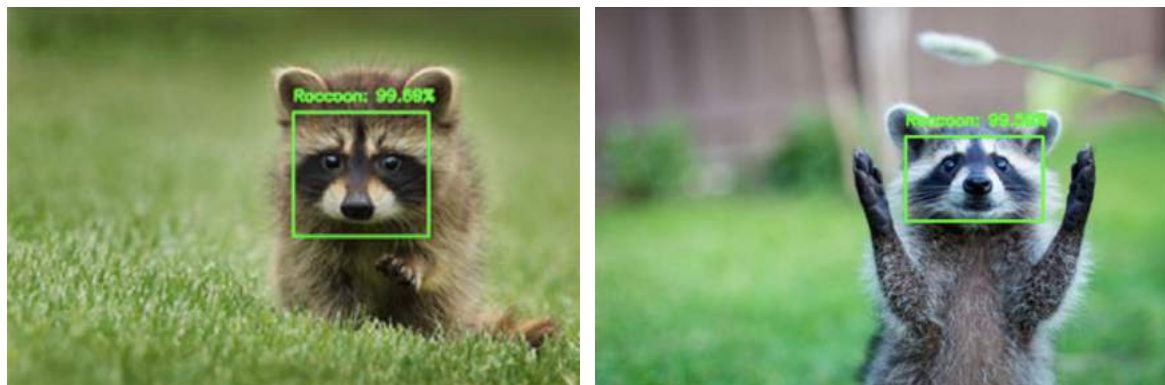


Fig. 23 – A few output images of the R-CNN model

A key point to note is the importance of Non-Maxima Suppression (NMS). As seen in Fig. 20, NMS plays an invaluable role to eliminate lesser accurate bounding boxes during object detection. A before-NMS and an after-NMS comparison has been made in Fig. 24.



(a) (b)
Fig. 24 - (a) Before NMS was applied, (b) After NMS was applied

As it can be seen, the redundant bounding boxes were eliminated upon applying NMS.

4.1.5 Key Takeaways

1. The Raccoon project was a good starting point to get accustomed to TensorFlow, Keras and the object detection pipeline.
2. Even though the accuracy of the model obtained was great and the results shown were promising, it can mostly be attributed to the smaller size of the dataset, with easily distinguishable objects in it.
3. To further the project and to reach closer to the objective of the project, the next step is to apply a better algorithm on databases that involve air-borne objects such as helicopters, aeroplanes as well as the ones that involve ships.

4.2 Phase 2: Implementation of Mask R-CNN

4.2.1 Objective

Upon the successful implementation of R-CNNs, the decision to take one further step in the direction of solving the problem statement of ship detection led to the usage of the moderately-sized dataset of ships satellite images, in contrast to the small-size dataset of Raccoons used previously. The Mask R-CNN algorithm was chosen due to its proven ability to tackle the problem of object detection such as ships and other maritime objects.

4.2.2 Dataset

For this phase, a public dataset from Kaggle on the Airbus Ship Detection Challenge was obtained. The dataset contains more than 100 thousand 768×768 images taken from satellite. The total size of the dataset is more than 30 Gb. Along with the images in the dataset is a

CSV file that lists all the images ids and their corresponding pixels coordinates. These coordinates represent segmentation bounding boxes of ships. Not having pixel coordinates for an image means that particular image doesn't have any ships.

4.2.3 Methodology and Code Snippets

The methodology followed has been outlined in the following steps -

Step 1: Initial Setup

Downloaded the dataset, installed required dependencies and directory files.

Step 2: Splitting into Training and Validation sets

The dataset is split into training and validation set with 80% and 20% images respectively. Since a few images had more than just a single ship, it was important to split each category of number of ships by an 80:20 ratio. This can be observed in Fig. 25.

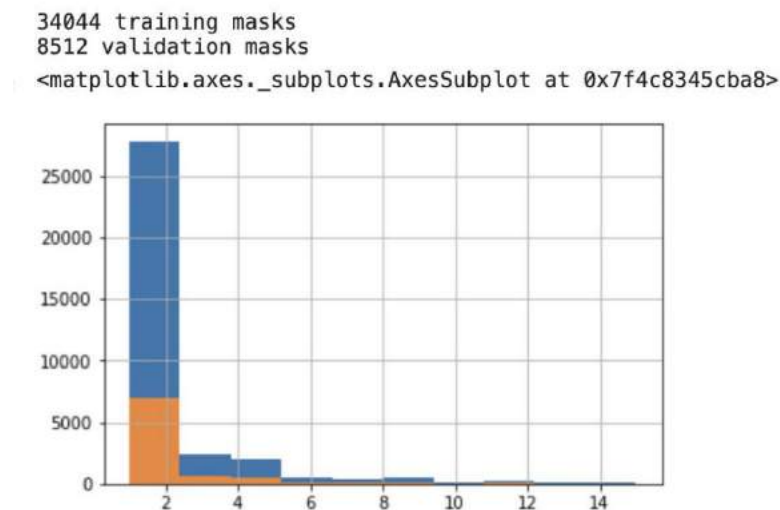


Fig. 25 – Graph depicting the 80:20 split

The colour blue denotes the training set and orange denotes the validation set. Upon execution, 34,044 training images and 8,512 validation images were generated. The X-axis denotes the number of ship masks in each image.

Step 3: Applying Masks to all images

The next step of the process was to apply masks to all images using a simple function written in Python. The code snippet has been presented in Fig. 26, and the output ship images with masks have been represented in Fig. 27.


```
def masks_as_image(in_mask_list, shape=SHAPE):
    all_masks = np.zeros(shape, dtype = np.int16)
    for mask in in_mask_list:
        if isinstance(mask, str):
            all_masks += rle_decode(mask)
    return np.expand_dims(all_masks, -1)
```

Fig. 26 – A Code Snippet of the Mask Generation Function

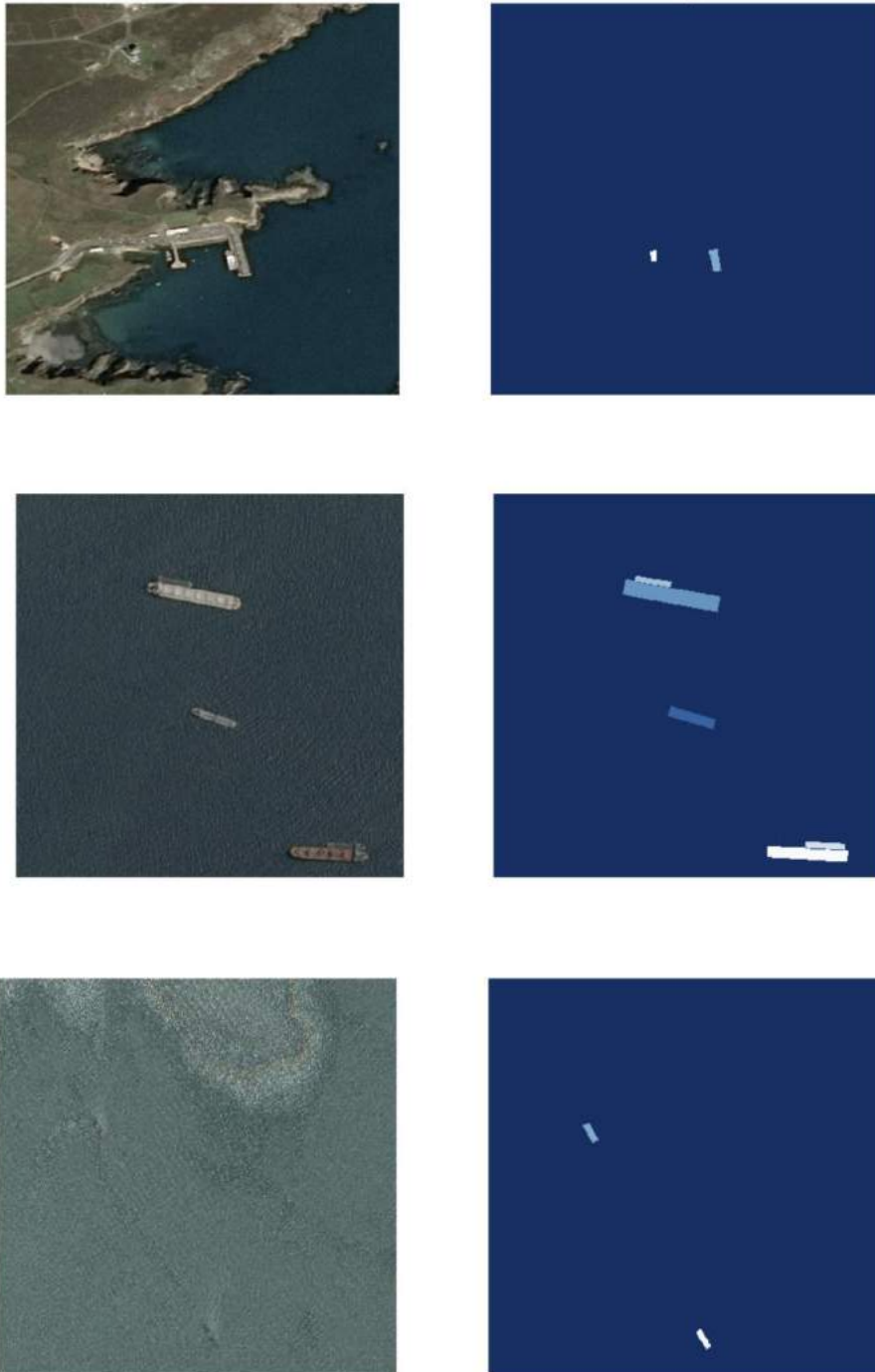


Fig. 27 – A few examples of masks generated for ship images

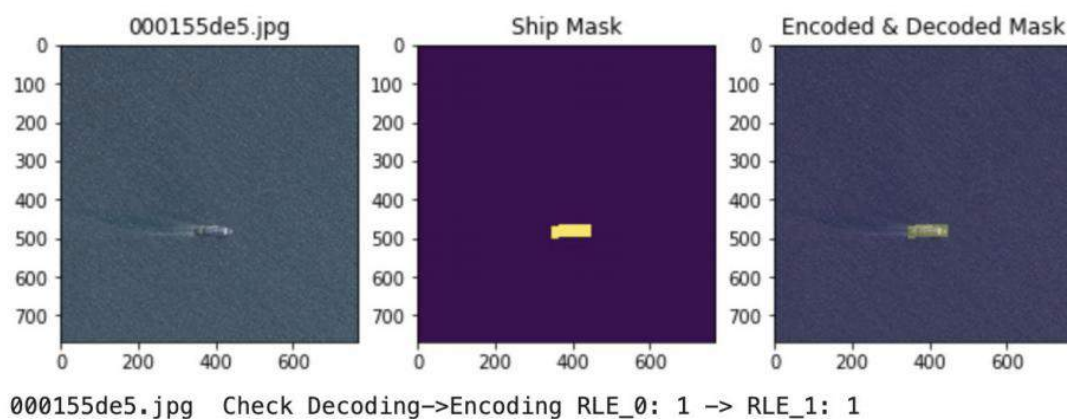
Step 4: Encoding and Decoding Masks

Next, a function was created for encoding and decoding mask on top of every image followed by which, a Mask R-CNN algorithm for segmenting the ships in image with a confidence score between 0 and 1 is set. The time required to mask out ships in the complete dataset was 9.82 seconds. While the function in Python to do so has been presented in Fig. 28, a few encoded and decoded masks are displayed in Fig. 29.

```
def shows_decode_encode(image_id, path=TRAIN_DATA_PATH):

    fig, axarr = plt.subplots(1, 3, figsize = (10, 5))
    img_0 = imread(os.path.join(path, image_id))
    axarr[0].imshow(img_0)
    axarr[0].set_title(image_id)
    rle_1 = masks.query('ImageId=="{}".format(image_id))['EncodedPixels']
    img_1 = masks_as_image(rle_1)
    axarr[1].imshow(img_1[:, :, 0])
    axarr[1].set_title('Ship Mask')
    rle_2 = multi_rle_encode(img_1)
    img_2 = masks_as_image(rle_2)
    axarr[2].imshow(img_0)
    axarr[2].imshow(img_2[:, :, 0], alpha=0.3)
    axarr[2].set_title('Encoded & Decoded Mask')
    plt.show()
    print(image_id , ' Check Decoding->Encoding',
          'RLE_0:', len(rle_1), '->',
          'RLE_1:', len(rle_2))
```

Fig. 28 – A Code Snippet with the Encode-Decode Mask Function



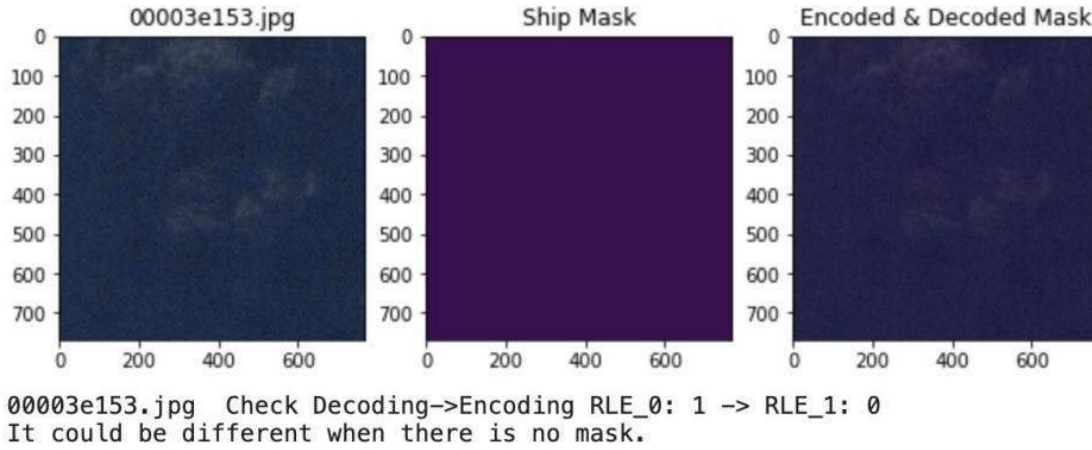


Fig. 29 – Encoded and Decoded Ship Masks

Step 5: Training the Model

Pre-trained MS COCO weights are used for training the model. Since these weights have already been trained on a large variety of objects, they provide a good place to start learning from. The results obtained with each passing epoch during training has been displayed in Fig. 30. The model was trained for 5 epochs.

```
Epoch 1/5
300/300 [=====] - 320s 1s/step - loss: 1.1494 - rpn_class_loss: 0.0137 - rpn_bbox_loss: 0.4088 - mrcnn_class_loss:
0.0550 - mrcnn_bbox_loss: 0.3604 - mrcnn_mask_loss: 0.3115 - val_loss: 1.3324 - val_rpn_class_loss: 0.0168 - val_rpn_bbox_loss: 0.6377 - val_
mrcnn_class_loss: 0.0551 - val_mrcnn_bbox_loss: 0.2997 - val_mrcnn_mask_loss: 0.3232
Epoch 2/5
300/300 [=====] - 253s 842ms/step - loss: 0.9392 - rpn_class_loss: 0.0126 - rpn_bbox_loss: 0.3684 - mrcnn_class_loss:
s: 0.0413 - mrcnn_bbox_loss: 0.2426 - mrcnn_mask_loss: 0.2743 - val_loss: 1.0614 - val_rpn_class_loss: 0.0109 - val_rpn_bbox_loss: 0.5029 - v
al_mrcnn_class_loss: 0.0233 - val_mrcnn_bbox_loss: 0.2541 - val_mrcnn_mask_loss: 0.2701
Epoch 3/5
300/300 [=====] - 250s 833ms/step - loss: 0.8708 - rpn_class_loss: 0.0102 - rpn_bbox_loss: 0.3276 - mrcnn_class_loss:
s: 0.0342 - mrcnn_bbox_loss: 0.2261 - mrcnn_mask_loss: 0.2727 - val_loss: 0.9469 - val_rpn_class_loss: 0.0096 - val_rpn_bbox_loss: 0.3484 - v
al_mrcnn_class_loss: 0.0396 - val_mrcnn_bbox_loss: 0.2488 - val_mrcnn_mask_loss: 0.3005
Epoch 4/5
300/300 [=====] - 249s 829ms/step - loss: 0.9003 - rpn_class_loss: 0.0098 - rpn_bbox_loss: 0.3483 - mrcnn_class_loss:
s: 0.0413 - mrcnn_bbox_loss: 0.2233 - mrcnn_mask_loss: 0.2777 - val_loss: 0.9566 - val_rpn_class_loss: 0.0088 - val_rpn_bbox_loss: 0.3909 - v
al_mrcnn_class_loss: 0.0271 - val_mrcnn_bbox_loss: 0.2334 - val_mrcnn_mask_loss: 0.2963
Epoch 5/5
300/300 [=====] - 247s 822ms/step - loss: 0.9356 - rpn_class_loss: 0.0104 - rpn_bbox_loss: 0.3824 - mrcnn_class_loss:
s: 0.0289 - mrcnn_bbox_loss: 0.2296 - mrcnn_mask_loss: 0.2843 - val_loss: 1.0667 - val_rpn_class_loss: 0.0103 - val_rpn_bbox_loss: 0.4052 - v
al_mrcnn_class_loss: 0.0382 - val_mrcnn_bbox_loss: 0.2681 - val_mrcnn_mask_loss: 0.3439
Train model: 1495.0196392536163
```

Fig. 30 – Training the Mask RCNN Model

4.2.4 Results

It is found that using mask shape of size 14*14 and Resnet101 backbone, the Mask RCNN algorithm is able to detect and segment ships with a mean average precision of 0.605. It took the model approximately 30 seconds to segment 30 images. The graph depicting training loss versus epoch has been visualised in Fig. 31, while a few snippets of the output images have been presented in Figure 32.

