A Project Report

on

Vessel detection from space borne images

Submitted in partial fulfilment of the requirements for the award of degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

17WH1A0507	**Ms. K NIKITHA**
17WH1A0547	**Ms. P SAI PREETHI**
18WH5A0502	**Ms. G NIKITHA**

Under the esteemed guidance of
Mrs. Chittiboina Jagadeeswari
Assistant Professor

Department of Computer Science and Engineering

BVRIT HYDERABAD College of Engineering for Women
(NBA Accredited EEE, ECE, CSE, IT B.Tech. Courses,
Accredited by NAAC with 'A' Grade)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

May 2021

BVRIT HYDERABAD College of Engineering for Women

(NBA Accredited EEE, ECE, CSE, IT B.Tech. Courses,
Accredited by NAAC with 'A' Grade)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the project work report entitled "**Vessel detection from space borne images**" is a bonafide work carried out by **Ms. K. Nikitha (17wh1a0507), Ms. P. Sai Preethi (17wh1a0547), Ms. G. Nikitha (18wh5a0502)** in partial fulfilment for the award of B. Tech degree in **Computer Science & Engineering**, **BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad under my guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Internal Guide	**Head of the Department**
Mrs. Chittiboina Jagadeeswari	**Dr. Srinivasa Reddy Konda**
Assistant Professor, CSE	**Professor and HOD, CSE**

External Examiner

DECLARATION

We hereby declare that the work presented in this project work entitled **"Vessel detection from space borne images"** submitted towards completion of Project work in IV Year of B.Tech of CSE at **BVRIT HYDERABAD College of Engineering for Women,** Hyderabad is an authentic record of our original work carried out under the guidance of **Mrs. Chittiboina Jagadeeswari, Assistant Professor, Department of CSE.**

Roll No	**Name**	**Signature**
17WH1A0507	Ms. K Nikitha	
17WH1A0547	Ms. P Sai Preethi	
18WH5A0502	Ms. G Nikitha	

ACKNOWLEDGEMENT

We would like to express our sincere thanks to **Dr. K. V. N. Sunitha**, **Principal, BVRIT HYDERABAD College of Engineering for Women,** for her support by providing the working facilities in the college.

Our sincere thanks and gratitude to **Dr. Srinivasa Reddy Konda,** Head, Department of CSE, BVRIT HYDERABAD College of Engineering for Women, for all timely support and valuable suggestions during the period of our project.

We are extremely thankful to our Internal Guide, **Mrs. Chittiboina Jagadeeswari, Assistant Professor, CSE,** BVRIT HYDERABAD College of Engineering for Women for her constant guidance and encouragement throughout the project.

We would like to record my sincere thankfulness to the major project coordinator **Dr. Ganti Naga Satish, Professor, CSE,** BVRIT HYDERABAD College of Engineering for Women for his valuable guidance and skillful management.

17WH1A0507	**Ms. K Nikitha**
17WH1A0547	**Ms. P Sai Preethi**
18WH5A0502	**Ms. G Nikitha**

TABLE OF CONTENTS

ABSTRACT

Vessel detection from remote sensing imagery is a crucial application for maritime security which includes traffic surveillance, protection against illegal fisheries, oil discharge control and sea pollution monitoring. This is currently manually being done through the use of an Automated Identification System (AIS), which uses Very High Frequency (VHF) radio frequencies. AIS is very effective at monitoring ships which are legally required to install a VHF transponder, but fail to detect those which are not, also in some cases people disconnect their transponder which is a quite common case. Satellite imagery can be used in these cases as it removes the human element out of the loop. Synthetic Aperture Radar (SAR) imagery uses radio waves to image the Earth's surface.

A number of factors can make vessel detection and segmentation a challenging task. Some of these factors include flaws in image quality like uneven brightness, obstruction, and also the fact that there are many images which have similar shape, color and texture.

The goal of this work is to build an algorithm to detect and segment ships in satellite images. The model uses custom Mask RCNN with Resnet-50 as a backbone to detect and segment ships in the satellite images. This model is very effective and hence can be used to detect and segment ships in real-time applications.

LIST OF FIGURES

1. INTRODUCTION

Vessel detection from satellite images is an important application for maritime applications like ship traffic surveillance, protection against illegal fisheries, oil discharge control and sea pollution monitoring. Machine Learning techniques are used to detect and segment vessels from satellite images. It is very effective and can be used for real time applications. This can be of great help to the maritime security and offshore operations in the surveillance, energy and military sector.

1.1 OBJECTIVE

Vessel detection is currently being manually done through AIS which uses very high frequency (VHF) radio frequencies. To remove the human element out of loop satellite images are used to detect vessels. The goal of the system is to build an algorithm to effectively detect and segment vessels from the satellite images. Using a public data set, a deep-learning network is trained to detect the images and segment them with better accuracy.

1.2 METHODOLOGY

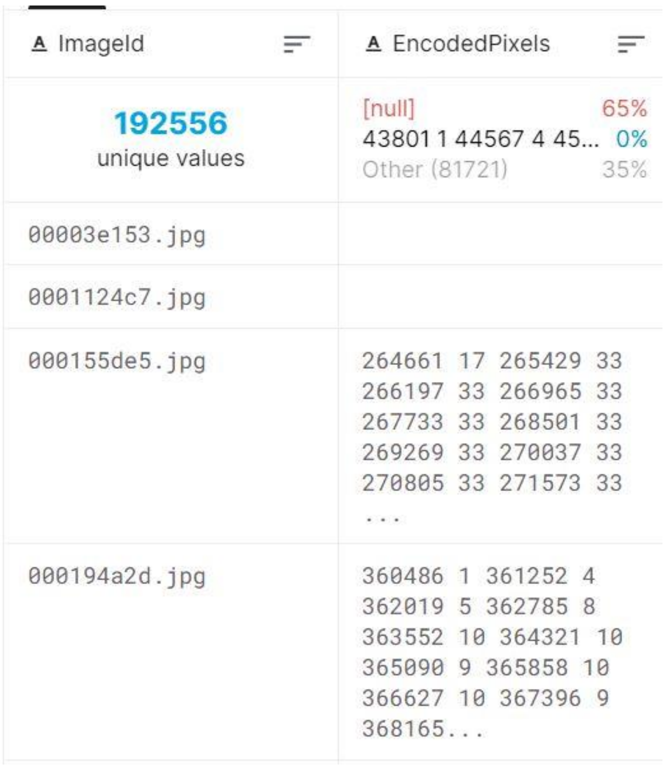
To detect and segment vessels a large collection of satellite image data is required. The images are downloaded from the Airbus Ship detection Challenge. In this section the methodology followed is discussed in detail

1.2.1 DATASET

In this work, a public dataset from Kaggle from the Airbus Ship Detection Challenge is obtained. The dataset contains more than 200 thousand 768 × 768 high-resolution images taken from satellite out of which 193k images are used for training and 15.6k images are used for testing. The total size of the dataset is 29.25 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.

Dataset Description

The dataset taken from Kaggle has train_v2 folder which is training data and has 193k high-resolution images and test_v2 folder which is testing data and has 15.6k 768 *768 high-resolution images. The train_ship_segmentations_v2.csv file has image id and its encoded pixel coordinates.



ImageId	EncodedPixels
192556 unique values	[null] 65% 43801 1 44567 4 45... 0% Other (81721) 35%
00003e153.jpg	
0001124c7.jpg	
000155de5.jpg	264661 17 265429 33 266197 33 266965 33 267733 33 268501 33 269269 33 270037 33 270805 33 271573 33 ...
000194a2d.jpg	360486 1 361252 4 362019 5 362785 8 363552 10 364321 10 365090 9 365858 10 366627 10 367396 9 368165...

Fig 1.2.1 Dataset

1.3 THE PROPOSED MODEL

The model has two tasks namely detection and segmentation. First phase of the model is to build an algorithm to detect if a satellite image has vessel or not and the second phase is to build an algorithm to segment or locate the vessel if it is detected.

For detection of vessel in an image the training data is fed into ResNet50. After Classification, Segmentation is done using Mask RCNN algorithm to locate or segment the ship.

ResNet-50:

ResNet-50 is one of the transfer-learning techniques. Transfer learning generally refers to a deep-learning process where a model trained on one problem is used in some way on a second related problem. Transfer learning has the benefit of decreasing the training time for a neural network model and can result in lower generalization error.
Residual Net (ResNet) is one such widely used model for transfer learning because of its performance. It is a pre-trained model provided by the Keras and can be efficiently used based on requirements. ResNet is a powerful backbone model that is used very frequently in many computer vision tasks. ResNet uses skip connection to add the output from an earlier layer to a later layer. This helps it mitigate the vanishing gradient problem

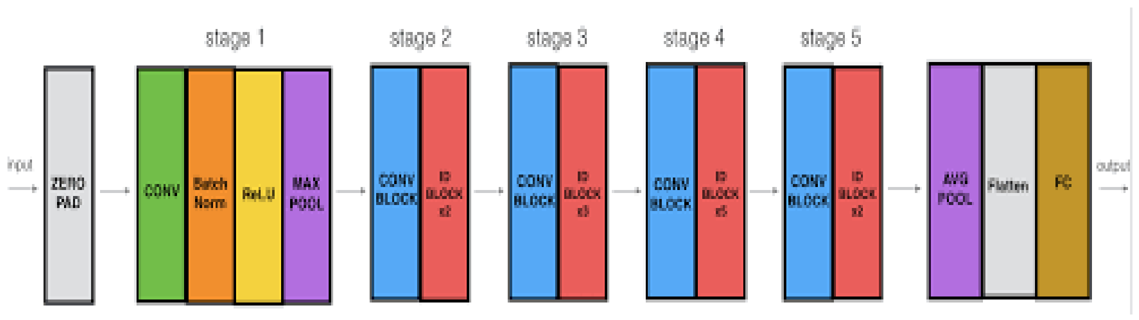


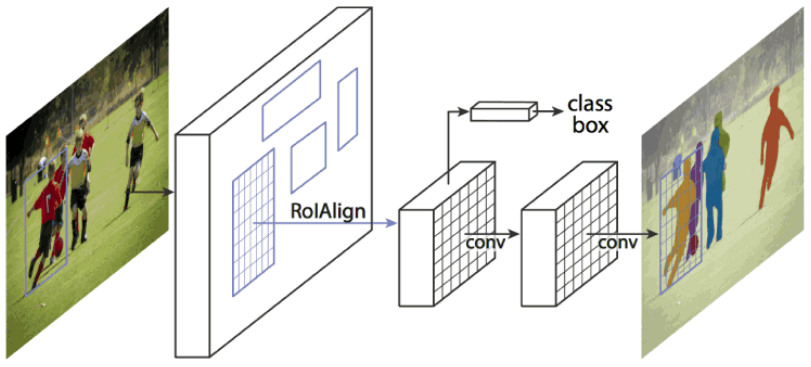
Fig 1.3.1 ResNet-50

Mask RCNN:

Mask RCNN is a deep neural network aimed to solve instance segmentation problem in machine learning or computer vision. In other words, it can separate different objects in an image or a video.

You give it an image, it gives you the object bounding boxes, classes and masks. There are two stages in it.

First, it generates proposals about the regions where there might be an object based on the input image. Second, it predicts the class of the object, refines the bounding box and generates a mask in pixel level of the object based on the first stage proposal. It is one of the most efficient networks to solve single instance segmentation problem.



The Mask R-CNN framework for instance segmentation

Fig 1.3.2 Mask RCNN framework

Using Mask RCNN with ResNet50 as a backbone, vessel can be detected and segmented in a satellite image with better efficiency.

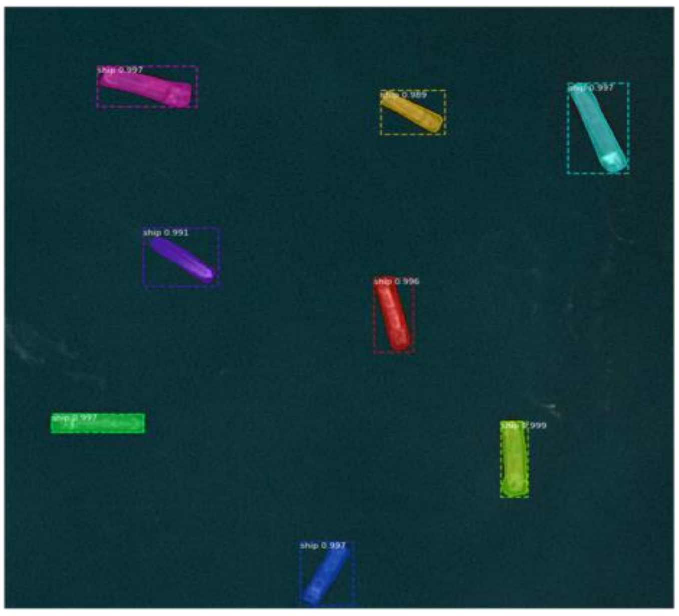


Fig 1.3.3 Instance Segmentation

1.4 ORGANIZATION OF PROJECT

The model takes uploaded image from a user and first detects if it has vessel or not. If the uploaded image has a vessel, the model then segments the vessel in the picture.

There are three modules in the project. They are as follows:

1. Detection of Vessel
2. Segmentation of Vessel
3. Graphical User Interface

1.Detection of Vessel:

Feeding the training data into ResNet-50 network and classifying if an image has vessel or not.

ResNet-50 is a residual network and is very efficient. ResNet-50 is pre-trained and can be implemented from the Keras Library

2.Segmentation of Vessel:

Feeding the training data into Mask RCNN to obtain the mask for testing data. Mask is used to segment the vessel.

3.Graphical User Interface:

Combining the detection model and segmentation model along with the Python Tkinter. It facilitates the user to upload a satellite image and the model detects if a vessel is there in it. If vessel is detected it provides the segmentation as to know where the vessel is located in the picture.

2 REQUIREMENTS

2.1 SOFTWARE REQUIREMENTS

Software requirements is a field within software engineering that deals with establishing the needs of stakeholders that are to be solved by software. The IEEE Standard Glossary of Software Engineering Terminology defines a requirement as:

- A condition or capability needed by a user to solve a problem or achieve an objective.
- A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.
- A documented representation of a condition or capability as in 1 or 2.

The activities related to working with software requirements can broadly be broken down into elicitation, analysis, specification, and management.

The software requirements specify the use of all required software products like data-management system. The required software product specifies the numbers and version. Each interface specifies the purpose of the interfacing software as related to this software product.

A software requirement can be of 3 types:

- **Functional requirements:** These are the requirements that the end user specifically Demands as basic facilities that the system should offer.
- **Non-functional requirements:** These are basically the quality constraints that the system must satisfy according to the project contract.
- **Domain requirements:** These are the requirements which are characteristic of a particular category or domain of projects

This project requirements are:

- **Programming Language -** Python Language.
- **Tools:** Kaggle Notebook, Visual Studio code
- **Packages** – numpy, pillow, tkinter, matplotlib, imageio, pandas, open-cv, skimage, Keras, Tensorflow

2.2 HARDWARE REQUIREMENTS

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, A **hardware requirements** list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

The hardware requirement specifies each interface of the software elements and the hardware elements of the system. These hardware requirements include configuration characteristics.