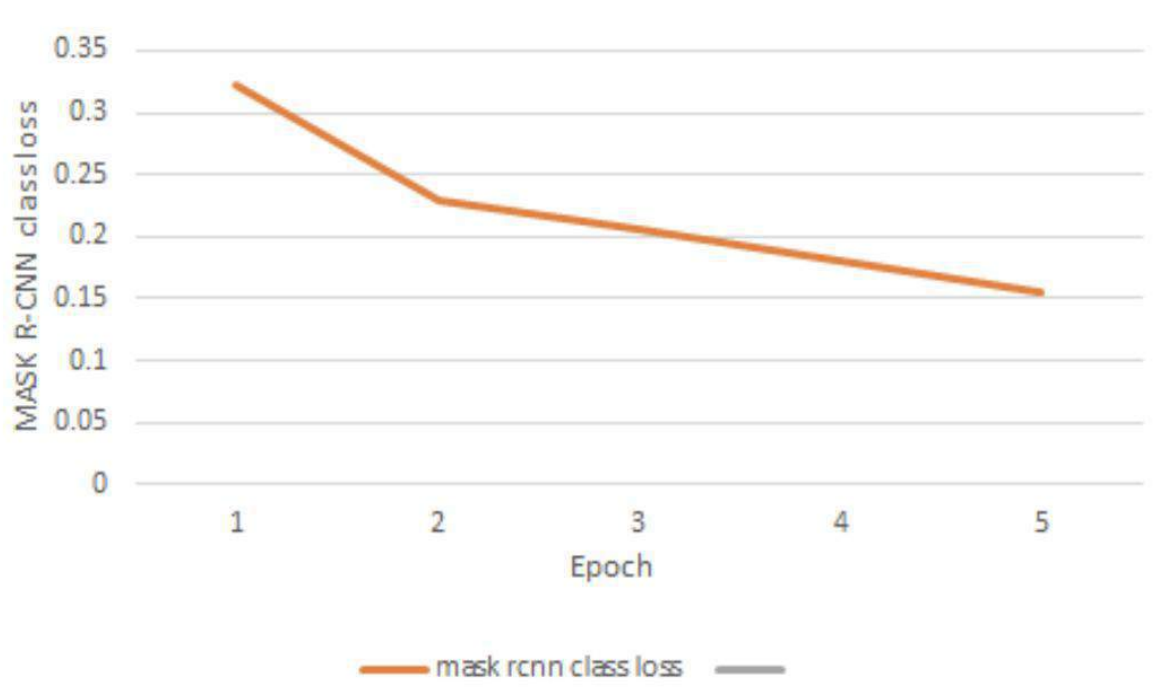
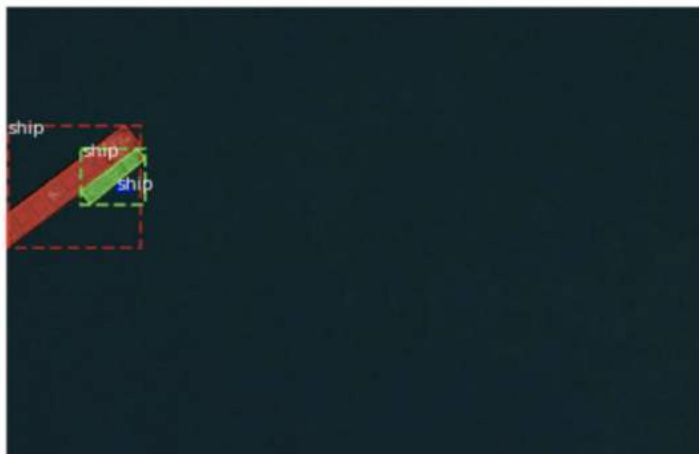


Fig. 31 – Training Loss vs Epoch for Mask R-CNN

Re-starting from epoch 3

original_image	shape: (768, 768, 3)	min: 0.00000	max: 250.00000	uint8
image_meta	shape: (14,)	min: 0.00000	max: 7698.00000	int64
gt_class_id	shape: (3,)	min: 1.00000	max: 1.00000	int32
gt_bbox	shape: (3, 4)	min: 0.00000	max: 263.00000	int32
gt_mask	shape: (768, 768, 3)	min: 0.00000	max: 1.00000	bool



Processing 1 images

image	shape: (768, 768, 3)	min: 0.00000	max: 255.00000	uint8
molded_images	shape: (1, 768, 768, 3)	min: -123.70000	max: 151.10000	float64
image metas	shape: (1, 14)	min: 0.00000	max: 768.00000	int64
anchors	shape: (1, 147312, 4)	min: -0.47202	max: 1.38858	float32



```

Processing 1 images
image          shape: (768, 768, 3)      min: 0.00000 max: 255.00000 uint8
molded_images  shape: (1, 768, 768, 3) min: -107.70000 max: 147.10000 float64
image metas    shape: (1, 14)          min: 0.00000 max: 768.00000 int64
anchors        shape: (1, 147312, 4)   min: -0.47202 max: 1.38858 float32
*** No instances to display ***

```



```

Processing 1 images
image          shape: (768, 768, 3)      min: 0.00000 max: 255.00000 uint8
molded_images  shape: (1, 768, 768, 3)   min: -123.70000 max: 151.10000 float64
image metas    shape: (1, 14)          min: 0.00000 max: 768.00000 int64
anchors        shape: (1, 147312, 4)   min: -0.47202 max: 1.38858 float32

```



Fig. 32 – A few snippets of the Mask R-CNN results

4.2.5 Key Takeaways

1. Even though the model was trained for 5 epochs till the loss was minimal, a few problems persist. In a few images, the land in and around the ships has also been getting detected, as seen in the Figure 33. A workaround for these false positives needed to be worked out.
2. Moreover, MASK R-CNN involves generation of mask as images that takes up a lot of storage space as well as computation power. It is also extremely slow to execute.

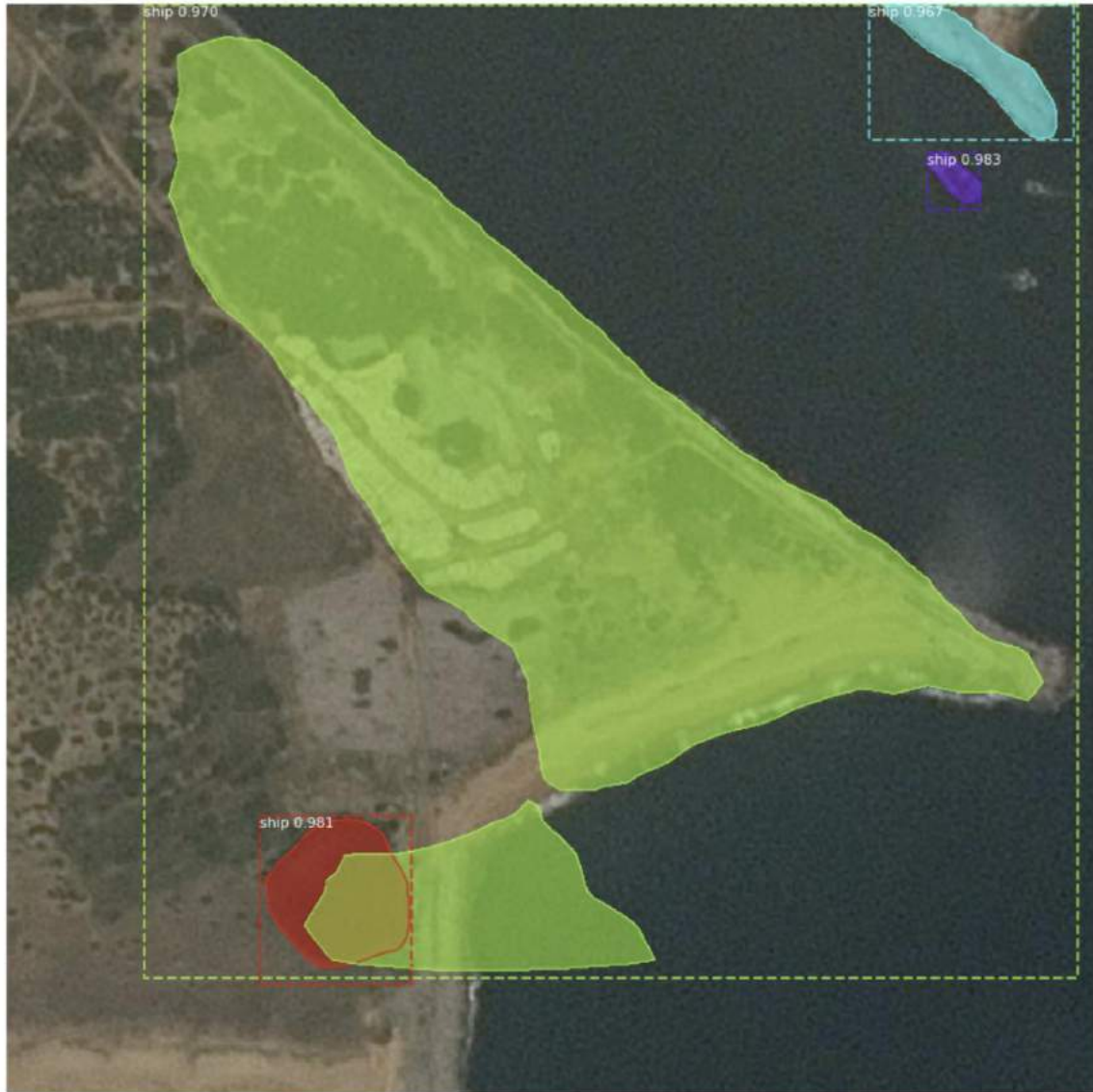


Fig. 33 – Drawback of using Mask R-CNN (Generation of False Positives)

4.3 Phase 3: Implementation of YOLO

4.3.1 Objective

For the final phase of the project, the YOLOv4 algorithm was picked since it has several advantages over general classifier-based systems. For example, YOLO looks at the whole image at test time so its predictions are informed by global context in the image. It also makes predictions with a single network evaluation unlike systems like R-CNN which require thousands for a single image. This makes it extremely fast, more than 1000x faster than R-CNN and 100x faster than Mask R-CNN.

4.3.2 Dataset

After the experimentation with small-sized and medium-sized databases previously, this phase used a large-scale Image dataset for Maritime Vessels, known as the MARVEL dataset. It consists of 2 million user uploaded images and their attributes including vessel identity, type, photograph category and year of built, collected from a community website. It can be categorized into 109 vessel type classes and 26 super-classes by combining heavily populated classes with a semi-automatic clustering scheme. A snippet of the dataset has been presented in Fig. 34.

However, DRDO handed over a mini-version of the MARVEL dataset to perform the preliminary analysis. This consisted of 2,800 images, spread uniformly across five major maritime classes, namely, cargo, carrier, cruise, military and tanker ships.



Fig. 34 – The Marvel Dataset

4.3.3 Methodology and Code Snippets

Step 1: Labelling of Ground-Truth Images

Since the MARVEL dataset did not come with pre-assigned labels, like other custom-made datasets, each image had to be labelled manually using a tool known as LabelMe. An open annotation tool developed by MIT, the goal of the open-source project was to develop datasets for computer vision and image processing.

A snippet of the interface has been provided in Fig. 35.

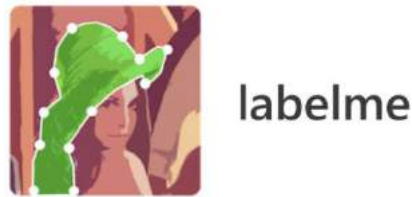


Fig. 35 – The LabelMe Annotation Tool Interface

The output of the LabelMe tool saves the coordinates of the bounding box of an image in a json file that looks like Fig. 36.

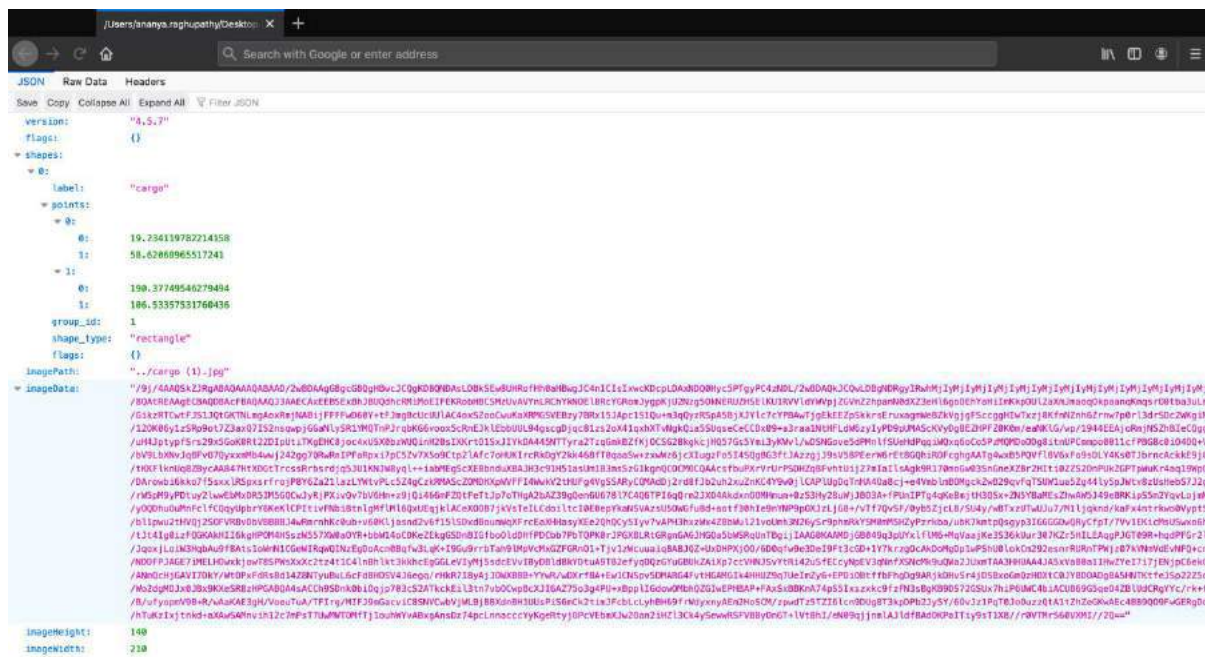
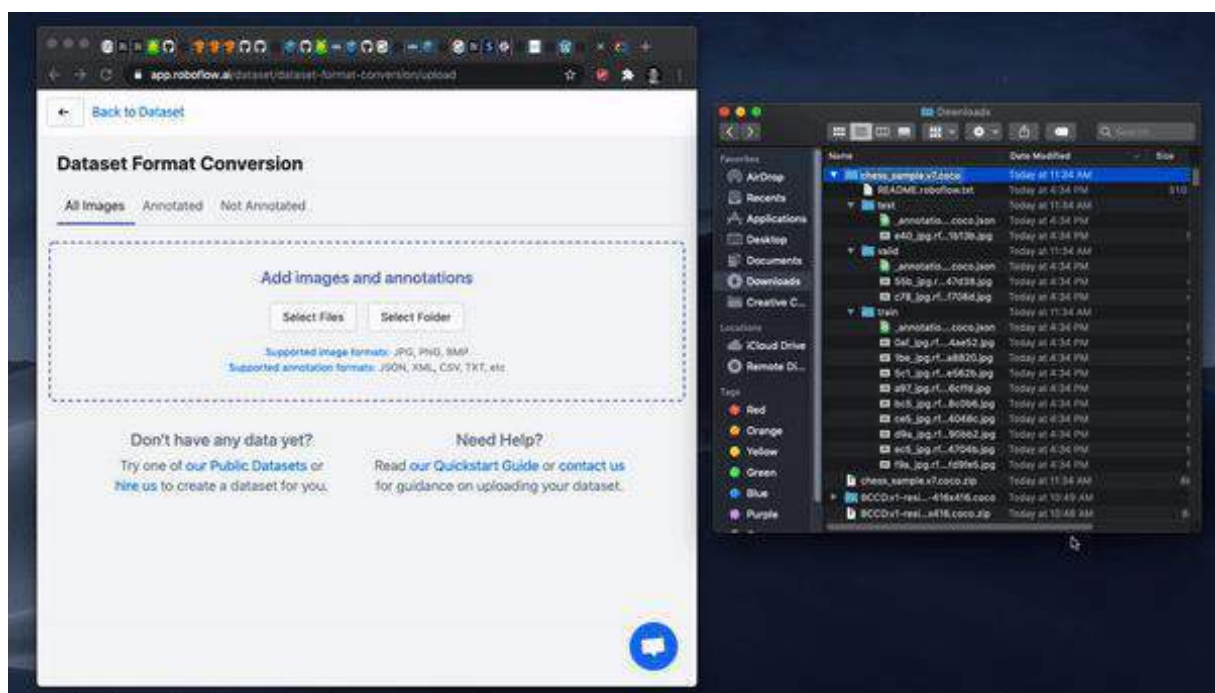


Fig. 36 – The output from the LabelMe Annotation Tool

Step 2: Converting the JSON files into YOLO .txt files

Since the LabelMe annotation tool saves the output of each image in JSON format, it is mandatory to convert them into a format that YOLO model can interpret. In YOLO labelling format, a .txt file with the same name is created for each image file in the same directory. Each .txt file contains the annotations for the corresponding image file, that is object class, object coordinates, height and width. Most YOLO versions can only read .txt files with a specific format.

As a result, to convert the JSON files to .txt files for each corresponding image, the services of the RoboFlow platform were utilised. The process followed has been displayed in screenshots, which are presented in Fig. 37.



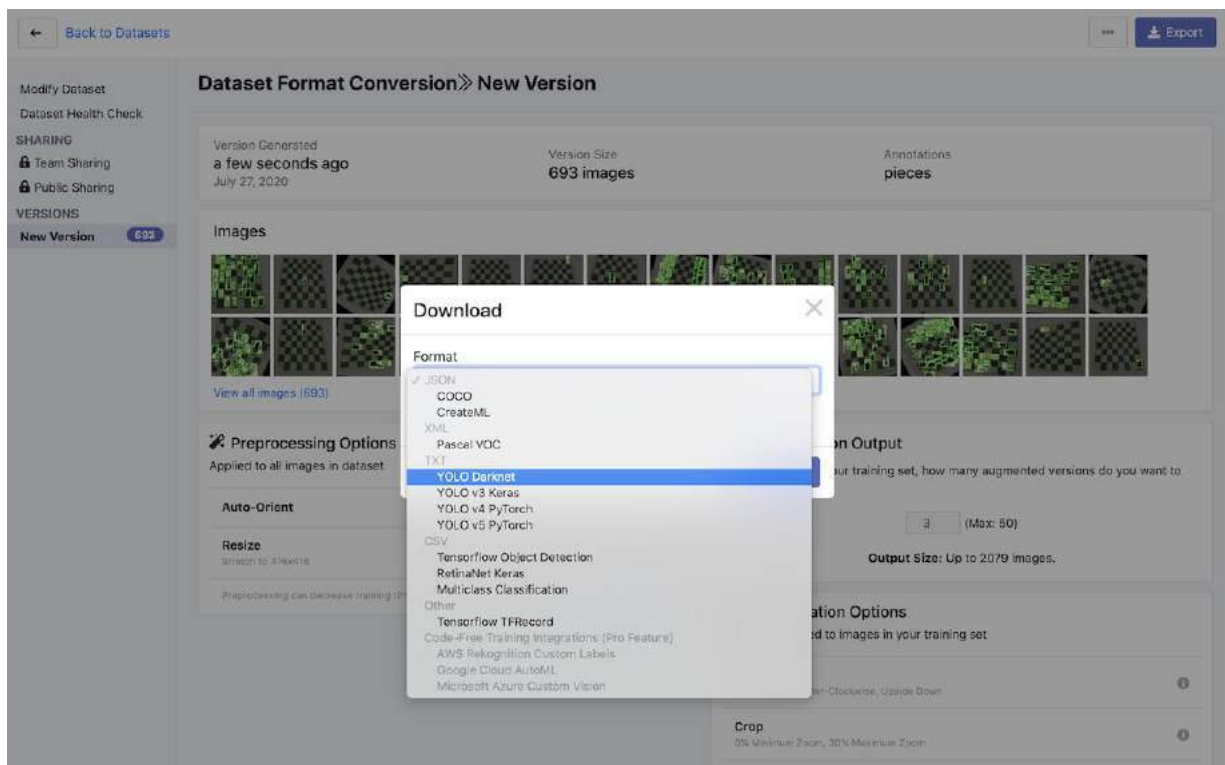
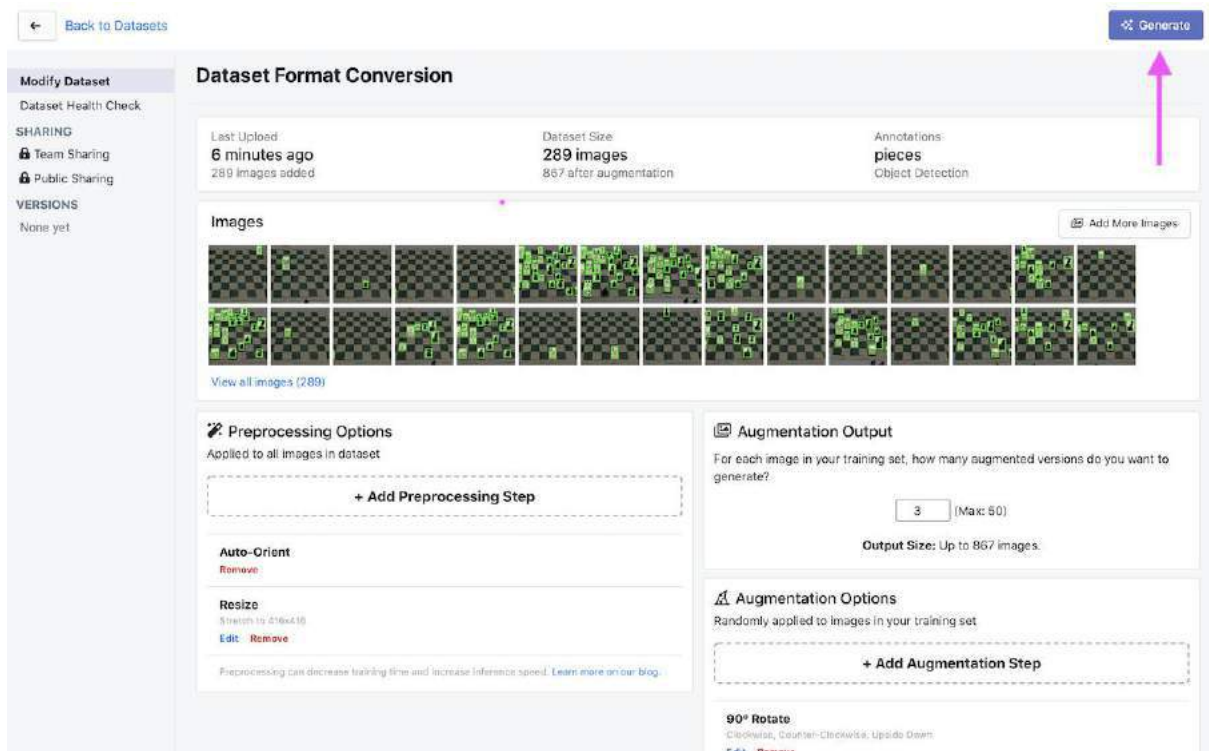


Fig. 37 – The RoboFlow Process

Step 3: Training the Model

As mentioned earlier, YOLO is an object detection algorithm touted to be one of the fastest. It was trained on the COCO dataset and achieved a mean Average Precision (mAP) of 33.0 at the speed of 51ms per image on Titan X GPU, which is commendable. The major highlight of

the algorithm is that it divides the input image into several individual grids, and each grid predicts the objects inside it. This way, the whole image is processed at once, and the inference time is reduced.

The YOLO model used in this project is trained on the Darknet Framework, which is an open-source neural network framework written in C and CUDA. It is fast, easy to install, and supports CPU and GPU computation.

After the initial configuration and setup, the YOLOv4 darknet model was trained for over 2000 iterations, at the end of which the following results were received.

4.3.4 Results

The results obtained after 2000 iterations have been presented in Fig. 38.

```
Last accuracy mAP@0.5 = 82.02 %, best = 82.02 %
2000: 0.594563, 0.990275 avg loss, 0.001000 rate, 6.906656 seconds, 128000 images, 11.574178 hours left
Resizing to initial size: 416 x 416 try to allocate additional workspace_size = 111.05 MB
CUDA allocate done!

calculation mAP (mean average precision)...
Detection layer: 139 - type = 28
Detection layer: 150 - type = 28
Detection layer: 161 - type = 28
180
detections_count = 651, unique_truth_count = 275
class_id = 0, name = ship, ap = 99.64% (TP = 52, FP = 2)
class_id = 1, name = cargo, ap = 98.39% (TP = 61, FP = 0)
class_id = 2, name = carrier, ap = 91.99% (TP = 100, FP = 18)
class_id = 3, name = cruise, ap = 39.35% (TP = 30, FP = 37)

for conf_thresh = 0.25, precision = 0.81, recall = 0.88, F1-score = 0.85
for conf_thresh = 0.25, TP = 243, FP = 57, FN = 32, average IoU = 67.08 %

IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
mean average precision (mAP@0.50) = 0.823406, or 82.34 %
Total Detection Time: 5 Seconds

Set -points flag:
`-points 101` for MS COCO
`-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data)
`-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset

mean_average_precision (mAP@0.5) = 0.823406
New best mAP!
Saving weights to /mydrive/yolov4/training/yolov4-custom_best.weights
Saving weights to /mydrive/yolov4/training/yolov4-custom_2000.weights
Saving weights to /mydrive/yolov4/training/yolov4-custom_last.weights
```

Fig. 38 – Results using the YOLOv4 model

The performance metric that was chosen was the mean average precision. The mean average precision (mAP) of a set of queries is defined by the formula in Fig. 39, where Q is the number of queries in the set and AveP(q) is the average precision (AP) for a given query, q.

$$MAP = \frac{\sum_{q=1}^Q AveP(q)}{Q} \dots\dots\dots (3)$$

Fig. 39 – The mAP formula

The mAP compares the ground-truth bounding box to the detected box and returns a score between 0 to 1. The higher the score, the more accurate the model is in its detections. With a mean average precision (mAP) value at 0.8234, the results obtained using the YOLO model were commendable.

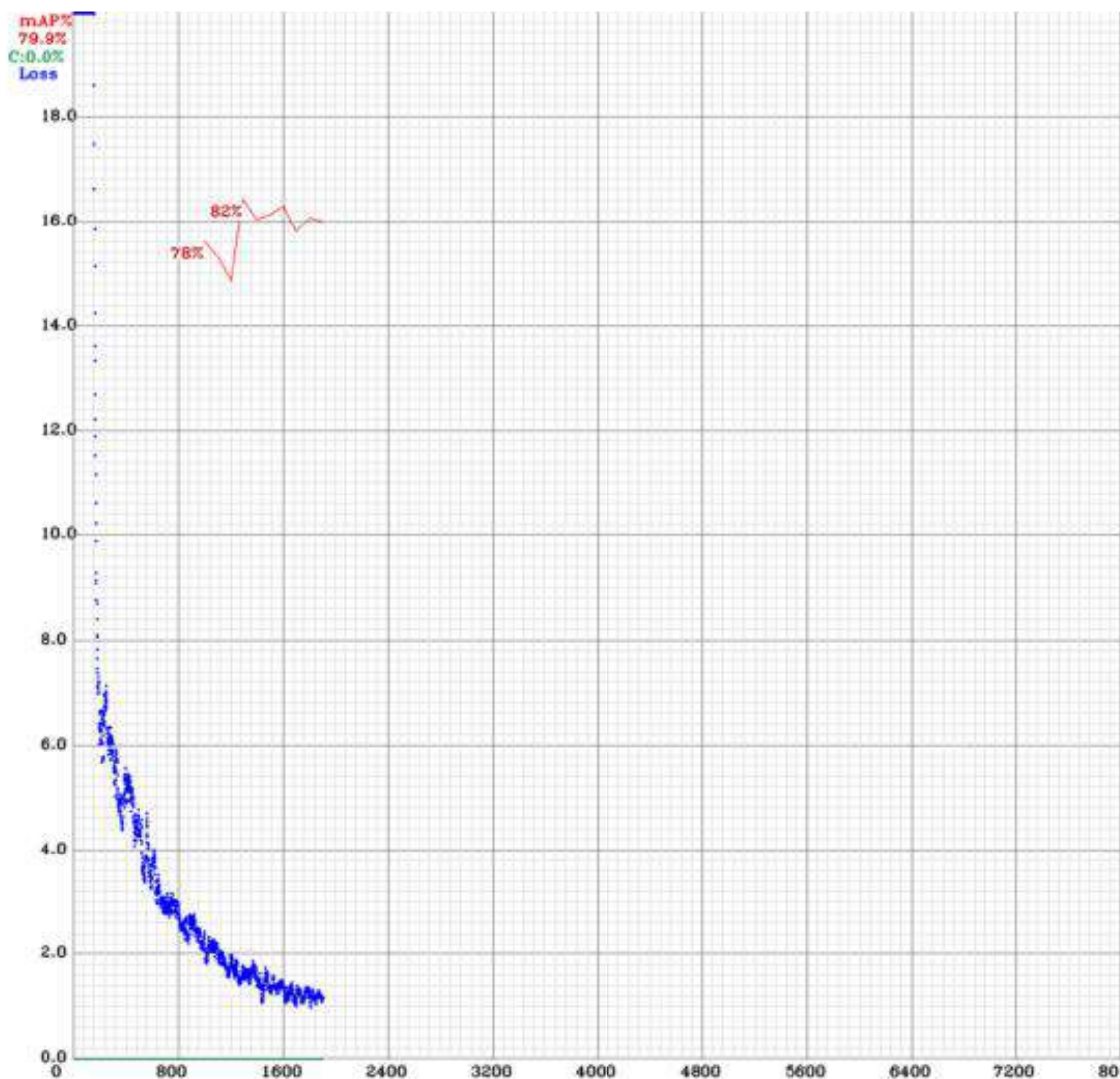
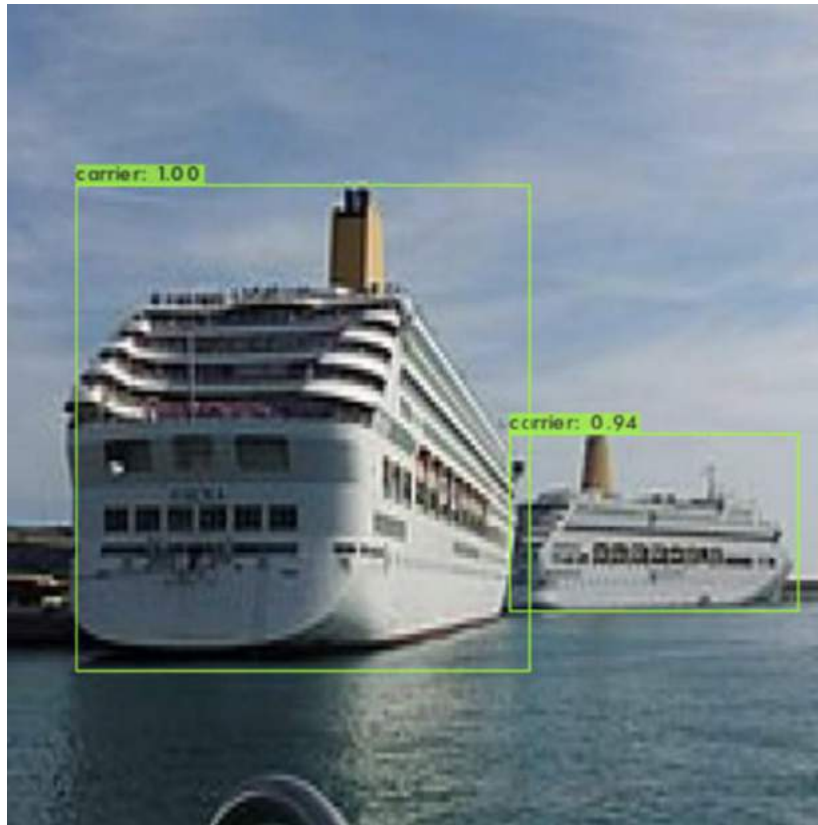


Fig. 40 – The loss and accuracy vs epoch of the YOLO model

A few results obtained by the YOLO model have been presented in Fig. 41.



Truth – Cargo, Cargo; Prediction – Carrier, Carrier



Truth – Carrier, Ship; Predicted – Ship, Cruise



Truth – Cargo, Cargo; Predicted – Carrier, Carrier

Fig. 41 – YOLO results

4.3.5 Key Takeaways

1. The YOLO model ran for 2000 iterations and returned a mAP value of 0.823. The mAP value however denotes the bounding box accuracy, and not the label accuracy. As seen in the results snapshots, the labels were not being predicted accurately by the YOLO model.
2. The YOLO model thus did a great job at object detection but didn't meet the project's requirements of object classification.
3. The next logical step is to implement a CNN Classifier that will be connected to the YOLO model such that, the images of the YOLO model will be passed on to the CNN Classifier as well.

4.4 Phase 4: Implementation of CNN Classifier

4.4.1 Objective

The implementation of the CNN Classifier aims to solve the shortcoming of the YOLOv4 model, and reach the project's objective of image classification. In other words, the CNN

Classifier will be used to classify the detected object of the YOLO model into one of the following three classes – cargo, carrier and cruise.