Operating system - Windows

Processor - Intel core i5

Memory - RAM 8GB

Storage - 1TB

2.3 TECHNOLOGIES DESCRIPTION

2.3.1 PYTHON:

- Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.
- Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.
- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP. Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code.

Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

2.3.2 NUMPY

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions

- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi- dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

2.3.3 TKINTER

The **tkinter** package (“Tk interface”) is the standard Python interface to the Tk GUI toolkit. Tkinter is not the only GUI Programming toolkit for Python. It is however the most commonly used one. Cameron Laird calls the yearly decision to keep TkInter "one of the minor traditions of the Python world." Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

- Import the Tkinter module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

2.3.4 PILLOW

Pillow is the friendly PIL fork by Alex Clark and Contributors. PIL is the Python Imaging Library by Fredrik Lundh and Contributors. This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.

The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool. The Python Imaging Library is ideal for image archival and batch processing applications. The Pillow library contains all the basic image processing functionality. You can do image resizing, rotation and transformation.

You can display images using Tk Photo Image, Bitmap Image and Windows DIB interface, which can be used with PythonWin and other Windows-based toolkits and many other Graphical User Interface (GUI) toolkits. For debugging purposes, there is show () method to save the image to disk which calls the external display utility.

2.3.5 OPENCV

OpenCV-Python is a Python wrapper for the original OpenCV C++ implementation. OpenCV-Python makes use of Numpy, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays. This also makes it easier to integrate with other libraries that use Numpy such as SciPy and Matplotlib.

2.3.6 MATPLOTLIB

Matplotlib is a Python 2D plotting library which produces publication quality figures in variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc. with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc., via an object-oriented interface or via a set functions familiar to MATLAB users.

2.3.7 IMAGEIO

Imageio is a Python library that provides an easy interface to read and write a wide range of image data, including animated images, volumetric data, and scientific formats. It is cross-platform, runs on Python 3.5+, and is easy to install.

2.3.8 SKIMAGE

- It is an open-source image processing library for the Python programming language. It includes algorithms for segmentation, geometric transformations, color space manipulation, analysis, filtering, morphology, feature detection.
- It is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

2.4 NEURAL NETWORK LIBRARIES

2.4.1 TENSORFLOW

- TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications.
- TensorFlow was originally developed by researchers and engineers working on the Google Brain team within Google's Machine Intelligence Research organization to conduct machine learning and deep neural networks research. The system is general enough to be applicable in a wide variety of other domains, as well.
- TensorFlow provides stable Python and C++ APIs, as well as non-guaranteed backward compatible API for other languages.

2.4.2 KERAS

- Keras is a minimalist Python library for deep learning that can run on top of Theano or TensorFlow.
- It was developed to make implementing deep learning models as fast and easy as possible for research and development.
- It runs on Python 2.7 or 3.5 and can seamlessly execute on GPUs and CPUs given the underlying frameworks. It is released under the permissive MIT license.
- Keras was developed and maintained by François Chollet and is designed for modularity and minimalism.

3. DESIGN

3.1 INTRODUCTION

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer’s goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement has been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word “Quality”. Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer’s view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

a. ARCHITECTURE DIAGRAM

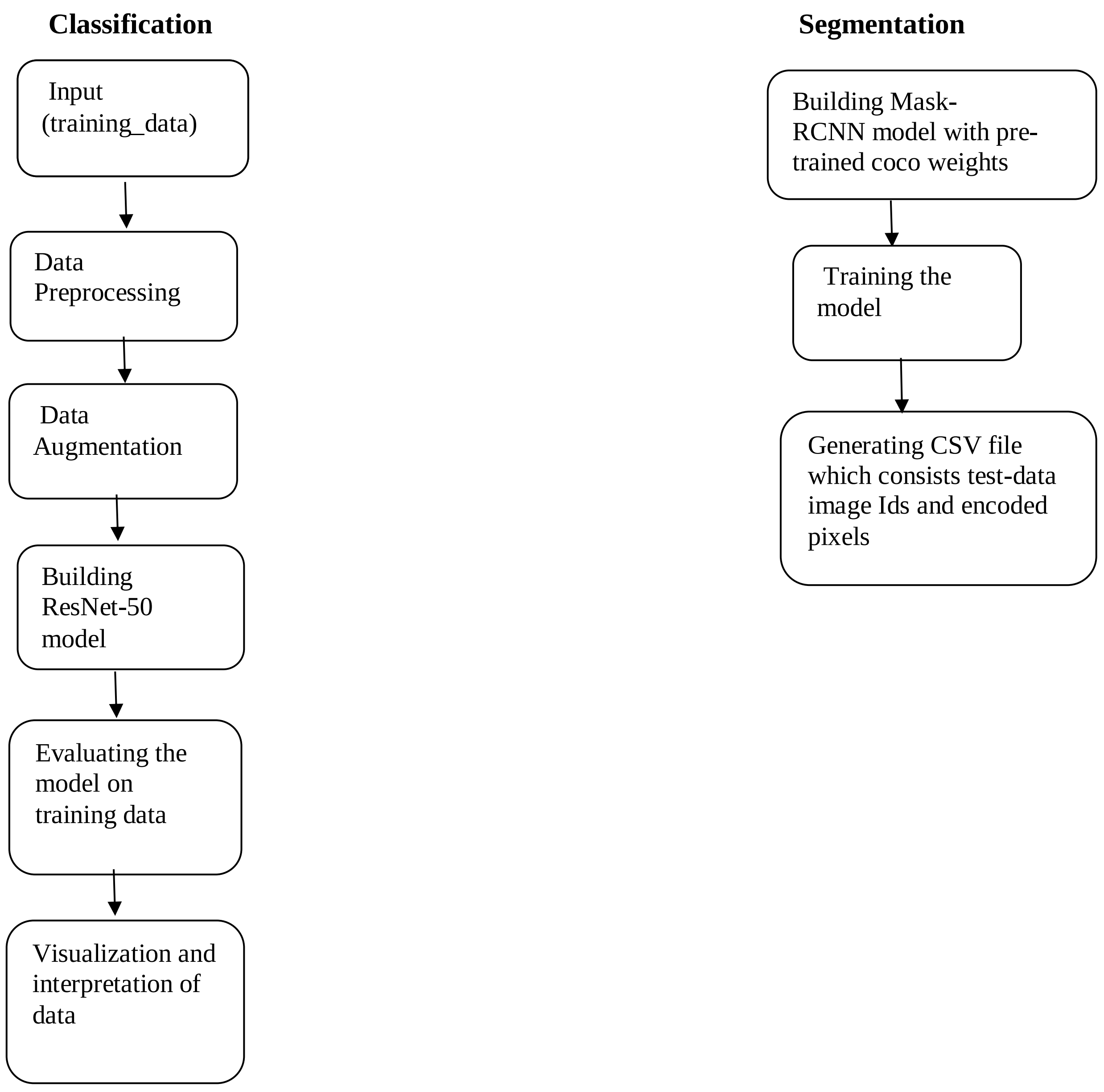

Classification
Input (training_data)
Data Preprocessing
Data Augmentation
Building ResNet-50 model
Evaluating the model on training data
Visualization and interpretation of data
Segmentation
Building Mask-RCNN model with pre-trained coco weights
Training the model
Generating CSV file which consists test-data image Ids and encoded pixels

Fig 3.2.1 Architecture Diagram – Neural Network

Building Graphical User Interface

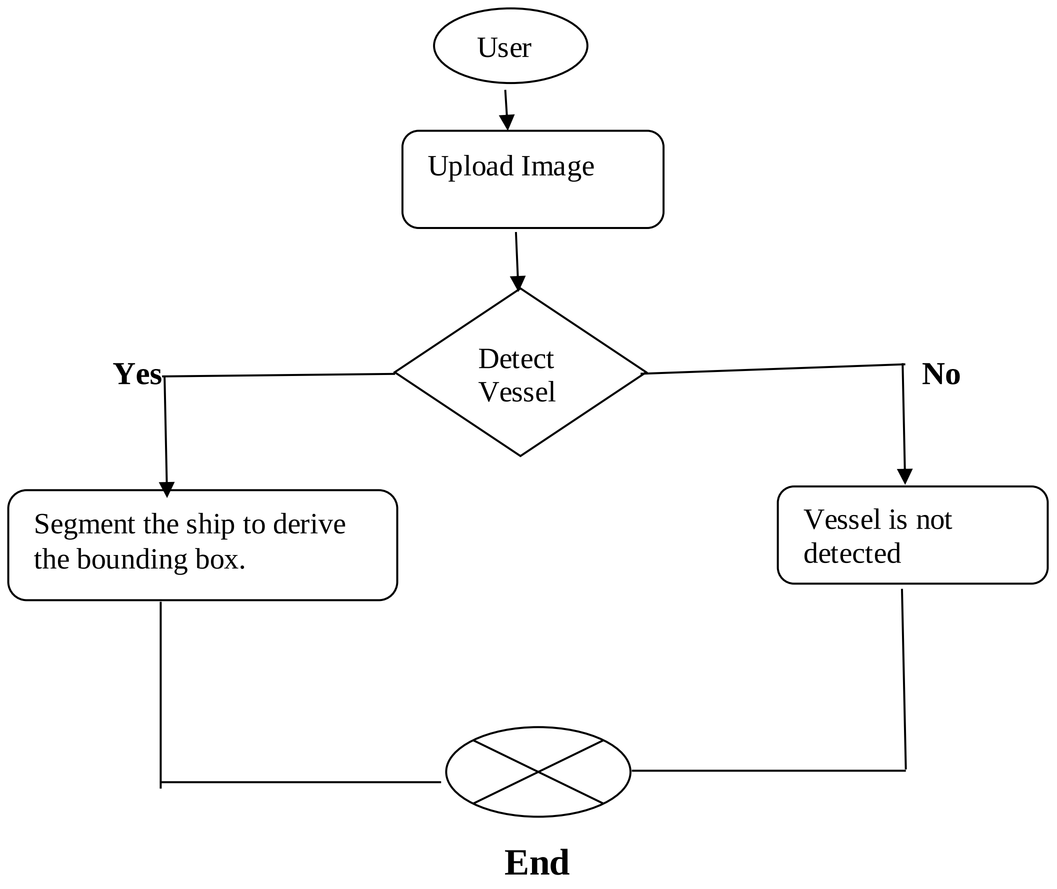


Fig 3.2.2 Architecture diagram- GUI

3.3 UML DIAGRAMS

3.3.1 Use Case Diagram

To model a system, the most important aspect is to capture the dynamic behavior. Dynamic behavior means the behavior of the system when it is running/operating.

Only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior. In UML, there are five diagrams available to model the dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature, there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. Use case diagrams consist of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

Hence to model the entire system, a number of use case diagrams are used.

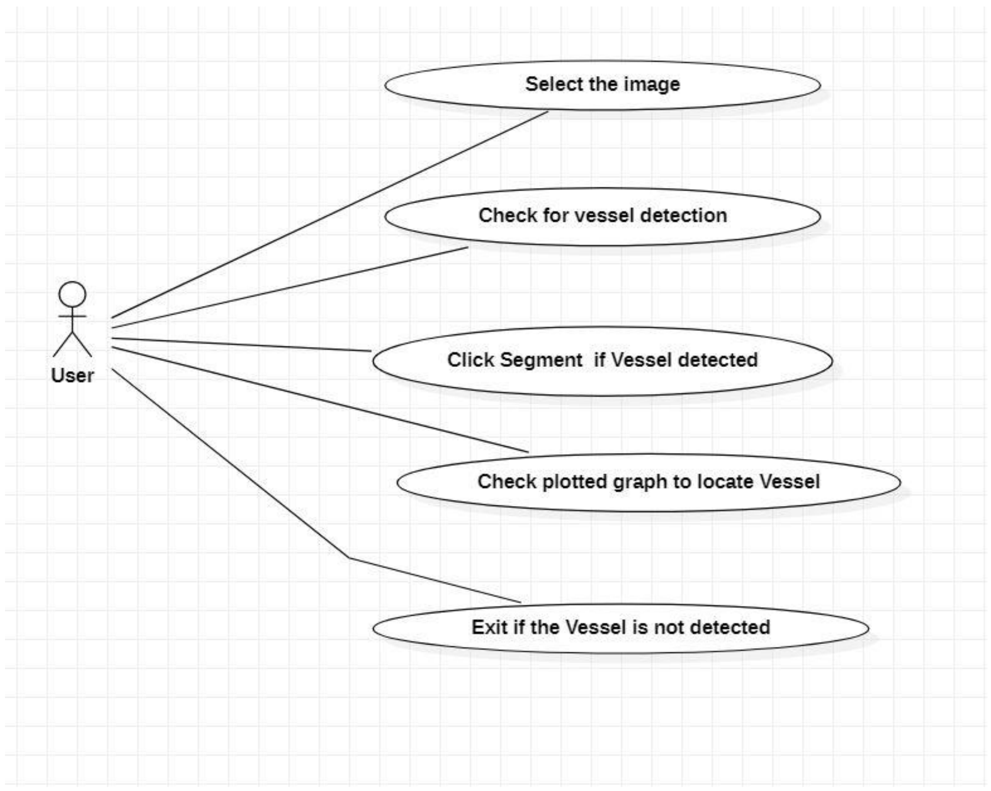

Select the image
Check for vessel detection
User
Click Segment if Vessel detected
Check plotted graph to locate Vessel
Exit if the Vessel is not detected

Fig 3.3.1 Use Case diagram

3.3.2 Sequence Diagram

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.

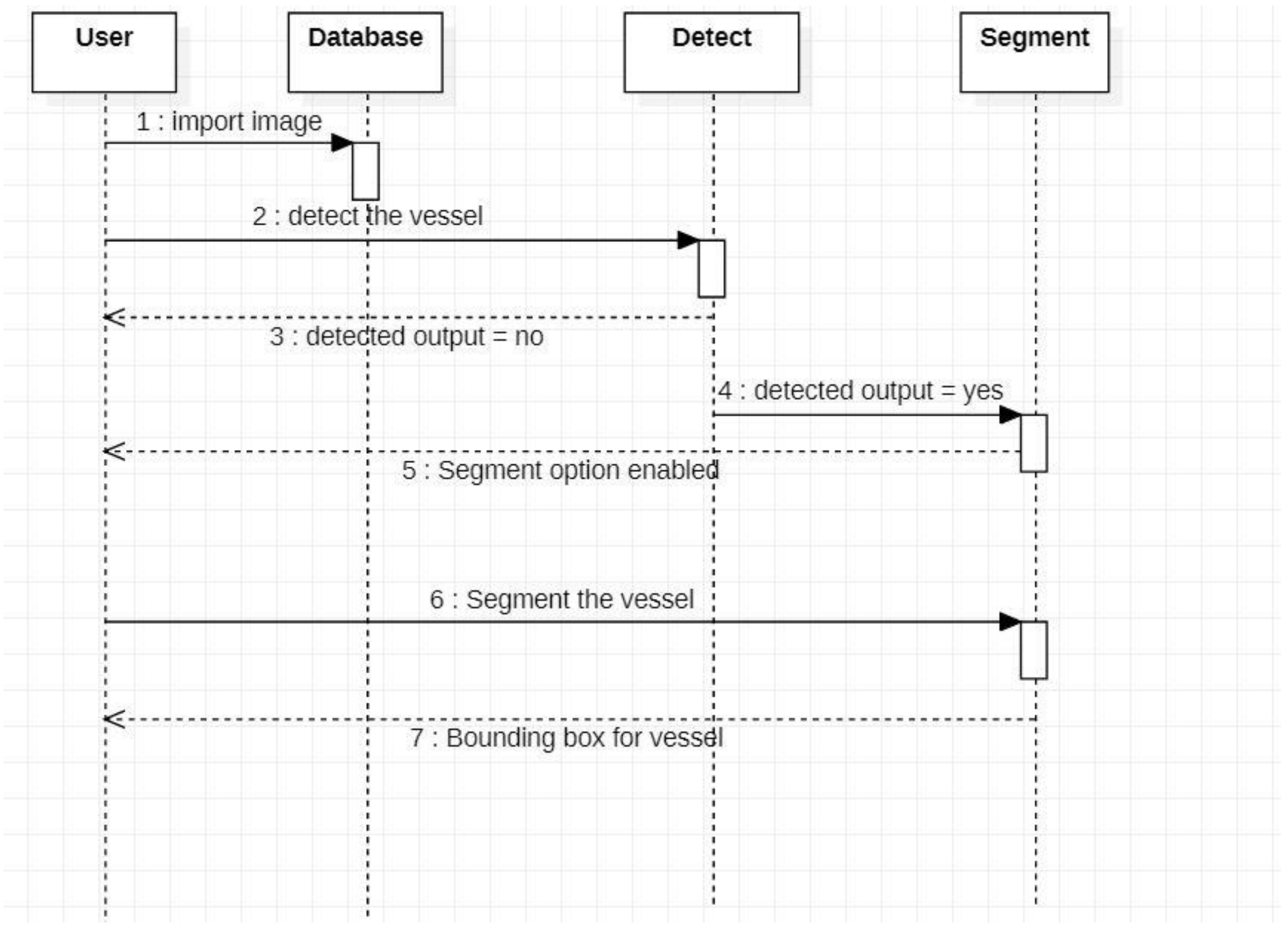


Fig 3.3.2: Sequence Diagram

3.3.3 Activity Diagram

Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

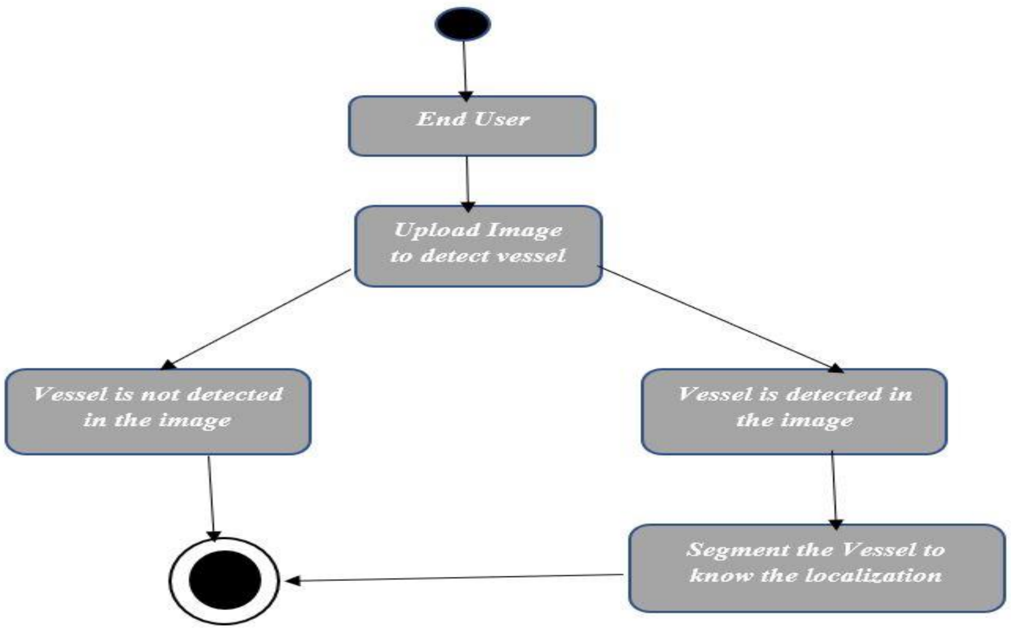

End User
Upload Image to detect vessel
Vessel is not detected in the image
Vessel is detected in the image
Segment the Vessel to know the localization

Fig 3.3.3: Activity Diagram

3.3.4 Collaboration Diagram

A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time. Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles. The messages between objects are shown as arrows connecting the relevant rectangles along with labels that define the message sequencing.

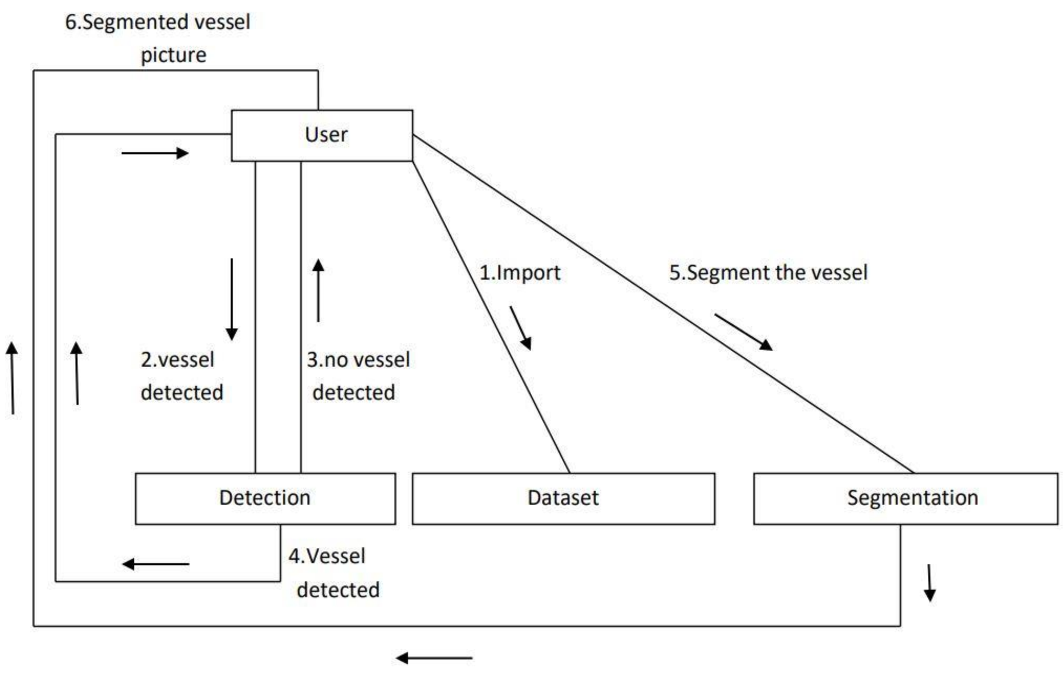


Fig 3.3.4:Collaboration Diagram

3.3.5 Class Diagram

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

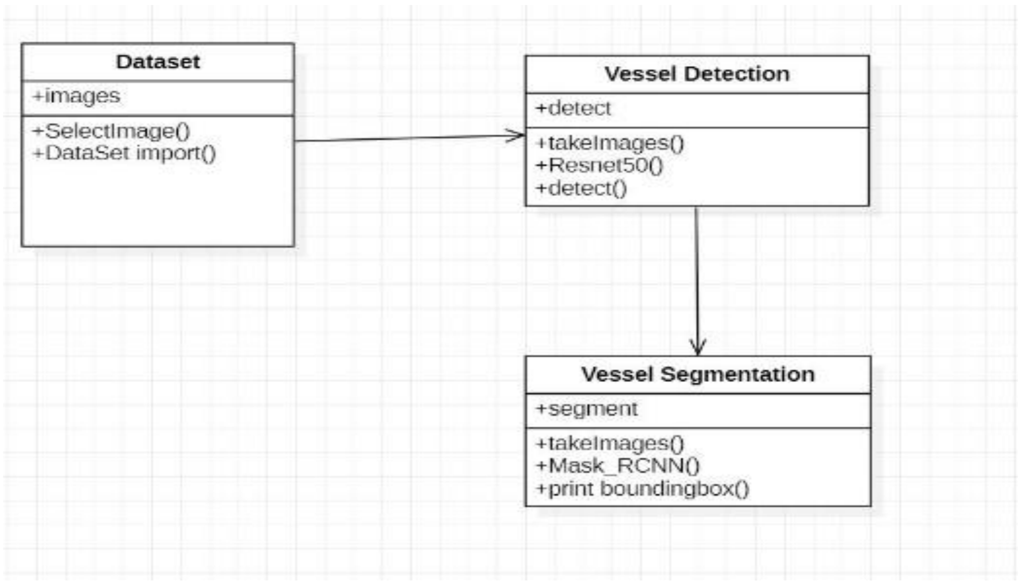


Fig 3.3.5: Class Diagram

4. IMPLEMENTATION

4.1 SOURCE CODE

Code for Classification:

The training data is taken and one hot encoded and augmented to fine tune the result. ResNet-50 is implemented from Keras inbuilt library and evaluated on the dataset with 30 epochs.

```
import os
import gc
import numpy as np
import pandas as pd
import time
```

```
SHIP_CLASS_NAME = 'ship'
IMAGE_WIDTH     = 768
IMAGE_HEIGHT    = 768
SHAPE           = (IMAGE_WIDTH, IMAGE_HEIGHT)
WORKING_DIR = '/kaggle/working'
INPUT_DIR = '/kaggle/input'
OUTPUT_DIR = '/kaggle/output'

TRAIN_SHIP_SEGMENTATIONS_PATH = os. path.join (INPUT_DIR, 'airbus-ship-detection/train_s
hip_segmentations_v2.csv')
TRAIN_DATA_PATH = os.path.join (INPUT_DIR, 'airbus-ship-detection/train_v2')
TEST_DATA_PATH = os.path.join (INPUT_DIR, 'airbus-ship-detection/test_v2')
```

```
train = pd.read_csv (TRAIN_SHIP_SEGMENTATIONS_PATH, engine='python')
train.head()
```

```
train['exist_ship'] = train['EncodedPixels'].fillna(0)
train.loc[train['exist_ship'] != 0, 'exist_ship'] = 1
del train['EncodedPixels']
```

```
print(len(train['ImageId']))
print(train['ImageId'].value_counts().shape[0])

train_gp = train.groupby('ImageId').sum().reset_index()
train_gp.loc[train_gp['exist_ship'] > 0,'exist_ship'] = 1
```

```
print(train_gp['exist_ship'].value_counts())
train_gp = train_gp.sort_values(by='exist_ship')
train_gp = train_gp.drop(train_gp.index[0:100000])
```

```
print(train_gp['exist_ship'].value_counts())
train_sample = train_gp.sample(5000)
print(train_sample['exist_ship'].value_counts())
print (train_sample.shape)
```

```
Train_path = TRAIN_DATA_PATH + '/'
Test_path = TEST_DATA_PATH + '/'

print(Train_path)
print(Test_path)
```

```
from PIL import Image

training_img_data = []
target_data = []

data = np.empty((len(train_sample['ImageId']), 256, 256, 3), dtype=np.uint8)
data_target = np.empty((len(train_sample['ImageId'])), dtype=np.uint8)
image_name_list = os.listdir(Train_path)
index = 0

for image_name in image_name_list:
    if image_name in list(train_sample['ImageId']):
        imageA = Image.open(Train_path + image_name).resize((256, 256)).convert('RGB')
        data[index] = imageA
        data_target[index] = train_sample[train_gp['ImageId'].str.contains(image_name)]['exist_ship'].iloc[
0]
        index += 1

print(data.shape)
print(data_target.shape)
```

```
from sklearn.preprocessing import OneHotEncoder

targets = data_target.reshape(len(data_target), -1)
encoder = OneHotEncoder()
encoder.fit(targets)
targets = encoder.transform(targets).toarray()

print(targets.shape)
```

```
from sklearn.model_selection import train_test_split

x_train, x_val, y_train, y_val = train_test_split(data, targets, test_size=0.2)
x_train.shape, x_val.shape, y_train.shape, y_val.shape
```

```
from keras.preprocessing.image import ImageDataGenerator

img_gen = ImageDataGenerator(rescale=1./255,
                             zca_whitening = False,
                             rotation_range = 90,
                             width_shift_range = 0.2,
                             height_shift_range = 0.2,
                             brightness_range = [0.5, 1.5],
                                shear_range = 0.2,
                                zoom_range = 0.2,
                                horizontal_flip = True,
                                vertical_flip = True)
from keras.applications.resnet50 import ResNet50 as ResModel

img_width, img_height = 256, 256
model = ResModel(weights='imagenet', include_top=False, input_shape=(img_width, img_height, 3))
```

```
from keras.layers import Dropout, Flatten, Dense, GlobalAveragePooling2D
from keras.models import Sequential, Model

for layer in model.layers:
    layer.trainable = False

x = model.output
x = Flatten()(x)
x = Dropout(0.8)(x)
x = Dense(1024, activation="relu")(x)
predictions = Dense(2, activation="softmax")(x)

model_final = Model(inputs = model.input, outputs = predictions)
```

```
from keras import optimizers

epochs = 30
lrate = 0.0005
decay = lrate/epochs
sgd = optimizers.SGD(lr=lrate, momentum=0.9, decay=decay, nesterov=False)
model_final.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
model_final.summary()
```

```
history = model_final.fit_generator(img_gen.flow(x_train, y_train, batch_size=32),
                                    steps_per_epoch=len(x_train)/32,
                                    validation_data=(x_val, y_val),
                                    epochs=epochs)

model_final.save('ShipResnetClassifier.h5')
```

Code for Segmentation:

Mask COCO weights are used and Mask RCNN is trained to generate the masks for the images to segment them. The output is the csv file that contains all the images of testing data along with their ids and encoded pixels. Encoded pixels are used to segment the vessel if it is found.

```
import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
  raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```

```
import os
import sys
import random
import math
import numpy as np
import cv2
import matplotlib.pyplot as plt
import json
from imgaug import augmenters as iaa
from tqdm import tqdm
import pandas as pd
import glob
```

```
DATA_DIR = '/kaggle/input/airbus-ship-detection'
ROOT_DIR = '/kaggle/working'
!git clone https://www.github.com/matterport/Mask_RCNN.git
os.chdir('Mask_RCNN')
```