4.4.2 Dataset

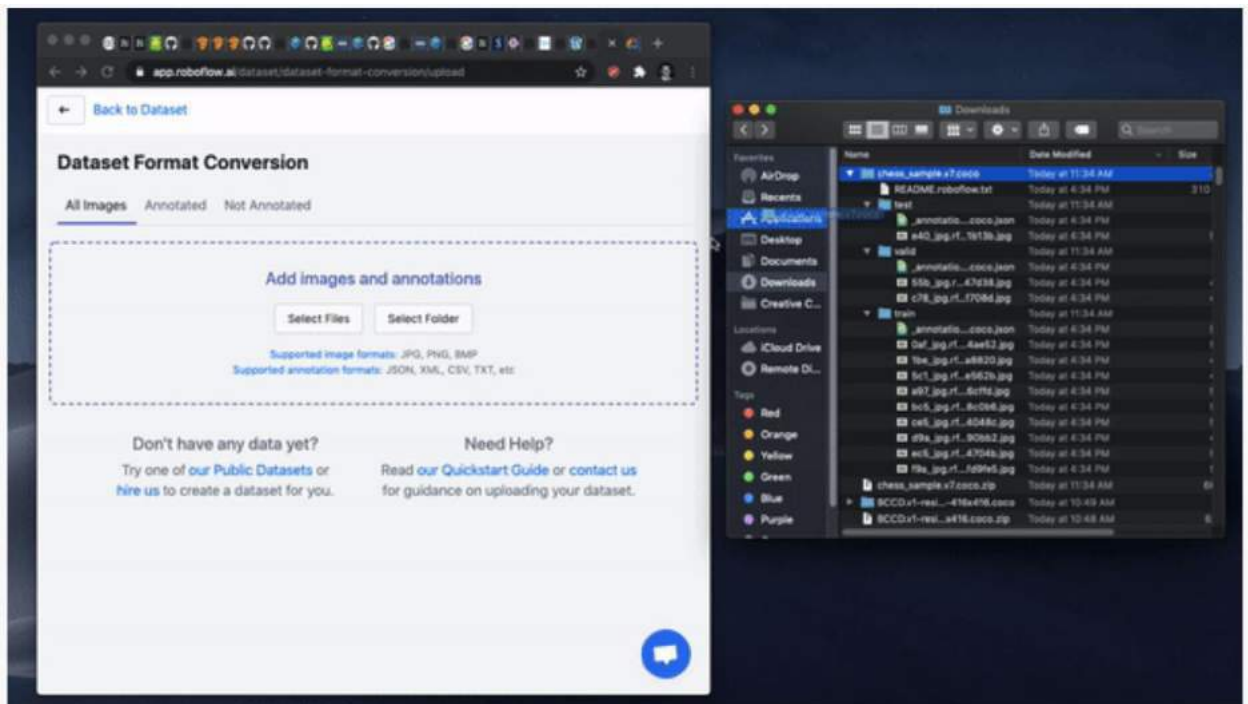
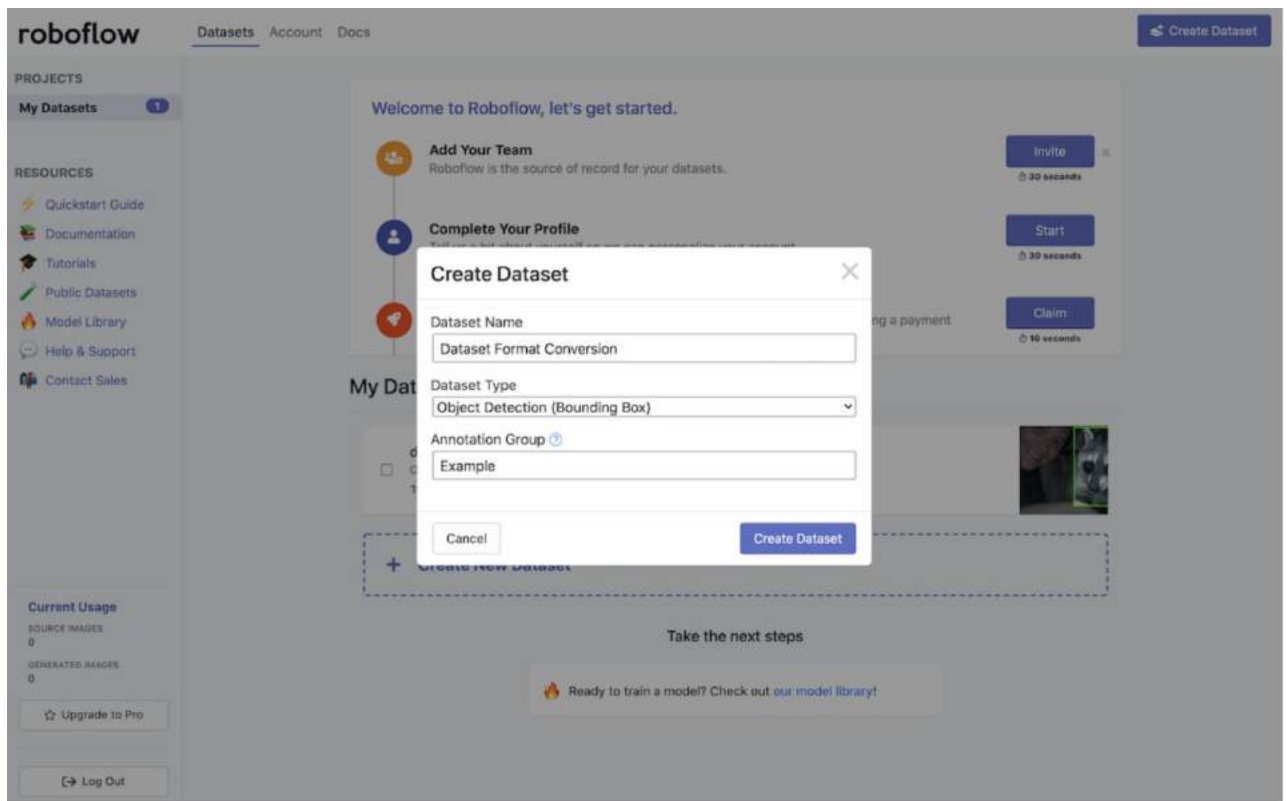
The CNN Classifier uses the MARVEL Maritime dataset, just like the YOLO Model. A snippet of the same has been presented previously in Fig. 34. However, since the LabelMe tool was used to assign labels to the images in a json format, there had to be modifications done to the dataset for it to be utilised by the CNN Classifier. Further details regarding the same have been elaborated in Step 1 of Section 4.3.3 - Methodology.

4.4.3 Methodology and Code Snippets

Step 1: Getting the Dataset ready

The dataset had to be first converted into a CNN Classifier format. A CNN Classifier requires all images to be in a folder along with a one-hot encoded .csv file with classification details regarding each image in the folder. To do so, the roboflow platform was utilised. The steps followed using the Roboflow platform, file organisation in the dataset and a snippet of the .csv file used by the CNN Classifier have been presented in Fig. 42, Fig. 43, and Fig. 44 respectively.





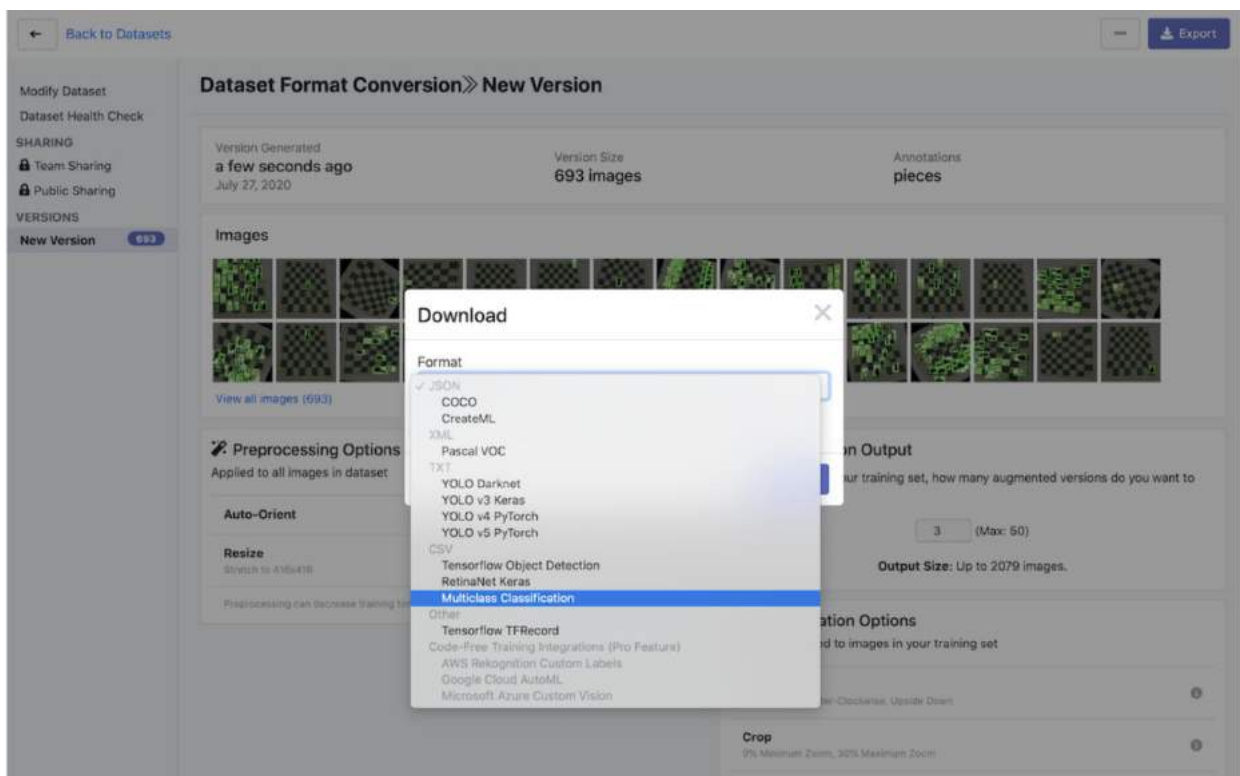
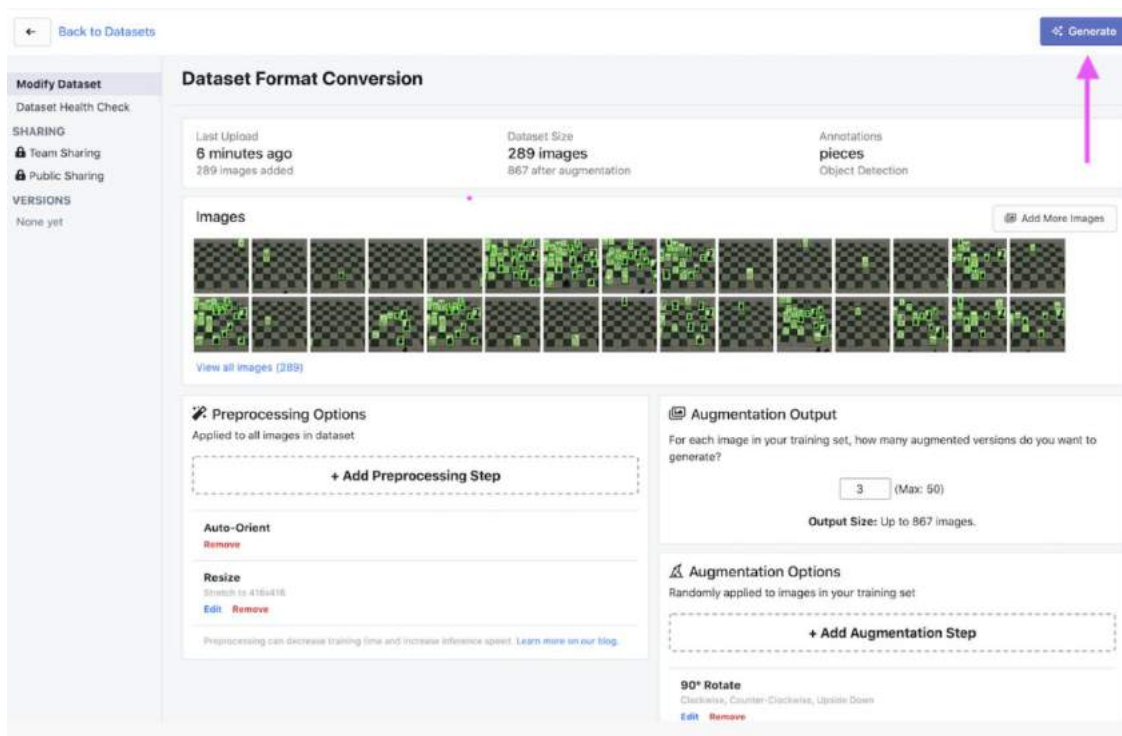


Fig. 42 – Steps followed on the RoboFlow interface

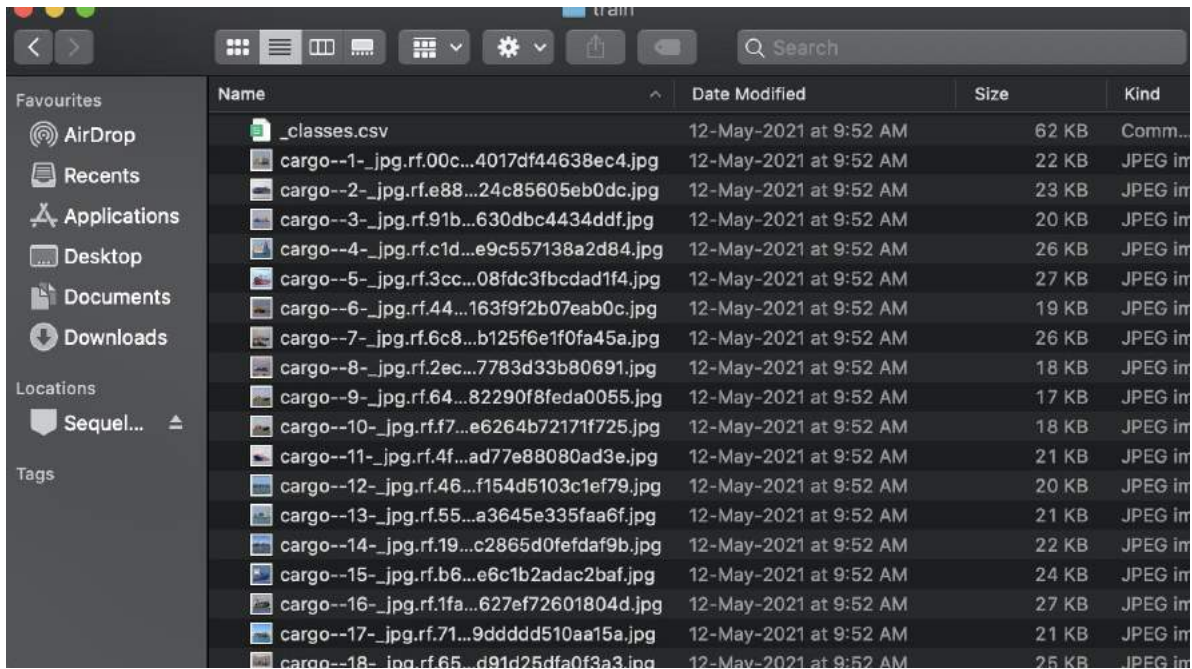


Fig. 43 – File organisation

	A	B	C	D	E
1	filename	cargo	carrier	cruise	ship
2	carrier--110-.jpg.rf.dcf16146ea537d231164	0	1	0	0
3	cruise--52-.jpg.rf.dcc5a44f5ba0365022c2e	0	0	1	0
4	cargo--121-.jpg.rf.da96e700b8c9e9196dee	1	0	0	0
5	carrier--152-.jpg.rf.d7f5e670ea077c77b805	0	1	0	0
6	carrier--210-.jpg.rf.dbdf51faa7c2d08809f85	0	1	0	0
7	cargo--22-.jpg.rf.d83dacd06ce35a972dbb4	1	0	0	0
8	cruise--78-.jpg.rf.dcc9d10367843b9b6aaf9	0	0	1	0
9	cargo--275-.jpg.rf.da817b68128f9e834a009	1	0	0	0
10	carrier--281-.jpg.rf.d763e100c0ee85eb2d9c	0	1	0	0
11	cruise--227-.jpg.rf.d7b5208614b2f496d45e	0	0	1	0
12	cruise--118-.jpg.rf.d8818c3bb0b422c97b94	0	0	1	1
13	cruise--40-.jpg.rf.d9cb450cfa1c0eedbf3e5	0	0	1	0
14	cruise--10-.jpg.rf.db6c46890e02efed6a229	0	0	1	0
15	cargo--289-.jpg.rf.d934253cbf5f0d6f3c2e4	1	0	0	0
16	carrier--131-.jpg.rf.d9bc0834da24cfe6aa56	0	1	0	0
17	cruise--21-.jpg.rf.d8536ade85186a8148c9f	0	0	1	0
18	carrier--299-.jpg.rf.d821200b5f24d5fc6e9	0	1	0	1
19	cargo--52-.jpg.rf.dc779bf40199585e97df89	1	0	0	0
20	cruise--8-.jpg.rf.d9c8f73c0c64f29a00d0193	0	0	1	0
21	carrier--173-.jpg.rf.d996ca4824d9b684c45f	0	1	0	0
22	carrier--169-.jpg.rf.db0569ba29ea43fbc516	0	1	0	0
23	carrier--63-.jpg.rf.dbdbf80b9ab212780cd82	0	1	0	0
24	cruise--112-.jpg.rf.dd4765ba84d957f89fedf	0	0	1	0
25	carrier--143-.jpg.rf.dcc0083e3ec49c1c6a7a	0	1	0	0
26	cruise--147-.jpg.rf.de1884db8712c6ea08db	0	0	1	0
27	cruise--119-.jpg.rf.dbbb713859a8a6a485d7	0	0	1	1
28	carrier--213-.jpg.rf.d89428900b8bb55da12f	0	1	0	0
29	cargo--83-.jpg.rf.dddc4956ca48b0d507642	1	0	0	0
30	cruise--126-.jpg.rf.df2d5d9ca055801e530c	0	0	1	1
31	cargo--63-.jpg.rf.dec67c76783fc3c094a3e2	1	0	0	0
32	carrier--135-.jpg.rf.e06faec415bdd7ee7813	0	1	0	0

Fig. 44 – A snippet of the .csv file with one-hot encoded data

Step 2: Splitting the dataset

Once the dataset is ready, it is split into a ratio of 80:10:10 to accommodate the train, test and validation sets.