```
sys.path.append(os.path.join(ROOT_DIR, 'Mask_RCNN'))
from mrcnn.config import Config
from mrcnn import utils
import mrcnn.model as modellib
from mrcnn import visualize
from mrcnn.model import log
train_dicom_dir = os.path.join(DATA_DIR, 'train_v2')
test_dicom_dir = os.path.join(DATA_DIR, 'test_v2')
```

```
!wget --quiet https://github.com/matterport/Mask_RCNN/releases/download/v2.0/mask_rcnn_coco.h5
!ls -lh mask_rcnn_coco.h5

COCO_WEIGHTS_PATH = "mask_rcnn_coco.h5"
```

```
class DetectorConfig(Config):

    NAME = 'airbus'
    GPU_COUNT = 1
    IMAGES_PER_GPU = 9

    BACKBONE = 'resnet50'

    NUM_CLASSES = 2

    IMAGE_MIN_DIM = 384
    IMAGE_MAX_DIM = 384
    RPN_ANCHOR_SCALES = (8, 16, 32, 64)
    TRAIN_ROIS_PER_IMAGE = 64
    MAX_GT_INSTANCES = 14
    DETECTION_MAX_INSTANCES = 10
    DETECTION_MIN_CONFIDENCE = 0.95
    DETECTION_NMS_THRESHOLD = 0.0

    STEPS_PER_EPOCH = 15 if debug else 150
    VALIDATION_STEPS = 10 if debug else 125
    LOSS_WEIGHTS = {
        "rpn_class_loss": 30.0,
        "rpn_bbox_loss": 0.8,
        "mrcnn_class_loss": 6.0,
        "mrcnn_bbox_loss": 1.0,
        "mrcnn_mask_loss": 1.2
    }

config = DetectorConfig()
config.display()
```

```
import os
import numpy as np
import pandas as pd
from skimage.io import imread
import matplotlib.pyplot as plt
from matplotlib.cm import get_cmap
from skimage.segmentation import mark_boundaries
from skimage.util import montage
from skimage.morphology import binary_opening, disk, label
import gc; gc.enable()

montage_rgb = lambda x: np.stack([montage(x[:, :, :, i]) for i in range(x.shape[3])], -1)

def multi_rle_encode(img, **kwargs):

    labels = label(img)
    if img.ndim > 2:
        return [rle_encode(np.sum(labels==k, axis=2), **kwargs) for k in np.unique(labels[labels>0])]
    else:
        return [rle_encode(labels==k, **kwargs) for k in np.unique(labels[labels>0])]
```

```
def rle_encode(img, min_max_threshold=1e-3, max_mean_threshold=None):

    if np.max(img) < min_max_threshold:
        return ''
    if max_mean_threshold and np.mean(img) > max_mean_threshold:
        return ''
    pixels = img.T.flatten()
    pixels = np.concatenate([[0], pixels, [0]])
    runs = np.where(pixels[1:] != pixels[:-1])[0] + 1
    runs[1::2] -= runs[::2]
    return ' '.join(str(x) for x in runs)

def rle_decode(mask_rle, shape=(768, 768)):

    s = mask_rle.split()
    starts, lengths = [np.asarray(x, dtype=int) for x in (s[0:][::2], s[1:][::2])]
    starts -= 1
    ends = starts + lengths
    img = np.zeros(shape[0]*shape[1], dtype=np.uint8)
    for lo, hi in zip(starts, ends):
        img[lo:hi] = 1
    return img.reshape(shape).T

def masks_as_image(in_mask_list):

    all_masks = np.zeros((768, 768), dtype = np.uint8)
    for mask in in_mask_list:
        if isinstance(mask, str):
            all_masks |= rle_decode(mask)
    return all_masks

def masks_as_color(in_mask_list):

    all_masks = np.zeros((768, 768), dtype = np.float)
    scale = lambda x: (len(in_mask_list)+x+1) / (len(in_mask_list)*2)
    for i,mask in enumerate(in_mask_list):
        if isinstance(mask, str):
            all_masks[:,:] += scale(i) * rle_decode(mask)
    return all_masks
```

```
from PIL import Image
from sklearn.model_selection import train_test_split

exclude_list = ['6384c3e78.jpg','13703f040.jpg', '14715c06d.jpg',  '33e0ff2d5.jpg',
                '4d4e09f2a.jpg', '877691df8.jpg', '8b909bb20.jpg', 'a8d99130e.jpg',
                'ad55c3143.jpg', 'c8260c541.jpg', 'd6c7f17c7.jpg', 'dc3e7c901.jpg',
                'e44dffe88.jpg', 'ef87bad36.jpg', 'f083256d8.jpg']
```

```
train_names = [f for f in os.listdir(train_dicom_dir) if f not in exclude_list]
test_names = [f for f in os.listdir(test_dicom_dir) if f not in exclude_list]

print(len(train_names), len(test_names))
```

```
SEGMENTATION = DATA_DIR + '/train_ship_segmentations_v2.csv'
anns = pd.read_csv(SEGMENTATION)
anns.head()
```

```
train_names = anns[anns.EncodedPixels.notnull()].ImageId.unique().tolist()

test_size = config.VALIDATION_STEPS * config.IMAGES_PER_GPU
image_fps_train, image_fps_val = train_test_split(train_names, test_size=test_size, random_state=42)

if debug:
    image_fps_train = image_fps_train[:100]
    image_fps_val = image_fps_val[:100]
    test_names = test_names[:100]

print(len(image_fps_train), len(image_fps_val), len(test_names))
```

```
class DetectorDataset(utils.Dataset):

    def __init__(self, image_fps, image_annotations, orig_height, orig_width):
        super().__init__(self)


        self.add_class('ship', 1, 'Ship')


        for i, fp in enumerate(image_fps):
            annotations = image_annotations.query('ImageId=="' + fp + '"')['EncodedPixels']
            self.add_image('ship', image_id=i, path=os.path.join(train_dicom_dir, fp),
                           annotations=annotations, orig_height=orig_height, orig_width=orig_width)

    def image_reference(self, image_id):
        info = self.image_info[image_id]
        return info['path']

    def load_image(self, image_id):
        info = self.image_info[image_id]
        fp = info['path']
        image = imread(fp)

        if len(image.shape) != 3 or image.shape[2] != 3:
            image = np.stack((image,) * 3, -1)
        return image

    def load_mask(self, image_id):
        info = self.image_info[image_id]
        annotations = info['annotations']
        count = len(annotations)
        if count == 0:
            mask = np.zeros((info['orig_height'], info['orig_width'], 1), dtype=np.uint8)
            class_ids = np.zeros((1,), dtype=np.int32)
        else:
            mask = np.zeros((info['orig_height'], info['orig_width'], count), dtype=np.uint8)
            class_ids = np.zeros((count,), dtype=np.int32)
            for i, a in enumerate(annotations):
                mask[:, :, i] = rle_decode(a)
                class_ids[i] = 1
        return mask.astype(np.bool), class_ids.astype(np.int32)
```

```
%%time

dataset_train = DetectorDataset(image_fps_train, image_annotations, ORIG_SIZE, ORIG_SIZE)
dataset_train.prepare()
```

```
%%time

dataset_val = DetectorDataset(image_fps_val, image_annotations, ORIG_SIZE, ORIG_SIZE)
dataset_val.prepare()
```

```
class_ids = [0]
while class_ids[0] == 0:
    image_id = random.choice(dataset_val.image_ids)
    image_fp = dataset_val.image_reference(image_id)
    image = dataset_val.load_image(image_id)
    mask, class_ids = dataset_val.load_mask(image_id)

print(image.shape)

plt.figure(figsize=(10, 10))
plt.subplot(1, 2, 1)
plt.imshow(image)
plt.axis('off')

plt.subplot(1, 2, 2)
masked = np.zeros(image.shape[:2])
for i in range(mask.shape[2]):
    masked += mask[:, :, i]
plt.imshow(masked, cmap='gray')
plt.axis('off')

print(image_fp)
print(class_ids)
```

```
augmentation = iaa.Sequential([
    iaa.OneOf([
        iaa.Affine(rotate=0),
        iaa.Affine(rotate=90),
        iaa.Affine(rotate=180),
        iaa.Affine(rotate=270),
    ]),
    iaa.Fliplr(0.5),
    iaa.Flipud(0.5),
    iaa.OneOf([
        iaa.Multiply((0.9, 1.1)),
        iaa.ContrastNormalization((0.9, 1.1)),
    ]),
    iaa.OneOf([
        iaa.GaussianBlur(sigma=(0.0, 0.1)),
        iaa.Sharpen(alpha=(0.0, 0.1)),
    ]),
])

imggrid = augmentation.draw_grid(image, cols=5, rows=2)
plt.figure(figsize=(30, 12))
_ = plt.imshow(imggrid.astype(int))
```

```
model = modellib.MaskRCNN(mode='training', config=config, model_dir=ROOT_DIR)



model.load_weights(COCO_WEIGHTS_PATH, by_name=True, exclude=[
    "mrcnn_class_logits", "mrcnn_bbox_fc",
    "mrcnn_bbox", "mrcnn_mask"])
```

```
LEARNING_RATE = 0.003



import warnings
warnings.filterwarnings("ignore")
```

```
%%time
model.train(dataset_train, dataset_val,
            learning_rate=LEARNING_RATE*2,
            epochs=2,
            layers='heads',
            augmentation=None)
history = model.keras_model.history.history
```

```
%%time
model.train(dataset_train, dataset_val,
            learning_rate=LEARNING_RATE,
            epochs=4 if debug else 14,
            layers='all',
            augmentation=augmentation)

new_history = model.keras_model.history.history
for k in new_history: history[k] = history[k] + new_history[k]
```

```
dir_names = next(os.walk(model.model_dir))[1]
key = config.NAME.lower()
dir_names = filter(lambda f: f.startswith(key), dir_names)
dir_names = sorted(dir_names)

if not dir_names:
    import errno
    raise FileNotFoundError(
        errno.ENOENT,
        "Could not find model directory under {}".format(self.model_dir))

fps = []

for d in dir_names:
    dir_name = os.path.join(model.model_dir, d)

    checkpoints = next(os.walk(dir_name))[2]
    checkpoints = filter(lambda f: f.startswith("mask_rcnn"), checkpoints)
    checkpoints = sorted(checkpoints)
    if not checkpoints:
        print('No weight files in {}'.format(dir_name))
    else:
        checkpoint = os.path.join(dir_name, checkpoints[best_epoch])
        fps.append(checkpoint)

model_path = sorted(fps)[-1]
print('Found model {}'.format(model_path))
```

```
class InferenceConfig(DetectorConfig):
    GPU_COUNT = 1
    IMAGES_PER_GPU = 1

inference_config = InferenceConfig()



model = modellib.MaskRCNN(mode='inference',
                          config=inference_config,
                          model_dir=ROOT_DIR)



assert model_path != "", "Provide path to trained weights"
print("Loading weights from ", model_path)
model.load_weights(model_path, by_name=True)
```

```
def get_colors_for_class_ids(class_ids):
    colors = []
    for class_id in class_ids:
        if class_id == 1:
            colors.append((.941, .204, .204))
    return colors
```

```
dataset = dataset_val
fig = plt.figure(figsize=(10, 40))

for i in range(8):

    image_id = random.choice(dataset.image_ids)

    original_image, image_meta, gt_class_id, gt_bbox, gt_mask =\
        modellib.load_image_gt(dataset_val, inference_config,
                               image_id, use_mini_mask=False)



    plt.subplot(8, 2, 2*i + 1)
    visualize.display_instances(original_image, gt_bbox, gt_mask, gt_class_id,
                                dataset.class_names,
                                colors=get_colors_for_class_ids(gt_class_id), ax=fig.axes[-1])

    plt.subplot(8, 2, 2*i + 2)
    results = model.detect([original_image])
    r = results[0]
    visualize.display_instances(original_image, r['rois'], r['masks'], r['class_ids'],
                                dataset.class_names, r['scores'],
                                colors=get_colors_for_class_ids(r['class_ids']), ax=fig.axes[-1])
```

```
THRESHOLD = 0.45
test_names_nothing = ship_detection.loc[ship_detection['p_ship'] <= THRESHOLD].index.tolist()
len(test_names_nothing), len(ship_detection), len(test_names_nothing)/len(ship_detection)
```

```
def predict(image_fps, filepath='submission.csv', min_conf=config.DETECTION_MIN_CONFIDENC
E):
    resize_factor = ORIG_SIZE / config.IMAGE_SHAPE[0]
    with open(filepath, 'w') as file:
        file.write("ImageId,EncodedPixels\n")

        for image_id in tqdm(image_fps):
            found = False

            if image_id not in test_names_nothing:
```

```
            image = imread(os.path.join(test_dicom_dir, image_id))

            if len(image.shape) != 3 or image.shape[2] != 3:
                image = np.stack((image,) * 3, -1)
            results = model.detect([image])
            r = results[0]

            assert( len(r['rois']) == len(r['class_ids']) == len(r['scores']) )
            if len(r['rois']) == 0:
                pass
            else:
                num_instances = len(r['rois'])

                for i in range(num_instances):
                    if r['scores'][i] > min_conf:
                        file.write(image_id + "," + rle_encode(r['masks'][...,i]) + "\n")
                        found = True

        if not found:
            file.write(image_id + ",\n")  ## no ship
```

```
submission_fp = os.path.join(ROOT_DIR, 'submission.csv')
predict(test_image_fps, filepath=submission_fp)
print(submission_fp)
```

```
sub = pd.read_csv(submission_fp)
print(sub.EncodedPixels.isnull().sum(), sub.ImageId.nunique(), sub.EncodedPixels.isnull().sum()/sub.I
mageId.nunique())
sub.head(50)
```

```
def visualize_test():
    image_id = random.choice(test_names)

    image = imread(os.path.join(test_dicom_dir, image_id))

    resize_factor = 1



    if len(image.shape) != 3 or image.shape[2] != 3:
        image = np.stack((image,) * 3, -1)



    results = model.detect([image])
    r = results[0]
    for bbox in r['rois']:
        x1 = int(bbox[1] * resize_factor)
        y1 = int(bbox[0] * resize_factor)
        x2 = int(bbox[3] * resize_factor)
        y2 = int(bbox[2]  * resize_factor)
        cv2.rectangle(image, (x1,y1), (x2,y2), (77, 255, 9), 3, 1)
        width = x2 - x1
        height = y2 - y1
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 5))
    ax1.set_title(f"{image_id}")
    ax1.imshow(image)
    ax2.set_title(f"{len(r['rois'])} masks if prob:{ship_detection.loc[image_id][0]:.6f}")
    ax2.imshow(masks_as_color(sub.query(f"ImageId=='{image_id}'")['EncodedPixels']))

for i in range(8):
    visualize_test()
```

GUI for Vessel detection and Segmentation:

```
import tkinter as tk
from typing import Text
import tensorflow as tf
from keras. preprocessing import image
from PIL import ImageTk, Image
from tkinter import filedialog, Label
from keras. models import load_model
from matplotlib import pyplot as plt
import cv2
import pandas as pd
import numpy as np
from tkinter.ttk import *
from imageio import imread
import matplotlib.pyplot as plt
import matplotlib as mplot
from skimage.segmentation import mark_boundaries
from skimage. measure import label, regionprops
from skimage. morphology import label
from tkinter. messagebox import *

import warnings
warnings.filterwarnings("ignore")

img_width = 768
img_height = 768
model = load_model ('ShipResnetClassifier.h5')


def ship_detect(img_path):
  img = tf. keras. preprocessing. image. load_img (img_path, target_size=(256, 256))
  x = image.img_to_array(img)
  x = np.expand_dims(x, axis=0)
  images = np.vstack([x])
  classes = model.predict(images)
  #print(classes)
  if(classes[0][1] > 0.80):
      return "Vessel is detected in the uploaded satellite image. Click on Segment to know where it is located"
  else:
    return "Vessel is not detected in the uploaded satellite image"

def rle_decode (mask_rle, shape):
    img = np.zeros(shape [0] *shape[1], dtype=np.uint8)
    s = mask_rle.split()
    starts, lengths = [np.asarray(x, dtype=int) for x in (s[0:][::2], s[1:][::2])]
    starts -= 1
    ends = starts + lengths

    for lo, hi in zip (starts, ends):
        img [lo: hi] = 1
    return img. reshape(shape). T


def masks_as_image(in_mask_list, all_masks=None):
    if all_masks is None:
        all_masks = np.zeros((768, 768), dtype = np.int16)
    for mask in in_mask_list:
        if isinstance(mask, str):
            all_masks += rle_decode(mask,(768,768))
    return np.expand_dims(all_masks, -1)

def display():

    #newWindow.geometry("300x300")

    button1 = tk. Button (root, text = 'Segment', command = mask_generate). place (x = 100, y = 500)



def mask_generate():

    ImageId = filename. split ("/") [-1]

    img = imread ('test_v2/' + ImageId)
    masks = pd. read_csv ("submission.csv")
    rle_mask = masks. Query ('ImageId=="'+ImageId+'"') ['EncodedPixels']
    mask = masks_as_image(rle_mask)
    lbl = label(mask)
    props = regionprops(lbl)
    img1 = img. copy ()

    for prop in props:
        print ('The coordinates of the ship: ', prop. bbox)
        cv2.rectangle(img1, (prop. bbox [1], prop.   bbox [0]), (prop. bbox [4], prop. bbox[3]), (255, 0, 0), 2)
    fig, (ax1, ax2) = plt. subplots (1, 2, figsize = (15, 5))
    ax1.imshow(img)
    ax1.set_title ('Uploaded Image')

    ax2.set_title ('Image with derived bounding box')
    ax2.imshow(img1)
    plt. show ()

def UploadAction (root = None):
    global filename
    filename = filedialog. Askopenfilename ()
    load = Image.open(filename)
    load = load. resize ((256,256), Image.ANTIALIAS)
    render = ImageTk.PhotoImage (load)
    img2 = Label(image=render)
    img2.image = render
    img2.place(x=100, y=160)
    #print(filename)
    img_path = filename. split ("/") [-2:]
    img_path = img_path [0] + '/' + img_path [1]
    #print(img_path)
    is_ship_there = ship_detect(img_path)
    Label (root, text=is_ship_there).place (x= 100, y = 420)
    If (is_ship_there == "Ship is detected in the picture.  Click on Segment to know where its located"):
display()

root = tk. Tk ()
root. title("Vessel Detection from space borne images")
root.minsize(600, 600)
root.resizable(width=True, height = True)
heading = Label(root, text="Select image").place(x=100, y=100)
button = tk.Button(root, text='Upload image', command=UploadAction).place(x=100, y=125) root.mainloop()
```

4.2 TESTING

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. The increasing visibility of software as a system element and attendant costs associated with a software failure are motivating factors for we planned, through testing. Testing is the process of executing a program with the intent of finding an error. The design of tests for software and other engineered products can be as challenging as the initial design of the product itself.
There of basically two types of testing approaches.

One is Black-Box testing – the specified function that a product has been designed to perform, tests can be conducted that demonstrate each function is fully operated.

The other is White-Box testing – knowing the internal workings of the product, tests can be conducted to ensure that the internal operation of the product performs according to specifications and all internal components have been adequately exercised.

White box and Black box testing methods have been used to test this package. The entire loop constructs have been tested for their boundary and intermediate conditions. The test data was designed with a view to check for all the conditions and logical decisions. Error handling has been taken care of by the use of exception handlers.

4.2.1 TESTING STRATEGIES

Testing is a set of activities that can be planned in advanced and conducted systematically. A strategy for software testing must accommodation low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high-level tests that validate major system functions against customer requirements.

Software testing is one element of verification and validation. Verification refers to the set of activities that ensure that software correctly implements as specific function. Validation refers to a different set of activities that ensure that the software that has been built is traceable to customer requirements.

The main objective of software is testing to uncover errors. To fulfill this objective, a series of test steps unit, integration, validation and system tests are planned and executed. Each test step is accomplished through a series of systematic test technique that assist in the design of test cases. With each testing step, the level of abstraction with which software is considered is broadened.

Testing is the only way to assure the quality of software and it is an umbrella activity rather than a separate phase. This is an activity to be performed in parallel with the software effort and one that consists of its own phases of analysis, design, implementation, execution and maintenance.

UNIT TESTING:

This testing method considers a module as single unit and checks the unit at interfaces and communicates with other modules rather than getting into details at statement level. Here the module will be treated as a black box, which will take some input and generate output. Outputs for a given set of input combination are pre-calculated and are generated by the module.

SYSTEM TESTING:

Here all the pre tested individual modules will be assembled to create the larger system and tests are carried out at system level to make sure that all modules are working in synchronous with each other. This testing methodology helps in making sure that all modules which are running perfectly when checked individually are also running in cohesion with other modules. For this testing we create test cases to check all modules once and then generated test combinations of test paths throughout the system to make sure that no path is making its way into chaos.

INTEGRATED TESTING

Testing is a major quality control measure employed during software development. Its basic function is to detect errors. Sub functions when combined may not produce than it is desired. Global data structures can represent the problems. Integrated testing is a systematic technique for constructing the program structure while conducting the tests. To uncover errors that are associated with interfacing the objective is to make unit test modules and built a program structure that has been detected by design. In a non - incremental integration all the modules are combined in advance and the program is tested as a whole. Here errors will appear in an endless loop function. In incremental testing the program is constructed and tested in small segments where the errors are isolated and corrected.

Different incremental integration strategies are top – down integration, bottom – up integration, regression testing.

REGRESSION TESTING

Each time a new module is added as a part of integration as the software changes. Regression testing is an actually that helps to ensure changes that do not introduce unintended behavior as additional errors.

Regression testing maybe conducted manually by executing a subset of all test cases or using automated capture play back tools enable the software engineer to capture the test case and results for subsequent playback and compression. The regression suit contains different classes of test cases.

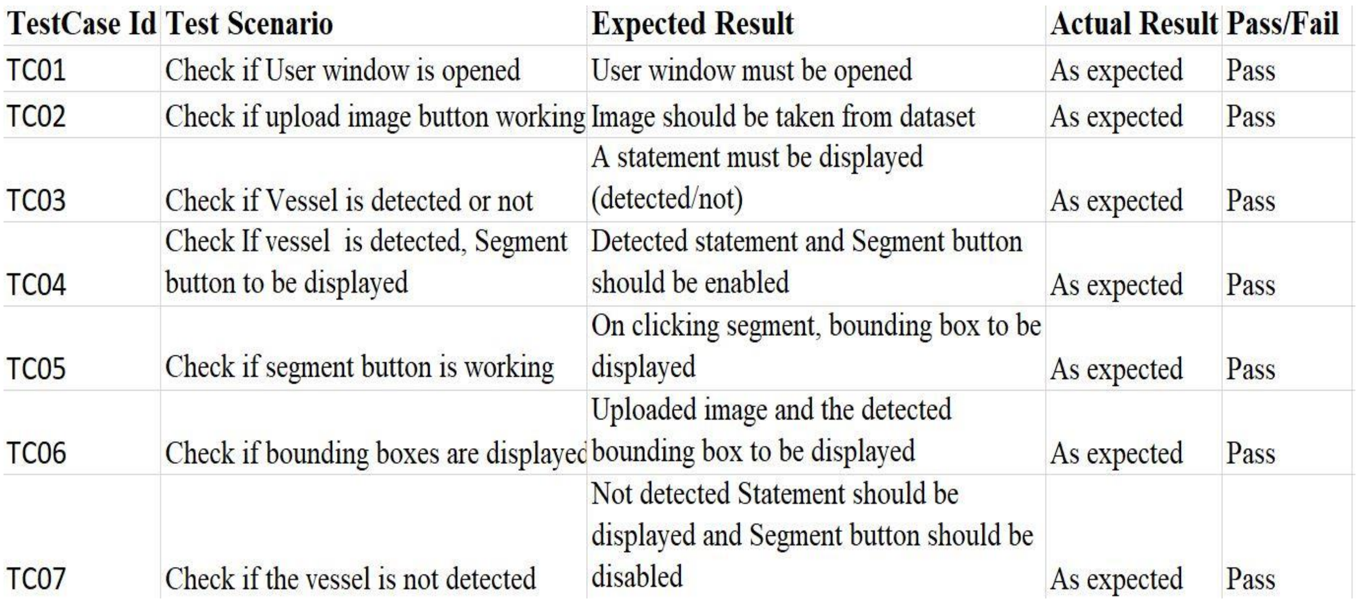
A representative sample to tests that will exercise all software functions.

Additional tests that focus on software functions that are likely to be affected by the change.

4.3 TEST CASES

Integrated and regression testing strategies are used in this application for testing

.



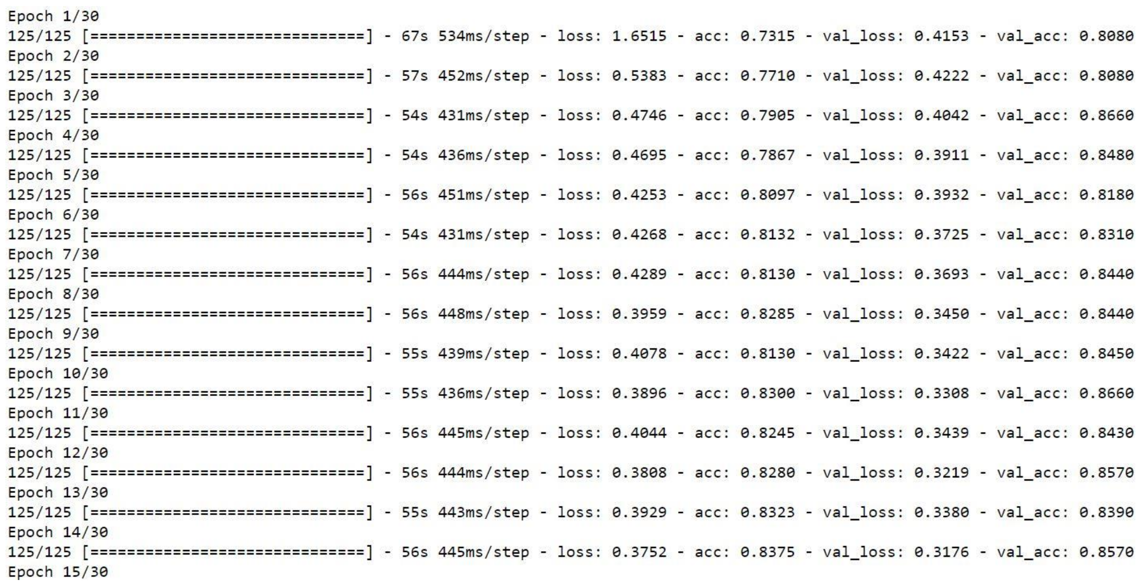
TestCase Id	Test Scenario	Expected Result	Actual Result	Pass/Fail
TC01	Check if User window is opened	User window must be opened	As expected	Pass
TC02	Check if upload image button working	Image should be taken from dataset	As expected	Pass
TC03	Check if Vessel is detected or not	A statement must be displayed (detected/not)	As expected	Pass
TC04	Check If vessel is detected, Segment button to be displayed	Detected statement and Segment button should be enabled	As expected	Pass
TC05	Check if segment button is working	On clicking segment, bounding box to be displayed	As expected	Pass
TC06	Check if bounding boxes are displayec	Uploaded image and the detected bounding box to be displayed	As expected	Pass
TC07	Check if the vessel is not detected	Not detected Statement should be displayed and Segment button should be disabled	As expected	Pass

Fig 4.3.1 Test cases

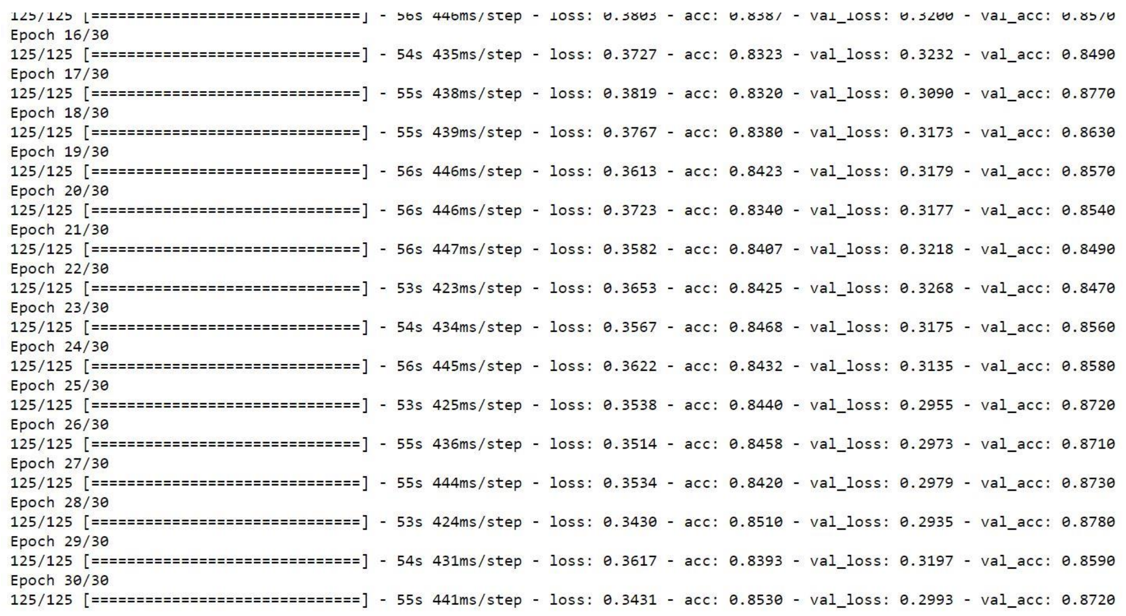
4.4 EXECUTION SCREENSHOTS

OUTPUTS:

Training the ResNet-50 model for 30 epochs. Achieved an accuracy of 83.75%. with 30 epochs and batch size of 32.