Step 3: Modelling the Classifier

In order to build the CNN Classifier, a mix of ReLU and Sigmoid Activation Functions were used. While the ReLU layer was used alternately, the Sigmoid Activation Function was placed as the last layer due to its ability to tackle multi-label classification efficiently. While the performance parametric was accuracy, the loss parametric was chosen as binary cross entropy – again, due to their suitability to the problem statement of multi label classification. Moreover, MaxPooling was also implemented. A snippet of the model used by the classifier has been presented in Fig. 45.

```
model = Sequential()
model.add(Conv2D(32, (3, 3), padding='same',
                input_shape=(100,100,3)))
model.add(Activation('relu'))
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(4, activation='sigmoid'))
model.compile(optimizer=RMSprop(lr=0.0001, decay=1e-6), loss="binary_crossentropy", metrics=["accuracy"])
```

Fig. 45 – The Classifier Model

Step 4: Training

```
Epoch 1/10
22/22 [=====] - 36s 138ms/step - loss: 0.6311 - accuracy: 0.3113 - val_loss: 0.6238 - val_accuracy: 0.5625
Epoch 2/10
22/22 [=====] - 2s 108ms/step - loss: 0.5883 - accuracy: 0.4236 - val_loss: 0.6365 - val_accuracy: 0.3906
Epoch 3/10
22/22 [=====] - 2s 110ms/step - loss: 0.5677 - accuracy: 0.4908 - val_loss: 0.5984 - val_accuracy: 0.4844
Epoch 4/10
22/22 [=====] - 3s 113ms/step - loss: 0.5478 - accuracy: 0.5173 - val_loss: 0.5818 - val_accuracy: 0.5781
Epoch 5/10
22/22 [=====] - 2s 112ms/step - loss: 0.5271 - accuracy: 0.5637 - val_loss: 0.5332 - val_accuracy: 0.5625
Epoch 6/10
22/22 [=====] - 2s 111ms/step - loss: 0.4815 - accuracy: 0.6568 - val_loss: 0.5049 - val_accuracy: 0.7500
Epoch 7/10
22/22 [=====] - 2s 112ms/step - loss: 0.4715 - accuracy: 0.6551 - val_loss: 0.5049 - val_accuracy: 0.6875
Epoch 8/10
22/22 [=====] - 3s 115ms/step - loss: 0.4490 - accuracy: 0.6895 - val_loss: 0.4699 - val_accuracy: 0.7500
Epoch 9/10
22/22 [=====] - 3s 116ms/step - loss: 0.4541 - accuracy: 0.6594 - val_loss: 0.4837 - val_accuracy: 0.6875
Epoch 10/10
22/22 [=====] - 3s 116ms/step - loss: 0.4336 - accuracy: 0.7350 - val_loss: 0.4645 - val_accuracy: 0.7969
<tensorflow.python.keras.callbacks.History at 0x7faafa385450>
```

Fig. 46 – Training the CNN Classifier

As seen in Fig. 46, the model was trained for 10 epochs, finally resulting in an accuracy of 73.5 percent, validation loss of 0.4645 and a validation accuracy of 79.69 percent.

4.4.4 Results

In order to check the results, a confusion matrix was plotted, as shown in Fig. 48. Confusion matrices represent counts from predicted and actual values. The output “TN” stands for True Negative which shows the number of negative examples classified accurately. Similarly, “TP” stands for True Positive which indicates the number of positive examples classified accurately. The term “FP” shows False Positive value, i.e., the number of actual negative examples classified as positive; and “FN” means a False Negative value which is the number of actual positive examples classified as negative. One of the most commonly used metrics while performing classification is accuracy. The accuracy of a model (through a confusion matrix) is calculated using the given formula in Fig. 47.

$$Accuracy = \frac{TN + TP}{TN + FP + FN + TP}. \quad \dots\dots\dots (4)$$

Fig. 47 – Accuracy for each class in a confusion matrix

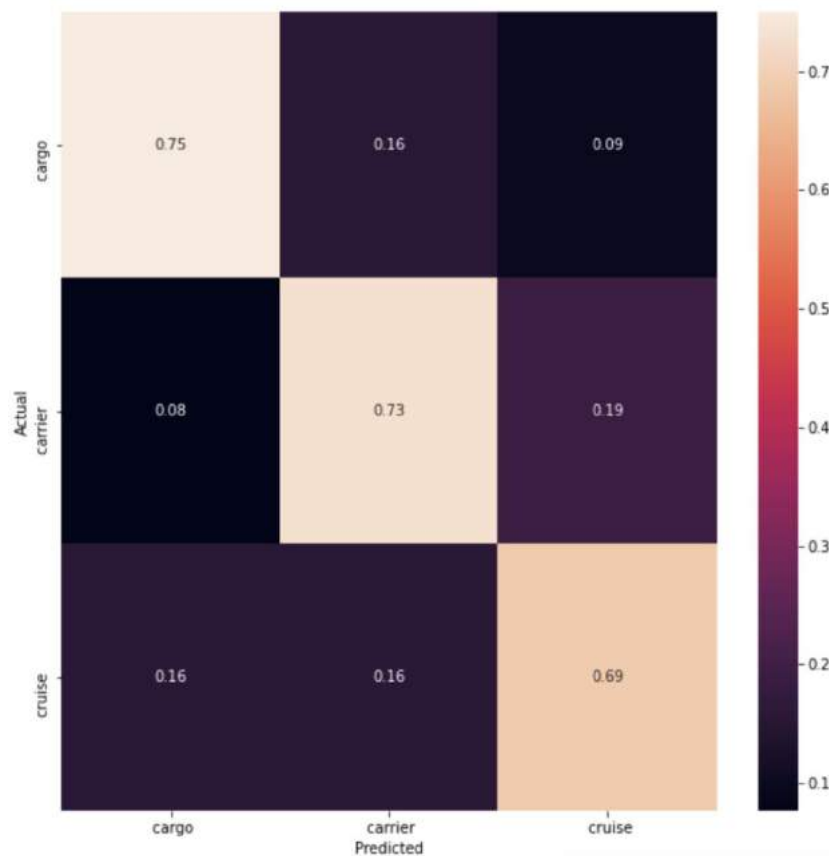


Fig. 48 – Confusion Matrix of the CNN Classifier

As seen in the confusion matrix, while 75 percent of cargos were correctly classified as cargos, 16 percent of them were falsely classified as carriers and 9 percent were falsely classified as cruises. Similarly, in the case of carriers, while 73 percent of them were correctly classified as carriers, 8 percent and 19 percent of them were falsely classified as cargos and cruises respectively. Lastly, cruises were correctly classified 69 percent of the times, while they were falsely classified as cargos and carriers 16 and 16 percent respectively.

4.4.5 Key Takeaways

1. The results of the CNN Classifier implemented were satisfactory, yielding an overall accuracy of 73.5 percent.
2. This enabled the images to be labelled properly, once they were passed through the YOLOv4 model, as discussed previously.
3. With the completion of this particular phase, both the objectives – object detection and object classification have successfully been met using the YOLOv4 and CNN Classifier models respectively.

5. PROJECT TIMELINE

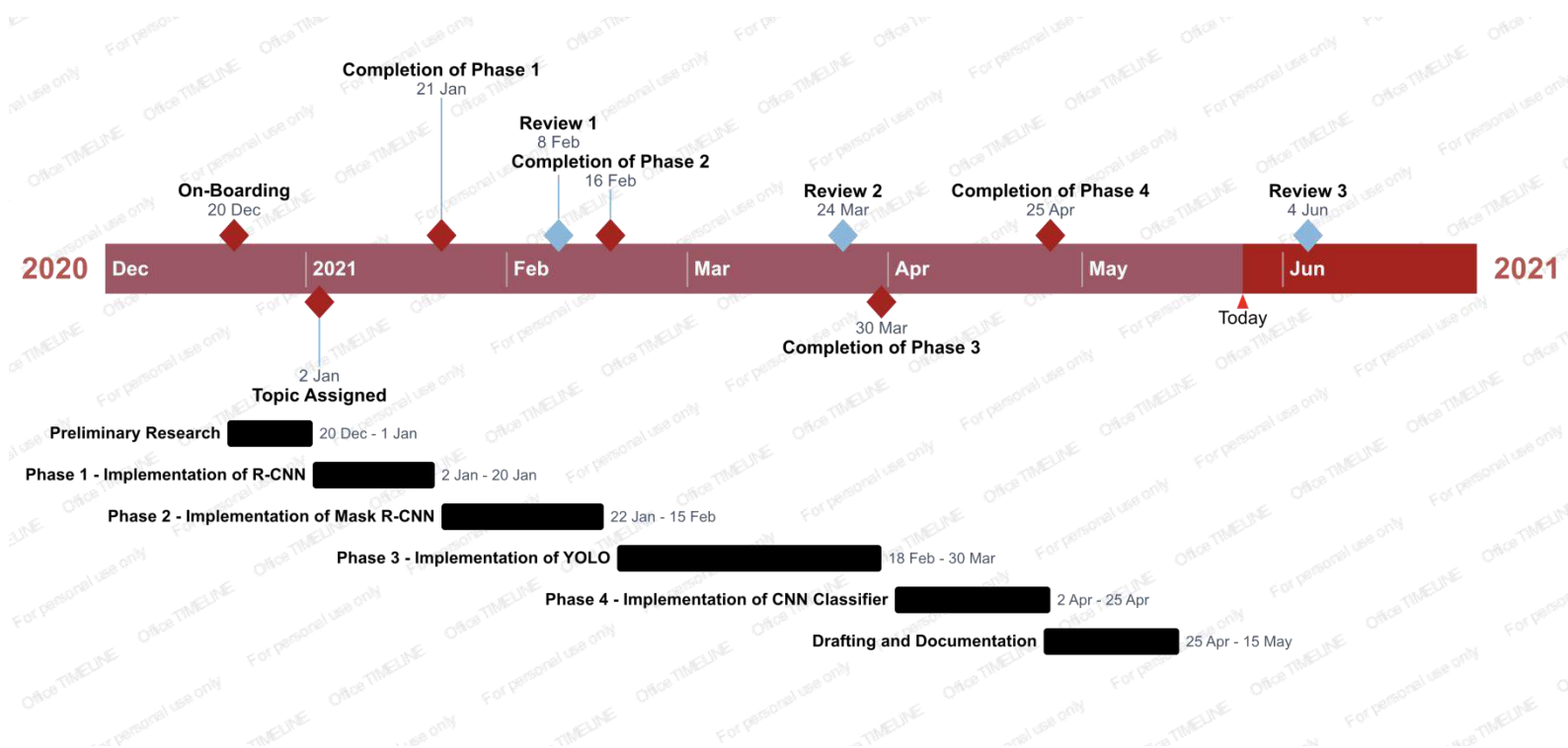


Fig. 49 – Project Timeline

6. SUMMARY

In this project, two primary tasks have been carried out, namely object detection/localisation and object classification. In order to meet this objective, machine learning models such as R-CNNs, Mask R-CNNs, YOLOv4 and CNN Classifiers have been implemented during the course of this project. This has also been broken down into different phases as seen by the project timeline. The final results have been obtained by clubbing the YOLOv4 model with a mAP of 0.82 and a CNN Classifier with accuracy of 73.5 percent, in order to detect and classify maritime objects (cargos, carriers, cruises, and ships) from the MARVEL Maritime dataset. The research still has a lot of scope and the clubbed YOLOv4 and CNN Classifier can be applied to SAR and Optical Sensing Images in the future. This project has been carried out under the supervision of Dr. Rajesh and Shri. Dhipu from DRDO, Bangalore and Dr. Valarmathi from VIT-Vellore for a span of 6 months from December 2020 to May 2021.

7. REFERENCES

- [1] He, H.; Lin, Y.; Chen, F.; Tai, H.-M.; Yin, Z. Inshore Ship Detection in Remote Sensing Images via Weighted Pose Voting. *IEEE Trans. Geosci. Remote Sens.* 2017, 55, 1–17.
- [2] Su, H.; Wei, S.; Liu, S.; Liang, J.; Wang, C.; Shi, J.; Zhang, X. HQ-ISNet: High-Quality Instance Segmentation for Remote Sensing Imagery. *Remote Sens.* 2020, 12, 989.
- [3] Dong, C.; Liu, J.; Xu, F. Ship Detection in Optical Remote Sensing Images Based on Saliency and a Rotation-Invariant Descriptor. *Remote Sens.* 2018, 10, 400.
- [4] M. A. Fischler and R. A. Elschlager, "The Representation and Matching of Pictorial Structures," in *IEEE Transactions on Computers*, vol. C-22, no. 1, pp. 67-92, Jan. 1973, doi: 10.1109/T-C.1973.223602.
- [5] Everingham, M., Gool, L. V., Williams, C., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *IJCV*, 88(2), 303–338.
- [6] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. (2015). ImageNet large scale visual recognition challenge. *IJCV*, 115(3), 211–252.
- [7] Hinton, G., & Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786), 504–507.
- [8] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521, 436–444.
- [9] Grauman, K., & Leibe, B. (2011). Visual object recognition. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 5(2), 1–181.
- [10] Zhang, X., Yang, Y., Han, Z., Wang, H., & Gao, C. (2013). Object class detection: A survey. *ACM Computing Surveys*, 46(1), 10:1–10:53.
- [11] Liu, L., Ouyang, W., Wang, X. et al. Deep Learning for Generic Object Detection: A Survey. *Int J Comput Vis* **128**, 261–318 (2020). <https://doi.org/10.1007/s11263-019-01247-4>
- [12] Geronimo, D., Lopez, A. M., Sappa, A. D., & Graf, T. (2010). Survey of pedestrian detection for advanced driver assistance systems. *IEEE TPAMI*, 32(7), 1239–1258.
- [13] Dollar, P., Wojek, C., Schiele, B., & Perona, P. (2012). Pedestrian detection: An evaluation of the state of the art. *IEEE TPAMI*, 34(4), 743–761.
- [14] Yang, M., Kriegman, D., & Ahuja, N. (2002). Detecting faces in images: A survey. *IEEE TPAMI*, 24(1), 34–58.
- [15] Zafeiriou, S., Zhang, C., & Zhang, Z. (2015). A survey on face detection in the wild: Past, present and future. *Computer Vision and Image Understanding*, 138, 1–24.
- [16] Sun, Z., Bebis, G., & Miller, R. (2006). On road vehicle detection: A review. *IEEE TPAMI*, 28(5), 694–711.

- [17] Ye, Q., & Doermann, D. (2015). Text detection and recognition in imagery: A survey. *IEEE TPAMI*, 37(7), 1480–1500.
- [18] Ponce, J., Hebert, M., Schmid, C., & Zisserman, A. (2007). *Toward category level object recognition*. Berlin: Springer.
- [19] Dickinson, S., Leonardis, A., Schiele, B., & Tarr, M. (2009). *The evolution of object categorization and the challenge of image abstraction in object categorization: Computer and human vision perspectives*. Cambridge: Cambridge University Press.
- [20] Galleguillos, C., & Belongie, S. (2010). Context based object categorization: A critical survey. *Computer Vision and Image Understanding*, 114, 712–722.
- [21] Andreopoulos, A., & Tsotsos, J. (2013). 50 years of object recognition: Directions forward. *Computer Vision and Image Understanding*, 117(8), 827–891.
- [22] Grauman, K., & Leibe, B. (2011). Visual object recognition. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 5(2), 1–181.
- [23] Zhang, X., Yang, Y., Han, Z., Wang, H., & Gao, C. (2013). Object class detection: A survey. *ACM Computing Surveys*, 46(1), 10:1–10:53.
- [24] Li, Y., Wang, S., Tian, Q., & Ding, X. (2015b). Feature representation for statistical learning-based object detection: A review. *Pattern Recognition*, 48(11), 3542–3559.
- [25] Borji, A., Cheng, M., Jiang, H., & Li, J. (2014). Salient object detection: A survey, 1, 1–26.
- [26] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE TPAMI*, 35(8), 1798–1828.
- [27] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521, 436–444.
- [28] Litjens, G., Kooi, T., Bejnordi, B., Setio, A., Ciompi, F., Ghafoorian, M., et al. (2017). A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42, 60–88.
- [29] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., et al. (2018). Recent advances in convolutional neural networks. *Pattern Recognition*, 77, 354–377.
- [30] Liu, L., Ouyang, W., Wang, X. et al. Deep Learning for Generic Object Detection: A Survey. *Int J Comput Vis* **128**, 261–318 (2020). <https://doi.org/10.1007/s11263-019-01247-4>
- [31] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *CVPR*, 2014.
- [32] X. Ren and D. Ramanan, “Histograms of sparse codes for object detection,” in *CVPR*, 2013.

Intelligent Ship Detection and Classification on Remote Sensing Images using Deep Learning

Ananya K R
School of Electronics Engineering
Vellore institute of Technology
Vellore, India
kr.ananyaa@gmail.com

Dr. J Valarmathi
School of Electronics Engineering
Vellore Institute of technology
Vellore, India
jvalarmathi@vit.ac.in

Dr. R Rajesh
Scientist-'F'
Centre for Airborne Systems - DRDO
Bangalore, India
rajesh81r@gmail.com

Shri Dhipu T.M
Scientist
Centre for Airborne Systems-DRDO
Bangalore, India
dhipuganesh@gmail.com

Abstract—Civil and Military surveys today increasingly rely on the intelligent detection and classification of ships from images. However, this job comes with a lot of hassles since various disturbances are present in the sea such as clouds, mist, islands, coastlines, ripples, and so on. Moreover, two primary setbacks in this regard are cluttered image scenes and varying ship sizes. This paper utilizes machine learning and neural network techniques clubbed with numerous methodologies to carry out ship detection from images with ease. The advantages and disadvantages of different detection and classification techniques of ship detection are highlighted as well. Machine learning techniques such as Region-Based Convolutional Neural Networks (R-CNNs), Mask R-CNNs, You Only Look Once (YOLO), and Convolutional Neural Network (CNN) Image Classifiers are applied on numerous datasets such as Dat-Tran's Raccoon dataset, Kaggle's Airbus dataset, and MARVEL Maritime dataset to arrive at key conclusions in this paper.

Index Terms—Object detection, R-CNN, Mask R-CNN, YOLO, CNN Classifier

I. INTRODUCTION

A. Objective

There is a growing need for effective ship recognition and detection for ensuring maritime security and civil management. Apart from this, there are a wide range of applications for ship detection as well right from traffic monitoring, and the defence of territory and naval battles, to harbour surveillance, fishery management, and sea pollution management [1].