```
Epoch 1/30
125/125 [==============================] - 67s 534ms/step - loss: 1.6515 - acc: 0.7315 - val_loss: 0.4153 - val_acc: 0.8080
Epoch 2/30
125/125 [==============================] - 57s 452ms/step - loss: 0.5383 - acc: 0.7710 - val_loss: 0.4222 - val_acc: 0.8080
Epoch 3/30
125/125 [==============================] - 54s 431ms/step - loss: 0.4746 - acc: 0.7905 - val_loss: 0.4042 - val_acc: 0.8660
Epoch 4/30
125/125 [==============================] - 54s 436ms/step - loss: 0.4695 - acc: 0.7867 - val_loss: 0.3911 - val_acc: 0.8480
Epoch 5/30
125/125 [==============================] - 56s 451ms/step - loss: 0.4253 - acc: 0.8097 - val_loss: 0.3932 - val_acc: 0.8180
Epoch 6/30
125/125 [==============================] - 54s 431ms/step - loss: 0.4268 - acc: 0.8132 - val_loss: 0.3725 - val_acc: 0.8310
Epoch 7/30
125/125 [==============================] - 56s 444ms/step - loss: 0.4289 - acc: 0.8130 - val_loss: 0.3693 - val_acc: 0.8440
Epoch 8/30
125/125 [==============================] - 56s 448ms/step - loss: 0.3959 - acc: 0.8285 - val_loss: 0.3450 - val_acc: 0.8440
Epoch 9/30
125/125 [==============================] - 55s 439ms/step - loss: 0.4078 - acc: 0.8130 - val_loss: 0.3422 - val_acc: 0.8450
Epoch 10/30
125/125 [==============================] - 55s 436ms/step - loss: 0.3896 - acc: 0.8300 - val_loss: 0.3308 - val_acc: 0.8660
Epoch 11/30
125/125 [==============================] - 56s 445ms/step - loss: 0.4044 - acc: 0.8245 - val_loss: 0.3439 - val_acc: 0.8430
Epoch 12/30
125/125 [==============================] - 56s 444ms/step - loss: 0.3808 - acc: 0.8280 - val_loss: 0.3219 - val_acc: 0.8570
Epoch 13/30
125/125 [==============================] - 55s 443ms/step - loss: 0.3929 - acc: 0.8323 - val_loss: 0.3380 - val_acc: 0.8390
Epoch 14/30
125/125 [==============================] - 56s 445ms/step - loss: 0.3752 - acc: 0.8375 - val_loss: 0.3176 - val_acc: 0.8570
Epoch 15/30
```



```
125/125 [==============================] - 56s 446ms/step - loss: 0.3803 - acc: 0.8387 - val_loss: 0.3200 - val_acc: 0.8570
Epoch 16/30
125/125 [==============================] - 54s 435ms/step - loss: 0.3727 - acc: 0.8323 - val_loss: 0.3232 - val_acc: 0.8490
Epoch 17/30
125/125 [==============================] - 55s 438ms/step - loss: 0.3819 - acc: 0.8320 - val_loss: 0.3090 - val_acc: 0.8770
Epoch 18/30
125/125 [==============================] - 55s 439ms/step - loss: 0.3767 - acc: 0.8380 - val_loss: 0.3173 - val_acc: 0.8630
Epoch 19/30
125/125 [==============================] - 56s 446ms/step - loss: 0.3613 - acc: 0.8423 - val_loss: 0.3179 - val_acc: 0.8570
Epoch 20/30
125/125 [==============================] - 56s 446ms/step - loss: 0.3723 - acc: 0.8340 - val_loss: 0.3177 - val_acc: 0.8540
Epoch 21/30
125/125 [==============================] - 56s 447ms/step - loss: 0.3582 - acc: 0.8407 - val_loss: 0.3218 - val_acc: 0.8490
Epoch 22/30
125/125 [==============================] - 53s 423ms/step - loss: 0.3653 - acc: 0.8425 - val_loss: 0.3268 - val_acc: 0.8470
Epoch 23/30
125/125 [==============================] - 54s 434ms/step - loss: 0.3567 - acc: 0.8468 - val_loss: 0.3175 - val_acc: 0.8560
Epoch 24/30
125/125 [==============================] - 56s 445ms/step - loss: 0.3622 - acc: 0.8432 - val_loss: 0.3135 - val_acc: 0.8580
Epoch 25/30
125/125 [==============================] - 53s 425ms/step - loss: 0.3538 - acc: 0.8440 - val_loss: 0.2955 - val_acc: 0.8720
Epoch 26/30
125/125 [==============================] - 55s 436ms/step - loss: 0.3514 - acc: 0.8458 - val_loss: 0.2973 - val_acc: 0.8710
Epoch 27/30
125/125 [==============================] - 55s 444ms/step - loss: 0.3534 - acc: 0.8420 - val_loss: 0.2979 - val_acc: 0.8730
Epoch 28/30
125/125 [==============================] - 53s 424ms/step - loss: 0.3430 - acc: 0.8510 - val_loss: 0.2935 - val_acc: 0.8780
Epoch 29/30
125/125 [==============================] - 54s 431ms/step - loss: 0.3617 - acc: 0.8393 - val_loss: 0.3197 - val_acc: 0.8590
Epoch 30/30
125/125 [==============================] - 55s 441ms/step - loss: 0.3431 - acc: 0.8530 - val_loss: 0.2993 - val_acc: 0.8720
```

Fig 4.4.1 Training the model -ResNet 50

Visualization of Training Accuracy vs Validation accuracy for ResNet-50:

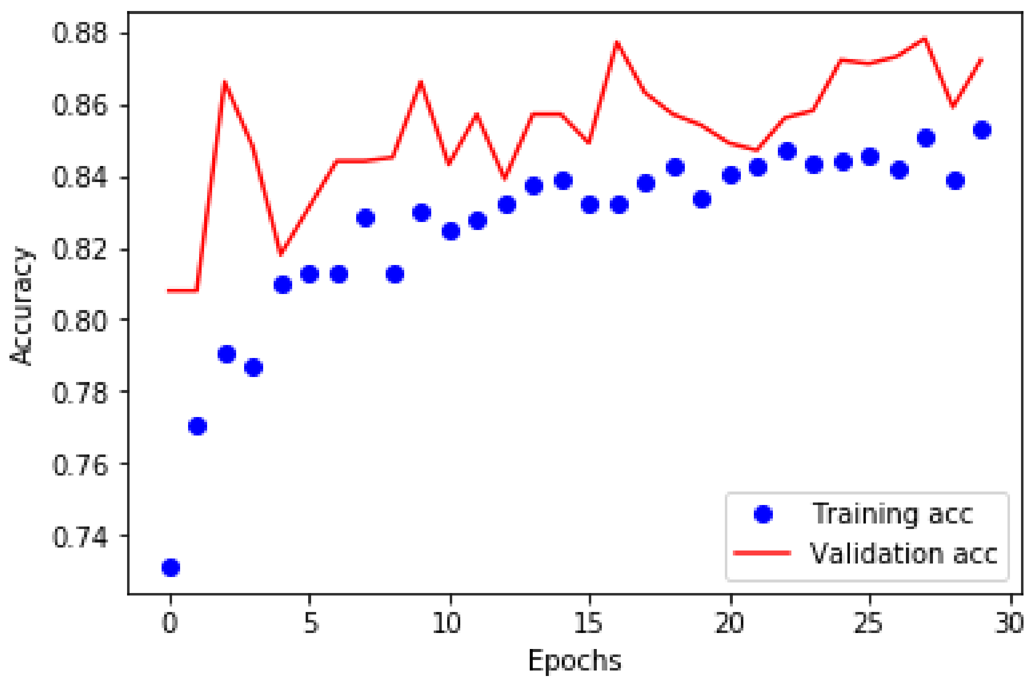
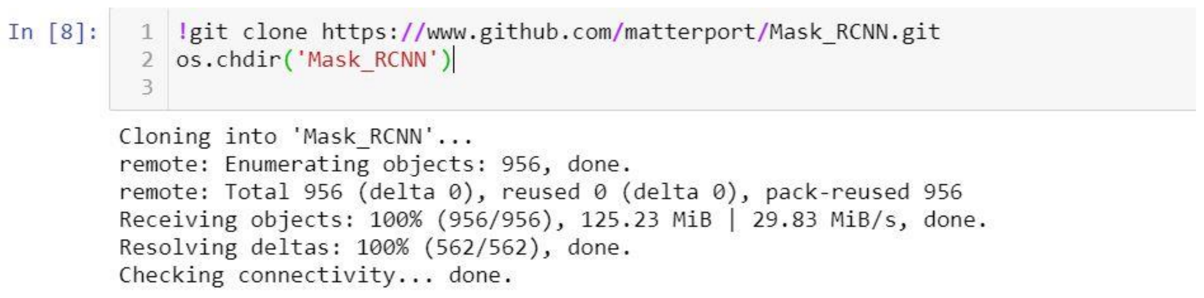


Fig 4.4.2 Loss Visualization

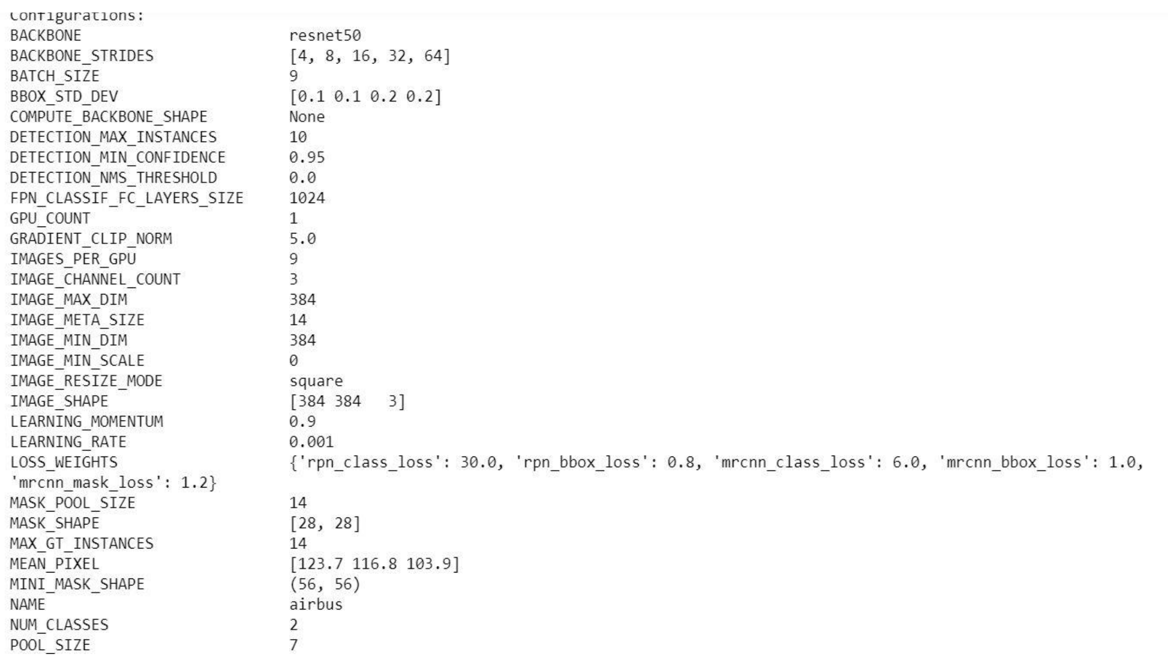
Loading pre-trained MS COCO weights for Segmentation:



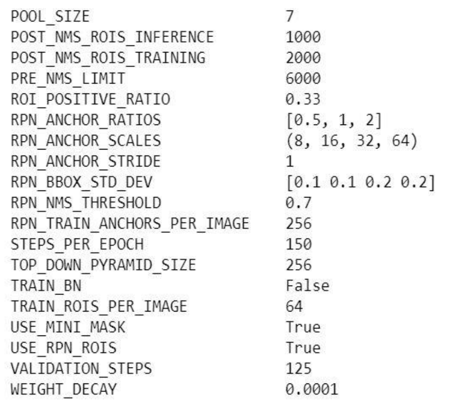
```
In [8]:  !git clone https://www.github.com/matterport/Mask_RCNN.git
         os.chdir('Mask_RCNN')