Under the guidance of Defence Research and Development Organisation (DRDO), Bangalore, this paper aims to develop an effective and accurate methodology using machine learning algorithms and models such as R-CNNs, Mask R-CNNs, YOLO, and CNN Classifiers to meet two primary objectives:

- Object Detection: to draw a bounding box around the most feasible position of the object/ship in images.
- Classification: to classify the detected maritime object/ship into one of the classes, namely, cargo, cruise, carrier, and so on.

B. Motivation

In the past few years, deep learning algorithms have seen massive growth in development and application in numerous fields, thanks to the continuous enhancement of hardware computing power and other technologies. Today, deep learning-based methods are widely used to detect common objects in daily life and achieve extremely high performance.

Moreover, satellite and aerial remote sensing technology have developed rapidly as well, and optical remote sensing images can provide detailed information with extremely high resolution [2]. Therefore, ship detection has been garnering a lot of attention in the field of optical remote sensing. Due to the large differences between the material of the ship and the sea surface in the radar images, it is easier to detect the ship target in synthetic aperture radar (SAR) images. SAR can work under all weather conditions and various climatic conditions, so ship detection is mostly completed in the SAR images. The optical remote sensing images provide more information and are easy to understand, due to which they have been preferred over SAR images in a lot of domains [3]. In addition, numerous satellites and unmanned aerial vehicles (UAVs) have made it possible to obtain massive high-resolution optical remote sensing images on the sea. Therefore, more detailed information can be obtained to detect the ship in the optical remote sensing images. Ship detection and tracking also play an important part in marine target monitoring.

C. Background

The goal of object detection is to determine whether there are any instances of objects from given categories (such as humans, cars, bicycles, dogs, or cats) in an image and, if present, to return the spatial location and extent of each object instance (e.g., via a bounding box [5]; [6]). As the cornerstone of image understanding and computer vision, object detection forms the basis for solving complex or high-level vision tasks such as segmentation, scene understanding, object tracking, image captioning, event detection, and activity

recognition. Object detection supports a wide range of applications in numerous fields. Recently, deep learning techniques ([7]; [8]) have emerged as powerful methods for learning feature representations automatically from data. In particular, these techniques have provided major improvements in object detection, as illustrated in Fig. 1, where VOC2012 stands for The Pascal Visual Object Classes Challenge conducted in 2012, and ILSVRC stands for ImageNet Large Scale Visual Recognition Challenge.

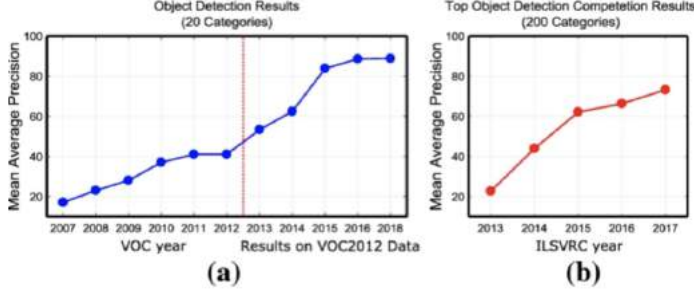


Fig. 1. An overview of recent object detection performance: a significant improvement can be observed in performance (measured as mean average precision) since the arrival of deep learning in 2012. (a) Detection results of winning entries in the VOC2007-2012 competitions, and (b) top object detection competition results in ILSVRC2013-2017

As illustrated in Fig. 2, object detection can be grouped into one of two types ([9]; [10]): detection of specific instances versus the detection of broad categories. The first type aims to detect instances of a particular object (such as Donald Trump's face, the Eiffel Tower, or a neighbour's dog), essentially a matching problem. The goal of the second type is to detect (usually previously unseen) instances of some predefined object categories (for example humans, cars, bicycles, and dogs). Historically, much of the effort in the field of object detection has focused on the detection of a single category (typically faces and pedestrians) or a few specific categories. In contrast, over the past several years, the research community has started moving towards the more challenging goal of building general purpose object detection systems where the breadth of object detection ability rivals that of humans.

II. PROJECT DESCRIPTION AND GOALS

Object detection is used to determine where objects are located in a given image, i.e., object localization, and which category each object belongs to, known as object classification.

A. Object Localisation

Humans can easily detect and identify objects present in an image. The human visual system is fast and accurate and can perform complex tasks like identifying multiple objects and detect obstacles with little conscious thought. With the availability of large amounts of data, faster GPUs, and better algorithms, computers can easily be trained to detect and classify multiple objects within an image with high accuracy.

An image classification or image recognition model simply detect the probability of an object in an image. In contrast

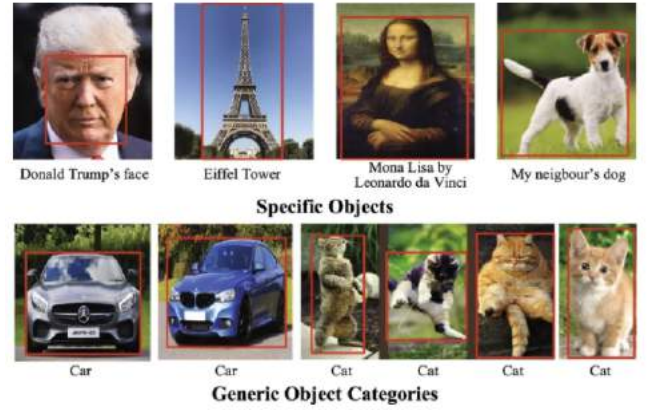


Fig. 2. Object detection includes localizing instances of a particular object (top), as well as generalizing to detecting object categories in general (bottom).

to this, object localization refers to identifying the location of an object in the image. An object localization algorithm will output the coordinates of the location of an object with respect to the image. In computer vision, the most popular way to localize an object in an image is to represent its location with the help of bounding boxes. Fig. 3 shows an example of a bounding box.

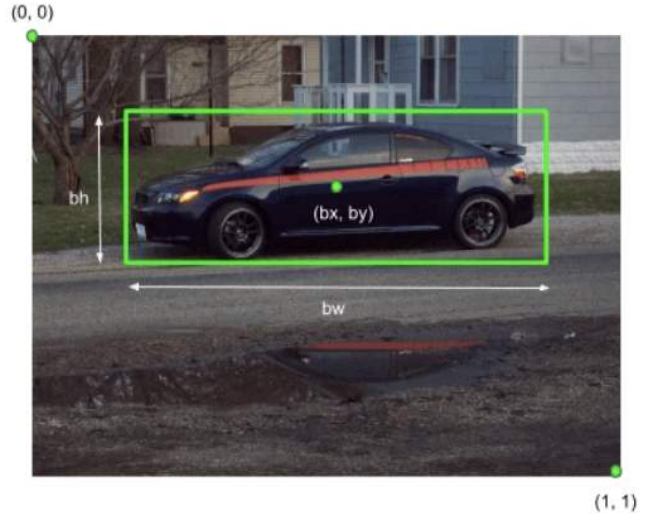


Fig. 3. Bounding box representation used for object localization

A bounding box can be initialized using the following parameters:

- bx, by : coordinates of the centre of the bounding box
- bw : width of the bounding box w.r.t the image width
- bh : height of the bounding box w.r.t the image height

A brief summary of the various machine learning algorithms used to carry out object localisation/bounding-box representation over the years can be presented in the flowchart in Fig. 4.

In this paper, the primary focus of interest is object localisation or bounding box detection using three major and proved

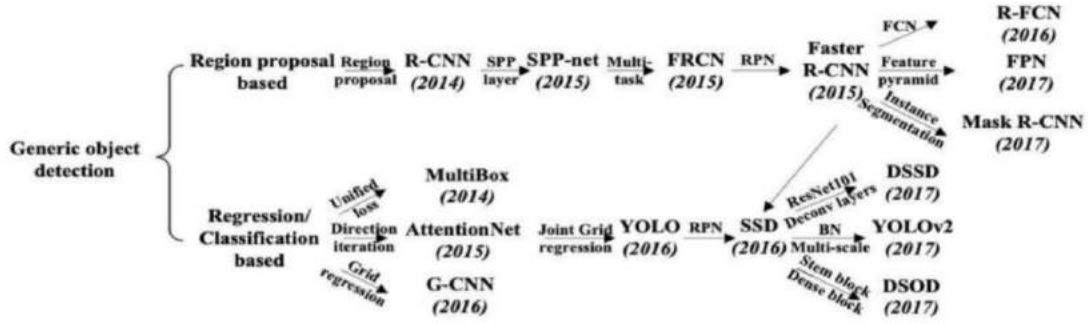


Fig. 4. Evolution of Deep Learning Object Detection/Localisation techniques over the years

to be widely successful machine learning algorithms as seen in the following sections.

1) *R-CNN*: R-CNN or Regional Convolutional Neural Networks were majorly adopted when there were two pressing requirements persistent in the field of object localisation - (a) to improve the quality of candidate boundary boxes, and (b) to use a deep architecture for extraction of high-level features. To solve these requirements, R-CNN [31] was proposed by Ross Girshick in 2014 and obtained a mean average precision (mAP) of 53.3 percent with more than 30 percent improvement over the previous best result.

Fig. 5 shows the flowchart of R-CNN, which can be divided into three stages as follows:

- **Region proposal generation**: The R-CNN adopts selective search to generate about 2k region proposals for each image. The selective search method relies on simple bottom-up grouping and saliency cues to provide more accurate candidate boxes of arbitrary sizes quickly and to reduce the searching space in object detection.
- **CNN based deep feature extraction**: In this stage, each region proposal is warped or cropped into a fixed resolution and the CNN module in is utilized to extract a 4096- dimensional feature as the final representation. Due to large learning capacity, dominant expressive power and hierarchical structure of CNNs, a high-level, semantic and robust feature representation for each region proposal can be obtained.
- **Classification and localization**: With pre-trained category specific linear SVMs for multiple classes, different region proposals are scored on a set of positive regions and background (negative) regions. The scored regions are then adjusted with bounding box regression and filtered with a greedy non-maximum suppression (NMS) to produce final bounding boxes for preserved object locations.

The advantages and disadvantages of the R-CNN model have been discussed in the later part of the paper, after its practical implementation on Dan-Tran's Raccoon Dataset.

2) *Mask R-CNN*: Instance segmentation is a challenging task that requires detecting all objects in an image and segmenting each instance (semantic segmentation). These two tasks are usually regarded as two independent processes. The most viable option before the discovery of Mask R-CNNs, i.e.,

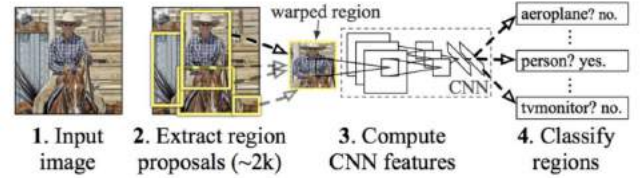


Fig. 5. The three stages of a typical R-CNN model

the multi-task scheme had a disadvantage that would create spurious edge and exhibit systematic errors on overlapping instances.

To solve this problem, parallel to the already existing branches in Faster R-CNN for classification and bounding box regression, the Mask R-CNN adds an additional branch to predict segmentation masks in a pixel-to-pixel manner as seen in Fig. 6. Different from the other two branches which are inevitably collapsed into short output vectors by FC layers, the segmentation mask branch encodes an $m \times m$ mask to maintain the explicit object spatial layout. This kind of fully convolutional representation requires fewer parameters but is more accurate than that of traditional R-CNN.

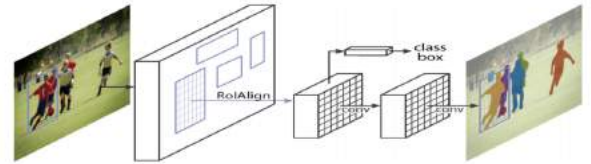


Fig. 6. The Mask R-CNN framework for Instance Segmentation

Formally, besides the two losses for classification and bounding box regression, an additional loss for segmentation mask branch is defined to reach a multi-task loss. This loss is only associated with ground-truth class and relies on the classification branch to predict the category. Because Region of Interest (RoI) pooling, the core operation in Faster R-CNN, performs a coarse spatial quantization for feature extraction, misalignment is introduced between the RoI and the features. It affects classification little because of its robustness to small translations. However, it has a large negative effect on pixel-

to-pixel mask prediction. To solve this problem, Mask R-CNN adopts a simple and quantization-free layer, namely RoIAlign, to preserve the explicit per-pixel spatial correspondence faithfully. RoIAlign is achieved by replacing the harsh quantization of RoI pooling with bilinear interpolation, computing the exact values of the input features at four regularly sampled locations in each RoI bin.

In spite of its simplicity, this seemingly minor change improves mask accuracy greatly, especially under strict localization metrics. Given the Faster R-CNN framework, the mask branch only adds a small computational burden and its cooperation with other tasks provides complementary information for object detection. As a result, Mask R-CNN is simple to implement with promising instance segmentation and object detection results. In a nutshell, Mask R-CNN is a flexible and efficient framework for instance-level recognition, which can be easily generalized to other tasks (e.g., human pose estimation) with minimal modification.

The advantages and disadvantages of the Mask R-CNN model have further been discussed in the later part of the paper, after its practical implementation on the Airbus Kaggle Dataset.

3) *YOLO*: YOLO (You Only Look Once) uses a totally different approach. YOLO is a clever convolutional neural network (CNN) for doing object detection in real-time. The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.

YOLO is popular because it achieves high accuracy while also being able to run in real-time. The algorithm “only looks once” at the image in the sense that it requires only one forward propagation pass through the neural network to make predictions. After non-max suppression (which makes sure the object detection algorithm only detects each object once), it then outputs recognized objects together with the bounding boxes.

With YOLO, a single CNN simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This model has a number of benefits over other object detection methods:

- YOLO is extremely fast
- YOLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance.
- YOLO learns generalizable representations of objects so that when trained on natural images and tested on artwork, the algorithm outperforms other top detection methods.

The major concept of YOLO is to build a CNN network to predict a $(7, 7, 30)$ tensor. It uses a CNN network to reduce the spatial dimension to 7×7 with 1024 output channels at each location. YOLO performs a linear regression using two fully connected layers to make $7 \times 7 \times 2$ boundary box predictions (the middle picture in Figure 9). To make a final prediction,

we keep those with high box confidence scores (greater than 0.25) as our final predictions (the right picture in Fig. 7).



Fig. 7. YOLO Detection Algorithm

The class confidence score for each prediction box is computed as a product of the box confidence score and its conditional class probability. YOLO measures the confidence on both the classification and the localization (where an object is located). Moreover, YOLO has 24 convolutional layers followed by 2 fully connected layers (FC). Some convolution layers use 1×1 reduction layers alternatively to reduce the depth of the feature’s maps. For the last convolution layer, it outputs a tensor with shape $(7, 7, 1024)$. The tensor is then flattened. Using 2 fully connected layers as a form of linear regression, it outputs $7 \times 7 \times 30$ parameters and then reshapes to $(7, 7, 30)$, i.e., 2 boundary box predictions per location. This can be visualised in Fig. 8.

A deeper insight into the advantages and disadvantages of the YOLO model have been discussed in the later part of the paper, after its practical implementation on the MARVEL Maritime Dataset.

B. Object Classification

Apart from implementing object localisation, this project also uses a convolutional neural network for object classification. A Convolutional Neural Network (CNN) is a multi-layered neural network with a special architecture to detect complex features in data. CNNs have been used in image recognition, powering vision in robots, and for self-driving vehicles. Images are made up of pixels. Each pixel is represented by a number between 0 and 255. Therefore, each image has a digital representation which is how computers are able to work with images. In the sections below, the components of a CNN are discussed in detail.

1) *Convolution*: A convolution is a combined integration of two functions that shows us how one function modifies the other. A convolution function has been represented in Equation 1 and 2.

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau \quad (1)$$

$$= \int_{-\infty}^{\infty} f(t - \tau)g(\tau)d\tau \quad (2)$$

There are three important items to mention in this process: the input image, the feature detector, and the feature map. The input image is the image being detected. The feature detector is a matrix, usually 3×3 (it could also be 7×7). A feature detector is also referred to as a kernel or a filter.

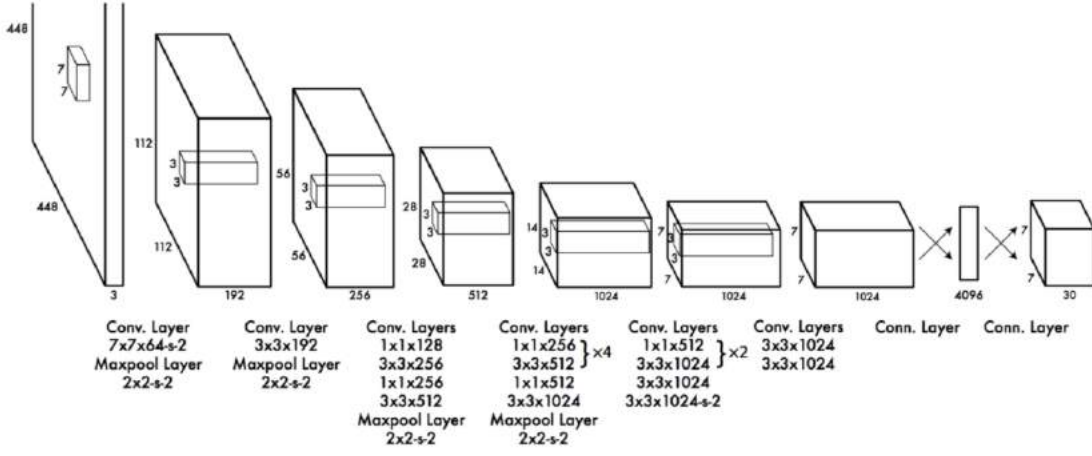


Fig. 8. YOLO Algorithm Architecture

Intuitively, the matrix representation of the input image is multiplied element-wise with the feature detector to produce a feature map, also known as a convolved feature or an activation map. The aim of this step is to reduce the size of the image and make processing faster and easier. Some of the features of the image are lost in this step.

However, the main features of the image that are important in image detection are retained. These features are the ones that are unique to identifying that specific object. For example, each animal has unique features that enable to identify it. The way the loss of image information is reduced by having many feature maps. Each feature map detects the location of certain features in the image.

2) *The Rectified Linear Unit*: The Rectifier Linear Unit (ReLU) function is used to increase non-linearity in the CNN. Images are made of different objects that are not linear to each other. Without applying this function, the image classification will be treated as a linear problem while it is actually a non-linear one.

3) *Pooling*: Spatial invariance is a concept where the location of an object in an image doesn't affect the ability of the neural network to detect its specific features. Pooling enables the CNN to detect features in various images irrespective of the difference in lighting in the pictures and different angles of the images.

There are different types of pooling, for example, max pooling and min pooling. Max pooling works by placing a matrix of 2x2 on the feature map and picking the largest value in that box. The 2x2 matrix is moved from left to right through the entire feature map picking the largest value in each pass. These values then form a new matrix called a pooled feature map. Max pooling works to preserve the main features while also reducing the size of the image. This helps reduce overfitting, which would occur if the CNN is given too much information, especially if that information is not relevant in classifying the image.

4) *Flattening*: Once the pooled featured map is obtained, the next step is to flatten it. Flattening involves transforming the entire pooled feature map matrix into a single column which is then fed to the neural network for processing.

5) *Full Connection*: After flattening, the flattened feature map is passed through a neural network. This step is made up of the input layer, the fully connected layer, and the output layer. The fully connected layer is similar to the hidden layer in ANNs but in this case it's fully connected. The output layer is where we get the predicted classes. The information is passed through the network and the error of prediction is calculated. The error is then backpropagated through the system to improve the prediction.

The final figures produced by the neural network don't usually add up to one. However, it is important that these figures are brought down to numbers between zero and one, which represent the probability of each class. This is the role of the SoftMax function.

$$\sigma : \mathbb{R}^K \rightarrow (0, 1)^K \quad (3)$$

$$(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \text{ for } j = 1, \dots, K \quad (4)$$

A deeper insight into the advantages and disadvantages of the CNN Classifier have been discussed in the later part of the paper, after its practical implementation on the MARVEL Maritime Dataset.

III. DESIGN APPROACH AND DETAILS

This paper has been broken down into four phases, with respect to the algorithms and dataset chosen, namely the implementation of R-CNN, Mask R-CNN, YOLOv4 and CNN Classifier. The details regarding the objectives, datasets, methodologies, code snippets and results have been outlined in the following sections.

A. Implementation of R-CNN

1) *Objective*: R-CNNs serve as a great starting point to solve any major object detection problem and to brush up skills of TensorFlow and Keras libraries, due to its well-maintained documentation and varied use cases.

2) *Dataset*: The dataset chosen to reach the objective was the Dat-Tran's 2017 Raccoon Dataset, that consists of 196 images of raccoons in varied sizes, shapes and orientation. Owing to more than a single raccoon present in a few images, the total number of bounding boxes in the ground-truth dataset equals to 213. As it can be observed, this is a single class problem, and a great starting point to perform and explore object detection. The images in this dataset vary from 0.04 to 2.67 megapixels and the median image size can be approximated at 480 x 360, i.e., 0.18 megapixels. A small snippet of the dataset can be seen in Fig. 9.

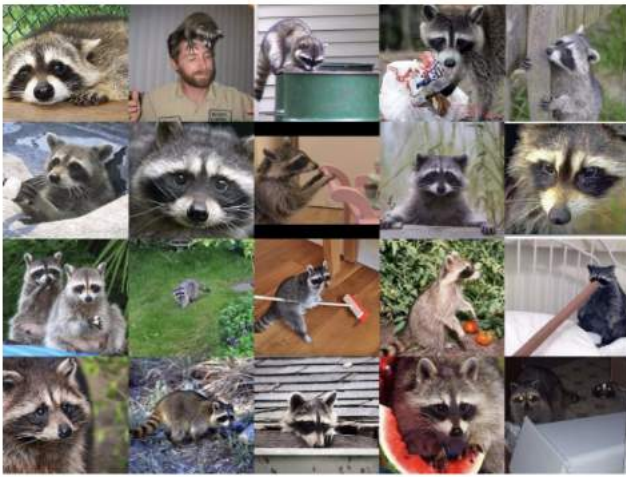


Fig. 9. A snippet of Dat-Tran's Raccoon Dataset

3) *Methodology*: Implementing an R-CNN object detector is a somewhat complex multistep process. A summary of the flow that has been followed has been summarised in the flowchart, as shown in Fig. 10.

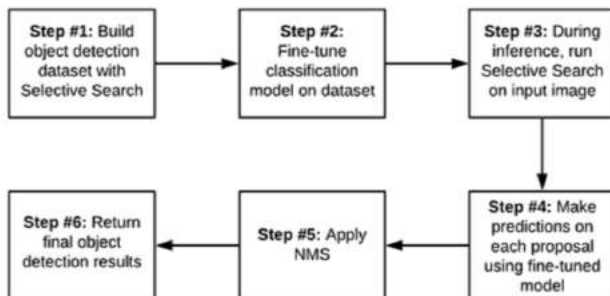


Fig. 10. The Object Detection Workflow

Firstly, Selective Search was implemented, along with a bit of post-processing logic, to identify regions of an input image that do and do not contain a potential object of interest, in this

case, a raccoon. These regions were used as the training data for the object detection model.

Next, a MobileNet (pre-trained on ImageNet) was fine-tuned to classify and recognise objects from the previously curated dataset.

Finally, the last step involves implementing a Python script for inference and prediction by applying Selective Search to an input image, classifying the region proposals generated, and then to display the final output of the R-CNN.

The accuracy of the object detection algorithm was measured by using a metric called “Intersection over Union (IOU)”, that can be calculated using the formula presented Fig. 11.

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

Fig. 11. The IOU Equation

Examining the equation in Fig. 11 and observing Fig. 12, it can be seen that Intersection over Union (IOU) is simply a ratio:

- In the numerator, the area of overlap between the predicted bounding box and the ground-truth bounding box is computed.
- The denominator is the area of union, or more simply, the area encompassed by both the predicted bounding box and the ground-truth bounding box.
- Dividing the area of overlap by the area of union yields the final score — the Intersection over Union (hence the name).

In this paper, IOU is used to measure object detection accuracy, including how much a given Selective Search proposal overlaps with a ground-truth bounding box (which is useful when generating positive and negative examples for the training data).

Another important concept that was implemented during the course of this paper is the Non-Maxima Suppression (NMS). Any typical object detection pipeline has one component for generating proposals for classification. Proposals are nothing but the candidate regions for the object of interest. Most of the approaches employ a sliding window over the feature map and assigns foreground/background scores depending on the features computed in that window. The neighbourhood windows have similar scores to some extent and are considered as candidate regions. This leads to hundreds of proposals. As the proposal generation method should have high recall, it is often advised to keep loose constraints in this stage. However,

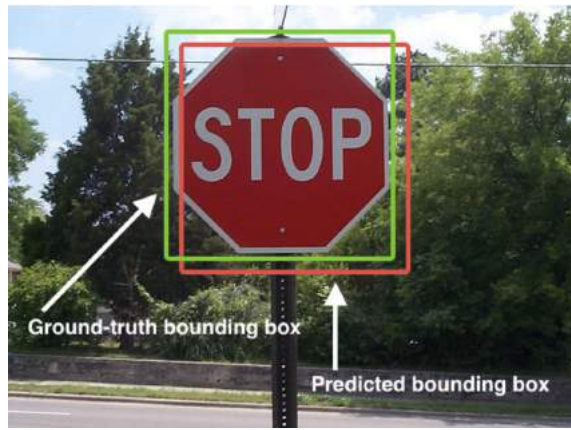


Fig. 12. Ground-truth and predicted bounding boxes of an image

processing these many proposals all through the classification network is cumbersome. This leads to a technique which filters the proposals based on some criteria called Non-maximum Suppression. This phenomenon can be observed in Fig. 13.



Fig. 13. A summary of NMS

4) *Results:* Upon training and testing the model for 5 epochs, results that have been highlighted in Figure 14 were obtained. A graph depicting the training loss and accuracy obtained during every epoch, the model was trained for, has been depicted in Figure 15. Once the accuracy and the loss ceased to change, as seen in the graph, the training of the model was stopped.

	precision	recall	f1-score	support
no_raccoon	1.00	0.98	0.99	440
raccoon	0.97	1.00	0.99	312
accuracy			0.99	752
macro avg	0.99	0.99	0.99	752
weighted avg	0.99	0.99	0.99	752

Fig. 14. Results obtained

Fig. 16 show snippets of a few output images with the bounding box as well as the IOU labelled on it. A key point to note is the importance of Non-Maxima Suppression (NMS). As seen in Fig. 13, NMS plays an invaluable role to eliminate lesser accurate bounding boxes during object detection. A

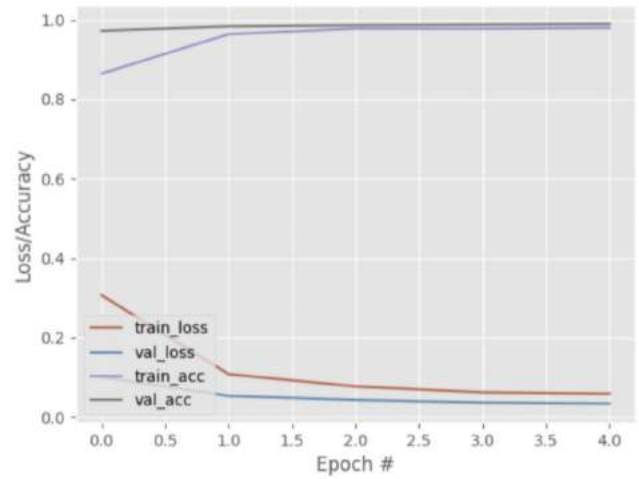


Fig. 15. Training Loss and Accuracy of the model

before-NMS and an after-NMS comparison has been made in Fig. 17. As it can be seen, the redundant bounding boxes were eliminated upon applying NMS.

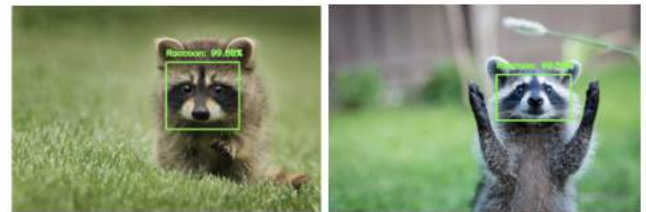


Fig. 16. A few output images of the R-CNN model

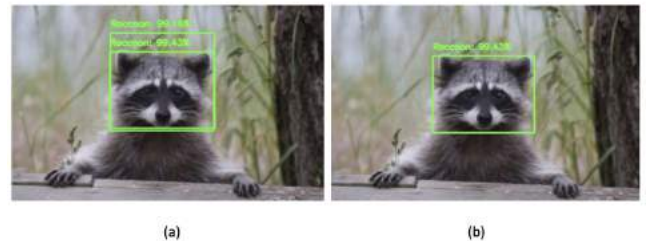


Fig. 17. (a) Before NMS was applied, (b) After NMS was applied

5) *Key Takeaways:* After the implementation of R-CNN, the following conclusions were reached.

- Even though the accuracy of the model obtained was great and the results shown were promising, it can mostly be attributed to the smaller size of the dataset, with easily distinguishable objects in it.
- To reach closer to the objective, the next step is to apply a better algorithm on databases that involve maritime or air-borne objects such as helicopters, aeroplanes and ships.

B. Implementation of Mask R-CNN

1) *Objective*: Upon the successful implementation of R-CNNs, the decision to take one further step in the direction of solving the problem statement of ship detection led to the usage of the moderately-sized dataset of ships satellite images, in contrast to the small-size dataset of Raccoons used previously. The Mask R-CNN algorithm was chosen due to its proven ability to tackle the problem of object detection such as ships and other maritime objects.

2) *Dataset*: A public dataset from Kaggle on the Airbus Ship Detection Challenge was obtained. The dataset contains more than 100 thousand 768×768 images taken from satellite. The total size of the dataset is more than 30 Gb. Along with the images in the dataset is a CSV file that lists all the images ids and their corresponding pixels coordinates. These coordinates represent segmentation bounding boxes of ships. Not having pixel coordinates for an image means that particular image doesn't have any ships. A snippet of the dataset has been visualised in Fig. 18.



Fig. 18. A snippet of the Kaggle Airbus Dataset

3) *Methodology*: Firstly, the dataset is split into training and validation set with 80 percent and 20 percent of images respectively. Since a few images had more than just a single ship, it was important to split each category of number of ships by an 80:20 ratio. This can be observed in Fig. 19. The colour blue denotes the training set and orange denotes the validation set. Upon execution, 34,044 training images and 8,512 validation images were generated. The X-axis denotes the number of ship masks in each image.

The next step of the process was to apply masks to all images using a simple function written in Python. The output ship images with masks have been represented in Fig. 20.

Next, a function was created for encoding and decoding mask on top of every image followed by which, a Mask R-CNN algorithm for segmenting the ships in image with a confidence score between 0 and 1 is set. The time required to mask out ships in the complete dataset was 9.82 seconds. A few encoded and decoded masks are displayed in Fig. 21.

Pre-trained MS COCO weights are used for training the model. Since these weights have already been trained on a large variety of objects, they provide a good place to start learning from. The model was trained for 5 epochs.

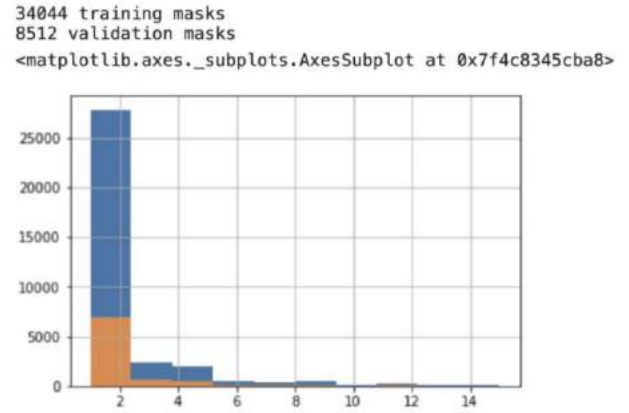


Fig. 19. Graph depicting the 80:20 split



Fig. 20. A few examples of masks generated for ship images

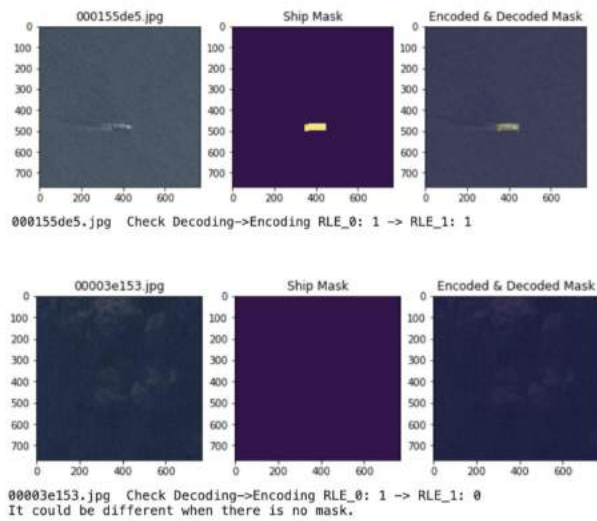


Fig. 21. Encoded and Decoded Ship Masks

4) *Results:* It is found that using mask shape of size 14*14 and Resnet101 backbone, the Mask RCNN algorithm is able to detect and segment ships with a mean average precision of 0.605. It took the model approximately 30 seconds to segment 30 images. The graph depicting training loss versus epoch has been visualised in Fig. 22, while a few snippets of the output images have been presented in Fig. 23, Fig. 24, and Fig. 25.

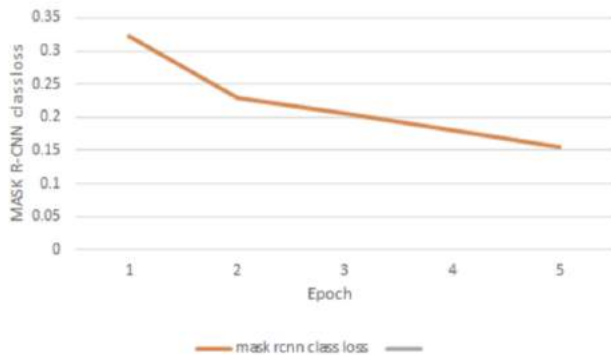


Fig. 22. Encoded and Decoded Ship Masks

5) *Key Takeaways:* After the implementation of Mask R-CNN, the following conclusions were reached.

- Even though the model was trained for 5 epochs till the loss was minimal, a few problems persist. In a few images, the land in and around the ships has also been getting detected, as seen in the Figure 26. A workaround for these false positives needed to be worked out.
- Moreover, MASK R-CNN involves generation of mask as images that takes up a lot of storage space as well as computation power. It is also extremely slow to execute.