Cloning into 'Mask_RCNN'...
remote: Enumerating objects: 956, done.
remote: Total 956 (delta 0), reused 0 (delta 0), pack-reused 956
Receiving objects: 100% (956/956), 125.23 MiB | 29.83 MiB/s, done.
Resolving deltas: 100% (562/562), done.
Checking connectivity... done.
```

Fig 4.4.3 Loading MS COCO weights

Mask RCNN- Configuration:



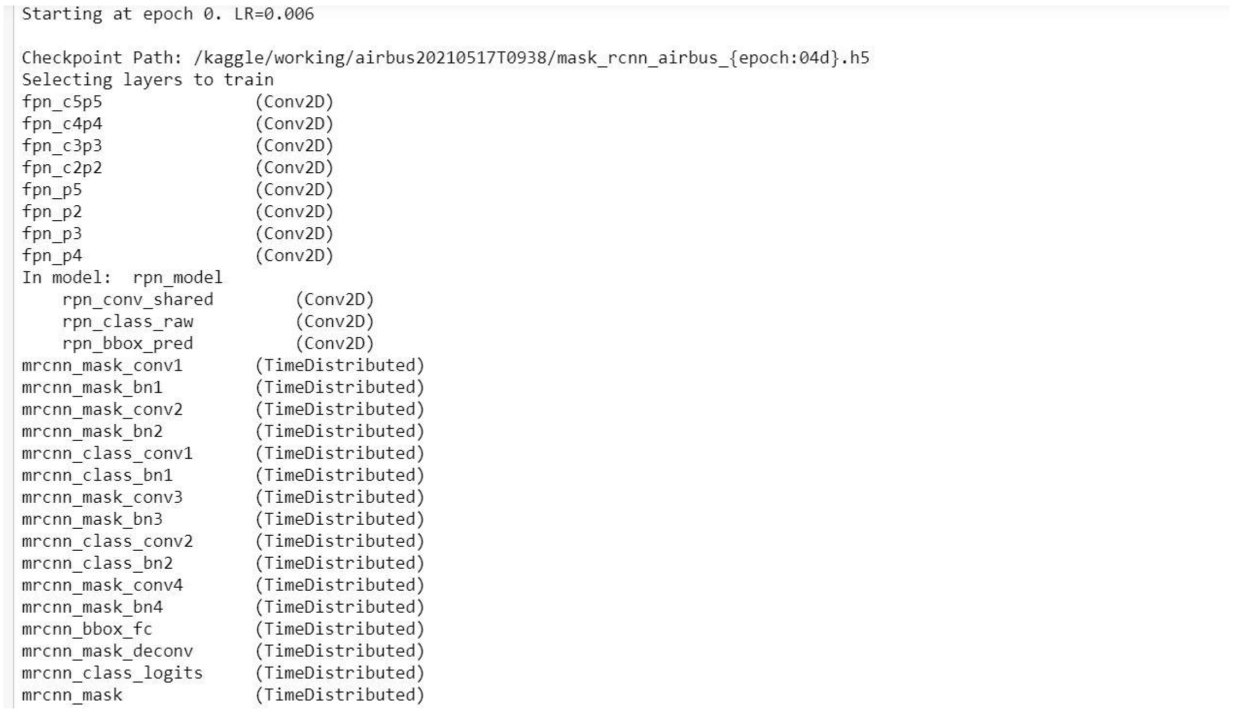
```
Configurations:
BACKBONE                       resnet50
BACKBONE_STRIDES               [4, 8, 16, 32, 64]
BATCH_SIZE                     9
BBOX_STD_DEV                   [0.1 0.1 0.2 0.2]
COMPUTE_BACKBONE_SHAPE         None
DETECTION_MAX_INSTANCES        10
DETECTION_MIN_CONFIDENCE       0.95
DETECTION_NMS_THRESHOLD        0.0
FPN_CLASSIF_FC_LAYERS_SIZE     1024
GPU_COUNT                      1
GRADIENT_CLIP_NORM             5.0
IMAGES_PER_GPU                 9
IMAGE_CHANNEL_COUNT            3
IMAGE_MAX_DIM                  384
IMAGE_META_SIZE                14
IMAGE_MIN_DIM                  384
IMAGE_MIN_SCALE                0
IMAGE_RESIZE_MODE              square
IMAGE_SHAPE                    [384 384   3]
LEARNING_MOMENTUM              0.9
LEARNING_RATE                  0.001
LOSS_WEIGHTS                   {'rpn_class_loss': 30.0, 'rpn_bbox_loss': 0.8, 'mrcnn_class_loss': 6.0, 'mrcnn_bbox_loss': 1.0,
'mrcnn_mask_loss': 1.2}
MASK_POOL_SIZE                 14
MASK_SHAPE                     [28, 28]
MAX_GT_INSTANCES               14
MEAN_PIXEL                     [123.7 116.8 103.9]
MINI_MASK_SHAPE                (56, 56)
NAME                           airbus
NUM_CLASSES                    2
POOL_SIZE                      7
```



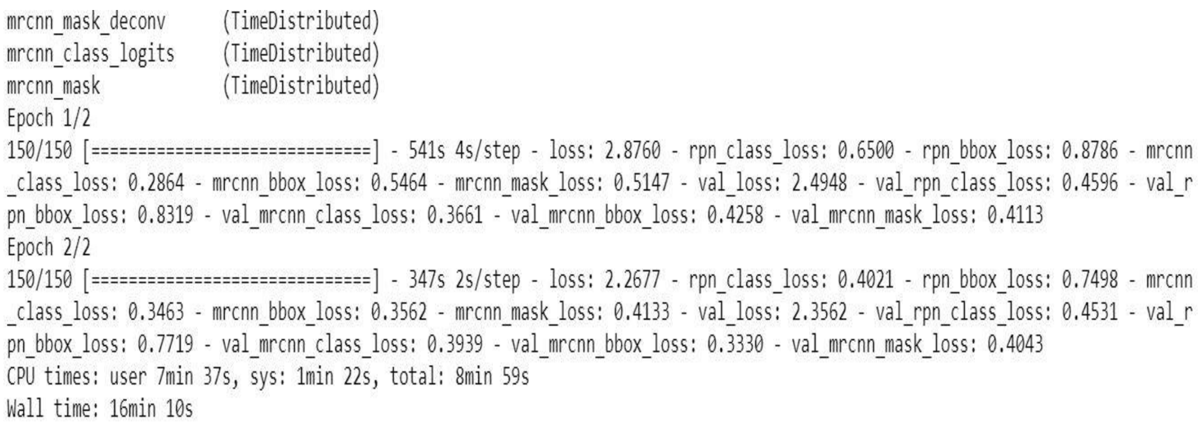
```
POOL_SIZE                      7
POST_NMS_ROIS_INFERENCE        1000
POST_NMS_ROIS_TRAINING         2000
PRE_NMS_LIMIT                  6000
ROI_POSITIVE_RATIO             0.33
RPN_ANCHOR_RATIOS              [0.5, 1, 2]
RPN_ANCHOR_SCALES              (8, 16, 32, 64)
RPN_ANCHOR_STRIDE              1
RPN_BBOX_STD_DEV               [0.1 0.1 0.2 0.2]
RPN_NMS_THRESHOLD              0.7
RPN_TRAIN_ANCHORS_PER_IMAGE    256
STEPS_PER_EPOCH                150
TOP_DOWN_PYRAMID_SIZE          256
TRAIN_BN                       False
TRAIN_ROIS_PER_IMAGE           64
USE_MINI_MASK                  True
USE_RPN_ROIS                   True
VALIDATION_STEPS               125
WEIGHT_DECAY                   0.0001
```

Fig 4.4.4 Mask RCNN configuration

Training Mask RCNN:



```
Starting at epoch 0. LR=0.006

Checkpoint Path: /kaggle/working/airbus20210517T0938/mask_rcnn_airbus_{epoch:04d}.h5
Selecting layers to train
fpn_c5p5               (Conv2D)
fpn_c4p4               (Conv2D)
fpn_c3p3               (Conv2D)
fpn_c2p2               (Conv2D)
fpn_p5                 (Conv2D)
fpn_p2                 (Conv2D)
fpn_p3                 (Conv2D)
fpn_p4                 (Conv2D)
In model:  rpn_model
    rpn_conv_shared        (Conv2D)
    rpn_class_raw          (Conv2D)
    rpn_bbox_pred          (Conv2D)
mrcnn_mask_conv1       (TimeDistributed)
mrcnn_mask_bn1         (TimeDistributed)
mrcnn_mask_conv2       (TimeDistributed)
mrcnn_mask_bn2         (TimeDistributed)
mrcnn_class_conv1      (TimeDistributed)
mrcnn_class_bn1        (TimeDistributed)
mrcnn_mask_conv3       (TimeDistributed)
mrcnn_mask_bn3         (TimeDistributed)
mrcnn_class_conv2      (TimeDistributed)
mrcnn_class_bn2        (TimeDistributed)
mrcnn_mask_conv4       (TimeDistributed)
mrcnn_mask_bn4         (TimeDistributed)
mrcnn_bbox_fc          (TimeDistributed)
mrcnn_mask_deconv      (TimeDistributed)
mrcnn_class_logits     (TimeDistributed)
mrcnn_mask             (TimeDistributed)
```



```
mrcnn_mask_deconv      (TimeDistributed)
mrcnn_class_logits     (TimeDistributed)
mrcnn_mask             (TimeDistributed)
Epoch 1/2
150/150 [==============================] - 541s 4s/step - loss: 2.8760 - rpn_class_loss: 0.6500 - rpn_bbox_loss: 0.8786 - mrcnn
_class_loss: 0.2864 - mrcnn_bbox_loss: 0.5464 - mrcnn_mask_loss: 0.5147 - val_loss: 2.4948 - val_rpn_class_loss: 0.4596 - val_r
pn_bbox_loss: 0.8319 - val_mrcnn_class_loss: 0.3661 - val_mrcnn_bbox_loss: 0.4258 - val_mrcnn_mask_loss: 0.4113
Epoch 2/2
150/150 [==============================] - 347s 2s/step - loss: 2.2677 - rpn_class_loss: 0.4021 - rpn_bbox_loss: 0.7498 - mrcnn
_class_loss: 0.3463 - mrcnn_bbox_loss: 0.3562 - mrcnn_mask_loss: 0.4133 - val_loss: 2.3562 - val_rpn_class_loss: 0.4531 - val_r
pn_bbox_loss: 0.7719 - val_mrcnn_class_loss: 0.3939 - val_mrcnn_bbox_loss: 0.3330 - val_mrcnn_mask_loss: 0.4043
CPU times: user 7min 37s, sys: 1min 22s, total: 8min 59s
Wall time: 16min 10s
```

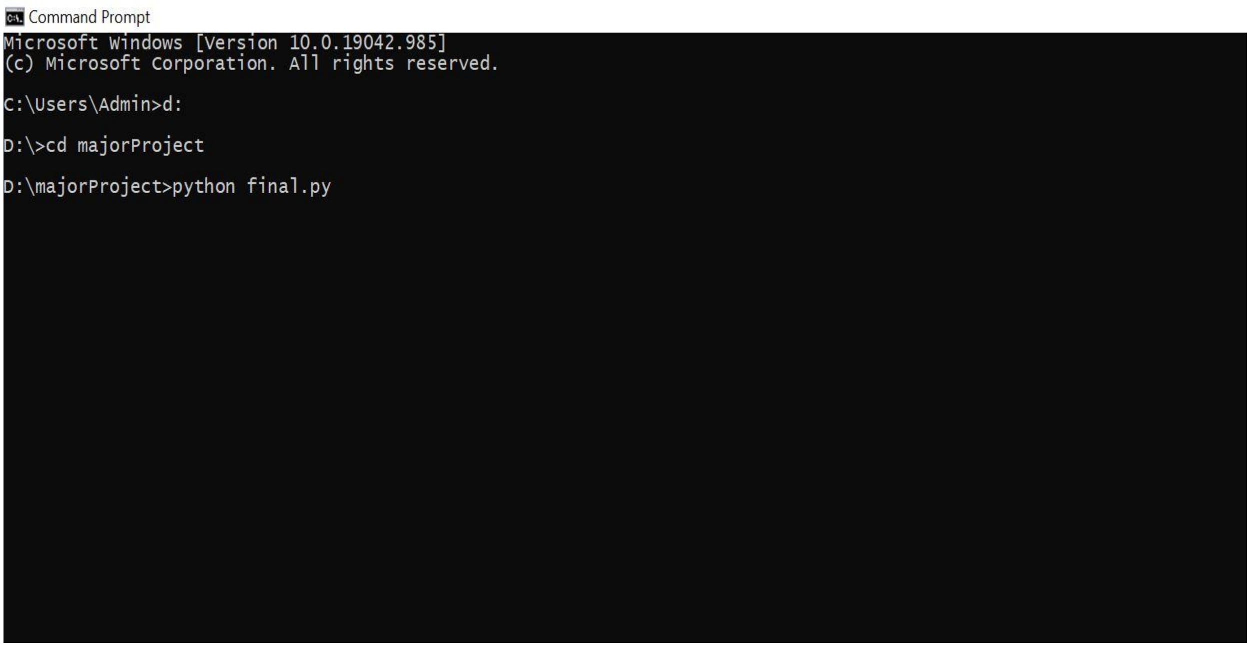
Fig 4.4.5 Training Mask RCNN

Test data CSV file (Sample):

	ImageId	EncodedPixels
0	4291f3a66.jpg	NaN
1	b0808caaf.jpg	194759 8 195523 17 196289 21 197055 24 197822 ...
2	9f582d5ce.jpg	NaN
3	916ae8dd3.jpg	NaN
4	6fa533973.jpg	NaN
5	e0cf877a8.jpg	NaN
6	5e5085535.jpg	NaN
7	51ffc3d65.jpg	NaN
8	6dce1205c.jpg	NaN
9	b0c08f452.jpg	NaN
10	08229754b.jpg	334530 28 335288 40 336048 49 336811 54 337571...
11	36cc4738f.jpg	233558 6 234326 7 235093 10 235861 11 236629 1...
12	7c209591f.jpg	NaN

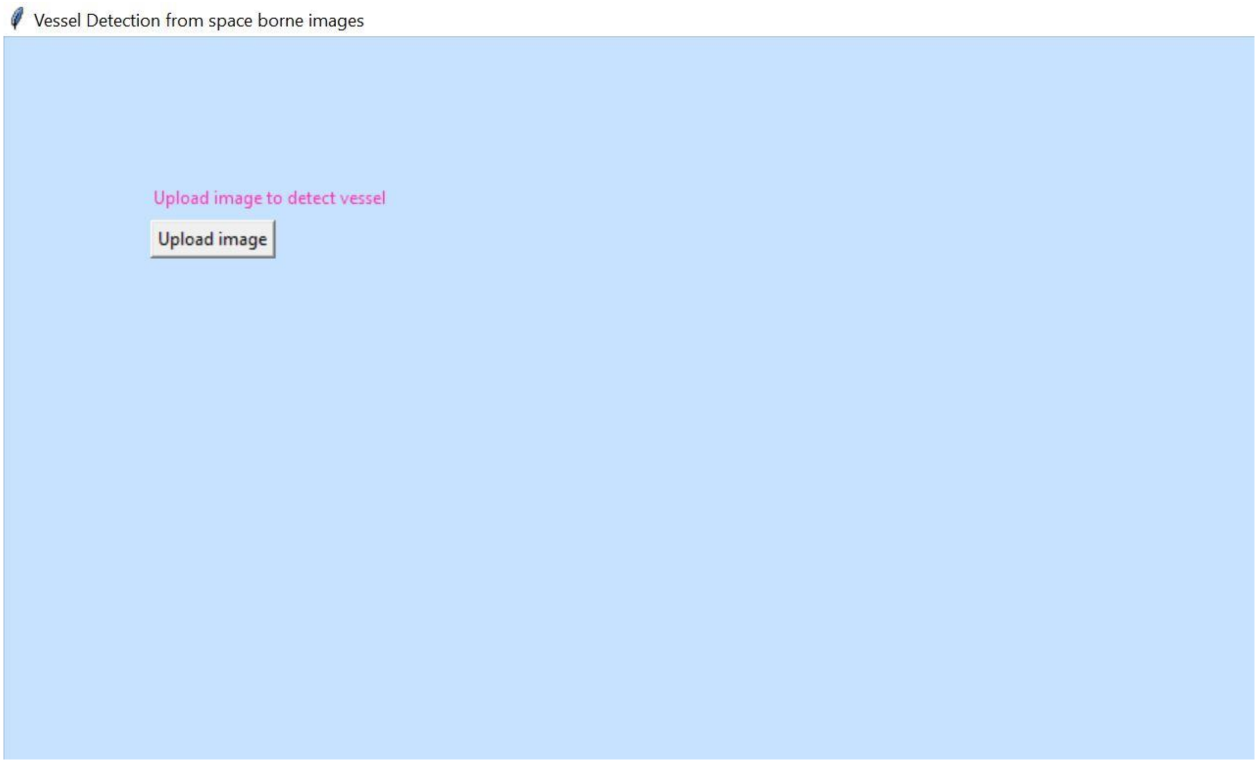
Fig 4.4.6 Test data Sample CSV

Final Execution:



```
Command Prompt
Microsoft Windows [Version 10.0.19042.985]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>d:

D:\>cd majorProject

D:\majorProject>python final.py
```

Fig 4.4.7 Execution Page



Vessel Detection from space borne images
Upload image to detect vessel
Upload image

Fig 4.4.8 Page for Image Selection

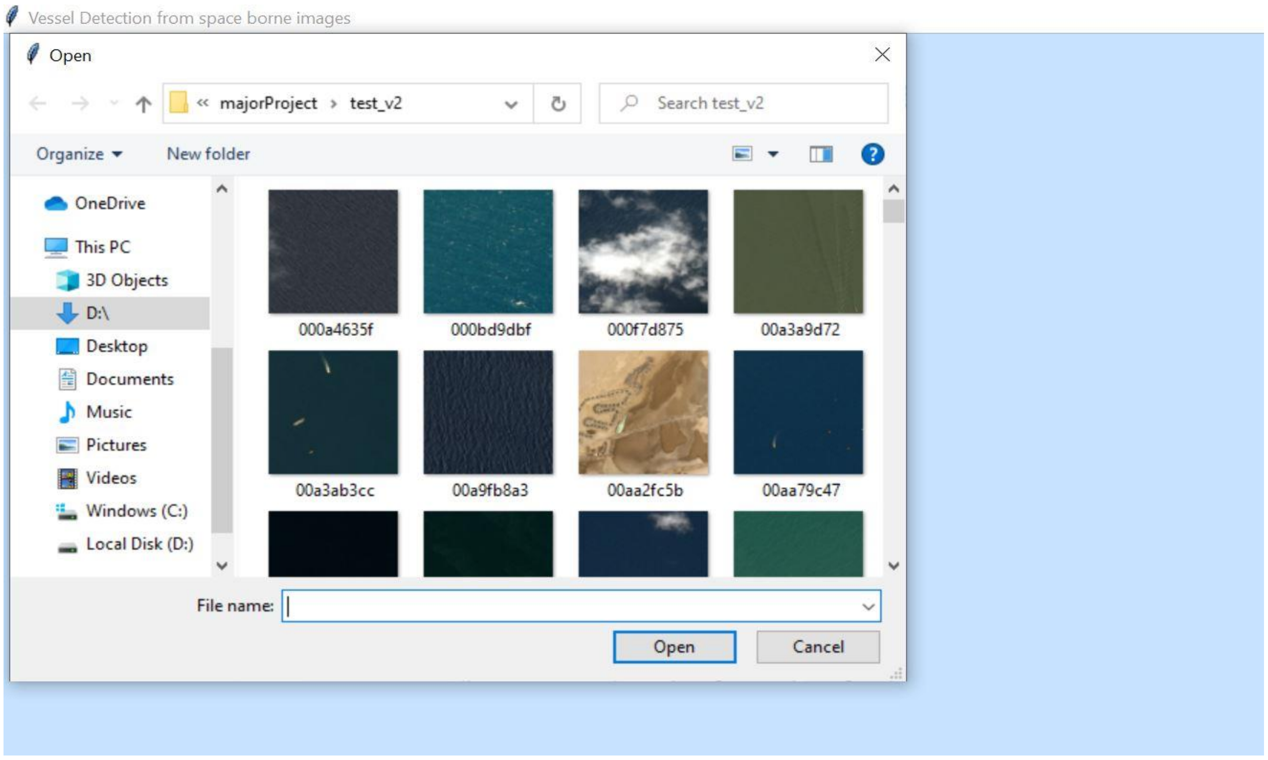
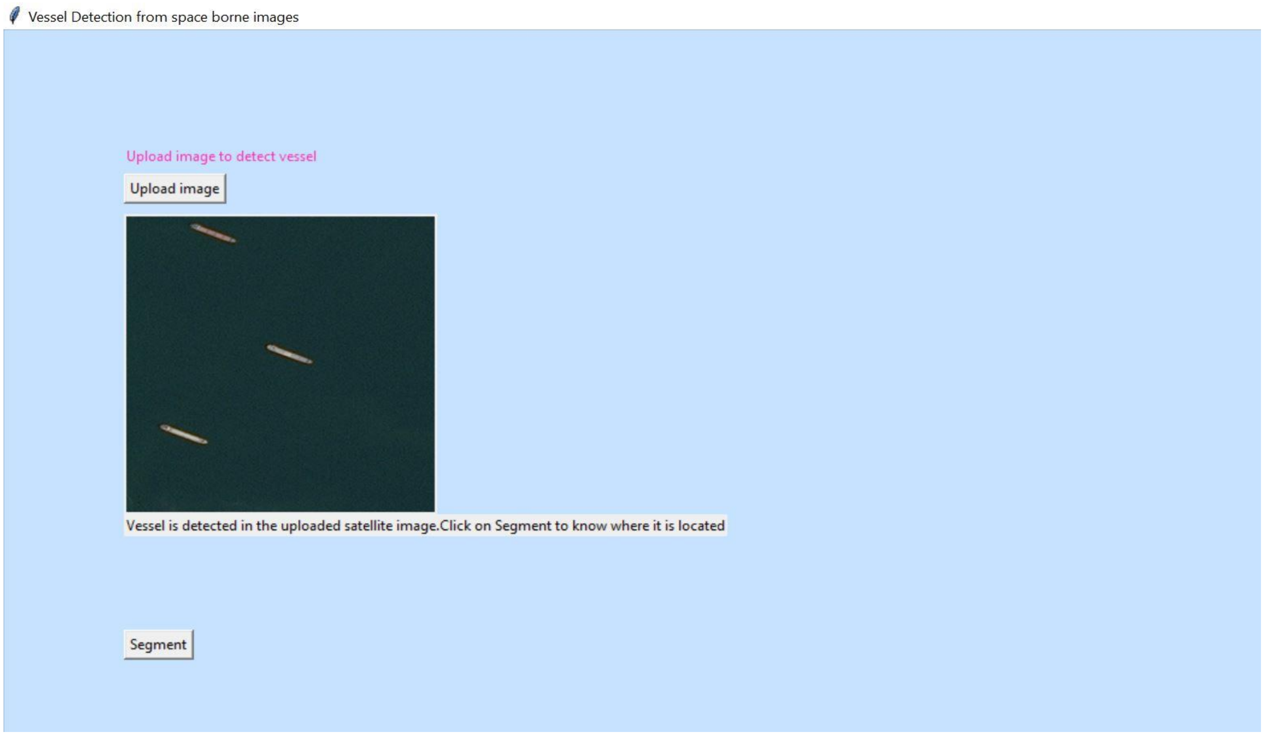


Fig 4.4.9 User browses and uploads Image

Execution for Vessel detected Case:


Vessel Detection from space borne images
Upload image to detect vessel
Upload image
Segment

Vessel is detected in the uploaded satellite image.Click on Segment to know where it is located

Fig 4.4.10 Vessel detected case

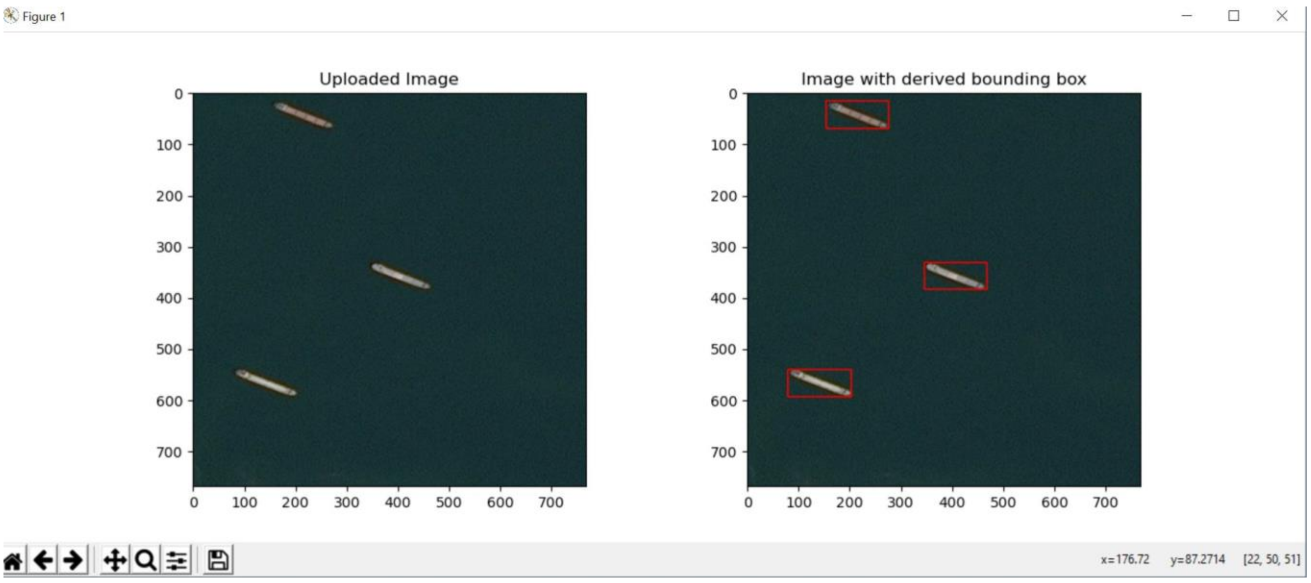
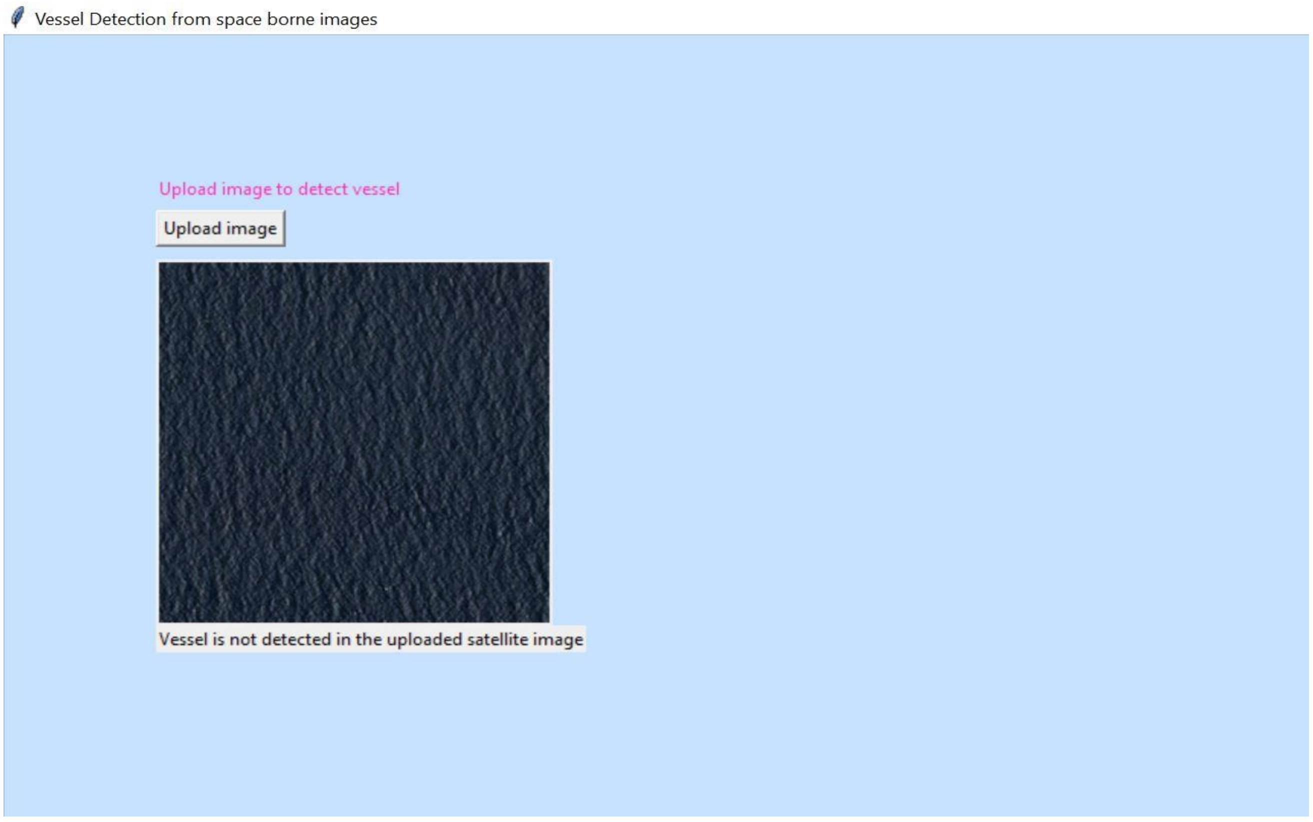


Figure 1
Uploaded Image
Image with derived bounding box
0
100
200
300
400
500
600
700
x=176.72 y=87.2714 [22, 50, 51]

Fig 4.4.11 Vessel Segmentation

Execution for No Vessel detected Case:


Vessel Detection from space borne images
Upload image to detect vessel
Upload image

Vessel is not detected in the uploaded satellite image

Fig 4.4.12 No vessel detected case

CONCLUSION AND FUTURE SCOPE

Thousands of vessels are constantly sailing in sea, oceans etc. These vessels are used for Global Positioning System (GPS) applications, vessel detection, communications and forecasting weather etc. Vessel detection from satellite (space borne) images is an important application for maritime applications like ship traffic surveillance, protection against illegal fisheries, oil discharge control and sea pollution monitoring. This is currently manually being done through the use of an Automated Identification System (AIS), which uses Very High Frequency (VHF) radio frequencies. AIS are very effective at monitoring ships which are legally required to install a VHF transponder, but fail to detect those which are not. Satellite imagery can be used in these cases as it removes the human element out of the loop.

This project uses satellite images to detect vessels and segment them. Resnet-50, a residual neural network along with Mask- RCNN, a deep neural network that is aimed to solve instance segmentation problem is used for effectively detecting and segmenting ships. Resnet -50 is the transfer-learning network which is very efficient than basic neural networks and therefore this project detects if a vessel is there in the image with better accuracy and if the vessel is detected, it also segments the vessel and encloses it with a bounding-box.

This application can be of great help to the maritime security and offshore operations in the surveillance, energy and military sector. It could bring a whole new dimension of transport for container ships and vessels by tracking ships from satellite images in real time. It can also be extended to navy for safety-monitoring, ship-tracking etc.

The further enhancements of this application are:

(i) Increasing the training data with wide variety of satellite images with islands that resemble the shape of vessel, images with many vessels, ports and whales for much accurate detection.

REFERENCES

- Real time detection and segmentation of ships
  http://www.ijstr.org/final-print/dec2019/Real-Time-Detection-And-Segmentation-Of-Ships-In-Satellite-Images.pdf
- An intelligent ship detection using CNN
  https://www.hindawi.com/journals/complexity/2020/1520872/
- Ship detection using high resolution satellite imagery
  https://earth.esa.int/eogateway/documents/20142/37627/8-Ship-Detection-High-Res-Satellite-Imagery.pdf
- Ship Detection and Classification on Optical Remote Sensing Images Using Deep Learning
  https://www.researchgate.net/publication/319485878_Ship_Detection_and_Classification_on_Optical_Remote_Sensing_Images_Using_Deep_Learning
- Airbus Ship - Traditional vs. Convolutional Neural Network Approach
  http://cs229.stanford.edu/proj2018/report/58.pdf
- Ship detection based on deep learning
  https://ieeexplore.ieee.org/abstract/document/8816265