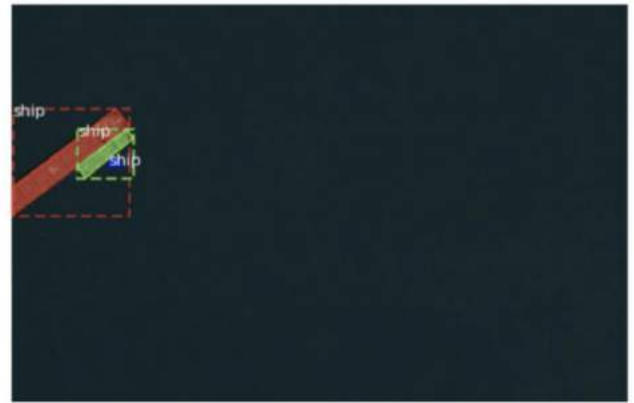


Fig. 23. A snippet of the Mask R-CNN results



Fig. 24. A snippet of the Mask R-CNN results



Fig. 25. A snippet of the Mask R-CNN results



Fig. 26. Drawback of using Mask R-CNN (Generation of False Positives)

C. YOLO

1) *Objective*: For the final phase of the project, the YOLOv4 algorithm was picked since it has several advantages over general classifier-based systems. For example, YOLO looks at the whole image at test time so its predictions are informed by global context in the image. It also makes predictions with a single network evaluation unlike systems like R-CNN which require thousands for a single image. This makes it extremely fast, more than 1000x faster than R-CNN and 100x faster than Mask R-CNN.

2) *Dataset*: After the experimentation with small-sized and medium-sized databases previously, large-scale Image dataset for Maritime Vessels, known as the MARVEL dataset was used this time around. It consists of 2 million user uploaded images and their attributes including vessel identity, type, photograph category and year of built, collected from a community website. It can be categorized into 109 vessel type classes and 26 super-classes by combining heavily populated classes with a semi-automatic clustering scheme. A snippet of the dataset has been presented in Fig. 27.

However, a mini-version of the MARVEL dataset to perform the pre-liminary analysis was used in this paper. This consisted of 2,800 images, spread uniformly across five major maritime classes, namely, cargo, carrier, cruise, military and tanker ships.

3) *Methodology*: Since the MARVEL dataset did not come with pre-assigned labels, like other custom-made datasets, each image had to be labelled manually using a tool known as LabelMe. An open annotation tool developed by MIT, the goal of the LabelMe open-source project was to develop datasets for computer vision and image processing.



Fig. 27. A snippet of the Marvel Maritime Dataset

Since the LabelMe annotation tool saves the output of each image in JSON format, it is mandatory to convert them into a format that YOLO model can interpret. In YOLO labelling format, a .txt file with the same name is created for each image file in the same directory. Each .txt file contains the annotations for the corresponding image file, that is object class, object coordinates, height and width. Most YOLO versions can only read .txt files with a specific format. As a result, to convert the JSON files to .txt files for each corresponding image, the services of the RoboFlow platform were utilised.

As mentioned earlier, YOLO is an object detection algorithm touted to be one of the fastest. It was trained on the COCO dataset and achieved a mean Average Precision (mAP) of 33.0 at the speed of 51ms per image on Titan X GPU, which is commendable. The major highlight of the algorithm is that it divides the input image into several individual grids, and each grid predicts the objects inside it. This way, the whole image is processed at once, and the inference time is reduced.

The YOLO model used in this project is trained on the Darknet Framework, which is an open-source neural network framework written in C and CUDA. It is fast, easy to install, and supports CPU and GPU computation. After the initial configuration and setup, the YOLOv4 darknet model was trained for over 2000 iterations, at the end of which the results in Figure 28 were received.

```
for conf_thresh = 0.25, precision = 0.81, recall = 0.88, F1-score = 0.85
for conf_thresh = 0.25, TP = 243, FP = 57, FN = 32, average IoU = 67.08 %

IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
mean average precision (mAP@0.50) = 0.823406, or 82.34 %
Total Detection Time: 5 Seconds
```

Fig. 28. Results using the YOLOv4 model

The performance metric that was chosen was the mean average precision. The mean average precision (mAP) of a set of queries is defined by the formula in Equation 5, where Q is the number of queries in the set and AveP(q) is the average precision (AP) for a given query, q.

$$MAP = \frac{\sum_{q=1}^Q AveP(q)}{Q} \quad (5)$$

The mAP compares the ground-truth bounding box to the detected box and returns a score between 0 to 1. The higher the score, the more accurate the model is in its detections. With a mean average precision (mAP) value at 0.8234, the results obtained using the YOLO model were commendable. The Loss and Accuracy versus Epoch graph has been visualised in Fig. 29, and few results obtained by the YOLO model have been presented in Fig. 30, Fig. 31 and Fig. 32.

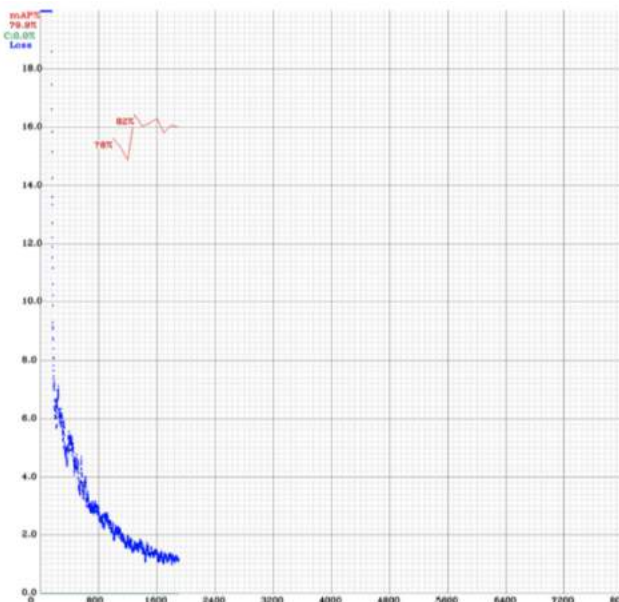
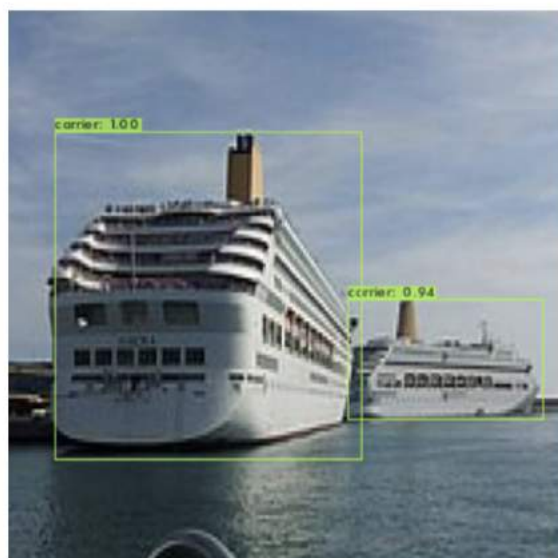


Fig. 29. The loss and accuracy vs epoch of the YOLO model



Truth – Cargo, Cargo; Prediction – Carrier, Carrier

Fig. 30. Results of the YOLO Model



Truth – Carrier, Ship; Predicted – Ship, Cruise

Fig. 31. Results of the YOLO Model



Truth – Cargo, Cargo; Predicted – Carrier, Carrier

Fig. 32. Results of the YOLO Model

4) *Key Takeaways:* After the implementation of YOLO, the following conclusions were reached.

- The YOLO model ran for 2000 iterations and returned a mAP value of 0.823. The mAP value however denotes the bounding box accuracy, and not the label accuracy. As seen in the results snapshots, the labels were not being predicted accurately by the YOLO model.
- The YOLO model thus did a great job at object detection but didn't meet the requirements of object classification.
- The next logical step is to implement a CNN Classifier that will be connected to the YOLO model such that, the images of the YOLO model will be passed on to the CNN Classifier as well.

D. Implementation of CNN Classifier

1) *Objective:* The implementation of the CNN Classifier aims to solve the shortcoming of the YOLOv4 model, and reach the project's objective of image classification. In other words, the CNN Classifier will be used to classify the detected object of the YOLO model into one of the following three classes – cargo, carrier and cruise.

2) *Dataset:* The CNN Classifier uses the MARVEL Maritime dataset, just like the YOLO Model. A snippet of the same has been presented previously in Fig. 27. However, since the LabelMe tool was used to assign labels to the images in a json format, there had to be modifications done to the dataset for it to be utilised by the CNN Classifier.

3) *Methodology:* The dataset had to be first converted into a CNN Classifier format. A CNN Classifier requires all images to be in a folder along with a one-hot encoded .csv file with classification details regarding each image in the folder. To do so, the roboflow platform was utilised. Once the dataset is ready, it is split into a ratio of 80:10:10 to accommodate the train, test and validation sets.

In order to build the CNN Classifier, a mix of ReLU and Sigmoid Activation Functions were used. While the ReLU layer was used alternately, the Sigmoid Activation Function was placed as the last layer due to its ability to tackle multi-label classification efficiently. While the performance parametric was accuracy, the loss parametric was chosen as binary cross entropy – again, due to their suitability to the problem statement of multi label classification. MaxPooling was also implemented.

4) *Results:* The model was trained for 10 epochs, finally resulting in an accuracy of 73.5 percent, validation loss of 0.4645 and a validation accuracy of 79.69 percent.

In order to check the results, a confusion matrix was plotted, as shown in Fig. 33. Confusion matrices represent counts from predicted and actual values. The output “TN” stands for True Negative which shows the number of negative examples classified accurately. Similarly, “TP” stands for True Positive which indicates the number of positive examples classified accurately. The term “FP” shows False Positive value, i.e., the number of actual negative examples classified as positive; and “FN” means a False Negative value which is the number of actual positive examples classified as negative. One of the

most commonly used metrics while performing classification is accuracy. The accuracy of a model (through a confusion matrix) is calculated using the given formula in Equation 6.

$$\text{Accuracy} = \frac{TN + TP}{TN + FP + FN + TP} \quad (6)$$

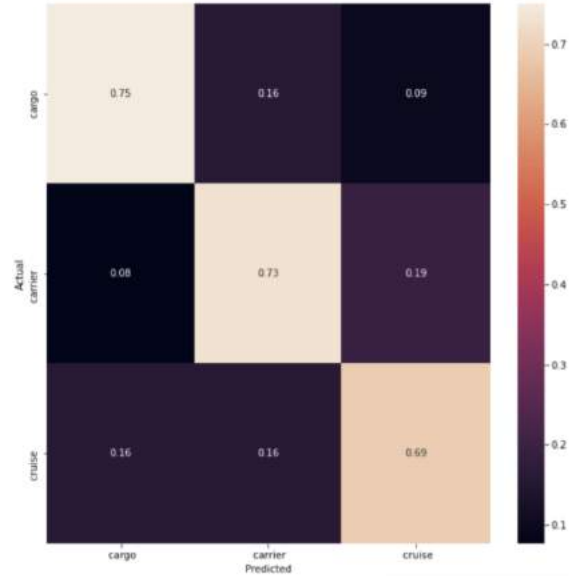


Fig. 33. Confusion Matrix of CNN Classifier

As seen in the confusion matrix, while 75 percent of cargos were correctly classified as cargos, 16 percent of them were falsely classified as carriers and 9 percent were falsely classified as cruises. Similarly, in the case of carriers, while 73 percent of them were correctly classified as carriers, 8 percent and 19 percent of them were falsely classified as cargos and cruises respectively. Lastly, cruises were correctly classified 69 percent of the times, while they were falsely classified as cargos and carriers 16 and 16 percent respectively.

5) *Key Takeaways:* After the implementation of CNN Classifier, the following conclusions were reached.

- The results of the CNN Classifier implemented were satisfactory, yielding an overall accuracy of 73.5 percent.
- This enabled the images to be labelled properly, once they were passed through the YOLOv4 model, as discussed previously.
- With the completion of this particular phase, both the objectives – object detection and object classification have successfully been met using the YOLOv4 and CNN Classifier models respectively.

IV. CONCLUSION

In this paper, two primary tasks have been carried out, namely object detection/localisation and object classification. In order to meet this objective, machine learning models such as R-CNNs, Mask R-CNNs, YOLOv4 and CNN Classifiers have been implemented during the course of this project. This has also been broken down into different phases as seen by

the project timeline. The final results have been obtained by clubbing the YOLOv4 model with a mAP of 0.82 and a CNN Classifier with accuracy of 73.5 percent, in order to detect and classify maritime objects (cargos, carriers, cruises, and ships) from the MARVEL Maritime dataset. The research still has a lot of scope and the clubbed YOLOv4 and CNN Classifier can be applied to SAR and Optical Sensing Images in the future. This project has been carried out under the supervision of Dr. Rajesh and Shri. Dhipu from DRDO, Bangalore and Dr. Valarmathi from VIT-Vellore for a span of 6 months from December 2020 to May 2021.

REFERENCES

- [1] He, H.; Lin, Y.; Chen, F.; Tai, H.-M.; Yin, Z. Inshore Ship Detection in Remote Sensing Images via Weighted Pose Voting. *IEEE Trans. Geosci. Remote Sens.* 2017, 55, 1–17.
- [2] Su, H.; Wei, S.; Liu, S.; Liang, J.; Wang, C.; Shi, J.; Zhang, X. HQ-ISNet: High-Quality Instance Segmentation for Remote Sensing Imagery. *Remote Sens.* 2020, 12, 989.
- [3] Dong, C.; Liu, J.; Xu, F. Ship Detection in Optical Remote Sensing Images Based on Saliency and a Rotation-Invariant Descriptor. *Remote Sens.* 2018, 10, 400.
- [4] M. A. Fischler and R. A. Elschlager, "The Representation and Matching of Pictorial Structures," in *IEEE Transactions on Computers*, vol. C-22, no. 1, pp. 67-92, Jan. 1973, doi: 10.1109/T-C.1973.223602.
- [5] Everingham, M., Gool, L. V., Williams, C., Winn, J., Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *IJCV*, 88(2), 303–338.
- [6] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. (2015). ImageNet large scale visual recognition challenge. *IJCV*, 115(3), 211–252.
- [7] Hinton, G., Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786), 504–507.
- [8] LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep learning. *Nature*, 521, 436–444.
- [9] Grauman, K., Leibe, B. (2011). Visual object recognition. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 5(2), 1–181.
- [10] Zhang, X., Yang, Y., Han, Z., Wang, H., Gao, C. (2013). Object class detection: A survey. *ACM Computing Surveys*, 46(1), 10:1–10:53.
- [11] Liu, L., Ouyang, W., Wang, X. et al. Deep Learning for Generic Object Detection: A Survey. *Int J Comput Vis* 128, 261–318 (2020). <https://doi.org/10.1007/s11263-019-01247-4>
- [12] Geronimo, D., Lopez, A. M., Sappa, A. D., Graf, T. (2010). Survey of pedestrian detection for advanced driver assistance systems. *IEEE TPAMI*, 32(7), 1239–1258.
- [13] Dollar, P., Wojek, C., Schiele, B., Perona, P. (2012). Pedestrian detection: An evaluation of the state of the art. *IEEE TPAMI*, 34(4), 743–761.
- [14] Yang, M., Kriegman, D., Ahuja, N. (2002). Detecting faces in images: A survey. *IEEE TPAMI*, 24(1), 34–58.
- [15] Zafeiriou, S., Zhang, C., Zhang, Z. (2015). A survey on face detection in the wild: Past, present and future. *Computer Vision and Image Understanding*, 138, 1–24.
- [16] Sun, Z., Bebis, G., Miller, R. (2006). On road vehicle detection: A review. *IEEE TPAMI*, 28(5), 694–711.
- [17] Ye, Q., Doermann, D. (2015). Text detection and recognition in imagery: A survey. *IEEE TPAMI*, 37(7), 1480–1500.
- [18] Ponce, J., Hebert, M., Schmid, C., Zisserman, A. (2007). *Toward category level object recognition*. Berlin: Springer.
- [19] Dickinson, S., Leonardis, A., Schiele, B., Tarr, M. (2009). *The evolution of object categorization and the challenge of image abstraction in object categorization: Computer and human vision perspectives*. Cambridge: Cambridge University Press.
- [20] Galleguillos, C., Belongie, S. (2010). Context based object categorization: A critical survey. *Computer Vision and Image Understanding*, 114, 712–722.
- [21] Andreopoulos, A., Tsotsos, J. (2013). 50 years of object recognition: Directions forward. *Computer Vision and Image Understanding*, 117(8), 827–891.
- [22] Grauman, K., Leibe, B. (2011). Visual object recognition. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 5(2), 1–181.
- [23] Zhang, X., Yang, Y., Han, Z., Wang, H., Gao, C. (2013). Object class detection: A survey. *ACM Computing Surveys*, 46(1), 10:1–10:53.
- [24] Li, Y., Wang, S., Tian, Q., Ding, X. (2015b). Feature representation for statistical learning-based object detection: A review. *Pattern Recognition*, 48(11), 3542–3559.
- [25] Borji, A., Cheng, M., Jiang, H., Li, J. (2014). Salient object detection: A survey, 1, 1–26.
- [26] Bengio, Y., Courville, A., Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE TPAMI*, 35(8), 1798–1828.
- [27] LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep learning. *Nature*, 521, 436–444.
- [28] Litjens, G., Kooi, T., Bejnordi, B., Setio, A., Ciompi, F., Ghafoorian, M., et al. (2017). A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42, 60–88.
- [29] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., et al. (2018). Recent advances in convolutional neural networks. *Pattern Recognition*, 77, 354–377.
- [30] Liu, L., Ouyang, W., Wang, X. et al. Deep Learning for Generic Object Detection: A Survey. *Int J Comput Vis* 128, 261–318 (2020). <https://doi.org/10.1007/s11263-019-01247-4>
- [31] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014.
- [32] X. Ren and D. Ramanan, "Histograms of sparse codes for object detection," in *CVPR*, 2013.